

---

**rdflib**

***Release 6.3.2***

**unknown**

**Aug 01, 2023**



## CONTENTS

<b>1</b>	<b>Getting started</b>	<b>3</b>
<b>2</b>	<b>In depth</b>	<b>27</b>
<b>3</b>	<b>Reference</b>	<b>47</b>
<b>4</b>	<b>For developers</b>	<b>617</b>
<b>5</b>	<b>Source Code</b>	<b>633</b>
<b>6</b>	<b>Further help &amp; Contact</b>	<b>635</b>
	<b>Bibliography</b>	<b>637</b>
	<b>Python Module Index</b>	<b>639</b>
	<b>Index</b>	<b>641</b>



RDFLib is a pure Python package for working with [RDF](#). It contains:

- **Parsers & Serializers**
  - for RDF/XML, N3, NTriples, N-Quads, Turtle, TriX, JSON-LD, HexTuples, RDFa and Microdata
- **Store implementations**
  - memory stores
  - persistent, on-disk stores, using databases such as BerkeleyDB
  - remote SPARQL endpoints
- **Graph interface**
  - to a single graph
  - or to multiple Named Graphs within a dataset
- **SPARQL 1.1 implementation**
  - both Queries and Updates are supported

**Caution:** RDFLib is designed to access arbitrary network and file resources, in some cases these are directly requested resources, in other cases they are indirectly referenced resources.

If you are using RDFLib to process untrusted documents or queries you should take measures to restrict file and network access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.



## GETTING STARTED

If you have never used RDFLib, the following will help get you started:

### 1.1 Getting started with RDFLib

#### 1.1.1 Installation

RDFLib is open source and is maintained in a [GitHub](#) repository. RDFLib releases, current and previous, are listed on [PyPi](#)

The best way to install RDFLib is to use `pip` (sudo as required):

```
$ pip install rdflib
```

If you want the latest code to run, clone the main branch of the GitHub repo and use that or you can `pip install` directly from GitHub:

```
$ pip install git+https://github.com/RDFLib/rdflib.git@main#egg=rdflib
```

#### 1.1.2 Support

Usage support is available via questions tagged with `[rdflib]` on [StackOverflow](#) and development support, notifications and detailed discussion through the `rdflib-dev` group (mailing list):

<http://groups.google.com/group/rdflib-dev>

If you notice an bug or want to request an enhancement, please do so via our Issue Tracker in Github:

<http://github.com/RDFLib/rdflib/issues>

#### 1.1.3 How it all works

*The package uses various Python idioms that offer an appropriate way to introduce RDF to a Python programmer who hasn't worked with RDF before.*

The primary interface that RDFLib exposes for working with RDF is a [Graph](#).

RDFLib graphs are un-sorted containers; they have ordinary Python set operations (e.g. `add()` to add a triple) plus methods that search triples and return them in arbitrary order.

RDFLib graphs also redefine certain built-in Python methods in order to behave in a predictable way. They do this by [emulating container types](#) and are best thought of as a set of 3-item tuples (“triples”, in RDF-speak):

```
[
    (subject0, predicate0, object0),
    (subject1, predicate1, object1),
    ...
    (subjectN, predicateN, objectN)
]
```

### 1.1.4 A tiny example

```
from rdflib import Graph

# Create a Graph
g = Graph()

# Parse in an RDF file hosted on the Internet
g.parse("http://www.w3.org/People/Berners-Lee/card")

# Loop through each triple in the graph (subj, pred, obj)
for subj, pred, obj in g:
    # Check if there is at least one triple in the Graph
    if (subj, pred, obj) not in g:
        raise Exception("It better be!")

# Print the number of "triples" in the Graph
print(f"Graph g has {len(g)} statements.")
# Prints: Graph g has 86 statements.

# Print out the entire Graph in the RDF Turtle format
print(g.serialize(format="turtle"))
```

Here a *Graph* is created and then an RDF file online, Tim Berners-Lee's social network details, is parsed into that graph. The `print()` statement uses the `len()` function to count the number of triples in the graph.

### 1.1.5 A more extensive example

```
from rdflib import Graph, Literal, RDF, URIRef
# rdflib knows about quite a few popular namespaces, like W3C ontologies, schema.org etc.
from rdflib.namespace import FOAF, XSD

# Create a Graph
g = Graph()

# Create an RDF URI node to use as the subject for multiple triples
donna = URIRef("http://example.org/donna")

# Add triples using store's add() method.
g.add((donna, RDF.type, FOAF.Person))
g.add((donna, FOAF.nick, Literal("donna", lang="en")))
g.add((donna, FOAF.name, Literal("Donna Fales")))
g.add((donna, FOAF.mbox, URIRef("mailto:donna@example.org")))
```

(continues on next page)



(continued from previous page)

```

# Add another person
ed = URIRef("http://example.org/edward")

# Add triples using store's add() method.
g.add((ed, RDF.type, FOAF.Person))
g.add((ed, FOAF.nick, Literal("ed", datatype=XSD.string)))
g.add((ed, FOAF.name, Literal("Edward Scissorhands")))
g.add((ed, FOAF.mbox, Literal("e.scissorhands@example.org", datatype=XSD.anyURI)))

# Iterate over triples in store and print them out.
print("--- printing raw triples ---")
for s, p, o in g:
    print((s, p, o))

# For each foaf:Person in the store, print out their mbox property's value.
print("--- printing mboxses ---")
for person in g.subjects(RDF.type, FOAF.Person):
    for mbox in g.objects(person, FOAF.mbox):
        print(mbox)

# Bind the FOAF namespace to a prefix for more readable output
g.bind("foaf", FOAF)

# print all the data in the Notation3 format
print("--- printing mboxses ---")
print(g.serialize(format='n3'))

```

### 1.1.6 A SPARQL query example

```

from rdflib import Graph

# Create a Graph, parse in Internet data
g = Graph().parse("http://www.w3.org/People/Berners-Lee/card")

# Query the data in g using SPARQL
# This query returns the 'name' of all ``foaf:Person`` instances
q = """
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>

    SELECT ?name
    WHERE {
        ?p rdf:type foaf:Person .

        ?p foaf:name ?name .
    }
    """

# Apply the query to the graph and iterate through results
for r in g.query(q):

```

(continues on next page)

(continued from previous page)

```
print(r["name"])

# prints: Timothy Berners-Lee
```

### 1.1.7 More examples

There are many more *examples* in the `examples` folder in the source distribution.

## 1.2 Loading and saving RDF

### 1.2.1 Reading RDF files

RDF data can be represented using various syntaxes (`turtle`, `rd/xml`, `n3`, `n-triples`, `trix`, `JSON-LD`, etc.). The simplest format is `ntriples`, which is a triple-per-line format.

Create the file `demo.nt` in the current directory with these two lines in it:

```
<http://example.com/drewp> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://
↳xmlns.com/foaf/0.1/Person> .
<http://example.com/drewp> <http://example.com/says> "Hello World" .
```

On line 1 this file says “drewp is a FOAF Person”. On line 2 it says “drep says “Hello World””.

RDFLib can guess what format the file is by the file ending (“`.nt`” is commonly used for `n-triples`) so you can just use `parse()` to read in the file. If the file had a non-standard RDF file ending, you could set the keyword-parameter `format` to specify either an Internet Media Type or the format name (a *list of available parsers* is available).

In an interactive python interpreter, try this:

```
from rdflib import Graph

g = Graph()
g.parse("demo.nt")

print(len(g))
# prints: 2

import pprint
for stmt in g:
    pprint.pprint(stmt)
# prints:
# (rdflib.term.URIRef('http://example.com/drewp'),
#  rdflib.term.URIRef('http://example.com/says'),
#  rdflib.term.Literal('Hello World'))
# (rdflib.term.URIRef('http://example.com/drewp'),
#  rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#type'),
#  rdflib.term.URIRef('http://xmlns.com/foaf/0.1/Person'))
```

The final lines show how RDFLib represents the two statements in the file: the statements themselves are just length-3 tuples (“triples”) and the subjects, predicates, and objects of the triples are all `rdflib` types.

## 1.2.2 Reading remote RDF

Reading graphs from the Internet is easy:

```
from rdflib import Graph

g = Graph()
g.parse("http://www.w3.org/People/Berners-Lee/card")
print(len(g))
# prints: 86
```

`rdflib.Graph.parse()` can process local files, remote data via a URL, as in this example, or RDF data in a string (using the `data` parameter).

## 1.2.3 Saving RDF

To store a graph in a file, use the `rdflib.Graph.serialize()` function:

```
from rdflib import Graph

g = Graph()
g.parse("http://www.w3.org/People/Berners-Lee/card")
g.serialize(destination="tbl.ttl")
```

This parses data from <http://www.w3.org/People/Berners-Lee/card> and stores it in a file `tbl.ttl` in this directory using the turtle format, which is the default RDF serialization (as of rdflib 6.0.0).

To read the same data and to save it as an RDF/XML format string in the variable `v`, do this:

```
from rdflib import Graph

g = Graph()
g.parse("http://www.w3.org/People/Berners-Lee/card")
v = g.serialize(format="xml")
```

The following table lists the RDF formats you can serialize data to with rdflib, out of the box, and the `format=KEYWORD` keyword used to reference them within `serialize()`:

RDF Format	Keyword	Notes
Turtle	turtle, ttl or turtle2	turtle2 is just turtle with more spacing & linebreaks
RDF/XML	xml or pretty-xml	Was the default format, rdflib < 6.0.0
JSON-LD	json-ld	There are further options for compact syntax and other JSON-LD variants
N-Triples	ntriples, nt or nt11	nt11 is exactly like nt, only utf8 encoded
Notation-3	n3	N3 is a superset of Turtle that also caters for rules and a few other things
Trig	trig	Turtle-like format for RDF triples + context (RDF quads) and thus multiple graphs
Trix	trix	RDF/XML-like format for RDF quads
N-Quads	nquads	N-Triples-like format for RDF quads

## 1.2.4 Working with multi-graphs

To read and query multi-graphs, that is RDF data that is context-aware, you need to use rdflib's *rdflib.ConjunctiveGraph* or *rdflib.Dataset* class. These are extensions to *rdflib.Graph* that know all about quads (triples + graph IDs).

If you had this multi-graph data file (in the `trig` format, using new-style PREFIX statement (not the older `@prefix`):

```
PREFIX eg: <http://example.com/person/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

eg:graph-1 {
    eg:drewp a foaf:Person .
    eg:drewp eg:says "Hello World" .
}

eg:graph-2 {
    eg:nick a foaf:Person .
    eg:nick eg:says "Hi World" .
}
```

You could parse the file and query it like this:

```
from rdflib import Dataset
from rdflib.namespace import RDF

g = Dataset()
g.parse("demo.trig")

for s, p, o, g in g.quads((None, RDF.type, None, None)):
    print(s, g)
```

This will print out:

```
http://example.com/person/drewp http://example.com/person/graph-1
http://example.com/person/nick http://example.com/person/graph-2
```

## 1.3 Creating RDF triples

### 1.3.1 Creating Nodes

RDF data is a graph where the nodes are URI references, Blank Nodes or Literals. In RDFS, these node types are represented by the classes *URIRef*, *BNode*, and *Literal*. URIRefs and BNodes can both be thought of as resources, such as a person, a company, a website, etc.

- A *BNode* is a node where the exact URI is not known - usually a node with identity only in relation to other nodes.
- A *URIRef* is a node where the exact URI is known. In addition to representing some subjects and predicates in RDF graphs, URIRefs are always used to represent properties/predicates
- *Literals* represent object values, such as a name, a date, a number, etc. The most common literal values are XML data types, e.g. string, int... but custom types can be declared too

Nodes can be created by the constructors of the node classes:

```

from rdflib import URIRef, BNode, Literal

bob = URIRef("http://example.org/people/Bob")
linda = BNode() # a GUID is generated

name = Literal("Bob") # passing a string
age = Literal(24) # passing a python int
height = Literal(76.5) # passing a python float

```

Literals can be created from Python objects, this creates data-typed literals. For the details on the mapping see *Literals*.

For creating many URIRefs in the same namespace, i.e. URIs with the same prefix, RDFLib has the `rdflib.namespace.Namespace` class

```

from rdflib import Namespace

n = Namespace("http://example.org/people/")

n.bob # == rdflib.term.URIRef("http://example.org/people/bob")
n.eve # == rdflib.term.URIRef("http://example.org/people/eve")

```

This is very useful for schemas where all properties and classes have the same URI prefix. RDFLib defines Namespaces for some common RDF/OWL schemas, including most W3C ones:

```

from rdflib.namespace import CSVW, DC, DCAT, DCTERMS, DOAP, FOAF, ODRL2, ORG, OWL, \
    PROF, PROV, RDF, RDFS, SDO, SH, SKOS, SOSA, SSN, TIME, \
    VOID, XMLNS, XSD

RDF.type
# == rdflib.term.URIRef("http://www.w3.org/1999/02/22-rdf-syntax-ns#type")

FOAF.knows
# == rdflib.term.URIRef("http://xmlns.com/foaf/0.1/knows")

PROF.isProfileOf
# == rdflib.term.URIRef("http://www.w3.org/ns/dx/prof/isProfileOf")

SOSA.Sensor
# == rdflib.term.URIRef("http://www.w3.org/ns/sosa/Sensor")

```

### 1.3.2 Adding Triples to a graph

We already saw in *Loading and saving RDF*, how triples can be added from files and online locations with the `parse()` function.

Triples can also be added within Python code directly, using the `add()` function:

Graph.`add(triple)`

Add a triple with self as context

#### Parameters

- `self` (`TypeVar(_GraphT, bound= Graph)`) –

- **triple** (Tuple[Node, Node, Node]) –

**Return type**

TypeVar(\_GraphT, bound= Graph)

`add()` takes a 3-tuple (a “triple”) of RDFLib nodes. Using the nodes and namespaces we defined previously:

```
from rdflib import Graph, URIRef, Literal, BNode
from rdflib.namespace import FOAF, RDF

g = Graph()
g.bind("foaf", FOAF)

bob = URIRef("http://example.org/people/Bob")
linda = BNode() # a GUID is generated

name = Literal("Bob")
age = Literal(24)

g.add((bob, RDF.type, FOAF.Person))
g.add((bob, FOAF.name, name))
g.add((bob, FOAF.age, age))
g.add((bob, FOAF.knows, linda))
g.add((linda, RDF.type, FOAF.Person))
g.add((linda, FOAF.name, Literal("Linda")))

print(g.serialize())
```

outputs:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.org/people/Bob> a foaf:Person ;
    foaf:age 24 ;
    foaf:knows [ a foaf:Person ;
        foaf:name "Linda" ] ;
    foaf:name "Bob" .
```

For some properties, only one value per resource makes sense (i.e they are *functional properties*, or have a max-cardinality of 1). The `set()` method is useful for this:

```
from rdflib import Graph, URIRef, Literal
from rdflib.namespace import FOAF

g = Graph()
bob = URIRef("http://example.org/people/Bob")

g.add((bob, FOAF.age, Literal(42)))
print(f"Bob is {g.value(bob, FOAF.age)}")
# prints: Bob is 42

g.set((bob, FOAF.age, Literal(43))) # replaces 42 set above
print(f"Bob is now {g.value(bob, FOAF.age)}")
# prints: Bob is now 43
```

`rdflib.graph.Graph.value()` is the matching query method. It will return a single value for a property, optionally raising an exception if there are more.

You can also add triples by combining entire graphs, see *Set Operations on RDFLib Graphs*.

### 1.3.3 Removing Triples

Similarly, triples can be removed by a call to `remove()`:

`Graph.remove(triple)`

Remove a triple from the graph

If the triple does not provide a context attribute, removes the triple from all contexts.

#### Parameters

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **triple** (`Tuple[Optional[Node], Optional[Node], Optional[Node]]`) –

#### Return type

`TypeVar(_GraphT, bound= Graph)`

When removing, it is possible to leave parts of the triple unspecified (i.e. passing `None`), this will remove all matching triples:

```
g.remove((bob, None, None)) # remove all triples about bob
```

### 1.3.4 An example

LiveJournal produces FOAF data for their users, but they seem to use `foaf:member_name` for a person's full name but `foaf:member_name` isn't in FOAF's namespace and perhaps they should have used `foaf:name`

To retrieve some LiveJournal data, add a `foaf:name` for every `foaf:member_name` and then remove the `foaf:member_name` values to ensure the data actually aligns with other FOAF data, we could do this:

```
from rdflib import Graph
from rdflib.namespace import FOAF

g = Graph()
# get the data
g.parse("http://danbri.livejournal.com/data/foaf")

# for every foaf:member_name, add foaf:name and remove foaf:member_name
for s, p, o in g.triples((None, FOAF['member_name'], None)):
    g.add((s, FOAF['name'], o))
    g.remove((s, FOAF['member_name'], o))
```

**Note:** Since rdflib 5.0.0, using `foaf:member_name` is somewhat prevented in RDFlib since FOAF is declared as a `ClosedNamespace()` class instance that has a closed set of members and `foaf:member_name` isn't one of them! If LiveJournal had used RDFlib 5.0.0, an error would have been raised for `foaf:member_name` when the triple was created.

### 1.3.5 Creating Containers & Collections

There are two convenience classes for RDF Containers & Collections which you can use instead of declaring each triple of a Containers or a Collections individually:

- `Container()` (also Bag, Seq & Alt) and
- `Collection()`

See their documentation for how.

## 1.4 Navigating Graphs

An RDF Graph is a set of RDF triples, and we try to mirror exactly this in RDFLib. The Python `Graph()` tries to emulate a container type.

### 1.4.1 Graphs as Iterators

RDFLib graphs override `__iter__()` in order to support iteration over the contained triples:

```
for s, p, o in someGraph:
    if not (s, p, o) in someGraph:
        raise Exception("Iterator / Container Protocols are Broken!!")
```

This loop iterates through all the subjects(s), predicates (p) & objects (o) in `someGraph`.

### 1.4.2 Contains check

Graphs implement `__contains__()`, so you can check if a triple is in a graph with a `triple in graph` syntax:

```
from rdflib import URIRef
from rdflib.namespace import RDF

bob = URIRef("http://example.org/people/bob")
if (bob, RDF.type, FOAF.Person) in graph:
    print("This graph knows that Bob is a person!")
```

Note that this triple does not have to be completely bound:

```
if (bob, None, None) in graph:
    print("This graph contains triples about Bob!")
```



### 1.4.3 Set Operations on RDFLib Graphs

Graphs override several python's operators: `__iadd__()`, `__isub__()`, etc. This supports addition, subtraction and other set-operations on Graphs:

operation	effect
<code>G1 + G2</code>	return new graph with union (triples on both)
<code>G1 += G2</code>	in place union / addition
<code>G1 - G2</code>	return new graph with difference (triples in G1, not in G2)
<code>G1 -= G2</code>	in place difference / subtraction
<code>G1 &amp; G2</code>	intersection (triples in both graphs)
<code>G1 ^ G2</code>	xor (triples in either G1 or G2, but not in both)

**Warning:** Set-operations on graphs assume Blank Nodes are shared between graphs. This may or may not be what you want. See *Merging graphs* for details.

### 1.4.4 Basic Triple Matching

Instead of iterating through all triples, RDFLib graphs support basic triple pattern matching with a `triples()` function. This function is a generator of triples that match a pattern given by arguments, i.e. arguments restrict the triples that are returned. Terms that are `None` are treated as a wildcard. For example:

```
g.parse("some_foaf.ttl")
# find all subjects (s) of type (rdf:type) person (foaf:Person)
for s, p, o in g.triples((None, RDF.type, FOAF.Person)):
    print(f"{s} is a person")

# find all subjects of any type
for s, p, o in g.triples((None, RDF.type, None)):
    print(f"{s} is a {o}")

# create a graph
bobgraph = Graph()
# add all triples with subject 'bob'
bobgraph += g.triples((bob, None, None))
```

If you are not interested in whole triples, you can get only the bits you want with the methods `objects()`, `subjects()`, `predicates()`, `predicate_objects()`, etc. Each take parameters for the components of the triple to constraint:

```
for person in g.subjects(RDF.type, FOAF.Person):
    print("{} is a person".format(person))
```

Finally, for some properties, only one value per resource makes sense (i.e they are *functional properties*, or have a max-cardinality of 1). The `value()` method is useful for this, as it returns just a single node, not a generator:

```
# get any name of bob
name = g.value(bob, FOAF.name)
# get the one person that knows bob and raise an exception if more are found
mbox = g.value(predicate = FOAF.name, object=bob, any=False)
```

### 1.4.5 Graph methods for accessing triples

Here is a list of all convenience methods for querying Graphs:

Graph.**triples**(*triple*)

Generator over the triple store

Returns triples that match the given triple pattern. If triple pattern does not provide a context, all contexts will be searched.

**Parameters**

**triple** (Tuple[Optional[Node], Union[Path, Node, None], Optional[Node]]) –

**Return type**

Generator[Union[Tuple[Node, Node, Node], Tuple[Node, Path, Node]], None, None]

Graph.**value**(*subject=None, predicate=rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#value'), object=None, default=None, any=True*)

Get a value for a pair of two criteria

Exactly one of subject, predicate, object must be None. Useful if one knows that there may only be one value.

It is one of those situations that occur a lot, hence this ‘macro’ like utility

Parameters: subject, predicate, object – exactly one must be None default – value to be returned if no values found any – if True, return any value in the case there is more than one, else, raise UniquenessError

**Parameters**

- **subject** (Optional[Node]) –
- **predicate** (Optional[Node]) –
- **object** (Optional[Node]) –
- **default** (Optional[Node]) –
- **any** (bool) –

**Return type**

Optional[Node]

Graph.**subjects**(*predicate=None, object=None, unique=False*)

A generator of (optionally unique) subjects with the given predicate and object

**Parameters**

- **predicate** (Union[None, Path, Node]) –
- **object** (Optional[Node]) –
- **unique** (bool) –

**Return type**

Generator[Node, None, None]

Graph.**objects**(*subject=None, predicate=None, unique=False*)

A generator of (optionally unique) objects with the given subject and predicate

**Parameters**

- **subject** (Optional[Node]) –
- **predicate** (Union[None, Path, Node]) –
- **unique** (bool) –

**Return type**`Generator[Node, None, None]``Graph.predicates(subject=None, object=None, unique=False)`

A generator of (optionally unique) predicates with the given subject and object

**Parameters**

- **subject** (`Optional[Node]`) –
- **object** (`Optional[Node]`) –
- **unique** (`bool`) –

**Return type**`Generator[Node, None, None]``Graph.subject_objects(predicate=None, unique=False)`

A generator of (optionally unique) (subject, object) tuples for the given predicate

**Parameters**

- **predicate** (`Union[None, Path, Node]`) –
- **unique** (`bool`) –

**Return type**`Generator[Tuple[Node, Node], None, None]``Graph.subject_predicates(object=None, unique=False)`

A generator of (optionally unique) (subject, predicate) tuples for the given object

**Parameters**

- **object** (`Optional[Node]`) –
- **unique** (`bool`) –

**Return type**`Generator[Tuple[Node, Node], None, None]``Graph.predicate_objects(subject=None, unique=False)`

A generator of (optionally unique) (predicate, object) tuples for the given subject

**Parameters**

- **subject** (`Optional[Node]`) –
- **unique** (`bool`) –

**Return type**`Generator[Tuple[Node, Node], None, None]`

## 1.5 Querying with SPARQL

### 1.5.1 Run a Query

The RDFLib comes with an implementation of the [SPARQL 1.1 Query](#) and [SPARQL 1.1 Update](#) query languages.

Queries can be evaluated against a graph with the `rdflib.graph.Graph.query()` method, and updates with `rdflib.graph.Graph.update()`.

The query method returns a `rdflib.query.Result` instance. For SELECT queries, iterating over this returns `rdflib.query.ResultRow` instances, each containing a set of variable bindings. For CONSTRUCT/DESCRIBE queries, iterating over the result object gives the triples. For ASK queries, iterating will yield the single boolean answer, or evaluating the result object in a boolean-context (i.e. `bool(result)`)

For example...

```
import rdflib
g = rdflib.Graph()
g.parse("http://danbri.org/foaf.rdf#")

knows_query = """
SELECT DISTINCT ?aname ?bname
WHERE {
    ?a foaf:knows ?b .
    ?a foaf:name ?aname .
    ?b foaf:name ?bname .
}"""

qres = g.query(knows_query)
for row in qres:
    print(f"{row.aname} knows {row.bname}")
```

The results are tuples of values in the same order as your SELECT arguments. Alternatively, the values can be accessed by variable name, either as attributes, or as items, e.g. `row.b` and `row["b"]` are equivalent. The above, given the appropriate data, would print something like:

```
Timothy Berners-Lee knows Edd Dumbill
Timothy Berners-Lee knows Jennifer Golbeck
Timothy Berners-Lee knows Nicholas Gibbins
...
```

As an alternative to using SPARQLs PREFIX, namespace bindings can be passed in with the `initNs` kwarg, see [Namespaces and Bindings](#).

Variables can also be pre-bound, using the `initBindings` kwarg which can pass in a dict of initial bindings. This is particularly useful for prepared queries, as described below.

## 1.5.2 Update Queries

Update queries are performed just like reading queries but using the `rdflib.graph.Graph.update()` method. An example:

```
from rdflib import Graph

# Create a Graph, add in some test data
g = Graph()
g.parse(
    data="""
        <x:> a <c:> .
        <y:> a <c:> .
    """,
    format="turtle"
)
```

(continues on next page)

(continued from previous page)

```

# Select all the things (s) that are of type (rdf:type) c:
qres = g.query("""SELECT ?s WHERE { ?s a <c:> }""")

for row in qres:
    print(f"{row.s}")
# prints:
# x:
# y:

# Add in a new triple using SPATQL UPDATE
g.update("""INSERT DATA { <z:> a <c:> }""")

# Select all the things (s) that are of type (rdf:type) c:
qres = g.query("""SELECT ?s WHERE { ?s a <c:> }""")

print("After update:")
for row in qres:
    print(f"{row.s}")
# prints:
# x:
# y:
# z:

# Change type of <y:> from <c:> to <d:>
g.update("""
    DELETE { <y:> a <c:> }
    INSERT { <y:> a <d:> }
    WHERE { <y:> a <c:> }
""")

print("After second update:")
qres = g.query("""SELECT ?s ?o WHERE { ?s a ?o }""")
for row in qres:
    print(f"{row.s} a {row.o}")
# prints:
# x: a c:
# z: a c:
# y: a d:

```

### 1.5.3 Querying a Remote Service

The SERVICE keyword of SPARQL 1.1 can send a query to a remote SPARQL endpoint.

```

import rdflib

g = rdflib.Graph()
qres = g.query(
    """
    SELECT ?s
    WHERE {
        SERVICE <http://dbpedia.org/sparql> {

```

(continues on next page)

(continued from previous page)

```

        ?s a ?o .
    }
}
LIMIT 3
"""
)

for row in qres:
    print(row.s)

```

This example sends a query to [DBpedia](#)'s SPARQL endpoint service so that it can run the query and then send back the result:

```

<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.openlinksw.com/schemas/
↳ virtcxml#FacetCategoryPattern>
<http://www.w3.org/2001/XMLSchema#anyURI> <http://www.w3.org/2000/01/rdf-schema#Datatype>
<http://www.w3.org/2001/XMLSchema#anyURI> <http://www.w3.org/2000/01/rdf-schema#Datatype>

```

### 1.5.4 Prepared Queries

RDFLib lets you *prepare* queries before execution, this saves re-parsing and translating the query into SPARQL Algebra each time.

The method `rdflib.plugins.sparql.prepareQuery()` takes a query as a string and will return a `rdflib.plugins.sparql.sparql.Query` object. This can then be passed to the `rdflib.graph.Graph.query()` method.

The `initBindings` kwarg can be used to pass in a dict of initial bindings:

```

q = prepareQuery(
    "SELECT ?s WHERE { ?person foaf:knows ?s .}",
    initNs = { "foaf": FOAF }
)

g = rdflib.Graph()
g.parse("foaf.rdf")

tim = rdflib.URIRef("http://www.w3.org/People/Berners-Lee/card#i")

for row in g.query(q, initBindings={'person': tim}):
    print(row)

```

### 1.5.5 Custom Evaluation Functions

For experts, it is possible to override how bits of SPARQL algebra are evaluated. By using the `setuptools` entry-point `rdf.plugins.sparqlevel`, or simply adding to an entry to `rdflib.plugins.sparql.CUSTOM_EVALS`, a custom function can be registered. The function will be called for each algebra component and may raise `NotImplementedError` to indicate that this part should be handled by the default implementation.

See `examples/custom_eval.py`

## 1.6 Utilities & convenience functions

For RDF programming, RDFLib and Python may not be the fastest tools, but we try hard to make them the easiest and most convenient to use and thus the *fastest* overall!

This is a collection of hints and pointers for hassle-free RDF coding.

### 1.6.1 Functional properties

Use `value()` and `set()` to work with *functional property* instances, i.e. properties than can only occur once for a resource.

```
from rdflib import Graph, URIRef, Literal, BNode
from rdflib.namespace import FOAF, RDF

g = Graph()
g.bind("foaf", FOAF)

# Add demo data
bob = URIRef("http://example.org/people/Bob")
g.add((bob, RDF.type, FOAF.Person))
g.add((bob, FOAF.name, Literal("Bob")))
g.add((bob, FOAF.age, Literal(38)))

# To get a single value, use 'value'
print(g.value(bob, FOAF.age))
# prints: 38

# To change a single of value, use 'set'
g.set((bob, FOAF.age, Literal(39)))
print(g.value(bob, FOAF.age))
# prints: 39
```

### 1.6.2 Slicing graphs

Python allows slicing arrays with a slice object, a triple of start, stop and step-size:

```
for i in range(20)[2:9:3]:
    print(i)
# prints:
# 2, 5, 8
```

RDFLib graphs override `__getitem__` and we pervert the slice triple to be a RDF triple instead. This lets slice syntax be a shortcut for `triples()`, `subject_predicates()`, `__contains__()`, and other Graph query-methods:

```
from rdflib import Graph, URIRef, Literal, BNode
from rdflib.namespace import FOAF, RDF

g = Graph()
g.bind("foaf", FOAF)
```

(continues on next page)

(continued from previous page)

```
# Add demo data
bob = URIRef("http://example.org/people/Bob")
bill = URIRef("http://example.org/people/Bill")
g.add((bob, RDF.type, FOAF.Person))
g.add((bob, FOAF.name, Literal("Bob")))
g.add((bob, FOAF.age, Literal(38)))
g.add((bob, FOAF.knows, bill))

print(g[:])
# same as
print(iter(g))

print(g[bob])
# same as
print(g.predicate_objects(bob))

print(g[bob: FOAF.knows])
# same as
print(g.objects(bob, FOAF.knows))

print(g[bob: FOAF.knows: bill])
# same as
print((bob, FOAF.knows, bill) in g)

print(g[:FOAF.knows])
# same as
print(g.subject_objects(FOAF.knows))
```

See [examples.slice](#) for a complete example.

---

**Note:** Slicing is convenient for run-once scripts for playing around in the Python REPL, however since slicing returns tuples of varying length depending on which parts of the slice are bound, you should be careful using it in more complicated programs. If you pass in variables, and they are `None` or `False`, you may suddenly get a generator of different length tuples back than you expect.

---

### 1.6.3 SPARQL Paths

SPARQL property paths are possible using overridden operators on `URIRefs`. See [examples.foafpaths](#) and [rdflib.paths](#).



### 1.6.4 Serializing a single term to N3

For simple output, or simple serialisation, you often want a nice readable representation of a term. All terms (URIRef, Literal etc.) have a `n3` method, which will return a suitable N3 format:

```
from rdflib import Graph, URIRef, Literal
from rdflib.namespace import FOAF

# A URIRef
person = URIRef("http://xmlns.com/foaf/0.1/Person")
print(person.n3())
# prints: <http://xmlns.com/foaf/0.1/Person>

# Simplifying the output with a namespace prefix:
g = Graph()
g.bind("foaf", FOAF)

print(person.n3(g.namespace_manager))
# prints foaf:Person

# A typed literal
l = Literal(2)
print(l.n3())
# prints "2"^^<http://www.w3.org/2001/XMLSchema#integer>

# Simplifying the output with a namespace prefix
# XSD is built in, so no need to bind() it!
l.n3(g.namespace_manager)
# prints: "2"^^xsd:integer
```

### 1.6.5 Parsing data from a string

You can parse data from a string with the `data` param:

```
from rdflib import Graph

g = Graph().parse(data="<a:> <p:> <p:>.")
for r in g.triples((None, None, None)):
    print(r)
# prints: (rdflib.term.URIRef('a:'), rdflib.term.URIRef('p:'), rdflib.term.URIRef('p:'))
```

### 1.6.6 Command Line tools

RDFLib includes a handful of commandline tools, see [rdflib.tools](#).

## 1.7 examples Package

These examples all live in `./examples` in the source-distribution of RDFLib.

### 1.7.1 conjunctive\_graphs Module

An RDFLib `ConjunctiveGraph` is an (unnamed) aggregation of all the `Named Graphs` within a `Store`. The `get_context()` method can be used to get a particular named graph for use, such as to add triples to, or the default graph can be used.

This example shows how to create `Named Graphs` and work with the conjunction (union) of all the graphs.

### 1.7.2 custom\_datatype Module

RDFLib can map between RDF data-typed literals and Python objects.

Mapping for integers, floats, dateTimes, etc. are already added, but you can also add your own.

This example shows how `rdflib.term.bind()` lets you register new mappings between literal datatypes and Python objects

### 1.7.3 custom\_eval Module

This example shows how a custom evaluation function can be added to handle certain SPARQL Algebra elements.

A custom function is added that adds `rdfs:subClassOf` “inference” when asking for `rdf:type` triples.

Here the custom eval function is added manually, normally you would use `setuptools` and `entry_points` to do it: i.e. in your `setup.py`:

```
entry_points = {
    'rdf.plugins.sparqleval': [
        'myfunc = mypackage:MyFunction',
    ],
}
```

`examples.custom_eval.customEval(ctx, part)`

Rewrite triple patterns to get super-classes

### 1.7.4 foafpaths Module

SPARQL 1.1 defines path operators for combining/repeating predicates in triple-patterns.

We overload some Python operators on `URIRefs` to allow creating path operators directly in Python.

Operator	Path
<code>p1 / p2</code>	Path sequence
<code>p1   p2</code>	Path alternative
<code>p1 * '*'</code>	chain of 0 or more p's
<code>p1 * '+'</code>	chain of 1 or more p's
<code>p1 * '?'</code>	0 or 1 p
<code>~p1</code>	p1 inverted, i.e. (s p1 o) <=> (o ~p1 s)
<code>-p1</code>	NOT p1, i.e. any property but p1

These can then be used in property position for s,p,o triple queries for any graph method.

See the docs for [rdflib.paths](#) for the details.

This example shows how to get the name of friends (i.e values two steps away x knows y, y name z) with a single query.

### 1.7.5 prepared\_query Module

SPARQL Queries be prepared (i.e parsed and translated to SPARQL algebra) by the [rdflib.plugins.sparql.prepareQuery\(\)](#) method.

initNs can be used instead of PREFIX values.

When executing, variables can be bound with the `initBindings` keyword parameter.

### 1.7.6 resource\_example Module

RDFLib has a [Resource](#) class, for a resource-centric API. The [Graph](#) class also has a `resource` function that can be used to create resources and manipulate them by quickly adding or querying for triples where this resource is the subject.

This example shows `g.resource()` in action.

### 1.7.7 berkeleydb\_example Module

BerkeleyDB in use as a persistent Graph store.

Example 1: simple actions

- creating a ConjunctiveGraph using the BerkeleyDB Store
- adding triples to it
- counting them
- closing the store, emptying the graph
- re-opening the store using the same DB files
- getting the same count of triples as before

Example 2: larger data

- loads multiple graphs downloaded from GitHub into a BerkeleyDB-backed graph stored in the folder `gsq_vocabs`.
- does not delete the DB at the end so you can see it on disk

`examples.berkeleydb_example.example_1()`

Creates a ConjunctiveGraph and performs some BerkeleyDB tasks with it

`examples.berkeleydb_example.example_2()`

Loads a number of SKOS vocabularies from GitHub into a BerkeleyDB-backed graph stored in the local folder `'gsq_vocabs'`

**Should print out the number of triples after each load, e.g.:**

177 248 289 379 421 628 764 813 965 1381 9666 9719 ...

### 1.7.8 slice Module

RDFLib Graphs (and Resources) can be “sliced” with [] syntax

This is a short-hand for iterating over triples.

Combined with SPARQL paths (see `foafpaths.py`) - quite complex queries can be realised.

See `rdflib.graph.Graph.__getitem__()` for details

### 1.7.9 smushing Module

A FOAF smushing example.

Filter a graph by normalizing all `foaf:Persons` into URIs based on their `mbox_sha1sum`.

Suppose I get two FOAF documents each talking about the same person (according to `mbox_sha1sum`) but they each used a `rdflib.term.BNode` for the subject. For this demo I’ve combined those two documents into one file:

This filters a graph by changing every subject with a `foaf:mbox_sha1sum` into a new subject whose URI is based on the `sha1sum`. This new graph might be easier to do some operations on.

An advantage of this approach over other methods for collapsing BNodes is that I can incrementally process new FOAF documents as they come in without having to access my ever-growing archive. Even if another `65b983bb397fb71849da910996741752ace8369b` document comes in next year, I would still give it the same stable subject URI that merges with my existing data.

### 1.7.10 sparql\_query\_example Module

SPARQL Query using `rdflib.graph.Graph.query()`

The method returns a `Result`, iterating over this yields `ResultRow` objects

The variable bindings can be accessed as attributes of the row objects For variable names that are not valid python identifiers, dict access (i.e. with `row[var]` / `__getitem__`) is also possible.

`vars` contains the variables

### 1.7.11 sparql\_update\_example Module

SPARQL Update statements can be applied with `rdflib.graph.Graph.update()`

### 1.7.12 sparqlstore\_example Module

Simple examples showing how to use the SPARQLStore

### 1.7.13 swap\_primer Module

This is a simple primer using some of the example stuff in the Primer on N3:

<http://www.w3.org/2000/10/swap/Primer>

### 1.7.14 transitive Module

An example illustrating how to use the `transitive_subjects()` and `transitive_objects()` graph methods

#### Formal definition

The `transitive_objects()` method finds all nodes such that there is a path from subject to one of those nodes using only the predicate property in the triples. The `transitive_subjects()` method is similar; it finds all nodes such that there is a path from the node to the object using only the predicate property.

#### Informal description, with an example

In brief, `transitive_objects()` walks forward in a graph using a particular property, and `transitive_subjects()` walks backward. A good example uses a property `ex:parent`, the semantics of which are biological parentage. The `transitive_objects()` method would get all the ancestors of a particular person (all nodes such that there is a parent path between the person and the object). The `transitive_subjects()` method would get all the descendants of a particular person (all nodes such that there is a parent path between the node and the person). So, say that your URI is `ex:person`.

This example would get all of your (known) ancestors, and then get all the (known) descendants of your maternal grandmother.

**Warning:** The `transitive_objects()` method has the start node as the *first* argument, but the `transitive_subjects()` method has the start node as the *second* argument.

#### User-defined transitive closures

The method `transitiveClosure()` returns transitive closures of user-defined functions.

### 1.7.15 secure\_with\_audit Module

This example demonstrates how to use [Python audit hooks](#) to block access to files and URLs.

It installs an audit hook with `sys.addaudithook` that blocks access to files and URLs that end with `blocked.jsonld`.

The code in the example then verifies that the audit hook is blocking access to URLs and files as expected.

`examples.secure_with_audit.audit_hook(name, args)`

An audit hook that blocks access when an attempt is made to open a file or URL that ends with `blocked.jsonld`.

Details of the audit events can be seen in the [audit events table](#).

#### Parameters

- **name** (`str`) – The name of the audit event.
- **args** (`Tuple[Any, ...]`) – The arguments of the audit event.

**Return type**`None`**Returns**`None` if the audit hook does not block access.**Raises****PermissionError** – If the file or URL being accessed ends with `blocked.jsonld`.`examples.secure_with_audit.main()`

The main code of the example.

The important steps are: :rtype: `None`

- Install a custom audit hook that blocks some URLs and files.
- Attempt to parse a JSON-LD document that will result in a blocked URL being accessed.
- Verify that the audit hook blocked access to the URL.
- Attempt to parse a JSON-LD document that will result in a blocked file being accessed.
- Verify that the audit hook blocked access to the file.

### 1.7.16 secure\_with\_urlopen Module

This example demonstrates how to use a custom global URL opener installed with `urllib.request.install_opener` to block access to URLs.

**class** `examples.secure_with_urlopen.SecuredHTTPHandler(debuglevel=0)`Bases: `HTTPHandler`

A HTTP handler that blocks access to URLs that end with “blocked.jsonld”.

`__module__ = 'examples.secure_with_urlopen'`**http\_open**(*req*)

Block access to URLs that end with “blocked.jsonld”.

**Parameters****req** (`Request`) – The request to open.**Return type**`HTTPResponse`**Returns**

The response.

**Raises****PermissionError** – If the URL ends with “blocked.jsonld”.`examples.secure_with_urlopen.main()`

The main code of the example.

The important steps are: :rtype: `None`

- Install a custom global URL opener that blocks some URLs.
- Attempt to parse a JSON-LD document that will result in a blocked URL being accessed.
- Verify that the URL opener blocked access to the URL.

If you are familiar with RDF and are looking for details on how RDFLib handles it, these are for you:

## 2.1 RDF terms in rdflib

Terms are the kinds of objects that can appear in a RDFLib's graph's triples. Those that are part of core RDF concepts are: `IRIs`, `Blank Node` and `Literal`, the latter consisting of a literal value and either a `datatype` or an `RFC 3066` language tag.

---

**Note:** RDFLib's class for representing IRIs/URIs is called “`URIRef`” because, at the time it was implemented, that was what the then current RDF specification called URIs/IRIs. We preserve that class name but refer to the RDF object as “`IRI`”.

---

### 2.1.1 Class hierarchy

All terms in RDFLib are sub-classes of the `rdflib.term.Identifier` class. A class diagram of the various terms is:

Fig. 1: Term Class Hierarchy

Nodes are a subset of the Terms that underlying stores actually persist.

The set of such Terms depends on whether or not the store is formula-aware. Stores that aren't formula-aware only persist those terms core to the RDF Model but those that are formula-aware also persist the N3 extensions. However, utility terms that only serve the purpose of matching nodes by term-patterns will probably only be terms and not nodes.

### 2.1.2 Python Classes

The three main RDF objects - *IRI*, *Blank Node* and *Literal* are represented in RDFLib by these three main Python classes:

## URIRef

An IRI (Internationalized Resource Identifier) is represented within RDFLib using the `URIRef` class. From [the RDF 1.1 specification's IRI section](#):

Here is the `URIRef` class' auto-built documentation:

```
class rdflib.term.URIRef(value: str, base: Optional[str] = None)
    RDF 1.1's IRI Section https://www.w3.org/TR/rdf11-concepts/#section-IRIs
```

---

**Note:** Documentation on RDF outside of RDFLib uses the term IRI or URI whereas this class is called `URIRef`. This is because it was made when the first version of the RDF specification was current, and it used the term *URIRef*, see [RDF 1.0 URIRef](#)

---

An IRI (Internationalized Resource Identifier) within an RDF graph is a Unicode string that conforms to the syntax defined in RFC 3987.

IRIs in the RDF abstract syntax **MUST** be absolute, and **MAY** contain a fragment identifier.

IRIs are a generalization of URIs [RFC3986] that permits a wider range of Unicode characters.

```
>>> from rdflib import URIRef
>>> uri = URIRef()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: __new__() missing 1 required positional argument: 'value'
>>> uri = URIRef('')
>>> uri
rdflib.term.URIRef('')
>>> uri = URIRef('http://example.com')
>>> uri
rdflib.term.URIRef('http://example.com')
>>> uri.n3()
'<http://example.com>'
```

## BNodes

In RDF, a blank node (also called `BNode`) is a node in an RDF graph representing a resource for which an IRI or literal is not given. The resource represented by a blank node is also called an anonymous resource. According to the RDF standard, a blank node can only be used as subject or object in a triple, although in some syntaxes like Notation 3 it is acceptable to use a blank node as a predicate. If a blank node has a node ID (not all blank nodes are labelled in all RDF serializations), it is limited in scope to a particular serialization of the RDF graph, i.e. the node `p1` in one graph does not represent the same node as a node named `p1` in any other graph – [wikipedia](#)

Here is the `BNode` class' auto-built documentation:

```
class rdflib.term.BNode(value: ~typing.Optional[str] = None, _sn_gen: ~typing.Callable[[], str] = <function
    _serial_number_generator.<locals>._generator>, _prefix: str = 'N')
```

RDF 1.1's Blank Nodes Section: <https://www.w3.org/TR/rdf11-concepts/#section-blank-nodes>

Blank Nodes are local identifiers for unnamed nodes in RDF graphs that are used in some concrete RDF syntaxes or RDF store implementations. They are always locally scoped to the file or RDF store, and are not persistent or portable identifiers for blank nodes. The identifiers for Blank Nodes are not part of the RDF abstract syntax, but are entirely dependent on particular concrete syntax or implementation (such as Turtle, JSON-LD).



—

RDFLib's `BNode` class makes unique IDs for all the Blank Nodes in a Graph but you should *never* expect, or reply on, BNodes' IDs to match across graphs, or even for multiple copies of the same graph, if they are regenerated from some non-RDFLib source, such as loading from RDF data.

```
>>> from rdflib import BNode
>>> bn = BNode()
>>> bn
rdflib.term.BNode('AFwALAKU0')
>>> bn.n3()
'_:AFwALAKU0'
```

## Literals

Literals are attribute values in RDF, for instance, a person's name, the date of birth, height, etc. and are stored using simple data types, e.g. *string*, *double*, *dateTime* etc. This usually looks something like this:

```
name = Literal("Nicholas") # the name 'Nicholas', as a string
age = Literal(39, datatype=XSD.integer) # the number 39, as an integer
```

A slightly special case is a *langString* which is a *string* with a language tag, e.g.:

```
name = Literal("Nicholas", lang="en") # the name 'Nicholas', as an English string
imie = Literal("Mikołaj", lang="pl") # the Polish version of the name 'Nicholas'
```

Special literal types indicated by use of a custom IRI for a literal's *datatype* value, for example the GeoSPARQL RDF standard invents a custom datatype, `geoJSONLiteral` to indicate GeoJSON geometry serializations like this:

```
GEO = Namespace("http://www.opengis.net/ont/geosparql#")

geojson_geometry = Literal(
    '{"type": "Point", "coordinates": [-83.38, 33.95]}',
    datatype=GEO.geoJSONLiteral)
```

Here is the `Literal` class' auto-built documentation, followed by notes on `Literal` from the [RDF 1.1 specification](#) 'Literals' section.

```
class rdflib.term.Literal(lexical_or_value: Any, lang: Optional[str] = None, datatype: Optional[str] =
                        None, normalize: Optional[bool] = None)
```

RDF 1.1's Literals Section: <http://www.w3.org/TR/rdf-concepts/#section-Graph-Literal>

Literals are used for values such as strings, numbers, and dates.

A literal in an RDF graph consists of two or three elements:

- a lexical form, being a Unicode string, which SHOULD be in Normal Form C
- a datatype IRI, being an IRI identifying a datatype that determines how the lexical form maps to a literal value, and
- if and only if the datatype IRI is <http://www.w3.org/1999/02/22-rdf-syntax-ns#langString>, a non-empty language tag. The language tag MUST be well-formed according to section 2.2.9 of [Tags for identifying languages](#).

A literal is a language-tagged string if the third element is present. Lexical representations of language tags MAY be converted to lower case. The value space of language tags is always in lower case.

For valid XSD datatypes, the lexical form is optionally normalized at construction time. Default behaviour is set by `rdflib.NORMALIZE_LITERALS` and can be overridden by the `normalize` parameter to `__new__`

Equality and hashing of Literals are done based on the lexical form, i.e.:

```
>>> from rdflib.namespace import XSD
```

```
>>> Literal('01') != Literal('1') # clear - strings differ
True
```

but with data-type they get normalized:

```
>>> Literal('01', datatype=XSD.integer) != Literal('1', datatype=XSD.integer)
False
```

unless disabled:

```
>>> Literal('01', datatype=XSD.integer, normalize=False) != Literal('1',
↳datatype=XSD.integer)
True
```

Value based comparison is possible:

```
>>> Literal('01', datatype=XSD.integer).eq(Literal('1', datatype=XSD.float))
True
```

The `eq` method also provides limited support for basic python types:

```
>>> Literal(1).eq(1) # fine - int compatible with xsd:integer
True
>>> Literal('a').eq('b') # fine - str compatible with plain-lit
False
>>> Literal('a', datatype=XSD.string).eq('a') # fine - str compatible with
↳xsd:string
True
>>> Literal('a').eq(1) # not fine, int incompatible with plain-lit
NotImplemented
```

Greater-than/less-than ordering comparisons are also done in value space, when compatible datatypes are used. Incompatible datatypes are ordered by DT, or by lang-tag. For other nodes the ordering is `None < BNode < URIRef < Literal`

Any comparison with non-`rdflib` Node are “NotImplemented” In PY3 this is an error.

```
>>> from rdflib import Literal, XSD
>>> lit2006 = Literal('2006-01-01',datatype=XSD.date)
>>> lit2006.toPython()
datetime.date(2006, 1, 1)
>>> lit2006 < Literal('2007-01-01',datatype=XSD.date)
True
>>> Literal(datetime.utcnow()).datatype
```

(continues on next page)

(continued from previous page)

```

rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#dateTime')
>>> Literal(1) > Literal(2) # by value
False
>>> Literal(1) > Literal(2.0) # by value
False
>>> Literal('1') > Literal(1) # by DT
True
>>> Literal('1') < Literal('1') # by lexical form
False
>>> Literal('a', lang='en') > Literal('a', lang='fr') # by lang-tag
False
>>> Literal(1) > URIRef('foo') # by node-type
True

```

The > < operators will eat this NotImplemented and throw a TypeError (py3k):

```

>>> Literal(1).__gt__(2.0)
NotImplemented

```

A literal in an RDF graph contains one or two named components.

All literals have a lexical form being a Unicode string, which SHOULD be in Normal Form C.

Plain literals have a lexical form and optionally a language tag as defined by [RFC 3066](#), normalized to lowercase. An exception will be raised if illegal language-tags are passed to `rdflib.term.Literal.__new__()`.

Typed literals have a lexical form and a datatype URI being an RDF URI reference.

---

**Note:** When using the language tag, care must be taken not to confuse language with locale. The language tag relates only to human language text. Presentational issues should be addressed in end-user applications.

---



---

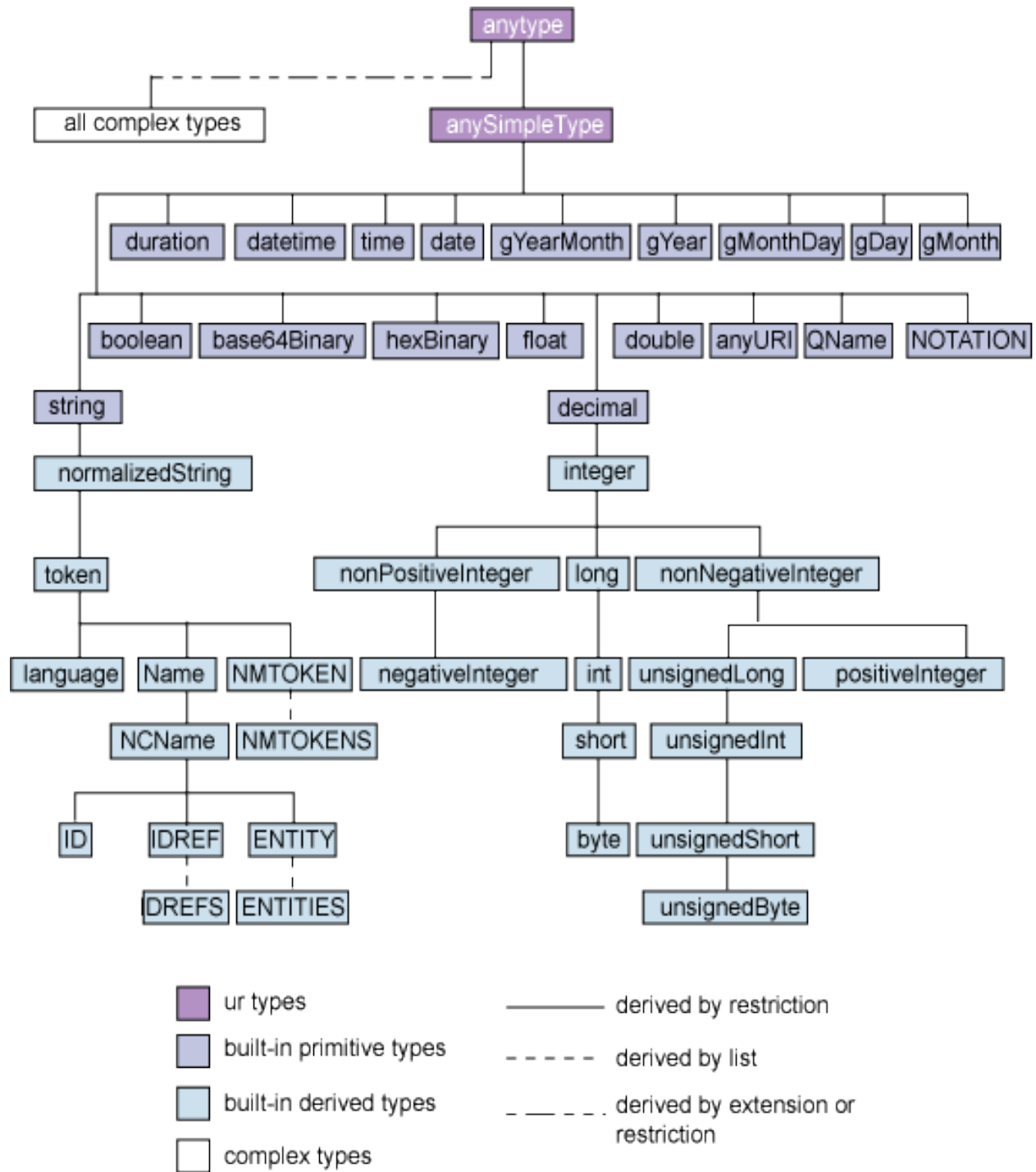
**Note:** The case normalization of language tags is part of the description of the abstract syntax, and consequently the abstract behaviour of RDF applications. It does not constrain an RDF implementation to actually normalize the case. Crucially, the result of comparing two language tags should not be sensitive to the case of the original input. – [RDF Concepts and Abstract Syntax](#)

---

## Common XSD datatypes

Most simple literals such as *string* or *integer* have XML Schema (XSD) datatypes defined for them, see the figure below. Additionally, these XSD datatypes are listed in the [XSD Namespace class](#) that ships with RDFLib, so many Python code editors will prompt you with autocomplete for them when using it.

Remember, you don't *have* to use XSD datatypes and can always make up your own, as GeoSPARQL does, as described above.



## Python conversions

RDFLib Literals essentially behave like unicode characters with an XML Schema datatype or language attribute.

The class provides a mechanism to both convert Python literals (and their built-ins such as time/date/datetime) into equivalent RDF Literals and (conversely) convert Literals to their Python equivalent. This mapping to and from Python literals is done as follows:

XML Datatype	Python type
None	None <sup>1</sup>
xsd:time	time <sup>2</sup>
xsd:date	date
xsd:dateTime	datetime
xsd:string	None
xsd:normalizedString	None
xsd:token	None
xsd:language	None
xsd:boolean	boolean
xsd:decimal	Decimal
xsd:integer	long
xsd:nonPositiveInteger	int
xsd:long	long
xsd:nonNegativeInteger	int
xsd:negativeInteger	int
xsd:int	long
xsd:unsignedLong	long
xsd:positiveInteger	int
xsd:short	int
xsd:unsignedInt	long
xsd:byte	int
xsd:unsignedShort	int
xsd:unsignedByte	int
xsd:float	float
xsd:double	float
xsd:base64Binary	base64
xsd:anyURI	None
rdf:XMLLiteral	xml.dom.minidom.Document <sup>3</sup>
rdf:HTML	xml.dom.minidom.DocumentFragment

An appropriate data-type and lexical representation can be found using:

`rdflib.term._castPythonToLiteral(obj, datatype)`

Casts a tuple of a python type and a special datatype URI to a tuple of the lexical value and a datatype URI (or None)

### Parameters

- **obj** (*Any*) –
- **datatype** (*Optional[str]*) –

<sup>1</sup> plain literals map directly to value space

<sup>2</sup> Date, time and datetime literals are mapped to Python instances using the *isodate* package).

<sup>3</sup> this is a bit dirty - by accident the *html5lib* parser produces *DocumentFragments*, and the *xml* parser *Documents*, letting us use this to decide what datatype when round-tripping.

**Return type**

`Tuple[Any, Optional[str]]`

and the other direction with

`rdflib.term._castLexicalToPython(lexical, datatype)`

Map a lexical form to the value-space for the given datatype :rtype: `Any` :returns: a python object for the value or None

**Parameters**

- **lexical** (`Union[str, bytes]`) –
- **datatype** (`Optional[URIRef]`) –

All this happens automatically when creating `Literal` objects by passing Python objects to the constructor, and you never have to do this manually.

You can add custom data-types with `rdflib.term.bind()`, see also [examples.custom\\_datatype](#)

## 2.2 Namespaces and Bindings

RDFLib provides several short-cuts to working with many URIs in the same namespace.

The `rdflib.namespace` defines the `rdflib.namespace.Namespace` class which lets you easily create URIs in a namespace:

```
from rdflib import Namespace

EX = Namespace("http://example.org/")
EX.Person  # a Python attribute for EX. This example is equivalent to rdflib.term.
↳ URIRef("http://example.org/Person")

# use dict notation for things that are not valid Python identifiers, e.g.:
n['first%20name'] # as rdflib.term.URIRef("http://example.org/first%20name")
```

These two styles of namespace creation - object attribute and dict - are equivalent and are made available just to allow for valid RDF namespaces and URIs that are not valid Python identifiers. This isn't just for syntactic things like spaces, as per the example of `first%20name` above, but also for Python reserved words like `class` or `while`, so for the URI `http://example.org/class`, create it with `EX['class']`, not `EX.class`.

### 2.2.1 Common Namespaces

The `namespace` module defines many common namespaces such as RDF, RDFS, OWL, FOAF, SKOS, PROF, etc. The list of the namespaces provided grows with user contributions to RDFLib.

These Namespaces, and any others that users define, can also be associated with prefixes using the `rdflib.namespace.NamespaceManager`, e.g. using `foaf` for `http://xmlns.com/foaf/0.1/`.

Each RDFLib graph has a `namespace_manager` that keeps a list of namespace to prefix mappings. The namespace manager is populated when reading in RDF, and these prefixes are used when serialising RDF, or when parsing SPARQL queries. Prefixes can be bound with the `rdflib.graph.Graph.bind()` method:

```
from rdflib import Graph, Namespace
from rdflib.namespace import FOAF
```

(continues on next page)

(continued from previous page)

```
EX = Namespace("http://example.org/")

g = Graph()
g.bind("foaf", FOAF) # bind an RDFLib-provided namespace to a prefix
g.bind("ex", EX)     # bind a user-declared namespace to a prefix
```

The `rdflib.graph.Graph.bind()` method is actually supplied by the `rdflib.namespace.NamespaceManager` class - see next.

## 2.2.2 NamespaceManager

Each RDFLib graph comes with a `rdflib.namespace.NamespaceManager` instance in the `namespace_manager` field; you can use the `bind()` method of this instance to bind a prefix to a namespace URI, as above, however note that the `NamespaceManager` automatically performs some bindings according to a selected strategy.

Namespace binding strategies are indicated with the `bind_namespaces` input parameter to `NamespaceManager` instances and may be set via `Graph` also:

```
from rdflib import Graph
from rdflib.namespace import NamespaceManager

g = Graph(bind_namespaces="rdflib") # bind via Graph

g2 = Graph()
nm = NamespaceManager(g2, bind_namespaces="rdflib") # bind via NamespaceManager
```

Valid strategies are:

- **core:**
  - binds several core RDF prefixes only
  - owl, rdf, rdfs, xsd, xml from the `NAMESPACE_PREFIXES_CORE` object
  - this is default
- **rdflib:**
  - binds all the namespaces shipped with RDFLib as `DefinedNamespace` instances
  - all the core namespaces and all the following: brick, csvw, dc, dcat
  - dcmitype, dcterms, dcam, doap, foaf, geo, odrl, org, prof, prov, qb, sdo
  - sh, skos, sosa, ssn, time, vann, void
  - see the `NAMESPACE_PREFIXES_RDFLIB` object in `rdflib.namespace` for up-to-date list
- **none:**
  - binds no namespaces to prefixes
  - note this is NOT default behaviour
- **cc:**
  - using prefix bindings from prefix.cc which is a online prefixes database
  - not implemented yet - this is aspirational

## Re-binding

Note that regardless of the strategy employed, prefixes for namespaces can be overwritten with users preferred prefixes, for example:

```
from rdflib import Graph
from rdflib.namespace import GEO # imports GeoSPARQL's namespace

g = Graph(bind_namespaces="rdflib") # binds GeoSPARQL's namespace to prefix 'geo'

g.bind('geosp', GEO, override=True)
```

`NamespaceManager` also has a method to normalize a given url:

```
from rdflib.namespace import NamespaceManager

nm = NamespaceManager(Graph())
nm.normalizeUri(t)
```

For simple output, or simple serialisation, you often want a nice readable representation of a term. All `RDFLib` terms have a `.n3()` method, which will return a suitable N3 format and into which you can supply a `NamespaceManager` instance to provide prefixes, i.e. `.n3(namespace_manager=some_nm)`:

```
>>> from rdflib import Graph, URIRef, Literal, BNode
>>> from rdflib.namespace import FOAF, NamespaceManager

>>> person = URIRef("http://xmlns.com/foaf/0.1/Person")
>>> person.n3()
'<http://xmlns.com/foaf/0.1/Person>'

>>> g = Graph()
>>> g.bind("foaf", FOAF)

>>> person.n3(g.namespace_manager)
'foaf:Person'

>>> l = Literal(2)
>>> l.n3()
'"2"^^<http://www.w3.org/2001/XMLSchema#integer>'

>>> l.n3(NamespaceManager(Graph(), bind_namespaces="core"))
'"2"^^xsd:integer'
```

The namespace manager also has a useful method `compute_qname` `g.namespace_manager.compute_qname(x)` (or just `g.compute_qname(x)`) which takes a URI and decomposes it into the parts:

```
self.assertEqual(g.compute_qname(URIRef("http://foo/bar#baz")),
                  ("ns2", URIRef("http://foo/bar#"), "baz"))
```



### 2.2.3 Namespaces in SPARQL Queries

The `initNs` argument supplied to `query()` is a dictionary of namespaces to be expanded in the query string. If you pass no `initNs` argument, the namespaces registered with the graphs `namespace_manager` are used:

```
from rdflib.namespace import FOAF
graph.query('SELECT * WHERE { ?p a foaf:Person }', initNs={'foaf': FOAF})
```

In order to use an empty prefix (e.g. `?a :knows ?b`), use a `PREFIX` directive with no prefix in the SPARQL query to set a default namespace:

```
PREFIX : <http://xmlns.com/foaf/0.1/>
```

## 2.3 Persistence

RDFLib provides an *abstracted Store API* for persistence of RDF and Notation 3. The *Graph* class works with instances of this API (as the first argument to its constructor) for triple-based management of an RDF store including: garbage collection, transaction management, update, pattern matching, removal, length, and database management (`open()` / `close()` / `destroy()`).

Additional persistence mechanisms can be supported by implementing this API for a different store.

### 2.3.1 Stores currently shipped with core RDFLib

- *Memory* - not persistent!
- *BerkeleyDB* - on disk persistence via Python's *berkeleydb* package
- *SPARQLStore* - a read-only wrapper around a remote SPARQL Query endpoint
- *SPARQLUpdateStore* - a read-write wrapper around a remote SPARQL query/update endpoint pair

### 2.3.2 Usage

In most cases, passing the name of the store to the *Graph* constructor is enough:

```
from rdflib import Graph

graph = Graph(store='BerkeleyDB')
```

Most stores offering on-disk persistence will need to be opened before reading or writing. When persisting a triplestore, rather than a *ConjunctiveGraph* quadstore, you need to specify an identifier with which you can open the graph:

```
graph = Graph('BerkeleyDB', identifier='mygraph')

# first time create the store:
graph.open('/home/user/data/myRDFLibStore', create=True)

# work with the graph:
data = """
    PREFIX : <https://example.org/>
```

(continues on next page)

(continued from previous page)

```
    :a :b :c .
    :d :e :f .
    :d :g :h .
    """"
graph.parse(data=data, format="ttl")

# when done!
graph.close()
```

When done, `close()` must be called to free the resources associated with the store.

### 2.3.3 Additional store plugins

More store implementations are available in RDFLib extension projects:

- `rdflib-sqlalchemy`, which supports stored on a wide-variety of RDBMs backends,
- `rdflib-leveldb` - a store on to of Google's `LevelDB` key-value store.
- `rdflib-kyotocabinet` - a store on to of the `Kyoto Cabinet` key-value store.

### 2.3.4 Example

- `examples.berkeleydb_example` contains an example for using a BerkeleyDB store.
- `examples.sparqlstore_example` contains an example for using a SPARQLStore.

## 2.4 Merging graphs

Graphs share blank nodes only if they are derived from graphs described by documents or other structures (such as an RDF dataset) that explicitly provide for the sharing of blank nodes between different RDF graphs. Simply downloading a web document does not mean that the blank nodes in a resulting RDF graph are the same as the blank nodes coming from other downloads of the same document or from the same RDF source.

RDF applications which manipulate concrete syntaxes for RDF which use blank node identifiers should take care to keep track of the identity of the blank nodes they identify. Blank node identifiers often have a local scope, so when RDF from different sources is combined, identifiers may have to be changed in order to avoid accidental conflation of distinct blank nodes.

For example, two documents may both use the blank node identifier “\_:x” to identify a blank node, but unless these documents are in a shared identifier scope or are derived from a common source, the occurrences of “\_:x” in one document will identify a different blank node than the one in the graph described by the other document. When graphs are formed by combining RDF from multiple sources, it may be necessary to standardize apart the blank node identifiers by replacing them by others which do not occur in the other document(s).

*(copied directly from <https://www.w3.org/TR/rdf11-mt/#shared-blank-nodes-unions-and-merges>)*

In RDFLib, blank nodes are given unique IDs when parsing, so graph merging can be done by simply reading several files into the same graph:

---

```
from rdflib import Graph
```

```
graph = Graph()
```

```
graph.parse(input1)
```

```
graph.parse(input2)
```

graph now contains the merged graph of input1 and input2.

---

**Note:** However, the set-theoretic graph operations in RDFLib are assumed to be performed in sub-graphs of some larger data-base (for instance, in the context of a *ConjunctiveGraph*) and assume shared blank node IDs, and therefore do NOT do *correct* merging, i.e.:

```
from rdflib import Graph
```

```
g1 = Graph()
```

```
g1.parse(input1)
```

```
g2 = Graph()
```

```
g2.parse(input2)
```

```
graph = g1 + g2
```

May cause unwanted collisions of blank-nodes in graph.

---

## 2.5 Upgrading 5.0.0 to 6.0.0

6.0.0 fully adopts Python 3 practices and drops Python 2 support so it is neater, faster and generally more modern than 5.0.0. It also tidies up the Graph API (removing duplicate functions) so it does include a few breaking changes. Additionally, there is a long list of PRs merged into 6.0.0 adding a number of small fixes and features which are listed below.

RDFLib version 5.0.0 was released in 2020, 3 years after the previous version (4.2.2) and is fundamentally 5.0.0 compatible with. If you need very long-term backwards-compatibility or Python 2 support, you need 5.0.0.

### 2.5.1 Major Changes

The most notable changes in RDFLib 6.0.0 are:

#### Python 3.7+

- The oldest version of python you can use to run RDFLib is now 3.7.
- This is a big jump from RDFLib 5.0.0 that worked on python 2.7 and 3.5.
- This change is to allow the library maintainers to adopt more modern development tools, newer language features, and avoid the need to support EOL versions of python in the future

## JSON-LD integration and JSON-LD 1.1

- The json-ld serializer/parser plugin was by far the most commonly used RDFLib addon.
- Last year we brought it under the RDFLib org in Github
- Now for 6.0.0 release the JSON-LD serializer and parser are integrated into RDFLib core
- This includes the experimental support for the JSON-LD v1.1 spec
- You no longer need to install the json-ld dependency separately.

## 2.5.2 All Changes

This list has been assembled from Pull Request and commit information.

### General Bugs Fixed:

- Pr 451 redux [PR #978](#)

### Enhanced Features:

- Register additional serializer plugins for SPARQL mime types. [PR #987](#)

### SPARQL Fixes:

- Total order patch patch [PR #862](#)

### Code Quality and Cleanups:

- a slightly opinionated autopep8 run [PR #870](#)

### Testing:

- 3.7 for travis [PR #864](#)

### Documentation Fixes:

- Fix a doc string in the query module [PR #976](#)

### Integrate JSON-LD into RDFLib:

[PR #1354](#)

## 2.6 Upgrading 4.2.2 to 5.0.0

RDFLib version 5.0.0 appeared over 3 years after the previous release, 4.2.2 and contains a large number of both enhancements and bug fixes. Fundamentally though, 5.0.0 is compatible with 4.2.2.

### 2.6.1 Major Changes

#### Literal Ordering

Literal total ordering [PR #793](#) is implemented. That means all literals can now be compared to be greater than or less than any other literal. This is required for implementing some specific SPARQL features, but it is counter-intuitive to those who are expecting a `TypeError` when certain normally-incompatible types are compared. For example, comparing a `Literal(int(1), datatype=xsd:integer)` to `Literal(datetime.date(10,01,2020), datatype=xsd:date)` using a `>` or `<` operator in rdflib 4.2.2 and earlier, would normally throw a `TypeError`, however in rdflib 5.0.0 this operation now returns a `True` or `False` according to the Literal Total Ordering according the rules outlined in [PR #793](#)

#### Removed RDF Parsers

The RDFa and Microdata format RDF parsers were removed from rdflib. There are still other python libraries available to implement these parsers.

### 2.6.2 All Changes

This list has been assembled from Pull Request and commit information.

#### General Bugs Fixed:

- Pr 451 redux [PR #978](#)
- NTriples fails to parse URIs with only a scheme [ISSUE #920 PR #974](#)
- cannot clone it on windows - Remove colons from test result files. Fix #901. [ISSUE #901 PR #971](#)
- Add requirement for requests to setup.py [PR #969](#)
- fixed URIRef including native unicode characters [PR #961](#)
- DCTERMS.format not working [ISSUE #932](#)
- infixowl.manchesterSyntax do not encode strings [PR #906](#)
- Fix blank node label to not contain '.\_:' during parsing [PR #886](#)
- rename new SPARQLWrapper to SPARQLConnector [PR #872](#)
- Fix #859. Unquote and Uriquote Literal Datatype. [PR #860](#)
- Parsing nquads [ISSUE #786](#)
- ntriples spec allows for upper-cased lang tag, fixes #782 [PR #784](#)
- Error parsing N-Triple file using RDFLib [ISSUE #782](#)
- Adds escaped single quote to literal parser [PR #736](#)
- N3 parse error on single quote within single quotes [ISSUE #732](#)

- Fixed #725 [PR #730](#)
- test for issue #725: canonicalization collapses BNodes [PR #726](#)
- RGDA1 graph canonicalization sometimes still collapses distinct BNodes [ISSUE #725](#)
- Accept header should use a q parameter [PR #720](#)
- Added test for Issue #682 and fixed. [PR #718](#)
- Incompatibility with Python3: unichr [ISSUE #687](#)
- namespace.py include colon in ALLOWED\_NAME\_CHARS [PR #663](#)
- namespace.py fix compute\_qname missing namespaces [PR #649](#)
- RDFa parsing Error! `__init__()` got an unexpected keyword argument 'encoding' [ISSUE #639](#)
- Bugfix: `term.Literal.__add__` [PR #451](#)
- fixup of #443 [PR #445](#)
- Microdata to rdf second edition bak [PR #444](#)

### Enhanced Features:

- Register additional serializer plugins for SPARQL mime types. [PR #987](#)
- Pr 388 redux [PR #979](#)
- Allows RDF terms introduced by JSON-LD 1.1 [PR #970](#)
- make SPARQLConnector work with DBpedia [PR #941](#)
- ClosedNamespace returns right exception for way of access [PR #866](#)
- Not adding all namespaces for n3 serializer [PR #832](#)
- Adds basic support of xsd:duration [PR #808](#)
- Add possibility to set authority and basepath to skolemize graph [PR #807](#)
- Change notation3 list realization to non-recursive function. [PR #805](#)
- Suppress warning for not using custom encoding. [PR #800](#)
- Add support to parsing large xml inputs [ISSUE #749](#) [PR #750](#)
- improve hash efficiency by directly using str/unicode hash [PR #746](#)
- Added the csvw prefix to the RDFa initial context. [PR #594](#)
- syncing changes from pyMicrodata [PR #587](#)
- Microdata parser: updated the parser to the latest version of the microdata->rdf note (published in December 2014) [PR #443](#)
- `Literal.toPython()` support for xsd:hexBinary [PR #388](#)

**SPARQL Fixes:**

- Total order patch patch [PR #862](#)
- use <=< instead of deprecated << [PR #861](#)
- Fix #847 [PR #856](#)
- RDF Literal “1”^^xsd:boolean should \_not\_ coerce to True [ISSUE #847](#)
- Makes NOW() return an UTC date [PR #844](#)
- NOW() SPARQL should return an xsd:dateTime with a timezone [ISSUE #843](#)
- fix property paths bug: issue #715 [PR #822](#) [ISSUE #715](#)
- MulPath: correct behaviour of n3() [PR #820](#)
- Literal total ordering [PR #793](#)
- Remove SPARQLWrapper dependency [PR #744](#)
- made UNION faster by not preventing duplicates [PR #741](#)
- added a hook to add custom functions to SPARQL [PR #723](#)
- Issue714 [PR #717](#)
- Use <=< instead of deprecated << in SPARQL parser [PR #417](#)
- Custom FILTER function for SPARQL engine [ISSUE #274](#)

**Code Quality and Cleanups:**

- a slightly opinionated autopep8 run [PR #870](#)
- remove rdfa and microdata parsers from core RDFLib [PR #828](#)
- ClosedNamespace KeyError -> AttributeError [PR #827](#)
- typo in rdflib/plugins/sparql/update.py [ISSUE #760](#)
- Fix logging in interactive mode [PR #731](#)
- make namespace module flake8-compliant, change exceptions in that mod... [PR #711](#)
- delete ez\_setup.py? [ISSUE #669](#)
- code duplication issue between rdflib and pymicrodata [ISSUE #582](#)
- Transition from 2to3 to use of six.py to be merged in 5.0.0-dev [PR #519](#)
- sparqlstore drop deprecated methods and args [PR #516](#)
- python3 code seems shockingly inefficient [ISSUE #440](#)
- removed md5\_term\_hash, fixes #240 [PR #439](#) [ISSUE #240](#)

### Testing:

- 3.7 for travis [PR #864](#)
- Added trig unit tests to highlight some current parsing/serializing issues [PR #431](#)

### Documentation Fixes:

- Fix a doc string in the query module [PR #976](#)
- setup.py: Make the license field use an SPDX identifier [PR #789](#)
- Update README.md [PR #764](#)
- Update namespaces\_and\_bindings.rst [PR #757](#)
- DOC: README.md: rdflib-jsonld, https uris [PR #712](#)
- make doctest support py2/py3 [ISSUE #707](#)
- `pip install rdflib` (as per README.md) gets OSError on Mint 18.1 [ISSUE #704](#) [PR #717](#)
- Use `<=>` instead of deprecated `<<` in SPARQL parser [PR #417](#)
- Custom FILTER function for SPARQL engine [ISSUE #274](#)

## 2.7 Security Considerations

RDFLib is designed to access arbitrary network and file resources, in some cases these are directly requested resources, in other cases they are indirectly referenced resources.

An example of where indirect resources are access is JSON-LD processing, where network or file resources referenced by `@context` values will be loaded and processed.

RDFLib also supports SPARQL, which has federated query capabilities that allow queries to query arbitrary remote endpoints.

If you are using RDFLib to process untrusted documents or queries you should take measures to restrict file and network access.

Some measures that can be taken to restrict file and network access are:

- *Operating System Security Measures.*
- *Python Runtime Audit Hooks.*
- *Custom URL Openers.*

Of these, operating system security measures are recommended. The other measures work, but they are not as effective as operating system security measures, and even if they are used they should be used in conjunction with operating system security measures.



## 2.7.1 Operating System Security Measures

Most operating systems provide functionality that can be used to restrict network and file access of a process.

Some examples of these include:

- [Open Container Initiative \(OCI\) Containers](#) (aka Docker containers).

Most OCI runtimes provide mechanisms to restrict network and file access of containers. For example, using Docker, you can limit your container to only being access files explicitly mapped into the container and only access the network through a firewall. For more information refer to the documentation of the tool you use to manage your OCI containers:

- [Kubernetes](#)
- [Docker](#)
- [Podman](#)

- [firejail](#) can be used to sandbox a process on Linux and restrict its network and file access.
- File and network access restrictions.

Most operating systems provide a way to restrict operating system users to only being able to access files and network resources that are explicitly allowed. Applications that process untrusted input could be run as a user with these restrictions in place.

Many other measures are available, however, listing them outside the scope of this document.

Of the listed measures OCI containers are recommended. In most cases, OCI containers are constrained by default and can't access the loopback interface and can only access files that are explicitly mapped into the container.

## 2.7.2 Python Runtime Audit Hooks

From Python 3.8 onwards, Python provides a mechanism to install runtime audit hooks that can be used to limit access to files and network resources.

The runtime audit hook system is described in more detail in [PEP 578 – Python Runtime Audit Hooks](#).

Runtime audit hooks can be installed using the [sys.addaudithook](#) function, and will then get called when audit events occur. The audit events raised by the Python runtime and standard library are described in Python's [audit events table](#).

RDFLib uses [urllib.request.urlopen](#) for HTTP, HTTPS and other network access, and this function raises a [urllib.Request](#) audit event. For file access, RDFLib uses [open](#), which raises an [open](#) audit event.

Users of RDFLib can install audit hooks that react to these audit events and raises an exception when an attempt is made to access files or network resources that are not explicitly allowed.

RDFLib's test suite includes tests which verify that audit hooks can block access to network and file resources.

RDFLib also includes an example that shows how runtime audit hooks can be used to restrict network and file access in [secure\\_with\\_audit](#).

### 2.7.3 Custom URL Openers

RDFLib uses the `urllib.request.urlopen` for HTTP, HTTPS and other network access. This function will use a `urllib.request.OpenerDirector` installed with `urllib.request.install_opener` to open the URLs.

Users of RDFLib can install a custom URL opener that raise an exception when an attempt is made to access network resources that are not explicitly allowed.

RDFLib's test suite includes tests which verify that custom URL openers can be used to block access to network resources.

RDFLib also includes an example that shows how a custom opener can be used to restrict network access in *[secure\\_with\\_urlopen](#)*.

## REFERENCE

The nitty-gritty details of everything.

API reference:

### 3.1 rdflib

#### 3.1.1 rdflib package

##### Subpackages

##### rdflib.extras package

##### Submodules

##### rdflib.extras.cmdlineutils module

`rdflib.extras.cmdlineutils.main(target, _help=<function _help>, options="", stdin=True)`

A main function for tools that read RDF from files given on commandline or from STDIN (if stdin parameter is true)

##### rdflib.extras.describer module

A Descriptor is a stateful utility for creating RDF statements in a semi-declarative manner. It has methods for creating literal values, rel and rev resource relations (somewhat resembling RDFa).

The *Descriptor.rel* and *Descriptor.rev* methods return a context manager which sets the current about to the referenced resource for the context scope (for use with the `with` statement).

Full example in the `to_rdf` method below:

```
>>> import datetime
>>> from rdflib.graph import Graph
>>> from rdflib.namespace import Namespace, RDFS, FOAF
>>>
>>> ORG_URI = "http://example.org/"
>>>
>>> CV = Namespace("http://purl.org/captsolo/resume-rdf/0.2/cv#")
>>>
```

(continues on next page)

(continued from previous page)

```

>>> class Person(object):
...     def __init__(self):
...         self.first_name = u"Some"
...         self.last_name = u"Body"
...         self.username = "some1"
...         self.presentation = u"Just a Python & RDF hacker."
...         self.image = "/images/persons/" + self.username + ".jpg"
...         self.site = "http://example.net/"
...         self.start_date = datetime.date(2009, 9, 4)
...     def get_full_name(self):
...         return u" ".join([self.first_name, self.last_name])
...     def get_absolute_url(self):
...         return "/persons/" + self.username
...     def get_thumbnail_url(self):
...         return self.image.replace('.jpg', '-thumb.jpg')
...
...     def to_rdf(self):
...         graph = Graph()
...         graph.bind('foaf', FOAF)
...         graph.bind('cv', CV)
...         lang = 'en'
...         d = Describer(graph, base=ORG_URI)
...         d.about(self.get_absolute_url()+'#person')
...         d.rdftype(FOAF.Person)
...         d.value(FOAF.name, self.get_full_name())
...         d.value(FOAF.givenName, self.first_name)
...         d.value(FOAF.familyName, self.last_name)
...         d.rel(FOAF.homepage, self.site)
...         d.value(RDFS.comment, self.presentation, lang=lang)
...         with d.rel(FOAF.depiction, self.image):
...             d.rdftype(FOAF.Image)
...             d.rel(FOAF.thumbnail, self.get_thumbnail_url())
...         with d.rev(CV.aboutPerson):
...             d.rdftype(CV.CV)
...             with d.rel(CV.hasWorkHistory):
...                 d.value(CV.startDate, self.start_date)
...                 d.rel(CV.employedIn, ORG_URI+"#company")
...         return graph
...
>>> person_graph = Person().to_rdf()
>>> expected = Graph().parse(data='''<?xml version="1.0" encoding="utf-8"?>
... <rdf:RDF
...   xmlns:foaf="http://xmlns.com/foaf/0.1/"
...   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
...   xmlns:cv="http://purl.org/captsolo/resume-rdf/0.2/cv#"
...   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
...   <foaf:Person rdf:about="http://example.org/persons/some1#person">
...     <foaf:name>Some Body</foaf:name>
...     <foaf:givenName>Some</foaf:givenName>
...     <foaf:familyName>Body</foaf:familyName>
...     <foaf:depiction>
...       <foaf:Image

```

(continues on next page)

(continued from previous page)

```

...     rdf:about=
...         "http://example.org/images/persons/some1.jpg">
...     <foaf:thumbnail
...     rdf:resource=
...         "http://example.org/images/persons/some1-thumb.jpg"/>
...     </foaf:Image>
... </foaf:depiction>
... <rdfs:comment xml:lang="en">
...     Just a Python & RDF hacker.
... </rdfs:comment>
... <foaf:homepage rdf:resource="http://example.net/" />
... </foaf:Person>
... <cv:CV>
...     <cv:aboutPerson
...         rdf:resource="http://example.org/persons/some1#person">
...     </cv:aboutPerson>
...     <cv:hasWorkHistory>
...         <rdf:Description>
...             <cv:startDate
...                 rdf:datatype="http://www.w3.org/2001/XMLSchema#date"
...                 >2009-09-04</cv:startDate>
...             <cv:employedIn rdf:resource="http://example.org/#company"/>
...         </rdf:Description>
...     </cv:hasWorkHistory>
... </cv:CV>
... </rdf:RDF>
... ''' , format="xml")
>>>
>>> from rdflib.compare import isomorphic
>>> isomorphic(person_graph, expected)
True

```

```
class rdflib.extras.describer.Describer(graph=None, about=None, base=None)
```

Bases: `object`

```
__dict__ = mappingproxy({'__module__': 'rdflib.extras.describer', '__init__':
<function Describer.__init__>, 'about': <function Describer.about>, 'value':
<function Describer.value>, 'rel': <function Describer.rel>, 'rev': <function
Describer.rev>, 'rdftype': <function Describer.rdftype>, '_current': <function
Describer._current>, '_subject_stack': <function Describer._subject_stack>,
'__dict__': <attribute '__dict__' of 'Describer' objects>, '__weakref__':
<attribute '__weakref__' of 'Describer' objects>, '__doc__': None,
'__annotations__': {}})
```

```
__init__(graph=None, about=None, base=None)
```

```
__module__ = 'rdflib.extras.describer'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
about(subject, **kws)
```

Sets the current subject. Will convert the given object into an URIRef if it's not an Identifier.

Usage:

```

>>> d = Describer()
>>> d._current()
rdflib.term.BNode(...)
>>> d.about("http://example.org/")
>>> d._current()
rdflib.term.URIRef(u'http://example.org/')

```

**rdftype(*t*)**

Shorthand for setting rdf:type of the current subject.

Usage:

```

>>> from rdflib import URIRef
>>> from rdflib.namespace import RDF, RDFS
>>> d = Describer(about="http://example.org/")
>>> d.rdftype(RDFS.Resource)
>>> (URIRef('http://example.org/'),
...   RDF.type, RDFS.Resource) in d.graph
True

```

**rel(*p*, *o*=None, \*\**kws*)**

Set an object for the given property. Will convert the given object into an URIRef if it's not an Identifier. If none is given, a new BNode is used.

Returns a context manager for use in a with block, within which the given object is used as current subject.

Usage:

```

>>> from rdflib import URIRef
>>> from rdflib.namespace import RDF, RDFS
>>> d = Describer(about="/", base="http://example.org/")
>>> _ctxt = d.rel(RDFS.seeAlso, "/about")
>>> d.graph.value(URIRef('http://example.org/'), RDFS.seeAlso)
rdflib.term.URIRef(u'http://example.org/about')

>>> with d.rel(RDFS.seeAlso, "/more"):
...     d.value(RDFS.label, "More")
>>> (URIRef('http://example.org/'), RDFS.seeAlso,
...   URIRef('http://example.org/more')) in d.graph
True
>>> d.graph.value(URIRef('http://example.org/more'), RDFS.label)
rdflib.term.Literal(u'More')

```

**rev(*p*, *s*=None, \*\**kws*)**

Same as rel, but uses current subject as *object* of the relation. The given resource is still used as subject in the returned context manager.

Usage:

```

>>> from rdflib import URIRef
>>> from rdflib.namespace import RDF, RDFS
>>> d = Describer(about="http://example.org/")
>>> with d.rev(RDFS.seeAlso, "http://example.net/"):
...     d.value(RDFS.label, "Net")
>>> (URIRef('http://example.net/'), RDFS.seeAlso,

```

(continues on next page)

(continued from previous page)

```

...         URIRef('http://example.org/')) in d.graph
True
>>> d.graph.value(URIRef('http://example.net/'), RDFS.label)
rdflib.term.Literal(u'Net')

```

**value**(*p*, *v*, *\*\*kws*)

Set a literal value for the given property. Will cast the value to an `Literal` if a plain literal is given.

Usage:

```

>>> from rdflib import URIRef
>>> from rdflib.namespace import RDF, RDFS
>>> d = Descriptor(about="http://example.org/")
>>> d.value(RDFS.label, "Example")
>>> d.graph.value(URIRef('http://example.org/'), RDFS.label)
rdflib.term.Literal(u'Example')

```

`rdflib.extras.describer.cast_identifier`(*ref*, *\*\*kws*)

`rdflib.extras.describer.cast_value`(*v*, *\*\*kws*)

### rdflib.extras.external\_graph\_libs module

`rdflib.extras.external_graph_libs.rdflib_to_graphtool`(*graph*, *v\_prop\_names*=['term'],  
*e\_prop\_names*=['term'],  
*transform\_s*=<function <lambda>>,<br/>
*transform\_p*=<function <lambda>>,<br/>
*transform\_o*=<function <lambda>>)

Converts the given graph into a `graph_tool.Graph()`.

The subjects and objects are the later vertices of the Graph. The predicates become edges.

#### Parameters

- *graph*: a `rdflib.Graph`.
- *v\_prop\_names*: a list of names for the vertex properties. The default is set to ['term'] (see *transform\_s*, *transform\_o* below).
- *e\_prop\_names*: a list of names for the edge properties.
- *transform\_s*: callable with *s*, *p*, *o* input. Should return a dictionary containing a value for each name in *v\_prop\_names*. By default is set to {'term': *s*} which in combination with *v\_prop\_names* = ['term'] adds *s* as 'term' property to the generated vertex for *s*.
- *transform\_p*: similar to *transform\_s*, but wrt. *e\_prop\_names*. By default returns {'term': *p*} which adds *p* as a property to the generated edge between the vertex for *s* and the vertex for *o*.
- *transform\_o*: similar to *transform\_s*.

Returns: `graph_tool.Graph()`

```

>>> from rdflib import Graph, URIRef, Literal
>>> g = Graph()
>>> a, b, l = URIRef('a'), URIRef('b'), Literal('l')

```

(continues on next page)

(continued from previous page)

```

>>> p, q = URIRef('p'), URIRef('q')
>>> edges = [(a, p, b), (a, q, b), (b, p, a), (b, p, l)]
>>> for t in edges:
...     g.add(t)
...
>>> mdg = rdflib_to_graphtool(g)
>>> len(list(mdg.edges()))
4
>>> from graph_tool import util as gt_util
>>> vpterm = mdg.vertex_properties['term']
>>> va = gt_util.find_vertex(mdg, vpterm, a)[0]
>>> vb = gt_util.find_vertex(mdg, vpterm, b)[0]
>>> vl = gt_util.find_vertex(mdg, vpterm, l)[0]
>>> (va, vb) in [(e.source(), e.target()) for e in list(mdg.edges())]
True
>>> epterm = mdg.edge_properties['term']
>>> len(list(gt_util.find_edge(mdg, epterm, p))) == 3
True
>>> len(list(gt_util.find_edge(mdg, epterm, q))) == 1
True

```

```

>>> mdg = rdflib_to_graphtool(
...     g,
...     e_prop_names=[str('name')],
...     transform_p=lambda s, p, o: {str('name'): unicode(p)})
>>> epterm = mdg.edge_properties['name']
>>> len(list(gt_util.find_edge(mdg, epterm, unicode(p)))) == 3
True
>>> len(list(gt_util.find_edge(mdg, epterm, unicode(q)))) == 1
True

```

### Parameters

- **graph** (*Graph*) –
- **v\_prop\_names** (*List[str]*) –
- **e\_prop\_names** (*List[str]*) –

`rdflib.extras.external_graph_libs.rdflib_to_networkx_digraph(graph, calc_weights=True, edge_attrs=<function <lambda>>, **kws)`

Converts the given graph into a `networkx.DiGraph`.

As an `rdflib.Graph()` can contain multiple edges between nodes, by default adds the a ‘triples’ attribute to the single `DiGraph` edge with a list of all triples between `s` and `o`. Also by default calculates the edge weight as the length of triples.

### Parameters

- **graph**: a `rdflib.Graph`.
- **calc\_weights**: If true calculate multi-graph edge-count as edge ‘weight’



- **edge\_attrs:** Callable to construct later edge attributes. It receives 3 variables (s, p, o) and should construct a dictionary that is passed to networkx's `add_edge(s, o, **attrs)` function.

By default this will include setting the 'triples' attribute here, which is treated specially by us to be merged. Other attributes of multi-edges will only contain the attributes of the first edge. If you don't want the 'triples' attribute for tracking, set this to `lambda s, p, o: {}`.

Returns: `networkx.DiGraph`

```
>>> from rdflib import Graph, URIRef, Literal
>>> g = Graph()
>>> a, b, l = URIRef('a'), URIRef('b'), Literal('1')
>>> p, q = URIRef('p'), URIRef('q')
>>> edges = [(a, p, b), (a, q, b), (b, p, a), (b, p, l)]
>>> for t in edges:
...     g.add(t)
...
>>> dg = rdflib_to_networkx_digraph(g)
>>> dg[a][b]['weight']
2
>>> sorted(dg[a][b]['triples']) == [(a, p, b), (a, q, b)]
True
>>> len(dg.edges())
3
>>> dg.size()
3
>>> dg.size(weight='weight')
4.0
```

```
>>> dg = rdflib_to_networkx_graph(g, False, edge_attrs=lambda s,p,o: {})
>>> 'weight' in dg[a][b]
False
>>> 'triples' in dg[a][b]
False
```

### Parameters

- **graph** (*Graph*) –
- **calc\_weights** (*bool*) –

`rdflib.extras.external_graph_libs.rdflib_to_networkx_graph(graph, calc_weights=True, edge_attrs=<function <lambda>>, **kws)`

Converts the given graph into a `networkx.Graph`.

As an `rdflib.Graph()` can contain multiple directed edges between nodes, by default adds the a 'triples' attribute to the single `DiGraph` edge with a list of triples between `s` and `o` in graph. Also by default calculates the edge weight as the `len(triples)`.

### Parameters

- **graph**: a `rdflib.Graph`.
- **calc\_weights**: If true calculate multi-graph edge-count as edge 'weight'

- **edge\_attrs**: Callable to construct later edge\_attributes. It receives 3 variables (s, p, o) and should construct a dictionary that is passed to networkx's `add_edge(s, o, **attrs)` function.

By default this will include setting the 'triples' attribute here, which is treated specially by us to be merged. Other attributes of multi-edges will only contain the attributes of the first edge. If you don't want the 'triples' attribute for tracking, set this to `lambda s, p, o: {}`.

**Returns:**

`networkx.Graph`

```
>>> from rdflib import Graph, URIRef, Literal
>>> g = Graph()
>>> a, b, l = URIRef('a'), URIRef('b'), Literal('l')
>>> p, q = URIRef('p'), URIRef('q')
>>> edges = [(a, p, b), (a, q, b), (b, p, a), (b, p, l)]
>>> for t in edges:
...     g.add(t)
...
>>> ug = rdflib_to_networkx_graph(g)
>>> ug[a][b]['weight']
3
>>> sorted(ug[a][b]['triples']) == [(a, p, b), (a, q, b), (b, p, a)]
True
>>> len(ug.edges())
2
>>> ug.size()
2
>>> ug.size(weight='weight')
4.0
```

```
>>> ug = rdflib_to_networkx_graph(g, False, edge_attrs=lambda s,p,o:{})
>>> 'weight' in ug[a][b]
False
>>> 'triples' in ug[a][b]
False
```

**Parameters**

- **graph** (*Graph*) –
- **calc\_weights** (*bool*) –

`rdflib.extras.external_graph_libs.rdflib_to_networkx_multidigraph`(*graph*, *edge\_attrs*=<function <lambda>>, *\*\*kws*)

Converts the given graph into a `networkx.MultiDiGraph`.

The subjects and objects are the later nodes of the `MultiDiGraph`. The predicates are used as edge keys (to identify multi-edges).

**Parameters**

- **graph**: a `rdflib.Graph`.

- **edge\_attrs:** Callable to construct later edge\_attributes. It receives 3 variables (s, p, o) and should construct a dictionary that is passed to networkx's add\_edge(s, o, \*\*attrs) function.

By default this will include setting the MultiDiGraph key=p here. If you don't want to be able to re-identify the edge later on, you can set this to `lambda s, p, o: {}`. In this case MultiDiGraph's default (increasing ints) will be used.

#### Returns:

networkx.MultiDiGraph

```
>>> from rdflib import Graph, URIRef, Literal
>>> g = Graph()
>>> a, b, l = URIRef('a'), URIRef('b'), Literal('l')
>>> p, q = URIRef('p'), URIRef('q')
>>> edges = [(a, p, b), (a, q, b), (b, p, a), (b, p, l)]
>>> for t in edges:
...     g.add(t)
...
>>> mdg = rdflib_to_networkx_multidigraph(g)
>>> len(mdg.edges())
4
>>> mdg.has_edge(a, b)
True
>>> mdg.has_edge(a, b, key=p)
True
>>> mdg.has_edge(a, b, key=q)
True
```

```
>>> mdg = rdflib_to_networkx_multidigraph(g, edge_attrs=lambda s,p,o: {})
>>> mdg.has_edge(a, b, key=0)
True
>>> mdg.has_edge(a, b, key=1)
True
```

#### Parameters

**graph** (*Graph*) –

### rdflib.extras.infixowl module

RDFLib Python binding for OWL Abstract Syntax

see: <http://www.w3.org/TR/owl-semantic-syntax.html>  
[http://owl-workshop.man.ac.uk/acceptedLong/submission\\_9.pdf](http://owl-workshop.man.ac.uk/acceptedLong/submission_9.pdf)

#### 3.2.3 Axioms for complete classes without using owl:equivalentClass

Named class description of type 2 (with owl:oneOf) or type 4-6 (with owl:intersectionOf, owl:unionOf or owl:complementOf)

Uses Manchester Syntax for `__repr__`

```
>>> exNs = Namespace('http://example.com/')
>>> namespace_manager = NamespaceManager(Graph())
```

(continues on next page)

(continued from previous page)

```
>>> namespace_manager.bind('ex', exNs, override=False)
>>> namespace_manager.bind('owl', OWL, override=False)
>>> g = Graph()
>>> g.namespace_manager = namespace_manager
```

Now we have an empty graph, we can construct OWL classes in it using the Python classes defined in this module

```
>>> a = Class(exNs.Opera, graph=g)
```

Now we can assert `rdfs:subClassOf` and `owl:equivalentClass` relationships (in the underlying graph) with other classes using the ‘`subClassOf`’ and ‘`equivalentClass`’ descriptors which can be set to a list of objects for the corresponding predicates.

```
>>> a.subClassOf = [exNs.MusicalWork]
```

We can then access the `rdfs:subClassOf` relationships

```
>>> print(list(a.subClassOf))
[Class: ex:MusicalWork ]
```

This can also be used against already populated graphs:

```
>>> owlGraph = Graph().parse(str(OWL))
>>> namespace_manager.bind('owl', OWL, override=False)
>>> owlGraph.namespace_manager = namespace_manager
>>> list(Class(OWL.Class, graph=owlGraph).subClassOf)
[Class: rdfs:Class ]
```

Operators are also available. For instance we can add `ex:Opera` to the extension of the `ex:CreativeWork` class via the ‘`+=`’ operator

```
>>> a
Class: ex:Opera SubClassOf: ex:MusicalWork
>>> b = Class(exNs.CreativeWork, graph=g)
>>> b += a
>>> print(sorted(a.subClassOf, key=lambda c:c.identifier))
[Class: ex:CreativeWork , Class: ex:MusicalWork ]
```

And we can then remove it from the extension as well

```
>>> b -= a
>>> a
Class: ex:Opera SubClassOf: ex:MusicalWork
```

Boolean class constructions can also be created with Python operators. For example, The `|` operator can be used to construct a class consisting of a `owl:unionOf` the operands:

```
>>> c = a | b | Class(exNs.Work, graph=g)
>>> c
( ex:Opera OR ex:CreativeWork OR ex:Work )
```

Boolean class expressions can also be operated as lists (using python list operators)

```
>>> del c[c.index(Class(exNs.Work, graph=g))]
>>> c
( ex:Opera OR ex:CreativeWork )
```

The ‘&’ operator can be used to construct class intersection:

```
>>> woman = Class(exNs.Female, graph=g) & Class(exNs.Human, graph=g)
>>> woman.identifier = exNs.Woman
>>> woman
( ex:Female AND ex:Human )
>>> len(woman)
2
```

Enumerated classes can also be manipulated

```
>>> contList = [Class(exNs.Africa, graph=g), Class(exNs.NorthAmerica, graph=g)]
>>> EnumeratedClass(members=contList, graph=g)
{ ex:Africa ex:NorthAmerica }
```

owl:Restrictions can also be instantiated:

```
>>> Restriction(exNs.hasParent, graph=g, allValuesFrom=exNs.Human)
( ex:hasParent ONLY ex:Human )
```

Restrictions can also be created using Manchester OWL syntax in ‘colloquial’ Python >>> exNs.hasParent << some >> Class(exNs.Physician, graph=g) ( ex:hasParent SOME ex:Physician )

```
>>> Property(exNs.hasParent, graph=g) << max >> Literal(1)
( ex:hasParent MAX 1 )
```

```
>>> print(g.serialize(format='pretty-xml'))
```

rdflib.extras.infixowl.AllClasses(*graph*)

rdflib.extras.infixowl.AllDifferent(*members*)

TODO: implement this function

DisjointClasses(‘ description description { description } ‘)

rdflib.extras.infixowl.AllProperties(*graph*)

**class** rdflib.extras.infixowl.**AnnotatableTerms**(*identifier, graph=None, nameAnnotation=None, nameIsLabel=False*)

Bases: *Individual*

Terms in an OWL ontology with rdfs:label and rdfs:comment

## Interface with ATTEMPTO (<http://attempto.ifi.uzh.ch/site>)

### Verbalisation of OWL entity IRIS

#### How are OWL entity IRIs verbalized?

The OWL verbalizer maps OWL entity IRIs to ACE content words such that

- OWL individuals map to ACE proper names (PN)
- OWL classes map to ACE common nouns (CN)

- OWL properties map to ACE transitive verbs (TV)

There are 6 morphological categories that determine the surface form of an IRI:

- singular form of a proper name (e.g. John)
- singular form of a common noun (e.g. man)
- plural form of a common noun (e.g. men)
- singular form of a transitive verb (e.g. mans)
- plural form of a transitive verb (e.g. man)
- past participle form a transitive verb (e.g. manned)

The user has full control over the eventual surface forms of the IRIs but has to choose them in terms of the above categories. Furthermore,

- the surface forms must be legal ACE content words (e.g. they should not contain punctuation symbols);
- the mapping of IRIs to surface forms must be bidirectional within the same word class, in order to be able to (if needed) parse the verbalization back into OWL in a semantics preserving way.

### Using the lexicon

It is possible to specify the mapping of IRIs to surface forms using the following annotation properties:

```
http://attempto.ifi.uzh.ch/ace_lexicon#PN_sg
http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg
http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl
http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg
http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl
http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg
```

For example, the following axioms state that if the IRI “#man” is used as a plural common noun, then the wordform men must be used by the verbalizer. If, however, it is used as a singular transitive verb, then mans must be used.

```
<AnnotationAssertion>
  <AnnotationProperty IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
  <IRI>#man</IRI>
  <Literal datatypeIRI="&xsd:string">men</Literal>
</AnnotationAssertion>

<AnnotationAssertion>
  <AnnotationProperty IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
  <IRI>#man</IRI>
  <Literal datatypeIRI="&xsd:string">mans</Literal>
</AnnotationAssertion>
```

```
__init__(identifier, graph=None, nameAnnotation=None, nameIsLabel=False)
```

```
__module__ = 'rdflib.extras.infixowl'
```

```
property comment
```

```
handleAnnotation(val)
```

```
property label
```

**property** `seeAlso`

**setupACEAnnotations()**

```
class rdflib.extras.infixowl.BooleanClass(identifier=None, operator=rdflib.term.URIRef('http://www.w3.org/2002/07/owl#intersectionOf'), members=None, graph=None)
```

Bases: *OWLRDFListProxy, Class*

See: <http://www.w3.org/TR/owl-ref/#Boolean>

owl:complementOf is an attribute of Class, however

```
__init__(identifier=None, operator=rdflib.term.URIRef('http://www.w3.org/2002/07/owl#intersectionOf'), members=None, graph=None)
```

```
__module__ = 'rdflib.extras.infixowl'
```

```
__or__(other)
```

Adds other to the list and returns self

```
__repr__()
```

Returns the Manchester Syntax equivalent for this class

**changeOperator**(*newOperator*)

Converts a unionOf / intersectionOf class expression into one that instead uses the given operator

```
>>> testGraph = Graph()
>>> Individual.factoryGraph = testGraph
>>> EX = Namespace("http://example.com/")
>>> namespace_manager = NamespaceManager(Graph())
>>> namespace_manager.bind('ex', EX, override=False)
>>> testGraph.namespace_manager = namespace_manager
>>> fire = Class(EX.Fire)
>>> water = Class(EX.Water)
>>> testClass = BooleanClass(members=[fire,water])
>>> testClass
( ex:Fire AND ex:Water )
>>> testClass.changeOperator(OWL.unionOf)
>>> testClass
( ex:Fire OR ex:Water )
>>> try:
...     testClass.changeOperator(OWL.unionOf)
... except Exception as e:
...     print(e)
The new operator is already being used!
```

**copy()**

Create a copy of this class

**getIntersections** = <rdflib.extras.infixowl.Callable object>

**getUnions** = <rdflib.extras.infixowl.Callable object>

**isPrimitive()**

**serialize**(*graph*)

`class rdflib.extras.infixowl.Callable(anycallable)`

Bases: `object`

```
__dict__ = mappingproxy({'__module__': 'rdflib.extras.infixowl', '__init__':
<function Callable.__init__>, '__dict__': <attribute '__dict__' of 'Callable'
objects>, '__weakref__': <attribute '__weakref__' of 'Callable' objects>,
'__doc__': None, '__annotations__': {}})
```

```
__init__(anycallable)
```

```
__module__ = 'rdflib.extras.infixowl'
```

```
__weakref__
```

list of weak references to the object (if defined)

`rdflib.extras.infixowl.CastClass(c, graph=None)`

```
class rdflib.extras.infixowl.Class(identifier=None, subClassOf=None, equivalentClass=None,
disjointWith=None, complementOf=None, graph=None,
skipOWLClassMembership=False, comment=None,
nounAnnotations=None, nameAnnotation=None, nameIsLabel=False)
```

Bases: `AnnotatableTerms`

‘General form’ for classes:

The Manchester Syntax (supported in Protege) is used as the basis for the form of this class

See: [http://owl-workshop.man.ac.uk/acceptedLong/submission\\_9.pdf](http://owl-workshop.man.ac.uk/acceptedLong/submission_9.pdf):

[Annotation] ‘Class:’ classID { Annotation ( (‘SubClassOf:’ ClassExpression) | (‘EquivalentTo’ ClassExpression) | (‘DisjointWith’ ClassExpression)) }

Appropriate excerpts from OWL Reference:

**“.. Subclass axioms provide us with partial definitions: they represent**

necessary but not sufficient conditions for establishing class membership of an individual.”

“.. A class axiom may contain (multiple) owl:equivalentClass statements”

“..A class axiom may also contain (multiple) owl:disjointWith statements..”

**“..An owl:complementOf property links a class to precisely one class**

description.”

```
__and__(other)
```

Construct an anonymous class description consisting of the intersection of this class and ‘other’ and return it

```
>>> exNs = Namespace('http://example.com/')
>>> namespace_manager = NamespaceManager(Graph())
>>> namespace_manager.bind('ex', exNs, override=False)
>>> namespace_manager.bind('owl', OWL, override=False)
>>> g = Graph()
>>> g.namespace_manager = namespace_manager
```

Chaining 3 intersections

```
>>> female = Class(exNs.Female, graph=g)
>>> human = Class(exNs.Human, graph=g)
```

(continues on next page)



(continued from previous page)

```

>>> youngPerson = Class(exNs.YoungPerson, graph=g)
>>> youngWoman = female & human & youngPerson
>>> youngWoman
ex:YoungPerson THAT ( ex:Female AND ex:Human )
>>> isinstance(youngWoman, BooleanClass)
True
>>> isinstance(youngWoman.identifier, BNode)
True

```

**\_\_eq\_\_**(*other*)

Return self==value.

**\_\_hash\_\_**()

```

>>> b = Class(OWL.Restriction)
>>> c = Class(OWL.Restriction)
>>> len(set([b,c]))
1

```

**\_\_iadd\_\_**(*other*)

**\_\_init\_\_**(*identifier=None, subClassOf=None, equivalentClass=None, disjointWith=None, complementOf=None, graph=None, skipOWLClassMembership=False, comment=None, nounAnnotations=None, nameAnnotation=None, nameIsLabel=False*)

**\_\_invert\_\_**()

Shorthand for Manchester syntax's not operator

**\_\_isub\_\_**(*other*)

**\_\_module\_\_** = 'rdflib.extras.infixowl'

**\_\_or\_\_**(*other*)

Construct an anonymous class description consisting of the union of this class and 'other' and return it

**\_\_repr\_\_**(*full=False, normalization=True*)

Returns the Manchester Syntax equivalent for this class

**property annotation**

**property complementOf**

**property disjointWith**

**property equivalentClass**

**property extent**

**property extentQuery**

**isPrimitive**()

**property parents**

computed attributes that returns a generator over taxonomic 'parents' by disjunction, conjunction, and subsumption

```

>>> from rdflib.util import first
>>> exNs = Namespace('http://example.com/')
>>> namespace_manager = NamespaceManager(Graph())
>>> namespace_manager.bind('ex', exNs, override=False)
>>> namespace_manager.bind('owl', OWL, override=False)
>>> g = Graph()
>>> g.namespace_manager = namespace_manager
>>> Individual.factoryGraph = g
>>> brother = Class(exNs.Brother)
>>> sister = Class(exNs.Sister)
>>> sibling = brother | sister
>>> sibling.identifier = exNs.Sibling
>>> sibling
( ex:Brother OR ex:Sister )
>>> first(brother.parents)
Class: ex:Sibling EquivalentTo: ( ex:Brother OR ex:Sister )
>>> parent = Class(exNs.Parent)
>>> male = Class(exNs.Male)
>>> father = parent & male
>>> father.identifier = exNs.Father
>>> list(father.parents)
[Class: ex:Parent , Class: ex:Male ]

```

**serialize**(*graph*)

**setupNounAnnotations**(*noun\_annotations*)

**property** **subClassOf**

**subSumpteeIds**()

**class** rdflib.extras.infixowl.**ClassNamespaceFactory**(*value: Union[str, bytes]*)

Bases: *Namespace*

**\_\_getattr\_\_**(*name*)

**\_\_getitem\_\_**(*key, default=None*)

Return self[key].

**\_\_module\_\_** = 'rdflib.extras.infixowl'

**term**(*name*)

rdflib.extras.infixowl.**CommonNSBindings**(*graph, additionalNS=None*)

Takes a graph and binds the common namespaces (rdf,rdfs, & owl)

rdflib.extras.infixowl.**ComponentTerms**(*cls*)

Takes a Class instance and returns a generator over the classes that are involved in its definition, ignoring unnamed classes

rdflib.extras.infixowl.**DeepClassClear**(*class\_to\_prune*)

Recursively clear the given class, continuing where any related class is an anonymous class

```

>>> EX = Namespace('http://example.com/')
>>> namespace_manager = NamespaceManager(Graph())

```

(continues on next page)

(continued from previous page)

```

>>> namespace_manager.bind('ex', EX, override=False)
>>> namespace_manager.bind('owl', OWL, override=False)
>>> g = Graph()
>>> g.namespace_manager = namespace_manager
>>> Individual.factoryGraph = g
>>> classB = Class(EX.B)
>>> classC = Class(EX.C)
>>> classD = Class(EX.D)
>>> classE = Class(EX.E)
>>> classF = Class(EX.F)
>>> anonClass = EX.someProp << some >> classD
>>> classF += anonClass
>>> list(anonClass.subClassOf)
[Class: ex:F ]
>>> classA = classE | classF | anonClass
>>> classB += classA
>>> classA.equivalentClass = [Class()]
>>> classB.subClassOf = [EX.someProp << some >> classC]
>>> classA
( ex:E OR ex:F OR ( ex:someProp SOME ex:D ) )
>>> DeepClassClear(classA)
>>> classA
( )
>>> list(anonClass.subClassOf)
[]
>>> classB
Class: ex:B SubClassOf: ( ex:someProp SOME ex:C )

```

```

>>> otherClass = classD | anonClass
>>> otherClass
( ex:D OR ( ex:someProp SOME ex:D ) )
>>> DeepClassClear(otherClass)
>>> otherClass
( )
>>> otherClass.delete()
>>> list(g.triples((otherClass.identifier, None, None)))
[]

```

**class** rdflib.extras.infixowl.**EnumeratedClass**(*identifier=None, members=None, graph=None*)

Bases: *OWLRDFListProxy, Class*

Class for owl:oneOf forms:

OWL Abstract Syntax is used

**axiom** ::= 'EnumeratedClass('

classID ['Deprecated'] { annotation } { individualID } ')'

```

>>> exNs = Namespace('http://example.com/')
>>> namespace_manager = NamespaceManager(Graph())
>>> namespace_manager.bind('ex', exNs, override=False)
>>> namespace_manager.bind('owl', OWL, override=False)
>>> g = Graph()

```

(continues on next page)

(continued from previous page)

```

>>> g.namespace_manager = namespace_manager
>>> Individual.factoryGraph = g
>>> ogbujiBros = EnumeratedClass(exNs.ogbujiBros,
...                             members=[exNs.chime,
...                                     exNs.uche,
...                                     exNs.ejike])
>>> ogbujiBros
{ ex:chime ex:uche ex:ejike }
>>> col = Collection(g, first(
...     g.objects(predicate=OWL.oneOf, subject=ogbujiBros.identifier)))
>>> sorted([g.qname(item) for item in col])
['ex:chime', 'ex:ejike', 'ex:uche']
>>> print(g.serialize(format='n3'))
@prefix ex: <http://example.com/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

ex:ogbujiBros a owl:Class;
    owl:oneOf ( ex:chime ex:uche ex:ejike ) .

```

```
__init__(identifier=None, members=None, graph=None)
```

```
__module__ = 'rdflib.extras.infixowl'
```

```
__repr__()
```

Returns the Manchester Syntax equivalent for this class

```
isPrimitive()
```

```
serialize(graph)
```

```
rdflib.extras.infixowl.GetIdentifiedClasses(graph)
```

```
class rdflib.extras.infixowl.Individual(identifier=None, graph=None)
```

Bases: `object`

A typed individual

```

__dict__ = mappingproxy({'__module__': 'rdflib.extras.infixowl', '__doc__': '\n A
typed individual\n ', 'factoryGraph': <Graph
identifier=N4779d5c5a6b445ce8ad3127f953ff4aa (<class 'rdflib.graph.Graph'>>),
'serialize': <function Individual.serialize>, '__init__': <function
Individual.__init__>, 'clearInDegree': <function Individual.clearInDegree>,
'clearOutDegree': <function Individual.clearOutDegree>, 'delete': <function
Individual.delete>, 'replace': <function Individual.replace>, '_get_type':
<function Individual._get_type>, '_set_type': <function Individual._set_type>,
'_delete_type': <function TermDeletionHelper.__call__.<locals>._remover>, 'type':
<property object>, '_get_identifier': <function Individual._get_identifier>,
'_set_identifier': <function Individual._set_identifier>, 'identifier': <property
object>, '_get_sameAs': <function Individual._get_sameAs>, '_set_sameAs':
<function Individual._set_sameAs>, '_delete_sameAs': <function
TermDeletionHelper.__call__.<locals>._remover>, 'sameAs': <property object>,
'__dict__': <attribute '__dict__' of 'Individual' objects>, '__weakref__':
<attribute '__weakref__' of 'Individual' objects>, '__annotations__': {}})

```

```

__init__(identifier=None, graph=None)

__module__ = 'rdflib.extras.infixowl'

__weakref__
    list of weak references to the object (if defined)

clearInDegree()

clearOutDegree()

delete()

factoryGraph = <Graph identifier=N4779d5c5a6b445ce8ad3127f953ff4aa (<class
'rdflib.graph.Graph'>)>

property identifier

replace(other)

property sameAs

serialize(graph)

property type

class rdflib.extras.infixowl.Infix(function)
    Bases: object

    __call__(value1, value2)
        Call self as a function.

    __dict__ = mappingproxy({'__module__': 'rdflib.extras.infixowl', '__init__':
    <function Infix.__init__>, '__rlshift__': <function Infix.__rlshift__>,
    '__rshift__': <function Infix.__rshift__>, '__rmatmul__': <function
    Infix.__rmatmul__>, '__matmul__': <function Infix.__matmul__>, '__call__':
    <function Infix.__call__>, '__dict__': <attribute '__dict__' of 'Infix' objects>,
    '__weakref__': <attribute '__weakref__' of 'Infix' objects>, '__doc__': None,
    '__annotations__': {}})

    __init__(function)

    __matmul__(other)

    __module__ = 'rdflib.extras.infixowl'

    __rlshift__(other)

    __rmatmul__(other)

    __rshift__(other)

    __weakref__
        list of weak references to the object (if defined)

exception rdflib.extras.infixowl.MalformedClass
    Bases: ValueError

    Deprecated since version TODO-NEXT-VERSION: This class will be removed in version 7.0.0.

```

```
__module__ = 'rdflib.extras.infixowl'

__weakref__
    list of weak references to the object (if defined)
exception rdflib.extras.infixowl.MalformedClassError(msg)
    Bases: MalformedClass
    __init__(msg)
    __module__ = 'rdflib.extras.infixowl'
    __repr__()
        Return repr(self).
class rdflib.extras.infixowl.OWLRDFListProxy(rdf_list, members=None, graph=None)
    Bases: object
    __contains__(item)
    __delitem__(key)

    __dict__ = mappingproxy({'__module__': 'rdflib.extras.infixowl', '__init__':
    <function OWLRDFListProxy.__init__>, '__eq__': <function OWLRDFListProxy.__eq__>,
    '__len__': <function OWLRDFListProxy.__len__>, 'index': <function
    OWLRDFListProxy.index>, '__getitem__': <function OWLRDFListProxy.__getitem__>,
    '__setitem__': <function OWLRDFListProxy.__setitem__>, '__delitem__': <function
    OWLRDFListProxy.__delitem__>, 'clear': <function OWLRDFListProxy.clear>,
    '__iter__': <function OWLRDFListProxy.__iter__>, '__contains__': <function
    OWLRDFListProxy.__contains__>, 'append': <function OWLRDFListProxy.append>,
    '__iadd__': <function OWLRDFListProxy.__iadd__>, '__dict__': <attribute '__dict__'
    of 'OWLRDFListProxy' objects>, '__weakref__': <attribute '__weakref__' of
    'OWLRDFListProxy' objects>, '__doc__': None, '__hash__': None, '__annotations__':
    {}})

    __eq__(other)
        Equivalence of boolean class constructors is determined by equivalence of its members
    __getitem__(key)
    __hash__ = None
    __iadd__(other)
    __init__(rdf_list, members=None, graph=None)
    __iter__()
    __len__()
    __module__ = 'rdflib.extras.infixowl'
    __setitem__(key, value)
    __weakref__
        list of weak references to the object (if defined)
    append(item)
```

```

clear()

index(item)

class rdflib.extras.infixowl.Ontology(identifier=None, imports=None, comment=None, graph=None)
    Bases: AnnotatableTerms
    The owl ontology metadata
    __init__(identifier=None, imports=None, comment=None, graph=None)
    __module__ = 'rdflib.extras.infixowl'

    property imports
    setVersion(version)

class rdflib.extras.infixowl.Property(identifier=None, graph=None, base-
                                     Type=rdflib.term.URIRef('http://www.w3.org/2002/07/owl#ObjectProperty'),
                                     subPropertyOf=None, domain=None, range=None,
                                     inverseOf=None, otherType=None, equivalentProperty=None,
                                     comment=None, verbAnnotations=None, nameAnnotation=None,
                                     nameIsLabel=False)
    Bases: AnnotatableTerms
    axiom ::= 'DatatypeProperty(' datavaluedPropertyID ['Deprecated']
        { annotation } { 'super(' datavaluedPropertyID ')' } [ 'Functional' ] { 'domain(' description ')' } { 'range('
        dataRange ')' } ' )' | 'ObjectProperty(' individualvaluedPropertyID ['Deprecated'] { annotation } { 'super('
        individualvaluedPropertyID ')' } [ 'inverseOf(' individualvaluedPropertyID ')' ] [ 'Symmetric' ] [ 'Func-
        tional' | 'InverseFunctional' | 'Functional' 'InverseFunctional' | 'Transitive' ] { 'domain(' description ')' }
        { 'range(' description ')' } ' )'
    __init__(identifier=None, graph=None,
             baseType=rdflib.term.URIRef('http://www.w3.org/2002/07/owl#ObjectProperty'),
             subPropertyOf=None, domain=None, range=None, inverseOf=None, otherType=None,
             equivalentProperty=None, comment=None, verbAnnotations=None, nameAnnotation=None,
             nameIsLabel=False)

    __module__ = 'rdflib.extras.infixowl'

    __repr__()
        Return repr(self).

    property domain
    property extent
    property inverseOf
    property range
    replace(other)
    serialize(graph)
    setupVerbAnnotations(verb_annotations)
        OWL properties map to ACE transitive verbs (TV)
        There are 6 morphological categories that determine the surface form of an IRI:

```

singular form of a transitive verb (e.g. mans) plural form of a transitive verb (e.g. man) past participle form a transitive verb (e.g. manned)

[http://attempto.ifi.uzh.ch/ace\\_lexicon#TV\\_sg](http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg)      [http://attempto.ifi.uzh.ch/ace\\_lexicon#TV\\_pl](http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl)  
[http://attempto.ifi.uzh.ch/ace\\_lexicon#TV\\_vbg](http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg)

**property** `subPropertyOf`

```
class rdflib.extras.infixowl.Restriction(onProperty, graph=None, allValuesFrom=None,
                                         someValuesFrom=None, value=None, cardinality=None,
                                         maxCardinality=None, minCardinality=None,
                                         identifier=None)
```

Bases: *Class*

```
restriction ::= 'restriction(' datavaluedPropertyID dataRestrictionComponent { dataRestrictionComponent } ')'
| 'restriction(' individualvaluedPropertyID individualRestrictionComponent { individualRestrictionComponent } ')'
```

`__eq__`(*other*)

Equivalence of restrictions is determined by equivalence of the property in question and the restriction 'range'

`__hash__`()

```
>>> b = Class(OWL.Restriction)
>>> c = Class(OWL.Restriction)
>>> len(set([b,c]))
1
```

```
__init__(onProperty, graph=None, allValuesFrom=None, someValuesFrom=None, value=None,
         cardinality=None, maxCardinality=None, minCardinality=None, identifier=None)
```

```
__module__ = 'rdflib.extras.infixowl'
```

`__repr__`()

Returns the Manchester Syntax equivalent for this restriction

**property** `allValuesFrom`

**property** `cardinality`

**property** `hasValue`

`isPrimitive`()

**property** `maxCardinality`

**property** `minCardinality`

**property** `onProperty`

`restrictionKind`()

```
restrictionKinds =
[rdflib.term.URIRef('http://www.w3.org/2002/07/owl#allValuesFrom'),
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#someValuesFrom'),
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#hasValue'),
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#maxCardinality'),
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#minCardinality')]
```



`serialize(graph)`

```
>>> g1 = Graph()
>>> g2 = Graph()
>>> EX = Namespace("http://example.com/")
>>> namespace_manager = NamespaceManager(g1)
>>> namespace_manager.bind('ex', EX, override=False)
>>> namespace_manager = NamespaceManager(g2)
>>> namespace_manager.bind('ex', EX, override=False)
>>> Individual.factoryGraph = g1
>>> prop = Property(EX.someProp, baseType=OWL.DatatypeProperty)
>>> restr1 = (Property(
...     EX.someProp,
...     baseType=OWL.DatatypeProperty)) << some >> (Class(EX.Foo))
>>> restr1
( ex:someProp SOME ex:Foo )
>>> restr1.serialize(g2)
>>> Individual.factoryGraph = g2
>>> list(Property(
...     EX.someProp, baseType=None).type
... )
[rdflib.term.URIRef(
    'http://www.w3.org/2002/07/owl#DatatypeProperty')]
```

**property someValuesFrom**

`rdflib.extras.infixowl.classOrIdentifier(thing)`

`rdflib.extras.infixowl.classOrTerm(thing)`

`rdflib.extras.infixowl.generateQName(graph, uri)`

`rdflib.extras.infixowl.manchesterSyntax(thing, store, boolean=None, transientList=False)`

Core serialization thing is a Class and is processed as a subject store is an RDFLib Graph to be queried about thing

`rdflib.extras.infixowl.propertyOrIdentifier(thing)`

## Module contents

### rdflib.namespace package

## Module contents

### Namespace Utilities

RDFLib provides mechanisms for managing Namespaces.

In particular, there is a *Namespace* class that takes as its argument the base URI of the namespace.

```
>>> from rdflib.namespace import Namespace
>>> RDFS = Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#")
```

Fully qualified URIs in the namespace can be constructed either by attribute or by dictionary access on Namespace instances:

```
>>> RDFS.seeAlso
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#seeAlso')
>>> RDFS['seeAlso']
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#seeAlso')
```

### Automatic handling of unknown predicates

As a programming convenience, a namespace binding is automatically created when `rdflib.term.URIRef` predicates are added to the graph.

### Importable namespaces

The following namespaces are available by directly importing from `rdflib`:

- BRICK
- CSVW
- DC
- DCAT
- DCMITYPE
- DCTERMS
- DCAM
- DOAP
- FOAF
- ODRL2
- ORG
- OWL
- PROF
- PROV
- QB
- RDF
- RDFS
- SDO
- SH
- SKOS
- SOSA
- SSN
- TIME
- VANN

- VOID
- WGS
- XSD

```
>>> from rdflib.namespace import RDFS
>>> RDFS.seeAlso
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#seeAlso')
```

**class** rdflib.namespace.ClosedNamespace(uri: str, terms: List[str])

Bases: *Namespace*

A namespace with a closed list of members

Trying to create terms not listed is an error

**\_\_annotations\_\_** = {'\_ClosedNamespace\_\_uris': typing.Dict[str, rdflib.term.URIRef]}

**\_\_contains\_\_**(ref)

Allows to check if a URI is within (starts with) this Namespace.

```
>>> from rdflib import URIRef
>>> namespace = Namespace('http://example.org/')
>>> uri = URIRef('http://example.org/foo')
>>> uri in namespace
True
>>> person_class = namespace['Person']
>>> person_class in namespace
True
>>> obj = URIRef('http://not.example.org/bar')
>>> obj in namespace
False
```

#### Parameters

**ref** (str) –

#### Return type

bool

**\_\_dir\_\_**()

Default dir() implementation.

#### Return type

List[str]

**\_\_getattr\_\_**(name)

#### Parameters

**name** (str) –

#### Return type

URIRef

**\_\_getitem\_\_**(key)

Return self[key].

#### Parameters

**key** (str) –

**Return type**

*URIRef*

`__module__ = 'rdflib.namespace'`

`static __new__(cls, uri, terms)`

**Parameters**

- **uri** (*str*) –
- **terms** (*List[str]*) –

`__repr__()`

Return repr(self).

**Return type**

*str*

`term(name)`

**Parameters**

**name** (*str*) –

**Return type**

*URIRef*

`property uri: str`

`class rdflib.namespace.DefinedNamespace`

Bases: *object*

A Namespace with an enumerated list of members. Warnings are emitted if unknown members are referenced if `_warn` is True

`class rdflib.namespace.Namespace(value: Union[str, bytes])`

Bases: *str*

Utility class for quickly generating URIRefs with a common prefix

```
>>> from rdflib.namespace import Namespace
>>> n = Namespace("http://example.org/")
>>> n.Person # as attribute
rdflib.term.URIRef('http://example.org/Person')
>>> n['first-name'] # as item - for things that are not valid python identifiers
rdflib.term.URIRef('http://example.org/first-name')
>>> n.Person in n
True
>>> n2 = Namespace("http://example2.org/")
>>> n.Person in n2
False
```

`__annotations__ = {}`

`__contains__(ref)`

Allows to check if a URI is within (starts with) this Namespace.

```

>>> from rdflib import URIRef
>>> namespace = Namespace('http://example.org/')
>>> uri = URIRef('http://example.org/foo')
>>> uri in namespace
True
>>> person_class = namespace['Person']
>>> person_class in namespace
True
>>> obj = URIRef('http://not.example.org/bar')
>>> obj in namespace
False

```

**Parameters**

**ref** (*str*) –

**Return type**

*bool*

```

__dict__ = mappingproxy({'__module__': 'rdflib.namespace', '__doc__': '\n Utility
class for quickly generating URIRefs with a common prefix\n\n >>> from
rdflib.namespace import Namespace\n >>> n = Namespace("http://example.org/")\n >>>
n.Person # as attribute\n rdflib.term.URIRef(\'http://example.org/Person\')\n >>>
n[\'first-name\'] # as item - for things that are not valid python identifiers\n
rdflib.term.URIRef(\'http://example.org/first-name\')\n >>> n.Person in n\n True\n
>>> n2 = Namespace("http://example2.org/")\n >>> n.Person in n2\n False\n ',
'__new__': <staticmethod object>, 'title': <property object>, 'term': <function
Namespace.term>, '__getitem__': <function Namespace.__getitem__>, '__getattr__':
<function Namespace.__getattr__>, '__repr__': <function Namespace.__repr__>,
'__contains__': <function Namespace.__contains__>, '__dict__': <attribute
'__dict__' of 'Namespace' objects>, '__weakref__': <attribute '__weakref__' of
'Namespace' objects>, '__annotations__': {}})

```

**\_\_getattr\_\_** (*name*)

**Parameters**

**name** (*str*) –

**Return type**

*URIRef*

**\_\_getitem\_\_** (*key*)

Return self[key].

**Parameters**

**key** (*str*) –

**Return type**

*URIRef*

**\_\_module\_\_** = 'rdflib.namespace'

**static \_\_new\_\_** (*cls, value*)

**Parameters**

**value** (*Union[str, bytes]*) –

**Return type***Namespace***\_\_repr\_\_()**

Return repr(self).

**Return type***str***\_\_weakref\_\_**

list of weak references to the object (if defined)

**term(name)****Parameters****name** (*str*) –**Return type***URIRef***property title:** *URIRef*

Return a version of the string where each word is titlecased.

More specifically, words start with uppercased characters and all remaining cased characters have lower case.

**class** rdflib.namespace.**NamespaceManager**(*graph*, *bind\_namespaces*='rdflib')Bases: *object*

Class for managing prefix =&gt; namespace mappings

This class requires an RDFlib Graph as an input parameter and may optionally have the parameter *bind\_namespaces* set. This second parameter selects a strategy which is one of the following:

- **core:**
  - binds several core RDF prefixes only
  - owl, rdf, rdfs, xsd, xml from the NAMESPACE\_PREFIXES\_CORE object
- **rdflib:**
  - binds all the namespaces shipped with RDFLib as DefinedNamespace instances
  - all the core namespaces and all the following: brick, csvw, dc, dcat
  - dcmitype, dcterms, dcam, doap, foaf, geo, odrl, org, prof, prov, qb, schema
  - sh, skos, sosa, ssn, time, vann, void
  - see the NAMESPACE\_PREFIXES\_RDFLIB object for the up-to-date list
  - this is default
- **none:**
  - binds no namespaces to prefixes
  - note this is NOT default behaviour
- **cc:**
  - using prefix bindings from prefix.cc which is a online prefixes database
  - not implemented yet - this is aspirational

**Attention:** The namespaces bound for specific values of `bind_namespaces` constitute part of RDFLib's public interface, so changes to them should only be additive within the same minor version. Removing values, or removing namespaces that are bound by default, constitutes a breaking change.

See the Sample usage

```
>>> import rdflib
>>> from rdflib import Graph
>>> from rdflib.namespace import Namespace, NamespaceManager
>>> EX = Namespace('http://example.com/')
>>> namespace_manager = NamespaceManager(Graph())
>>> namespace_manager.bind('ex', EX, override=False)
>>> g = Graph()
>>> g.namespace_manager = namespace_manager
>>> all_ns = [n for n in g.namespace_manager.namespaces()]
>>> assert ('ex', rdflib.term.URIRef('http://example.com/')) in all_ns
>>>
```

#### Parameters

- **graph** (*Graph*) –
- **bind\_namespaces** (Literal['core', 'rdflib', 'none']) –

**\_\_contains\_\_**(*ref*)

#### Parameters

**ref** (*str*) –

#### Return type

*bool*

```
__dict__ = mappingproxy({'__module__': 'rdflib.namespace', '__doc__': "Class for
managing prefix => namespace mappings\n\n This class requires an RDFlib Graph as an
input parameter and may optionally have\n the parameter bind_namespaces set. This
second parameter selects a strategy which\n is one of the following:\n\n * core:\n *
binds several core RDF prefixes only\n * owl, rdf, rdfs, xsd, xml from the
NAMESPACE_PREFIXES_CORE object\n * rdflib:\n * binds all the namespaces shipped with
RDFlib as DefinedNamespace instances\n * all the core namespaces and all the
following: brick, csvw, dc, dcat\n * dcmitype, dcterms, dcam, doap, foaf, geo,
odrl, org, prof, prov, qb, schema\n * sh, skos, sosa, ssn, time, vann, void\n * see
the NAMESPACE_PREFIXES_RDFLIB object for the up-to-date list\n * this is default\n *
none:\n * binds no namespaces to prefixes\n * note this is NOT default behaviour\n *
cc:\n * using prefix bindings from prefix.cc which is a online prefixes database\n *
not implemented yet - this is aspirational\n\n .. attention::\n\n The namespaces
bound for specific values of ``bind_namespaces``\n constitute part of RDFlib's
public interface, so changes to them should\n only be additive within the same minor
version. Removing values, or\n removing namespaces that are bound by default,
constitutes a breaking\n change.\n\n See the\n Sample usage\n\n .. code-block::
pycon\n\n >>> import rdflib\n >>> from rdflib import Graph\n >>> from
rdflib.namespace import Namespace, NamespaceManager\n >>> EX =
Namespace('http://example.com/')\n >>> namespace_manager =
NamespaceManager(Graph())\n >>> namespace_manager.bind('ex', EX, override=False)\n
>>> g = Graph()\n >>> g.namespace_manager = namespace_manager\n >>> all_ns = [n for
n in g.namespace_manager.namespaces()]\n >>> assert ('ex',
rdflib.term.URIRef('http://example.com/')) in all_ns\n >>>\n ", '__init__':
<function NamespaceManager.__init__>, '__contains__': <function
NamespaceManager.__contains__>, 'reset': <function NamespaceManager.reset>,
'store': <property object>, 'qname': <function NamespaceManager.qname>,
'qname_strict': <function NamespaceManager.qname_strict>, 'normalizeUri':
<function NamespaceManager.normalizeUri>, 'compute_qname': <function
NamespaceManager.compute_qname>, 'compute_qname_strict': <function
NamespaceManager.compute_qname_strict>, 'expand_curie': <function
NamespaceManager.expand_curie>, '_store_bind': <function
NamespaceManager._store_bind>, 'bind': <function NamespaceManager.bind>,
'namespaces': <function NamespaceManager.namespaces>, 'absolutize': <function
NamespaceManager.absolutize>, '__dict__': <attribute '__dict__' of
'NamespaceManager' objects>, '__weakref__': <attribute '__weakref__' of
'NamespaceManager' objects>, '__annotations__': {'__cache': 'Dict[str, Tuple[str,
URIRef, str]]', '__cache_strict': 'Dict[str, Tuple[str, URIRef, str]]', '__strie':
'Dict[str, Any]', '__trie': 'Dict[str, Any]'}})
```

```
__init__(graph, bind_namespaces='rdflib')
```

#### Parameters

- `graph` (*Graph*) –
- `bind_namespaces` (Literal['core', 'rdflib', 'none']) –

```
__module__ = 'rdflib.namespace'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
absolutize(uri, defrag=1)
```

#### Parameters



- **uri** (*str*) –
- **defrag** (*int*) –

**Return type***URIRef***bind**(*prefix*, *namespace*, *override=True*, *replace=False*)

Bind a given namespace to the prefix

If override, rebind, even if the given namespace is already bound to another prefix.

If replace, replace any existing prefix with the new namespace

**Parameters**

- **prefix** (*Optional[str]*) –
- **namespace** (*Any*) –
- **override** (*bool*) –
- **replace** (*bool*) –

**Return type***None***compute\_qname**(*uri*, *generate=True*)**Parameters**

- **uri** (*str*) –
- **generate** (*bool*) –

**Return type***Tuple[str, URIRef, str]***compute\_qname\_strict**(*uri*, *generate=True*)**Parameters**

- **uri** (*str*) –
- **generate** (*bool*) –

**Return type***Tuple[str, str, str]***expand\_curie**(*curie*)

Expand a CURIE of the form &lt;prefix:element&gt;, e.g. “rdf:type” into its full expression:

```
>>> import rdflib
>>> g = rdflib.Graph()
>>> g.namespace_manager.expand_curie("rdf:type")
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#type')
```

Raises exception if a namespace is not bound to the prefix.

**Parameters****curie** (*str*) –**Return type***URIRef*

**namespaces()**

**Return type**

`Iterable[Tuple[str, URIRef]]`

**normalizeUri(*rdfTerm*)**

Takes an RDF Term and ‘normalizes’ it into a QName (using the registered prefix) or (unlike `compute_qname`) the Notation 3 form for URIs: <...URI...>

**Parameters**

**rdfTerm** (*str*) –

**Return type**

`str`

**qname(*uri*)**

**Parameters**

**uri** (*str*) –

**Return type**

`str`

**qname\_strict(*uri*)**

**Parameters**

**uri** (*str*) –

**Return type**

`str`

**reset()**

**Return type**

`None`

**property store:** *Store*

**rdflib.namespace.is\_ncname(*name*)**

**Parameters**

**name** (*str*) –

**Return type**

`int`

**rdflib.namespace.split\_uri(*uri*, *split\_start*=['Ll', 'Lu', 'Lo', 'Lt', 'Nl', 'Nd'])**

**Parameters**

- **uri** (*str*) –
- **split\_start** (*List[str]*) –

**Return type**

`Tuple[str, str]`

## rdflib.plugins package

### Subpackages

## rdflib.plugins.parsers package

### Submodules

## rdflib.plugins.parsers.RDFVOC module

**class** rdflib.plugins.parsers.RDFVOC.RDFVOC

Bases: *RDF*

Description: *URIRef* =

rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#Description')

ID: *URIRef* = rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#ID')

RDF: *URIRef* = rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#RDF')

about: *URIRef* =

rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#about')

datatype: *URIRef* =

rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#datatype')

li: *URIRef* = rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#li')

nodeID: *URIRef* =

rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#nodeID')

parseType: *URIRef* =

rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#parseType')

resource: *URIRef* =

rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#resource')

## rdflib.plugins.parsers.hext module

This is a rdflib plugin for parsing Hextuple files, which are Newline-Delimited JSON (ndjson) files, into Conjunctive. The store that backs the graph *must* be able to handle contexts, i.e. multiple graphs.

**class** rdflib.plugins.parsers.hext.HextuplesParser

Bases: *Parser*

An RDFLib parser for Hextuples

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.hext', '__doc__':
'\n An RDFLib parser for Hextuples\n\n ', '__init__': <function
HextuplesParser.__init__>, '_load_json_line': <function
HextuplesParser._load_json_line>, '_parse_hextuple': <function
HextuplesParser._parse_hextuple>, 'parse': <function HextuplesParser.parse>,
'__dict__': <attribute '__dict__' of 'HextuplesParser' objects>, '__weakref__':
<attribute '__weakref__' of 'HextuplesParser' objects>, '__annotations__': {}})
```

```
__init__()  
__module__ = 'rdflib.plugins.parsers.hexst'  
__weakref__  
    list of weak references to the object (if defined)  
parse(source, graph, **kwargs)
```

**Parameters**

- **source** (*InputSource*) –
- **graph** (*Graph*) –
- **kwargs** (*Any*) –

**Return type**

None

**rdflib.plugins.parsers.jsonld module**

This parser will interpret a JSON-LD document as an RDF Graph. See:

<http://json-ld.org/>

Example usage:

```
>>> from rdflib import Graph, URIRef, Literal  
>>> test_json = '''  
... {  
...     "@context": {  
...         "dc": "http://purl.org/dc/terms/",  
...         "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",  
...         "rdfs": "http://www.w3.org/2000/01/rdf-schema#"  
...     },  
...     "@id": "http://example.org/about",  
...     "dc:title": {  
...         "@language": "en",  
...         "@value": "Someone's Homepage"  
...     }  
... }  
... '''  
>>> g = Graph().parse(data=test_json, format='json-ld')  
>>> list(g) == [(URIRef('http://example.org/about'),  
...     URIRef('http://purl.org/dc/terms/title'),  
...     Literal("Someone's Homepage", lang='en'))]  
True
```

**class rdflib.plugins.parsers.jsonld.JsonLDParser**

Bases: *Parser*

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.jsonld', '__init__':  
<function JsonLDParser.__init__>, 'parse': <function JsonLDParser.parse>,  
'__dict__': <attribute '__dict__' of 'JsonLDParser' objects>, '__weakref__':  
<attribute '__weakref__' of 'JsonLDParser' objects>, '__doc__': None,  
'__annotations__': {}})
```

```

__init__()

__module__ = 'rdflib.plugins.parsers.jsonld'

__weakref__
    list of weak references to the object (if defined)

parse(source, sink, **kwargs)

```

#### Parameters

- **source** (*InputSource*) –
- **sink** (*Graph*) –
- **kwargs** (*Any*) –

#### Return type

*None*

```
rdflib.plugins.parsers.jsonld.to_rdf(data, dataset, base=None, context_data=None, version=None,
                                     generalized_rdf=False, allow_lists_of_lists=None)
```

#### Parameters

- **data** (*Any*) –
- **dataset** (*Graph*) –
- **base** (*Optional[str]*) –
- **context\_data** (*Optional[bool]*) –
- **version** (*Optional[float]*) –
- **generalized\_rdf** (*bool*) –
- **allow\_lists\_of\_lists** (*Optional[bool]*) –

### rdflib.plugins.parsers.notation3 module

notation3.py - Standalone Notation3 Parser Derived from CWM, the Closed World Machine

Authors of the original suite:

- Dan Connolly <@ @>
- Tim Berners-Lee <@ @>
- Yosi Scharf <@ @>
- Joseph M. Reagle Jr. <reagle@w3.org>
- Rich Salz <rsalz@zolera.com>

<http://www.w3.org/2000/10/swap/notation3.py>

Copyright 2000-2007, World Wide Web Consortium. Copyright 2001, MIT. Copyright 2001, Zolera Systems Inc.

License: W3C Software License <http://www.w3.org/Consortium/Legal/copyright-software>

Modified by Sean B. Palmer Copyright 2007, Sean B. Palmer.

Modified to work with rdflib by Gunnar Aastrand Grimnes Copyright 2010, Gunnar A. Grimnes

**exception** rdflib.plugins.parsers.notation3.**BadSyntax**(*uri, lines, argstr, i, why*)

Bases: `SyntaxError`

**Parameters**

- **uri** (`str`) –
- **lines** (`int`) –
- **argstr** (`str`) –
- **i** (`int`) –
- **why** (`str`) –

**\_\_init\_\_**(*uri, lines, argstr, i, why*)

**Parameters**

- **uri** (`str`) –
- **lines** (`int`) –
- **argstr** (`str`) –
- **i** (`int`) –
- **why** (`str`) –

**\_\_module\_\_** = 'rdflib.plugins.parsers.notation3'

**\_\_str\_\_**()

Return str(self).

**Return type**

`str`

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**property message:** `str`

**class** rdflib.plugins.parsers.notation3.**Formula**(*parent*)

Bases: `object`

**Parameters**

**parent** (`Graph`) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.notation3',
'number': 0, '__init__': <function Formula.__init__>, '__str__': <function
Formula.__str__>, 'id': <function Formula.id>, 'newBlankNode': <function
Formula.newBlankNode>, 'newUniversal': <function Formula.newUniversal>,
'declareExistential': <function Formula.declareExistential>, 'close': <function
Formula.close>, '__dict__': <attribute '__dict__' of 'Formula' objects>,
'__weakref__': <attribute '__weakref__' of 'Formula' objects>, '__doc__': None,
'__annotations__': {'existentials': 'Dict[str, BNode]', 'universals': 'Dict[str,
BNode]'}})
```

**\_\_init\_\_**(*parent*)

**Parameters**

**parent** (`Graph`) –

```

__module__ = 'rdflib.plugins.parsers.notation3'

__str__()
    Return str(self).

    Return type
        str

__weakref__
    list of weak references to the object (if defined)

close()

    Return type
        QuotedGraph

declareExistential(x)

    Parameters
        x (str) –

    Return type
        None

id()

    Return type
        BNode

newBlankNode(uri=None, why=None)

    Parameters
        • uri (Optional[str]) –
        • why (Optional[Any]) –

    Return type
        BNode

newUniversal(uri, why=None)

    Parameters
        • uri (str) –
        • why (Optional[Any]) –

    Return type
        Variable

number = 0

class rdflib.plugins.parsers.notation3.N3Parser
    Bases: TurtleParser
    An RDFLib parser for Notation3
    See http://www.w3.org/DesignIssues/Notation3.html

    __init__()

    __module__ = 'rdflib.plugins.parsers.notation3'

```

**parse**(*source*, *graph*, *encoding*='utf-8')

**Parameters**

- **source** (*InputSource*) –
- **graph** (*Graph*) –
- **encoding** (*Optional*[*str*]) –

**Return type**

*None*

**class** `rdflib.plugins.parsers.notation3.RDFSink`(*graph*)

Bases: *object*

**Parameters**

**graph** (*Graph*) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.notation3',
'__init__': <function RDFSink.__init__>, 'newFormula': <function
RDFSink.newFormula>, 'newGraph': <function RDFSink.newGraph>, 'newSymbol':
<function RDFSink.newSymbol>, 'newBlankNode': <function RDFSink.newBlankNode>,
'newLiteral': <function RDFSink.newLiteral>, 'newList': <function
RDFSink.newList>, 'newSet': <function RDFSink.newSet>, 'setDefaultNamespace':
<function RDFSink.setDefaultNamespace>, 'makeStatement': <function
RDFSink.makeStatement>, 'normalise': <function RDFSink.normalise>, 'intern':
<function RDFSink.intern>, 'bind': <function RDFSink.bind>, 'startDoc': <function
RDFSink.startDoc>, 'endDoc': <function RDFSink.endDoc>, '__dict__': <attribute
'__dict__' of 'RDFSink' objects>, '__weakref__': <attribute '__weakref__' of
'RDFSink' objects>, '__doc__': None, '__annotations__': {'rootFormula':
'Optional[Formula]'}}})
```

**\_\_init\_\_**(*graph*)

**Parameters**

**graph** (*Graph*) –

**\_\_module\_\_** = 'rdflib.plugins.parsers.notation3'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**bind**(*pf<sub>x</sub>*, *uri*)

**Return type**

*None*

**endDoc**(*formula*)

**Parameters**

**formula** (*Optional*[*Formula*]) –

**Return type**

*None*

**intern**(*something*)

**Parameters**

**something** (*TypeVar*(*\_AnyT*)) –



**Return type**`TypeVar(_AnyT)``makeStatement(quadruple, why=None)`**Parameters**

- **quadruple** (`Tuple[Union[Formula, Graph, None], Node, Node, Node]`) –
- **why** (`Optional[Any]`) –

**Return type**`None``newBlankNode(arg=None, uri=None, why=None)`**Parameters**

- **arg** (`Union[Formula, Graph, Any, None]`) –
- **uri** (`Optional[str]`) –
- **why** (`Optional[Callable[[], None]]`) –

**Return type**`BNode``newFormula()`**Return type**`Formula``newGraph(identifier)`**Parameters****identifier** (`Identifier`) –**Return type**`Graph``newList(n, f)`**Parameters**

- **n** (`List[Any]`) –
- **f** (`Optional[Formula]`) –

**Return type**`IdentifiedNode``newLiteral(s, dt, lang)`**Parameters**

- **s** (`str`) –
- **dt** (`Optional[URIRef]`) –
- **lang** (`Optional[str]`) –

**Return type**`Literal`

**newSet**(\*args)

**Parameters**

**args** (*TypeVar*(*\_AnyT*)) –

**Return type**

*Set*[*TypeVar*(*\_AnyT*)]

**newSymbol**(\*args)

**Parameters**

**args** (*str*) –

**Return type**

*URIRef*

**normalise**(*f*, *n*)

**Parameters**

- **f** (*Optional*[*Formula*]) –
- **n** (*Union*[*Tuple*[*int*, *str*], *bool*, *int*, *Decimal*, *float*, *TypeVar*(*\_AnyT*)]) –

**Return type**

*Union*[*URIRef*, *Literal*, *BNode*, *TypeVar*(*\_AnyT*)]

**setDefaultNamespace**(\*args)

**Parameters**

**args** (*bytes*) –

**Return type**

*str*

**startDoc**(*formula*)

**Parameters**

**formula** (*Optional*[*Formula*]) –

**Return type**

*None*

**class** `rdflib.plugins.parsers.notation3.SinkParser`(*store*, *openFormula*=*None*, *thisDoc*="",  
*baseURI*=*None*, *genPrefix*="", *why*=*None*,  
*turtle*=*False*)

Bases: *object*

**Parameters**

- **store** (*RDFSink*) –
- **openFormula** (*Optional*[*Formula*]) –
- **thisDoc** (*str*) –
- **baseURI** (*Optional*[*str*]) –
- **genPrefix** (*str*) –
- **why** (*Optional*[*Callable*[[], *None*]]) –
- **turtle** (*bool*) –

**BadSyntax**(*argstr*, *i*, *msg*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **msg** (*str*) –

**Return type**

*NoReturn*

**UEScape**(*argstr*, *i*, *startline*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **startline** (*int*) –

**Return type**

*Tuple*[*int*, *str*]

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.notation3',
'__init__': <function SinkParser.__init__>, 'here': <function SinkParser.here>,
'formula': <function SinkParser.formula>, 'loadStream': <function
SinkParser.loadStream>, 'loadBuf': <function SinkParser.loadBuf>, 'feed':
<function SinkParser.feed>, 'directiveOrStatement': <function
SinkParser.directiveOrStatement>, 'tok': <function SinkParser.tok>, 'sparqlTok':
<function SinkParser.sparqlTok>, 'directive': <function SinkParser.directive>,
'sparqlDirective': <function SinkParser.sparqlDirective>, 'bind': <function
SinkParser.bind>, 'setKeywords': <function SinkParser.setKeywords>, 'startDoc':
<function SinkParser.startDoc>, 'endDoc': <function SinkParser.endDoc>,
'makeStatement': <function SinkParser.makeStatement>, 'statement': <function
SinkParser.statement>, 'subject': <function SinkParser.subject>, 'verb': <function
SinkParser.verb>, 'prop': <function SinkParser.prop>, 'item': <function
SinkParser.item>, 'blankNode': <function SinkParser.blankNode>, 'path': <function
SinkParser.path>, 'anonymousNode': <function SinkParser.anonymousNode>, 'node':
<function SinkParser.node>, 'property_list': <function SinkParser.property_list>,
'commaSeparatedList': <function SinkParser.commaSeparatedList>, 'objectList':
<function SinkParser.objectList>, 'checkDot': <function SinkParser.checkDot>,
'uri_ref2': <function SinkParser.uri_ref2>, 'skipSpace': <function
SinkParser.skipSpace>, 'variable': <function SinkParser.variable>, 'bareWord':
<function SinkParser.bareWord>, 'qname': <function SinkParser.qname>, 'object':
<function SinkParser.object>, 'nodeOrLiteral': <function SinkParser.nodeOrLiteral>,
'uriOf': <function SinkParser.uriOf>, 'strconst': <function SinkParser.strconst>,
'_unicodeEscape': <function SinkParser._unicodeEscape>, 'uEscape': <function
SinkParser.uEscape>, 'UEScape': <function SinkParser.UEscape>, 'BadSyntax':
<function SinkParser.BadSyntax>, '__dict__': <attribute '__dict__' of 'SinkParser'
objects>, '__weakref__': <attribute '__weakref__' of 'SinkParser' objects>,
'__doc__': None, '__annotations__': {'_anonymousNodes': 'Dict[str, BNode]',
'_variables': 'Dict[str, Variable]', '_parentVariables': 'Dict[str, Variable]',
'_reason2': 'Optional[Callable[..., None]]', '_baseURI': 'Optional[str]',
'_formula': 'Optional[Formula]', '_context': 'Optional[Formula]',
'_parentContext': 'Optional[Formula]'}})
```

**\_\_init\_\_**(*store*, *openFormula*=None, *thisDoc*="", *baseURI*=None, *genPrefix*="", *why*=None, *turtle*=False)

note: namespace names should *not* end in # ; the # will get added during qname processing

**Parameters**

- **store** (*RDFSink*) –
- **openFormula** (*Optional[Formula]*) –
- **thisDoc** (*str*) –
- **baseURI** (*Optional[str]*) –
- **genPrefix** (*str*) –
- **why** (*Optional[Callable[[], None]]*) –
- **turtle** (*bool*) –

**\_\_module\_\_** = 'rdflib.plugins.parsers.notation3'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**anonymousNode**(*ln*)

Remember or generate a term for one of these \_: anonymous nodes

**Parameters**

**ln** (*str*) –

**Return type**

*BNode*

**bareWord**(*argstr*, *i*, *res*)

abc -> :abc

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **res** (*MutableSequence[Any]*) –

**Return type**

*int*

**bind**(*qn*, *uri*)

**Parameters**

- **qn** (*str*) –
- **uri** (*bytes*) –

**Return type**

*None*

**blankNode**(*uri*=None)

**Parameters**

**uri** (*Optional[str]*) –

**Return type**

*BNode*

**checkDot**(*argstr*, *i*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –

**Return type**

*int*

**commaSeparatedList**(*argstr*, *j*, *res*, *what*)

return value: -1 bad syntax; >1 new position in argstr res has things found appended

**Parameters**

- **argstr** (*str*) –
- **j** (*int*) –
- **res** (*MutableSequence*[*Any*]) –
- **what** (*Callable*[[*str*, *int*, *MutableSequence*[*Any*]], *int*) –

**Return type**

*int*

**directive**(*argstr*, *i*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –

**Return type**

*int*

**directiveOrStatement**(*argstr*, *h*)

**Parameters**

- **argstr** (*str*) –
- **h** (*int*) –

**Return type**

*int*

**endDoc**()

Signal end of document and stop parsing. returns formula

**Return type**

*Optional*[*Formula*]

**feed**(*octets*)

Feed an octet stream to the parser

if *BadSyntax* is raised, the string passed in the exception object is the remainder after any statements have been parsed. So if there is more data to feed to the parser, it should be straightforward to recover.

**Parameters**

**octets** (*Union*[*str*, *bytes*]) –

**Return type**

*None*

**formula()**

**Return type**

`Optional[Formula]`

**here(*i*)**

String generated from position in file

This is for repeatability when referring people to bnodes in a document. This has diagnostic uses less formally, as it should point one to which bnode the arbitrary identifier actually is. It gives the line and character number of the '[' character or path character which introduced the blank node. The first blank node is boringly \_L1C1. It used to be used only for tracking, but for tests in general it makes the canonical ordering of bnodes repeatable.

**Parameters**

**i** (`int`) –

**Return type**

`str`

**item(*argstr*, *i*, *res*)**

**Parameters**

- **argstr** (`str`) –
- **res** (`MutableSequence[Any]`) –

**Return type**

`int`

**loadBuf(*buf*)**

Parses a buffer and returns its top level formula

**Parameters**

**buf** (`Union[str, bytes]`) –

**Return type**

`Optional[Formula]`

**loadStream(*stream*)**

**Parameters**

**stream** (`Union[IO[str], IO[bytes]]`) –

**Return type**

`Optional[Formula]`

**makeStatement(*quadruple*)**

**Return type**

`None`

**node(*argstr*, *i*, *res*, *subjectAlready=None*)**

Parse the <node> production. Space is now skipped once at the beginning instead of in multiple calls to `self.skipSpace()`.

**Parameters**

- **argstr** (`str`) –
- **i** (`int`) –
- **res** (`MutableSequence[Any]`) –

- **subjectAlready** (*Optional*[*Node*]) –

**Return type**

*int*

**nodeOrLiteral**(*argstr*, *i*, *res*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **res** (*MutableSequence*[*Any*]) –

**Return type**

*int*

**object**(*argstr*, *i*, *res*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **res** (*MutableSequence*[*Any*]) –

**Return type**

*int*

**objectList**(*argstr*, *i*, *res*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **res** (*MutableSequence*[*Any*]) –

**Return type**

*int*

**path**(*argstr*, *i*, *res*)

Parse the path production.

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **res** (*MutableSequence*[*Any*]) –

**Return type**

*int*

**prop**(*argstr*, *i*, *res*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **res** (*MutableSequence*[*Any*]) –

**Return type**

`int`

**property\_list**(*argstr*, *i*, *subj*)

Parse property list Leaves the terminating punctuation in the buffer

**Parameters**

- **argstr** (`str`) –
- **i** (`int`) –
- **subj** (`Node`) –

**Return type**

`int`

**qname**(*argstr*, *i*, *res*)

xyz:def -> ('xyz', 'def') If not in keywords and keywordsSet: def -> ('', 'def') :def -> ('', 'def')

**Parameters**

- **argstr** (`str`) –
- **i** (`int`) –
- **res** (`MutableSequence[Union[Identifier, Tuple[str, str]]]`) –

**Return type**

`int`

**setKeywords**(*k*)

Takes a list of strings

**Parameters**

**k** (`Optional[List[str]]`) –

**Return type**

`None`

**skipSpace**(*argstr*, *i*)

Skip white space, newlines and comments. return -1 if EOF, else position of first non-ws character

**Parameters**

- **argstr** (`str`) –
- **i** (`int`) –

**Return type**

`int`

**sparqlDirective**(*argstr*, *i*)

turtle and trig support BASE/PREFIX without @ and without terminating .

**Parameters**

- **argstr** (`str`) –
- **i** (`int`) –

**Return type**

`int`



**sparqlTok**(*tok, argstr, i*)

Check for SPARQL keyword. Space must have been stripped on entry and we must not be at end of file.  
Case insensitive and not preceded by @

**Parameters**

- **tok** (*str*) –
- **argstr** (*str*) –
- **i** (*int*) –

**Return type**

*int*

**startDoc**()

**Return type**

*None*

**statement**(*argstr, i*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –

**Return type**

*int*

**strconst**(*argstr, i, delim*)

parse an N3 string constant delimited by delim. return index, val

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **delim** (*str*) –

**Return type**

*Tuple[int, str]*

**subject**(*argstr, i, res*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **res** (*MutableSequence[Any]*) –

**Return type**

*int*

**tok**(*tok, argstr, i, colon=False*)

Check for keyword. Space must have been stripped on entry and we must not be at end of file.

if colon, then keyword followed by colon is ok (@prefix:<blah> is ok, rdf:type shortcut a must be followed by ws)

**Parameters**

- **tok** (*str*) –

- **argstr** (*str*) –
- **i** (*int*) –
- **colon** (*bool*) –

**Return type**

*int*

**uEscape**(*argstr*, *i*, *startline*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **startline** (*int*) –

**Return type**

*Tuple*[*int*, *str*]

**uriOf**(*sym*)

**Parameters**

**sym** (*Union*[*Identifier*, *Tuple*[*str*, *str*]]) –

**Return type**

*str*

**uri\_ref2**(*argstr*, *i*, *res*)

Generate uri from n3 representation.

Note that the RDF convention of directly concatenating NS and local name is now used though I prefer inserting a '#' to make the namespaces look more like what XML folks expect.

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **res** (*MutableSequence*[*Any*]) –

**Return type**

*int*

**variable**(*argstr*, *i*, *res*)

?abc -> variable(:abc)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –

**Return type**

*int*

**verb**(*argstr*, *i*, *res*)

has \_prop\_ is \_prop\_ of a = \_prop\_ >- prop -> <- prop <- \_operator\_

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –

- **res** (`MutableSequence[Any]`) –

**Return type**

`int`

**class** `rdflib.plugins.parsers.notation3.TurtleParser`

Bases: `Parser`

An RDFLib parser for Turtle

See <http://www.w3.org/TR/turtle/>

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.notation3',
'__doc__': '\n An RDFLib parser for Turtle\n\n See http://www.w3.org/TR/turtle/\n',
'__init__': <function TurtleParser.__init__>, 'parse': <function TurtleParser.parse>,
'__dict__': <attribute '__dict__' of 'TurtleParser' objects>,
'__weakref__': <attribute '__weakref__' of 'TurtleParser' objects>,
'__annotations__': {}})
```

```
__init__()
```

```
__module__ = 'rdflib.plugins.parsers.notation3'
```

```
__weakref__
```

list of weak references to the object (if defined)

**parse**(*source*, *graph*, *encoding*='utf-8', *turtle*=True)

**Parameters**

- **source** (`InputSource`) –
- **graph** (`Graph`) –
- **encoding** (`Optional[str]`) –
- **turtle** (`bool`) –

**Return type**

`None`

`rdflib.plugins.parsers.notation3.base()`

The base URI for this process - the Web equiv of cwd

Relative or absolute unix-standard filenames parsed relative to this yield the URI of the file. If we had a reliable way of getting a computer name, we should put it in the hostname just to prevent ambiguity

**Return type**

`str`

`rdflib.plugins.parsers.notation3.hexify(ustr)`

Use URL encoding to return an ASCII string corresponding to the given UTF8 string

```
>>> hexify("http://example/a b")
b'http://example/a%20b'
```

**Parameters**

**ustr** (`str`) –

**Return type**

`bytes`

`rdflib.plugins.parsers.notation3.join(here, there)`

join an absolute URI and URI reference (non-ascii characters are supported/doctested; haven't checked the details of the IRI spec though)

`here` is assumed to be absolute. `there` is URI reference.

```
>>> join('http://example/x/y/z', '../abc')
'http://example/x/abc'
```

Raise `ValueError` if there uses relative path syntax but here has no hierarchical path.

```
>>> join('mid:foo@example', '../foo')
Traceback (most recent call last):
  raise ValueError(here)
ValueError: Base <mid:foo@example> has no slash
after colon - with relative '../foo'.
```

```
>>> join('http://example/x/y/z', '')
'http://example/x/y/z'
```

```
>>> join('mid:foo@example', '#foo')
'mid:foo@example#foo'
```

We grok IRIs

```
>>> len(u'Andr\xe9')
5
```

```
>>> join('http://example.org/', u'#Andr\xe9')
u'http://example.org/#Andr\xe9'
```

#### Parameters

- **here** (`str`) –
- **there** (`str`) –

#### Return type

`str`

`rdflib.plugins.parsers.notation3.runNamespace()`

Returns a URI suitable as a namespace for run-local objects

#### Return type

`str`

`rdflib.plugins.parsers.notation3.splitFragP(uriref, punc=0)`

split a URI reference before the fragment

Punctuation is kept.

e.g.

```
>>> splitFragP("abc#def")
('abc', '#def')
```

```
>>> splitFragP("abcdef")
('abcdef', '')
```

**Parameters**

- **uriref** (*str*) –
- **punc** (*int*) –

**Return type**

`Tuple[str, str]`

`rdflib.plugins.parsers.notation3.uniqueURI()`

A unique URI

**Return type**

`str`

**rdflib.plugins.parsers.nquads module**

This is a rdflib plugin for parsing NQuad files into Conjunctive graphs that can be used and queried. The store that backs the graph *must* be able to handle contexts.

```
>>> from rdflib import ConjunctiveGraph, URIRef, Namespace
>>> g = ConjunctiveGraph()
>>> data = open("test/data/nquads.rdflib/example.nquads", "rb")
>>> g.parse(data, format="nquads")
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> assert len(g.store) == 449
>>> # There should be 16 separate contexts
>>> assert len([x for x in g.store.contexts()]) == 16
>>> # is the name of entity E10009 "Arco Publications"?
>>> # (in graph http://bibliographica.org/entity/E10009)
>>> # Looking for:
>>> # <http://bibliographica.org/entity/E10009>
>>> # <http://xmlns.com/foaf/0.1/name>
>>> # "Arco Publications"
>>> # <http://bibliographica.org/entity/E10009>
>>> s = URIRef("http://bibliographica.org/entity/E10009")
>>> FOAF = Namespace("http://xmlns.com/foaf/0.1/")
>>> assert(g.value(s, FOAF.name).eq("Arco Publications"))
```

**class** `rdflib.plugins.parsers.nquads.NQuadsParser`(*sink=None, bnode\_context=None*)

Bases: `W3CNTriplesParser`

**Parameters**

- **sink** (`Union[DummySink, NTGraphSink, None]`) –
- **bnode\_context** (`Optional[MutableMapping[str, BNode]]`) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.nquads', 'parse':  
<function NQuadsParser.parse>, 'parseline': <function NQuadsParser.parseline>,  
'__dict__': <attribute '__dict__' of 'NQuadsParser' objects>, '__weakref__':  
<attribute '__weakref__' of 'NQuadsParser' objects>, '__doc__': None,  
'__annotations__': {'sink': "Union[DummySink, 'NTGraphSink']", 'buffer':  
'Optional[str]', 'file': 'Optional[Union[TextIO, codecs.StreamReader]]', 'line':  
'Optional[str]'}}})
```

```
__module__ = 'rdflib.plugins.parsers.nquads'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
buffer: Optional[str]
```

```
file: Optional[Union[TextIO, codecs.StreamReader]]
```

```
line: Optional[str]
```

```
parse(inputsource, sink, bnode_context=None, **kwargs)
```

Parse inputsource as an N-Quads file.

#### Parameters

- **inputsource** (*InputSource*) – the source of N-Quads-formatted data
- **sink** (*ConjunctiveGraph*) – where to send parsed triples
- **bnode\_context** (*Optional[MutableMapping[str, BNode]]*) – a dict mapping blank node identifiers to *rdflib.term.BNode* instances. See *.W3CNTriplesParser.parse*
- **kwargs** (*Any*) –

#### Return type

*ConjunctiveGraph*

```
parseline(bnode_context=None)
```

#### Parameters

**bnode\_context** (*Optional[MutableMapping[str, BNode]]*) –

#### Return type

*None*

```
sink: Union[DummySink, 'NTGraphSink']
```

## rdflib.plugins.parsers.ntriples module

N-Triples Parser License: GPL 2, W3C, BSD, or MIT Author: Sean B. Palmer, inamidst.com

```
class rdflib.plugins.parsers.ntriples.DummySink
```

Bases: *object*

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.ntriples',  
'__init__': <function DummySink.__init__>, 'triple': <function DummySink.triple>,  
'__dict__': <attribute '__dict__' of 'DummySink' objects>, '__weakref__':  
<attribute '__weakref__' of 'DummySink' objects>, '__doc__': None,  
'__annotations__': {}}})
```

```

__init__()

__module__ = 'rdflib.plugins.parsers.ntriples'

__weakref__
    list of weak references to the object (if defined)

triple(s, p, o)

class rdflib.plugins.parsers.ntriples.NTGraphSink(graph)
    Bases: object

        Parameters
            graph (Graph) –

__init__(graph)

        Parameters
            graph (Graph) –

__module__ = 'rdflib.plugins.parsers.ntriples'

__slots__ = ('g',)

g

triple(s, p, o)

        Parameters
            • s (Node) –
            • p (Node) –
            • o (Node) –

        Return type
            None

class rdflib.plugins.parsers.ntriples.NTParser
    Bases: Parser

    parser for the ntriples format, often stored with the .nt extension

    See http://www.w3.org/TR/rdf-testcases/#ntriples

__module__ = 'rdflib.plugins.parsers.ntriples'

__slots__ = ()

classmethod parse(source, sink, **kwargs)
    Parse the NT format

        Parameters
            • source (InputSource) – the source of NT-formatted data
            • sink (Graph) – where to send parsed triples
            • kwargs (Any) – Additional arguments to pass to .W3CNTriplesParser.parse

        Return type
            None

```

```
class rdflib.plugins.parsers.ntriples.W3CNTriplesParser(sink=None, bnode_context=None)
```

Bases: *object*

An N-Triples Parser. This is a legacy-style Triples parser for NTriples provided by W3C Usage:

```
p = W3CNTriplesParser(sink=MySink())
sink = p.parse(f) # file; use parsestring for a string
```

To define a context in which blank node identifiers refer to the same blank node across instances of NTriplesParser, pass the same dict as `bnode_context` to each instance. By default, a new blank node context is created for each instance of *W3CNTriplesParser*.

#### Parameters

- **sink** (`Union[DummySink, NTGraphSink, None]`) –
- **bnode\_context** (`Optional[MutableMapping[str, BNode]]`) –

```
__init__(sink=None, bnode_context=None)
```

#### Parameters

- **sink** (`Union[DummySink, NTGraphSink, None]`) –
- **bnode\_context** (`Optional[MutableMapping[str, BNode]]`) –

```
__module__ = 'rdflib.plugins.parsers.ntriples'
```

```
__slots__ = ('_bnode_ids', 'sink', 'buffer', 'file', 'line')
```

**buffer:** `Optional[str]`

**eat**(*pattern*)

#### Parameters

**pattern** (`Pattern[str]`) –

#### Return type

`Match[str]`

**file:** `Union[TextIO, StreamReader, None]`

**line:** `Optional[str]`

**literal**()

#### Return type

`Union[Literal[False], Literal]`

**nodeid**(*bnode\_context=None*)

#### Parameters

**bnode\_context** (`Optional[MutableMapping[str, BNode]]`) –

#### Return type

`Union[Literal[False], BNode]`

**object**(*bnode\_context=None*)

#### Parameters

**bnode\_context** (`Optional[MutableMapping[str, BNode]]`) –

#### Return type

`Union[URIRef, BNode, Literal]`



**parse**(*f*, *bnode\_context*=None)

Parse *f* as an N-Triples file.

**Parameters**

- **f** (`Union[TextIO, IO[bytes], StreamReader]`) – the N-Triples source
- **bnode\_context** (`Optional[MutableMapping[str, BNode]]`) – a dict mapping blank node identifiers (e.g., `a` in `_:a`) to `~rdflib.term.BNode` instances. An empty dict can be passed in to define a distinct context for a given call to *parse*.

**Return type**

`Union[DummySink, NTGraphSink]`

**parseline**(*bnode\_context*=None)

**Parameters**

**bnode\_context** (`Optional[MutableMapping[str, BNode]]`) –

**Return type**

`None`

**parsestring**(*s*, *\*\*kwargs*)

Parse *s* as an N-Triples string.

**Parameters**

**s** (`Union[bytes, bytearray, str]`) –

**Return type**

`None`

**peek**(*token*)

**Parameters**

**token** (`str`) –

**Return type**

`bool`

**predicate**()

**Return type**

`URIRef`

**readline**()

Read an N-Triples line from buffered input.

**Return type**

`Optional[str]`

**sink**: `Union[DummySink, NTGraphSink]`

**subject**(*bnode\_context*=None)

**Return type**

`Union[BNode, URIRef]`

**uriref**()

**Return type**

`Union[Literal[False], URIRef]`

`rdflib.plugins.parsers.ntriples.unquote(s)`

Unquote an N-Triples string.

**Parameters**

**s** (*str*) –

**Return type**

*str*

`rdflib.plugins.parsers.ntriples.uriquote(uri)`

**Parameters**

**uri** (*str*) –

**Return type**

*str*

### `rdflib.plugins.parsers.rdfxml` module

An RDF/XML parser for RDFLib

**class** `rdflib.plugins.parsers.rdfxml.BagID`(*value: str, base: Optional[str] = None*)

Bases: *URIRef*

`__init__`(*val*)

`__module__` = `'rdflib.plugins.parsers.rdfxml'`

`__slots__` = `['li']`

`li`

`next_li`()

**class** `rdflib.plugins.parsers.rdfxml.ElementHandler`

Bases: *object*

`__init__`()

`__module__` = `'rdflib.plugins.parsers.rdfxml'`

`__slots__` = `['start', 'char', 'end', 'li', 'id', 'base', 'subject', 'predicate', 'object', 'list', 'language', 'datatype', 'declared', 'data']`

`base`

`char`

`data`

`datatype`

`declared`

`end`

`id`

`language`

**li**  
**list**  
**next\_li()**  
**object**  
**predicate**  
**start**  
**subject**

**class** rdflib.plugins.parsers.rdfxml.RDFXMLHandler(*store*)

Bases: [ContentHandler](#)

**Parameters**

**store** ([Graph](#)) –

**\_\_init\_\_**(*store*)

**Parameters**

**store** ([Graph](#)) –

**\_\_module\_\_** = 'rdflib.plugins.parsers.rdfxml'

**absolutize**(*uri*)

**Parameters**

**uri** ([str](#)) –

**Return type**

[URIRef](#)

**add\_reified**(*sid, spo*)

**Parameters**

- **sid** ([Identifier](#)) –
- **spo** ([Tuple](#)[[Node](#), [Node](#), [Node](#)]) –

**characters**(*content*)

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity so that the Locator provides useful information.

**Parameters**

**content** ([str](#)) –

**Return type**

[None](#)

**convert**(*name, qname, attrs*)

**Parameters**

- **name** ([Tuple](#)[[Optional](#)[[str](#)], [str](#)]) –
- **attrs** ([AttributesImpl](#)) –

**Return type**`Tuple[URIRef, Dict[URIRef, str]]`**property current:** `Optional[ElementHandler]`**document\_element\_start**(*name, qname, attrs*)**Parameters**

- **name** (`Tuple[str, str]`) –
- **attrs** (`AttributesImpl`) –

**Return type**`None`**endElementNS**(*name, qname*)

Signals the end of an element in namespace mode.

The name parameter contains the name of the element type, just as with the startElementNS event.

**Parameters****name** (`Tuple[Optional[str], str]`) –**Return type**`None`**endPrefixMapping**(*prefix*)

End the scope of a prefix-URI mapping.

See startPrefixMapping for details. This event will always occur after the corresponding endElement event, but the order of endPrefixMapping events is not otherwise guaranteed.

**Parameters****prefix** (`Optional[str]`) –**Return type**`None`**error**(*message*)**Parameters****message** (`str`) –**Return type**`NoReturn`**get\_current**()**Return type**`Optional[ElementHandler]`**get\_next**()**Return type**`Optional[ElementHandler]`**get\_parent**()**Return type**`Optional[ElementHandler]`

**ignorableWhitespace**(*content*)

Receive notification of ignorable whitespace in element content.

Validating Parsers must use this method to report each chunk of ignorable whitespace (see the W3C XML 1.0 recommendation, section 2.10): non-validating parsers may also use this method if they are capable of parsing and using content models.

SAX parsers may return all contiguous whitespace in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

**Return type**

`None`

**list\_node\_element\_end**(*name*, *qname*)**Parameters**

**name** (`Tuple[str, str]`) –

**Return type**

`None`

**literal\_element\_char**(*data*)**Parameters**

**data** (`str`) –

**Return type**

`None`

**literal\_element\_end**(*name*, *qname*)**Parameters**

**name** (`Tuple[str, str]`) –

**Return type**

`None`

**literal\_element\_start**(*name*, *qname*, *attrs*)**Parameters**

- **name** (`Tuple[str, str]`) –
- **attrs** (`AttributesImpl`) –

**Return type**

`None`

property next: `Optional[ElementHandler]`

**node\_element\_end**(*name*, *qname*)**Parameters**

**name** (`Tuple[str, str]`) –

**Return type**

`None`

**node\_element\_start**(*name*, *qname*, *attrs*)**Parameters**

- **name** (`Tuple[str, str]`) –

- **attrs** (`AttributesImpl`) –

**Return type**`None`**property parent:** `Optional[ElementHandler]`**processingInstruction**(*target, data*)

Receive notification of a processing instruction.

The Parser will invoke this method once for each processing instruction found: note that processing instructions may occur before or after the main document element.

A SAX parser should never report an XML declaration (XML 1.0, section 2.8) or a text declaration (XML 1.0, section 4.3.1) using this method.

**Return type**`None`**property\_element\_char**(*data*)**Parameters****data** (`str`) –**Return type**`None`**property\_element\_end**(*name, qname*)**Parameters****name** (`Tuple[str, str]`) –**Return type**`None`**property\_element\_start**(*name, qname, attrs*)**Parameters**

- **name** (`Tuple[str, str]`) –
- **attrs** (`AttributesImpl`) –

**Return type**`None`**reset**()**Return type**`None`**setDocumentLocator**(*locator*)

Called by the parser to give the application a locator for locating the origin of document events.

SAX parsers are strongly encouraged (though not absolutely required) to supply a locator: if it does so, it must supply the locator to the application by invoking this method before invoking any of the other methods in the `DocumentHandler` interface.

The locator allows the application to determine the end position of any document-related event, even if the parser is not reporting an error. Typically, the application will use this information for reporting its own errors (such as character content that does not match an application's business rules). The information returned by the locator is probably not sufficient for use with a search engine.

Note that the locator will return correct information only during the invocation of the events in this interface. The application should not attempt to use it at any other time.

**Parameters**

**locator** (*Locator*) –

**startDocument()**

Receive notification of the beginning of a document.

The SAX parser will invoke this method only once, before any other methods in this interface or in DTD-Handler (except for setDocumentLocator).

**Return type**

*None*

**startElementNS**(*name, qname, attrs*)

Signals the start of an element in namespace mode.

The name parameter contains the name of the element type as a (uri, localname) tuple, the qname parameter the raw XML 1.0 name used in the source document, and the attrs parameter holds an instance of the Attributes class containing the attributes of the element.

The uri part of the name tuple is None for elements which have no namespace.

**Parameters**

- **name** (*Tuple[Optional[str], str]*) –
- **attrs** (*AttributesImpl*) –

**Return type**

*None*

**startPrefixMapping**(*prefix, namespace*)

Begin the scope of a prefix-URI Namespace mapping.

The information from this event is not necessary for normal Namespace processing: the SAX XML reader will automatically replace prefixes for element and attribute names when the <http://xml.org/sax/features/namespaces> feature is true (the default).

There are cases, however, when applications need to use prefixes in character data or in attribute values, where they cannot safely be expanded automatically; the start/endPrefixMapping event supplies the information to the application to expand prefixes in those contexts itself, if necessary.

Note that start/endPrefixMapping events are not guaranteed to be properly nested relative to each-other: all startPrefixMapping events will occur before the corresponding startElement event, and all endPrefixMapping events will occur after the corresponding endElement event, but their order is not guaranteed.

**Parameters**

- **prefix** (*Optional[str]*) –
- **namespace** (*str*) –

**Return type**

*None*

**class** `rdflib.plugins.parsers.rdfxml.RDFXMLParser`

Bases: *Parser*

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.rdfxml', '__init__':  
<function RDFXMLParser.__init__>, 'parse': <function RDFXMLParser.parse>,  
'__dict__': <attribute '__dict__' of 'RDFXMLParser' objects>, '__weakref__':  
<attribute '__weakref__' of 'RDFXMLParser' objects>, '__doc__': None,  
'__annotations__': {}})
```

```
__init__()
```

```
__module__ = 'rdflib.plugins.parsers.rdfxml'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
parse(source, sink, **args)
```

#### Parameters

- **source** (*InputSource*) –
- **sink** (*Graph*) –
- **args** (*Any*) –

#### Return type

*None*

```
rdflib.plugins.parsers.rdfxml.create_parser(target, store)
```

#### Parameters

- **target** (*InputSource*) –
- **store** (*Graph*) –

#### Return type

*XMLReader*

### rdflib.plugins.parsers.trig module

```
class rdflib.plugins.parsers.trig.TrigParser
```

Bases: *Parser*

An RDFLib parser for TriG

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.trig', '__doc__':  
'\n An RDFLib parser for TriG\n\n ', '__init__': <function TrigParser.__init__>,  
'parse': <function TrigParser.parse>, '__dict__': <attribute '__dict__' of  
'TrigParser' objects>, '__weakref__': <attribute '__weakref__' of 'TrigParser'  
objects>, '__annotations__': {}})
```

```
__init__()
```

```
__module__ = 'rdflib.plugins.parsers.trig'
```

```
__weakref__
```

list of weak references to the object (if defined)



**parse**(*source*, *graph*, *encoding*='utf-8')

**Parameters**

- **source** (*InputSource*) –
- **graph** (*Graph*) –
- **encoding** (*str*) –

**Return type**

*None*

```
class rdflib.plugins.parsers.trig.TrigSinkParser(store, openFormula=None, thisDoc="",
                                                baseURI=None, genPrefix="", why=None,
                                                turtle=False)
```

Bases: *SinkParser*

**Parameters**

- **store** (*RDFSink*) –
- **openFormula** (*Optional[Formula]*) –
- **thisDoc** (*str*) –
- **baseURI** (*Optional[str]*) –
- **genPrefix** (*str*) –
- **why** (*Optional[Callable[[], None]]*) –
- **turtle** (*bool*) –

```
__module__ = 'rdflib.plugins.parsers.trig'
```

**directiveOrStatement**(*argstr*, *h*)

**Parameters**

- **argstr** (*str*) –
- **h** (*int*) –

**Return type**

*int*

**graph**(*argstr*, *i*)

Parse trig graph, i.e.

```
<urn:graphname> = { .. triples .. }
```

return -1 if it doesn't look like a graph-decl raise Exception if it looks like a graph, but isn't.

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –

**Return type**

*int*

**labelOrSubject**(*argstr*, *i*, *res*)

**Parameters**

- **argstr** (*str*) –
- **i** (*int*) –
- **res** (*MutableSequence[Any]*) –

**Return type**

*int*

`rdflib.plugins.parsers.trig.becauseSubGraph(*args, **kwargs)`

### **rdflib.plugins.parsers.trix module**

A TriX parser for RDFLib

**class** `rdflib.plugins.parsers.trix.TriXHandler`(*store*)

Bases: `ContentHandler`

An Sax Handler for TriX. See <http://sw.nokia.com/trix/>

**Parameters**

**store** (*Store*) –

**\_\_init\_\_**(*store*)

**Parameters**

**store** (*Store*) –

**\_\_module\_\_** = `'rdflib.plugins.parsers.trix'`

**characters**(*content*)

Receive notification of character data.

The Parser will call this method to report each chunk of character data. SAX parsers may return all contiguous character data in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity so that the Locator provides useful information.

**Parameters**

**content** (*str*) –

**Return type**

*None*

**endElementNS**(*name, qname*)

Signals the end of an element in namespace mode.

The name parameter contains the name of the element type, just as with the startElementNS event.

**Parameters**

**name** (*Tuple[Optional[str], str]*) –

**Return type**

*None*

**endPrefixMapping**(*prefix*)

End the scope of a prefix-URI mapping.

See startPrefixMapping for details. This event will always occur after the corresponding endElement event, but the order of endPrefixMapping events is not otherwise guaranteed.

**Parameters**

**prefix** (*Optional[str]*) –

**Return type**

`None`

**error**(*message*)

**Parameters**

**message** (`str`) –

**Return type**

`NoReturn`

**get\_bnode**(*label*)

**Parameters**

**label** (`str`) –

**Return type**

`BNode`

**ignorableWhitespace**(*content*)

Receive notification of ignorable whitespace in element content.

Validating Parsers must use this method to report each chunk of ignorable whitespace (see the W3C XML 1.0 recommendation, section 2.10): non-validating parsers may also use this method if they are capable of parsing and using content models.

SAX parsers may return all contiguous whitespace in a single chunk, or they may split it into several chunks; however, all of the characters in any single event must come from the same external entity, so that the Locator provides useful information.

**Return type**

`None`

**processingInstruction**(*target*, *data*)

Receive notification of a processing instruction.

The Parser will invoke this method once for each processing instruction found: note that processing instructions may occur before or after the main document element.

A SAX parser should never report an XML declaration (XML 1.0, section 2.8) or a text declaration (XML 1.0, section 4.3.1) using this method.

**Return type**

`None`

**reset**()

**Return type**

`None`

**setDocumentLocator**(*locator*)

Called by the parser to give the application a locator for locating the origin of document events.

SAX parsers are strongly encouraged (though not absolutely required) to supply a locator: if it does so, it must supply the locator to the application by invoking this method before invoking any of the other methods in the DocumentHandler interface.

The locator allows the application to determine the end position of any document-related event, even if the parser is not reporting an error. Typically, the application will use this information for reporting its own errors (such as character content that does not match an application's business rules). The information returned by the locator is probably not sufficient for use with a search engine.

Note that the locator will return correct information only during the invocation of the events in this interface. The application should not attempt to use it at any other time.

**Parameters**

**locator** (*Locator*) –

**startDocument()**

Receive notification of the beginning of a document.

The SAX parser will invoke this method only once, before any other methods in this interface or in DTD-Handler (except for setDocumentLocator).

**Return type**

*None*

**startElementNS**(*name, qname, attrs*)

Signals the start of an element in namespace mode.

The name parameter contains the name of the element type as a (uri, localname) tuple, the qname parameter the raw XML 1.0 name used in the source document, and the attrs parameter holds an instance of the Attributes class containing the attributes of the element.

The uri part of the name tuple is None for elements which have no namespace.

**Parameters**

- **name** (*Tuple[Optional[str], str]*) –
- **attrs** (*AttributesImpl*) –

**Return type**

*None*

**startPrefixMapping**(*prefix, namespace*)

Begin the scope of a prefix-URI Namespace mapping.

The information from this event is not necessary for normal Namespace processing: the SAX XML reader will automatically replace prefixes for element and attribute names when the <http://xml.org/sax/features/namespaces> feature is true (the default).

There are cases, however, when applications need to use prefixes in character data or in attribute values, where they cannot safely be expanded automatically; the start/endPrefixMapping event supplies the information to the application to expand prefixes in those contexts itself, if necessary.

Note that start/endPrefixMapping events are not guaranteed to be properly nested relative to each-other: all startPrefixMapping events will occur before the corresponding startElement event, and all endPrefixMapping events will occur after the corresponding endElement event, but their order is not guaranteed.

**Parameters**

- **prefix** (*Optional[str]*) –
- **namespace** (*str*) –

**Return type**

*None*

**class** rdflib.plugins.parsers.trix.TriXParser

Bases: *Parser*

A parser for TriX. See <http://sw.nokia.com/trix/>

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.parsers.trix', '__doc__':
'A parser for TriX. See http://sw.nokia.com/trix/', '__init__': <function
TriXParser.__init__>, 'parse': <function TriXParser.parse>, '__dict__': <attribute
'__dict__' of 'TriXParser' objects>, '__weakref__': <attribute '__weakref__' of
'TriXParser' objects>, '__annotations__': {}})
```

```
__init__()
```

```
__module__ = 'rdflib.plugins.parsers.trix'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
parse(source, sink, **args)
```

#### Parameters

- **source** (*InputSource*) –
- **sink** (*Graph*) –
- **args** (*Any*) –

#### Return type

*None*

```
rdflib.plugins.parsers.trix.create_parser(store)
```

#### Parameters

**store** (*Store*) –

#### Return type

*XMLReader*

## Module contents

### rdflib.plugins.serializers package

#### Submodules

#### rdflib.plugins.serializers.hext module

HextuplesSerializer RDF graph serializer for RDFLib. See <<https://github.com/ontola/hextuples>> for details about the format.

```
class rdflib.plugins.serializers.hext.HextuplesSerializer(store)
```

Bases: *Serializer*

Serializes RDF graphs to NTriples format.

#### Parameters

**store** (*Union[Graph, ConjunctiveGraph]*) –

```
__init__(store)
```

#### Parameters

**store** (*Union[Graph, ConjunctiveGraph]*) –

```
__module__ = 'rdflib.plugins.serializers.hex'
```

**base:** `Optional[str]`

**encoding:** `str`

**serialize**(*stream*, *base=None*, *encoding='utf-8'*, *\*\*kwargs*)

Abstract method

**Parameters**

- **stream** (`IO[bytes]`) –
- **base** (`Optional[str]`) –
- **encoding** (`Optional[str]`) –

**store:** `Graph`

### rdflib.plugins.serializers.jsonld module

This serializer will output an RDF Graph as a JSON-LD formatted document. See:

<http://json-ld.org/>

Example usage:

```
>>> from rdflib import Graph
>>> testrdf = '''
... @prefix dc: <http://purl.org/dc/terms/> .
... <http://example.org/about>
...   dc:title "Someone's Homepage"@en .
... '''

>>> g = Graph().parse(data=testrdf, format='n3')

>>> print(g.serialize(format='json-ld', indent=4))
[
  {
    "@id": "http://example.org/about",
    "http://purl.org/dc/terms/title": [
      {
        "@language": "en",
        "@value": "Someone's Homepage"
      }
    ]
  }
]
```

**class** `rdflib.plugins.serializers.jsonld.JsonLDSerializer`(*store*)

Bases: `Serializer`

**Parameters**

**store** (`Graph`) –

```
__init__(store)
```

Parameters

**store** (*Graph*) –

```
__module__ = 'rdflib.plugins.serializers.jsonld'
```

**base:** *Optional[str]*

**encoding:** *str*

```
serialize(stream, base=None, encoding=None, **kwargs)
```

Abstract method

Parameters

- **stream** (*IO[bytes]*) –
- **base** (*Optional[str]*) –
- **encoding** (*Optional[str]*) –

**store:** *Graph*

```
rdflib.plugins.serializers.jsonld.from_rdf(graph, context_data=None, base=None,
                                           use_native_types=False, use_rdf_type=False,
                                           auto_compact=False, startnode=None, index=False)
```

### rdflib.plugins.serializers.longturtle module

LongTurtle RDF graph serializer for RDFLib. See <<http://www.w3.org/TeamSubmission/turtle/>> for syntax specification.

This variant, longturtle as opposed to just turtle, makes some small format changes to turtle - the original turtle serializer. It:

- uses PREFIX instead of @prefix
- uses BASE instead of @base
- adds a new line at RDF.type, or ‘a’
- adds a newline and an indent for all triples with more than one object (object list)
- **adds a new line and ‘;’ for the last triple in a set with ‘.’**  
on the start of the next line
- uses default encoding (encode()) is used instead of “latin-1”
- Nicholas Car, 2021

```
class rdflib.plugins.serializers.longturtle.LongTurtleSerializer(store)
```

Bases: *RecursiveSerializer*

```
__init__(store)
```

```
__module__ = 'rdflib.plugins.serializers.longturtle'
```

```
addNamespace(prefix, namespace)
```

**base:** *Optional[str]*

**doList**(*l\_*)

**encoding**: **str**

**endDocument**()

**getQName**(*uri*, *gen\_prefix=True*)

**indentString** = ' '

**isValidList**(*l\_*)

Checks if *l* is a valid RDF list, i.e. no nodes have other properties.

**label**(*node*, *position*)

**objectList**(*objects*)

**p\_default**(*node*, *position*, *newline=False*)

**p\_squared**(*node*, *position*, *newline=False*)

**path**(*node*, *position*, *newline=False*)

**predicateList**(*subject*, *newline=False*)

**preprocessTriple**(*triple*)

**reset**()

**s\_default**(*subject*)

**s\_squared**(*subject*)

**serialize**(*stream*, *base=None*, *encoding=None*, *spacious=None*, *\*\*args*)

Abstract method

**short\_name** = 'longturtle'

**startDocument**()

**statement**(*subject*)

**store**: **Graph**

**verb**(*node*, *newline=False*)

### rdflib.plugins.serializers.n3 module

Notation 3 (N3) RDF graph serializer for RDFS.

**class** `rdflib.plugins.serializers.n3.N3Serializer`(*store*, *parent=None*)

Bases: `TurtleSerializer`

#### Parameters

**store** (`Graph`) –

**\_\_init\_\_**(*store*, *parent=None*)

#### Parameters

**store** (`Graph`) –



```

__module__ = 'rdflib.plugins.serializers.n3'

base: Optional[str]

encoding: str

endDocument()

getQName(uri, gen_prefix=True)

indent(modifier=0)
    Returns indent string multiplied by the depth

p_clause(node, position)

path(node, position, newline=False)

preprocessTriple(triple)

reset()

s_clause(subject)

short_name = 'n3'

statement(subject)

store: Graph

```

### rdflib.plugins.serializers.nquads module

```

class rdflib.plugins.serializers.nquads.NQuadsSerializer(store)
    Bases: Serializer

    Parameters
        store (Graph) –

    __init__(store)

    Parameters
        store (Graph) –

    __module__ = 'rdflib.plugins.serializers.nquads'

    base: Optional[str]

    encoding: str

    serialize(stream, base=None, encoding=None, **args)
        Abstract method

    Parameters
        • stream (IO[bytes]) –
        • base (Optional[str]) –
        • encoding (Optional[str]) –

    store: Graph

```

**rdflib.plugins.serializers.nt module**

```
class rdflib.plugins.serializers.nt.NTSerializer(store)
```

Bases: *Serializer*

Serializes RDF graphs to NTriples format.

**Parameters**

**store** (*Graph*) –

**\_\_init\_\_**(*store*)

**Parameters**

**store** (*Graph*) –

**\_\_module\_\_** = 'rdflib.plugins.serializers.nt'

**base**: *Optional*[*str*]

**encoding**: *str*

**serialize**(*stream*, *base=None*, *encoding='utf-8'*, *\*\*args*)

Abstract method

**Parameters**

- **stream** (*IO*[*bytes*]) –
- **base** (*Optional*[*str*]) –
- **encoding** (*Optional*[*str*]) –

**Return type**

*None*

**store**: *Graph*

**rdflib.plugins.serializers.rdfxml module**

```
class rdflib.plugins.serializers.rdfxml.PrettyXMLSerializer(store, max_depth=3)
```

Bases: *Serializer*

**Parameters**

**store** (*Graph*) –

**\_\_init\_\_**(*store*, *max\_depth=3*)

**Parameters**

**store** (*Graph*) –

**\_\_module\_\_** = 'rdflib.plugins.serializers.rdfxml'

**base**: *Optional*[*str*]

**encoding**: *str*

**predicate**(*predicate*, *object*, *depth=1*)

**serialize**(*stream*, *base=None*, *encoding=None*, *\*\*args*)

Abstract method

**Parameters**

- **stream** (*IO[bytes]*) –
- **base** (*Optional[str]*) –
- **encoding** (*Optional[str]*) –

**store:** *Graph*

**subject**(*subject*, *depth=1*)

**Parameters**

- **subject** (*IdentifiedNode*) –
- **depth** (*int*) –

**class** `rdflib.plugins.serializers.rdfxml.XMLSerializer`(*store*)

Bases: *Serializer*

**Parameters**

**store** (*Graph*) –

**\_\_init\_\_**(*store*)

**Parameters**

**store** (*Graph*) –

**\_\_module\_\_** = `'rdflib.plugins.serializers.rdfxml'`

**base:** *Optional[str]*

**encoding:** *str*

**predicate**(*predicate*, *object*, *depth=1*)

**serialize**(*stream*, *base=None*, *encoding=None*, *\*\*args*)

Abstract method

**Parameters**

- **stream** (*IO[bytes]*) –
- **base** (*Optional[str]*) –
- **encoding** (*Optional[str]*) –

**store:** *Graph*

**subject**(*subject*, *depth=1*)

`rdflib.plugins.serializers.rdfxml.fix`(*val*)

strip off `_`: from nodeIDs... as they are not valid NCNames

**rdflib.plugins.serializers.trig module**

Trig RDF graph serializer for RDFLib. See <<http://www.w3.org/TR/trig/>> for syntax specification.

```
class rdflib.plugins.serializers.trig.TrigSerializer(store)
    Bases: TurtleSerializer

    Parameters
        store (Union[Graph, ConjunctiveGraph]) –

    __init__(store)

    Parameters
        store (Union[Graph, ConjunctiveGraph]) –

    __module__ = 'rdflib.plugins.serializers.trig'

    base: Optional[str]

    encoding: str

    indentString = ' '

    preprocess()

    Return type
        None

    reset()

    Return type
        None

    serialize(stream, base=None, encoding=None, spacious=None, **args)
        Abstract method

        Parameters
            • stream (IO[bytes]) –
            • base (Optional[str]) –
            • encoding (Optional[str]) –
            • spacious (Optional[bool]) –

    short_name = 'trig'

    store: Graph
```

**rdflib.plugins.serializers.trix module**

```
class rdflib.plugins.serializers.trix.TriXSerializer(store)
    Bases: Serializer

    Parameters
        store (Graph) –
```

```

__init__(store)

    Parameters
        store (Graph) –

__module__ = 'rdflib.plugins.serializers.trix'

base: Optional[str]

encoding: str

serialize(stream, base=None, encoding=None, **args)
    Abstract method

    Parameters
        • stream (IO[bytes]) –
        • base (Optional[str]) –
        • encoding (Optional[str]) –

store: Graph

```

### rdflib.plugins.serializers.turtle module

Turtle RDF graph serializer for RDFLib. See <<http://www.w3.org/TeamSubmission/turtle/>> for syntax specification.

```
class rdflib.plugins.serializers.turtle.RecursiveSerializer(store)
```

Bases: *Serializer*

```

__init__(store)

__module__ = 'rdflib.plugins.serializers.turtle'

addNamespace(prefix, uri)

base: Optional[str]

buildPredicateHash(subject)
    Build a hash key by predicate to a list of objects for the given subject

checkSubject(subject)
    Check to see if the subject should be serialized yet

encoding: str

indent(modifier=0)
    Returns indent string multiplied by the depth

indentString = ' '

isDone(subject)
    Return true if subject is serialized

maxDepth = 10

orderSubjects()

```

```
predicateOrder =
[rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#type'),
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#label')]

preprocess()

preprocessTriple(spo)

reset()

roundtrip_prefixes = ()

sortProperties(properties)
    Take a hash from predicate uris to lists of values. Sort the lists of values. Return a sorted list of properties.

store: Graph

subjectDone(subject)
    Mark a subject as done.

topClasses = [rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#Class')]

write(text)
    Write text in given encoding.

class rdflib.plugins.serializers.turtle.TurtleSerializer(store)
    Bases: RecursiveSerializer
    __init__(store)
    __module__ = 'rdflib.plugins.serializers.turtle'
    addNamespace(prefix, namespace)
    base: Optional[str]
    doList(l_)
    encoding: str
    endDocument()
    getQName(uri, gen_prefix=True)
    indentString = ' '
    isValidList(l_)
        Checks if l is a valid RDF list, i.e. no nodes have other properties.
    label(node, position)
    objectList(objects)
    p_default(node, position, newline=False)
    p_squared(node, position, newline=False)
    path(node, position, newline=False)
    predicateList(subject, newline=False)
```

```

preprocessTriple(triple)

reset()

s_default(subject)

s_squared(subject)

serialize(stream, base=None, encoding=None, spacious=None, **args)
    Abstract method

short_name = 'turtle'

startDocument()

statement(subject)

store: Graph

verb(node, newline=False)

```

### rdflib.plugins.serializers.xmlwriter module

```

class rdflib.plugins.serializers.xmlwriter.XMLWriter(stream, namespace_manager, encoding=None,
                                                    decl=1, extra_ns=None)

```

Bases: `object`

```

__dict__ = mappingproxy({'__module__': 'rdflib.plugins.serializers.xmlwriter',
'__init__': <function XMLWriter.__init__>, '_XMLWriter__get_indent': <function
XMLWriter.__get_indent>, 'indent': <property object>,
'_XMLWriter__close_start_tag': <function XMLWriter.__close_start_tag>, 'push':
<function XMLWriter.push>, 'pop': <function XMLWriter.pop>, 'element': <function
XMLWriter.element>, 'namespaces': <function XMLWriter.namespaces>, 'attribute':
<function XMLWriter.attribute>, 'text': <function XMLWriter.text>, 'qname':
<function XMLWriter.qname>, '__dict__': <attribute '__dict__' of 'XMLWriter'
objects>, '__weakref__': <attribute '__weakref__' of 'XMLWriter' objects>,
'__doc__': None, '__annotations__': {}})

```

```

__init__(stream, namespace_manager, encoding=None, decl=1, extra_ns=None)

```

```

__module__ = 'rdflib.plugins.serializers.xmlwriter'

```

```

__weakref__

```

list of weak references to the object (if defined)

```

attribute(uri, value)

```

```

element(uri, content, attributes={})

```

Utility method for adding a complete simple element

```

property indent

```

```

namespaces(namespaces=None)

```

```

pop(uri=None)

```

```

push(uri)

```

**qname**(*uri*)

Compute qname for a uri using our extra namespaces, or the given namespace manager

**text**(*text*)

## Module contents

**rdflib.plugins.shared** package

## Subpackages

**rdflib.plugins.shared.jsonld** package

## Submodules

**rdflib.plugins.shared.jsonld.context** module

Implementation of the JSON-LD Context structure. See:

<http://json-ld.org/>

**class** `rdflib.plugins.shared.jsonld.context.Context`(*source=None, base=None, version=None*)

Bases: `object`

### Parameters

- **source** (`Optional[Any]`) –
- **base** (`Optional[str]`) –
- **version** (`Optional[float]`) –



```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.shared.jsonld.context',
'__init__': <function Context.__init__>, 'base': <property object>, 'subcontext':
<function Context.subcontext>, '_subcontext': <function Context._subcontext>,
'clear': <function Context._clear>, 'get_context_for_term': <function
Context.get_context_for_term>, 'get_context_for_type': <function
Context.get_context_for_type>, 'get_id': <function Context.get_id>, 'get_type':
<function Context.get_type>, 'get_language': <function Context.get_language>,
'get_value': <function Context.get_value>, 'get_graph': <function
Context.get_graph>, 'get_list': <function Context.get_list>, 'get_set': <function
Context.get_set>, 'get_rev': <function Context.get_rev>, '_get': <function
Context._get>, 'get_key': <function Context.get_key>, 'get_keys': <function
Context.get_keys>, 'lang_key': <property object>, 'id_key': <property object>,
'type_key': <property object>, 'value_key': <property object>, 'list_key':
<property object>, 'rev_key': <property object>, 'graph_key': <property object>,
'add_term': <function Context.add_term>, 'find_term': <function
Context.find_term>, 'resolve': <function Context.resolve>, 'resolve_iri':
<function Context.resolve_iri>, 'isblank': <function Context.isblank>, 'expand':
<function Context.expand>, 'shrink_iri': <function Context.shrink_iri>,
'to_symbol': <function Context.to_symbol>, 'load': <function Context.load>,
'_accept_term': <function Context._accept_term>, '_prep_sources': <function
Context._prep_sources>, '_fetch_context': <function Context._fetch_context>,
'_read_source': <function Context._read_source>, '_read_term': <function
Context._read_term>, '_rec_expand': <function Context._rec_expand>, '_prep_expand':
<function Context._prep_expand>, '_get_source_id': <function
Context._get_source_id>, '_term_dict': <function Context._term_dict>, 'to_dict':
<function Context.to_dict>, '__dict__': <attribute '__dict__' of 'Context'
objects>, '__weakref__': <attribute '__weakref__' of 'Context' objects>, '__doc__':
None, '__annotations__': {'version': 'float', 'vocab': 'Optional[str]', '_base':
'Optional[str]', 'terms': 'Dict[str, Any]', '_alias': 'Dict[str, List[str]]',
'_lookup': 'Dict[Tuple[str, Any, Union[Defined, str], bool], Term]', '_prefixes':
'Dict[str, Any]', 'parent': 'Optional[Context]', '_context_cache': 'Dict[str,
Any]'}}})
```

```
__init__(source=None, base=None, version=None)
```

#### Parameters

- **source** ([Optional\[Any\]](#)) –
- **base** ([Optional\[str\]](#)) –
- **version** ([Optional\[float\]](#)) –

```
__module__ = 'rdflib.plugins.shared.jsonld.context'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
add_term(name, idref, coercion=0, container=0, index=None, language=0, reverse=False, context=0,
prefix=None, protected=False)
```

#### Parameters

- **name** ([str](#)) –
- **idref** ([str](#)) –
- **coercion** ([Union\[Defined, str\]](#)) –

- **container** (`Union[Collection[Any], str, Defined]`) –
- **index** (`Union[str, Defined, None]`) –
- **language** (`Union[str, Defined, None]`) –
- **reverse** (`bool`) –
- **context** (`Any`) –
- **prefix** (`Optional[bool]`) –
- **protected** (`bool`) –

**property base:** `Optional[str]`

**expand**(*term\_curie\_or\_iri*, *use\_vocab=True*)

**Parameters**

- **term\_curie\_or\_iri** (`Any`) –
- **use\_vocab** (`bool`) –

**Return type**

`Optional[str]`

**find\_term**(*idref*, *coercion=None*, *container=0*, *language=None*, *reverse=False*)

**Parameters**

- **idref** (`str`) –
- **coercion** (`Union[str, Defined, None]`) –
- **container** (`Union[Defined, str]`) –
- **language** (`Optional[str]`) –
- **reverse** (`bool`) –

**get\_context\_for\_term**(*term*)

**Parameters**

**term** (`Optional[Term]`) –

**Return type**

`Context`

**get\_context\_for\_type**(*node*)

**Parameters**

**node** (`Any`) –

**Return type**

`Optional[Context]`

**get\_graph**(*obj*)

**Parameters**

**obj** (`Dict[str, Any]`) –

**Return type**

`Any`

**get\_id**(*obj*)

**Parameters**

**obj** (`Dict[str, Any]`) –

**Return type**

`Any`

**get\_key**(*key*)

**Parameters**

**key** (`str`) –

**Return type**

`str`

**get\_keys**(*key*)

**Parameters**

**key** (`str`) –

**Return type**

`Generator[str, None, None]`

**get\_language**(*obj*)

**Parameters**

**obj** (`Dict[str, Any]`) –

**Return type**

`Any`

**get\_list**(*obj*)

**Parameters**

**obj** (`Dict[str, Any]`) –

**Return type**

`Any`

**get\_rev**(*obj*)

**Parameters**

**obj** (`Dict[str, Any]`) –

**Return type**

`Any`

**get\_set**(*obj*)

**Parameters**

**obj** (`Dict[str, Any]`) –

**Return type**

`Any`

**get\_type**(*obj*)

**Parameters**

**obj** (`Dict[str, Any]`) –

**Return type**

`Any`

**get\_value**(*obj*)

**Parameters**

**obj** (`Dict[str, Any]`) –

**Return type**

`Any`

**property graph\_key**

**property id\_key**

**isblank**(*ref*)

**Parameters**

**ref** (`str`) –

**Return type**

`bool`

**property lang\_key**

**property list\_key**

**load**(*source*, *base=None*, *referenced\_contexts=None*)

**Parameters**

- **source** (`Union[List[Any], Any, None]`) –
- **base** (`Optional[str]`) –
- **referenced\_contexts** (`Optional[Set[Any]]`) –

**resolve**(*curie\_or\_iri*)

**Parameters**

**curie\_or\_iri** (`str`) –

**Return type**

`str`

**resolve\_iri**(*iri*)

**Parameters**

**iri** (`str`) –

**Return type**

`str`

**property rev\_key**

**shrink\_iri**(*iri*)

**Parameters**

**iri** (`str`) –

**Return type**

`str`

**subcontext**(*source*, *propagate=True*)

**Parameters**

- **source** (`Any`) –

- **propagate** (*bool*) –

**Return type**  
*Context*

**to\_dict()**

Returns a dictionary representation of the context that can be serialized to JSON.

**Return type**  
*Dict[str, Any]*

**Returns**  
a dictionary representation of the context.

**to\_symbol(*iri*)**

**Parameters**  
**iri** (*str*) –

**Return type**  
*Optional[str]*

**property type\_key**

**property value\_key**

**class** `rdflib.plugins.shared.jsonld.context.Defined`

Bases: *int*

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.shared.jsonld.context',
'__dict__': <attribute '__dict__' of 'Defined' objects>, '__doc__': None,
'__annotations__': {}})
```

```
__module__ = 'rdflib.plugins.shared.jsonld.context'
```

**class** `rdflib.plugins.shared.jsonld.context.Term(id, name, type, container, index, language, reverse, context, prefix, protected)`

Bases: *tuple*

**\_\_getnewargs\_\_()**

Return self as a plain tuple. Used by copy and pickle.

```
__module__ = 'rdflib.plugins.shared.jsonld.context'
```

```
static __new__(_cls, id, name, type=0, container=0, index=0, language=0, reverse=False, context=0,
prefix=False, protected=False)
```

Create new instance of Term(*id, name, type, container, index, language, reverse, context, prefix, protected*)

**\_\_repr\_\_()**

Return a nicely formatted representation string

```
__slots__ = ()
```

**container**

Alias for field number 3

**context**

Alias for field number 7

**id**  
Alias for field number 0

**index**  
Alias for field number 4

**language**  
Alias for field number 5

**name**  
Alias for field number 1

**prefix**  
Alias for field number 8

**protected**  
Alias for field number 9

**reverse**  
Alias for field number 6

**type**  
Alias for field number 2

### rdflib.plugins.shared.jsonld.errors module

**exception** rdflib.plugins.shared.jsonld.errors.JSONLDException

Bases: `ValueError`

**\_\_module\_\_** = 'rdflib.plugins.shared.jsonld.errors'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

### rdflib.plugins.shared.jsonld.keys module

### rdflib.plugins.shared.jsonld.util module

rdflib.plugins.shared.jsonld.util.**context\_from\_urlinputsource**(*source*)

Please note that JSON-LD documents served with the application/ld+json media type MUST have all context information, including references to external contexts, within the body of the document. Contexts linked via a <http://www.w3.org/ns/json-ld#context> HTTP Link Header MUST be ignored for such documents.

**Parameters**

**source** (*URLInputSource*) –

**Return type**

`Optional[str]`

rdflib.plugins.shared.jsonld.util.**norm\_url**(*base, url*)

```

>>> norm_url('http://example.org/', '/one')
'http://example.org/one'
>>> norm_url('http://example.org/', '/one#')
'http://example.org/one#'
>>> norm_url('http://example.org/one', 'two')
'http://example.org/two'
>>> norm_url('http://example.org/one/', 'two')
'http://example.org/one/two'
>>> norm_url('http://example.org/', 'http://example.net/one')
'http://example.net/one'
>>> norm_url('http://example.org/', 'http://example.org/one')
'http://example.org/one'

```

**Parameters**

- **base** (*str*) –
- **url** (*str*) –

**Return type***str*

rdflib.plugins.shared.jsonld.util.**source\_to\_json**(*source*)

**Parameters**

**source** (Union[IO[bytes], TextIO, *InputSource*, *str*, bytes, PurePath, None]) –

**Return type***Optional*[Any]

rdflib.plugins.shared.jsonld.util.**split\_iri**(*iri*)

**Parameters**

**iri** (*str*) –

**Return type***Tuple*[*str*, *Optional*[*str*]]**Module contents****Module contents**

rdflib.plugins.sparql package

**Subpackages**

rdflib.plugins.sparql.results package

**Submodules**

rdflib.plugins.sparql.results.csvresults module

```
class rdflib.plugins.sparql.results.csvresults.CSVResultParser
    Bases: ResultParser
    __init__()
    __module__ = 'rdflib.plugins.sparql.results.csvresults'
    convertTerm(t)
        Parameters
            t (str) –
        Return type
            Union[BNode, URIRef, Literal, None]
    parse(source, content_type=None)
        return a Result object
        Parameters
            • source (IO) –
            • content_type (Optional[str]) –
        Return type
            Result
    parseRow(row, v)
        Parameters
            • row (List[str]) –
            • v (List[Variable]) –
        Return type
            Dict[Variable, Union[BNode, URIRef, Literal]]
class rdflib.plugins.sparql.results.csvresults.CSVResultSerializer(result)
    Bases: ResultSerializer
    Parameters
        result (SPARQLResult) –
    __init__(result)
        Parameters
            result (SPARQLResult) –
    __module__ = 'rdflib.plugins.sparql.results.csvresults'
    serialize(stream, encoding='utf-8', **kwargs)
        return a string properly serialized
        Parameters
            • stream (IO) –
            • encoding (str) –
        Return type
            None
```



**serializeTerm**(*term*, *encoding*)

**Parameters**

- **term** (*Optional*[*Identifier*]) –
- **encoding** (*str*) –

**Return type**

*Union*[*str*, *Identifier*]

## rdflib.plugins.sparql.results.graph module

**class** rdflib.plugins.sparql.results.graph.GraphResultParser

Bases: *ResultParser*

**\_\_module\_\_** = 'rdflib.plugins.sparql.results.graph'

**parse**(*source*, *content\_type*)

return a Result object

**Parameters**

- **source** (*IO*) –
- **content\_type** (*Optional*[*str*]) –

**Return type**

*Result*

## rdflib.plugins.sparql.results.jsonresults module

**class** rdflib.plugins.sparql.results.jsonresults.JSONResult(*json*)

Bases: *Result*

**Parameters**

**json** (*Dict*[*str*, *Any*]) –

**\_\_init\_\_**(*json*)

**Parameters**

**json** (*Dict*[*str*, *Any*]) –

**\_\_module\_\_** = 'rdflib.plugins.sparql.results.jsonresults'

**askAnswer**: *Optional*[*bool*]

**graph**: *Optional*['Graph']

**vars**: *Optional*[*List*['Variable']]

variables contained in the result.

**class** rdflib.plugins.sparql.results.jsonresults.JSONResultParser

Bases: *ResultParser*

**\_\_module\_\_** = 'rdflib.plugins.sparql.results.jsonresults'

**parse**(*source*, *content\_type*=None)

return a Result object

**Parameters**

- **source** (IO) –
- **content\_type** (Optional[str]) –

**Return type**

*Result*

**class** rdflib.plugins.sparql.results.jsonresults.JSONResultSerializer(*result*)

Bases: *ResultSerializer*

**Parameters**

**result** (*Result*) –

**\_\_init\_\_**(*result*)

**Parameters**

**result** (*Result*) –

**\_\_module\_\_** = 'rdflib.plugins.sparql.results.jsonresults'

**serialize**(*stream*, *encoding*=None)

return a string properly serialized

**Parameters**

- **stream** (IO) –
- **encoding** (Optional[str]) –

**Return type**

None

rdflib.plugins.sparql.results.jsonresults.parseJsonTerm(*d*)

rdflib object (Literal, URIRef, BNode) for the given json-format dict.

**input is like:**

{ 'type': 'uri', 'value': 'http://famegame.com/2006/01/username' } { 'type': 'literal', 'value': 'drewp' }

**Parameters**

**d** (Dict[str, str]) –

**Return type**

*Identifier*

rdflib.plugins.sparql.results.jsonresults.termToJSON(*self*, *term*)

**Parameters**

- **self** (*JSONResultSerializer*) –
- **term** (Optional[*Identifier*]) –

**Return type**

Optional[Dict[str, str]]

**rdflib.plugins.sparql.results.rdfresults module**

```
class rdflib.plugins.sparql.results.rdfresults.RDFResult(source, **kwargs)
```

Bases: [Result](#)

**Parameters**

- **source** ([Union](#)[[IO](#), [Graph](#)]) –
- **kwargs** ([Any](#)) –

```
__init__(source, **kwargs)
```

**Parameters**

- **source** ([Union](#)[[IO](#), [Graph](#)]) –
- **kwargs** ([Any](#)) –

```
__module__ = 'rdflib.plugins.sparql.results.rdfresults'
```

```
askAnswer: Optional[bool]
```

```
graph: Optional['Graph']
```

```
vars: Optional[List['Variable']]
```

variables contained in the result.

```
class rdflib.plugins.sparql.results.rdfresults.RDFResultParser
```

Bases: [ResultParser](#)

```
__module__ = 'rdflib.plugins.sparql.results.rdfresults'
```

```
parse(source, **kwargs)
```

return a Result object

**Parameters**

- **source** ([Union](#)[[IO](#), [Graph](#)]) –
- **kwargs** ([Any](#)) –

**Return type**

[Result](#)

**rdflib.plugins.sparql.results.tsvresults module**

This implements the Tab Separated SPARQL Result Format

It is implemented with pyparsing, reusing the elements from the SPARQL Parser

```
class rdflib.plugins.sparql.results.tsvresults.TSVResultParser
```

Bases: [ResultParser](#)

```
__module__ = 'rdflib.plugins.sparql.results.tsvresults'
```

```
convertTerm(t)
```

**Parameters**

**t** ([Union](#)[[object](#), [Literal](#), [BNode](#), [CompValue](#), [URIRef](#)]) –

**Return type**`Union[object, BNode, URIRef, Literal, None]`**parse**(*source*, *content\_type*=None)

return a Result object

**Parameters**

- **source** (`IO`) –
- **content\_type** (`Optional[str]`) –

**Return type**`Result`**rdflib.plugins.sparql.results.txtresults module****class** `rdflib.plugins.sparql.results.txtresults.TXTResultSerializer`(*result*)Bases: `ResultSerializer`

A write only QueryResult serializer for text/ascii tables

**Parameters****result** (`Result`) –**\_\_module\_\_** = `'rdflib.plugins.sparql.results.txtresults'`**serialize**(*stream*, *encoding*, *namespace\_manager*=None)

return a text table of query results

**Parameters**

- **stream** (`IO`) –
- **encoding** (`str`) –
- **namespace\_manager** (`Optional[NamespaceManager]`) –

**Return type**`None`**rdflib.plugins.sparql.results.xmlresults module****class** `rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter`(*output*, *encoding*='utf-8')Bases: `object`

Python saxutils-based SPARQL XML Writer

**Parameters**

- **output** (`IO`) –
- **encoding** (`str`) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.results.xmlresults',
'__doc__': '\n Python saxutils-based SPARQL XML Writer\n ', '__init__': <function
SPARQLXMLWriter.__init__>, 'write_header': <function SPARQLXMLWriter.write_header>,
'write_ask': <function SPARQLXMLWriter.write_ask>, 'write_results_header':
<function SPARQLXMLWriter.write_results_header>, 'write_start_result': <function
SPARQLXMLWriter.write_start_result>, 'write_end_result': <function
SPARQLXMLWriter.write_end_result>, 'write_binding': <function
SPARQLXMLWriter.write_binding>, 'close': <function SPARQLXMLWriter.close>,
'__dict__': <attribute '__dict__' of 'SPARQLXMLWriter' objects>, '__weakref__':
<attribute '__weakref__' of 'SPARQLXMLWriter' objects>, '__annotations__': {}})
```

```
__init__(output, encoding='utf-8')
```

#### Parameters

- **output** (*IO*) –
- **encoding** (*str*) –

```
__module__ = 'rdflib.plugins.sparql.results.xmlresults'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
close()
```

#### Return type

*None*

```
write_ask(val)
```

#### Parameters

- **val** (*bool*) –

#### Return type

*None*

```
write_binding(name, val)
```

#### Parameters

- **name** (*Variable*) –
- **val** (*Identifier*) –

#### Return type

*None*

```
write_end_result()
```

#### Return type

*None*

```
write_header(allvarsL)
```

#### Parameters

- **allvarsL** (*Sequence[Variable]*) –

#### Return type

*None*

**write\_results\_header()**

**Return type**

*None*

**write\_start\_result()**

**Return type**

*None*

**class** rdflib.plugins.sparql.results.xmlresults.**XMLResult**(*source, content\_type=None*)

Bases: *Result*

**Parameters**

- **source** (*IO*) –
- **content\_type** (*Optional[str]*) –

**\_\_init\_\_**(*source, content\_type=None*)

**Parameters**

- **source** (*IO*) –
- **content\_type** (*Optional[str]*) –

**\_\_module\_\_** = 'rdflib.plugins.sparql.results.xmlresults'

**askAnswer**: *Optional[bool]*

**graph**: *Optional['Graph']*

**vars**: *Optional[List['Variable']]*

variables contained in the result.

**class** rdflib.plugins.sparql.results.xmlresults.**XMLResultParser**

Bases: *ResultParser*

**\_\_module\_\_** = 'rdflib.plugins.sparql.results.xmlresults'

**parse**(*source, content\_type=None*)

return a Result object

**Parameters**

- **source** (*IO*) –
- **content\_type** (*Optional[str]*) –

**Return type**

*Result*

**class** rdflib.plugins.sparql.results.xmlresults.**XMLResultSerializer**(*result*)

Bases: *ResultSerializer*

**Parameters**

**result** (*Result*) –

**\_\_init\_\_**(*result*)

**Parameters**

**result** (*Result*) –

```
__module__ = 'rdflib.plugins.sparql.results.xmlresults'
```

```
serialize(stream, encoding='utf-8', **kwargs)
```

return a string properly serialized

#### Parameters

- **stream** (*IO*) –
- **encoding** (*str*) –
- **kwargs** (*Any*) –

#### Return type

*None*

```
rdflib.plugins.sparql.results.xmlresults.log = <Logger
```

```
rdflib.plugins.sparql.results.xmlresults (WARNING)>
```

A Parser for SPARQL results in XML:

<http://www.w3.org/TR/rdf-sparql-XMLres/>

Bits and pieces borrowed from: <http://projects.bigasterisk.com/sparqlhttp/>

Authors: Drew Perttula, Gunnar Aastrand Grimnes

```
rdflib.plugins.sparql.results.xmlresults.parseTerm(element)
```

rdflib object (Literal, URIRef, BNode) for the given elementtree element

#### Parameters

**element** (*Element*) –

#### Return type

*Union*[*URIRef*, *Literal*, *BNode*]

## Module contents

Parsers and serializers for SPARQL Result formats

## Submodules

### rdflib.plugins.sparql.aggregates module

```
class rdflib.plugins.sparql.aggregates.Accumulator(aggregation)
```

Bases: *object*

abstract base class for different aggregation functions

#### Parameters

**aggregation** (*CompValue*) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.aggregates',
'__doc__': 'abstract base class for different aggregation functions', '__init__':
<function Accumulator.__init__>, 'dont_care': <function Accumulator.dont_care>,
'use_row': <function Accumulator.use_row>, 'set_value': <function
Accumulator.set_value>, '__dict__': <attribute '__dict__' of 'Accumulator'
objects>, '__weakref__': <attribute '__weakref__' of 'Accumulator' objects>,
'__annotations__': {'get_value': 'Callable[[], Optional[Literal]]', 'update':
"Callable[[FrozenBindings, 'Aggregator'], None]", 'seen': 'Set[Any]'}})
```

**\_\_init\_\_**(*aggregation*)

**Parameters**

**aggregation** (*CompValue*) –

**\_\_module\_\_** = 'rdflib.plugins.sparql.aggregates'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**dont\_care**(*row*)

skips distinct test

**Parameters**

**row** (*FrozenBindings*) –

**Return type**

*bool*

**set\_value**(*bindings*)

sets final value in bindings

**Parameters**

**bindings** (*MutableMapping*[*Variable*, *Identifier*]) –

**Return type**

*None*

**use\_row**(*row*)

tests distinct with set

**Parameters**

**row** (*FrozenBindings*) –

**Return type**

*bool*

**class** rdflib.plugins.sparql.aggregates.**Aggregator**(*aggregations*)

Bases: *object*

combines different Accumulator objects

**Parameters**

**aggregations** (*List*[*CompValue*]) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.aggregates',
'__doc__': 'combines different Accumulator objects', 'accumulator_classes':
{'Aggregate_Count': <class 'rdflib.plugins.sparql.aggregates.Counter'>,
'Aggregate_Sample': <class 'rdflib.plugins.sparql.aggregates.Sample'>,
'Aggregate_Sum': <class 'rdflib.plugins.sparql.aggregates.Sum'>, 'Aggregate_Avg':
<class 'rdflib.plugins.sparql.aggregates.Average'>, 'Aggregate_Min': <class
'rdflib.plugins.sparql.aggregates.Minimum'>, 'Aggregate_Max': <class
'rdflib.plugins.sparql.aggregates.Maximum'>, 'Aggregate_GroupConcat': <class
'rdflib.plugins.sparql.aggregates.GroupConcat'>}, '__init__': <function
Aggregator.__init__>, 'update': <function Aggregator.update>, 'get_bindings':
<function Aggregator.get_bindings>, '__dict__': <attribute '__dict__' of
'Aggregator' objects>, '__weakref__': <attribute '__weakref__' of 'Aggregator'
objects>, '__annotations__': {'bindings': 'Dict[Variable, Identifier]',
'accumulators': 'Dict[str, Accumulator]'}}
```



```

__init__(aggregations)

    Parameters
        aggregations (List[CompValue]) –

__module__ = 'rdflib.plugins.sparql.aggregates'

__weakref__
    list of weak references to the object (if defined)

accumulator_classes = {'Aggregate_Avg': <class
'rdflib.plugins.sparql.aggregates.Average'>, 'Aggregate_Count': <class
'rdflib.plugins.sparql.aggregates.Counter'>, 'Aggregate_GroupConcat': <class
'rdflib.plugins.sparql.aggregates.GroupConcat'>, 'Aggregate_Max': <class
'rdflib.plugins.sparql.aggregates.Maximum'>, 'Aggregate_Min': <class
'rdflib.plugins.sparql.aggregates.Minimum'>, 'Aggregate_Sample': <class
'rdflib.plugins.sparql.aggregates.Sample'>, 'Aggregate_Sum': <class
'rdflib.plugins.sparql.aggregates.Sum'>}

get_bindings()
    calculate and set last values

    Return type
        Mapping[Variable, Identifier]

update(row)
    update all own accumulators

    Parameters
        row (FrozenBindings) –

    Return type
        None

class rdflib.plugins.sparql.aggregates.Average(accumulation)
    Bases: Accumulator

    Parameters
        aggregation (CompValue) –

    __init__(aggregation)

    Parameters
        aggregation (CompValue) –

__module__ = 'rdflib.plugins.sparql.aggregates'

get_value()

    Return type
        Literal

seen: Set[Any]

update(row, aggregator)

    Parameters
        • row (FrozenBindings) –

```

- **aggregator** (*Aggregator*) –

**Return type**

*None*

**class** `rdflib.plugins.sparql.aggregates.Counter`(*aggregation*)

Bases: *Accumulator*

**Parameters**

**aggregation** (*CompValue*) –

**\_\_init\_\_**(*aggregation*)

**Parameters**

**aggregation** (*CompValue*) –

**\_\_module\_\_** = 'rdflib.plugins.sparql.aggregates'

**eval\_full\_row**(*row*)

**Parameters**

**row** (*FrozenBindings*) –

**Return type**

*FrozenBindings*

**eval\_row**(*row*)

**Parameters**

**row** (*FrozenBindings*) –

**Return type**

*Identifier*

**get\_value**()

**Return type**

*Literal*

**seen:** *Set*[*Any*]

**update**(*row*, *aggregator*)

**Parameters**

- **row** (*FrozenBindings*) –

- **aggregator** (*Aggregator*) –

**Return type**

*None*

**use\_row**(*row*)

tests distinct with set

**Parameters**

**row** (*FrozenBindings*) –

**Return type**

*bool*

```

class rdflib.plugins.sparql.aggregates.Extremum(aggregation)
    Bases: Accumulator
    abstract base class for Minimum and Maximum

    Parameters
        aggregation (CompValue) –

    __init__(aggregation)

    Parameters
        aggregation (CompValue) –

    __module__ = 'rdflib.plugins.sparql.aggregates'

    get_value: Callable[[], Optional[Literal]]

    seen: Set[Any]

    set_value(bindings)
        sets final value in bindings

    Parameters
        bindings (MutableMapping[Variable, Identifier]) –

    Return type
        None

    update(row, aggregator)

    Parameters
        • row (FrozenBindings) –
        • aggregator (Aggregator) –

    Return type
        None

class rdflib.plugins.sparql.aggregates.GroupConcat(aggregation)
    Bases: Accumulator
    __init__(aggregation)

    __module__ = 'rdflib.plugins.sparql.aggregates'

    get_value()

    Return type
        Literal

    seen: Set[Any]

    update(row, aggregator)

    Parameters
        • row (FrozenBindings) –
        • aggregator (Aggregator) –

    Return type
        None

```

```
class rdflib.plugins.sparql.aggregates.Maximum(aggregation)
    Bases: Extremum

    Parameters
        aggregation (CompValue) –

    __module__ = 'rdflib.plugins.sparql.aggregates'

    compare(val1, val2)

        Parameters

            • val1 (TypeVar(_ValueT, Variable, BNode, URIRef, Literal)) –

            • val2 (TypeVar(_ValueT, Variable, BNode, URIRef, Literal)) –

        Return type
            TypeVar(_ValueT, Variable, BNode, URIRef, Literal)

    get_value: Callable[[], Optional[Literal]]

    seen: Set[Any]

    value: Any

class rdflib.plugins.sparql.aggregates.Minimum(aggregation)
    Bases: Extremum

    Parameters
        aggregation (CompValue) –

    __module__ = 'rdflib.plugins.sparql.aggregates'

    compare(val1, val2)

        Parameters

            • val1 (TypeVar(_ValueT, Variable, BNode, URIRef, Literal)) –

            • val2 (TypeVar(_ValueT, Variable, BNode, URIRef, Literal)) –

        Return type
            TypeVar(_ValueT, Variable, BNode, URIRef, Literal)

    get_value: Callable[[], Optional[Literal]]

    seen: Set[Any]

    value: Any

class rdflib.plugins.sparql.aggregates.Sample(aggregation)
    Bases: Accumulator

    takes the first eligible value

    __init__(aggregation)

    __module__ = 'rdflib.plugins.sparql.aggregates'

    get_value()

        Return type
            None
```

seen: `Set[Any]`

`update(row, aggregator)`

Parameters

- `row` (`FrozenBindings`) –
- `aggregator` (`Aggregator`) –

Return type

`None`

`class rdflib.plugins.sparql.aggregates.Sum(aggregation)`

Bases: `Accumulator`

Parameters

`aggregation` (`CompValue`) –

`__init__(aggregation)`

Parameters

`aggregation` (`CompValue`) –

`__module__` = `'rdflib.plugins.sparql.aggregates'`

`get_value()`

Return type

`Literal`

seen: `Set[Any]`

`update(row, aggregator)`

Parameters

- `row` (`FrozenBindings`) –
- `aggregator` (`Aggregator`) –

Return type

`None`

`rdflib.plugins.sparql.aggregates.type_safe_numbers(*args)`

Parameters

`args` (`Union[Decimal, float, int]`) –

Return type

`Iterable[Union[float, int]]`

### `rdflib.plugins.sparql.algebra` module

`rdflib.plugins.sparql.algebra.BGP(triples=None)`

Parameters

`triples` (`Optional[List[Tuple[Identifier, Identifier, Identifier]]]`) –

Return type

`CompValue`

**exception** `rdflib.plugins.sparql.algebra.ExpressionNotCoveredException`

Bases: `Exception`

`__module__` = `'rdflib.plugins.sparql.algebra'`

`__weakref__`

list of weak references to the object (if defined)

`rdflib.plugins.sparql.algebra.Extend(p, expr, var)`

**Parameters**

- `p` (`CompValue`) –
- `expr` (`Union[Identifier, Expr]`) –
- `var` (`Variable`) –

**Return type**

`CompValue`

`rdflib.plugins.sparql.algebra.Filter(expr, p)`

**Parameters**

- `expr` (`Expr`) –
- `p` (`CompValue`) –

**Return type**

`CompValue`

`rdflib.plugins.sparql.algebra.Graph(term, graph)`

**Parameters**

- `term` (`Identifier`) –
- `graph` (`CompValue`) –

**Return type**

`CompValue`

`rdflib.plugins.sparql.algebra.Group(p, expr=None)`

**Parameters**

- `p` (`CompValue`) –
- `expr` (`Optional[List[Variable]]`) –

**Return type**

`CompValue`

`rdflib.plugins.sparql.algebra.Join(p1, p2)`

**Parameters**

- `p1` (`CompValue`) –
- `p2` (`Optional[CompValue]`) –

**Return type**

`CompValue`

`rdflib.plugins.sparql.algebra.LeftJoin(p1, p2, expr)`

**Parameters**

- **p1** (*CompValue*) –
- **p2** (*CompValue*) –

**Return type**

*CompValue*

`rdflib.plugins.sparql.algebra.Minus(p1, p2)`

**Parameters**

- **p1** (*CompValue*) –
- **p2** (*CompValue*) –

**Return type**

*CompValue*

`rdflib.plugins.sparql.algebra.OrderBy(p, expr)`

**Parameters**

- **p** (*CompValue*) –
- **expr** (*List[CompValue]*) –

**Return type**

*CompValue*

`rdflib.plugins.sparql.algebra.Project(p, PV)`

**Parameters**

- **p** (*CompValue*) –
- **PV** (*List[Variable]*) –

**Return type**

*CompValue*

**exception** `rdflib.plugins.sparql.algebra.StopTraversal(rv)`

Bases: `Exception`

**Parameters**

**rv** (*bool*) –

`__init__(rv)`

**Parameters**

**rv** (*bool*) –

`__module__` = `'rdflib.plugins.sparql.algebra'`

`__weakref__`

list of weak references to the object (if defined)

`rdflib.plugins.sparql.algebra.ToMultiSet(p)`

**Parameters**

**p** (*Union[List[Dict[Variable, str]], CompValue]*) –

**Return type***CompValue*`rdflib.plugins.sparql.algebra.Union(p1, p2)`**Parameters**

- **p1** (*CompValue*) –
- **p2** (*CompValue*) –

**Return type***CompValue*`rdflib.plugins.sparql.algebra.Values(res)`**Parameters****res** (*List*[*Dict*[*Variable*, *str*]]) –**Return type***CompValue*`rdflib.plugins.sparql.algebra.analyse(n, children)`

Some things can be lazily joined. This propegates whether they can up the tree and sets lazy flags for all joins

**Parameters**

- **n** (*Any*) –
- **children** (*Any*) –

**Return type***bool*`rdflib.plugins.sparql.algebra.collectAndRemoveFilters(parts)`

FILTER expressions apply to the whole group graph pattern in which they appear.

<http://www.w3.org/TR/sparql11-query/#sparqlCollectFilters>

**Parameters****parts** (*List*[*CompValue*]) –**Return type***Optional*[*Expr*]`rdflib.plugins.sparql.algebra.pprintAlgebra(q)`**Return type***None*`rdflib.plugins.sparql.algebra.reorderTriples(l_)`

Reorder triple patterns so that we execute the ones with most bindings first

**Parameters****l\_** (*Iterable*[*Tuple*[*Identifier*, *Identifier*, *Identifier*]]) –**Return type***List*[*Tuple*[*Identifier*, *Identifier*, *Identifier*]]`rdflib.plugins.sparql.algebra.simplify(n)`

Remove joins to empty BGPs

**Parameters****n** (*Any*) –



**Return type**`Optional[CompValue]``rdflib.plugins.sparql.algebra.translate(q)`<http://www.w3.org/TR/sparql11-query/#convertSolMod>**Parameters**`q (CompValue) –`**Return type**`Tuple[Optional[CompValue], List[Variable]]``rdflib.plugins.sparql.algebra.translateAggregates(q, M)`**Parameters**

- `q (CompValue) –`
- `M (CompValue) –`

**Return type**`Tuple[CompValue, List[Tuple[Variable, Variable]]]``rdflib.plugins.sparql.algebra.translateAlgebra(query_algebra)`**Parameters**`query_algebra (Query) –` An algebra returned by the function call `algebra.translateQuery(parse_tree)`.**Return type**`str`**Returns**

The query form generated from the SPARQL 1.1 algebra tree for select queries.

`rdflib.plugins.sparql.algebra.translateExists(e)`Translate the graph pattern used by EXISTS and NOT EXISTS <http://www.w3.org/TR/sparql11-query/#sparqlCollectFilters>**Parameters**`e (Union[Expr, Literal, Variable, URIRef]) –`**Return type**`Union[Expr, Literal, Variable, URIRef]``rdflib.plugins.sparql.algebra.translateGraphGraphPattern(graphPattern)`**Parameters**`graphPattern (CompValue) –`**Return type**`CompValue``rdflib.plugins.sparql.algebra.translateGroupGraphPattern(graphPattern)`<http://www.w3.org/TR/sparql11-query/#convertGraphPattern>**Parameters**`graphPattern (CompValue) –`**Return type**`CompValue`

`rdflib.plugins.sparql.algebra.translateGroupOrUnionGraphPattern(graphPattern)`

**Parameters**

**graphPattern** (*CompValue*) –

**Return type**

*Optional[CompValue]*

`rdflib.plugins.sparql.algebra.translateInlineData(graphPattern)`

**Parameters**

**graphPattern** (*CompValue*) –

**Return type**

*CompValue*

`rdflib.plugins.sparql.algebra.translatePName(p, prologue)`

Expand prefixed/relative URIs

**Parameters**

- **p** (*Union[CompValue, str]*) –
- **prologue** (*Prologue*) –

**Return type**

*Optional[Identifier]*

`rdflib.plugins.sparql.algebra.translatePath(p)`

Translate PropertyPath expressions

**Parameters**

**p** (*Union[CompValue, URIRef]*) –

**Return type**

*Optional[Path]*

`rdflib.plugins.sparql.algebra.translatePrologue(p, base, initNs=None, prologue=None)`

**Parameters**

- **p** (*ParseResults*) –
- **base** (*Optional[str]*) –
- **initNs** (*Optional[Mapping[str, Any]]*) –
- **prologue** (*Optional[Prologue]*) –

**Return type**

*Prologue*

`rdflib.plugins.sparql.algebra.translateQuads(quads)`

**Parameters**

**quads** (*CompValue*) –

**Return type**

*Tuple[List[Tuple[Identifier, Identifier, Identifier]], DefaultDict[str, List[Tuple[Identifier, Identifier, Identifier]]]]*

`rdflib.plugins.sparql.algebra.translateQuery(q, base=None, initNs=None)`

Translate a query-parsetree to a SPARQL Algebra Expression

Return a `rdflib.plugins.sparql.sparql.Query` object

**Parameters**

- **q** (ParseResults) –
- **base** (Optional[str]) –
- **initNs** (Optional[Mapping[str, Any]]) –

**Return type***Query*

```
rdflib.plugins.sparql.algebra.translateUpdate(q, base=None, initNs=None)
```

Returns a list of SPARQL Update Algebra expressions

**Parameters**

- **q** (CompValue) –
- **base** (Optional[str]) –
- **initNs** (Optional[Mapping[str, Any]]) –

**Return type***Update*

```
rdflib.plugins.sparql.algebra.translateUpdate1(u, prologue)
```

**Parameters**

- **u** (CompValue) –
- **prologue** (Prologue) –

**Return type***CompValue*

```
rdflib.plugins.sparql.algebra.translateValues(v)
```

**Parameters**

**v** (CompValue) –

**Return type**

Union[List[Dict[Variable, str]], CompValue]

```
rdflib.plugins.sparql.algebra.traverse(tree, visitPre=<function <lambda>>, visitPost=<function  
<lambda>>, complete=None)
```

Traverse tree, visit each node with visit function visit function may raise StopTraversal to stop traversal if complete!=None, it is returned on complete traversal, otherwise the transformed tree is returned

**Parameters**

- **visitPre** (Callable[[Any], Any]) –
- **visitPost** (Callable[[Any], Any]) –
- **complete** (Optional[bool]) –

**Return type***Any*

```
rdflib.plugins.sparql.algebra.triples(l)
```

**Parameters**

**l** (Union[List[List[Identifier]], List[Tuple[Identifier, Identifier, Identifier]]) –

**Return type**

`List[Tuple[Identifier, Identifier, Identifier]]`

**rdflib.plugins.sparql.datatypes module**

`rdflib.plugins.sparql.datatypes.type_promotion(t1, t2)`

**Parameters**

- **t1** (`URIRef`) –
- **t2** (`Optional[URIRef]`) –

**Return type**

`URIRef`

**rdflib.plugins.sparql.evaluate module**

`rdflib.plugins.sparql.evaluate.evalAggregateJoin(ctx, agg)`

**Parameters**

- **ctx** (`QueryContext`) –
- **agg** (`CompValue`) –

**Return type**

`Generator[FrozenBindings, None, None]`

`rdflib.plugins.sparql.evaluate.evalAskQuery(ctx, query)`

**Parameters**

- **ctx** (`QueryContext`) –
- **query** (`CompValue`) –

**Return type**

`Mapping[str, Union[str, bool]]`

`rdflib.plugins.sparql.evaluate.evalBGP(ctx, bgp)`

A basic graph pattern

**Parameters**

- **ctx** (`QueryContext`) –
- **bgp** (`List[Tuple[Identifier, Identifier, Identifier]]`) –

**Return type**

`Generator[FrozenBindings, None, None]`

`rdflib.plugins.sparql.evaluate.evalConstructQuery(ctx, query)`

**Parameters**

- **ctx** (`QueryContext`) –
- **query** (`CompValue`) –

**Return type**

`Mapping[str, Union[str, Graph]]`

`rdflib.plugins.sparql.evaluate.evalDescribeQuery(ctx, query)`

**Parameters**

**ctx** (*QueryContext*) –

**Return type**

`Dict[str, Union[str, Graph]]`

`rdflib.plugins.sparql.evaluate.evalDistinct(ctx, part)`

**Parameters**

- **ctx** (*QueryContext*) –
- **part** (*CompValue*) –

**Return type**

`Generator[FrozenBindings, None, None]`

`rdflib.plugins.sparql.evaluate.evalExtend(ctx, extend)`

**Parameters**

- **ctx** (*QueryContext*) –
- **extend** (*CompValue*) –

**Return type**

`Generator[FrozenBindings, None, None]`

`rdflib.plugins.sparql.evaluate.evalFilter(ctx, part)`

**Parameters**

- **ctx** (*QueryContext*) –
- **part** (*CompValue*) –

**Return type**

`Generator[FrozenBindings, None, None]`

`rdflib.plugins.sparql.evaluate.evalGraph(ctx, part)`

**Parameters**

- **ctx** (*QueryContext*) –
- **part** (*CompValue*) –

**Return type**

`Generator[FrozenBindings, None, None]`

`rdflib.plugins.sparql.evaluate.evalGroup(ctx, group)`

[http://www.w3.org/TR/sparql11-query/#defn\\_algGroup](http://www.w3.org/TR/sparql11-query/#defn_algGroup)

**Parameters**

- **ctx** (*QueryContext*) –
- **group** (*CompValue*) –

`rdflib.plugins.sparql.evaluate.evalJoin(ctx, join)`

**Parameters**

- **ctx** (*QueryContext*) –

- `join (CompValue)` –

**Return type**`Generator[FrozenDict, None, None]``rdflib.plugins.sparql.evaluate.evalLazyJoin(ctx, join)`

A lazy join will push the variables bound in the first part to the second part, essentially doing the join implicitly hopefully evaluating much fewer triples

**Parameters**

- `ctx (QueryContext)` –
- `join (CompValue)` –

**Return type**`Generator[FrozenBindings, None, None]``rdflib.plugins.sparql.evaluate.evalLeftJoin(ctx, join)`**Parameters**

- `ctx (QueryContext)` –
- `join (CompValue)` –

**Return type**`Generator[FrozenBindings, None, None]``rdflib.plugins.sparql.evaluate.evalMinus(ctx, minus)`**Parameters**

- `ctx (QueryContext)` –
- `minus (CompValue)` –

**Return type**`Generator[FrozenDict, None, None]``rdflib.plugins.sparql.evaluate.evalMultiset(ctx, part)`**Parameters**

- `ctx (QueryContext)` –
- `part (CompValue)` –

`rdflib.plugins.sparql.evaluate.evalOrderBy(ctx, part)`**Parameters**

- `ctx (QueryContext)` –
- `part (CompValue)` –

**Return type**`Generator[FrozenBindings, None, None]``rdflib.plugins.sparql.evaluate.evalPart(ctx, part)`**Parameters**

- `ctx (QueryContext)` –
- `part (CompValue)` –

**Return type**`Any``rdflib.plugins.sparql.evaluate.evalProject(ctx, project)`**Parameters**

- **ctx** (`QueryContext`) –
- **project** (`CompValue`) –

`rdflib.plugins.sparql.evaluate.evalQuery(graph, query, initBindings, base=None)`

**Caution:** This method can access indirectly requested network endpoints, for example, query processing will attempt to access network endpoints specified in SERVICE directives.

When processing untrusted or potentially malicious queries, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

**Parameters**

- **graph** (`Graph`) –
- **query** (`Query`) –
- **initBindings** (`Mapping[str, Identifier]`) –
- **base** (`Optional[str]`) –

**Return type**`Mapping[Any, Any]``rdflib.plugins.sparql.evaluate.evalReduced(ctx, part)`

apply REDUCED to result

REDUCED is not as strict as DISTINCT, but if the incoming rows were sorted it should produce the same result with limited extra memory and time per incoming row.

**Parameters**

- **ctx** (`QueryContext`) –
- **part** (`CompValue`) –

**Return type**`Generator[FrozenBindings, None, None]``rdflib.plugins.sparql.evaluate.evalSelectQuery(ctx, query)`**Parameters**

- **ctx** (`QueryContext`) –
- **query** (`CompValue`) –

**Return type**`Mapping[str, Union[str, List[Variable], Iterable[FrozenDict]]]`

`rdflib.plugins.sparql.evaluate.evalServiceQuery(ctx, part)`

**Parameters**

- **ctx** (*QueryContext*) –
- **part** (*CompValue*) –

`rdflib.plugins.sparql.evaluate.evalSlice(ctx, slice)`

**Parameters**

- **ctx** (*QueryContext*) –
- **slice** (*CompValue*) –

`rdflib.plugins.sparql.evaluate.evalUnion(ctx, union)`

**Parameters**

- **ctx** (*QueryContext*) –
- **union** (*CompValue*) –

**Return type**

*Iterable[FrozenBindings]*

`rdflib.plugins.sparql.evaluate.evalValues(ctx, part)`

**Parameters**

- **ctx** (*QueryContext*) –
- **part** (*CompValue*) –

**Return type**

*Generator[FrozenBindings, None, None]*

## **rdflib.plugins.sparql.evalutils module**

## **rdflib.plugins.sparql.operators module**

`rdflib.plugins.sparql.operators.AdditiveExpression(e, ctx)`

**Parameters**

- **e** (*Expr*) –
- **ctx** (*Union[QueryContext, FrozenBindings]*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_ABS(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-abs>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*



`rdflib.plugins.sparql.operators.Builtin_BNODE(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-bnode>

**Return type**

*BNode*

`rdflib.plugins.sparql.operators.Builtin_BOUND(e, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-bound>

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_CEIL(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-ceil>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_COALESCE(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-coalesce>

**Parameters**

**expr** (*Expr*) –

`rdflib.plugins.sparql.operators.Builtin_CONCAT(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-concat>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_CONTAINS(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-strcontains>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_DATATYPE(e, ctx)`

**Parameters**

**e** (*Expr*) –

**Return type**

`Optional[str]`

`rdflib.plugins.sparql.operators.Builtin_DAY(e, ctx)`

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_ENCODE_FOR_URI(expr, ctx)`

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_EXISTS(e, ctx)`

**Parameters**

- **e** (*Expr*) –
- **ctx** (*FrozenBindings*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_FLOOR(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-floor>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_HOURS(e, ctx)`

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_IF(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-if>

**Parameters**

**expr** (*Expr*) –

`rdflib.plugins.sparql.operators.Builtin_IRI(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-iri>

**Parameters**

- **expr** (*Expr*) –
- **ctx** (*FrozenBindings*) –

**Return type**

*URIRef*

`rdflib.plugins.sparql.operators.Builtin_LANG(e, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-lang>

Returns the language tag of *ltrl*, if it has one. It returns "" if *ltrl* has no language tag. Note that the RDF data model does not include literals with an empty language tag.

**Parameters**

**e** (*Expr*) –

**Return type***Literal*`rdflib.plugins.sparql.operators.Builtin_LANGMATCHES(e, ctx)`<http://www.w3.org/TR/sparql11-query/#func-langMatches>**Parameters****e** (*Expr*) –**Return type***Literal*`rdflib.plugins.sparql.operators.Builtin_LCASE(e, ctx)`**Parameters****e** (*Expr*) –**Return type***Literal*`rdflib.plugins.sparql.operators.Builtin_MD5(expr, ctx)`**Parameters****expr** (*Expr*) –**Return type***Literal*`rdflib.plugins.sparql.operators.Builtin_MINUTES(e, ctx)`**Parameters****e** (*Expr*) –**Return type***Literal*`rdflib.plugins.sparql.operators.Builtin_MONTH(e, ctx)`**Parameters****e** (*Expr*) –**Return type***Literal*`rdflib.plugins.sparql.operators.Builtin_NOW(e, ctx)`<http://www.w3.org/TR/sparql11-query/#func-now>**Parameters****e** (*Expr*) –**Return type***Literal*`rdflib.plugins.sparql.operators.Builtin_RANDOM(expr, ctx)`<http://www.w3.org/TR/sparql11-query/#idp2133952>**Parameters****expr** (*Expr*) –**Return type***Literal*

`rdflib.plugins.sparql.operators.Builtin_REGEX(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-regex> Invokes the XPath fn:matches function to match text against a regular expression pattern. The regular expression language is defined in XQuery 1.0 and XPath 2.0 Functions and Operators section 7.6.1 Regular Expression Syntax

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_REPLACE(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-substr>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_ROUND(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-round>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_SECONDS(e, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-seconds>

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_SHA1(expr, ctx)`

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_SHA256(expr, ctx)`

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_SHA384(expr, ctx)`

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_SHA512(expr, ctx)`

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_STR(e, ctx)`

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_STRAFTER(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-strafter>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_STRBEFORE(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-strbefore>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_STRDT(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-strdt>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_STRENDS(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-strends>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_STRLANG(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-strlang>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_STRLEN(e, ctx)`

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_STRSTARTS(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-strstarts>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_STRUUID(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-strdt>

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_SUBSTR(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-substr>

**Parameters**

**expr** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_TIMEZONE(e, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-timezone>

**Return type**

*Literal*

**Returns**

the timezone part of arg as an xsd:dayTimeDuration.

**Raises**

an error if there is no timezone.

**Parameters**

**e** (*Expr*) –

`rdflib.plugins.sparql.operators.Builtin_TZ(e, ctx)`

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_UCASE(e, ctx)`

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_UUID(expr, ctx)`

<http://www.w3.org/TR/sparql11-query/#func-strdt>

**Parameters**

**expr** (*Expr*) –

**Return type**

*URIRef*

`rdflib.plugins.sparql.operators.Builtin_YEAR(e, ctx)`

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_isBLANK(expr, ctx)`

**Parameters**

- **expr** (*Expr*) –
- **ctx** (*FrozenBindings*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_isIRI(expr, ctx)`

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_isLITERAL(expr, ctx)`

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_isNUMERIC(expr, ctx)`

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.Builtin_sameTerm(e, ctx)`

**Parameters**

**e** (*Expr*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.ConditionalAndExpression(e, ctx)`

**Parameters**

- **e** (*Expr*) –
- **ctx** (*Union[QueryContext, FrozenBindings]*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.ConditionalOrExpression(e, ctx)`

**Parameters**

- **e** (*Expr*) –
- **ctx** (`Union[QueryContext, FrozenBindings]`) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.EBV(rt)`

Effective Boolean Value (EBV)

- If the argument is a typed literal with a datatype of `xsd:boolean`, the EBV is the value of that argument.
- If the argument is a plain literal or a typed literal with a datatype of `xsd:string`, the EBV is false if the operand value has zero length; otherwise the EBV is true.
- If the argument is a numeric type or a typed literal with a datatype derived from a numeric type, the EBV is false if the operand value is NaN or is numerically equal to zero; otherwise the EBV is true.
- All other arguments, including unbound arguments, produce a type error.

**Parameters**

**rt** (`Union[Identifier, SPARQLError, Expr]`) –

**Return type**

*bool*

`rdflib.plugins.sparql.operators.Function(e, ctx)`

Custom functions and casts

**Parameters**

- **e** (*Expr*) –
- **ctx** (*FrozenBindings*) –

**Return type**

*Node*

`rdflib.plugins.sparql.operators.MultiplicativeExpression(e, ctx)`

**Parameters**

- **e** (*Expr*) –
- **ctx** (`Union[QueryContext, FrozenBindings]`) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.RelationalExpression(e, ctx)`

**Parameters**

- **e** (*Expr*) –
- **ctx** (`Union[QueryContext, FrozenBindings]`) –

**Return type**

*Literal*



`rdflib.plugins.sparql.operators.UnaryMinus(expr, ctx)`

**Parameters**

- **expr** (*Expr*) –
- **ctx** (*FrozenBindings*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.UnaryNot(expr, ctx)`

**Parameters**

- **expr** (*Expr*) –
- **ctx** (*FrozenBindings*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.UnaryPlus(expr, ctx)`

**Parameters**

- **expr** (*Expr*) –
- **ctx** (*FrozenBindings*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.and_(*args)`

**Parameters**

**args** (*Expr*) –

**Return type**

*Expr*

`rdflib.plugins.sparql.operators.calculateDuration(obj1, obj2)`

returns the duration Literal between two datetime

**Parameters**

- **obj1** (*Union[date, datetime]*) –
- **obj2** (*Union[date, datetime]*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.operators.calculateFinalDateTime(obj1, dt1, obj2, dt2, operation)`

Calculates the final dateTime/date/time resultant after addition/ subtraction of duration/dayTimeDuration/yearMonthDuration

**Parameters**

- **obj1** (*Union[date, datetime]*) –
- **dt1** (*URIRef*) –
- **obj2** (*Union[Duration, timedelta]*) –
- **dt2** (*URIRef*) –

- **operation** (*str*) –

**Return type**  
*Literal*

`rdflib.plugins.sparql.operators.custom_function(uri, override=False, raw=False)`

Decorator version of `register_custom_function()`.

**Parameters**

- **uri** (*URIRef*) –
- **override** (*bool*) –
- **raw** (*bool*) –

**Return type**  
`Callable[[Callable[[Expr, FrozenBindings], Node]], Callable[[Expr, FrozenBindings], Node]]`

`rdflib.plugins.sparql.operators.date(e)`

**Parameters**

**e** (*Literal*) –

**Return type**  
*date*

`rdflib.plugins.sparql.operators.dateTimeObjects(expr)`

return a dataTime/date/time/duration/dayTimeDuration/yearMonthDuration python objects from a literal

**Parameters**

**expr** (*Literal*) –

**Return type**  
*Any*

`rdflib.plugins.sparql.operators.datetime(e)`

**Parameters**

**e** (*Literal*) –

**Return type**  
*datetime*

`rdflib.plugins.sparql.operators.default_cast(e, ctx)`

**Parameters**

- **e** (*Expr*) –
- **ctx** (*FrozenBindings*) –

**Return type**  
*Literal*

`rdflib.plugins.sparql.operators.isCompatibleDateTimeDatatype(obj1, dt1, obj2, dt2)`

Returns a boolean indicating if first object is compatible with operation(+/-) over second object.

**Parameters**

- **obj1** (*Union*[*date*, *datetime*]) –
- **dt1** (*URIRef*) –
- **obj2** (*Union*[*Duration*, *timedelta*]) –

- **dt2** (*URIRef*) –

**Return type***bool*rdflib.plugins.sparql.operators.**literal**(*s*)**Parameters****s** (*Literal*) –**Return type***Literal*rdflib.plugins.sparql.operators.**not\_**(*arg*)**Return type***Expr*rdflib.plugins.sparql.operators.**numeric**(*expr*)return a number from a literal <http://www.w3.org/TR/xpath20/#promotion>

or TypeError

**Parameters****expr** (*Literal*) –**Return type***Any*rdflib.plugins.sparql.operators.**register\_custom\_function**(*uri, func, override=False, raw=False*)

Register a custom SPARQL function.

By default, the function will be passed the RDF terms in the argument list. If raw is True, the function will be passed an Expression and a Context.

The function must return an RDF term, or raise a SparqlError.

**Parameters**

- **uri** (*URIRef*) –
- **func** (*Callable*[[*Expr*, *FrozenBindings*], *Node*]) –
- **override** (*bool*) –
- **raw** (*bool*) –

**Return type***None*rdflib.plugins.sparql.operators.**simplify**(*expr*)**Parameters****expr** (*Any*) –**Return type***Any*rdflib.plugins.sparql.operators.**string**(*s*)

Make sure the passed thing is a string literal i.e. plain literal, xsd:string literal or lang-tagged literal

**Parameters****s** (*Literal*) –

**Return type***Literal*`rdflib.plugins.sparql.operators.unregister_custom_function(uri, func=None)`

The ‘func’ argument is included for compatibility with existing code. A previous implementation checked that the function associated with the given uri was actually ‘func’, but this is not necessary as the uri should uniquely identify the function.

**Parameters**

- **uri** (*URIRef*) –
- **func** (*Optional[Callable[... Any]]*) –

**Return type***None***rdflib.plugins.sparql.parser module**

SPARQL 1.1 Parser

based on pyparsing

`rdflib.plugins.sparql.parser.expandBNodeTriples(terms)`

expand [ ?p ?o ] syntax for implicit bnodes

**Parameters****terms** (*ParseResults*) –**Return type***List[Any]*`rdflib.plugins.sparql.parser.expandCollection(terms)`

expand ( 1 2 3 ) notation for collections

**Parameters****terms** (*ParseResults*) –**Return type***List[List[Any]]*`rdflib.plugins.sparql.parser.expandTriples(terms)`

Expand ; and , syntax for repeat predicates, subjects

**Parameters****terms** (*ParseResults*) –**Return type***List[Any]*`rdflib.plugins.sparql.parser.expandUnicodeEscapes(q)`

The syntax of the SPARQL Query Language is expressed over code points in Unicode [UNICODE]. The encoding is always UTF-8 [RFC3629]. Unicode code points may also be expressed using an uXXXX (U+0 to U+FFFF) or UXXXXXXXX syntax (for U+10000 onwards) where X is a hexadecimal digit [0-9A-F]

**Parameters****q** (*str*) –**Return type***str*

`rdflib.plugins.sparql.parser.neg(literal)`

**Parameters**

**literal** (*Literal*) –

**Return type**

*Literal*

`rdflib.plugins.sparql.parser.parseQuery(q)`

**Parameters**

**q** (`Union[str, bytes, TextIO, BinaryIO]`) –

**Return type**

`ParseResults`

`rdflib.plugins.sparql.parser.parseUpdate(q)`

**Parameters**

**q** (`Union[str, bytes, TextIO, BinaryIO]`) –

**Return type**

*CompValue*

`rdflib.plugins.sparql.parser.setDataType(terms)`

**Parameters**

**terms** (`Tuple[Any, Optional[str]]`) –

**Return type**

*Literal*

`rdflib.plugins.sparql.parser.setLanguage(terms)`

**Parameters**

**terms** (`Tuple[Any, Optional[str]]`) –

**Return type**

*Literal*

## rdflib.plugins.sparql.parserutils module

`class rdflib.plugins.sparql.parserutils.Comp(name, expr)`

Bases: `TokenConverter`

A pyparsing token for grouping together things with a label Any sub-tokens that are not Params will be ignored.

Returns `CompValue` / `Expr` objects - depending on whether `evalFn` is set.

**Parameters**

**name** (`str`) –

`__abstractmethods__` = `frozenset({})`

`__init__`(*name*, *expr*)

**Parameters**

**name** (`str`) –

`__module__` = `'rdflib.plugins.sparql.parserutils'`

```
__slotnames__ = []
```

```
postParse(instring, loc, tokenList)
```

**Parameters**

- **instring** (`str`) –
- **loc** (`int`) –
- **tokenList** (`ParseResults`) –

**Return type**

`Union[Expr, CompValue]`

```
setEvalFn(evalfn)
```

**Parameters**

**evalfn** (`Callable[[Any, Any], Any]`) –

**Return type**

`Comp`

```
class rdflib.plugins.sparql.parserutils.CompValue(name, **values)
```

Bases: `OrderedDict`

The result of parsing a Comp Any included Params are available as Dict keys or as attributes

**Parameters**

**name** (`str`) –

```
__getattr__(a)
```

**Parameters**

**a** (`str`) –

**Return type**

`Any`

```
__getitem__(a)
```

`x.__getitem__(y) <==> x[y]`

```
__init__(name, **values)
```

**Parameters**

**name** (`str`) –

```
__module__ = 'rdflib.plugins.sparql.parserutils'
```

```
__repr__()
```

Return `repr(self)`.

**Return type**

`str`

```
__str__()
```

Return `str(self)`.

**Return type**

`str`

**clone()**

**Return type**

*CompValue*

**get**(*a*, *variables=False*, *errors=False*)

Return the value for key if key is in the dictionary, else default.

**Parameters**

- **variables** (*bool*) –
- **errors** (*bool*) –

**class** rdflib.plugins.sparql.parserutils.**Expr**(*name*, *evalfn=None*, *\*\*values*)

Bases: *CompValue*

A CompValue that is evaluable

**Parameters**

- **name** (*str*) –
- **evalfn** (*Optional*[*Callable*[[*Any*, *Any*], *Any*]]) –

**\_\_init\_\_**(*name*, *evalfn=None*, *\*\*values*)

**Parameters**

- **name** (*str*) –
- **evalfn** (*Optional*[*Callable*[[*Any*, *Any*], *Any*]]) –

**\_\_module\_\_** = 'rdflib.plugins.sparql.parserutils'

**eval**(*ctx={}*)

**Parameters**

**ctx** (*Any*) –

**Return type**

*Union*[*SPARQLError*, *Any*]

**class** rdflib.plugins.sparql.parserutils.**Param**(*name*, *expr*, *isList=False*)

Bases: *TokenConverter*

A pyparsing token for labelling a part of the parse-tree if isList is true repeat occurrences of ParamList have their values merged in a list

**Parameters**

- **name** (*str*) –
- **isList** (*bool*) –

**\_\_abstractmethods\_\_** = *frozenset*({})

**\_\_init\_\_**(*name*, *expr*, *isList=False*)

**Parameters**

- **name** (*str*) –
- **isList** (*bool*) –

```
__module__ = 'rdflib.plugins.sparql.parserutils'
```

```
__slotnames__ = []
```

```
postParse2(tokenList)
```

Parameters

**tokenList** (`Union[List[Any], ParseResults]`) –

Return type

`ParamValue`

```
class rdflib.plugins.sparql.parserutils.ParamList(name, expr)
```

Bases: `Param`

A shortcut for a Param with isList=True

Parameters

**name** (`str`) –

```
__abstractmethods__ = frozenset({})
```

```
__init__(name, expr)
```

Parameters

**name** (`str`) –

```
__module__ = 'rdflib.plugins.sparql.parserutils'
```

```
__slotnames__ = []
```

```
failAction: typing.Optional[ParseFailAction]
```

```
ignoreExprs: List['ParserElement']
```

```
parseAction: List[ParseAction]
```

```
suppress_warnings_: List[Diagnostics]
```

```
class rdflib.plugins.sparql.parserutils.ParamValue(name, tokenList, isList)
```

Bases: `object`

The result of parsing a Param This just keeps the name/value All cleverness is in the CompValue

Parameters

- **name** (`str`) –

- **tokenList** (`Union[List[Any], ParseResults]`) –

- **isList** (`bool`) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.parserutils',
'__doc__': '\n The result of parsing a Param\n This just keeps the name/value\n All
cleverness is in the CompValue\n ', '__init__': <function ParamValue.__init__>,
'__str__': <function ParamValue.__str__>, '__dict__': <attribute '__dict__' of
'ParamValue' objects>, '__weakref__': <attribute '__weakref__' of 'ParamValue'
objects>, '__annotations__': {}})
```



`__init__(name, tokenList, isList)`

**Parameters**

- **name** (`str`) –
- **tokenList** (`Union[List[Any], ParseResults]`) –
- **isList** (`bool`) –

`__module__ = 'rdflib.plugins.sparql.parserutils'`

`__str__()`

Return str(self).

**Return type**

`str`

`__weakref__`

list of weak references to the object (if defined)

`rdflib.plugins.sparql.parserutils.prettify_parsetree(t, indent="", depth=0)`

**Parameters**

- **t** (`ParseResults`) –
- **indent** (`str`) –
- **depth** (`int`) –

**Return type**

`str`

`rdflib.plugins.sparql.parserutils.value(ctx, val, variables=False, errors=False)`

utility function for evaluating something...

Variables will be looked up in the context Normally, non-bound vars is an error, set variables=True to return unbound vars

Normally, an error raises the error, set errors=True to return error

**Parameters**

- **ctx** (`FrozenBindings`) –
- **val** (`Any`) –
- **variables** (`bool`) –
- **errors** (`bool`) –

**Return type**

`Any`

**rdflib.plugins.sparql.processor module**

Code for tying SPARQL Engine into RDFLib

These should be automatically registered with RDFLib

```
class rdflib.plugins.sparql.processor.SPARQLProcessor(graph)
```

Bases: *Processor*

```
__init__(graph)
```

```
__module__ = 'rdflib.plugins.sparql.processor'
```

```
query(strOrQuery, initBindings={}, initNs={}, base=None, DEBUG=False)
```

Evaluate a query with the given initial bindings, and initial namespaces. The given base is used to resolve relative URIs in the query and will be overridden by any BASE given in the query.

**Caution:** This method can access indirectly requested network endpoints, for example, query processing will attempt to access network endpoints specified in SERVICE directives.

When processing untrusted or potentially malicious queries, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

**Parameters**

- **strOrQuery** (*Union*[*str*, *Query*]) –
- **initBindings** (*Mapping*[*str*, *Identifier*]) –
- **initNs** (*Mapping*[*str*, *Any*]) –
- **base** (*Optional*[*str*]) –
- **DEBUG** (*bool*) –

**Return type**

*Mapping*[*str*, *Any*]

```
class rdflib.plugins.sparql.processor.SPARQLResult(res)
```

Bases: *Result*

**Parameters**

**res** (*Mapping*[*str*, *Any*]) –

```
__init__(res)
```

**Parameters**

**res** (*Mapping*[*str*, *Any*]) –

```
__module__ = 'rdflib.plugins.sparql.processor'
```

**askAnswer:** *Optional*[*bool*]

**graph:** *Optional*['*Graph*']

**vars:** *Optional*[*List*['*Variable*']]

variables contained in the result.

```
class rdflib.plugins.sparql.processor.SPARQLUpdateProcessor(graph)
```

Bases: [UpdateProcessor](#)

```
__init__(graph)
```

```
__module__ = 'rdflib.plugins.sparql.processor'
```

```
update(strOrQuery, initBindings={}, initNs={})
```

**Caution:** This method can access indirectly requested network endpoints, for example, query processing will attempt to access network endpoints specified in SERVICE directives.

When processing untrusted or potentially malicious queries, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib [Security Considerations](#) documentation.

#### Parameters

- **strOrQuery** ([Union](#)[[str](#), [Update](#)]) –
- **initBindings** ([Mapping](#)[[str](#), [Identifier](#)]) –
- **initNs** ([Mapping](#)[[str](#), [Any](#)]) –

#### Return type

[None](#)

```
rdflib.plugins.sparql.processor.prepareQuery(queryString, initNs={}, base=None)
```

Parse and translate a SPARQL Query

#### Parameters

- **queryString** ([str](#)) –
- **initNs** ([Mapping](#)[[str](#), [Any](#)]) –
- **base** ([Optional](#)[[str](#)]) –

#### Return type

[Query](#)

```
rdflib.plugins.sparql.processor.prepareUpdate(updateString, initNs={}, base=None)
```

Parse and translate a SPARQL Update

#### Parameters

- **updateString** ([str](#)) –
- **initNs** ([Mapping](#)[[str](#), [Any](#)]) –
- **base** ([Optional](#)[[str](#)]) –

#### Return type

[Update](#)

```
rdflib.plugins.sparql.processor.processUpdate(graph, updateString, initBindings={}, initNs={},
                                              base=None)
```

Process a SPARQL Update Request returns Nothing on success or raises Exceptions on error

#### Parameters

- **graph** (*Graph*) –
- **updateString** (*str*) –
- **initBindings** (*Mapping[str, Identifier]*) –
- **initNs** (*Mapping[str, Any]*) –
- **base** (*Optional[str]*) –

**Return type**

*None*

## rdflib.plugins.sparql.sparql module

**exception** `rdflib.plugins.sparql.sparql.AlreadyBound`

Bases: *SPARQLError*

Raised when trying to bind a variable that is already bound!

**\_\_init\_\_**()

**\_\_module\_\_** = `'rdflib.plugins.sparql.sparql'`

**class** `rdflib.plugins.sparql.sparql.Bindings`(*outer=None, d=[]*)

Bases: *MutableMapping*

A single level of a stack of variable-value bindings. Each dict keeps a reference to the dict below it, any failed lookup is propagated back

In python 3.3 this could be a `collections.ChainMap`

**Parameters**

**outer** (*Optional[Bindings]*) –

**\_\_abstractmethods\_\_** = `frozenset({})`

**\_\_contains\_\_**(*key*)

**Parameters**

**key** (*Any*) –

**Return type**

*bool*

**\_\_delitem\_\_**(*key*)

**Parameters**

**key** (*str*) –

**Return type**

*None*

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.sparql', '__doc__':
'\n\n A single level of a stack of variable-value bindings.\n Each dict keeps a
reference to the dict below it,\n any failed lookup is propegated back\n\n In python
3.3 this could be a collections.ChainMap\n ', '__init__': <function
Bindings.__init__>, '__getitem__': <function Bindings.__getitem__>, '__contains__':
<function Bindings.__contains__>, '__setitem__': <function Bindings.__setitem__>,
'__delitem__': <function Bindings.__delitem__>, '__len__': <function
Bindings.__len__>, '__iter__': <function Bindings.__iter__>, '__str__': <function
Bindings.__str__>, '__repr__': <function Bindings.__repr__>, '__dict__':
<attribute '__dict__' of 'Bindings' objects>, '__weakref__': <attribute
'__weakref__' of 'Bindings' objects>, '__abstractmethods__': frozenset(),
'__abc_impl': <_abc._abc_data object>, '__annotations__': {'_d': 'Dict[str,
str]'}}})
```

```
__getitem__(key)
```

**Parameters**

**key** (*str*) –

**Return type**

*str*

```
__init__(outer=None, d=[])
```

**Parameters**

**outer** (*Optional[Bindings]*) –

```
__iter__()
```

**Return type**

*Generator[str, None, None]*

```
__len__()
```

**Return type**

*int*

```
__module__ = 'rdflib.plugins.sparql.sparql'
```

```
__repr__()
```

Return repr(self).

**Return type**

*str*

```
__setitem__(key, value)
```

**Parameters**

- **key** (*str*) –
- **value** (*Any*) –

**Return type**

*None*

```
__str__()
```

Return str(self).

**Return type**

*str*

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**class** rdflib.plugins.sparql.sparql.FrozenBindings(*ctx*, \**args*, \*\**kwargs*)

Bases: *FrozenDict*

**Parameters**

**ctx** (*QueryContext*) –

**\_\_abstractmethods\_\_** = frozenset({})

**\_\_getitem\_\_**(*key*)

**Parameters**

**key** (*Union*[*Identifier*, *str*]) –

**Return type**

*Identifier*

**\_\_init\_\_**(*ctx*, \**args*, \*\**kwargs*)

**Parameters**

**ctx** (*QueryContext*) –

**\_\_module\_\_** = 'rdflib.plugins.sparql.sparql'

**property** bnodes: *Mapping*[*Identifier*, *BNode*]

**forget**(*before*, *\_except*=None)

return a frozen dict only of bindings made in self since before

**Parameters**

• **before** (*QueryContext*) –

• **\_except** (*Optional*[*Container*[*Variable*]]) –

**Return type**

*FrozenBindings*

**merge**(*other*)

**Parameters**

**other** (*Mapping*[*Identifier*, *Identifier*]) –

**Return type**

*FrozenBindings*

**property** now: *datetime*

**project**(*vars*)

**Parameters**

**vars** (*Container*[*Variable*]) –

**Return type**

*FrozenBindings*

**property** prologue: *Optional*[*Prologue*]

**remember**(*these*)

return a frozen dict only of bindings in these

**Return type**

*FrozenBindings*

**class** rdflib.plugins.sparql.sparql.FrozenDict(\*args, \*\*kwargs)

Bases: *Mapping*

An immutable hashable dict

Taken from <http://stackoverflow.com/a/2704866/81121>

**Parameters**

- **args** (*Any*) –
- **kwargs** (*Any*) –

**\_\_abstractmethods\_\_** = frozenset({})

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.sparql', '__doc__':
'\n An immutable hashable dict\n\n Taken from
http://stackoverflow.com/a/2704866/81121\n\n ', '__init__': <function
FrozenDict.__init__>, '__iter__': <function FrozenDict.__iter__>, '__len__':
<function FrozenDict.__len__>, '__getitem__': <function FrozenDict.__getitem__>,
'__hash__': <function FrozenDict.__hash__>, 'project': <function
FrozenDict.project>, 'disjointDomain': <function FrozenDict.disjointDomain>,
'compatible': <function FrozenDict.compatible>, 'merge': <function
FrozenDict.merge>, '__str__': <function FrozenDict.__str__>, '__repr__': <function
FrozenDict.__repr__>, '__dict__': <attribute '__dict__' of 'FrozenDict' objects>,
'__weakref__': <attribute '__weakref__' of 'FrozenDict' objects>,
'__abstractmethods__': frozenset(), '_abc_impl': <_abc._abc_data object>,
'__annotations__': {'_d': 'Dict[Identifier, Identifier]', '_hash':
'Optional[int]'}})
```

**\_\_getitem\_\_**(*key*)

**Parameters**

**key** (*Identifier*) –

**Return type**

*Identifier*

**\_\_hash\_\_**()

Return hash(self).

**Return type**

*int*

**\_\_init\_\_**(\*args, \*\*kwargs)

**Parameters**

- **args** (*Any*) –
- **kwargs** (*Any*) –

**\_\_iter\_\_**()

`__len__()`

Return type

`int`

`__module__ = 'rdflib.plugins.sparql.sparql'`

`__repr__()`

Return repr(self).

Return type

`str`

`__str__()`

Return str(self).

Return type

`str`

`__weakref__`

list of weak references to the object (if defined)

`compatible(other)`

Parameters

**other** (`Mapping`[`Identifier`, `Identifier`]) –

Return type

`bool`

`disjointDomain(other)`

Parameters

**other** (`Mapping`[`Identifier`, `Identifier`]) –

Return type

`bool`

`merge(other)`

Parameters

**other** (`Mapping`[`Identifier`, `Identifier`]) –

Return type

`FrozenDict`

`project(vars)`

Parameters

**vars** (`Container`[`Variable`]) –

Return type

`FrozenDict`

**exception** `rdflib.plugins.sparql.sparql.NotBoundError(msg=None)`

Bases: `SPARQLError`

Parameters

**msg** (`Optional`[`str`]) –



```
__init__(msg=None)
```

**Parameters**

**msg** (`Optional[str]`) –

```
__module__ = 'rdflib.plugins.sparql.sparql'
```

```
class rdflib.plugins.sparql.sparql.Prologue
```

Bases: `object`

A class for holding prefixing bindings and base URI information

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.sparql', '__doc__':
'\n A class for holding prefixing bindings and base URI information\n ', '__init__':
<function Prologue.__init__>, 'resolvePName': <function Prologue.resolvePName>,
'bind': <function Prologue.bind>, 'absolutize': <function Prologue.absolutize>,
'__dict__': <attribute '__dict__' of 'Prologue' objects>, '__weakref__':
<attribute '__weakref__' of 'Prologue' objects>, '__annotations__': {'base':
'Optional[str]'}})
```

```
__init__()
```

```
__module__ = 'rdflib.plugins.sparql.sparql'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
absolutize(iri)
```

Apply BASE / PREFIXes to URIs (and to datatypes in Literals)

TODO: Move resolving URIs to pre-processing

**Parameters**

**iri** (`Union[CompValue, str, None]`) –

**Return type**

`Union[CompValue, str, None]`

```
bind(prefix, uri)
```

**Parameters**

- **prefix** (`Optional[str]`) –
- **uri** (`Any`) –

**Return type**

`None`

```
resolvePName(prefix, localname)
```

**Parameters**

- **prefix** (`Optional[str]`) –
- **localname** (`Optional[str]`) –

**Return type**

`URIRef`

```
class rdflib.plugins.sparql.sparql.Query(prologue, algebra)
```

Bases: `object`

A parsed and translated query

**Parameters**

- **prologue** (*Prologue*) –
- **algebra** (*CompValue*) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.sparql', '__doc__':  
'\n A parsed and translated query\n ', '__init__': <function Query.__init__>,  
'__dict__': <attribute '__dict__' of 'Query' objects>, '__weakref__': <attribute  
'__weakref__' of 'Query' objects>, '__annotations__': {'_original_args':  
'Tuple[str, Mapping[str, str], Optional[str]]'}}
```

```
__init__(prologue, algebra)
```

**Parameters**

- **prologue** (*Prologue*) –
- **algebra** (*CompValue*) –

```
__module__ = 'rdflib.plugins.sparql.sparql'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
class rdflib.plugins.sparql.sparql.QueryContext(graph=None, bindings=None, initBindings=None)
```

Bases: `object`

Query context - passed along when evaluating the query

**Parameters**

- **graph** (*Optional[Graph]*) –
- **bindings** (*Union[Bindings, FrozenBindings, List[Any], None]*) –
- **initBindings** (*Optional[Mapping[str, Identifier]]*) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.sparql', '__doc__':  
'\n Query context - passed along when evaluating the query\n ', '__init__':  
<function QueryContext.__init__>, 'now': <property object>, 'clone': <function  
QueryContext.clone>, 'dataset': <property object>, 'load': <function  
QueryContext.load>, '__getitem__': <function QueryContext.__getitem__>, 'get':  
<function QueryContext.get>, 'solution': <function QueryContext.solution>,  
'__setitem__': <function QueryContext.__setitem__>, 'pushGraph': <function  
QueryContext.pushGraph>, 'push': <function QueryContext.push>, 'clean': <function  
QueryContext.clean>, 'thaw': <function QueryContext.thaw>, '__dict__': <attribute  
'__dict__' of 'QueryContext' objects>, '__weakref__': <attribute '__weakref__' of  
'QueryContext' objects>, '__annotations__': {'graph': 'Optional[Graph]',  
'dataset': 'Optional[ConjunctiveGraph]', 'prologue': 'Optional[Prologue]',  
'now': 'Optional[datetime.datetime]', 'bnodes': 't.MutableMapping[Identifier,  
BNode]'}}
```

```
__getitem__(key)
```

**Parameters**

- **key** (*Union[str, Path]*) –

**Return type**`Union[str, Path, None]``__init__(graph=None, bindings=None, initBindings=None)`**Parameters**

- **graph** (`Optional[Graph]`) –
- **bindings** (`Union[Bindings, FrozenBindings, List[Any], None]`) –
- **initBindings** (`Optional[Mapping[str, Identifier]]`) –

`__module__ = 'rdflib.plugins.sparql.sparql'``__setitem__(key, value)`**Parameters**

- **key** (`str`) –
- **value** (`str`) –

**Return type**`None``__weakref__`

list of weak references to the object (if defined)

`clean()`**Return type**`QueryContext``clone(bindings=None)`**Parameters****bindings** (`Union[Bindings, FrozenBindings, List[Any], None]`) –**Return type**`QueryContext`**property dataset:** `ConjunctiveGraph`

“current dataset

`get(key, default=None)`**Parameters**

- **key** (`str`) –
- **default** (`Optional[Any]`) –

**Return type**`Any``load(source, default=False, **kwargs)`**Parameters**

- **source** (`URIRef`) –
- **default** (`bool`) –
- **kwargs** (`Any`) –

**Return type**

*None*

**property now:** *datetime*

**push()**

**Return type**

*QueryContext*

**pushGraph**(*graph*)

**Parameters**

**graph** (*Optional[Graph]*) –

**Return type**

*QueryContext*

**solution**(*vars=None*)

Return a static copy of the current variable bindings as dict

**Parameters**

**vars** (*Optional[Iterable[Variable]]*) –

**Return type**

*FrozenBindings*

**thaw**(*frozenbindings*)

Create a new read/write query context from the given solution

**Parameters**

**frozenbindings** (*FrozenBindings*) –

**Return type**

*QueryContext*

**exception** `rdflib.plugins.sparql.sparql.SPARQLError`(*msg=None*)

Bases: *Exception*

**Parameters**

**msg** (*Optional[str]*) –

**\_\_init\_\_**(*msg=None*)

**Parameters**

**msg** (*Optional[str]*) –

**\_\_module\_\_** = `'rdflib.plugins.sparql.sparql'`

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**exception** `rdflib.plugins.sparql.sparql.SPARQLTypeError`(*msg*)

Bases: *SPARQLError*

**Parameters**

**msg** (*Optional[str]*) –

**\_\_init\_\_**(*msg*)

**Parameters**

**msg** (*Optional[str]*) –

```

__module__ = 'rdflib.plugins.sparql.sparql'

class rdflib.plugins.sparql.sparql.Update(prologue, algebra)
    Bases: object
    A parsed and translated update

    Parameters
        • prologue (Prologue) –
        • algebra (List[CompValue]) –

    __dict__ = mappingproxy({'__module__': 'rdflib.plugins.sparql.sparql', '__doc__':
'\n A parsed and translated update\n ', '__init__': <function Update.__init__>,
'__dict__': <attribute '__dict__' of 'Update' objects>, '__weakref__': <attribute
'__weakref__' of 'Update' objects>, '__annotations__': {'_original_args':
'Tuple[str, Mapping[str, str], Optional[str]]'}}})

    __init__(prologue, algebra)

    Parameters
        • prologue (Prologue) –
        • algebra (List[CompValue]) –

    __module__ = 'rdflib.plugins.sparql.sparql'

    __weakref__
        list of weak references to the object (if defined)

```

### rdflib.plugins.sparql.update module

Code for carrying out Update Operations

```
rdflib.plugins.sparql.update.evalAdd(ctx, u)
```

add all triples from src to dst

<http://www.w3.org/TR/sparql11-update/#add>

#### Parameters

- ctx (*QueryContext*) –
- u (*CompValue*) –

#### Return type

None

```
rdflib.plugins.sparql.update.evalClear(ctx, u)
```

<http://www.w3.org/TR/sparql11-update/#clear>

#### Parameters

- ctx (*QueryContext*) –
- u (*CompValue*) –

#### Return type

None

`rdflib.plugins.sparql.update.evalCopy(ctx, u)`  
remove all triples from dst add all triples from src to dst  
<http://www.w3.org/TR/sparql11-update/#copy>

**Parameters**

- **ctx** (*QueryContext*) –
- **u** (*CompValue*) –

**Return type**  
None

`rdflib.plugins.sparql.update.evalCreate(ctx, u)`  
<http://www.w3.org/TR/sparql11-update/#create>

**Parameters**

- **ctx** (*QueryContext*) –
- **u** (*CompValue*) –

**Return type**  
None

`rdflib.plugins.sparql.update.evalDeleteData(ctx, u)`  
<http://www.w3.org/TR/sparql11-update/#deleteData>

**Parameters**

- **ctx** (*QueryContext*) –
- **u** (*CompValue*) –

**Return type**  
None

`rdflib.plugins.sparql.update.evalDeleteWhere(ctx, u)`  
<http://www.w3.org/TR/sparql11-update/#deleteWhere>

**Parameters**

- **ctx** (*QueryContext*) –
- **u** (*CompValue*) –

**Return type**  
None

`rdflib.plugins.sparql.update.evalDrop(ctx, u)`  
<http://www.w3.org/TR/sparql11-update/#drop>

**Parameters**

- **ctx** (*QueryContext*) –
- **u** (*CompValue*) –

**Return type**  
None

`rdflib.plugins.sparql.update.evalInsertData(ctx, u)`  
<http://www.w3.org/TR/sparql11-update/#insertData>

**Parameters**

- **ctx** (*QueryContext*) –
- **u** (*CompValue*) –

**Return type**

None

rdflib.plugins.sparql.update.**evalLoad**(ctx, u)<http://www.w3.org/TR/sparql11-update/#load>**Parameters**

- **ctx** (*QueryContext*) –
- **u** (*CompValue*) –

**Return type**

None

rdflib.plugins.sparql.update.**evalModify**(ctx, u)**Parameters**

- **ctx** (*QueryContext*) –
- **u** (*CompValue*) –

**Return type**

None

rdflib.plugins.sparql.update.**evalMove**(ctx, u)

remove all triples from dst add all triples from src to dst remove all triples from src

<http://www.w3.org/TR/sparql11-update/#move>**Parameters**

- **ctx** (*QueryContext*) –
- **u** (*CompValue*) –

**Return type**

None

rdflib.plugins.sparql.update.**evalUpdate**(graph, update, initBindings={})<http://www.w3.org/TR/sparql11-update/#updateLanguage>

‘A request is a sequence of operations [...] Implementations MUST ensure that operations of a single request are executed in a fashion that guarantees the same effects as executing them in lexical order.

Operations all result either in success or failure.

If multiple operations are present in a single request, then a result of failure from any operation MUST abort the sequence of operations, causing the subsequent operations to be ignored.’

This will return None on success and raise Exceptions on error

**Caution:** This method can access indirectly requested network endpoints, for example, query processing will attempt to access network endpoints specified in SERVICE directives.

When processing untrusted or potentially malicious queries, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

**Parameters**

- **graph** (*Graph*) –
- **update** (*Update*) –
- **initBindings** (*Mapping[str, Identifier]*) –

**Return type***None***Module contents**

SPARQL implementation for RDFLib

New in version 4.0.

`rdflib.plugins.sparql.CUSTOM_EVALS = {}`

Custom evaluation functions

These must be functions taking (ctx, part) and raise `NotImplementedError` if they cannot handle a certain part

`rdflib.plugins.sparql.prepareQuery(queryString, initNs={}, base=None)`

Parse and translate a SPARQL Query

**Parameters**

- **queryString** (*str*) –
- **initNs** (*Mapping[str, Any]*) –
- **base** (*Optional[str]*) –

**Return type***Query*

`rdflib.plugins.sparql.prepareUpdate(updateString, initNs={}, base=None)`

Parse and translate a SPARQL Update

**Parameters**

- **updateString** (*str*) –
- **initNs** (*Mapping[str, Any]*) –
- **base** (*Optional[str]*) –

**Return type***Update*

`rdflib.plugins.sparql.processUpdate(graph, updateString, initBindings={}, initNs={}, base=None)`

Process a SPARQL Update Request returns `Nothing` on success or raises `Exceptions` on error

**Parameters**

- **graph** (*Graph*) –
- **updateString** (*str*) –
- **initBindings** (*Mapping[str, Identifier]*) –
- **initNs** (*Mapping[str, Any]*) –
- **base** (*Optional[str]*) –



**Return type**`None`**rdflib.plugins.stores package****Submodules****rdflib.plugins.stores.auditable module**

This wrapper intercepts calls through the store interface and implements thread-safe logging of destructive operations (adds / removes) in reverse. This is persisted on the store instance and the reverse operations are executed In order to return the store to the state it was when the transaction began Since the reverse operations are persisted on the store, the store itself acts as a transaction.

Calls to commit or rollback, flush the list of reverse operations This provides thread-safe atomicity and isolation (assuming concurrent operations occur with different store instances), but no durability (transactions are persisted in memory and wont be available to reverse operations after the system fails): A and I out of ACID.

**class** `rdflib.plugins.stores.auditable.AuditableStore(store)`

Bases: `Store`

**Parameters**

**store** (`Store`) –

**\_\_init\_\_**(store)

identifier: URIRef of the Store. Defaults to CWD configuration: string containing information open can use to connect to datastore.

**Parameters**

**store** (`Store`) –

**\_\_len\_\_**(context=None)

Number of statements in the store. This should only account for non- quoted (asserted) statements if the context is not specified, otherwise it should return the number of statements in the formula or context given.

**Parameters**

**context** (`Optional[Graph]`) – a graph instance to query or None

**\_\_module\_\_** = 'rdflib.plugins.stores.auditable'

**add**(triple, context, quoted=False)

Adds the given statement to a specific context or to the model. The quoted argument is interpreted by formula-aware stores to indicate this statement is quoted/hypothetical It should be an error to not specify a context and have the quoted argument be True. It should also be an error for the quoted argument to be True when the store is not formula-aware.

**Parameters**

- **triple** (`Tuple[Node, Node, Node]`) –
- **context** (`Graph`) –
- **quoted** (`bool`) –

**Return type**`None`

**bind**(*prefix*, *namespace*, *override=True*)

**Parameters**

- **override** (*bool*) – rebind, even if the given namespace is already bound to another prefix.
- **prefix** (*str*) –
- **namespace** (*URIRef*) –

**Return type**

*None*

**close**(*commit\_pending\_transaction=False*)

This closes the database connection. The `commit_pending_transaction` parameter specifies whether to commit all pending transactions before closing (if the store is transactional).

**Parameters**

**commit\_pending\_transaction** (*bool*) –

**Return type**

*None*

**commit**()

**Return type**

*None*

**contexts**(*triple=None*)

Generator over all contexts in the graph. If `triple` is specified, a generator over all contexts the triple is in.

if store is `graph_aware`, may also return empty contexts

**Return type**

*Generator*[*Graph*, *None*, *None*]

**Returns**

a generator over Nodes

**Parameters**

**triple** (*Optional*[*Tuple*[*Node*, *Node*, *Node*]]) –

**destroy**(*configuration*)

This destroys the instance of the store identified by the configuration string.

**Parameters**

**configuration** (*str*) –

**Return type**

*None*

**namespace**(*prefix*)

**Parameters**

**prefix** (*str*) –

**Return type**

*Optional*[*URIRef*]

**namespaces**()

**Return type**

*Iterator*[*Tuple*[*str*, *URIRef*]]

**open**(*configuration*, *create=True*)

Opens the store specified by the configuration string. If create is True a store will be created if it does not already exist. If create is False and a store does not already exist an exception is raised. An exception is also raised if a store exists, but there is insufficient permissions to open the store. This should return one of: VALID\_STORE, CORRUPTED\_STORE, or NO\_STORE

**Parameters**

- **configuration** (*str*) –
- **create** (*bool*) –

**Return type**

*Optional*[*int*]

**prefix**(*namespace*)

**Parameters**

**namespace** (*URIRef*) –

**Return type**

*Optional*[*str*]

**query**(\**args*, \*\**kw*)

If stores provide their own SPARQL implementation, override this.

queryGraph is None, a URIRef or ‘\_\_UNION\_\_’ If None the graph is specified in the query-string/object If URIRef it specifies the graph to query, If ‘\_\_UNION\_\_’ the union of all named graphs should be queried (This is used by ConjunctiveGraphs Values other than None obviously only makes sense for context-aware stores.)

**Parameters**

- **args** (*Any*) –
- **kw** (*Any*) –

**Return type**

*Result*

**remove**(*spo*, *context=None*)

Remove the set of triples matching the pattern from the store

**Parameters**

- **spo** (*Tuple*[*Optional*[*Node*], *Optional*[*Node*], *Optional*[*Node*]]) –
- **context** (*Optional*[*Graph*]) –

**Return type**

*None*

**rollback**()

**Return type**

*None*

**triples**(*triple*, *context=None*)

A generator over all the triples matching the pattern. Pattern can include any objects for used for comparing against nodes in the store, for example, REGEXTerm, URIRef, Literal, BNode, Variable, Graph, QuotedGraph, Date? DateRange?

**Parameters**

- **context** (`Optional[Graph]`) – A conjunctive query can be indicated by either providing a value of `None`, or a specific context can be queries by passing a `Graph` instance (if store is context aware).
- **triple** (`Tuple[Optional[Node], Optional[Node], Optional[Node]]`) –

**Return type**

`Iterator[Tuple[Tuple[Node, Node, Node], Iterator[Optional[Graph]]]]`

## rdflib.plugins.stores.berkeleydb module

**class** `rdflib.plugins.stores.berkeleydb.BerkeleyDB(configuration=None, identifier=None)`

Bases: `Store`

A store that allows for on-disk persistent using BerkeleyDB, a fast key/value DB.

This store implementation used to be known, previous to rdflib 6.0.0 as ‘Sleepycat’ due to that being the then name of the Python wrapper for BerkeleyDB.

This store allows for quads as well as triples. See examples of use in both the *examples.berkeleydb\_example* and *test/test\_store/test\_store\_berkeleydb.py* files.

**NOTE on installation:**

To use this store, you must have BerkeleyDB installed on your system separately to Python (brew install berkeley-db on a Mac) and also have the BerkeleyDB Python wrapper installed (pip install berkeleydb). You may need to install BerkeleyDB Python wrapper like this: `YES_I_HAVE_THE_RIGHT_TO_USE_THIS_BERKELEY_DB_VERSION=1 pip install berkeleydb`

**Parameters**

- **configuration** (`Optional[str]`) –
- **identifier** (`Optional[Identifier]`) –

`__annotations__` = {'db\_env': 'db.DBEnv'}

`__init__` (`configuration=None, identifier=None`)

identifier: URIRef of the Store. Defaults to CWD configuration: string containing information open can use to connect to datastore.

**Parameters**

- **configuration** (`Optional[str]`) –
- **identifier** (`Optional[Identifier]`) –

`__len__` (`context=None`)

Number of statements in the store. This should only account for non- quoted (asserted) statements if the context is not specified, otherwise it should return the number of statements in the formula or context given.

**Parameters**

**context** (`Optional[Graph]`) – a graph instance to query or `None`

**Return type**

`int`

`__module__` = 'rdflib.plugins.stores.berkeleydb'

**add**(triple, context, quoted=False, txn=None)

Add a triple to the store of triples.

**Parameters**

- **triple** (Tuple[Node, Node, Node]) –
- **context** (Graph) –
- **quoted** (bool) –
- **txn** (Optional[Any]) –

**Return type**

None

**add\_graph**(graph)

Add a graph to the store, no effect if the graph already exists. :type graph: *Graph* :param graph: a Graph instance

**Return type**

None

**bind**(prefix, namespace, override=True)

**Parameters**

- **override** (bool) – rebind, even if the given namespace is already bound to another prefix.
- **prefix** (str) –
- **namespace** (URIRef) –

**Return type**

None

**close**(commit\_pending\_transaction=False)

This closes the database connection. The commit\_pending\_transaction parameter specifies whether to commit all pending transactions before closing (if the store is transactional).

**Parameters**

- **commit\_pending\_transaction** (bool) –

**Return type**

None

**context\_aware:** bool = True

**contexts**(triple=None)

Generator over all contexts in the graph. If triple is specified, a generator over all contexts the triple is in. if store is graph\_aware, may also return empty contexts

**Return type**

Generator[Graph, None, None]

**Returns**

a generator over Nodes

**Parameters**

- **triple** (Optional[Tuple[Node, Node, Node]]) –

**db\_env:** db.DBEnv = None

**formula\_aware:** bool = True

**graph\_aware:** bool = True

**property identifier:** Optional[Identifier]

**is\_open()**

**Return type**

bool

**namespace(prefix)**

**Parameters**

**prefix** (str) –

**Return type**

Optional[URIRef]

**namespaces()**

**Return type**

Generator[Tuple[str, URIRef], None, None]

**open(path, create=True)**

Opens the store specified by the configuration string. If create is True a store will be created if it does not already exist. If create is False and a store does not already exist an exception is raised. An exception is also raised if a store exists, but there is insufficient permissions to open the store. This should return one of: VALID\_STORE, CORRUPTED\_STORE, or NO\_STORE

**Parameters**

- **path** (str) –
- **create** (bool) –

**Return type**

Optional[int]

**prefix(namespace)**

**Parameters**

**namespace** (URIRef) –

**Return type**

Optional[str]

**remove(spo, context, txn=None)**

Remove the set of triples matching the pattern from the store

**Parameters**

- **spo** (Tuple[Optional[Node], Optional[Node], Optional[Node]]) –
- **context** (Optional[Graph]) –
- **txn** (Optional[Any]) –

**Return type**

None

**remove\_graph**(*graph*)

Remove a graph from the store, this should also remove all triples in the graph

**Parameters**

- **graphid** – a Graph instance
- **graph** (*Graph*) –

**sync**()

**Return type**

*None*

**transaction\_aware:** *bool* = *False*

**triples**(*spo*, *context=None*, *txn=None*)

A generator over all the triples matching

**Parameters**

- **spo** (*Tuple*[*Optional*[*Node*], *Optional*[*Node*], *Optional*[*Node*]]) –
- **context** (*Optional*[*Graph*]) –
- **txn** (*Optional*[*Any*]) –

**Return type**

*Generator*[*Tuple*[*Tuple*[*Node*, *Node*, *Node*], *Generator*[*Optional*[*Graph*], *None*, *None*]], *None*, *None*]

## rdflib.plugins.stores.concurrent module

**class** *rdflib.plugins.stores.concurrent.ConcurrentStore*(*store*)

Bases: *object*

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.stores.concurrent',
'__init__': <function ConcurrentStore.__init__>, 'add': <function
ConcurrentStore.add>, 'remove': <function ConcurrentStore.remove>, 'triples':
<function ConcurrentStore.triples>, '__len__': <function ConcurrentStore.__len__>,
'__ConcurrentStore__begin_read': <function ConcurrentStore.__begin_read>,
'__ConcurrentStore__end_read': <function ConcurrentStore.__end_read>, '__dict__':
<attribute '__dict__' of 'ConcurrentStore' objects>, '__weakref__': <attribute
'__weakref__' of 'ConcurrentStore' objects>, '__doc__': None, '__annotations__':
{}})
```

**\_\_init\_\_**(*store*)

**\_\_len\_\_**()

**\_\_module\_\_** = 'rdflib.plugins.stores.concurrent'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**add**(*triple*)

**remove**(*triple*)

**triples**(*triple*)

**class** rdflib.plugins.stores.concurrent.**ResponsibleGenerator**(*gen, cleanup*)

Bases: `object`

A generator that will help clean up when it is done being used.

`__del__`()

`__init__`(*gen, cleanup*)

`__iter__`()

`__module__` = 'rdflib.plugins.stores.concurrent'

`__next__`()

`__slots__` = ['cleanup', 'gen']

`cleanup`

`gen`

### rdflib.plugins.stores.memory module

**class** rdflib.plugins.stores.memory.**Memory**(*configuration=None, identifier=None*)

Bases: `Store`

An in memory implementation of a triple store.

Same as SimpleMemory above, but is Context-aware, Graph-aware, and Formula-aware Authors: Ashley Sommer

#### Parameters

- **configuration** (`Optional[str]`) –
- **identifier** (`Optional[Identifier]`) –

`__init__`(*configuration=None, identifier=None*)

`identifier`: URIRef of the Store. Defaults to CWD configuration: string containing information open can use to connect to datastore.

#### Parameters

- **configuration** (`Optional[str]`) –
- **identifier** (`Optional[Identifier]`) –

`__len__`(*context=None*)

Number of statements in the store. This should only account for non- quoted (asserted) statements if the context is not specified, otherwise it should return the number of statements in the formula or context given.

#### Parameters

**context** (`Optional[Graph]`) – a graph instance to query or None

#### Return type

`int`

`__module__` = 'rdflib.plugins.stores.memory'



**add**(triple, context, quoted=False)

Add a triple to the store of triples.

**Parameters**

- **triple** (Tuple[Node, Node, Node]) –
- **context** (Graph) –
- **quoted** (bool) –

**Return type**

None

**add\_graph**(graph)

Add a graph to the store, no effect if the graph already exists. :type graph: *Graph* :param graph: a Graph instance

**Return type**

None

**bind**(prefix, namespace, override=True)

**Parameters**

- **override** (bool) – rebind, even if the given namespace is already bound to another prefix.
- **prefix** (str) –
- **namespace** (URIRef) –

**Return type**

None

**context\_aware:** bool = True

**contexts**(triple=None)

Generator over all contexts in the graph. If triple is specified, a generator over all contexts the triple is in. if store is graph\_aware, may also return empty contexts

**Return type**

Generator[Graph, None, None]

**Returns**

a generator over Nodes

**Parameters**

**triple** (Optional[Tuple[Node, Node, Node]]) –

**formula\_aware:** bool = True

**graph\_aware:** bool = True

**namespace**(prefix)

**Parameters**

**prefix** (str) –

**Return type**

Optional[URIRef]

**namespaces()**

**Return type**

`Iterator[Tuple[str, URIRef]]`

**prefix(namespace)**

**Parameters**

**namespace** (`URIRef`) –

**Return type**

`Optional[str]`

**query(query, initNs, initBindings, queryGraph, \*\*kwargs)**

If stores provide their own SPARQL implementation, override this.

queryGraph is None, a URIRef or ‘\_\_UNION\_\_’ If None the graph is specified in the query-string/object  
If URIRef it specifies the graph to query, If ‘\_\_UNION\_\_’ the union of all named graphs should be queried  
(This is used by ConjunctiveGraphs Values other than None obviously only makes sense for context-aware stores.)

**Parameters**

- **query** (`Union[Query, str]`) –
- **initNs** (`Mapping[str, Any]`) –
- **initBindings** (`Mapping[str, Identifier]`) –
- **queryGraph** (`str`) –

**Return type**

`Result`

**remove(triple\_pattern, context=None)**

Remove the set of triples matching the pattern from the store

**Parameters**

- **triple\_pattern** (`Tuple[Optional[Node], Optional[Node], Optional[Node]]`) –
- **context** (`Optional[Graph]`) –

**Return type**

`None`

**remove\_graph(graph)**

Remove a graph from the store, this should also remove all triples in the graph

**Parameters**

- **graphid** – a Graph instance
- **graph** (`Graph`) –

**Return type**

`None`

**triples(triple\_pattern, context=None)**

A generator over all the triples matching

**Parameters**

- **triple\_pattern** (`Tuple[Optional[Node], Optional[Node], Optional[Node]]`) –
- **context** (`Optional[Graph]`) –

**Return type**

`Generator[Tuple[Tuple[Node, Node, Node], Generator[Optional[Graph], None, None]], None, None]`

**update**(*update*, *initNs*, *initBindings*, *queryGraph*, *\*\*kwargs*)

If stores provide their own (SPARQL) Update implementation, override this.

*queryGraph* is *None*, a *URIRef* or ‘\_\_UNION\_\_’ If *None* the graph is specified in the query-string/object If *URIRef* it specifies the graph to query, If ‘\_\_UNION\_\_’ the union of all named graphs should be queried (This is used by *ConjunctiveGraphs* Values other than *None* obviously only makes sense for context-aware stores.)

**Parameters**

- **update** (*Union[Update, Any]*) –
- **initNs** (*Mapping[str, Any]*) –
- **initBindings** (*Mapping[str, Identifier]*) –
- **queryGraph** (*str*) –

**Return type**

*None*

**class** `rdflib.plugins.stores.memory.SimpleMemory`(*configuration=None*, *identifier=None*)

Bases: *Store*

A fast naive in memory implementation of a triple store.

This triple store uses nested dictionaries to store triples. Each triple is stored in two such indices as follows  $spo[s][p][o] = 1$  and  $pos[p][o][s] = 1$ .

Authors: Michel Pelletier, Daniel Krech, Stefan Niederhauser

**Parameters**

- **configuration** (*Optional[str]*) –
- **identifier** (*Optional[Identifier]*) –

**\_\_init\_\_**(*configuration=None*, *identifier=None*)

*identifier*: *URIRef* of the Store. Defaults to CWD configuration: string containing information open can use to connect to datastore.

**Parameters**

- **configuration** (*Optional[str]*) –
- **identifier** (*Optional[Identifier]*) –

**\_\_len\_\_**(*context=None*)

Number of statements in the store. This should only account for non- quoted (asserted) statements if the context is not specified, otherwise it should return the number of statements in the formula or context given.

**Parameters**

**context** (*Optional[Graph]*) – a graph instance to query or *None*

**Return type**

*int*

**\_\_module\_\_** = 'rdflib.plugins.stores.memory'

**add**(triple, context, quoted=False)

Add a triple to the store of triples.

**Parameters**

- **triple** (Tuple[Node, Node, Node]) –
- **context** (Graph) –
- **quoted** (bool) –

**Return type**

None

**bind**(prefix, namespace, override=True)

**Parameters**

- **override** (bool) – rebind, even if the given namespace is already bound to another prefix.
- **prefix** (str) –
- **namespace** (URIRef) –

**Return type**

None

**namespace**(prefix)

**Parameters**

- **prefix** (str) –

**Return type**

Optional[URIRef]

**namespaces**()

**Return type**

Iterator[Tuple[str, URIRef]]

**prefix**(namespace)

**Parameters**

- **namespace** (URIRef) –

**Return type**

Optional[str]

**query**(query, initNs, initBindings, queryGraph, \*\*kwargs)

If stores provide their own SPARQL implementation, override this.

queryGraph is None, a URIRef or ‘\_\_UNION\_\_’ If None the graph is specified in the query-string/object  
If URIRef it specifies the graph to query, If ‘\_\_UNION\_\_’ the union of all named graphs should be queried  
(This is used by ConjunctiveGraphs Values other than None obviously only makes sense for context-aware stores.)

**Parameters**

- **query** (Union[Query, str]) –
- **initNs** (Mapping[str, Any]) –
- **initBindings** (Mapping[str, Identifier]) –
- **queryGraph** (str) –

- **kwargs** (*Any*) –

**Return type**

*Result*

**remove**(*triple\_pattern*, *context=None*)

Remove the set of triples matching the pattern from the store

**Parameters**

- **triple\_pattern** (*Tuple*[*Optional*[*Node*], *Optional*[*Node*], *Optional*[*Node*]]) –
- **context** (*Optional*[*Graph*]) –

**Return type**

*None*

**triples**(*triple\_pattern*, *context=None*)

A generator over all the triples matching

**Parameters**

- **triple\_pattern** (*Tuple*[*Optional*[*Node*], *Optional*[*Node*], *Optional*[*Node*]]) –
- **context** (*Optional*[*Graph*]) –

**Return type**

*Iterator*[*Tuple*[*Tuple*[*Node*, *Node*, *Node*], *Iterator*[*Optional*[*Graph*]]]

**update**(*update*, *initNs*, *initBindings*, *queryGraph*, *\*\*kwargs*)

If stores provide their own (SPARQL) Update implementation, override this.

*queryGraph* is *None*, a *URIRef* or ‘\_\_UNION\_\_’ If *None* the graph is specified in the query-string/object If *URIRef* it specifies the graph to query, If ‘\_\_UNION\_\_’ the union of all named graphs should be queried (This is used by *ConjunctiveGraphs* Values other than *None* obviously only makes sense for context-aware stores.)

**Parameters**

- **update** (*Union*[*Update*, *str*]) –
- **initNs** (*Mapping*[*str*, *Any*]) –
- **initBindings** (*Mapping*[*str*, *Identifier*]) –
- **queryGraph** (*str*) –
- **kwargs** (*Any*) –

**Return type**

*None*

### rdflib.plugins.stores.regexmatching module

This wrapper intercepts calls through the store interface which make use of the *REGEXTerm* class to represent matches by REGEX instead of literal comparison.

Implemented for stores that don’t support this and essentially provides the support by replacing the *REGEXTerms* by wildcards (*None*) and matching against the results from the store it’s wrapping.

**class** *rdflib.plugins.stores.regexmatching.REGEXMatching*(*storage*)

Bases: *Store*

**\_\_init\_\_**(*storage*)

identifier: URIRef of the Store. Defaults to CWD configuration: string containing information open can use to connect to datastore.

**\_\_len\_\_**(*context=None*)

Number of statements in the store. This should only account for non- quoted (asserted) statements if the context is not specified, otherwise it should return the number of statements in the formula or context given.

**Parameters**

**context** – a graph instance to query or None

**\_\_module\_\_** = 'rdflib.plugins.stores.regexmatching'

**add**(*triple, context, quoted=False*)

Adds the given statement to a specific context or to the model. The quoted argument is interpreted by formula-aware stores to indicate this statement is quoted/hypothetical It should be an error to not specify a context and have the quoted argument be True. It should also be an error for the quoted argument to be True when the store is not formula-aware.

**bind**(*prefix, namespace, override=True*)

**Parameters**

**override** – rebind, even if the given namespace is already bound to another prefix.

**close**(*commit\_pending\_transaction=False*)

This closes the database connection. The commit\_pending\_transaction parameter specifies whether to commit all pending transactions before closing (if the store is transactional).

**commit**()

**contexts**(*triple=None*)

Generator over all contexts in the graph. If triple is specified, a generator over all contexts the triple is in.

if store is graph\_aware, may also return empty contexts

**Returns**

a generator over Nodes

**destroy**(*configuration*)

This destroys the instance of the store identified by the configuration string.

**namespace**(*prefix*)

**namespaces**()

**open**(*configuration, create=True*)

Opens the store specified by the configuration string. If create is True a store will be created if it does not already exist. If create is False and a store does not already exist an exception is raised. An exception is also raised if a store exists, but there is insufficient permissions to open the store. This should return one of: VALID\_STORE, CORRUPTED\_STORE, or NO\_STORE

**prefix**(*namespace*)

**remove**(*triple, context=None*)

Remove the set of triples matching the pattern from the store

**remove\_context**(*identifier*)

**rollback**()

**triples**(*triple*, *context=None*)

A generator over all the triples matching the pattern. Pattern can include any objects for used for comparing against nodes in the store, for example, REGEXTerm, URIRef, Literal, BNode, Variable, Graph, QuotedGraph, Date? DateRange?

#### Parameters

**context** – A conjunctive query can be indicated by either providing a value of None, or a specific context can be queries by passing a Graph instance (if store is context aware).

**class** rdflib.plugins.stores.regexmatching.REGEXTerm(*expr*)

Bases: `str`

REGEXTerm can be used in any term slot and is interpreted as a request to perform a REGEX match (not a string comparison) using the value (pre-compiled) for checking rdf:type matches

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.stores.regexmatching',
'__doc__': '\n REGEXTerm can be used in any term slot and is interpreted as a
request to\n perform a REGEX match (not a string comparison) using the value\n
(pre-compiled) for checking rdf:type matches\n ', '__init__': <function
REGEXTerm.__init__>, '__reduce__': <function REGEXTerm.__reduce__>, '__dict__':
<attribute '__dict__' of 'REGEXTerm' objects>, '__weakref__': <attribute
'__weakref__' of 'REGEXTerm' objects>, '__annotations__': {}})
```

**\_\_init\_\_**(*expr*)

**\_\_module\_\_** = 'rdflib.plugins.stores.regexmatching'

**\_\_reduce\_\_**()

Helper for pickle.

**\_\_weakref\_\_**

list of weak references to the object (if defined)

rdflib.plugins.stores.regexmatching.regexCompareQuad(*quad*, *regexQuad*)

### rdflib.plugins.stores.sparqlconnector module

**class** rdflib.plugins.stores.sparqlconnector.SPARQLConnector(*query\_endpoint=None*,  
*update\_endpoint=None*,  
*returnFormat='xml'*, *method='GET'*,  
*auth=None*, *\*\*kwargs*)

Bases: `object`

this class deals with nitty gritty details of talking to a SPARQL server

#### Parameters

- **query\_endpoint** (Optional[`str`]) –
- **update\_endpoint** (Optional[`str`]) –
- **returnFormat** (`str`) –
- **method** (Literal['GET', 'POST', 'POST\_FORM']) –
- **auth** (Optional[Tuple[`str`, `str`]]) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.plugins.stores.sparqlconnector',
'__doc__': '\n this class deals with nitty gritty details of talking to a SPARQL
server\n ', '__init__': <function SPARQLConnector.__init__>, 'method': <property
object>, 'query': <function SPARQLConnector.query>, 'update': <function
SPARQLConnector.update>, '__dict__': <attribute '__dict__' of 'SPARQLConnector'
objects>, '__weakref__': <attribute '__weakref__' of 'SPARQLConnector' objects>,
'__annotations__': {'method': 'str'}})
```

```
__init__(query_endpoint=None, update_endpoint=None, returnFormat='xml', method='GET', auth=None,
        **kwargs)
```

auth, if present, must be a tuple of (username, password) used for Basic Authentication

Any additional keyword arguments will be passed to to the request, and can be used to setup timeouts etc.

#### Parameters

- `query_endpoint` (Optional[str]) –
- `update_endpoint` (Optional[str]) –
- `returnFormat` (str) –
- `method` (Literal['GET', 'POST', 'POST\_FORM']) –
- `auth` (Optional[Tuple[str, str]]) –

```
__module__ = 'rdflib.plugins.stores.sparqlconnector'
```

```
__weakref__
```

list of weak references to the object (if defined)

**property method:** `str`

```
query(query, default_graph=None, named_graph=None)
```

#### Parameters

- `query` (str) –
- `default_graph` (Optional[str]) –
- `named_graph` (Optional[str]) –

#### Return type

`Result`

```
update(query, default_graph=None, named_graph=None)
```

#### Parameters

- `query` (str) –
- `default_graph` (Optional[str]) –
- `named_graph` (Optional[str]) –

#### Return type

`None`

**exception** `rdflib.plugins.stores.sparqlconnector.SPARQLConnectorException`

Bases: `Exception`

```
__module__ = 'rdflib.plugins.stores.sparqlconnector'
```



**\_\_weakref\_\_**

list of weak references to the object (if defined)

**rdflib.plugins.stores.sparqlstore module**

This is an RDFLib store around Ivan Herman et al.'s SPARQL service wrapper. This was first done in layer-cake, and then ported to RDFLib

```
class rdflib.plugins.stores.sparqlstore.SPARQLStore(query_endpoint=None, sparql11=True,
                                                    context_aware=True, node_to_sparql=<function
                                                    _node_to_sparql>, returnFormat='xml',
                                                    auth=None, **sparqlconnector_kwargs)
```

Bases: *SPARQLConnector*, *Store*

An RDFLib store around a SPARQL endpoint

This is context-aware and should work as expected when a context is specified.

For ConjunctiveGraphs, reading is done from the “default graph”. Exactly what this means depends on your endpoint, because SPARQL does not offer a simple way to query the union of all graphs as it would be expected for a ConjunctiveGraph. This is why we recommend using Dataset instead, which is motivated by the SPARQL 1.1.

Fuseki/TDB has a flag for specifying that the default graph is the union of all graphs (`tdb:unionDefaultGraph` in the Fuseki config).

**Warning:** By default the SPARQL Store does not support blank-nodes!

As blank-nodes act as variables in SPARQL queries, there is no way to query for a particular blank node without using non-standard SPARQL extensions.

See <http://www.w3.org/TR/sparql11-query/#BGPsparqlBNodes>

You can make use of such extensions through the `node_to_sparql` argument. For example if you want to transform `BNode('0001')` into “<bnode:b0001>”, you can use a function like this:

```
>>> def my_bnode_ext(node):
...     if isinstance(node, BNode):
...         return '<bnode:b%s>' % node
...     return _node_to_sparql(node)
>>> store = SPARQLStore('http://dbpedia.org/sparql',
...                     node_to_sparql=my_bnode_ext)
```

You can request a particular result serialization with the `returnFormat` parameter. This is a string that must have a matching plugin registered. Built in is support for `xml`, `json`, `csv`, `tsv` and `application/rdf+xml`.

The underlying `SPARQLConnector` uses the `urllib` library. Any extra `kwargs` passed to the `SPARQLStore` connector are passed to `urllib` when doing HTTP calls. I.e. you have full control of cookies/auth/headers.

Form example:

```
>>> store = SPARQLStore('...my endpoint ...', auth=('user', 'pass'))
```

will use HTTP basic auth.

**Parameters**

- **query\_endpoint** (*Optional*[*str*]) –
- **sparql11** (*bool*) –
- **context\_aware** (*bool*) –
- **node\_to\_sparql** (*Callable*[[*Node*], *str*]) –
- **returnFormat** (*str*) –
- **auth** (*Optional*[*Tuple*[*str*, *str*]]) –

**\_\_init\_\_** (*query\_endpoint=None*, *sparql11=True*, *context\_aware=True*, *node\_to\_sparql=<function \_node\_to\_sparql>*, *returnFormat='xml'*, *auth=None*, *\*\*sparqlconnector\_kwargs*)

*auth*, if present, must be a tuple of (username, password) used for Basic Authentication

Any additional keyword arguments will be passed to to the request, and can be used to setup timeouts etc.

#### Parameters

- **query\_endpoint** (*Optional*[*str*]) –
- **sparql11** (*bool*) –
- **context\_aware** (*bool*) –
- **node\_to\_sparql** (*Callable*[[*Node*], *str*]) –
- **returnFormat** (*str*) –
- **auth** (*Optional*[*Tuple*[*str*, *str*]]) –

**\_\_len\_\_** (*context=None*)

Number of statements in the store. This should only account for non- quoted (asserted) statements if the context is not specified, otherwise it should return the number of statements in the formula or context given.

#### Parameters

**context** (*Optional*[*Graph*]) – a graph instance to query or None

#### Return type

*int*

**\_\_module\_\_** = 'rdflib.plugins.stores.sparqlstore'

**add** (*\_*, *context=None*, *quoted=False*)

Adds the given statement to a specific context or to the model. The quoted argument is interpreted by formula-aware stores to indicate this statement is quoted/hypothetical. It should be an error to not specify a context and have the quoted argument be True. It should also be an error for the quoted argument to be True when the store is not formula-aware.

#### Parameters

- **\_** (*Tuple*[*Node*, *Node*, *Node*]) –
- **context** (*Optional*[*Graph*]) –
- **quoted** (*bool*) –

#### Return type

*None*

**addN** (*quads*)

Adds each item in the list of statements to a specific context. The quoted argument is interpreted by formula-aware stores to indicate this statement is quoted/hypothetical. Note that the default implementation is a redirect to add

**Parameters****quads** (*Iterable*[*Tuple*[*Node*, *Node*, *Node*, *Graph*]]) –**Return type***None***add\_graph**(*graph*)

Add a graph to the store, no effect if the graph already exists. :type graph: *Graph* :param graph: a Graph instance

**Return type***None***bind**(*prefix*, *namespace*, *override=True*)**Parameters**

- **override** (*bool*) – rebind, even if the given namespace is already bound to another prefix.
- **prefix** (*str*) –
- **namespace** (*URIRef*) –

**Return type***None***commit**()**Return type***None***contexts**(*triple=None*)

Iterates over results to “SELECT ?NAME { GRAPH ?NAME { ?s ?p ?o } }” or “SELECT ?NAME { GRAPH ?NAME { } }” if triple is *None*.

Returns instances of this store with the SPARQL wrapper object updated via addNamedGraph(?NAME).

This causes a named-graph-uri key / value pair to be sent over the protocol.

Please note that some SPARQL endpoints are not able to find empty named graphs.

**Parameters****triple** (*Optional*[*Tuple*[*Node*, *Node*, *Node*]]) –**Return type***Generator*[*IdentifiedNode*, *None*, *None*]**create**(*configuration*)**Parameters****configuration** (*str*) –**Return type***None***destroy**(*configuration*)

This destroys the instance of the store identified by the configuration string.

**Parameters****configuration** (*str*) –**Return type***None*

**formula\_aware:** bool = False

**graph\_aware:** bool = True

**namespace**(*prefix*)

**Parameters**

**prefix** (str) –

**Return type**

Optional[URIRef]

**namespaces**()

**Return type**

Iterator[Tuple[str, URIRef]]

**objects**(*subject=None, predicate=None*)

A generator of objects with the given subject and predicate

**Parameters**

- **subject** (Optional[Node]) –
- **predicate** (Optional[Node]) –

**Return type**

Generator[Node, None, None]

**open**(*configuration, create=False*)

This method is included so that calls to this Store via Graph, e.g. Graph(“SPARQLStore”), can set the required parameters

**Parameters**

- **configuration** (str) –
- **create** (bool) –

**Return type**

Optional[int]

**predicate\_objects**(*subject=None*)

A generator of (predicate, object) tuples for the given subject

**Parameters**

**subject** (Optional[Node]) –

**Return type**

Generator[Tuple[Node, Node], None, None]

**predicates**(*subject=None, object=None*)

A generator of predicates with the given subject and object

**Parameters**

- **subject** (Optional[Node]) –
- **object** (Optional[Node]) –

**Return type**

Generator[Node, None, None]

**prefix**(*namespace*)

**Parameters**

**namespace** (*URIRef*) –

**Return type**

*Optional*[*str*]

**query**(*query*, *initNs=None*, *initBindings=None*, *queryGraph=None*, *DEBUG=False*)

If stores provide their own SPARQL implementation, override this.

*queryGraph* is *None*, a *URIRef* or ‘\_\_UNION\_\_’ If *None* the graph is specified in the query-string/object If *URIRef* it specifies the graph to query, If ‘\_\_UNION\_\_’ the union of all named graphs should be queried (This is used by *ConjunctiveGraphs* Values other than *None* obviously only makes sense for context-aware stores.)

**Parameters**

- **query** (*Union*[*Query*, *str*]) –
- **initNs** (*Optional*[*Mapping*[*str*, *Any*]]) –
- **initBindings** (*Optional*[*Mapping*[*str*, *Identifier*]]) –
- **queryGraph** (*Optional*[*str*]) –
- **DEBUG** (*bool*) –

**Return type**

*Result*

**regex\_matching** = 0

**remove**(\_, *context*)

Remove the set of triples matching the pattern from the store

**Parameters**

- **\_** (*Tuple*[*Optional*[*Node*], *Optional*[*Node*], *Optional*[*Node*]]) –
- **context** (*Optional*[*Graph*]) –

**Return type**

*None*

**remove\_graph**(*graph*)

Remove a graph from the store, this should also remove all triples in the graph

**Parameters**

- **graphid** – a *Graph* instance
- **graph** (*Graph*) –

**Return type**

*None*

**rollback**()

**Return type**

*None*

**subject\_objects**(*predicate=None*)

A generator of (subject, object) tuples for the given predicate

**Parameters**

**predicate** (*Optional[Node]*) –

**Return type**

*Generator[Tuple[Node, Node], None, None]*

**subject\_predicates**(*object=None*)

A generator of (subject, predicate) tuples for the given object

**Parameters**

**object** (*Optional[Node]*) –

**Return type**

*Generator[Tuple[Node, Node], None, None]*

**subjects**(*predicate=None, object=None*)

A generator of subjects with the given predicate and object

**Parameters**

- **predicate** (*Optional[Node]*) –

- **object** (*Optional[Node]*) –

**Return type**

*Generator[Node, None, None]*

**transaction\_aware:** `bool = False`

**triples**(*spo, context=None*)

- tuple (**s**, **o**, **p**) the triple used as filter for the SPARQL select. (None, None, None) means anything.
- context **context** the graph effectively calling this method.

Returns a tuple of triples executing essentially a SPARQL like `SELECT ?subj ?pred ?obj WHERE { ?subj ?pred ?obj }`

**context** may include three parameter to refine the underlying query:

- **LIMIT**: an integer to limit the number of results
- **OFFSET**: an integer to enable paging of results
- **ORDERBY**: an instance of `Variable('s')`, `Variable('o')` or `Variable('p')` or, by default, the first 'None' from the given triple

**Warning:**

- Using **LIMIT** or **OFFSET** automatically include **ORDERBY** otherwise this is because the results are retrieved in a not deterministic way (depends on the walking path on the graph)
- Using **OFFSET** without defining **LIMIT** will discard the first **OFFSET** - 1 results

```
a_graph.LIMIT = limit
a_graph.OFFSET = offset
triple_generator = a_graph.triples(mytriple):
# do something
```

(continues on next page)

(continued from previous page)

```
# Removes LIMIT and OFFSET if not required for the next triple() calls
del a_graph.LIMIT
del a_graph.OFFSET
```

**Parameters**

- **spo** (Tuple[Optional[Node], Optional[Node], Optional[Node]]) –
- **context** (Optional[Graph]) –

**Return type**

Iterator[Tuple[Tuple[Node, Node, Node], None]]

**triples\_choices**(\_, context=None)

A variant of triples that can take a list of terms instead of a single term in any slot. Stores can implement this to optimize the response time from the import default ‘fallback’ implementation, which will iterate over each term in the list and dispatch to triples.

**Parameters**

- **\_** (Tuple[Union[Node, List[Node]], Union[Node, List[Node]], Union[Node, List[Node]]]) –
- **context** (Optional[Graph]) –

**Return type**

Generator[Tuple[Tuple[Node, Node, Node], Iterator[Optional[Graph]]], None, None]

**update**(query, initNs={}, initBindings={}, queryGraph=None, DEBUG=False)

If stores provide their own (SPARQL) Update implementation, override this.

queryGraph is None, a URIRef or ‘\_\_UNION\_\_’ If None the graph is specified in the query-string/object If URIRef it specifies the graph to query, If ‘\_\_UNION\_\_’ the union of all named graphs should be queried (This is used by ConjunctiveGraphs Values other than None obviously only makes sense for context-aware stores.)

**Parameters**

- **query** (Union[Update, str]) –
- **initNs** (Dict[str, Any]) –
- **initBindings** (Dict[str, Identifier]) –
- **queryGraph** (Optional[Identifier]) –
- **DEBUG** (bool) –

**Return type**

None

```
class rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore(query_endpoint=None,
                                                         update_endpoint=None, sparql11=True,
                                                         context_aware=True,
                                                         postAsEncoded=True,
                                                         autocommit=True, dirty_reads=False,
                                                         **kws)
```

Bases: *SPARQLStore*

A store using SPARQL queries for reading and SPARQL Update for changes.

This can be context-aware, if so, any changes will be to the given named graph only.

In favor of the SPARQL 1.1 motivated Dataset, we advise against using this with `ConjunctiveGraphs`, as it reads and writes from and to the “default graph”. Exactly what this means depends on the endpoint and can result in confusion.

For Graph objects, everything works as expected.

See the *SPARQLStore* base class for more information.

## Parameters

- `query_endpoint` (Optional[str]) –
- `update_endpoint` (Optional[str]) –
- `sparql11` (bool) –
- `context_aware` (bool) –
- `postAsEncoded` (bool) –
- `autocommit` (bool) –
- `dirty_reads` (bool) –

BLOCK\_END = '}'

```
BLOCK_FINDING_PATTERN = re.compile('(P<block_start>{)|(P<block_end>}|(?P<block_content>(\\'([^\'\\\\]|\\\\\\\\)*)'|\"([^\"]\\\\\\\\|\\\\\\\\\\\\\\\\)*\")|\\'\\'\\'\\'(\\'\\'|\\\\\\\\)?(\\'([^\'\\\\]|\\\\\\\\)*)'|\"\"\"(\"\"\"|\"\"\"?)?([^\"]\\\\\\\\|\\\\\\\\\\\\\\\\)*\"\"\"))(<([>]\"{1}|^\\\\\\\\))
```

BLOCK\_START = '{'

[illegible][illegible]

```
COMMENT = '#[^\x0D\x0A]*(\x0D\x0A|\\Z)'
```

**ESCAPED** = '\\\\.'

```
IRIREF = '<([^>"{}|^\`\\]\\\\\\\\[\\\\x00-\\\\x20])*>'
```

```
STRING_LITERAL1 = "'([^\\"
```

```
STRING_LITERAL2 = '"([^\\"\\\\]|\\\\\\\\.)*"'
```

```
STRING_LITERAL_LONG1 = '''(''|')?([^\\"\\\\]|\\\\\\\\.)*'''
```

```
STRING_LITERAL_LONG2 = '""'(('|"')?([^\\"\\\\]|\\\\\\.)*)''
```

```
String = '\('([^\'\\\\]|\\\\\\.)*\'|\"([^\\"\\\\]|\\\\\\.)*\"|\'\\\'\'(\'|\\\'\\')?([^\'\\\\\|\\\\\\.)*\'\\\'\\'|\"\"\"(\"|\"\")?([^\'\\\\\|\\\\\\.)*\"\"\"'
```



**\_\_init\_\_**(*query\_endpoint=None, update\_endpoint=None, sparql11=True, context\_aware=True, postAsEncoded=True, autocommit=True, dirty\_reads=False, \*\*kwargs*)

:param autocommit if set, the store will commit after every writing operations. If False, we only make queries on the server once commit is called.

:param dirty\_reads if set, we do not commit before reading. So you cannot read what you wrote before manually calling commit.

#### Parameters

- **query\_endpoint** (*Optional[str]*) –
- **update\_endpoint** (*Optional[str]*) –
- **sparql11** (*bool*) –
- **context\_aware** (*bool*) –
- **postAsEncoded** (*bool*) –
- **autocommit** (*bool*) –
- **dirty\_reads** (*bool*) –

**\_\_len\_\_**(*\*args, \*\*kwargs*)

Number of statements in the store. This should only account for non- quoted (asserted) statements if the context is not specified, otherwise it should return the number of statements in the formula or context given.

#### Parameters

- **context** – a graph instance to query or None
- **args** (*Any*) –
- **kwargs** (*Any*) –

#### Return type

*int*

**\_\_module\_\_** = 'rdflib.plugins.stores.sparqlstore'

**add**(*spo, context=None, quoted=False*)

Add a triple to the store of triples.

#### Parameters

- **spo** (*Tuple[Node, Node, Node]*) –
- **context** (*Optional[Graph]*) –
- **quoted** (*bool*) –

#### Return type

*None*

**addN**(*quads*)

Add a list of quads to the store.

#### Parameters

**quads** (*Iterable[Tuple[Node, Node, Node, Graph]]*) –

#### Return type

*None*

**add\_graph**(*graph*)

Add a graph to the store, no effect if the graph already exists. :type graph: *Graph* :param graph: a Graph instance

**Return type***None***commit**()

add(), addN(), and remove() are transactional to reduce overhead of many small edits. Read and update() calls will automatically commit any outstanding edits. This should behave as expected most of the time, except that alternating writes and reads can degenerate to the original call-per-triple situation that originally existed.

**Return type***None***contexts**(\*args, \*\*kwargs)

Iterates over results to “SELECT ?NAME { GRAPH ?NAME { ?s ?p ?o } }” or “SELECT ?NAME { GRAPH ?NAME { } }” if triple is *None*.

Returns instances of this store with the SPARQL wrapper object updated via addNamedGraph(?NAME).

This causes a named-graph-uri key / value pair to be sent over the protocol.

Please note that some SPARQL endpoints are not able to find empty named graphs.

**Parameters**

- **args** (*Any*) –
- **kwargs** (*Any*) –

**Return type***Generator*[*IdentifiedNode*, *None*, *None*]**nsBindings**: Dict[str, Any]**objects**(*subject=None*, *predicate=None*)

A generator of objects with the given subject and predicate

**Parameters**

- **subject** (*Optional*[*Node*]) –
- **predicate** (*Optional*[*Node*]) –

**Return type***Generator*[*Node*, *None*, *None*]**open**(*configuration*, *create=False*)

sets the endpoint URLs for this SPARQLStore

**Parameters**

- **configuration** (*Union*[*str*, *Tuple*[*str*, *str*]]) – either a tuple of (query\_endpoint, update\_endpoint), or a string with the endpoint which is configured as query and update endpoint
- **create** (*bool*) – if True an exception is thrown.

**Return type***None*

**predicate\_objects**(*subject=None*)

A generator of (predicate, object) tuples for the given subject

**Parameters**

**subject** (Optional[Node]) –

**Return type**

Generator[Tuple[Node, Node], None, None]

**predicates**(*subject=None, object=None*)

A generator of predicates with the given subject and object

**Parameters**

• **subject** (Optional[Node]) –

• **object** (Optional[Node]) –

**Return type**

Generator[Node, None, None]

**query**(\*args, \*\*kwargs)

If stores provide their own SPARQL implementation, override this.

queryGraph is None, a URIRef or ‘\_\_UNION\_\_’ If None the graph is specified in the query-string/object If URIRef it specifies the graph to query, If ‘\_\_UNION\_\_’ the union of all named graphs should be queried (This is used by ConjunctiveGraphs Values other than None obviously only makes sense for context-aware stores.)

**Parameters**

• **args** (Any) –

• **kwargs** (Any) –

**Return type**

Result

**remove**(*spo, context*)

Remove a triple from the store

**Parameters**

• **spo** (Tuple[Optional[Node], Optional[Node], Optional[Node]]) –

• **context** (Optional[Graph]) –

**Return type**

None

**remove\_graph**(*graph*)

Remove a graph from the store, this should also remove all triples in the graph

**Parameters**

• **graphid** – a Graph instance

• **graph** (Graph) –

**Return type**

None

**rollback()**

**Return type**

`None`

**setTimeout(*timeout*)**

**Return type**

`None`

**subject\_objects(*predicate=None*)**

A generator of (subject, object) tuples for the given predicate

**Parameters**

**predicate** (`Optional[Node]`) –

**Return type**

`Generator[Tuple[Node, Node], None, None]`

**subject\_predicates(*object=None*)**

A generator of (subject, predicate) tuples for the given object

**Parameters**

**object** (`Optional[Node]`) –

**Return type**

`Generator[Tuple[Node, Node], None, None]`

**subjects(*predicate=None, object=None*)**

A generator of subjects with the given predicate and object

**Parameters**

- **predicate** (`Optional[Node]`) –
- **object** (`Optional[Node]`) –

**Return type**

`Generator[Node, None, None]`

**triples(\*args, \*\*kwargs)**

- tuple (**s**, **o**, **p**) the triple used as filter for the SPARQL select. (None, None, None) means anything.
- context **context** the graph effectively calling this method.

Returns a tuple of triples executing essentially a SPARQL like `SELECT ?subj ?pred ?obj WHERE { ?subj ?pred ?obj }`

**context** may include three parameter to refine the underlying query:

- **LIMIT**: an integer to limit the number of results
- **OFFSET**: an integer to enable paging of results
- **ORDERBY**: an instance of `Variable('s')`, `Variable('o')` or `Variable('p')` or, by default, the first 'None' from the given triple

**Warning:**

- Using **LIMIT** or **OFFSET** automatically include **ORDERBY** otherwise this is because the results are retrieved in a not deterministic way (depends on the walking path on the graph)

- Using OFFSET without defining LIMIT will discard the first OFFSET - 1 results

```
a_graph.LIMIT = limit
a_graph.OFFSET = offset
triple_generator = a_graph.triples(mytriple):
# do something
# Removes LIMIT and OFFSET if not required for the next triple() calls
del a_graph.LIMIT
del a_graph.OFFSET
```

#### Parameters

- **args** (*Any*) –
- **kwargs** (*Any*) –

#### Return type

`Iterator[Tuple[Tuple[Node, Node, Node], None]]`

**update**(*query*, *initNs*={}, *initBindings*={}, *queryGraph*=None, *DEBUG*=False)

Perform a SPARQL Update Query against the endpoint, INSERT, LOAD, DELETE etc. Setting *initNs* adds PREFIX declarations to the beginning of the update. Setting *initBindings* adds inline VALUES to the beginning of every WHERE clause. By the SPARQL grammar, all operations that support variables (namely INSERT and DELETE) require a WHERE clause. Important: *initBindings* fails if the update contains the substring 'WHERE {' which does not denote a WHERE clause, e.g. if it is part of a literal.

#### Context-aware query rewriting

- **When:** If context-awareness is enabled and the graph is not the default graph of the store.
- **Why:** To ensure consistency with the *Memory* store. The graph must accept “local” SPARQL requests (requests with no GRAPH keyword) as if it was the default graph.
- **What is done:** These “local” queries are rewritten by this store. The content of each block of a SPARQL Update operation is wrapped in a GRAPH block except if the block is empty. This basically causes INSERT, INSERT DATA, DELETE, DELETE DATA and WHERE to operate only on the context.
- **Example:** "INSERT DATA { <urn:michel> <urn:likes> <urn:pizza> }" is converted into "INSERT DATA { GRAPH <urn:graph> { <urn:michel> <urn:likes> <urn:pizza> } }".
- **Warning:** Queries are presumed to be “local” but this assumption is **not checked**. For instance, if the query already contains GRAPH blocks, the latter will be wrapped in new GRAPH blocks.
- **Warning:** A simplified grammar is used that should tolerate extensions of the SPARQL grammar. Still, the process may fail in uncommon situations and produce invalid output.

#### Parameters

- **query** (*Union[Update, str]*) –
- **initNs** (*Dict[str, Any]*) –
- **initBindings** (*Dict[str, Identifier]*) –
- **queryGraph** (*Optional[str]*) –

- `DEBUG` (`bool`) –

```
where_pattern = re.compile('(P<where>WHERE\\s*\\{)', re.IGNORECASE)
```

## Module contents

This package contains modules for additional RDFLib stores

## Module contents

Default plugins for rdflib.

This is a namespace package and contains the default plugins for rdflib.

## rdflib.tools package

### Submodules

#### rdflib.tools.chunk\_serializer module

This file provides a single function *serialize\_in\_chunks()* which can serialize a Graph into a number of NT files with a maximum number of triples or maximum file size.

There is an option to preserve any prefixes declared for the original graph in the first file, which will be a Turtle file.

```
rdflib.tools.chunk_serializer.serialize_in_chunks(g, max_triples=10000, max_file_size_kb=None,  
                                                  file_name_stem='chunk', output_dir=None,  
                                                  write_prefixes=False)
```

Serializes a given Graph into a series of n-triples with a given length.

#### Parameters

- **g** (*Graph*) – The graph to serialize.
- **max\_file\_size\_kb** (*Optional[int]*) – Maximum size per NT file in kB (1,000 bytes) Equivalent to ~6,000 triples, depending on Literal sizes.
- **max\_triples** (*int*) – Maximum size per NT file in triples Equivalent to lines in file.  
If both this parameter and `max_file_size_kb` are set, `max_file_size_kb` will be used.
- **file\_name\_stem** (*str*) – Prefix of each file name. e.g. “chunk” = chunk\_000001.nt, chunk\_000002.nt...
- **output\_dir** (*Optional[Path]*) – The directory you want the files to be written to.
- **write\_prefixes** (*bool*) – The first file created is a Turtle file containing original graph prefixes.

#### Return type

`None`

See `../test/test_tools/test_chunk_serializer.py` for examples of this in use.

## rdflib.tools.csv2rdf module

A commandline tool for semi-automatically converting CSV to RDF.

See also <https://github.com/RDFLib/pyTARQL> in the RDFlib family of tools

try: csv2rdf --help

**class** rdflib.tools.csv2rdf.CSV2RDF

Bases: `object`

```
__dict__ = mappingproxy({'__module__': 'rdflib.tools.csv2rdf', '__init__':
<function CSV2RDF.__init__>, 'triple': <function CSV2RDF.triple>, 'convert':
<function CSV2RDF.convert>, '__dict__': <attribute '__dict__' of 'CSV2RDF'
objects>, '__weakref__': <attribute '__weakref__' of 'CSV2RDF' objects>, '__doc__':
None, '__annotations__': {}})
```

`__init__()`

`__module__` = 'rdflib.tools.csv2rdf'

`__weakref__`

list of weak references to the object (if defined)

`convert(csvreader)`

`triple(s, p, o)`

## rdflib.tools.defined\_namespace\_creator module

rdflib.tools.defined\_namespace\_creator.get\_target\_namespace\_elements(*g*, *target\_namespace*)

### Parameters

- *g* (*Graph*) –
- *target\_namespace* (*str*) –

### Return type

`Tuple[List[Tuple[str, str]], List[str]]`

rdflib.tools.defined\_namespace\_creator.make\_dn\_file(*output\_file\_name*, *target\_namespace*,  
*elements\_strs*, *object\_id*, *fail*)

### Parameters

- *output\_file\_name* (*Path*) –
- *target\_namespace* (*str*) –
- *elements\_strs* (*Iterable[str]*) –
- *object\_id* (*str*) –
- *fail* (*bool*) –

### Return type

`None`

```
rdflib.tools.defined_namespace_creator.validate_namespace(namespace)
```

**Parameters**

**namespace** (*str*) –

**Return type**

*None*

```
rdflib.tools.defined_namespace_creator.validate_object_id(object_id)
```

**Parameters**

**object\_id** (*str*) –

**Return type**

*None*

### rdflib.tools.graphisomorphism module

A commandline tool for testing if RDF graphs are isomorphic, i.e. equal if BNode labels are ignored.

```
class rdflib.tools.graphisomorphism.IsomorphicTestableGraph(**kargs)
```

Bases: *Graph*

Ported from: <http://www.w3.org/2001/sw/DataAccess/proto-tests/tools/rdfdiff.py> (Sean B Palmer's RDF Graph Isomorphism Tester)

```
__eq__(G)
```

Graph isomorphism testing.

```
__hash__ = None
```

```
__init__(**kargs)
```

```
__module__ = 'rdflib.tools.graphisomorphism'
```

```
__ne__(G)
```

Negative graph isomorphism testing.

```
hashtriples()
```

```
internal_hash()
```

This is defined instead of `__hash__` to avoid a circular recursion scenario with the Memory store for rdflib which requires a hash lookup in order to return a generator of triples

```
vhash(term, done=False)
```

```
vhashtriple(triple, term, done)
```

```
vhashtriples(term, done)
```

```
rdflib.tools.graphisomorphism.main()
```



### rdflib.tools.rdf2dot module

A commandline tool for drawing RDF graphs in Graphviz DOT format

You can draw the graph of an RDF file directly:

```
rdflib.tools.rdf2dot.main()
```

```
rdflib.tools.rdf2dot.rdf2dot(g, stream, opts={})
```

Convert the RDF graph to DOT writes the dot output to the stream

### rdflib.tools.rdfpipe module

A commandline tool for parsing RDF in different formats and serializing the resulting graph to a chosen format.

```
rdflib.tools.rdfpipe.main()
```

```
rdflib.tools.rdfpipe.make_option_parser()
```

```
rdflib.tools.rdfpipe.parse_and_serialize(input_files, input_format, guess, outfile, output_format,
                                         ns_bindings, store_conn="", store_type=None)
```

### rdflib.tools.rdfs2dot module

A commandline tool for drawing RDFS Class diagrams in Graphviz DOT format

You can draw the graph of an RDFS file directly:

```
rdflib.tools.rdfs2dot.main()
```

```
rdflib.tools.rdfs2dot.rdfs2dot(g, stream, opts={})
```

Convert the RDFS schema in a graph writes the dot output to the stream

## Module contents

Various commandline tools for working with RDFLib

## Submodules

### rdflib.collection module

```
class rdflib.collection.Collection(graph, uri, seq=[])
```

Bases: `object`

See “Emulating container types”: <https://docs.python.org/reference/datamodel.html#emulating-container-types>

```

>>> from rdflib.term import Literal
>>> from rdflib.graph import Graph
>>> from pprint import pprint
>>> listname = BNode()
>>> g = Graph('Memory')
>>> listItem1 = BNode()

```

(continues on next page)

(continued from previous page)

```

>>> listItem2 = BNode()
>>> g.add((listname, RDF.first, Literal(1)))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listname, RDF.rest, listItem1))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listItem1, RDF.first, Literal(2)))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listItem1, RDF.rest, listItem2))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listItem2, RDF.rest, RDF.nil))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listItem2, RDF.first, Literal(3)))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> c = Collection(g, listname)
>>> pprint([term.n3() for term in c])
[u'"1"^^<http://www.w3.org/2001/XMLSchema#integer>',
 u'"2"^^<http://www.w3.org/2001/XMLSchema#integer>',
 u'"3"^^<http://www.w3.org/2001/XMLSchema#integer>']

```

```

>>> Literal(1) in c
True
>>> len(c)
3
>>> c._get_container(1) == listItem1
True
>>> c.index(Literal(2)) == 1
True

```

### Parameters

- **graph** (*Graph*) –
- **uri** (*Node*) –
- **seq** (*List[Node]*) –

**\_\_delitem\_\_**(*key*)

```

>>> from rdflib.namespace import RDF, RDFS
>>> from rdflib import Graph
>>> from pprint import pformat
>>> g = Graph()
>>> a = BNode('foo')
>>> b = BNode('bar')
>>> c = BNode('baz')
>>> g.add((a, RDF.first, RDF.type))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((a, RDF.rest, b))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((b, RDF.first, RDFS.label))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((b, RDF.rest, c))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>

```

(continues on next page)

(continued from previous page)

```

>>> g.add((c, RDF.first, RDFS.comment))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((c, RDF.rest, RDF.nil))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> len(g)
6
>>> def listAncestry(node, graph):
...     for i in graph.subjects(RDF.rest, node):
...         yield i
>>> [str(node.n3())
...     for node in g.transitiveClosure(listAncestry, RDF.nil)]
['_:baz', ' _:bar', ' _:foo']
>>> lst = Collection(g, a)
>>> len(lst)
3
>>> b == lst._get_container(1)
True
>>> c == lst._get_container(2)
True
>>> del lst[1]
>>> len(lst)
2
>>> len(g)
4

```

**Parameters****key** (`int`) –**Return type**`None`

```
__dict__ = mappingproxy({'__module__': 'rdflib.collection', '__doc__': '\n See
"Emulating container types":\n
https://docs.python.org/reference/datamodel.html#emulating-container-types\n\n >>>
from rdflib.term import Literal\n >>> from rdflib.graph import Graph\n >>> from
pprint import pprint\n >>> listname = BNode()\n >>> g = Graph(\'Memory\')\n >>>
listItem1 = BNode()\n >>> listItem2 = BNode()\n >>> g.add((listname, RDF.first,
Literal(1))) # doctest: +ELLIPSIS\n <Graph identifier=... (<class
\'rdflib.graph.Graph\')>)\n >>> g.add((listname, RDF.rest, listItem1)) # doctest:
+ELLIPSIS\n <Graph identifier=... (<class \'rdflib.graph.Graph\')>)\n >>>
g.add((listItem1, RDF.first, Literal(2))) # doctest: +ELLIPSIS\n <Graph
identifier=... (<class \'rdflib.graph.Graph\')>)\n >>> g.add((listItem1, RDF.rest,
listItem2)) # doctest: +ELLIPSIS\n <Graph identifier=... (<class
\'rdflib.graph.Graph\')>)\n >>> g.add((listItem2, RDF.rest, RDF.nil)) # doctest:
+ELLIPSIS\n <Graph identifier=... (<class \'rdflib.graph.Graph\')>)\n >>>
g.add((listItem2, RDF.first, Literal(3))) # doctest: +ELLIPSIS\n <Graph
identifier=... (<class \'rdflib.graph.Graph\')>)\n >>> c = Collection(g,listname)\n
>>> pprint([term.n3() for term in c])\n
[u\'"1"^^<http://www.w3.org/2001/XMLSchema#integer>\',\n
u\'"2"^^<http://www.w3.org/2001/XMLSchema#integer>\',\n
u\'"3"^^<http://www.w3.org/2001/XMLSchema#integer>\']\n\n >>> Literal(1) in c\n
True\n >>> len(c)\n 3\n >>> c._get_container(1) == listItem1\n True\n >>>
c.index(Literal(2)) == 1\n True\n ', '__init__': <function Collection.__init__>,
'n3': <function Collection.n3>, '_get_container': <function
Collection._get_container>, '__len__': <function Collection.__len__>, 'index':
<function Collection.index>, '__getitem__': <function Collection.__getitem__>,
'__setitem__': <function Collection.__setitem__>, '__delitem__': <function
Collection.__delitem__>, '__iter__': <function Collection.__iter__>, '_end':
<function Collection._end>, 'append': <function Collection.append>, '__iadd__':
<function Collection.__iadd__>, 'clear': <function Collection.clear>, '__dict__':
<attribute \'__dict__\' of \'Collection\' objects>, '__weakref__': <attribute
\'__weakref__\' of \'Collection\' objects>, '__annotations__': {}}}
```

`__getitem__(key)`

TODO

**Parameters**

**key** (`int`) –

**Return type**

`Node`

`__iadd__(other)`

**Parameters**

**other** (`Iterable[Node]`) –

`__init__(graph, uri, seq=[])`

**Parameters**

- **graph** (`Graph`) –
- **uri** (`Node`) –
- **seq** (`List[Node]`) –

`__iter__()`

Iterator over items in Collections

**Return type**`Iterator[Node]``__len__()`

length of items in collection.

**Return type**`int``__module__ = 'rdflib.collection'``__setitem__(key, value)`

TODO

**Parameters**

- **key** (`int`) –
- **value** (`Node`) –

**Return type**`None``__weakref__`

list of weak references to the object (if defined)

`append(item)`

```
>>> from rdflib.term import Literal
>>> from rdflib.graph import Graph
>>> listname = BNode()
>>> g = Graph()
>>> c = Collection(g, listname, [Literal(1), Literal(2)])
>>> links = [
...     list(g.subjects(object=i, predicate=RDF.first))[0] for i in c]
>>> len([i for i in links if (i, RDF.rest, RDF.nil) in g])
1
```

**Parameters****item** (`Node`) –**Return type**`Collection``clear()``index(item)`

Returns the 0-based numerical index of the item in the list

**Parameters****item** (`Node`) –**Return type**`int``n3()`

```

>>> from rdflib.term import Literal
>>> from rdflib.graph import Graph
>>> listname = BNode()
>>> g = Graph('Memory')
>>> listItem1 = BNode()
>>> listItem2 = BNode()
>>> g.add((listname, RDF.first, Literal(1)))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listname, RDF.rest, listItem1))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listItem1, RDF.first, Literal(2)))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listItem1, RDF.rest, listItem2))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listItem2, RDF.rest, RDF.nil))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((listItem2, RDF.first, Literal(3)))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> c = Collection(g, listname)
>>> print(c.n3())
( "1"^^<http://www.w3.org/2001/XMLSchema#integer>
:rtype: :py:class:`str`

```

```

"2"^^<http://www.w3.org/2001/XMLSchema#integer>
XMLSchema#integer> )

```

```

"3"^^<http://www.w3.org/2001/

```

## rdflib.compare module

A collection of utilities for canonicalizing and inspecting graphs.

Among other things, they solve of the problem of deterministic bnode comparisons.

Warning: the time to canonicalize bnodes may increase exponentially on degenerate larger graphs. Use with care!

Example of comparing two graphs:

```

>>> g1 = Graph().parse(format='n3', data='''
... @prefix : <http://example.org/ns#> .
... <http://example.org> :rel
...   <http://example.org/same>,
...   [ :label "Same" ],
...   <http://example.org/a>,
...   [ :label "A" ] .
... ''')
>>> g2 = Graph().parse(format='n3', data='''
... @prefix : <http://example.org/ns#> .
... <http://example.org> :rel
...   <http://example.org/same>,
...   [ :label "Same" ],
...   <http://example.org/b>,
...   [ :label "B" ] .
... ''')
>>>

```

(continues on next page)

(continued from previous page)

```
>>> iso1 = to_isomorphic(g1)
>>> iso2 = to_isomorphic(g2)
```

These are not isomorphic:

```
>>> iso1 == iso2
False
```

Diff the two graphs:

```
>>> in_both, in_first, in_second = graph_diff(iso1, iso2)
```

Present in both:

```
>>> def dump_nt_sorted(g):
...     for l in sorted(g.serialize(format='nt').splitlines()):
...         if l: print(l.decode('ascii'))

>>> dump_nt_sorted(in_both)
<http://example.org>
  <http://example.org/ns#rel> <http://example.org/same> .
<http://example.org>
  <http://example.org/ns#rel> _:cbcaabaaba17fecbc304a64f8edee4335e .
_:cbcaabaaba17fecbc304a64f8edee4335e
  <http://example.org/ns#label> "Same" .
```

Only in first:

```
>>> dump_nt_sorted(in_first)
<http://example.org>
  <http://example.org/ns#rel> <http://example.org/a> .
<http://example.org>
  <http://example.org/ns#rel> _:cb124e4c6da0579f810c0ffe4eff485bd9 .
_:cb124e4c6da0579f810c0ffe4eff485bd9
  <http://example.org/ns#label> "A" .
```

Only in second:

```
>>> dump_nt_sorted(in_second)
<http://example.org>
  <http://example.org/ns#rel> <http://example.org/b> .
<http://example.org>
  <http://example.org/ns#rel> _:cb558f30e21ddfc05ca53108348338ade8 .
_:cb558f30e21ddfc05ca53108348338ade8
  <http://example.org/ns#label> "B" .
```

**class** `rdflib.compare.IsomorphicGraph(**kwargs)`

Bases: `ConjunctiveGraph`

An implementation of the RGDA1 graph digest algorithm.

An implementation of RGDA1 (publication below), a combination of Sayers & Karp's graph digest algorithm using sum and SHA-256 <<http://www.hpl.hp.com/techreports/2003/HPL-2003-235R1.pdf>> and traces <<http://pallini.di.uniroma1.it>>, an average case polynomial time algorithm for graph canonicalization.

McCusker, J. P. (2015). WebSig: A Digital Signature Framework for the Web. Rensselaer Polytechnic Institute, Troy, NY. <http://gradworks.umi.com/3727015.pdf>

`__eq__(other)`

Graph isomorphism testing.

`__hash__()`

Return hash(self).

`__init__(**kwargs)`

`__module__ = 'rdflib.compare'`

`__ne__(other)`

Negative graph isomorphism testing.

`graph_digest(stats=None)`

Synonym for `IsomorphicGraph.internal_hash`.

`internal_hash(stats=None)`

This is defined instead of `__hash__` to avoid a circular recursion scenario with the Memory store for rdflib which requires a hash lookup in order to return a generator of triples.

`rdflib.compare.graph_diff(g1, g2)`

Returns three sets of triples: “in both”, “in first” and “in second”.

#### Parameters

- `g1 (Graph)` –
- `g2 (Graph)` –

#### Return type

`Tuple[Graph, Graph, Graph]`

`rdflib.compare.isomorphic(graph1, graph2)`

Compare graph for equality.

Uses an algorithm to compute unique hashes which takes bnodes into account.

Examples:

```
>>> g1 = Graph().parse(format='n3', data='''
...   @prefix : <http://example.org/ns#> .
...   <http://example.org> :rel <http://example.org/a> .
...   <http://example.org> :rel <http://example.org/b> .
...   <http://example.org> :rel [ :label "A bnode." ] .
... ''')
>>> g2 = Graph().parse(format='n3', data='''
...   @prefix ns: <http://example.org/ns#> .
...   <http://example.org> ns:rel [ ns:label "A bnode." ] .
...   <http://example.org> ns:rel <http://example.org/b>,
...   <http://example.org/a> .
... ''')
>>> isomorphic(g1, g2)
True

>>> g3 = Graph().parse(format='n3', data='''
...   @prefix : <http://example.org/ns#> .
```

(continues on next page)



(continued from previous page)

```

...     <http://example.org> :rel <http://example.org/a> .
...     <http://example.org> :rel <http://example.org/b> .
...     <http://example.org> :rel <http://example.org/c> .
...     '''
>>> isomorphic(g1, g3)
False

```

**Parameters**

- **graph1** (*Graph*) –
- **graph2** (*Graph*) –

**Return type***bool*

`rdflib.compare.similar(g1, g2)`

Checks if the two graphs are “similar”.

Checks if the two graphs are “similar”, by comparing sorted triples where all bnodes have been replaced by a singular mock bnode (the `_MOCK_BNODE`).

This is a much cheaper, but less reliable, alternative to the comparison algorithm in `isomorphic`.

**Parameters**

- **g1** (*Graph*) –
- **g2** (*Graph*) –

`rdflib.compare.to_canonical_graph(g1, stats=None)`

Creates a canonical, read-only graph.

Creates a canonical, read-only graph where all bnode id:s are based on deterministical SHA-256 checksums, correlated with the graph contents.

**Parameters**

- **g1** (*Graph*) –
- **stats** (*Optional[Dict[str, Union[int, str]]]*) –

**Return type***ReadOnlyGraphAggregate*

`rdflib.compare.to_isomorphic(graph)`

**Parameters**

**graph** (*Graph*) –

**Return type***IsomorphicGraph*

### rdflib.compat module

Utility functions and objects to ease Python 2/3 compatibility, and different versions of support libraries.

`rdflib.compat.ascii(stream)`

`rdflib.compat.bopen(*args, **kwargs)`

`rdflib.compat.cast_bytes(s, enc='utf-8')`

`rdflib.compat.decodeStringEscape(s)`

`rdflib.compat.decodeUnicodeEscape(escaped)`

**Parameters**

**escaped** (`str`) –

**Return type**

`str`

`rdflib.compat.sign(n)`

### rdflib.container module

`class rdflib.container.Alt(graph, uri, seq=[])`

Bases: `Container`

`__init__(graph, uri, seq=[])`

Creates a Container

**Parameters**

- **graph** – a Graph instance
- **uri** – URI or Blank Node of the Container
- **seq** – the elements of the Container
- **rtype** – the type of Container, one of “Bag”, “Seq” or “Alt”

`__module__ = 'rdflib.container'`

`anyone()`

`class rdflib.container.Bag(graph, uri, seq=[])`

Bases: `Container`

Unordered container (no preference order of elements)

`__init__(graph, uri, seq=[])`

Creates a Container

**Parameters**

- **graph** – a Graph instance
- **uri** – URI or Blank Node of the Container
- **seq** – the elements of the Container
- **rtype** – the type of Container, one of “Bag”, “Seq” or “Alt”

```
__module__ = 'rdflib.container'
```

```
class rdflib.container.Container(graph, uri, seq=[], rtype='Bag')
```

Bases: `object`

A class for constructing RDF containers, as per <https://www.w3.org/TR/rdf11-mt/#rdf-containers>

Basic usage, creating a Bag and adding to it:

```
>>> from rdflib import Graph, BNode, Literal, Bag
>>> g = Graph()
>>> b = Bag(g, BNode(), [Literal("One"), Literal("Two"), Literal("Three")])
>>> print(g.serialize(format="turtle"))
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

[] a rdf:Bag ;
   rdf:_1 "One" ;
   rdf:_2 "Two" ;
   rdf:_3 "Three" .

>>> # print out an item using an index reference
>>> print(b[2])
Two

>>> # add a new item
>>> b.append(Literal("Hello"))
<rdflib.container.Bag object at ...>
>>> print(g.serialize(format="turtle"))
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

[] a rdf:Bag ;
   rdf:_1 "One" ;
   rdf:_2 "Two" ;
   rdf:_3 "Three" ;
   rdf:_4 "Hello" .
```

```
__delitem__(key)
```

Removing the item with index key or predicate `rdf:_key`

```
__dict__ = mappingproxy({'__module__': 'rdflib.container', '__doc__': 'A class for
constructing RDF containers, as per
https://www.w3.org/TR/rdf11-mt/#rdf-containers\n\n Basic usage, creating a ``Bag``
and adding to it:\n\n >>> from rdflib import Graph, BNode, Literal, Bag\n >>> g =
Graph()\n >>> b = Bag(g, BNode(), [Literal("One"), Literal("Two"),
Literal("Three")])\n >>> print(g.serialize(format="turtle"))\n @prefix rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n <BLANKLINE>\n [] a rdf:Bag ;\n
rdf:_1 "One" ;\n rdf:_2 "Two" ;\n rdf:_3 "Three" .\n <BLANKLINE>\n <BLANKLINE>\n\n
>>> # print out an item using an index reference\n >>> print(b[2])\n Two\n\n >>> #
add a new item\n >>> b.append(Literal("Hello")) # doctest: +ELLIPSIS\n
<rdflib.container.Bag object at ...>\n >>> print(g.serialize(format="turtle"))\n
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n <BLANKLINE>\n [] a
rdf:Bag ;\n rdf:_1 "One" ;\n rdf:_2 "Two" ;\n rdf:_3 "Three" ;\n rdf:_4 "Hello" .\n
<BLANKLINE>\n <BLANKLINE>\n\n ', '__init__': <function Container.__init__>, 'n3':
<function Container.n3>, '_get_container': <function Container._get_container>,
'_len__': <function Container._len__>, 'type_of_container': <function
Container.type_of_container>, 'index': <function Container.index>, '__getitem__':
<function Container._getitem__>, '__setitem__': <function Container._setitem__>,
'_delitem__': <function Container._delitem__>, 'items': <function
Container.items>, 'end': <function Container.end>, 'append': <function
Container.append>, 'append_multiple': <function Container.append_multiple>,
'clear': <function Container.clear>, '__dict__': <attribute '__dict__' of
'Container' objects>, '__weakref__': <attribute '__weakref__' of 'Container'
objects>, '__annotations__': {}}}
```

**\_\_getitem\_\_**(*key*)

Returns item of the container at index key

**\_\_init\_\_**(*graph, uri, seq=[], rtype='Bag'*)

Creates a Container

#### Parameters

- **graph** – a Graph instance
- **uri** – URI or Blank Node of the Container
- **seq** – the elements of the Container
- **rtype** – the type of Container, one of “Bag”, “Seq” or “Alt”

**\_\_len\_\_**()

Number of items in container

**\_\_module\_\_** = 'rdflib.container'

**\_\_setitem\_\_**(*key, value*)

Sets the item at index key or predicate rdf:\_key of the container to value

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**append**(*item*)

Adding item to the end of the container

**append\_multiple**(*other*)

Adding multiple elements to the container to the end which are in python list other

**clear()**

Removing all elements from the container

**end()**

**index(*item*)**

Returns the 1-based numerical index of the item in the container

**items()**

Returns a list of all items in the container

**n3()**

**type\_of\_container()**

**exception** `rdflib.container.NoElementException`(*message='rdf:Alt Container is empty'*)

Bases: `Exception`

**\_\_init\_\_**(*message='rdf:Alt Container is empty'*)

**\_\_module\_\_** = `'rdflib.container'`

**\_\_str\_\_**()

Return str(self).

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**class** `rdflib.container.Seq`(*graph, uri, seq=[]*)

Bases: `Container`

**\_\_init\_\_**(*graph, uri, seq=[]*)

Creates a Container

#### Parameters

- **graph** – a Graph instance
- **uri** – URI or Blank Node of the Container
- **seq** – the elements of the Container
- **rtype** – the type of Container, one of “Bag”, “Seq” or “Alt”

**\_\_module\_\_** = `'rdflib.container'`

**add\_at\_position**(*pos, item*)

## rdflib.events module

### Dirt Simple Events

A Dispatcher (or a subclass of Dispatcher) stores event handlers that are ‘fired’ simple event objects when interesting things happen.

Create a dispatcher:

```
>>> d = Dispatcher()
```

Now create a handler for the event and subscribe it to the dispatcher to handle Event events. A handler is a simple function or method that accepts the event as an argument:

```
>>> def handler1(event): print(repr(event))
>>> d.subscribe(Event, handler1)
<rdflib.events.Dispatcher object at ...>
```

Now dispatch a new event into the dispatcher, and see handler1 get fired:

```
>>> d.dispatch(Event(foo='bar', data='yours', used_by='the event handlers'))
<rdflib.events.Event ['data', 'foo', 'used_by']>
```

### **class** rdflib.events.Dispatcher

Bases: `object`

An object that can dispatch events to a privately managed group of subscribers.

```
__dict__ = mappingproxy({'__module__': 'rdflib.events', '__doc__': '\n An object
that can dispatch events to a privately managed group of\n subscribers.\n ',
'_dispatch_map': None, 'set_map': <function Dispatcher.set_map>, 'get_map':
<function Dispatcher.get_map>, 'subscribe': <function Dispatcher.subscribe>,
'dispatch': <function Dispatcher.dispatch>, '__dict__': <attribute '__dict__' of
'Dispatcher' objects>, '__weakref__': <attribute '__weakref__' of 'Dispatcher'
objects>, '__annotations__': {}})
```

```
__module__ = 'rdflib.events'
```

```
__weakref__
```

list of weak references to the object (if defined)

**dispatch**(*event*)

Dispatch the given event to the subscribed handlers for the event's type

**get\_map**()

**set\_map**(*amap*)

**subscribe**(*event\_type*, *handler*)

Subscribe the given handler to an event\_type. Handlers are called in the order they are subscribed.

### **class** rdflib.events.Event(\*\**kw*)

Bases: `object`

An event is a container for attributes. The source of an event creates this object, or a subclass, gives it any kind of data that the events handlers need to handle the event, and then calls `notify(event)`.

The target of an event registers a function to handle the event it is interested with `subscribe()`. When a sources calls `notify(event)`, each subscriber to that event will be called in no particular order.

```
__dict__ = mappingproxy({'__module__': 'rdflib.events', '__doc__': '\n An event is
a container for attributes. The source of an event\n creates this object, or a
subclass, gives it any kind of data that\n the events handlers need to handle the
event, and then calls\n notify(event).\n\n The target of an event registers a
function to handle the event it\n is interested with subscribe(). When a sources
calls\n notify(event), each subscriber to that event will be called in no\n
particular order.\n ', '__init__': <function Event.__init__>, '__repr__':
<function Event.__repr__>, '__dict__': <attribute '__dict__' of 'Event' objects>,
'__weakref__': <attribute '__weakref__' of 'Event' objects>, '__annotations__':
{}})
```

```

__init__(**kw)

__module__ = 'rdflib.events'

__repr__()
    Return repr(self).

__weakref__
    list of weak references to the object (if defined)

```

## rdflib.exceptions module

TODO:

**exception** rdflib.exceptions.**Error**(*msg=None*)

Bases: [Exception](#)

Base class for rdflib exceptions.

### Parameters

**msg** ([Optional](#)[[str](#)]) –

**\_\_init\_\_**(*msg=None*)

### Parameters

**msg** ([Optional](#)[[str](#)]) –

**\_\_module\_\_** = 'rdflib.exceptions'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**exception** rdflib.exceptions.**ParserError**(*msg*)

Bases: [Error](#)

RDF Parser error.

### Parameters

**msg** ([str](#)) –

**\_\_init\_\_**(*msg*)

### Parameters

**msg** ([str](#)) –

**\_\_module\_\_** = 'rdflib.exceptions'

**\_\_str\_\_**()

Return str(self).

### Return type

[str](#)

**exception** rdflib.exceptions.**UniquenessError**(*values*)

Bases: [Error](#)

A uniqueness assumption was made in the context, and that is not true

### Parameters

**values** ([Any](#)) –

```
__init__(values)

    Parameters
    values (Any) –

__module__ = 'rdflib.exceptions'
```

## rdflib.graph module

RDFLib defines the following kinds of Graphs:

- [\*Graph\*](#)
- [\*QuotedGraph\*](#)
- [\*ConjunctiveGraph\*](#)
- [\*Dataset\*](#)

### Graph

An RDF graph is a set of RDF triples. Graphs support the python `in` operator, as well as iteration and some operations like union, difference and intersection.

see [\*Graph\*](#)

### Conjunctive Graph

A Conjunctive Graph is the most relevant collection of graphs that are considered to be the boundary for closed world assumptions. This boundary is equivalent to that of the store instance (which is itself uniquely identified and distinct from other instances of [\*Store\*](#) that signify other Conjunctive Graphs). It is equivalent to all the named graphs within it and associated with a `_default_` graph which is automatically assigned a [\*BNode\*](#) for an identifier - if one isn't given.

see [\*ConjunctiveGraph\*](#)

### Quoted graph

The notion of an RDF graph [14] is extended to include the concept of a formula node. A formula node may occur wherever any other kind of node can appear. Associated with a formula node is an RDF graph that is completely disjoint from all other graphs; i.e. has no nodes in common with any other graph. (It may contain the same labels as other RDF graphs; because this is, by definition, a separate graph, considerations of tidiness do not apply between the graph at a formula node and any other graph.)

This is intended to map the idea of “{ N3-expression }” that is used by N3 into an RDF graph upon which RDF semantics is defined.

see [\*QuotedGraph\*](#)



## Dataset

The RDF 1.1 Dataset, a small extension to the Conjunctive Graph. The primary term is “graphs in the datasets” and not “contexts with quads” so there is a separate method to set/retrieve a graph in a dataset and to operate with dataset graphs. As a consequence of this approach, dataset graphs cannot be identified with blank nodes, a name is always required (RDFLib will automatically add a name if one is not provided at creation time). This implementation includes a convenience method to directly add a single quad to a dataset graph.

see [Dataset](#)

## Working with graphs

Instantiating Graphs with default store (Memory) and default identifier (a BNode):

```
>>> g = Graph()
>>> g.store.__class__
<class 'rdflib.plugins.stores.memory.Memory'>
>>> g.identifier.__class__
<class 'rdflib.term.BNode'>
```

Instantiating Graphs with a Memory store and an identifier - <https://rdflib.github.io>:

```
>>> g = Graph('Memory', URIRef("https://rdflib.github.io"))
>>> g.identifier
rdflib.term.URIRef('https://rdflib.github.io')
>>> str(g)
"<https://rdflib.github.io> a rdfg:Graph;rdflib:storage
 [a rdflib:Store;rdfs:label 'Memory']."
```

Creating a ConjunctiveGraph - The top level container for all named Graphs in a “database”:

```
>>> g = ConjunctiveGraph()
>>> str(g.default_context)
"[a rdfg:Graph;rdflib:storage [a rdflib:Store;rdfs:label 'Memory']]."
```

Adding / removing reified triples to Graph and iterating over it directly or via triple pattern:

```
>>> g = Graph()
>>> statementId = BNode()
>>> print(len(g))
0
>>> g.add((statementId, RDF.type, RDF.Statement))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((statementId, RDF.subject,
...      URIRef("https://rdflib.github.io/store/ConjunctiveGraph")))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((statementId, RDF.predicate, namespace.RDFS.label))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((statementId, RDF.object, Literal("Conjunctive Graph")))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> print(len(g))
4
>>> for s, p, o in g:
```

(continues on next page)

(continued from previous page)

```
...     print(type(s))
...
<class 'rdflib.term.BNode'>
<class 'rdflib.term.BNode'>
<class 'rdflib.term.BNode'>
<class 'rdflib.term.BNode'>
```

```
>>> for s, p, o in g.triples((None, RDF.object, None)):
...     print(o)
...
Conjunctive Graph
>>> g.remove((statementId, RDF.type, RDF.Statement))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> print(len(g))
3
```

None terms in calls to `triples()` can be thought of as “open variables”.

Graph support set-theoretic operators, you can add/subtract graphs, as well as intersection (with multiplication operator  $g1 * g2$ ) and xor ( $g1 \wedge g2$ ).

Note that BNode IDs are kept when doing set-theoretic operations, this may or may not be what you want. Two named graphs within the same application probably want share BNode IDs, two graphs with data from different sources probably not. If your BNode IDs are all generated by RDFLib they are UUIDs and unique.

```
>>> g1 = Graph()
>>> g2 = Graph()
>>> u = URIRef("http://example.com/foo")
>>> g1.add([u, namespace.RDFS.label, Literal("foo")])
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g1.add([u, namespace.RDFS.label, Literal("bar")])
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g2.add([u, namespace.RDFS.label, Literal("foo")])
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g2.add([u, namespace.RDFS.label, Literal("bing")])
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> len(g1 + g2) # adds bing as label
3
>>> len(g1 - g2) # removes foo
1
>>> len(g1 * g2) # only foo
1
>>> g1 += g2 # now g1 contains everything
```

Graph Aggregation - ConjunctiveGraphs and ReadOnlyGraphAggregate within the same store:

```
>>> store = plugin.get("Memory", Store)()
>>> g1 = Graph(store)
>>> g2 = Graph(store)
>>> g3 = Graph(store)
>>> stmt1 = BNode()
>>> stmt2 = BNode()
>>> stmt3 = BNode()
```

(continues on next page)

(continued from previous page)

```

>>> g1.add((stmt1, RDF.type, RDF.Statement))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g1.add((stmt1, RDF.subject,
...     URIRef('https://rdflib.github.io/store/ConjunctiveGraph')))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g1.add((stmt1, RDF.predicate, namespace.RDFS.label))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g1.add((stmt1, RDF.object, Literal('Conjunctive Graph')))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g2.add((stmt2, RDF.type, RDF.Statement))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g2.add((stmt2, RDF.subject,
...     URIRef('https://rdflib.github.io/store/ConjunctiveGraph')))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g2.add((stmt2, RDF.predicate, RDF.type))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g2.add((stmt2, RDF.object, namespace.RDFS.Class))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g3.add((stmt3, RDF.type, RDF.Statement))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g3.add((stmt3, RDF.subject,
...     URIRef('https://rdflib.github.io/store/ConjunctiveGraph')))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g3.add((stmt3, RDF.predicate, namespace.RDFS.comment))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g3.add((stmt3, RDF.object, Literal(
...     'The top-level aggregate graph - The sum ' +
...     'of all named graphs within a Store')))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> len(list(ConjunctiveGraph(store).subjects(RDF.type, RDF.Statement)))
3
>>> len(list(ReadOnlyGraphAggregate([g1,g2]).subjects(
...     RDF.type, RDF.Statement)))
2

```

ConjunctiveGraphs have a `quads()` method which returns quads instead of triples, where the fourth item is the Graph (or subclass thereof) instance in which the triple was asserted:

```

>>> uniqueGraphNames = set(
...     [graph.identifier for s, p, o, graph in ConjunctiveGraph(store
...     ).quads((None, RDF.predicate, None))])
>>> len(uniqueGraphNames)
3
>>> unionGraph = ReadOnlyGraphAggregate([g1, g2])
>>> uniqueGraphNames = set(
...     [graph.identifier for s, p, o, graph in unionGraph.quads(
...     (None, RDF.predicate, None))])
>>> len(uniqueGraphNames)
2

```

Parsing N3 from a string

```

>>> g2 = Graph()
>>> src = '''
... @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
... @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
... [ a rdf:Statement ;
...   rdf:subject <https://rdflib.github.io/store#ConjunctiveGraph>;
...   rdf:predicate rdfs:label;
...   rdf:object "Conjunctive Graph" ] .
... '''
>>> g2 = g2.parse(data=src, format="n3")
>>> print(len(g2))
4

```

Using Namespace class:

```

>>> RDFLib = Namespace("https://rdflib.github.io/")
>>> RDFLib.ConjunctiveGraph
rdflib.term.URIRef('https://rdflib.github.io/ConjunctiveGraph')
>>> RDFLib["Graph"]
rdflib.term.URIRef('https://rdflib.github.io/Graph')

```

**class** rdflib.graph.**BatchAddGraph**(graph, batch\_size=1000, batch\_addn=False)

Bases: `object`

Wrapper around graph that turns batches of calls to Graph's add (and optionally, addN) into calls to batched calls to addN.

#### Parameters

- graph: The graph to wrap
- batch\_size: The maximum number of triples to buffer before passing to Graph's addN
- batch\_addn: If True, then even calls to `addN` will be batched according to batch\_size

graph: The wrapped graph count: The number of triples buffered since initialization or the last call to reset batch: The current buffer of triples

#### Parameters

- **graph** (*Graph*) –
- **batch\_size** (*int*) –
- **batch\_addn** (*bool*) –

```

__dict__ = mappingproxy({'__module__': 'rdflib.graph', '__doc__': "\n Wrapper
around graph that turns batches of calls to Graph's add\n (and optionally, addN)
into calls to batched calls to addN`.\n\n :Parameters:\n\n - graph: The graph to
wrap\n - batch_size: The maximum number of triples to buffer before passing to\n
Graph's addN\n - batch_addn: If True, then even calls to `addN` will be batched
according to\n batch_size\n\n graph: The wrapped graph\n count: The number of
triples buffered since initialization or the last call to reset\n batch: The
current buffer of triples\n\n ", '__init__': <function BatchAddGraph.__init__>,
'reset': <function BatchAddGraph.reset>, 'add': <function BatchAddGraph.add>,
'addN': <function BatchAddGraph.addN>, '__enter__': <function
BatchAddGraph.__enter__>, '__exit__': <function BatchAddGraph.__exit__>,
'__dict__': <attribute '__dict__' of 'BatchAddGraph' objects>, '__weakref__':
<attribute '__weakref__' of 'BatchAddGraph' objects>, '__annotations__': {}})

```

`__enter__()`

**Return type**

*BatchAddGraph*

`__exit__(*exc)`

**Return type**

*None*

`__init__(graph, batch_size=1000, batch_addn=False)`

**Parameters**

- **graph** (*Graph*) –
- **batch\_size** (*int*) –
- **batch\_addn** (*bool*) –

`__module__ = 'rdflib.graph'`

`__weakref__`

list of weak references to the object (if defined)

`add(triple_or_quad)`

Add a triple to the buffer

**Parameters**

- **triple** – The triple to add
- **triple\_or\_quad** (*Union[Tuple[Node, Node, Node], Tuple[Node, Node, Node, Graph]]*) –

**Return type**

*BatchAddGraph*

`addN(quads)`

**Parameters**

**quads** (*Iterable[Tuple[Node, Node, Node, Graph]]*) –

**Return type**

*BatchAddGraph*

`reset()`

Manually clear the buffered triples and reset the count to zero

**Return type**

*BatchAddGraph*

`class rdflib.graph.ConjunctiveGraph(store='default', identifier=None, default_graph_base=None)`

Bases: *Graph*

A ConjunctiveGraph is an (unnamed) aggregation of all the named graphs in a store.

It has a `default` graph, whose name is associated with the graph throughout its life. `__init__()` can take an identifier to use as the name of this default graph or it will assign a BNode.

All methods that add triples work against this default graph.

All queries are carried out against the union of all graphs.

**Parameters**

- **store** (`Union[Store, str]`) –
- **identifier** (`Union[IdentifiedNode, str, None]`) –
- **default\_graph\_base** (`Optional[str]`) –

**\_\_contains\_\_** (*triple\_or\_quad*)

Support for ‘triple/quad in graph’ syntax

**Parameters**

**triple\_or\_quad** (`Union[Tuple[Optional[Node], Optional[Node], Optional[Node]], Tuple[Optional[Node], Optional[Node], Optional[Node], Optional[Graph]]]`) –

**Return type**

`bool`

**\_\_init\_\_** (*store*='default', *identifier*=None, *default\_graph\_base*=None)

**Parameters**

- **store** (`Union[Store, str]`) –
- **identifier** (`Union[IdentifiedNode, str, None]`) –
- **default\_graph\_base** (`Optional[str]`) –

**\_\_len\_\_** ()

Number of triples in the entire conjunctive graph

**Return type**

`int`

**\_\_module\_\_** = 'rdflib.graph'

**\_\_reduce\_\_** ()

Helper for pickle.

**Return type**

`Tuple[Type[Graph], Tuple[Store, IdentifiedNode]]`

**\_\_str\_\_** ()

Return str(self).

**Return type**

`str`

**add** (*triple\_or\_quad*)

Add a triple or quad to the store.

if a triple is given it is added to the default context

**Parameters**

- **self** (`TypeVar(_ConjunctiveGraphT, bound= ConjunctiveGraph)`) –
- **triple\_or\_quad** (`Union[Tuple[Node, Node, Node], Tuple[Node, Node, Node, Optional[Graph]]]`) –

**Return type**

`TypeVar(_ConjunctiveGraphT, bound= ConjunctiveGraph)`

**addN(*quads*)**

Add a sequence of triples with context

**Parameters**

- **self** (`TypeVar(_ConjunctiveGraphT, bound= ConjunctiveGraph)`) –
- **quads** (`Iterable[Tuple[Node, Node, Node, Graph]]`) –

**Return type**

`TypeVar(_ConjunctiveGraphT, bound= ConjunctiveGraph)`

**context\_id(*uri*, *context\_id=None*)**

URI#context

**Parameters**

- **uri** (`str`) –
- **context\_id** (`Optional[str]`) –

**Return type**

`URIRef`

**contexts(*triple=None*)**

Iterate over all contexts in the graph

If triple is specified, iterate over all contexts the triple is in.

**Parameters**

**triple** (`Optional[Tuple[Node, Node, Node]]`) –

**Return type**

`Generator[Graph, None, None]`

**get\_context(*identifier*, *quoted=False*, *base=None*)**

Return a context graph for the given identifier

identifier must be a URIRef or BNode.

**Parameters**

- **identifier** (`Union[IdentifiedNode, str, None]`) –
- **quoted** (`bool`) –
- **base** (`Optional[str]`) –

**Return type**

`Graph`

**get\_graph(*identifier*)**

Returns the graph identified by given identifier

**Parameters**

**identifier** (`IdentifiedNode`) –

**Return type**

`Optional[Graph]`

**parse(*source=None*, *publicID=None*, *format=None*, *location=None*, *file=None*, *data=None*, *\*\*args*)**

Parse source adding the resulting triples to its own context (sub graph of this graph).

See `rdflib.graph.Graph.parse()` for documentation on arguments.

**Returns**

The graph into which the source was parsed. In the case of n3 it returns the root context.

**Caution:** This method can access directly or indirectly requested network or file resources, for example, when parsing JSON-LD documents with `@context` directives that point to a network location.

When processing untrusted or potentially malicious documents, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

#### Parameters

- **source** (`Union[IO[bytes], TextIO, InputSource, str, bytes, PurePath, None]`) –
- **publicID** (`Optional[str]`) –
- **format** (`Optional[str]`) –
- **location** (`Optional[str]`) –
- **file** (`Union[BinaryIO, TextIO, None]`) –
- **data** (`Union[str, bytes, None]`) –
- **args** (`Any`) –

#### Return type

*Graph*

**quads**(*triple\_or\_quad=None*)

Iterate over all the quads in the entire conjunctive graph

#### Parameters

**triple\_or\_quad** (`Union[Tuple[Optional[Node], Optional[Node], Optional[Node]], Tuple[Optional[Node], Optional[Node], Optional[Node], Optional[Graph]], None]`) –

#### Return type

`Generator[Tuple[Node, Node, Node, Optional[Graph]], None, None]`

**remove**(*triple\_or\_quad*)

Removes a triple or quads

if a triple is given it is removed from all contexts

a quad is removed from the given context only

#### Parameters

- **self** (`TypeVar(_ConjunctiveGraphT, bound= ConjunctiveGraph)`) –
- **triple\_or\_quad** (`Union[Tuple[Node, Node, Node], Tuple[Node, Node, Node, Optional[Graph]]]`) –

#### Return type

`TypeVar(_ConjunctiveGraphT, bound= ConjunctiveGraph)`

**remove\_context**(*context*)

Removes the given context from the graph



**Parameters****context** (*Graph*) –**Return type***None***triples**(*triple\_or\_quad*, *context=None*)

Iterate over all the triples in the entire conjunctive graph

For legacy reasons, this can take the context to query either as a fourth element of the quad, or as the explicit context keyword parameter. The kw param takes precedence.

**Parameters**

- **triple\_or\_quad** (*Union*[*Tuple*[*Optional*[*Node*], *Union*[*Path*, *Node*, *None*], *Optional*[*Node*]], *Tuple*[*Optional*[*Node*], *Union*[*Path*, *Node*, *None*], *Optional*[*Node*], *Optional*[*Graph*]]]) –
- **context** (*Optional*[*Graph*]) –

**Return type***Generator*[*Union*[*Tuple*[*Node*, *Node*, *Node*], *Tuple*[*Node*, *Path*, *Node*]], *None*, *None*]**triples\_choices**(*triple*, *context=None*)

Iterate over all the triples in the entire conjunctive graph

**Parameters**

- **triple** (*Union*[*Tuple*[*List*[*Node*], *Node*, *Node*], *Tuple*[*Node*, *List*[*Node*], *Node*], *Tuple*[*Node*, *Node*, *List*[*Node*]]]) –
- **context** (*Optional*[*Graph*]) –

**Return type***Generator*[*Tuple*[*Node*, *Node*, *Node*], *None*, *None*]**class** `rdflib.graph.Dataset`(*store='default'*, *default\_union=False*, *default\_graph\_base=None*)Bases: *ConjunctiveGraph*

RDF 1.1 Dataset. Small extension to the Conjunctive Graph: - the primary term is graphs in the datasets and not contexts with quads, so there is a separate method to set/retrieve a graph in a dataset and operate with graphs - graphs cannot be identified with blank nodes - added a method to directly add a single quad

Examples of usage:

```
>>> # Create a new Dataset
>>> ds = Dataset()
>>> # simple triples goes to default graph
>>> ds.add((URIRef("http://example.org/a"),
...        URIRef("http://www.example.org/b"),
...        Literal("foo")))
<Graph identifier=... (<class 'rdflib.graph.Dataset'>)>
>>>
>>> # Create a graph in the dataset, if the graph name has already been
>>> # used, the corresponding graph will be returned
>>> # (ie, the Dataset keeps track of the constituent graphs)
>>> g = ds.graph(URIRef("http://www.example.com/gr"))
>>>
>>> # add triples to the new graph as usual
>>> g.add(
```

(continues on next page)

(continued from previous page)

```

...     (URIRef("http://example.org/x"),
...     URIRef("http://example.org/y"),
...     Literal("bar")) )
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> # alternatively: add a quad to the dataset -> goes to the graph
>>> ds.add(
...     (URIRef("http://example.org/x"),
...     URIRef("http://example.org/z"),
...     Literal("foo-bar"),g) )
<Graph identifier=... (<class 'rdflib.graph.Dataset'>)>
>>>
>>> # querying triples return them all regardless of the graph
>>> for t in ds.triples((None,None,None)):
...     print(t)
(rdflib.term.URIRef("http://example.org/a"),
 rdflib.term.URIRef("http://www.example.org/b"),
 rdflib.term.Literal("foo"))
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/z"),
 rdflib.term.Literal("foo-bar"))
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/y"),
 rdflib.term.Literal("bar"))
>>>
>>> # querying quads() return quads; the fourth argument can be unrestricted
>>> # (None) or restricted to a graph
>>> for q in ds.quads((None, None, None, None)):
...     print(q)
(rdflib.term.URIRef("http://example.org/a"),
 rdflib.term.URIRef("http://www.example.org/b"),
 rdflib.term.Literal("foo"),
 None)
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/y"),
 rdflib.term.Literal("bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/z"),
 rdflib.term.Literal("foo-bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
>>>
>>> # unrestricted looping is equivalent to iterating over the entire Dataset
>>> for q in ds:
...     print(q)
(rdflib.term.URIRef("http://example.org/a"),
 rdflib.term.URIRef("http://www.example.org/b"),
 rdflib.term.Literal("foo"),
 None)
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/y"),
 rdflib.term.Literal("bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))

```

(continues on next page)

(continued from previous page)

```

(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/z"),
 rdflib.term.Literal("foo-bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
>>>
>>> # restricting iteration to a graph:
>>> for q in ds.quads((None, None, None, g)):
...     print(q)
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/y"),
 rdflib.term.Literal("bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/z"),
 rdflib.term.Literal("foo-bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
>>> # Note that in the call above -
>>> # ds.quads((None, None, None, "http://www.example.com/gr"))
>>> # would have been accepted, too
>>>
>>> # graph names in the dataset can be queried:
>>> for c in ds.graphs():
...     print(c) # doctest:
DEFAULT
http://www.example.com/gr
>>> # A graph can be created without specifying a name; a skolemized genid
>>> # is created on the fly
>>> h = ds.graph()
>>> for c in ds.graphs():
...     print(c)
DEFAULT
https://rdflib.github.io/.well-known/genid/rdflib/N...
http://www.example.com/gr
>>> # Note that the Dataset.graphs() call returns names of empty graphs,
>>> # too. This can be restricted:
>>> for c in ds.graphs(empty=False):
...     print(c)
DEFAULT
http://www.example.com/gr
>>>
>>> # a graph can also be removed from a dataset via ds.remove_graph(g)

```

New in version 4.0.

#### Parameters

- **store** (`Union[Store, str]`) –
- **default\_union** (`bool`) –
- **default\_graph\_base** (`Optional[str]`) –

`__getstate__()`

**Return type**

```
    Tuple[Store, IdentifiedNode, Graph, bool]
__init__(store='default', default_union=False, default_graph_base=None)

    Parameters
        • store (Union[Store, str]) –
        • default_union (bool) –
        • default_graph_base (Optional[str]) –

__iter__()
    Iterates over all quads in the store

    Return type
        Generator[Tuple[Node, Node, Node, Optional[IdentifiedNode]], None, None]

__module__ = 'rdflib.graph'

__reduce__()
    Helper for pickle.

    Return type
        Tuple[Type[Dataset], Tuple[Store, bool]]

__setstate__(state)

    Parameters
        state (Tuple[Store, IdentifiedNode, Graph, bool]) –

    Return type
        None

__str__()
    Return str(self).

    Return type
        str

add_graph(g)
    alias of graph for consistency

    Parameters
        g (Union[IdentifiedNode, Graph, str, None]) –

    Return type
        Graph

contexts(triple=None)
    Iterate over all contexts in the graph
    If triple is specified, iterate over all contexts the triple is in.

    Parameters
        triple (Optional[Tuple[Node, Node, Node]]) –

    Return type
        Generator[Graph, None, None]

default_context: Graph
```

**graph**(*identifier=None, base=None*)

**Parameters**

- **identifier** (Union[*IdentifiedNode*, *Graph*, str, None]) –
- **base** (Optional[str]) –

**Return type**

*Graph*

**graphs**(*triple=None*)

Iterate over all contexts in the graph

If triple is specified, iterate over all contexts the triple is in.

**Parameters**

**triple** (Optional[Tuple[*Node*, *Node*, *Node*]]) –

**Return type**

Generator[*Graph*, None, None]

**parse**(*source=None, publicID=None, format=None, location=None, file=None, data=None, \*\*args*)

**Caution:** This method can access directly or indirectly requested network or file resources, for example, when parsing JSON-LD documents with @context directives that point to a network location.

When processing untrusted or potentially malicious documents, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

**Parameters**

- **source** (Union[IO[bytes], TextIO, *InputSource*, str, bytes, PurePath, None]) –
- **publicID** (Optional[str]) –
- **format** (Optional[str]) –
- **location** (Optional[str]) –
- **file** (Union[BinaryIO, TextIO, None]) –
- **data** (Union[str, bytes, None]) –
- **args** (*Any*) –

**Return type**

*Graph*

**quads**(*quad=None*)

Iterate over all the quads in the entire conjunctive graph

**Parameters**

**quad** (Union[Tuple[Optional[*Node*], Optional[*Node*], Optional[*Node*]], Tuple[Optional[*Node*], Optional[*Node*], Optional[*Node*], Optional[*Graph*]], None]) –

**Return type**

Generator[Tuple[*Node*, *Node*, *Node*, Optional[*IdentifiedNode*]], None, None]

**remove\_graph**(*g*)

**Parameters**

- **self** (*TypeVar*(*\_DatasetT*, bound= *Dataset*)) –
- **g** (*Union*[*IdentifiedNode*, *Graph*, *str*, *None*]) –

**Return type**

*TypeVar*(*\_DatasetT*, bound= *Dataset*)

**class** `rdflib.graph.Graph`(*store*='default', *identifier*=*None*, *namespace\_manager*=*None*, *base*=*None*,  
                              *bind\_namespaces*='rdflib')

Bases: *Node*

An RDF Graph

The constructor accepts one argument, the “store” that will be used to store the graph data (see the “store” package for stores currently shipped with rdflib).

Stores can be context-aware or unaware. Unaware stores take up (some) less space but cannot support features that require context, such as true merging/demerging of sub-graphs and provenance.

Even if used with a context-aware store, Graph will only expose the quads which belong to the default graph. To access the rest of the data, *ConjunctiveGraph* or *Dataset* classes can be used instead.

The Graph constructor can take an identifier which identifies the Graph by name. If none is given, the graph is assigned a BNode for its identifier.

For more on named graphs, see: <http://www.w3.org/2004/03/trix/>

**Parameters**

- **store** (*Union*[*Store*, *str*]) –
- **identifier** (*Union*[*IdentifiedNode*, *str*, *None*]) –
- **namespace\_manager** (*Optional*[*NamespaceManager*]) –
- **base** (*Optional*[*str*]) –
- **bind\_namespaces** (*Literal*['core', 'rdflib', 'none']) –

**\_\_add\_\_**(*other*)

Set-theoretic union BNode IDs are not changed.

**Parameters**

**other** (*Graph*) –

**Return type**

*Graph*

**\_\_and\_\_**(*other*)

Set-theoretic intersection. BNode IDs are not changed.

**Parameters**

**other** (*Graph*) –

**Return type**

*Graph*

**\_\_cmp\_\_**(*other*)

**Return type**

*int*

**\_\_contains\_\_**(*triple*)

Support for 'triple in graph' syntax

**Parameters**

**triple** (Tuple[Optional[Node], Optional[Node], Optional[Node]]) –

**Return type**

bool

```

__dict__ = mappingproxy({'__module__': 'rdflib.graph', '__doc__': 'An RDF
Graph\n\n The constructor accepts one argument, the "store"\n that will be used to
store the graph data (see the "store"\n package for stores currently shipped with
rdflib).\n\n Stores can be context-aware or unaware. Unaware stores take up\n (some)
less space but cannot support features that require\n context, such as true
merging/demerging of sub-graphs and\n provenance.\n\n Even if used with a
context-aware store, Graph will only expose the quads which\n belong to the default
graph. To access the rest of the data, `ConjunctiveGraph` or\n `Dataset` classes can
be used instead.\n\n The Graph constructor can take an identifier which identifies
the Graph\n by name. If none is given, the graph is assigned a BNode for its\n
identifier.\n\n For more on named graphs, see: http://www.w3.org/2004/03/trix/\n ',
'__init__': <function Graph.__init__>, 'store': <property object>, 'identifier':
<property object>, 'namespace_manager': <property object>, '__repr__': <function
Graph.__repr__>, '__str__': <function Graph.__str__>, 'toPython': <function
Graph.toPython>, 'destroy': <function Graph.destroy>, 'commit': <function
Graph.commit>, 'rollback': <function Graph.rollback>, 'open': <function
Graph.open>, 'close': <function Graph.close>, 'add': <function Graph.add>, 'addN':
<function Graph.addN>, 'remove': <function Graph.remove>, 'triples': <function
Graph.triples>, '__getitem__': <function Graph.__getitem__>, '__len__': <function
Graph.__len__>, '__iter__': <function Graph.__iter__>, '__contains__': <function
Graph.__contains__>, '__hash__': <function Graph.__hash__>, '__cmp__': <function
Graph.__cmp__>, '__eq__': <function Graph.__eq__>, '__lt__': <function
Graph.__lt__>, '__le__': <function Graph.__le__>, '__gt__': <function
Graph.__gt__>, '__ge__': <function Graph.__ge__>, '__iadd__': <function
Graph.__iadd__>, '__isub__': <function Graph.__isub__>, '__add__': <function
Graph.__add__>, '__mul__': <function Graph.__mul__>, '__sub__': <function
Graph.__sub__>, '__xor__': <function Graph.__xor__>, '__or__': <function
Graph.__add__>, '__and__': <function Graph.__mul__>, 'set': <function Graph.set>,
'subjects': <function Graph.subjects>, 'predicates': <function Graph.predicates>,
'objects': <function Graph.objects>, 'subject_predicates': <function
Graph.subject_predicates>, 'subject_objects': <function Graph.subject_objects>,
'predicate_objects': <function Graph.predicate_objects>, 'triples_choices':
<function Graph.triples_choices>, 'value': <function Graph.value>, 'items':
<function Graph.items>, 'transitiveClosure': <function Graph.transitiveClosure>,
'transitive_objects': <function Graph.transitive_objects>, 'transitive_subjects':
<function Graph.transitive_subjects>, 'qname': <function Graph.qname>,
'compute_qname': <function Graph.compute_qname>, 'bind': <function Graph.bind>,
'namespaces': <function Graph.namespaces>, 'absolutize': <function
Graph.absolutize>, 'serialize': <function Graph.serialize>, 'print': <function
Graph.print>, 'parse': <function Graph.parse>, 'query': <function Graph.query>,
'update': <function Graph.update>, 'n3': <function Graph.n3>, '__reduce__':
<function Graph.__reduce__>, 'isomorphic': <function Graph.isomorphic>,
'connected': <function Graph.connected>, 'all_nodes': <function Graph.all_nodes>,
'collection': <function Graph.collection>, 'resource': <function Graph.resource>,
'_process_skolem_tuples': <function Graph._process_skolem_tuples>, 'skolemize':
<function Graph.skolemize>, 'de_skolemize': <function Graph.de_skolemize>, 'cbd':
<function Graph.cbd>, '__dict__': <attribute '__dict__' of 'Graph' objects>,
'__weakref__': <attribute '__weakref__' of 'Graph' objects>, '__annotations__':
{'__identifier': '_ContextIdentifierType', '__store': 'Store'}})

```

`__eq__(other)`

Return self==value.

Return type



bool

**\_\_ge\_\_**(*other*)

Return self>=value.

**Parameters**

**other** (*Graph*) –

**Return type**

bool

**\_\_getitem\_\_**(*item*)

A graph can be “sliced” as a shortcut for the triples method The python slice syntax is (ab)used for specifying triples. A generator over matches is returned, the returned tuples include only the parts not given

```
>>> import rdflib
>>> g = rdflib.Graph()
>>> g.add((rdflib.URIRef("urn:bob"), namespace.RDFS.label, rdflib.Literal("Bob
↳"))))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
```

```
>>> list(g[rdflib.URIRef("urn:bob")]) # all triples about bob
[(rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#label'), rdflib.term.
↳Literal('Bob'))]
```

```
>>> list(g[:namespace.RDFS.label]) # all label triples
[(rdflib.term.URIRef('urn:bob'), rdflib.term.Literal('Bob'))]
```

```
>>> list(g[:,rdflib.Literal("Bob")]) # all triples with bob as object
[(rdflib.term.URIRef('urn:bob'), rdflib.term.URIRef('http://www.w3.org/2000/01/
↳rdf-schema#label'))]
```

Combined with SPARQL paths, more complex queries can be written concisely:

Name of all Bobs friends:

```
g[bob : FOAF.knows/FOAF.name ]
```

Some label for Bob:

```
g[bob : DC.title|FOAF.name|RDFS.label]
```

All friends and friends of friends of Bob

```
g[bob : FOAF.knows * "+"]
```

etc.

New in version 4.0.

**\_\_gt\_\_**(*other*)

Return self>value.

**Return type**

bool

**\_\_hash\_\_**()

Return hash(self).

**Return type**

int

**\_\_iadd\_\_**(*other*)

Add all triples in Graph *other* to Graph. BNode IDs are not changed.

**Parameters**

- **self** (`TypeVar`(`_GraphT`, bound= `Graph`)) –
- **other** (`Iterable`[`Tuple`[`Node`, `Node`, `Node`]]) –

**Return type**`TypeVar`(`_GraphT`, bound= `Graph`)**\_\_init\_\_**(*store*='default', *identifier*=None, *namespace\_manager*=None, *base*=None, *bind\_namespaces*='rdflib')**Parameters**

- **store** (`Union`[`Store`, `str`]) –
- **identifier** (`Union`[`IdentifiedNode`, `str`, `None`]) –
- **namespace\_manager** (`Optional`[`NamespaceManager`]) –
- **base** (`Optional`[`str`]) –
- **bind\_namespaces** (`Literal`['core', 'rdflib', 'none']) –

**\_\_isub\_\_**(*other*)

Subtract all triples in Graph *other* from Graph. BNode IDs are not changed.

**Parameters**

- **self** (`TypeVar`(`_GraphT`, bound= `Graph`)) –
- **other** (`Iterable`[`Tuple`[`Node`, `Node`, `Node`]]) –

**Return type**`TypeVar`(`_GraphT`, bound= `Graph`)**\_\_iter\_\_**()

Iterates over all triples in the store

**Return type**`Generator`[`Tuple`[`Node`, `Node`, `Node`], `None`, `None`]**\_\_le\_\_**(*other*)

Return self<=value.

**Parameters**

- **other** (`Graph`) –

**Return type**`bool`**\_\_len\_\_**()

Returns the number of triples in the graph

If context is specified then the number of triples in the context is returned instead.

**Return type**`int`

```

__lt__(other)
    Return self<value.

    Return type
    bool

__module__ = 'rdflib.graph'

__mul__(other)
    Set-theoretic intersection. BNode IDs are not changed.

    Parameters
    other (Graph) –

    Return type
    Graph

__or__(other)
    Set-theoretic union BNode IDs are not changed.

    Parameters
    other (Graph) –

    Return type
    Graph

__reduce__()
    Helper for pickle.

    Return type
    Tuple[Type[Graph], Tuple[Store, IdentifiedNode]]

__repr__()
    Return repr(self).

    Return type
    str

__str__()
    Return str(self).

    Return type
    str

__sub__(other)
    Set-theoretic difference. BNode IDs are not changed.

    Parameters
    other (Graph) –

    Return type
    Graph

__weakref__
    list of weak references to the object (if defined)

__xor__(other)
    Set-theoretic XOR. BNode IDs are not changed.

    Parameters
    other (Graph) –

```

**Return type***Graph***absolutize**(*uri*, *defrag*=1)

Turn uri into an absolute URI if it's not one already

**Parameters**

- **uri** (*str*) –
- **defrag** (*int*) –

**Return type***URIRef***add**(*triple*)

Add a triple with self as context

**Parameters**

- **self** (*TypeVar*(*\_GraphT*, bound= *Graph*)) –
- **triple** (*Tuple*[*Node*, *Node*, *Node*]) –

**Return type***TypeVar*(*\_GraphT*, bound= *Graph*)**addN**(*quads*)

Add a sequence of triple with context

**Parameters**

- **self** (*TypeVar*(*\_GraphT*, bound= *Graph*)) –
- **quads** (*Iterable*[*Tuple*[*Node*, *Node*, *Node*, *Graph*]]) –

**Return type***TypeVar*(*\_GraphT*, bound= *Graph*)**all\_nodes**()**Return type***Set*[*Node*]**bind**(*prefix*, *namespace*, *override*=*True*, *replace*=*False*)

Bind prefix to namespace

If override is True will bind namespace to given prefix even if namespace was already bound to a different prefix.

if replace, replace any existing prefix with the new namespace

for example: graph.bind("foaf", "http://xmlns.com/foaf/0.1/")

**Parameters**

- **prefix** (*Optional*[*str*]) –
- **namespace** (*Any*) –
- **override** (*bool*) –
- **replace** (*bool*) –

**Return type***None*

**cbd(resource)**

Retrieves the Concise Bounded Description of a Resource from a Graph

Concise Bounded Description (CBD) is defined in [1] as:

Given a particular node (the starting node) in a particular RDF graph (the source graph), a subgraph of that particular graph, taken to comprise a concise bounded description of the resource denoted by the starting node, can be identified as follows:

1. **Include in the subgraph all statements in the source graph where the subject of the statement is the**  
starting node;
2. **Recursively, for all statements identified in the subgraph thus far having a blank node object, include**  
in the subgraph all statements in the source graph where the subject of the statement is the blank node in question and which are not already included in the subgraph.
3. **Recursively, for all statements included in the subgraph thus far, for all reifications of each statement**  
in the source graph, include the concise bounded description beginning from the `rdf:Statement` node of each reification.

This results in a subgraph where the object nodes are either URI references, literals, or blank nodes not serving as the subject of any statement in the graph.

[1] <https://www.w3.org/Submission/CBD/>

**Parameters**

**resource** (*Node*) – a `URIRef` object, of the Resource for queried for

**Return type**

*Graph*

**Returns**

a `Graph`, subgraph of self

**close(commit\_pending\_transaction=False)**

Close the graph store

Might be necessary for stores that require closing a connection to a database or releasing some resource.

**Parameters**

**commit\_pending\_transaction** (*bool*) –

**Return type**

*None*

**collection(identifier)**

Create a new `Collection` instance.

Parameters:

- **identifier**: a `URIRef` or `BNode` instance.

Example:

```
>>> graph = Graph()
>>> uri = URIRef("http://example.org/resource")
>>> collection = graph.collection(uri)
>>> assert isinstance(collection, Collection)
>>> assert collection.uri is uri
```

(continues on next page)

(continued from previous page)

```
>>> assert collection.graph is graph
>>> collection += [ Literal(1), Literal(2) ]
```

**Parameters****identifier** (*Node*) –**Return type***Collection***commit()**

Commits active transactions

**Parameters****self** (*TypeVar*(*\_GraphT*, bound= *Graph*)) –**Return type***TypeVar*(*\_GraphT*, bound= *Graph*)**compute\_qname(uri, generate=True)****Parameters**• **uri** (*str*) –• **generate** (*bool*) –**Return type***Tuple*[*str*, *URIRef*, *str*]**connected()**

Check if the Graph is connected

The Graph is considered undirectional.

Performs a search on the Graph, starting from a random node. Then iteratively goes depth-first through the triplets where the node is subject and object. Return True if all nodes have been visited and False if it cannot continue and there are still unvisited nodes left.

**Return type***bool***de\_skolemize(new\_graph=None, uriref=None)****Parameters**• **new\_graph** (*Optional*[*Graph*]) –• **uriref** (*Optional*[*URIRef*]) –**Return type***Graph***destroy(configuration)**

Destroy the store identified by configuration if supported

**Parameters**• **self** (*TypeVar*(*\_GraphT*, bound= *Graph*)) –• **configuration** (*str*) –

**Return type**`TypeVar(_GraphT, bound= Graph)`**property identifier:** `IdentifiedNode`**isomorphic**(*other*)

does a very basic check if these graphs are the same If no BNodes are involved, this is accurate.

See `rdflib.compare` for a correct implementation of isomorphism checks**Parameters****other** (`Graph`) –**Return type**`bool`**items**(*list*)

Generator over all items in the resource specified by list

list is an RDF collection.

**Parameters****list** (`Node`) –**Return type**`Generator[Node, None, None]`**n3**()

Return an n3 identifier for the Graph

**Return type**`str`**property namespace\_manager:** `NamespaceManager`

this graph's namespace-manager

**namespaces**()

Generator over all the prefix, namespace tuples

**Return type**`Generator[Tuple[str, URIRef], None, None]`**objects**(*subject=None, predicate=None, unique=False*)

A generator of (optionally unique) objects with the given subject and predicate

**Parameters**

- **subject** (`Optional[Node]`) –
- **predicate** (`Union[None, Path, Node]`) –
- **unique** (`bool`) –

**Return type**`Generator[Node, None, None]`**open**(*configuration, create=False*)

Open the graph store

Might be necessary for stores that require opening a connection to a database or acquiring some resource.

**Parameters**

- **configuration** (`str`) –

- **create**(bool) –

**Return type**

Optional[int]

**parse**(source=None, publicID=None, format=None, location=None, file=None, data=None, \*\*args)

Parse an RDF source adding the resulting triples to the Graph.

The source is specified using one of source, location, file or data.

**Caution:** This method can access directly or indirectly requested network or file resources, for example, when parsing JSON-LD documents with `@context` directives that point to a network location.

When processing untrusted or potentially malicious documents, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib [Security Considerations](#) documentation.

**Parameters**

- **source:** An InputSource, file-like object, or string. In the case of a string the string is the location of the source.
- **location:** A string indicating the relative or absolute URL of the source. Graph's `absolutize` method is used if a relative location is specified.
- **file:** A file-like object.
- **data:** A string containing the data to be parsed.
- **format:** Used if format can not be determined from source, e.g. file extension or Media Type. Defaults to `text/turtle`. Format support can be extended with plugins, but `"xml"`, `"n3"` (use for turtle), `"nt"` & `"trix"` are built in.
- **publicID:** the logical URI to use as the document base. If None specified the document location is used (at least in the case where there is a document location).

**Returns**

- self, the graph instance.

Examples:

```
>>> my_data = '''
... <rdf:RDF
...   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
...   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
... >
...   <rdf:Description>
...     <rdfs:label>Example</rdfs:label>
...     <rdfs:comment>This is really just an example.</rdfs:comment>
...   </rdf:Description>
... </rdf:RDF>
... '''
>>> import os, tempfile
>>> fd, file_name = tempfile.mkstemp()
>>> f = os.fdopen(fd, "w")
```

(continues on next page)



(continued from previous page)

```
>>> dummy = f.write(my_data) # Returns num bytes written
>>> f.close()
```

```
>>> g = Graph()
>>> result = g.parse(data=my_data, format="application/rdf+xml")
>>> len(g)
2
```

```
>>> g = Graph()
>>> result = g.parse(location=file_name, format="application/rdf+xml")
>>> len(g)
2
```

```
>>> g = Graph()
>>> with open(file_name, "r") as f:
...     result = g.parse(f, format="application/rdf+xml")
>>> len(g)
2
```

```
>>> os.remove(file_name)
```

```
>>> # default turtle parsing
>>> result = g.parse(data="<http://example.com/a> <http://example.com/a> <http://
↳ /example.com/a> .")
>>> len(g)
3
```

### Parameters

- **source** (`Union[IO[bytes], TextIO, InputSource, str, bytes, PurePath, None]`) –
- **publicID** (`Optional[str]`) –
- **format** (`Optional[str]`) –
- **location** (`Optional[str]`) –
- **file** (`Union[BinaryIO, TextIO, None]`) –
- **data** (`Union[str, bytes, None]`) –
- **args** (`Any`) –

### Return type

*Graph*

### **predicate\_objects**(*subject=None, unique=False*)

A generator of (optionally unique) (predicate, object) tuples for the given subject

### Parameters

- **subject** (`Optional[Node]`) –
- **unique** (`bool`) –

### Return type

`Generator[Tuple[Node, Node], None, None]`

**predicates**(*subject=None, object=None, unique=False*)

A generator of (optionally unique) predicates with the given subject and object

**Parameters**

- **subject** (*Optional[Node]*) –
- **object** (*Optional[Node]*) –
- **unique** (*bool*) –

**Return type**

*Generator[Node, None, None]*

**print**(*format='turtle', encoding='utf-8', out=None*)

**Parameters**

- **format** (*str*) –
- **encoding** (*str*) –
- **out** (*Optional[TextIO]*) –

**Return type**

*None*

**qname**(*uri*)

**Parameters**

**uri** (*str*) –

**Return type**

*str*

**query**(*query\_object, processor='sparql', result='sparql', initNs=None, initBindings=None, use\_store\_provided=True, \*\*kwargs*)

Query this graph.

A type of ‘prepared queries’ can be realised by providing initial variable bindings with `initBindings`

Initial namespaces are used to resolve prefixes used in the query, if none are given, the namespaces from the graph’s namespace manager are used.

**Caution:** This method can access indirectly requested network endpoints, for example, query processing will attempt to access network endpoints specified in `SERVICE` directives.

When processing untrusted or potentially malicious queries, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

**Returntype**

*Result*

**Parameters**

- **query\_object** (*Union[str, Query]*) –
- **processor** (*Union[str, Processor]*) –
- **result** (*Union[str, Type[Result]]*) –

- **initNs** (`Optional[Mapping[str, Any]]`) –
- **initBindings** (`Optional[Mapping[str, Identifier]]`) –
- **use\_store\_provided** (`bool`) –
- **kwargs** (`Any`) –

**Return type**  
`Result`

**remove**(*triple*)

Remove a triple from the graph

If the triple does not provide a context attribute, removes the triple from all contexts.

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **triple** (`Tuple[Optional[Node], Optional[Node], Optional[Node]]`) –

**Return type**  
`TypeVar(_GraphT, bound= Graph)`

**resource**(*identifier*)

Create a new Resource instance.

Parameters:

- **identifier**: a URIRef or BNode instance.

Example:

```
>>> graph = Graph()
>>> uri = URIRef("http://example.org/resource")
>>> resource = graph.resource(uri)
>>> assert isinstance(resource, Resource)
>>> assert resource.identifier is uri
>>> assert resource.graph is graph
```

**Parameters**  
**identifier** (`Union[Node, str]`) –

**Return type**  
`Resource`

**rollback**()

Rollback active transactions

**Parameters**  
**self** (`TypeVar(_GraphT, bound= Graph)`) –

**Return type**  
`TypeVar(_GraphT, bound= Graph)`

**serialize**(*destination=None, format='turtle', base=None, encoding=None, \*\*args*)

Serialize the graph.

**Parameters**

- **destination** (`Union[str, PurePath, IO[bytes], None]`) – The destination to serialize the graph to. This can be a path as a `str` or `PurePath` object, or it can be a `IO[bytes]` like object. If this parameter is not supplied the serialized graph will be returned.
- **format** (`str`) – The format that the output should be written in. This value references a `Serializer` plugin. Format support can be extended with plugins, but "xml", "n3", "turtle", "nt", "pretty-xml", "trix", "trig", "nquads", "json-ld" and "hex" are built in. Defaults to "turtle".
- **base** (`Optional[str]`) – The base IRI for formats that support it. For the turtle format this will be used as the @base directive.
- **encoding** (`Optional[str]`) – Encoding of output.
- **args** (`Any`) – Additional arguments to pass to the `Serializer` that will be used.
- **self** (`TypeVar(_GraphT, bound= Graph)`) –

**Returns**

The serialized graph if *destination* is *None*. The serialized graph is returned as *str* if no encoding is specified, and as *bytes* if an encoding is specified.

**Return type**

`bytes` if *destination* is *None* and encoding is not *None*.

**Return type**

`str` if *destination* is *None* and encoding is *None*.

**Returns**

*self* (i.e. the `Graph` instance) if *destination* is not *None*.

**Return type**

`Graph` if *destination* is not *None*.

**set(*triple*)**

Convenience method to update the value of object

Remove any existing triples for subject and predicate before adding (subject, predicate, object).

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **triple** (`Tuple[Node, Node, Node]`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

**skolemize(*new\_graph=None, bnode=None, authority=None, basepath=None*)****Parameters**

- **new\_graph** (`Optional[Graph]`) –
- **bnode** (`Optional[BNode]`) –
- **authority** (`Optional[str]`) –
- **basepath** (`Optional[str]`) –

**Return type**

`Graph`

property store: `Store`

**subject\_objects**(*predicate=None, unique=False*)

A generator of (optionally unique) (subject, object) tuples for the given predicate

**Parameters**

- **predicate** (`Union[None, Path, Node]`) –
- **unique** (`bool`) –

**Return type**

`Generator[Tuple[Node, Node], None, None]`

**subject\_predicates**(*object=None, unique=False*)

A generator of (optionally unique) (subject, predicate) tuples for the given object

**Parameters**

- **object** (`Optional[Node]`) –
- **unique** (`bool`) –

**Return type**

`Generator[Tuple[Node, Node], None, None]`

**subjects**(*predicate=None, object=None, unique=False*)

A generator of (optionally unique) subjects with the given predicate and object

**Parameters**

- **predicate** (`Union[None, Path, Node]`) –
- **object** (`Optional[Node]`) –
- **unique** (`bool`) –

**Return type**

`Generator[Node, None, None]`

**toPython()**

**Parameters**

**self** (`TypeVar(_GraphT, bound= Graph)`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

**transitiveClosure**(*func, arg, seen=None*)

Generates transitive closure of a user-defined function against the graph

```
>>> from rdflib.collection import Collection
>>> g=Graph()
>>> a=BNode("foo")
>>> b=BNode("bar")
>>> c=BNode("baz")
>>> g.add((a,RDF.first,RDF.type))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((a,RDF.rest,b))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((b,RDF.first,namespace.RDFS.label))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((b,RDF.rest,c))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
```

(continues on next page)

(continued from previous page)

```

>>> g.add((c,RDF.first,namespace.RDFS.comment))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((c,RDF.rest,RDF.nil))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> def topList(node,g):
...     for s in g.subjects(RDF.rest, node):
...         yield s
>>> def reverseList(node,g):
...     for f in g.objects(node, RDF.first):
...         print(f)
...     for s in g.subjects(RDF.rest, node):
...         yield s

```

```

>>> [rt for rt in g.transitiveClosure(
...     topList,RDF.nil)]
[rdflib.term.BNode('baz'),
 rdflib.term.BNode('bar'),
 rdflib.term.BNode('foo')]

```

```

>>> [rt for rt in g.transitiveClosure(
...     reverseList,RDF.nil)]
http://www.w3.org/2000/01/rdf-schema#comment
http://www.w3.org/2000/01/rdf-schema#label
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
[rdflib.term.BNode('baz'),
 rdflib.term.BNode('bar'),
 rdflib.term.BNode('foo')]

```

**Parameters**

- **func** (Callable[[TypeVar(\_TCArgT), *Graph*], Iterable[TypeVar(\_TCArgT)]] –
- **arg** (TypeVar(\_TCArgT)) –
- **seen** (Optional[Dict[TypeVar(\_TCArgT), int]]) –

**transitive\_objects**(*subject*, *predicate*, *remember=None*)

Transitively generate objects for the *predicate* relationship

Generated objects belong to the depth first transitive closure of the *predicate* relationship starting at *subject*.

**Parameters**

- **subject** (Optional[*Node*]) –
- **predicate** (Optional[*Node*]) –
- **remember** (Optional[Dict[Optional[*Node*], int]]) –

**Return type**

Generator[Optional[*Node*], None, None]

**transitive\_subjects**(*predicate*, *object*, *remember=None*)

Transitively generate subjects for the *predicate* relationship

Generated subjects belong to the depth first transitive closure of the predicate relationship starting at object.

#### Parameters

- **predicate** (`Optional[Node]`) –
- **object** (`Optional[Node]`) –
- **remember** (`Optional[Dict[Optional[Node], int]]`) –

#### Return type

`Generator[Optional[Node], None, None]`

### **triples**(*triple*)

Generator over the triple store

Returns triples that match the given triple pattern. If triple pattern does not provide a context, all contexts will be searched.

#### Parameters

**triple** (`Tuple[Optional[Node], Union[Path, Node, None], Optional[Node]]`) –

#### Return type

`Generator[Union[Tuple[Node, Node, Node], Tuple[Node, Path, Node]], None, None]`

### **triples\_choices**(*triple*, *context=None*)

#### Parameters

- **triple** (`Union[Tuple[List[Node], Node, Node], Tuple[Node, List[Node], Node], Tuple[Node, Node, List[Node]]]`) –
- **context** (`Optional[Graph]`) –

#### Return type

`Generator[Tuple[Node, Node, Node], None, None]`

### **update**(*update\_object*, *processor='sparql'*, *initNs=None*, *initBindings=None*, *use\_store\_provided=True*, *\*\*kwargs*)

Update this graph with the given update query.

**Caution:** This method can access indirectly requested network endpoints, for example, query processing will attempt to access network endpoints specified in SERVICE directives.

When processing untrusted or potentially malicious queries, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

#### Parameters

- **update\_object** (`Union[Update, str]`) –
- **processor** (`Union[str, UpdateProcessor]`) –
- **initNs** (`Optional[Mapping[str, Any]]`) –
- **initBindings** (`Optional[Mapping[str, Identifier]]`) –
- **use\_store\_provided** (`bool`) –

- **kwargs** (*Any*) –

**Return type***None*

**value**(*subject=None, predicate=rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#value'), object=None, default=None, any=True*)

Get a value for a pair of two criteria

Exactly one of subject, predicate, object must be None. Useful if one knows that there may only be one value.

It is one of those situations that occur a lot, hence this ‘macro’ like utility

Parameters: subject, predicate, object – exactly one must be None default – value to be returned if no values found any – if True, return any value in the case there is more than one, else, raise UniquenessError

**Parameters**

- **subject** (*Optional[Node]*) –
- **predicate** (*Optional[Node]*) –
- **object** (*Optional[Node]*) –
- **default** (*Optional[Node]*) –
- **any** (*bool*) –

**Return type***Optional[Node]*

**exception** `rdflib.graph.ModificationException`

Bases: *Exception*

**\_\_init\_\_**()

**\_\_module\_\_** = `'rdflib.graph'`

**\_\_str\_\_**()

Return str(self).

**Return type***str*

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**class** `rdflib.graph.QuotedGraph(store, identifier)`

Bases: *Graph*

Quoted Graphs are intended to implement Notation 3 formulae. They are associated with a required identifier that the N3 parser *must* provide in order to maintain consistent formulae identification for scenarios such as implication and other such processing.

**Parameters**

- **store** (*Union[Store, str]*) –
- **identifier** (*Union[IdentifiedNode, str, None]*) –



`__init__(store, identifier)`

**Parameters**

- **store** (`Union[Store, str]`) –
- **identifier** (`Union[IdentifiedNode, str, None]`) –

`__module__ = 'rdflib.graph'`

`__reduce__()`

Helper for pickle.

**Return type**

`Tuple[Type[Graph], Tuple[Store, IdentifiedNode]]`

`__str__()`

Return str(self).

**Return type**

`str`

`add(triple)`

Add a triple with self as context

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **triple** (`Tuple[Node, Node, Node]`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

`addN(quads)`

Add a sequence of triple with context

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **quads** (`Iterable[Tuple[Node, Node, Node, Graph]]`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

`n3()`

Return an n3 identifier for the Graph

**Return type**

`str`

`class rdflib.graph.ReadOnlyGraphAggregate(graphs, store='default')`

Bases: `ConjunctiveGraph`

Utility class for treating a set of graphs as a single graph

Only read operations are supported (hence the name). Essentially a `ConjunctiveGraph` over an explicit subset of the entire store.

**Parameters**

- **graphs** (`List[Graph]`) –
- **store** (`Union[str, Store]`) –

`__cmp__(other)`

**Return type**

`int`

`__contains__(triple_or_quad)`

Support for 'triple/quad in graph' syntax

**Parameters**

**triple\_or\_quad** (`Union[Tuple[Optional[Node], Optional[Node], Optional[Node]], Tuple[Optional[Node], Optional[Node], Optional[Node], Optional[Graph]]]`) –

**Return type**

`bool`

`__hash__()`

Return hash(self).

**Return type**

`NoReturn`

`__iadd__(other)`

Add all triples in Graph other to Graph. BNode IDs are not changed.

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **other** (`Iterable[Tuple[Node, Node, Node]]`) –

**Return type**

`NoReturn`

`__init__(graphs, store='default')`

**Parameters**

- **graphs** (`List[Graph]`) –
- **store** (`Union[str, Store]`) –

`__isub__(other)`

Subtract all triples in Graph other from Graph. BNode IDs are not changed.

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **other** (`Iterable[Tuple[Node, Node, Node]]`) –

**Return type**

`NoReturn`

`__len__()`

Number of triples in the entire conjunctive graph

**Return type**

`int`

`__module__ = 'rdflib.graph'`

**`__reduce__()`**

Helper for pickle.

**Return type**

`NoReturn`

**`__repr__()`**

Return repr(self).

**Return type**

`str`

**`absolutize(uri, defrag=1)`**

Turn uri into an absolute URI if it's not one already

**Parameters**

- **uri** (`str`) –
- **defrag** (`int`) –

**Return type**

`NoReturn`

**`add(triple)`**

Add a triple or quad to the store.

if a triple is given it is added to the default context

**Parameters**

**triple** (`Union[Tuple[Node, Node, Node], Tuple[Node, Node, Node, Optional[Graph]]]`) –

**Return type**

`NoReturn`

**`addN(quads)`**

Add a sequence of triples with context

**Parameters**

**quads** (`Iterable[Tuple[Node, Node, Node, Graph]]`) –

**Return type**

`NoReturn`

**`bind(prefix, namespace, override=True)`**

Bind prefix to namespace

If override is True will bind namespace to given prefix even if namespace was already bound to a different prefix.

if replace, replace any existing prefix with the new namespace

for example: graph.bind("foaf", "http://xmlns.com/foaf/0.1/")

**Parameters**

- **prefix** (`Optional[str]`) –
- **namespace** (`Any`) –
- **override** (`bool`) –

**Return type**

`NoReturn`

**close()**

Close the graph store

Might be necessary for stores that require closing a connection to a database or releasing some resource.

**Return type**

`None`

**commit()**

Commits active transactions

**Return type**

`NoReturn`

**compute\_qname(uri, generate=True)****Parameters**

- **uri** (`str`) –
- **generate** (`bool`) –

**Return type**

`Tuple[str, URIRef, str]`

**default\_context:** `Graph`**destroy(configuration)**

Destroy the store identified by `configuration` if supported

**Parameters**

**configuration** (`str`) –

**Return type**

`NoReturn`

**n3()**

Return an n3 identifier for the Graph

**Return type**

`NoReturn`

**namespaces()**

Generator over all the prefix, namespace tuples

**Return type**

`Generator[Tuple[str, URIRef], None, None]`

**open(configuration, create=False)**

Open the graph store

Might be necessary for stores that require opening a connection to a database or acquiring some resource.

**Parameters**

- **configuration** (`str`) –
- **create** (`bool`) –

**Return type**

`None`

**parse**(*source*, *publicID*=None, *format*=None, *\*\*args*)

Parse source adding the resulting triples to its own context (sub graph of this graph).

See `rdflib.graph.Graph.parse()` for documentation on arguments.

#### Returns

The graph into which the source was parsed. In the case of n3 it returns the root context.

**Caution:** This method can access directly or indirectly requested network or file resources, for example, when parsing JSON-LD documents with `@context` directives that point to a network location.

When processing untrusted or potentially malicious documents, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

#### Parameters

- **source** (`Union[IO[bytes], TextIO, InputSource, str, bytes, PurePath, None]`) –
- **publicID** (`Optional[str]`) –
- **format** (`Optional[str]`) –
- **args** (`Any`) –

#### Return type

`NoReturn`

**qname**(*uri*)

#### Parameters

- **uri** (`str`) –

#### Return type

`str`

**quads**(*triple\_or\_quad*)

Iterate over all the quads in the entire aggregate graph

#### Parameters

- **triple\_or\_quad** (`Union[Tuple[Optional[Node], Union[Path, Node, None], Optional[Node]], Tuple[Optional[Node], Union[Path, Node, None], Optional[Node], Optional[Graph]]]`) –

#### Return type

`Generator[Tuple[Node, Union[Path, Node], Node, Graph], None, None]`

**remove**(*triple*)

Removes a triple or quads

if a triple is given it is removed from all contexts

a quad is removed from the given context only

#### Parameters

- **triple** (`Union[Tuple[Node, Node, Node], Tuple[Node, Node, Node, Optional[Graph]]]`) –

**Return type**`NoReturn`**rollback()**

Rollback active transactions

**Return type**`NoReturn`**triples(triple)**

Iterate over all the triples in the entire conjunctive graph

For legacy reasons, this can take the context to query either as a fourth element of the quad, or as the explicit context keyword parameter. The kw param takes precedence.

**Parameters**

**triple** (`Tuple[Optional[Node], Union[Path, Node, None], Optional[Node]]`) –

**Return type**`Generator[Union[Tuple[Node, Node, Node], Tuple[Node, Path, Node]], None, None]`**triples\_choices(triple, context=None)**

Iterate over all the triples in the entire conjunctive graph

**Parameters**

- **triple** (`Union[Tuple[List[Node], Node, Node], Tuple[Node, List[Node], Node], Tuple[Node, Node, List[Node]]]`) –
- **context** (`Optional[Graph]`) –

**Return type**`Generator[Tuple[Node, Node, Node], None, None]`**class rdflib.graph.Seq(graph, subject)**

Bases: `object`

Wrapper around an RDF Seq resource

It implements a container type in Python with the order of the items returned corresponding to the Seq content. It is based on the natural ordering of the predicate names `_1`, `_2`, `_3`, etc, which is the ‘implementation’ of a sequence in RDF terms.

**Parameters**

- **graph** (`Graph`) –
- **subject** (`Node`) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.graph', '__doc__': "Wrapper around an RDF Seq resource\n\n It implements a container type in Python with the order of the items\n returned corresponding to the Seq content. It is based on the natural\n ordering of the predicate names _1, _2, _3, etc, which is the\n 'implementation' of a sequence in RDF terms.\n ", '__init__': <function Seq.__init__>, 'toPython': <function Seq.toPython>, '__iter__': <function Seq.__iter__>, '__len__': <function Seq.__len__>, '__getitem__': <function Seq.__getitem__>, '__dict__': <attribute '__dict__' of 'Seq' objects>, '__weakref__': <attribute '__weakref__' of 'Seq' objects>, '__annotations__': {'_list': 'List[Tuple[int, _ObjectType]]'}})
```

**\_\_getitem\_\_(index)**

Item given by index from the Seq

**Return type***Node***\_\_init\_\_**(*graph*, *subject*)

Parameters:

- **graph:**  
the graph containing the Seq
- **subject:**  
the subject of a Seq. Note that the init does not check whether this is a Seq, this is done in whoever creates this instance!

**Parameters**

- **graph** (*Graph*) –
- **subject** (*Node*) –

**\_\_iter\_\_**()

Generator over the items in the Seq

**Return type***Generator*[*Node*, *None*, *None*]**\_\_len\_\_**()

Length of the Seq

**Return type***int***\_\_module\_\_** = 'rdflib.graph'**\_\_weakref\_\_**

list of weak references to the object (if defined)

**toPython**()**Return type***Seq***exception** rdflib.graph.UnSupportedAggregateOperationBases: *Exception***\_\_init\_\_**()**\_\_module\_\_** = 'rdflib.graph'**\_\_str\_\_**()

Return str(self).

**Return type***str***\_\_weakref\_\_**

list of weak references to the object (if defined)

## rdflib.parser module

Parser plugin interface.

This module defines the parser plugin interface and contains other related parser support code.

The module is mainly useful for those wanting to write a parser that can plugin to rdflib. If you are wanting to invoke a parser you likely want to do so through the Graph class parse method.

**class** rdflib.parser.FileInputSource(*file*)

Bases: *InputSource*

**Parameters**

**file** (*Union*[*BinaryIO*, *TextIO*, *TextIOBase*, *RawIOBase*, *BufferedIOBase*]) –

**\_\_init\_\_**(*file*)

**Parameters**

**file** (*Union*[*BinaryIO*, *TextIO*, *TextIOBase*, *RawIOBase*, *BufferedIOBase*]) –

**\_\_module\_\_** = 'rdflib.parser'

**\_\_repr\_\_**()

Return repr(self).

**Return type**

*str*

**class** rdflib.parser.InputSource(*system\_id=None*)

Bases: *InputSource*

TODO:

**Parameters**

**system\_id** (*Optional*[*str*]) –

**\_\_init\_\_**(*system\_id=None*)

**Parameters**

**system\_id** (*Optional*[*str*]) –

**\_\_module\_\_** = 'rdflib.parser'

**close**()

**Return type**

*None*

**class** rdflib.parser.Parser

Bases: *object*

**\_\_init\_\_**()

**\_\_module\_\_** = 'rdflib.parser'

**\_\_slots\_\_** = ()

**parse**(*source, sink*)

**Parameters**

- **source** (*InputSource*) –



- **sink** (*Graph*) –

**Return type**

*None*

**class** rdflib.parser.**PythonInputSource**(*data, system\_id=None*)

Bases: *InputSource*

Constructs an RDFLib Parser InputSource from a Python data structure, for example, loaded from JSON with `json.load` or `json.loads`:

```
>>> import json
>>> as_string = """{
...   "@context" : {"ex" : "http://example.com/ns#"},
...   "@graph": [{"@type": "ex:item", "@id": "#example"}]
... }"""
>>> as_python = json.loads(as_string)
>>> source = create_input_source(data=as_python)
>>> isinstance(source, PythonInputSource)
True
```

**Parameters**

- **data** (*Any*) –
- **system\_id** (*Optional[str]*) –

**\_\_init\_\_**(*data, system\_id=None*)

**Parameters**

- **data** (*Any*) –
- **system\_id** (*Optional[str]*) –

**\_\_module\_\_** = 'rdflib.parser'

**close**()

**Return type**

*None*

**content\_type**: *Optional[str]*

**getPublicId**()

Returns the public identifier of this InputSource.

**Return type**

*Optional[str]*

**getSystemId**()

Returns the system identifier of this InputSource.

**Return type**

*Optional[str]*

**setPublicId**(*public\_id*)

Sets the public identifier of this InputSource.

**Parameters**

- **public\_id** (*Optional[str]*) –

**Return type**`None`**setSystemId**(*system\_id*)

Sets the system identifier of this InputSource.

**Parameters****system\_id** (`Optional[str]`) –**Return type**`None`**class** `rdflib.parser.StringInputSource`(*value*, *encoding*='utf-8', *system\_id*=None)Bases: `InputSource`

Constructs an RDFLib Parser InputSource from a Python String or Bytes

**Parameters**

- **value** (`Union[str, bytes]`) –
- **encoding** (`str`) –
- **system\_id** (`Optional[str]`) –

`__init__`(*value*, *encoding*='utf-8', *system\_id*=None)**Parameters**

- **value** (`Union[str, bytes]`) –
- **encoding** (`str`) –
- **system\_id** (`Optional[str]`) –

`__module__` = 'rdflib.parser'`content_type`: `Optional[str]`**class** `rdflib.parser.URLInputSource`(*system\_id*=None, *format*=None)Bases: `InputSource`

Constructs an RDFLib Parser InputSource from a URL to read it from the Web.

**Parameters**

- **system\_id** (`Optional[str]`) –
- **format** (`Optional[str]`) –

`__annotations__` = {'links': 'List[str]'}  
`__init__`(*system\_id*=None, *format*=None)**Parameters**

- **system\_id** (`Optional[str]`) –
- **format** (`Optional[str]`) –

`__module__` = 'rdflib.parser'

```

__repr__()
    Return repr(self).

    Return type
        str

get_alternates(type_=None)

    Parameters
        type_ (Optional[str]) –

    Return type
        List[str]

classmethod get_links(response)

    Parameters
        response (addinfourl) –

    Return type
        List[str]

classmethod getallmatchingheaders(message, name)

    Parameters
        message (Message) –

    Return type
        List[str]

links: List[str]

```

## rdflib.paths module

This module implements the SPARQL 1.1 Property path operators, as defined in:

<http://www.w3.org/TR/sparql11-query/#propertypaths>

In SPARQL the syntax is as follows:

Syntax	Matches
iri	An IRI. A path of length one.
^elt	Inverse path (object to subject).
elt1 / elt2	A sequence path of elt1 followed by elt2.
elt1   elt2	A alternative path of elt1 or elt2 (all possibilities are tried).
elt*	A path that connects the subject and object of the path by zero or more matches of elt.
elt+	A path that connects the subject and object of the path by one or more matches of elt.
elt?	A path that connects the subject and object of the path by zero or one matches of elt.
!iri or !(iri <sub>1</sub>   ...  iri <sub>n</sub> )	Negated property set. An IRI which is not one of iri <sub>1</sub> ...iri <sub>n</sub> . !iri is short for !(iri).
!^iri or !(^iri <sub>1</sub>   ...  ^iri <sub>n</sub> )	Negated property set where the excluded matches are based on reversed path. That is, not one of iri <sub>1</sub> ...iri <sub>n</sub> as reverse paths. !^iri is short for !(^iri).
!(iri <sub>1</sub>   ...  iri <sub>j</sub>   ^iri <sub>j+1</sub>   ...   ^iri <sub>n</sub> )	A combination of forward and reverse properties in a negated property set.
(elt)	A group path elt, brackets control precedence.

This module is used internally by the SPARQL engine, but the property paths can also be used to query RDFLib Graphs directly.

Where possible the SPARQL syntax is mapped to Python operators, and property path objects can be constructed from existing URIRefs.

```
>>> from rdflib import Graph, Namespace
>>> from rdflib.namespace import FOAF
```

```
>>> ~FOAF.knows
Path(~http://xmlns.com/foaf/0.1/knows)
```

```
>>> FOAF.knows/FOAF.name
Path(http://xmlns.com/foaf/0.1/knows / http://xmlns.com/foaf/0.1/name)
```

```
>>> FOAF.name|FOAF.givenName
Path(http://xmlns.com/foaf/0.1/name | http://xmlns.com/foaf/0.1/givenName)
```

Modifiers (?, \*, +) are done using \* (the multiplication operator) and the strings '\*', '?', '+', also defined as constants in this file.

```
>>> FOAF.knows*OneOrMore
Path(http://xmlns.com/foaf/0.1/knows+)
```

The path objects can also be used with the normal graph methods.

First some example data:

```
>>> g=Graph()
```

```
>>> g=g.parse(data='''
... @prefix : <ex:> .
...
... :a :p1 :c ; :p2 :f .
... :c :p2 :e ; :p3 :g .
... :g :p3 :h ; :p2 :j .
... :h :p3 :a ; :p2 :g .
...
... :q :px :q .
...
... ''', format='n3')
```

```
>>> e = Namespace('ex:')
```

Graph contains:

```
>>> (e.a, e.p1/e.p2, e.e) in g
True
```

Graph generator functions, triples, subjects, objects, etc. :

```
>>> list(g.objects(e.c, (e.p3*OneOrMore)/e.p2))
[rdflib.term.URIRef('ex:j'), rdflib.term.URIRef('ex:g'),
 rdflib.term.URIRef('ex:f')]
```

A more complete set of tests:

```
>>> list(eval_path(g, (None, e.p1/e.p2, None)))==[(e.a, e.e)]
True
>>> list(eval_path(g, (e.a, e.p1|e.p2, None)))==[(e.a,e.c), (e.a,e.f)]
True
>>> list(eval_path(g, (e.c, ~e.p1, None))) == [ (e.c, e.a) ]
True
>>> list(eval_path(g, (e.a, e.p1*ZeroOrOne, None))) == [(e.a, e.a), (e.a, e.c)]
True
>>> list(eval_path(g, (e.c, e.p3*OneOrMore, None))) == [
...     (e.c, e.g), (e.c, e.h), (e.c, e.a)]
True
>>> list(eval_path(g, (e.c, e.p3*ZeroOrMore, None))) == [(e.c, e.c),
...     (e.c, e.g), (e.c, e.h), (e.c, e.a)]
True
>>> list(eval_path(g, (e.a, -e.p1, None))) == [(e.a, e.f)]
True
>>> list(eval_path(g, (e.a, -(e.p1|e.p2), None))) == []
True
>>> list(eval_path(g, (e.g, ~~e.p2, None))) == [(e.g, e.j)]
True
>>> list(eval_path(g, (e.e, ~(e.p1/e.p2), None))) == [(e.e, e.a)]
True
>>> list(eval_path(g, (e.a, e.p1/e.p3/e.p3, None))) == [(e.a, e.h)]
True
```

```
>>> list(eval_path(g, (e.q, e.px*OneOrMore, None)))
[(rdflib.term.URIRef('ex:q'), rdflib.term.URIRef('ex:q'))]
```

```
>>> list(eval_path(g, (None, e.p1|e.p2, e.c)))
[(rdflib.term.URIRef('ex:a'), rdflib.term.URIRef('ex:c'))]
```

```
>>> list(eval_path(g, (None, ~e.p1, e.a))) == [ (e.c, e.a) ]
True
>>> list(eval_path(g, (None, e.p1*ZeroOrOne, e.c)))
[(rdflib.term.URIRef('ex:c'), rdflib.term.URIRef('ex:c')),
 (rdflib.term.URIRef('ex:a'), rdflib.term.URIRef('ex:c'))]
```

```
>>> list(eval_path(g, (None, e.p3*OneOrMore, e.a)))
[(rdflib.term.URIRef('ex:h'), rdflib.term.URIRef('ex:a')),
 (rdflib.term.URIRef('ex:g'), rdflib.term.URIRef('ex:a')),
 (rdflib.term.URIRef('ex:c'), rdflib.term.URIRef('ex:a'))]
```

```
>>> list(eval_path(g, (None, e.p3*ZeroOrMore, e.a)))
[(rdflib.term.URIRef('ex:a'), rdflib.term.URIRef('ex:a')),
 (rdflib.term.URIRef('ex:h'), rdflib.term.URIRef('ex:a')),
 (rdflib.term.URIRef('ex:g'), rdflib.term.URIRef('ex:a')),
 (rdflib.term.URIRef('ex:c'), rdflib.term.URIRef('ex:a'))]
```

```
>>> list(eval_path(g, (None, -e.p1, e.f))) == [(e.a, e.f)]
True
```

(continues on next page)

(continued from previous page)

```
>>> list(eval_path(g, (None, -(e.p1|e.p2), e.c))) == []
True
>>> list(eval_path(g, (None, ~e.p2, e.j))) == [(e.g, e.j)]
True
>>> list(eval_path(g, (None, ~(e.p1/e.p2), e.a))) == [(e.e, e.a)]
True
>>> list(eval_path(g, (None, e.p1/e.p3/e.p3, e.h))) == [(e.a, e.h)]
True
```

```
>>> list(eval_path(g, (e.q, e.px*OneOrMore, None)))
[(rdflib.term.URIRef('ex:q'), rdflib.term.URIRef('ex:q'))]
```

```
>>> list(eval_path(g, (e.c, (e.p2|e.p3)*ZeroOrMore, e.j)))
[(rdflib.term.URIRef('ex:c'), rdflib.term.URIRef('ex:j'))]
```

No vars specified:

```
>>> sorted(list(eval_path(g, (None, e.p3*OneOrMore, None))))
[(rdflib.term.URIRef('ex:c'), rdflib.term.URIRef('ex:a')),
 (rdflib.term.URIRef('ex:c'), rdflib.term.URIRef('ex:g')),
 (rdflib.term.URIRef('ex:c'), rdflib.term.URIRef('ex:h')),
 (rdflib.term.URIRef('ex:g'), rdflib.term.URIRef('ex:a')),
 (rdflib.term.URIRef('ex:g'), rdflib.term.URIRef('ex:h')),
 (rdflib.term.URIRef('ex:h'), rdflib.term.URIRef('ex:a'))]
```

```
class rdflib.paths.AlternativePath(*args)
```

Bases: *Path*

#### Parameters

**args** (*Union*[*Path*, *URIRef*]) –

**\_\_init\_\_** (\*args)

#### Parameters

**args** (*Union*[*Path*, *URIRef*]) –

**\_\_module\_\_** = 'rdflib.paths'

**\_\_repr\_\_** ()

Return repr(self).

#### Return type

*str*

**eval** (graph, subj=None, obj=None)

#### Parameters

- **graph** (*Graph*) –
- **subj** (*Optional*[*Node*]) –
- **obj** (*Optional*[*Node*]) –

#### Return type

*Generator*[*Tuple*[*Node*, *Node*], None, None]

```

n3(namespace_manager=None)

    Parameters
        namespace_manager (Optional[NamespaceManager]) –

    Return type
        str

class rdflib.paths.InvPath(arg)
    Bases: Path

    Parameters
        arg (Union[Path, URIRef]) –

    __init__(arg)

    Parameters
        arg (Union[Path, URIRef]) –

    __module__ = 'rdflib.paths'

    __repr__()
        Return repr(self).

    Return type
        str

eval(graph, subj=None, obj=None)

    Parameters
        • graph (Graph) –
        • subj (Optional[Node]) –
        • obj (Optional[Node]) –

    Return type
        Generator[Tuple[Node, Node], None, None]

n3(namespace_manager=None)

    Parameters
        namespace_manager (Optional[NamespaceManager]) –

    Return type
        str

class rdflib.paths.MulPath(path, mod)
    Bases: Path

    Parameters
        • path (Union[Path, URIRef]) –
        • mod (Literal['*', '+', '?']) –

    __init__(path, mod)

    Parameters
        • path (Union[Path, URIRef]) –
        • mod (Literal['*', '+', '?']) –

```

```
__module__ = 'rdflib.paths'
```

```
__repr__()
```

Return repr(self).

**Return type**

`str`

```
eval(graph, subj=None, obj=None, first=True)
```

**Parameters**

- **graph** (*Graph*) –
- **subj** (*Optional[Node]*) –
- **obj** (*Optional[Node]*) –
- **first** (*bool*) –

**Return type**

`Generator[Tuple[Node, Node], None, None]`

```
n3(namespace_manager=None)
```

**Parameters**

**namespace\_manager** (*Optional[NamespaceManager]*) –

**Return type**

`str`

```
class rdflib.paths.NegatedPath(arg)
```

Bases: *Path*

**Parameters**

**arg** (*Union[AlternativePath, InvPath, URIRef]*) –

```
__init__(arg)
```

**Parameters**

**arg** (*Union[AlternativePath, InvPath, URIRef]*) –

```
__module__ = 'rdflib.paths'
```

```
__repr__()
```

Return repr(self).

**Return type**

`str`

```
eval(graph, subj=None, obj=None)
```

```
n3(namespace_manager=None)
```

**Parameters**

**namespace\_manager** (*Optional[NamespaceManager]*) –

**Return type**

`str`

```
class rdflib.paths.Path
```

Bases: *object*



```

__annotations__ = {'__invert__': "Callable[['Path'], 'InvPath']", '__mul__':
"Callable[['Path', str], 'MulPath']", '__neg__': "Callable[['Path'],
'NegatedPath']", '__or__': "Callable[['Path', Union['URIRef', 'Path']],
'AlternativePath']", '__truediv__': "Callable[['Path', Union['URIRef', 'Path']],
'SequencePath']"}

__dict__ = mappingproxy({'__module__': 'rdflib.paths', '__annotations__':
{'__or__': "Callable[['Path', Union['URIRef', 'Path']], 'AlternativePath']",
'__invert__': "Callable[['Path'], 'InvPath']", '__neg__': "Callable[['Path'],
'NegatedPath']", '__truediv__': "Callable[['Path', Union['URIRef', 'Path']],
'SequencePath']", '__mul__': "Callable[['Path', str], 'MulPath']"}, 'eval':
<function Path.eval>, '__hash__': <function Path.__hash__>, '__eq__': <function
Path.__eq__>, '__lt__': <function Path.__lt__>, '__dict__': <attribute '__dict__'
of 'Path' objects>, '__weakref__': <attribute '__weakref__' of 'Path' objects>,
'__doc__': None, '__gt__': <function _gt_from_lt>, '__le__': <function
_le_from_lt>, '__ge__': <function _ge_from_lt>, '__invert__': <function inv_path>,
'__neg__': <function neg_path>, '__mul__': <function mul_path>, '__or__':
<function path_alternative>, '__truediv__': <function path_sequence>})

```

**\_\_eq\_\_**(*other*)

Return self==value.

**\_\_ge\_\_**(*other*, *NotImplemented*=*NotImplemented*)

Return a >= b. Computed by @total\_ordering from (not a < b).

**\_\_gt\_\_**(*other*, *NotImplemented*=*NotImplemented*)

Return a > b. Computed by @total\_ordering from (not a < b) and (a != b).

**\_\_hash\_\_**()

Return hash(self).

**\_\_invert\_\_**()

inverse path

**Parameters**

**p** ([Union](#)[[URIRef](#), [Path](#)]) –

**Return type**

[InvPath](#)

**\_\_le\_\_**(*other*, *NotImplemented*=*NotImplemented*)

Return a <= b. Computed by @total\_ordering from (a < b) or (a == b).

**\_\_lt\_\_**(*other*)

Return self<value.

**Parameters**

**other** ([Any](#)) –

**Return type**

[bool](#)

**\_\_module\_\_** = 'rdflib.paths'

**\_\_mul\_\_**(*mul*)

cardinality path

**Parameters**

- **p** (`Union[URIRef, Path]`) –
- **mul** (`Literal['*', '+', '?']`) –

**Return type**  
`MulPath`

**\_\_neg\_\_()**  
negated path

**Parameters**  
**p** (`Union[URIRef, AlternativePath, InvPath]`) –

**Return type**  
`NegatedPath`

**\_\_or\_\_(other)**  
alternative path

**Parameters**

- **self** (`Union[URIRef, Path]`) –
- **other** (`Union[URIRef, Path]`) –

**\_\_truediv\_\_(other)**  
sequence path

**Parameters**

- **self** (`Union[URIRef, Path]`) –
- **other** (`Union[URIRef, Path]`) –

**\_\_weakref\_\_**  
list of weak references to the object (if defined)

**eval(graph, subj=None, obj=None)**

**Parameters**

- **graph** (`Graph`) –
- **subj** (`Optional[Node]`) –
- **obj** (`Optional[Node]`) –

**Return type**  
`Iterator[Tuple[Node, Node]]`

**class rdflib.paths.PathList(iterable=(), /)**  
Bases: `list`

**\_\_dict\_\_** = `mappingproxy({'__module__': 'rdflib.paths', '__dict__': <attribute '__dict__' of 'PathList' objects>, '__weakref__': <attribute '__weakref__' of 'PathList' objects>, '__doc__': None, '__annotations__': {}})`

**\_\_module\_\_** = `'rdflib.paths'`

**\_\_weakref\_\_**  
list of weak references to the object (if defined)

```
class rdflib.paths.SequencePath(*args)
```

Bases: *Path*

**Parameters**

**args** (Union[*Path*, *URIRef*]) –

```
__init__(*args)
```

**Parameters**

**args** (Union[*Path*, *URIRef*]) –

```
__module__ = 'rdflib.paths'
```

```
__repr__()
```

Return repr(self).

**Return type**

*str*

```
eval(graph, subj=None, obj=None)
```

**Parameters**

- **graph** (*Graph*) –
- **subj** (Optional[*Node*]) –
- **obj** (Optional[*Node*]) –

**Return type**

Generator[Tuple[*Node*, *Node*], None, None]

```
n3(namespace_manager=None)
```

**Parameters**

**namespace\_manager** (Optional[*NamespaceManager*]) –

**Return type**

*str*

```
rdflib.paths.evalPath(graph, t)
```

**Parameters**

- **graph** (*Graph*) –
- **t** (Tuple[Optional[*Node*], Union[None, *Path*, *Node*], Optional[*Node*]]) –

**Return type**

Iterator[Tuple[*Node*, *Node*]]

```
rdflib.paths.eval_path(graph, t)
```

**Parameters**

- **graph** (*Graph*) –
- **t** (Tuple[Optional[*Node*], Union[None, *Path*, *Node*], Optional[*Node*]]) –

**Return type**

Iterator[Tuple[*Node*, *Node*]]

`rdflib.paths.inv_path(p)`

inverse path

**Parameters**

`p` (`Union[URIRef, Path]`) –

**Return type**

`InvPath`

`rdflib.paths.mul_path(p, mul)`

cardinality path

**Parameters**

- `p` (`Union[URIRef, Path]`) –
- `mul` (`Literal['*', '+', '?']`) –

**Return type**

`MulPath`

`rdflib.paths.neg_path(p)`

negated path

**Parameters**

`p` (`Union[URIRef, AlternativePath, InvPath]`) –

**Return type**

`NegatedPath`

`rdflib.paths.path_alternative(self, other)`

alternative path

**Parameters**

- `self` (`Union[URIRef, Path]`) –
- `other` (`Union[URIRef, Path]`) –

`rdflib.paths.path_sequence(self, other)`

sequence path

**Parameters**

- `self` (`Union[URIRef, Path]`) –
- `other` (`Union[URIRef, Path]`) –

## rdflib.plugin module

Plugin support for rdf.

There are a number of plugin points for rdf: parser, serializer, store, query processor, and query result. Plugins can be registered either through `setuptools` entry\_points or by calling `rdf.plugin.register` directly.

If you have a package that uses a `setuptools` based `setup.py` you can add the following to your setup:

```
entry_points = {
    'rdf.plugins.parser': [
        'nt = rdf.plugins.parsers.ntriples:NTParser',
    ],
}
```

(continues on next page)

(continued from previous page)

```

'rdf.plugins.serializer': [
    'nt =      rdf.plugins.serializers.NTSerializer:NTSerializer',
],
}

```

See the [setuptools](#) dynamic discovery of services and plugins for more information.

**class** rdflib.plugin.PKGPlugin(name, kind, ep)

Bases: [Plugin](#)[[PluginT](#)]

#### Parameters

- **name** ([str](#)) –
- **kind** ([Type](#)[[TypeVar](#)([PluginT](#))]) –
- **ep** ([EntryPoint](#)) –

**\_\_init\_\_**(name, kind, ep)

#### Parameters

- **name** ([str](#)) –
- **kind** ([Type](#)[[TypeVar](#)([PluginT](#))]) –
- **ep** ([EntryPoint](#)) –

**\_\_module\_\_** = 'rdflib.plugin'

**\_\_orig\_bases\_\_** = (rdflib.plugin.Plugin[~PluginT],)

**\_\_parameters\_\_** = (~PluginT,)

**getClass**()

#### Return type

[Type](#)[[TypeVar](#)([PluginT](#))]

**class** rdflib.plugin.Plugin(name, kind, module\_path, class\_name)

Bases: [Generic](#)[[PluginT](#)]

#### Parameters

- **name** ([str](#)) –
- **kind** ([Type](#)[[TypeVar](#)([PluginT](#))]) –
- **module\_path** ([str](#)) –
- **class\_name** ([str](#)) –

```

__dict__ = mappingproxy({'__module__': 'rdflib.plugin', '__init__': <function
Plugin.__init__>, 'getClass': <function Plugin.getClass>, '__orig_bases__':
(typing.Generic[~PluginT],), '__dict__': <attribute '__dict__' of 'Plugin'
objects>, '__weakref__': <attribute '__weakref__' of 'Plugin' objects>, '__doc__':
None, '__parameters__': (~PluginT,), '__annotations__': {'_class':
'Optional[Type[PluginT]]'}})

```

`__init__(name, kind, module_path, class_name)`

**Parameters**

- **name** (`str`) –
- **kind** (`Type[TypeVar(PluginT)]`) –
- **module\_path** (`str`) –
- **class\_name** (`str`) –

`__module__` = 'rdflib.plugin'

`__orig_bases__` = (`typing.Generic[~PluginT]`,)

`__parameters__` = (`~PluginT`,)

`__weakref__`

list of weak references to the object (if defined)

`getClass()`

**Return type**

`Type[TypeVar(PluginT)]`

**exception** `rdflib.plugin.PluginException(msg=None)`

Bases: `Error`

**Parameters**

**msg** (`Optional[str]`) –

`__module__` = 'rdflib.plugin'

`rdflib.plugin.PluginT`

A generic type variable for plugins

alias of `TypeVar('PluginT')`

`rdflib.plugin.get(name, kind)`

Return the class for the specified (name, kind). Raises a `PluginException` if unable to do so.

**Parameters**

- **name** (`str`) –
- **kind** (`Type[TypeVar(PluginT)]`) –

**Return type**

`Type[TypeVar(PluginT)]`

`rdflib.plugin.plugins(name=None, kind=None)`

A generator of the plugins.

Pass in name and kind to filter... else leave None to match all.

**Parameters**

- **name** (`Optional[str]`) –
- **kind** (`Optional[Type[TypeVar(PluginT)]]`) –

**Return type**

`Iterator[Plugin[TypeVar(PluginT)]]`

`rdflib.plugin.register(name, kind, module_path, class_name)`

Register the plugin for (name, kind). The module\_path and class\_name should be the path to a plugin class.

**Parameters**

- **name** (`str`) –
- **kind** (`Type[Any]`) –

## rdflib.query module

`class rdflib.query.EncodeOnlyUnicode(stream)`

Bases: `object`

This is a crappy work-around for <http://bugs.python.org/issue11649>

**Parameters**

**stream** (`BinaryIO`) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.query', '__doc__': '\n This is a
crappy work-around for\n http://bugs.python.org/issue11649\n\n\n ', '__init__':
<function EncodeOnlyUnicode.__init__>, 'write': <function EncodeOnlyUnicode.write>,
'__getattr__': <function EncodeOnlyUnicode.__getattr__>, '__dict__': <attribute
'__dict__' of 'EncodeOnlyUnicode' objects>, '__weakref__': <attribute '__weakref__'
of 'EncodeOnlyUnicode' objects>, '__annotations__': {}})
```

`__getattr__(name)`

**Parameters**

**name** (`str`) –

**Return type**

`Any`

`__init__(stream)`

**Parameters**

**stream** (`BinaryIO`) –

`__module__` = 'rdflib.query'

`__weakref__`

list of weak references to the object (if defined)

`write(arg)`

`class rdflib.query.Processor(graph)`

Bases: `object`

Query plugin interface.

This module is useful for those wanting to write a query processor that can plugin to rdf. If you are wanting to execute a query you likely want to do so through the Graph class query method.

**Parameters**

**graph** (`Graph`) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.query', '__doc__': '\n Query plugin
interface.\n\n This module is useful for those wanting to write a query processor\n
that can plugin to rdf. If you are wanting to execute a query you\n likely want to
do so through the Graph class query method.\n\n ', '__init__': <function
Processor.__init__>, 'query': <function Processor.query>, '__dict__': <attribute
'__dict__' of 'Processor' objects>, '__weakref__': <attribute '__weakref__' of
'Processor' objects>, '__annotations__': {}})
```

```
__init__(graph)
```

**Parameters**

**graph** (*Graph*) –

```
__module__ = 'rdflib.query'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
query(strOrQuery, initBindings={}, initNs={}, DEBUG=False)
```

**Parameters**

- **strOrQuery** (*Union[str, Query]*) –
- **initBindings** (*Mapping[str, Identifier]*) –
- **initNs** (*Mapping[str, Any]*) –
- **DEBUG** (*bool*) –

**Return type**

*Mapping[str, Any]*

```
class rdflib.query.Result(type_)
```

Bases: *object*

A common class for representing query result.

There is a bit of magic here that makes this appear like different Python objects, depending on the type of result.

If the type is “SELECT”, iterating will yield lists of *ResultRow* objects

If the type is “ASK”, iterating will yield a single bool (or bool(result) will return the same bool)

If the type is “CONSTRUCT” or “DESCRIBE” iterating will yield the triples.

len(result) also works.

**Parameters**

**type\_** (*str*) –

```
__bool__()
```

**Return type**

*bool*



```
__dict__ = mappingproxy({'__module__': 'rdflib.query', '__doc__': '\n A common
class for representing query result.\n\n There is a bit of magic here that makes
this appear like different\n Python objects, depending on the type of result.\n\n If
the type is "SELECT", iterating will yield lists of ResultRow objects\n\n If the
type is "ASK", iterating will yield a single bool (or\n bool(result) will return the
same bool)\n\n If the type is "CONSTRUCT" or "DESCRIBE" iterating will yield the\n
triples.\n\n len(result) also works.\n\n ', '__init__': <function Result.__init__>,
'bindings': <property object>, 'parse': <staticmethod object>, 'serialize':
<function Result.serialize>, '__len__': <function Result.__len__>, '__bool__':
<function Result.__bool__>, '__iter__': <function Result.__iter__>, '__getattr__':
<function Result.__getattr__>, '__eq__': <function Result.__eq__>, '__dict__':
<attribute '__dict__' of 'Result' objects>, '__weakref__': <attribute '__weakref__'
of 'Result' objects>, '__hash__': None, '__annotations__': {'vars':
"Optional[List['Variable']]", '_bindings': "MutableSequence[Mapping['Variable',
'Identifier']]", '_genbindings': "Optional[Iterator[Mapping['Variable',
'Identifier']]", 'askAnswer': 'Optional[bool]', 'graph': "Optional['Graph']"}})
```

`__eq__(other)`

Return self==value.

**Parameters**

**other** (*Any*) –

**Return type**

*bool*

`__getattr__(name)`

**Parameters**

**name** (*str*) –

**Return type**

*Any*

`__hash__ = None`

`__init__(type_)`

**Parameters**

**type\_** (*str*) –

`__iter__()`

**Return type**

*Iterator[Union[Tuple[Node, Node, Node], bool, ResultRow]]*

`__len__()`

**Return type**

*int*

`__module__ = 'rdflib.query'`

`__weakref__`

list of weak references to the object (if defined)

**property bindings:** *MutableSequence[Mapping[Variable, Identifier]]*

a list of variable bindings as dicts

```
static parse(source=None, format=None, content_type=None, **kwargs)
```

**Parameters**

- **source** (Optional[IO]) –
- **format** (Optional[str]) –
- **content\_type** (Optional[str]) –
- **kwargs** (Any) –

**Return type**

*Result*

```
serialize(destination=None, encoding='utf-8', format='xml', **args)
```

Serialize the query result.

The `format` argument determines the Serializer class to use.

- `csv`: *CSVResultSerializer*
- `json`: *JSONResultSerializer*
- `txt`: *TXTResultSerializer*
- `xml`: *XMLResultSerializer*

**Parameters**

- **destination** (Union[str, IO, None]) – Path of file output or BufferedIOBase object to write the output to.
- **encoding** (str) – Encoding of output.
- **format** (str) – One of ['csv', 'json', 'txt', 'xml']
- **args** (Any) –

**Return type**

Optional[bytes]

**Returns**

bytes

```
vars: Optional[List[Variable]]
```

variables contained in the result.

```
exception rdflib.query.ResultException
```

Bases: *Exception*

```
__module__ = 'rdflib.query'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
class rdflib.query.ResultParser
```

Bases: *object*

```
__dict__ = mappingproxy({'__module__': 'rdflib.query', '__init__': <function  
ResultParser.__init__>, 'parse': <function ResultParser.parse>, '__dict__':  
<attribute '__dict__' of 'ResultParser' objects>, '__weakref__': <attribute  
'__weakref__' of 'ResultParser' objects>, '__doc__': None, '__annotations__': {}})
```

```
__init__()
```

```
__module__ = 'rdflib.query'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
parse(source, **kwargs)
```

return a Result object

#### Parameters

- **source** (*IO*) –
- **kwargs** (*Any*) –

#### Return type

*Result*

```
class rdflib.query.ResultRow(values: Mapping[Variable, Identifier], labels: List[Variable])
```

Bases: Tuple[Identifier, ...]

a single result row allows accessing bindings as attributes or with []

```
>>> from rdflib import URIRef, Variable
>>> rr=ResultRow({ Variable('a'): URIRef('urn:cake') }, [Variable('a')])
```

```
>>> rr[0]
rdflib.term.URIRef(u'urn:cake')
>>> rr[1]
Traceback (most recent call last):
...
IndexError: tuple index out of range
```

```
>>> rr.a
rdflib.term.URIRef(u'urn:cake')
>>> rr.b
Traceback (most recent call last):
...
AttributeError: b
```

```
>>> rr['a']
rdflib.term.URIRef(u'urn:cake')
>>> rr['b']
Traceback (most recent call last):
...
KeyError: 'b'
```

```
>>> rr[Variable('a')]
rdflib.term.URIRef(u'urn:cake')
```

New in version 4.0.

```
__annotations__ = {'labels': 'Mapping[str, int]'}
```

```
__dict__ = mappingproxy({'__module__': 'rdflib.query', '__annotations__':
{'labels': 'Mapping[str, int]'}, '__doc__': "\n a single result row\n allows
accessing bindings as attributes or with []\n\n >>> from rdflib import URIRef,
Variable\n >>> rr=ResultRow({ Variable('a'):  URIRef('urn:cake') },
[Variable('a')])\n\n >>> rr[0]\n rdflib.term.URIRef(u'urn:cake')\n\n >>> rr[1]\n
Traceback (most recent call last):\n ...\n IndexError:  tuple index out of range\n\n
>>> rr.a\n rdflib.term.URIRef(u'urn:cake')\n\n >>> rr.b\n Traceback (most recent call
last):\n ...\n AttributeError:  b\n\n >>> rr['a']\n
rdflib.term.URIRef(u'urn:cake')\n\n >>> rr['b']\n Traceback (most recent call last):\n
...\n KeyError:  'b'\n\n >>> rr[Variable('a')]\n rdflib.term.URIRef(u'urn:cake')\n\n
.. versionadded:: 4.0\n\n ", '__new__': <staticmethod object>, '__getattr__':
<function ResultRow.__getattr__>, '__getitem__': <function ResultRow.__getitem__>,
'get': <function ResultRow.get>, 'asdict': <function ResultRow.asdict>,
'__orig_bases__': (typing.Tuple[ForwardRef('Identifier'), ...],), '__dict__':
<attribute '__dict__' of 'ResultRow' objects>, '__parameters__': ()})
```

**\_\_getattr\_\_**(*name*)

**Parameters**

**name** (*str*) –

**Return type**

*Identifier*

**\_\_getitem\_\_**(*name*)

Return self[key].

**Parameters**

**name** (*Union[str, int, Any]*) –

**Return type**

*Identifier*

**\_\_module\_\_** = 'rdflib.query'

**static** **\_\_new\_\_**(*cls, values, labels*)

**Parameters**

• **values** (*Mapping[Variable, Identifier]*) –

• **labels** (*List[Variable]*) –

**\_\_orig\_bases\_\_** = (typing.Tuple[ForwardRef('Identifier'), ...],)

**\_\_parameters\_\_** = ()

**asdict**()

**Return type**

*Dict[str, Identifier]*

**get**(*name, default=None*)

**Parameters**

• **name** (*str*) –

• **default** (*Optional[Identifier]*) –

**Return type**

*Optional[Identifier]*

labels: Mapping[str, int]

class rdflib.query.ResultSerializer(result)

Bases: object

#### Parameters

result (*Result*) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.query', '__init__': <function
ResultSerializer.__init__>, 'serialize': <function ResultSerializer.serialize>,
'__dict__': <attribute '__dict__' of 'ResultSerializer' objects>, '__weakref__':
<attribute '__weakref__' of 'ResultSerializer' objects>, '__doc__': None,
'__annotations__': {}})
```

\_\_init\_\_(result)

#### Parameters

result (*Result*) –

\_\_module\_\_ = 'rdflib.query'

\_\_weakref\_\_

list of weak references to the object (if defined)

serialize(stream, encoding='utf-8', \*\*kwargs)

return a string properly serialized

#### Parameters

- stream (*IO*) –
- encoding (*str*) –
- kwargs (*Any*) –

#### Return type

None

class rdflib.query.UpdateProcessor(graph)

Bases: object

Update plugin interface.

This module is useful for those wanting to write an update processor that can plugin to rdflib. If you are wanting to execute an update statement you likely want to do so through the Graph class update method.

New in version 4.0.

#### Parameters

graph (*Graph*) –

```
__dict__ = mappingproxy({'__module__': 'rdflib.query', '__doc__': '\n Update
plugin interface.\n\n This module is useful for those wanting to write an update\n
processor that can plugin to rdflib. If you are wanting to execute\n an update
statement you likely want to do so through the Graph\n class update method.\n\n ..
versionadded:: 4.0\n\n ', '__init__': <function UpdateProcessor.__init__>,
'update': <function UpdateProcessor.update>, '__dict__': <attribute '__dict__' of
'UpdateProcessor' objects>, '__weakref__': <attribute '__weakref__' of
'UpdateProcessor' objects>, '__annotations__': {}})
```

`__init__(graph)`

**Parameters**

**graph** (*Graph*) –

`__module__ = 'rdflib.query'`

`__weakref__`

list of weak references to the object (if defined)

`update(strOrQuery, initBindings={}, initNs={})`

**Parameters**

- **strOrQuery** (*Union[str, Update]*) –
- **initBindings** (*Mapping[str, Identifier]*) –
- **initNs** (*Mapping[str, Any]*) –

**Return type**

*None*

## rdflib.resource module

The *Resource* class wraps a *Graph* and a resource reference (i.e. a *rdflib.term.URIRef* or *rdflib.term.BNode*) to support a resource-oriented way of working with a graph.

It contains methods directly corresponding to those methods of the Graph interface that relate to reading and writing data. The difference is that a Resource also binds a resource identifier, making it possible to work without tracking both the graph and a current subject. This makes for a “resource oriented” style, as compared to the triple orientation of the Graph API.

Resulting generators are also wrapped so that any resource reference values (*rdflib.term.URIRef* and *rdflib.term.BNode*) are in turn wrapped as Resources. (Note that this behaviour differs from the corresponding methods in *Graph*, where no such conversion takes place.)

## Basic Usage Scenario

Start by importing things we need and define some namespaces:

```
>>> from rdflib import *
>>> FOAF = Namespace("http://xmlns.com/foaf/0.1/")
>>> CV = Namespace("http://purl.org/captsolo/resume-rdf/0.2/cv#")
```

Load some RDF data:

```
>>> graph = Graph().parse(format='n3', data='''
... @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
... @prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
... @prefix foaf: <http://xmlns.com/foaf/0.1/> .
... @prefix cv: <http://purl.org/captsolo/resume-rdf/0.2/cv#> .
...
... @base <http://example.org/> .
...
... </person/some1#self> a foaf:Person;
```

(continues on next page)

(continued from previous page)

```

...   rdfs:comment "Just a Python & RDF hacker."@en;
...   foaf:depiction </images/person/some1.jpg>;
...   foaf:homepage <http://example.net/>;
...   foaf:name "Some Body" .
...
... </images/person/some1.jpg> a foaf:Image;
...   rdfs:label "some 1"@en;
...   rdfs:comment "Just an image"@en;
...   foaf:thumbnail </images/person/some1-thumb.jpg> .
...
... </images/person/some1-thumb.jpg> a foaf:Image .
...
... [] a cv:CV;
...   cv:aboutPerson </person/some1#self>;
...   cv:hasWorkHistory [ cv:employedIn </#company>;
...                       cv:startDate "2009-09-04"^^xsd:date ] .
... '''

```

Create a Resource:

```

>>> person = Resource(
...     graph, URIRef("http://example.org/person/some1#self"))

```

Retrieve some basic facts:

```

>>> person.identifier
rdflib.term.URIRef(u'http://example.org/person/some1#self')

>>> person.value(FOAF.name)
rdflib.term.Literal(u'Some Body')

>>> person.value(RDFS.comment)
rdflib.term.Literal(u'Just a Python & RDF hacker.', lang=u'en')

```

Resources can be sliced (like graphs, but the subject is fixed):

```

>>> for name in person[FOAF.name]:
...     print(name)
Some Body
>>> person[FOAF.name : Literal("Some Body")]
True

```

Resources as unicode are represented by their identifiers as unicode:

```

>>> %(unicode)s(person)
u'Resource(http://example.org/person/some1#self)'

```

Resource references are also Resources, so you can easily get e.g. a qname for the type of a resource, like:

```

>>> person.value(RDF.type).qname()
u'foaf:Person'

```

Or for the predicates of a resource:

```
>>> sorted(
...     p.qname() for p in person.predicates()
... )
[u'foaf:depiction', u'foaf:homepage',
 u'foaf:name', u'rdf:type', u'rdfs:comment']
```

Follow relations and get more data from their Resources as well:

```
>>> for pic in person.objects(FOAF.depiction):
...     print(pic.identifier)
...     print(pic.value(RDF.type).qname())
...     print(pic.value(FOAF.thumbnail).identifier)
http://example.org/images/person/some1.jpg
foaf:Image
http://example.org/images/person/some1-thumb.jpg

>>> for cv in person.subjects(CV.aboutPerson):
...     work = list(cv.objects(CV.hasWorkHistory))[0]
...     print(work.value(CV.employedIn).identifier)
...     print(work.value(CV.startDate))
http://example.org/#company
2009-09-04
```

It's just as easy to work with the predicates of a resource:

```
>>> for s, p in person.subject_predicates():
...     print(s.value(RDF.type).qname())
...     print(p.qname())
...     for s, o in p.subject_objects():
...         print(s.value(RDF.type).qname())
...         print(o.value(RDF.type).qname())
cv:CV
cv:aboutPerson
cv:CV
foaf:Person
```

This is useful for e.g. inspection:

```
>>> thumb_ref = URIRef("http://example.org/images/person/some1-thumb.jpg")
>>> thumb = Resource(graph, thumb_ref)
>>> for p, o in thumb.predicate_objects():
...     print(p.qname())
...     print(o.qname())
rdf:type
foaf:Image
```



## Schema Example

With this artificial schema data:

```
>>> graph = Graph().parse(format='n3', data='''
... @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
... @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
... @prefix owl: <http://www.w3.org/2002/07/owl#> .
... @prefix v: <http://example.org/def/v#> .
...
... v:Artifact a owl:Class .
...
... v:Document a owl:Class;
...     rdfs:subClassOf v:Artifact .
...
... v:Paper a owl:Class;
...     rdfs:subClassOf v:Document .
...
... v:Choice owl:oneOf (v:One v:Other) .
...
... v:Stuff a rdf:Seq; rdf:_1 v:One; rdf:_2 v:Other .
...
... ''')
```

From this class:

```
>>> artifact = Resource(graph, URIRef("http://example.org/def/v#Artifact"))
```

we can get at subclasses:

```
>>> subclasses = list(artifact.transitive_subjects(RDFS.subClassOf))
>>> [c.qname() for c in subclasses]
[u'v:Artifact', u'v:Document', u'v:Paper']
```

and superclasses from the last subclass:

```
>>> [c.qname() for c in subclasses[-1].transitive_objects(RDFS.subClassOf)]
[u'v:Paper', u'v:Document', u'v:Artifact']
```

Get items from the Choice:

```
>>> choice = Resource(graph, URIRef("http://example.org/def/v#Choice"))
>>> [it.qname() for it in choice.value(OWL.oneOf).items()]
[u'v:One', u'v:Other']
```

**On add, other resources are auto-unboxed:**

```
>>> paper = Resource(graph, URIRef("http://example.org/def/v#Paper"))
>>> paper.add(RDFS.subClassOf, artifact)
>>> artifact in paper.objects(RDFS.subClassOf) # checks Resource instance
True
>>> (paper._identifier, RDFS.subClassOf, artifact._identifier) in graph
True
```

## Technical Details

Comparison is based on graph and identifier:

```
>>> g1 = Graph()
>>> t1 = Resource(g1, URIRef("http://example.org/thing"))
>>> t2 = Resource(g1, URIRef("http://example.org/thing"))
>>> t3 = Resource(g1, URIRef("http://example.org/other"))
>>> t4 = Resource(Graph(), URIRef("http://example.org/other"))

>>> t1 is t2
False

>>> t1 == t2
True
>>> t1 != t2
False

>>> t1 == t3
False
>>> t1 != t3
True

>>> t3 != t4
True

>>> t3 < t1 and t1 > t3
True
>>> t1 >= t1 and t1 >= t3
True
>>> t1 <= t1 and t3 <= t1
True

>>> t1 < t1 or t1 < t3 or t3 > t1 or t3 > t3
False
```

Hash is computed from graph and identifier:

```
>>> g1 = Graph()
>>> t1 = Resource(g1, URIRef("http://example.org/thing"))

>>> hash(t1) == hash(Resource(g1, URIRef("http://example.org/thing")))
True

>>> hash(t1) == hash(Resource(Graph(), t1.identifier))
False
>>> hash(t1) == hash(Resource(Graph(), URIRef("http://example.org/thing")))
False
```

The Resource class is suitable as a base class for mapper toolkits. For example, consider this utility for accessing RDF properties via qname-like attributes:

```
>>> class Item(Resource):
```

(continues on next page)

(continued from previous page)

```

...
...     def __getattr__(self, p):
...         return list(self.objects(self._to_ref(*p.split('_', 1))))
...
...     def _to_ref(self, pfx, name):
...         return URIRef(self._graph.store.namespace(pfx) + name)

```

It works as follows:

```

>>> graph = Graph().parse(format='n3', data='''
... @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
... @prefix foaf: <http://xmlns.com/foaf/0.1/> .
...
... @base <http://example.org/> .
... </person/some1#self>
...     foaf:name "Some Body";
...     foaf:depiction </images/person/some1.jpg> .
... </images/person/some1.jpg> rdfs:comment "Just an image"@en .
... ''')

>>> person = Item(graph, URIRef("http://example.org/person/some1#self"))

>>> print(person.foaf_name[0])
Some Body

```

The mechanism for wrapping references as resources cooperates with subclasses. Therefore, accessing referenced resources automatically creates new `Item` objects:

```

>>> isinstance(person.foaf_depiction[0], Item)
True

>>> print(person.foaf_depiction[0].rdfs_comment[0])
Just an image

```

```

class rdflib.resource.Resource(graph, subject)
    Bases: object

```

```
__dict__ = mappingproxy({'__module__': 'rdflib.resource', '__init__': <function
Resource.__init__>, 'graph': <property object>, 'identifier': <property object>,
'__hash__': <function Resource.__hash__>, '__eq__': <function Resource.__eq__>,
'__ne__': <function Resource.__ne__>, '__lt__': <function Resource.__lt__>,
'__gt__': <function Resource.__gt__>, '__le__': <function Resource.__le__>,
'__ge__': <function Resource.__ge__>, '__unicode__': <function
Resource.__unicode__>, 'add': <function Resource.add>, 'remove': <function
Resource.remove>, 'set': <function Resource.set>, 'subjects': <function
Resource.subjects>, 'predicates': <function Resource.predicates>, 'objects':
<function Resource.objects>, 'subject_predicates': <function
Resource.subject_predicates>, 'subject_objects': <function
Resource.subject_objects>, 'predicate_objects': <function
Resource.predicate_objects>, 'value': <function Resource.value>, 'items':
<function Resource.items>, 'transitive_objects': <function
Resource.transitive_objects>, 'transitive_subjects': <function
Resource.transitive_subjects>, 'qname': <function Resource.qname>,
'_resource_pairs': <function Resource._resource_pairs>, '_resource_triples':
<function Resource._resource_triples>, '_resources': <function
Resource._resources>, '_cast': <function Resource._cast>, '__iter__': <function
Resource.__iter__>, '__getitem__': <function Resource.__getitem__>, '__setitem__':
<function Resource.__setitem__>, '_new': <function Resource._new>, '__str__':
<function Resource.__str__>, '__repr__': <function Resource.__repr__>, '__dict__':
<attribute '__dict__' of 'Resource' objects>, '__weakref__': <attribute
'__weakref__' of 'Resource' objects>, '__doc__': None, '__annotations__': {}})
```

**\_\_eq\_\_**(*other*)

Return self==value.

**\_\_ge\_\_**(*other*)

Return self>=value.

**\_\_getitem\_\_**(*item*)

**\_\_gt\_\_**(*other*)

Return self>value.

**\_\_hash\_\_**()

Return hash(self).

**\_\_init\_\_**(*graph, subject*)

**\_\_iter\_\_**()

**\_\_le\_\_**(*other*)

Return self<=value.

**\_\_lt\_\_**(*other*)

Return self<value.

**\_\_module\_\_** = 'rdflib.resource'

**\_\_ne\_\_**(*other*)

Return self!=value.

**\_\_repr\_\_**()

Return repr(self).

```

__setitem__(item, value)

__str__()
    Return str(self).

__unicode__()

__weakref__
    list of weak references to the object (if defined)

add(p, o)

property graph

property identifier

items()

objects(predicate=None)

predicate_objects()

predicates(o=None)

qname()

remove(p, o=None)

set(p, o)

subject_objects()

subject_predicates()

subjects(predicate=None)

transitive_objects(predicate, remember=None)

transitive_subjects(predicate, remember=None)

value(p=rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#value'), o=None, default=None,
      any=True)

```

## rdflib.serializer module

```
class rdflib.serializer.Serializer(store)
```

Bases: `object`

### Parameters

`store` (`Graph`) –

```

__dict__ = mappingproxy({'__module__': 'rdflib.serializer', '__init__': <function
Serializer.__init__>, 'serialize': <function Serializer.serialize>, 'relativize':
<function Serializer.relativize>, '__dict__': <attribute '__dict__' of 'Serializer'
objects>, '__weakref__': <attribute '__weakref__' of 'Serializer' objects>,
'__doc__': None, '__annotations__': {'store': "'Graph'", 'encoding': 'str',
'base': 'Optional[str]'}})

```

`__init__(store)`

Parameters

`store` (*Graph*) –

`__module__` = 'rdflib.serializer'

`__weakref__`

list of weak references to the object (if defined)

`relativize(uri)`

Parameters

`uri` (*TypeVar*(*\_StrT*, bound= *str*)) –

Return type

*Union*[*TypeVar*(*\_StrT*, bound= *str*), *URIRef*]

`serialize(stream, base=None, encoding=None, **args)`

Abstract method

Parameters

- `stream` (*IO*[*bytes*]) –
- `base` (*Optional*[*str*]) –
- `encoding` (*Optional*[*str*]) –
- `args` (*Any*) –

Return type

*None*

## rdflib.store module

`class rdflib.store.NodePickler`

Bases: *object*

```
__dict__ = mappingproxy({'__module__': 'rdflib.store', '__init__': <function
NodePickler.__init__>, '_get_ids': <function NodePickler._get_ids>, 'register':
<function NodePickler.register>, 'loads': <function NodePickler.loads>, 'dumps':
<function NodePickler.dumps>, '__getstate__': <function NodePickler.__getstate__>,
'__setstate__': <function NodePickler.__setstate__>, '__dict__': <attribute
'__dict__' of 'NodePickler' objects>, '__weakref__': <attribute '__weakref__' of
'NodePickler' objects>, '__doc__': None, '__annotations__': {'_objects':
'Dict[str, Any]', '_ids': 'Dict[Any, str]'}})
```

`__getstate__()`

Return type

*Mapping*[*str*, *Any*]

`__init__()`

`__module__` = 'rdflib.store'

**\_\_setstate\_\_**(*state*)

**Parameters**

**state** (`Mapping[str, Any]`) –

**Return type**

`None`

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**dumps**(*obj*, *protocol=None*, *bin=None*)

**Parameters**

- **obj** (`Node`) –
- **protocol** (`Optional[Any]`) –
- **bin** (`Optional[Any]`) –

**loads**(*s*)

**Parameters**

**s** (`bytes`) –

**Return type**

`Node`

**register**(*object*, *id*)

**Parameters**

- **object** (`Any`) –
- **id** (`str`) –

**Return type**

`None`

**class** `rdflib.store.Store`(*configuration=None*, *identifier=None*)

Bases: `object`

**Parameters**

- **configuration** (`Optional[str]`) –
- **identifier** (`Optional[Identifier]`) –

**\_\_annotations\_\_** = {'context\_aware': 'bool', 'formula\_aware': 'bool',  
'graph\_aware': 'bool', 'transaction\_aware': 'bool'}

```
__dict__ = mappingproxy({'__module__': 'rdflib.store', '__annotations__':
{'context_aware': 'bool', 'formula_aware': 'bool', 'transaction_aware': 'bool',
'graph_aware': 'bool', '__node_pickler': 'Optional[NodePickler]'},
'context_aware': False, 'formula_aware': False, 'transaction_aware': False,
'graph_aware': False, '__init__': <function Store.__init__>, 'node_pickler':
<property object>, 'create': <function Store.create>, 'open': <function
Store.open>, 'close': <function Store.close>, 'destroy': <function Store.destroy>,
'gc': <function Store.gc>, 'add': <function Store.add>, 'addN': <function
Store.addN>, 'remove': <function Store.remove>, 'triples_choices': <function
Store.triples_choices>, 'triples': <function Store.triples>, '__len__': <function
Store.__len__>, 'contexts': <function Store.contexts>, 'query': <function
Store.query>, 'update': <function Store.update>, 'bind': <function Store.bind>,
'prefix': <function Store.prefix>, 'namespace': <function Store.namespace>,
'namespaces': <function Store.namespaces>, 'commit': <function Store.commit>,
'rollback': <function Store.rollback>, 'add_graph': <function Store.add_graph>,
'remove_graph': <function Store.remove_graph>, '__dict__': <attribute '__dict__'
of 'Store' objects>, '__weakref__': <attribute '__weakref__' of 'Store' objects>,
'__doc__': None})
```

**\_\_init\_\_**(*configuration=None, identifier=None*)

identifier: URIRef of the Store. Defaults to CWD configuration: string containing information open can use to connect to datastore.

#### Parameters

- **configuration** (*Optional[str]*) –
- **identifier** (*Optional[Identifier]*) –

**\_\_len\_\_**(*context=None*)

Number of statements in the store. This should only account for non- quoted (asserted) statements if the context is not specified, otherwise it should return the number of statements in the formula or context given.

#### Parameters

- **context** (*Optional[Graph]*) – a graph instance to query or None

#### Return type

*int*

**\_\_module\_\_** = 'rdflib.store'

**\_\_weakref\_\_**

list of weak references to the object (if defined)

**add**(*triple, context, quoted=False*)

Adds the given statement to a specific context or to the model. The quoted argument is interpreted by formula-aware stores to indicate this statement is quoted/hypothetical It should be an error to not specify a context and have the quoted argument be True. It should also be an error for the quoted argument to be True when the store is not formula-aware.

#### Parameters

- **triple** (*Tuple[Node, Node, Node]*) –
- **context** (*Graph*) –
- **quoted** (*bool*) –

#### Return type

*None*



### **addN(*quads*)**

Adds each item in the list of statements to a specific context. The quoted argument is interpreted by formula-aware stores to indicate this statement is quoted/hypothetical. Note that the default implementation is a redirect to add

#### **Parameters**

**quads** (*Iterable*[*Tuple*[*Node*, *Node*, *Node*, *Graph*]]) –

#### **Return type**

*None*

### **add\_graph(*graph*)**

Add a graph to the store, no effect if the graph already exists. :type graph: *Graph* :param graph: a Graph instance

#### **Return type**

*None*

### **bind(*prefix*, *namespace*, *override=True*)**

#### **Parameters**

- **override** (*bool*) – rebind, even if the given namespace is already bound to another prefix.
- **prefix** (*str*) –
- **namespace** (*URIRef*) –

#### **Return type**

*None*

### **close(*commit\_pending\_transaction=False*)**

This closes the database connection. The *commit\_pending\_transaction* parameter specifies whether to commit all pending transactions before closing (if the store is transactional).

#### **Parameters**

**commit\_pending\_transaction** (*bool*) –

#### **Return type**

*None*

### **commit()**

#### **Return type**

*None*

**context\_aware:** *bool* = *False*

### **contexts(*triple=None*)**

Generator over all contexts in the graph. If triple is specified, a generator over all contexts the triple is in. if store is *graph\_aware*, may also return empty contexts

#### **Return type**

*Generator*[*Graph*, *None*, *None*]

#### **Returns**

a generator over Nodes

#### **Parameters**

**triple** (*Optional*[*Tuple*[*Node*, *Node*, *Node*]]) –

**create**(*configuration*)

**Parameters**

**configuration** (*str*) –

**Return type**

*None*

**destroy**(*configuration*)

This destroys the instance of the store identified by the configuration string.

**Parameters**

**configuration** (*str*) –

**Return type**

*None*

**formula\_aware:** *bool* = *False*

**gc**()

Allows the store to perform any needed garbage collection

**Return type**

*None*

**graph\_aware:** *bool* = *False*

**namespace**(*prefix*)

**Parameters**

**prefix** (*str*) –

**Return type**

*Optional*[*URIRef*]

**namespaces**()

**Return type**

*Iterator*[*Tuple*[*str*, *URIRef*]]

**property node\_pickler:** *NodePickler*

**open**(*configuration*, *create=False*)

Opens the store specified by the configuration string. If create is True a store will be created if it does not already exist. If create is False and a store does not already exist an exception is raised. An exception is also raised if a store exists, but there is insufficient permissions to open the store. This should return one of: *VALID\_STORE*, *CORRUPTED\_STORE*, or *NO\_STORE*

**Parameters**

- **configuration** (*str*) –

- **create** (*bool*) –

**Return type**

*Optional*[*int*]

**prefix**(*namespace*)

**Parameters**

**namespace** (*URIRef*) –

**Return type**`Optional[str]`**query**(*query*, *initNs*, *initBindings*, *queryGraph*, *\*\*kwargs*)

If stores provide their own SPARQL implementation, override this.

*queryGraph* is `None`, a `URIRef` or `'__UNION__'` If `None` the graph is specified in the query-string/object If `URIRef` it specifies the graph to query, If `'__UNION__'` the union of all named graphs should be queried (This is used by `ConjunctiveGraphs` Values other than `None` obviously only makes sense for context-aware stores.)

**Parameters**

- **query** (`Union[Query, str]`) –
- **initNs** (`Mapping[str, Any]`) –
- **initBindings** (`Mapping[str, Identifier]`) –
- **queryGraph** (`str`) –
- **kwargs** (`Any`) –

**Return type**`Result`**remove**(*triple*, *context=None*)

Remove the set of triples matching the pattern from the store

**Parameters**

- **triple** (`Tuple[Optional[Node], Optional[Node], Optional[Node]]`) –
- **context** (`Optional[Graph]`) –

**Return type**`None`**remove\_graph**(*graph*)

Remove a graph from the store, this should also remove all triples in the graph

**Parameters**

- **graphid** – a `Graph` instance
- **graph** (`Graph`) –

**Return type**`None`**rollback**()**Return type**`None`**transaction\_aware:** `bool = False`**triples**(*triple\_pattern*, *context=None*)

A generator over all the triples matching the pattern. Pattern can include any objects for used for comparing against nodes in the store, for example, `REGEXTerm`, `URIRef`, `Literal`, `BNode`, `Variable`, `Graph`, `QuotedGraph`, `Date?` `DateRange?`

**Parameters**

- **context** (`Optional[Graph]`) – A conjunctive query can be indicated by either providing a value of `None`, or a specific context can be queries by passing a `Graph` instance (if store is context aware).
- **triple\_pattern** (`Tuple[Optional[Node], Optional[Node], Optional[Node]]`) –

**Return type**

`Iterator[Tuple[Tuple[Node, Node, Node], Iterator[Optional[Graph]]]]`

**triples\_choices**(*triple*, *context=None*)

A variant of triples that can take a list of terms instead of a single term in any slot. Stores can implement this to optimize the response time from the default ‘fallback’ implementation, which will iterate over each term in the list and dispatch to triples

**Parameters**

- **triple** (`Union[Tuple[List[Node], Node, Node], Tuple[Node, List[Node], Node], Tuple[Node, Node, List[Node]]]`) –
- **context** (`Optional[Graph]`) –

**Return type**

`Generator[Tuple[Tuple[Node, Node, Node], Iterator[Optional[Graph]]], None, None]`

**update**(*update*, *initNs*, *initBindings*, *queryGraph*, *\*\*kwargs*)

If stores provide their own (SPARQL) Update implementation, override this.

*queryGraph* is `None`, a `URIRef` or ‘\_\_UNION\_\_’ If `None` the graph is specified in the query-string/object If `URIRef` it specifies the graph to query, If ‘\_\_UNION\_\_’ the union of all named graphs should be queried (This is used by `ConjunctiveGraphs` Values other than `None` obviously only makes sense for context-aware stores.)

**Parameters**

- **update** (`Union[Update, str]`) –
- **initNs** (`Mapping[str, Any]`) –
- **initBindings** (`Mapping[str, Identifier]`) –
- **queryGraph** (`str`) –
- **kwargs** (`Any`) –

**Return type**

`None`

**class** `rdflib.store.StoreCreatedEvent`(*\*\*kw*)

Bases: `Event`

This event is fired when the Store is created, it has the following attribute:

- **configuration**: string used to create the store

**\_\_module\_\_** = ‘rdflib.store’

**class** `rdflib.store.TripleAddedEvent`(*\*\*kw*)

Bases: `Event`

This event is fired when a triple is added, it has the following attributes:

- the triple added to the graph
- the context of the triple, if any

- the graph to which the triple was added

```
__module__ = 'rdflib.store'
```

```
class rdflib.store.TripleRemovedEvent(**kw)
```

Bases: *Event*

This event is fired when a triple is removed, it has the following attributes:

- the triple removed from the graph
- the context of the triple, if any
- the graph from which the triple was removed

```
__module__ = 'rdflib.store'
```

### rdflib.term module

This module defines the different types of terms. Terms are the kinds of objects that can appear in a quoted/asserted triple. This includes those that are core to RDF:

- *Blank Nodes*
- *URI References*
- *Literals* (which consist of a literal value, datatype and language tag)

Those that extend the RDF model into N3:

- *Formulae*
- *Universal Quantifications (Variables)*

And those that are primarily for matching against 'Nodes' in the underlying Graph:

- REGEX Expressions
- Date Ranges
- Numerical Ranges

```
class rdflib.term.BNode(value: ~typing.Optional[str] = None, _sn_gen: ~typing.Callable[[], str] = <function
    _serial_number_generator.<locals>._generator>, _prefix: str = 'N')
```

Bases: *IdentifiedNode*

RDF 1.1's Blank Nodes Section: <https://www.w3.org/TR/rdf11-concepts/#section-blank-nodes>

Blank Nodes are local identifiers for unnamed nodes in RDF graphs that are used in some concrete RDF syntaxes or RDF store implementations. They are always locally scoped to the file or RDF store, and are not persistent or portable identifiers for blank nodes. The identifiers for Blank Nodes are not part of the RDF abstract syntax, but are entirely dependent on particular concrete syntax or implementation (such as Turtle, JSON-LD).

—

RDFLib's BNode class makes unique IDs for all the Blank Nodes in a Graph but you should *never* expect, or reply on, BNodes' IDs to match across graphs, or even for multiple copies of the same graph, if they are regenerated from some non-RDFLib source, such as loading from RDF data.

```
__module__ = 'rdflib.term'
```

```
static __new__(cls, value=None, _sn_gen=<function _serial_number_generator.<locals>._generator>,
              _prefix='N')
```

# only store implementations should pass in a value

**Parameters**

- **value** (Optional[str]) –
- **\_sn\_gen** (Callable[[], str]) –
- **\_prefix** (str) –

**Return type**

*BNode*

```
__reduce__()
```

Helper for pickle.

**Return type**

Tuple[Type[BNode], Tuple[str]]

```
__repr__()
```

Return repr(self).

**Return type**

str

```
__slots__ = ()
```

```
n3(namespace_manager=None)
```

**Parameters**

**namespace\_manager** (Optional[NamespaceManager]) –

**Return type**

str

```
skolemize(authority=None, basepath=None)
```

Create a URIRef “skolem” representation of the BNode, in accordance with <http://www.w3.org/TR/rdf11-concepts/#section-skolemization>

New in version 4.0.

**Parameters**

- **authority** (Optional[str]) –
- **basepath** (Optional[str]) –

**Return type**

*URIRef*

```
class rdflib.term.IdentifiedNode(value: str)
```

Bases: *Identifier*

An abstract class, primarily defined to identify Nodes that are not Literals.

The name “Identified Node” is not explicitly defined in the RDF specification, but can be drawn from this section: <https://www.w3.org/TR/rdf-concepts/#section-URI-Vocabulary>

```
__dict__ = mappingproxy({'__module__': 'rdflib.term', '__doc__': '\n An abstract
class, primarily defined to identify Nodes that are not Literals.\n\n The name
"Identified Node" is not explicitly defined in the RDF specification, but can be
drawn from this section:
https://www.w3.org/TR/rdf-concepts/#section-URI-Vocabulary\n ', '__getnewargs__':
<function IdentifiedNode.__getnewargs__>, 'toPython': <function
IdentifiedNode.toPython>, '__dict__': <attribute '__dict__' of 'IdentifiedNode'
objects>, '__weakref__': <attribute '__weakref__' of 'IdentifiedNode' objects>,
'__annotations__': {}})
```

```
__getnewargs__()
```

**Return type**

`Tuple[str]`

```
__module__ = 'rdflib.term'
```

```
__weakref__
```

list of weak references to the object (if defined)

```
toPython()
```

**Return type**

`str`

```
class rdflib.term.Identifier(value: str)
```

Bases: `Node`, `str`

See <http://www.w3.org/2002/07/rdf-identifer-terminology/> regarding choice of terminology.

```
__eq__(other)
```

Equality for Nodes.

```
>>> BNode("foo")==None
False
>>> BNode("foo")==URIRef("foo")
False
>>> URIRef("foo")==BNode("foo")
False
>>> BNode("foo")!=URIRef("foo")
True
>>> URIRef("foo")!=BNode("foo")
True
>>> Variable('a")!=URIRef('a')
True
>>> Variable('a")!=Variable('a')
False
```

**Parameters**

**other** (*Any*) –

**Return type**

`bool`

```
__ge__(other)
```

Return self>=value.

**Parameters****other** (*Any*) –**Return type***bool***\_\_gt\_\_**(*other*)

This implements ordering for Nodes,

This tries to implement this: <http://www.w3.org/TR/sparql11-query/#modOrderBy>

Variables are not included in the SPARQL list, but they are greater than BNodes and smaller than everything else

**Parameters****other** (*Any*) –**Return type***bool***\_\_hash\_\_**()

Return hash(self).

**\_\_le\_\_**(*other*)

Return self&lt;=value.

**Parameters****other** (*Any*) –**Return type***bool***\_\_lt\_\_**(*other*)

Return self&lt;value.

**Parameters****other** (*Any*) –**Return type***bool***\_\_module\_\_** = 'rdflib.term'**\_\_ne\_\_**(*other*)

Return self!=value.

**Parameters****other** (*Any*) –**Return type***bool***static** **\_\_new\_\_**(*cls, value*)**Parameters****value** (*str*) –**Return type***Identifier***\_\_slots\_\_** = ()



**eq**(*other*)

A “semantic”/interpreted equality function, by default, same as `__eq__`

**Parameters**

**other** (*Any*) –

**Return type**

`bool`

**neq**(*other*)

A “semantic”/interpreted not equal function, by default, same as `__ne__`

**Parameters**

**other** (*Any*) –

**Return type**

`bool`

**startswith**(*prefix*, *start=Ellipsis*, *end=Ellipsis*)

Return True if S starts with the specified prefix, False otherwise. With optional start, test S beginning at that position. With optional end, stop comparing S at that position. prefix can also be a tuple of strings to try.

**Parameters**

**prefix** (*str*) –

**Return type**

`bool`

**class** `rdflib.term.Literal`(*lexical\_or\_value: Any*, *lang: Optional[str] = None*, *datatype: Optional[str] = None*, *normalize: Optional[bool] = None*)

Bases: `Identifier`

RDF 1.1’s Literals Section: <http://www.w3.org/TR/rdf-concepts/#section-Graph-Literal>

Literals are used for values such as strings, numbers, and dates.

A literal in an RDF graph consists of two or three elements:

- a lexical form, being a Unicode string, which SHOULD be in Normal Form C
- a datatype IRI, being an IRI identifying a datatype that determines how the lexical form maps to a literal value, and
- if and only if the datatype IRI is `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString`, a non-empty language tag. The language tag MUST be well-formed according to section 2.2.9 of [Tags for identifying languages](#).

A literal is a language-tagged string if the third element is present. Lexical representations of language tags MAY be converted to lower case. The value space of language tags is always in lower case.

—

For valid XSD datatypes, the lexical form is optionally normalized at construction time. Default behaviour is set by `rdflib.NORMALIZE_LITERALS` and can be overridden by the `normalize` parameter to `__new__`

Equality and hashing of Literals are done based on the lexical form, i.e.:

```
>>> from rdflib.namespace import XSD
```

```
>>> Literal('01') != Literal('1') # clear - strings differ
True
```

but with data-type they get normalized:

```
>>> Literal('01', datatype=XSD.integer) != Literal('1', datatype=XSD.integer)
False
```

unless disabled:

```
>>> Literal('01', datatype=XSD.integer, normalize=False) != Literal('1',
↳datatype=XSD.integer)
True
```

Value based comparison is possible:

```
>>> Literal('01', datatype=XSD.integer).eq(Literal('1', datatype=XSD.float))
True
```

The eq method also provides limited support for basic python types:

```
>>> Literal(1).eq(1) # fine - int compatible with xsd:integer
True
>>> Literal('a').eq('b') # fine - str compatible with plain-lit
False
>>> Literal('a', datatype=XSD.string).eq('a') # fine - str compatible with
↳xsd:string
True
>>> Literal('a').eq(1) # not fine, int incompatible with plain-lit
NotImplemented
```

Greater-than/less-than ordering comparisons are also done in value space, when compatible datatypes are used. Incompatible datatypes are ordered by DT, or by lang-tag. For other nodes the ordering is None < BNode < URIRef < Literal

Any comparison with non-rdflib Node are “NotImplemented” In PY3 this is an error.

```
>>> from rdflib import Literal, XSD
>>> lit2006 = Literal('2006-01-01',datatype=XSD.date)
>>> lit2006.toPython()
datetime.date(2006, 1, 1)
>>> lit2006 < Literal('2007-01-01',datatype=XSD.date)
True
>>> Literal(datetime.utcnow()).datatype
rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#dateTime')
>>> Literal(1) > Literal(2) # by value
False
>>> Literal(1) > Literal(2.0) # by value
False
>>> Literal('1') > Literal(1) # by DT
True
>>> Literal('1') < Literal('1') # by lexical form
False
>>> Literal('a', lang='en') > Literal('a', lang='fr') # by lang-tag
False
>>> Literal(1) > URIRef('foo') # by node-type
True
```

The > < operators will eat this NotImplemented and throw a TypeError (py3k):

```
>>> Literal(1).__gt__(2.0)
NotImplemented
```

**\_\_abs\_\_()**

**Return type**

*Literal*

```
>>> abs(Literal(-1))
rdflib.term.Literal(u'1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#integer'))
```

```
>>> from rdflib.namespace import XSD
>>> abs(Literal("-1", datatype=XSD.integer))
rdflib.term.Literal(u'1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#integer'))
```

```
>>> abs(Literal("1"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Not a number; rdflib.term.Literal(u'1')
```

**\_\_add\_\_(val)**

```
>>> from rdflib.namespace import XSD
>>> Literal(1) + 1
rdflib.term.Literal(u'2', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#integer'))
>>> Literal("1") + "1"
rdflib.term.Literal(u'11')
```

```
# Handling dateTime/date/time based operations in Literals >>> a = Literal('2006-01-01T20:50:00',
datatype=XSD.dateTime) >>> b = Literal('P31D', datatype=XSD.duration) >>> (a + b)
rdflib.term.Literal('2006-02-01T20:50:00', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#dateTime'))
>>> from rdflib.namespace import XSD >>> a = Literal('2006-07-01T20:52:00', datatype=XSD.dateTime) >>> b = Literal('P122DT15H58M', datatype=XSD.duration)
>>> (a + b) rdflib.term.Literal('2006-11-01T12:50:00', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#dateTime'))
```

**Parameters**

**val** (*Any*) –

**Return type**

*Literal*

```
__annotations__ = {'_datatype': typing.Optional[rdflib.term.URIRef], '_ill_typed':
typing.Optional[bool], '_language': typing.Optional[str], '_value': typing.Any}
```

**\_\_bool\_\_()**

Is the Literal “True” This is used for if statements, bool(literal), etc.

**Return type**

*bool*

`__eq__(other)`

Literals are only equal to other literals.

“Two literals are equal if and only if all of the following hold: \* The strings of the two lexical forms compare equal, character by character. \* Either both or neither have language tags. \* The language tags, if any, compare equal. \* Either both or neither have datatype URIs. \* The two datatype URIs, if any, compare equal, character by character.” – 6.5.1 Literal Equality (RDF: Concepts and Abstract Syntax)

```
>>> Literal("1", datatype=URIRef("foo")) == Literal("1", datatype=URIRef("foo"))
True
>>> Literal("1", datatype=URIRef("foo")) == Literal("1", datatype=URIRef("foo2
→"))
False
```

```
>>> Literal("1", datatype=URIRef("foo")) == Literal("2", datatype=URIRef("foo"))
False
>>> Literal("1", datatype=URIRef("foo")) == "asdf"
False
>>> from rdflib import XSD
>>> Literal('2007-01-01', datatype=XSD.date) == Literal('2007-01-01',
→datatype=XSD.date)
True
>>> Literal('2007-01-01', datatype=XSD.date) == date(2007, 1, 1)
False
>>> Literal("one", lang="en") == Literal("one", lang="en")
True
>>> Literal("hast", lang='en') == Literal("hast", lang='de')
False
>>> Literal("1", datatype=XSD.integer) == Literal(1)
True
>>> Literal("1", datatype=XSD.integer) == Literal("01", datatype=XSD.integer)
True
```

**Parameters**

**other** (*Any*) –

**Return type**

*bool*

`__ge__(other)`

Return self>=value.

**Parameters**

**other** (*Any*) –

**Return type**

*bool*

`__getstate__()`**Return type**

*Tuple[None, Dict[str, Optional[str]]]*

`__gt__(other)`

This implements ordering for Literals, the other comparison methods delegate here

This tries to implement this: <http://www.w3.org/TR/sparql11-query/#modOrderBy>

In short, Literals with compatible data-types are ordered in value space, i.e. >>> from rdflib import XSD

```
>>> Literal(1) > Literal(2) # int/int
False
>>> Literal(2.0) > Literal(1) # double/int
True
>>> from decimal import Decimal
>>> Literal(Decimal("3.3")) > Literal(2.0) # decimal/double
True
>>> Literal(Decimal("3.3")) < Literal(4.0) # decimal/double
True
>>> Literal('b') > Literal('a') # plain lit/plain lit
True
>>> Literal('b') > Literal('a', datatype=XSD.string) # plain lit/xsd:str
True
```

Incompatible datatype mismatches ordered by DT

```
>>> Literal(1) > Literal("2") # int>string
False
```

Langtagged literals by lang tag >>> Literal("a", lang="en") > Literal("a", lang="fr") False

#### Parameters

**other** (Any) –

#### Return type

bool

#### \_\_hash\_\_()

```
>>> from rdflib.namespace import XSD
>>> a = {Literal('1', datatype=XSD.integer): 'one'}
:rtype: :py:class:`int`
```

```
>>> Literal('1', datatype=XSD.double) in a
False
```

“Called for the key object for dictionary operations, and by the built-in function hash(). Should return a 32-bit integer usable as a hash value for dictionary operations. The only required property is that objects which compare equal have the same hash value; it is advised to somehow mix together (e.g., using exclusive or) the hash values for the components of the object that also play a part in comparison of objects.” – 3.4.1 Basic customization (Python)

“Two literals are equal if and only if all of the following hold: \* The strings of the two lexical forms compare equal, character by character. \* Either both or neither have language tags. \* The language tags, if any, compare equal. \* Either both or neither have datatype URIs. \* The two datatype URIs, if any, compare equal, character by character.” – 6.5.1 Literal Equality (RDF: Concepts and Abstract Syntax)

#### \_\_invert\_\_()

#### Return type

Literal

```
>>> ~(Literal(-1))
rdflib.term.Literal(u'0', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#integer'))
```

```
>>> from rdflib.namespace import XSD
>>> ~( Literal("-1", datatype=XSD.integer))
rdflib.term.Literal(u'0', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
```

Not working:

```
>>> ~(Literal("1"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Not a number; rdflib.term.Literal(u'1')
```

`__le__(other)`

```
>>> from rdflib.namespace import XSD
>>> Literal('2007-01-01T10:00:00', datatype=XSD.dateTime
...         ) <= Literal('2007-01-01T10:00:00', datatype=XSD.dateTime)
True
```

**Parameters**

**other** (Any) –

**Return type**

bool

`__lt__(other)`

Return self<value.

**Parameters**

**other** (Any) –

**Return type**

bool

`__module__` = 'rdflib.term'

`__neg__()`

```
>>> (- Literal(1))
rdflib.term.Literal(u'-1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
>>> (- Literal(10.5))
rdflib.term.Literal(u'-10.5', datatype=rdflib.term.URIRef(u'http://www.w3.org/
↳2001/XMLSchema#double'))
>>> from rdflib.namespace import XSD
:rtype: :py:class:`~rdflib.term.Literal`
```

```
>>> (- Literal("1", datatype=XSD.integer))
rdflib.term.Literal(u'-1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
```

```
>>> (- Literal("1"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

(continues on next page)

(continued from previous page)

```
TypeError: Not a number; rdflib.term.Literal(u'1')
>>>
```

```
static __new__(cls, lexical_or_value, lang=None, datatype=None, normalize=None)
```

**Parameters**

- **lexical\_or\_value** (*Any*) –
- **lang** (*Optional[str]*) –
- **datatype** (*Optional[str]*) –
- **normalize** (*Optional[bool]*) –

**Return type***Literal*

```
__pos__()
```

```
>>> (+ Literal(1))
rdflib.term.Literal(u'1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
>>> (+ Literal(-1))
rdflib.term.Literal(u'-1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
>>> from rdflib.namespace import XSD
:rtype: :py:class: `~rdflib.term.Literal`
```

```
>>> (+ Literal("-1", datatype=XSD.integer))
rdflib.term.Literal(u'-1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
```

```
>>> (+ Literal("1"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Not a number; rdflib.term.Literal(u'1')
```

```
__reduce__()
```

Helper for pickle.

**Return type***Tuple[Type[Literal], Tuple[str, Optional[str], Optional[str]]]*

```
__repr__()
```

Return repr(self).

**Return type***str*

```
__setstate__(arg)
```

**Parameters**

**arg** (*Tuple[Any, Dict[str, Any]]*) –

**Return type***None*

```
__slots__ = ('_language', '_datatype', '_value', '_ill_typed')
__sub__(val)
```

```
>>> from rdflib.namespace import XSD
>>> Literal(2) - 1
rdflib.term.Literal('1', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#integer'))
>>> Literal(1.1) - 1.0
rdflib.term.Literal('0.100000000000000009', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#double'))
>>> Literal(1.1) - 1
rdflib.term.Literal('0.1', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#decimal'))
>>> Literal(1.1, datatype=XSD.float) - Literal(1.0, datatype=XSD.float)
rdflib.term.Literal('0.100000000000000009', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#float'))
>>> Literal("1.1") - 1.0
Traceback (most recent call last):
...
TypeError: Not a number; rdflib.term.Literal('1.1')
>>> Literal(1.1, datatype=XSD.integer) - Literal(1.0, datatype=XSD.integer)
rdflib.term.Literal('0.100000000000000009', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#integer'))
```

```
# Handling dateTime/date/time based operations in Literals >>> a = Literal('2006-01-01T20:50:00',
datatype=XSD.dateTime) >>> b = Literal('2006-02-01T20:50:00', datatype=XSD.dateTime) >>> (b -
a) rdflib.term.Literal('P31D', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#duration'))
>>> from rdflib.namespace import XSD >>> a = Literal('2006-07-01T20:52:00',
datatype=XSD.dateTime) >>> b = Literal('2006-11-01T12:50:00', datatype=XSD.dateTime)
>>> (a - b) rdflib.term.Literal('-P122DT15H58M', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#duration'))
>>> (b - a) rdflib.term.Literal('P122DT15H58M', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#duration'))
```

#### Parameters

**val** (*Any*) –

#### Return type

*Literal*

**property datatype:** *Optional*[*URIRef*]

**eq**(*other*)

Compare the value of this literal with something else

Either, with the value of another literal comparisons are then done in literal “value space”, and according to the rules of XSD subtype-substitution/type-promotion

OR, with a python object:

basestring objects can be compared with plain-literals, or those with datatype xsd:string

bool objects with xsd:boolean

a int, long or float with numeric xsd types

isodate date,time,datetime objects with xsd:date,xsd:time or xsd:datetime

Any other operations returns NotImplemented



**Parameters****other** (*Any*) –**Return type***bool***property ill\_typed:** *Optional*[*bool*]

For *recognized datatype IRIs*, this value will be *True* if the literal is ill formed, otherwise it will be *False*. *Literal.value* (i.e. the *literal value*) should always be defined if this property is *False*, but should not be considered reliable if this property is *True*.

If the literal's datatype is *None* or not in the set of *recognized datatype IRIs* this value will be *None*.

**property language:** *Optional*[*str*]**n3**(*namespace\_manager=None*)

Returns a representation in the N3 format.

Examples:

```
>>> Literal("foo").n3()
u'"foo"'
```

Strings with newlines or triple-quotes:

```
>>> Literal("foo\nbar").n3()
u'"""foo\nbar"""'
```

```
>>> Literal("'\"'").n3()
u'"\'\\"''
```

```
>>> Literal('""').n3()
u'"\\"\\\"\\\"\\\"'
```

Language:

```
>>> Literal("hello", lang="en").n3()
u'"hello"@en'
```

Datatypes:

```
>>> Literal(1).n3()
u'"1"^^<http://www.w3.org/2001/XMLSchema#integer>'
```

```
>>> Literal(1.0).n3()
u'"1.0"^^<http://www.w3.org/2001/XMLSchema#double>'
```

```
>>> Literal(True).n3()
u'"true"^^<http://www.w3.org/2001/XMLSchema#boolean>'
```

Datatype and language isn't allowed (datatype takes precedence):

```
>>> Literal(1, lang="en").n3()
u'"1"^^<http://www.w3.org/2001/XMLSchema#integer>'
```

Custom datatype:

```
>>> footype = URIRef("http://example.org/ns#foo")
>>> Literal("1", datatype=footype).n3()
u'"1"^^<http://example.org/ns#foo>'
```

Passing a namespace-manager will use it to abbreviate datatype URIs:

```
>>> from rdflib import Graph
>>> Literal(1).n3(Graph().namespace_manager)
u'"1"^^xsd:integer'
```

#### Parameters

**namespace\_manager** (*Optional*[*NamespaceManager*]) –

#### Return type

*str*

#### **neq**(*other*)

A “semantic”/interpreted not equal function, by default, same as `__ne__`

#### Parameters

**other** (*Any*) –

#### Return type

*bool*

#### **normalize**()

Returns a new literal with a normalised lexical representation of this literal >>> from rdflib import XSD >>> Literal("01", datatype=XSD.integer, normalize=False).normalize() rdflib.term.Literal(u'1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#integer'))

Illegal lexical forms for the datatype given are simply passed on >>> Literal("a", datatype=XSD.integer, normalize=False) rdflib.term.Literal(u'a', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#integer'))

#### Return type

*Literal*

#### **toPython**()

Returns an appropriate python datatype derived from this RDF Literal

#### Return type

*Any*

**property value:** *Any*

#### **class** rdflib.term.Node

Bases: *object*

A Node in the Graph.

**\_\_module\_\_** = 'rdflib.term'

**\_\_slots\_\_** = ()

#### **class** rdflib.term.URIRef(*value: str, base: Optional[str] = None*)

Bases: *IdentifiedNode*

RDF 1.1’s IRI Section <https://www.w3.org/TR/rdf11-concepts/#section-IRIs>

---

**Note:** Documentation on RDF outside of RDFLib uses the term IRI or URI whereas this class is called `URIRef`. This is because it was made when the first version of the RDF specification was current, and it used the term *URIRef*, see [RDF 1.0 URIRef](#)

---

An IRI (Internationalized Resource Identifier) within an RDF graph is a Unicode string that conforms to the syntax defined in RFC 3987.

IRIs in the RDF abstract syntax MUST be absolute, and MAY contain a fragment identifier.

IRIs are a generalization of URIs [RFC3986] that permits a wider range of Unicode characters.

`__add__(other)`

Return self+value.

**Return type**

*URIRef*

```
__annotations__ = {'__invert__': typing.Callable[[ForwardRef('URIRef')],
ForwardRef('InvPath')], '__neg__': typing.Callable[[ForwardRef('URIRef')],
ForwardRef('NegatedPath')], '__or__': typing.Callable[[ForwardRef('URIRef'),
typing.Union[ForwardRef('URIRef'), ForwardRef('Path')]],
ForwardRef('AlternativePath')], '__truediv__':
typing.Callable[[ForwardRef('URIRef'), typing.Union[ForwardRef('URIRef'),
ForwardRef('Path')]], ForwardRef('SequencePath')]]}
```

`__invert__()`

inverse path

**Parameters**

**p** (*Union*[*URIRef*, *Path*]) –

**Return type**

*InvPath*

`__mod__(other)`

Return self%value.

**Return type**

*URIRef*

`__module__ = 'rdflib.term'`

`__mul__(mul)`

cardinality path

**Parameters**

- **p** (*Union*[*URIRef*, *Path*]) –
- **mul** (*Literal*['\*', '+', '?']) –

**Return type**

*MulPath*

`__neg__()`

negated path

**Parameters**

**p** (*Union*[*URIRef*, *AlternativePath*, *InvPath*]) –

**Return type***NegatedPath***static** `__new__(cls, value, base=None)`**Parameters**

- **value** (*str*) –
- **base** (*Optional[str]*) –

**Return type***URIRef*`__or__(other)`

alternative path

**Parameters**

- **self** (*Union[URIRef, Path]*) –
- **other** (*Union[URIRef, Path]*) –

`__radd__(other)`**Return type***URIRef*`__reduce__()`

Helper for pickle.

**Return type***Tuple[Type[URIRef], Tuple[str]]*`__repr__()`

Return repr(self).

**Return type***str*`__slots__ = ()``__truediv__(other)`

sequence path

**Parameters**

- **self** (*Union[URIRef, Path]*) –
- **other** (*Union[URIRef, Path]*) –

`de_skolemize()`

Create a Blank Node from a skolem URI, in accordance with <http://www.w3.org/TR/rdf11-concepts/#section-skolemization>. This function accepts only rdflib type skolemization, to provide a round-tripping within the system. :rtype: *BNode*

New in version 4.0.

`defrag()`**Return type***URIRef*

**property fragment:** `str`

Return the URL Fragment

```
>>> URIRef("http://example.com/some/path/#some-fragment").fragment
'some-fragment'
>>> URIRef("http://example.com/some/path/").fragment
''
```

**n3(namespace\_manager=None)**

This will do a limited check for valid URIs, essentially just making sure that the string includes no illegal characters (<, >, ", {, }, |, \, `, ^)

**Parameters**

**namespace\_manager** (`Optional[NamespaceManager]`) – if not None, will be used to make up a prefixed name

**Return type**

`str`

**class rdflib.term.Variable(value: str)**

Bases: `Identifier`

A Variable - this is used for querying, or in Formula aware graphs, where Variables can be stored

**\_\_module\_\_** = 'rdflib.term'

**static \_\_new\_\_(cls, value)**

**Parameters**

**value** (`str`) –

**Return type**

`Variable`

**\_\_reduce\_\_()**

Helper for pickle.

**Return type**

`Tuple[Type[Variable], Tuple[str]]`

**\_\_repr\_\_()**

Return repr(self).

**Return type**

`str`

**\_\_slots\_\_** = ()

**n3(namespace\_manager=None)**

**Parameters**

**namespace\_manager** (`Optional[NamespaceManager]`) –

**Return type**

`str`

**toPython()**

**Return type**

`str`

`rdflib.term.bind(datatype, pythontype, constructor=None, lexicalizer=None, datatype_specific=False)`

register a new datatype<->pythontype binding

#### Parameters

- **constructor** (`Optional[Callable[[str], Any]]`) – an optional function for converting lexical forms into a Python instances, if not given the pythontype is used directly
- **lexicalizer** (`Optional[Callable[[Any], Union[str, bytes]]]`) – an optional function for converting python objects to lexical form, if not given object.\_\_str\_\_ is used
- **datatype\_specific** (`bool`) – makes the lexicalizer function be accessible from the pair (pythontype, datatype) if set to True or from the pythontype otherwise. False by default
- **datatype** (`str`) –
- **pythontype** (`Type[Any]`) –

#### Return type

`None`

### rdflib.util module

`rdflib.util.date_time(t=None, local_time_zone=False)`

<http://www.w3.org/TR/NOTE-datetime> ex: 1997-07-16T19:20:30Z

```
>>> date_time(1126482850)
'2005-09-11T23:54:10Z'
```

@@ this will change depending on where it is run #>>> date\_time(1126482850, local\_time\_zone=True) #'2005-09-11T19:54:10-04:00'

```
>>> date_time(1)
'1970-01-01T00:00:01Z'
```

```
>>> date_time(0)
'1970-01-01T00:00:00Z'
```

`rdflib.util.find_roots(graph, prop, roots=None)`

Find the roots in some sort of transitive hierarchy.

`find_roots(graph, rdflib.RDFS.subClassOf)` will return a set of all roots of the sub-class hierarchy

Assumes triple of the form (child, prop, parent), i.e. the direction of `RDFS.subClassOf` or `SKOS.broader`

#### Parameters

- **graph** (`Graph`) –
- **prop** (`URIRef`) –
- **roots** (`Optional[Set[Node]]`) –

#### Return type

`Set[Node]`

`rdflib.util.first(seq)`

return the first element in a python sequence for graphs, use `graph.value` instead

#### Parameters

**seq** (`Iterable[TypeVar(_AnyT)]`) –

**Return type**`Optional[TypeVar(_AnyT)]``rdflib.util.from_n3(s, default=None, backend=None, nsm=None)`

Creates the Identifier corresponding to the given n3 string.

```

>>> from rdflib.term import URIRef, Literal
>>> from rdflib.namespace import NamespaceManager
>>> from_n3('<http://ex.com/foo>') == URIRef('http://ex.com/foo')
True
>>> from_n3('"foo"@de') == Literal('foo', lang='de')
True
>>> from_n3('"""multi\nline\nstring"""@en') == Literal(
...     'multi\nline\nstring', lang='en')
True
>>> from_n3('42') == Literal(42)
True
>>> from_n3(Literal(42).n3()) == Literal(42)
True
>>> from_n3('"42"^^xsd:integer') == Literal(42)
True
>>> from rdflib import RDFS
>>> from_n3('rdfs:label') == RDFS['label']
True
>>> nsm = NamespaceManager(rdflib.graph.Graph())
>>> nsm.bind('dbpedia', 'http://dbpedia.org/resource/')
>>> berlin = URIRef('http://dbpedia.org/resource/Berlin')
>>> from_n3('dbpedia:Berlin', nsm=nsm) == berlin
True

```

**Parameters**

- **s** (`str`) –
- **default** (`Optional[str]`) –
- **backend** (`Optional[str]`) –
- **nsm** (`Optional[NamespaceManager]`) –

**Return type**`Union[Node, str, None]`

`rdflib.util.get_tree(graph, root, prop, mapper=<function <lambda>>, sortkey=None, done=None, dir='down')`

Return a nested list/tuple structure representing the tree built by the transitive property given, starting from the root given

i.e.

```

get_tree(graph,
         rdflib.URIRef("http://xmlns.com/foaf/0.1/Person"), rdflib.RDFS.subClassOf)

```

will return the structure for the subClassTree below person.

`dir='down'` assumes triple of the form (child, prop, parent), i.e. the direction of `RDFS.subClassOf` or `SKOS.broader` Any other `dir` traverses in the other direction

**Parameters**

- **graph** (*Graph*) –
- **root** (*Node*) –
- **prop** (*URIRef*) –
- **mapper** (*Callable*[[*Node*], *Node*]) –
- **sortkey** (*Optional*[*Callable*[[*Any*], *Any*]]) –
- **done** (*Optional*[*Set*[*Node*]]) –
- **dir** (*str*) –

**Return type**

*Optional*[*Tuple*[*Node*, *List*[*Any*]]]

`rdflib.util.guess_format(fpath, fmap=None)`

Guess RDF serialization based on file suffix. Uses SUFFIX\_FORMAT\_MAP unless `fmap` is provided. Examples:

```
>>> guess_format('path/to/file.rdf')
'xml'
>>> guess_format('path/to/file.owl')
'xml'
>>> guess_format('path/to/file.ttl')
'turtle'
>>> guess_format('path/to/file.json')
'json-ld'
>>> guess_format('path/to/file.xhtml')
'rdfa'
>>> guess_format('path/to/file.svg')
'rdfa'
>>> guess_format('path/to/file.xhtml', {'xhtml': 'grddl'})
'grddl'
```

This also works with just the suffixes, with or without leading dot, and regardless of letter case:

```
>>> guess_format('.rdf')
'xml'
>>> guess_format('rdf')
'xml'
>>> guess_format('RDF')
'xml'
```

**Parameters**

- **fpath** (*str*) –
- **fmap** (*Optional*[*Dict*[*str*, *str*]]) –

**Return type**

*Optional*[*str*]

`rdflib.util.list2set(seq)`

Return a new list without duplicates. Preserves the order, unlike `set(seq)`

**Parameters**

**seq** (*Iterable*[*TypeVar*(*\_HashableT*, bound=*Hashable*)] –



**Return type**`List[TypeVar(_HashableT, bound= Hashable)]``rdflib.util.more_than(sequence, number)`

Returns 1 if sequence has more items than number and 0 if not.

**Parameters**

- **sequence** (`Iterable[Any]`) –
- **number** (`int`) –

**Return type**`int``rdflib.util.parse_date_time(val)`

always returns seconds in UTC

# tests are written like this to make any errors easier to understand >>> parse\_date\_time('2005-09-11T23:54:10Z') - 1126482850.0 0.0

```
>>> parse_date_time('2005-09-11T16:54:10-07:00') - 1126482850.0
0.0
```

```
>>> parse_date_time('1970-01-01T00:00:01Z') - 1.0
0.0
```

```
>>> parse_date_time('1970-01-01T00:00:00Z') - 0.0
0.0
>>> parse_date_time("2005-09-05T10:42:00") - 1125916920.0
0.0
```

**Parameters**`val (str)` –**Return type**`int``rdflib.util.to_term(s, default=None)`

Creates and returns an Identifier of type corresponding to the pattern of the given positional argument string s:

“” returns the default keyword argument value or None

‘<s>’ returns `URIRef(s)` (i.e. without angle brackets)

“s” returns `Literal(s)` (i.e. without doublequotes)

‘\_s’ returns `BNode(s)` (i.e. without leading underscore)

**Parameters**

- **s** (`Optional[str]`) –
- **default** (`Optional[Identifier]`) –

**Return type**`Optional[Identifier]`

`rdflib.util.uniq(sequence, strip=0)`

removes duplicate strings from the sequence.

**Parameters**

- **sequence** (`Iterable[str]`) –
- **strip** (`int`) –

**Return type**

`Set[str]`

## rdflib.void module

`rdflib.void.generateVoID(g, dataset=None, res=None, distinctForPartitions=True)`

Returns a new graph with a VoID description of the passed dataset

For more info on Vocabulary of Interlinked Datasets (VoID), see: <http://vocab.deri.ie/void>

This only makes two passes through the triples (once to detect the types of things)

The tradeoff is that lots of temporary structures are built up in memory meaning lots of memory may be consumed :) I imagine at least a few copies of your original graph.

the `distinctForPartitions` parameter controls whether `distinctSubjects/objects` are tracked for each `class/propertyPartition` this requires more memory again

## Module contents

A pure Python package providing the core RDF constructs.

The package is intended to provide the core RDF types and interfaces for working with RDF. The package defines a plugin interface for parsers, stores, and serializers that other packages can use to implement parsers, stores, and serializers that will plug into the `rdflib` package.

The primary interface `rdflib` exposes to work with RDF is `rdflib.graph.Graph`.

A tiny example:

```
>>> from rdflib import Graph, URIRef, Literal
```

```
>>> g = Graph()
>>> result = g.parse("http://www.w3.org/2000/10/swap/test/meet/blue.rdf")
```

```
>>> print("graph has %s statements." % len(g))
graph has 4 statements.
>>>
>>> for s, p, o in g:
...     if (s, p, o) not in g:
...         raise Exception("It better be!")
```

```
>>> s = g.serialize(format='nt')
>>>
>>> sorted(g) == [
...     (URIRef("http://meetings.example.com/cal#m1"),
...     URIRef("http://www.example.org/meeting_organization#homePage"),
```

(continues on next page)

(continued from previous page)

```

...   URIRef("http://meetings.example.com/m1/hp")),
...   (URIRef("http://www.example.org/people#fred"),
...    URIRef("http://www.example.org/meeting_organization#attending"),
...    URIRef("http://meetings.example.com/cal#m1")),
...   (URIRef("http://www.example.org/people#fred"),
...    URIRef("http://www.example.org/personal_details#GivenName"),
...    Literal("Fred")),
...   (URIRef("http://www.example.org/people#fred"),
...    URIRef("http://www.example.org/personal_details#hasEmail"),
...    URIRef("mailto:fred@example.com"))
... ]
True

```

```

class rdflib.BNode(value: ~typing.Optional[str] = None, _sn_gen: ~typing.Callable[[], str] = <function
    _serial_number_generator.<locals>._generator>, _prefix: str = 'N')

```

Bases: *IdentifiedNode*

RDF 1.1's Blank Nodes Section: <https://www.w3.org/TR/rdf11-concepts/#section-blank-nodes>

Blank Nodes are local identifiers for unnamed nodes in RDF graphs that are used in some concrete RDF syntaxes or RDF store implementations. They are always locally scoped to the file or RDF store, and are not persistent or portable identifiers for blank nodes. The identifiers for Blank Nodes are not part of the RDF abstract syntax, but are entirely dependent on particular concrete syntax or implementation (such as Turtle, JSON-LD).

—

RDFLib's BNode class makes unique IDs for all the Blank Nodes in a Graph but you should *never* expect, or rely on, BNodes' IDs to match across graphs, or even for multiple copies of the same graph, if they are regenerated from some non-RDFLib source, such as loading from RDF data.

```
__annotations__ = {}
```

```
__module__ = 'rdflib.term'
```

```

static __new__(cls, value=None, _sn_gen=<function _serial_number_generator.<locals>._generator>,
    _prefix='N')

```

# only store implementations should pass in a value

#### Parameters

- **value** (*Optional*[str]) –
- **\_sn\_gen** (*Callable*[[], str]) –
- **\_prefix** (str) –

#### Return type

*BNode*

```
__reduce__()
```

Helper for pickle.

#### Return type

*Tuple*[*Type*[*BNode*], *Tuple*[str]]

```
__repr__()
```

Return repr(self).

**Return type**`str``__slots__ = ()``n3(namespace_manager=None)`**Parameters**`namespace_manager` (`Optional[NamespaceManager]`) –**Return type**`str``skolemize(authority=None, basepath=None)`

Create a `URIRef` “skolem” representation of the `BNode`, in accordance with <http://www.w3.org/TR/rdf11-concepts/#section-skolemization>

New in version 4.0.

**Parameters**

- `authority` (`Optional[str]`) –
- `basepath` (`Optional[str]`) –

**Return type**`URIRef``class rdflib.BRICK`

Bases: `DefinedNamespace`

Brick Ontology classes, properties and entity properties. See <https://brickschema.org/> for more information.

Generated from: <https://github.com/BrickSchema/Brick/releases/download/nightly/Brick.ttl> Date: 2021-09-22T14:32:56

`AED: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#AED')``AHU: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#AHU')``Ablutions_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Ablutions_Room')``Absorption_Chiller: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Absorption_Chiller')``Acceleration_Time_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Acceleration_Time_Setpoint')``Access_Control_Equipment: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Access_Control_Equipment')``Access_Reader: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Access_Reader')``Active_Chilled_Beam: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Active_Chilled_Beam')``Active_Power_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Active_Power_Sensor')`

```
Adjust_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Adjust_Sensor')  
  
Air: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air')  
  
Air_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Alarm')  
  
Air_Differential_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Air_Differential_Pressure_Sensor')  
  
Air_Differential_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Air_Differential_Pressure_Setpoint')  
  
Air_Diffuser: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Diffuser')  
  
Air_Enthalpy_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Enthalpy_Sensor')  
  
Air_Flow_Deadband_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Air_Flow_Deadband_Setpoint')  
  
Air_Flow_Demand_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Flow_Demand_Setpoint')  
  
Air_Flow_Loss_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Flow_Loss_Alarm')  
  
Air_Flow_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Flow_Sensor')  
  
Air_Flow_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Flow_Setpoint')  
  
Air_Flow_Setpoint_Limit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Flow_Setpoint_Limit')  
  
Air_Grains_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Grains_Sensor')  
  
Air_Handler_Unit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Handler_Unit')  
  
Air_Handling_Unit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Handling_Unit')  
  
Air_Humidity_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Humidity_Setpoint')  
  
Air_Loop: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Loop')  
  
Air_Plenum: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Plenum')
```

```
Air_Quality_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Quality_Sensor')  
  
Air_Static_Pressure_Step_Parameter: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Air_Static_Pressure_Step_Parameter')  
  
Air_System: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_System')  
  
Air_Temperature_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Temperature_Alarm')  
  
Air_Temperature_Integral_Time_Parameter: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Air_Temperature_Integral_Time_Parameter')  
  
Air_Temperature_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Temperature_Sensor')  
  
Air_Temperature_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Air_Temperature_Setpoint')  
  
Air_Temperature_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Air_Temperature_Setpoint_Limit')  
  
Air_Temperature_Step_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Air_Temperature_Step_Parameter')  
  
Air_Wet_Bulb_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Air_Wet_Bulb_Temperature_Sensor')  
  
Alarm: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Alarm')  
  
Alarm_Delay_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Alarm_Delay_Parameter')  
  
Angle_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Angle_Sensor')  
  
Auditorium: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Auditorium')  
  
Automated_External_Defibrillator: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Automated_External_Defibrillator')  
  
Automatic_Mode_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Automatic_Mode_Command')  
  
Availability_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Availability_Status')  
  
Average_Cooling_Demand_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Average_Cooling_Demand_Sensor')  
  
Average_Discharge_Air_Flow_Sensor: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Average_Discharge_Air_Flow_Sensor')
```

```

Average_Exhaust_Air_Static_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Average_Exhaust_Air_Static_Pressure_Sensor')

Average_Heating_Demand_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Average_Heating_Demand_Sensor')

Average_Supply_Air_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Average_Supply_Air_Flow_Sensor')

Average_Zone_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Average_Zone_Air_Temperature_Sensor')

Baseboard_Radiator: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Baseboard_Radiator')

Basement: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Basement')

Battery: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Battery')

Battery_Energy_Storage_System: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Battery_Energy_Storage_System')

Battery_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Battery_Room')

Battery_Voltage_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Battery_Voltage_Sensor')

Bench_Space: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Bench_Space')

Blowdown_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Blowdown_Water')

Boiler: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Boiler')

Booster_Fan: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Booster_Fan')

Box_Mode_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Box_Mode_Command')

Break_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Break_Room')

Breaker_Panel: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Breaker_Panel')

Breakroom: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Breakroom')

Broadcast_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Broadcast_Room')

```

```

Building: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Building')

Building_Air: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Building_Air')

Building_Air_Humidity_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Building_Air_Humidity_Setpoint')

Building_Air_Static_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Building_Air_Static_Pressure_Sensor')

Building_Air_Static_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Building_Air_Static_Pressure_Setpoint')

Building_Chilled_Water_Meter: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Building_Chilled_Water_Meter')

Building_Electrical_Meter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Building_Electrical_Meter')

Building_Gas_Meter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Building_Gas_Meter')

Building_Hot_Water_Meter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Building_Hot_Water_Meter')

Building_Meter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Building_Meter')

Building_Water_Meter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Building_Water_Meter')

Bus_Riser: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Bus_Riser')

Bypass_Air: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Bypass_Air')

Bypass_Air_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Bypass_Air_Flow_Sensor')

Bypass_Air_Humidity_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Bypass_Air_Humidity_Setpoint')

Bypass_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Bypass_Command')

Bypass_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Bypass_Valve')

Bypass_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Bypass_Water')

Bypass_Water_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Bypass_Water_Flow_Sensor')

```



```
Bypass_Water_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Bypass_Water_Flow_Setpoint')

CAV: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#CAV')

CO: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO')

CO2: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO2')

CO2_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO2_Alarm')

CO2_Differential_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO2_Differential_Sensor')

CO2_Level_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO2_Level_Sensor')

CO2_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO2_Sensor')

CO2_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO2_Setpoint')

CO_Differential_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO_Differential_Sensor')

CO_Level_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO_Level_Sensor')

CO_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#CO_Sensor')

CRAC: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#CRAC')

Cafeteria: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cafeteria')

Camera: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Camera')

Capacity_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Capacity_Sensor')

Ceiling_Fan: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Ceiling_Fan')

Centrifugal_Chiller: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Centrifugal_Chiller')

Change_Filter_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Change_Filter_Alarm')

Chilled_Beam: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chilled_Beam')

Chilled_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chilled_Water')
```

```

Chilled_Water_Coil: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chilled_Water_Coil')

Chilled_Water_Differential_Pressure_Deadband_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Chilled_Water_Differential_Pressure_Deadband_Setpoint')

Chilled_Water_Differential_Pressure_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Chilled_Water_Differential_Pressure_Integral_Time_Parameter')

Chilled_Water_Differential_Pressure_Load_Shed_Reset_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Chilled_Water_Differential_Pressure_Load_Shed_Reset_Status')

Chilled_Water_Differential_Pressure_Load_Shed_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Chilled_Water_Differential_Pressure_Load_Shed_Setpoint')

Chilled_Water_Differential_Pressure_Load_Shed_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Chilled_Water_Differential_Pressure_Load_Shed_Status')

Chilled_Water_Differential_Pressure_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Chilled_Water_Differential_Pressure_Proportional_Band_Parameter')

Chilled_Water_Differential_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_Differential_Pressure_Sensor')

Chilled_Water_Differential_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_Differential_Pressure_Setpoint')

Chilled_Water_Differential_Pressure_Step_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Chilled_Water_Differential_Pressure_Step_Parameter')

Chilled_Water_Differential_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
/brickschema.org/schema/Brick#Chilled_Water_Differential_Temperature_Sensor')

Chilled_Water_Discharge_Flow_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_Discharge_Flow_Sensor')

Chilled_Water_Discharge_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_Discharge_Flow_Setpoint')

Chilled_Water_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chilled_Water_Flow_Sensor')

Chilled_Water_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Chilled_Water_Flow_Setpoint')

Chilled_Water_Loop: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chilled_Water_Loop')

```

```

Chilled_Water_Meter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chilled_Water_Meter')

Chilled_Water_Pump: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chilled_Water_Pump')

Chilled_Water_Pump_Differential_Pressure_Deadband_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Chilled_Water_Pump_Differential_Pressure_Deadband_Setpoint')

Chilled_Water_Return_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Chilled_Water_Return_Flow_Sensor')

Chilled_Water_Return_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_Return_Temperature_Sensor')

Chilled_Water_Static_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_Static_Pressure_Setpoint')

Chilled_Water_Supply_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Chilled_Water_Supply_Flow_Sensor')

Chilled_Water_Supply_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_Supply_Flow_Setpoint')

Chilled_Water_Supply_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_Supply_Temperature_Sensor')

Chilled_Water_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chilled_Water_System')

Chilled_Water_System_Enable_Command: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_System_Enable_Command')

Chilled_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Chilled_Water_Temperature_Sensor')

Chilled_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Chilled_Water_Temperature_Setpoint')

Chilled_Water_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chilled_Water_Valve')

Chiller: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Chiller')

Class: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Class')

Close_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Close_Limit')

Coil: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Coil')

Cold_Box: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cold_Box')

```

```
Coldest_Zone_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Coldest_Zone_Air_Temperature_Sensor')

Collection: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Collection')

Collection_Basin_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Collection_Basin_Water')

Collection_Basin_Water_Heater: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Collection_Basin_Water_Heater')

Collection_Basin_Water_Level_Alarm: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Collection_Basin_Water_Level_Alarm')

Collection_Basin_Water_Level_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Collection_Basin_Water_Level_Sensor')

Collection_Basin_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Collection_Basin_Water_Temperature_Sensor')

Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Command')

Common_Space: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Common_Space')

Communication_Loss_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Communication_Loss_Alarm')

Compressor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Compressor')

Computer_Room_Air_Conditioning: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Computer_Room_Air_Conditioning')

Concession: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Concession')

Condensate_Leak_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Condensate_Leak_Alarm')

Condenser: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Condenser')

Condenser_Heat_Exchanger: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Condenser_Heat_Exchanger')

Condenser_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Condenser_Water')

Condenser_Water_Bypass_Valve: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Condenser_Water_Bypass_Valve')

Condenser_Water_Isolation_Valve: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Condenser_Water_Isolation_Valve')
```

```
Condenser_Water_Pump: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Condenser_Water_Pump')  
  
Condenser_Water_System: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Condenser_Water_System')  
  
Condenser_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Condenser_Water_Temperature_Sensor')  
  
Condenser_Water_Valve: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Condenser_Water_Valve')  
  
Condensing_Natural_Gas_Boiler: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Condensing_Natural_Gas_Boiler')  
  
Conductivity_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Conductivity_Sensor')  
  
Conference_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Conference_Room')  
  
Constant_Air_Volume_Box: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Constant_Air_Volume_Box')  
  
Contact_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Contact_Sensor')  
  
Control_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Control_Room')  
  
Cooling_Coil: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cooling_Coil')  
  
Cooling_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cooling_Command')  
  
Cooling_Demand_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cooling_Demand_Sensor')  
  
Cooling_Demand_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cooling_Demand_Setpoint')  
  
Cooling_Discharge_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Cooling_Discharge_Air_Flow_Setpoint')  
  
Cooling_Discharge_Air_Temperature_Deadband_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Cooling_Discharge_Air_Temperature_Deadband_Setpoint')  
  
Cooling_Discharge_Air_Temperature_Integral_Time_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Cooling_Discharge_Air_Temperature_Integral_Time_Parameter')  
  
Cooling_Discharge_Air_Temperature_Proportional_Band_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Cooling_Discharge_Air_Temperature_Proportional_Band_Parameter')
```

```
Cooling_Start_Stop_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cooling_Start_Stop_Status')  
  
Cooling_Supply_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Cooling_Supply_Air_Flow_Setpoint')  
  
Cooling_Supply_Air_Temperature_Deadband_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Cooling_Supply_Air_Temperature_Deadband_Setpoint')  
  
Cooling_Supply_Air_Temperature_Integral_Time_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Cooling_Supply_Air_Temperature_Integral_Time_Parameter')  
  
Cooling_Supply_Air_Temperature_Proportional_Band_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Cooling_Supply_Air_Temperature_Proportional_Band_Parameter')  
  
Cooling_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Cooling_Temperature_Setpoint')  
  
Cooling_Tower: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cooling_Tower')  
  
Cooling_Tower_Fan: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cooling_Tower_Fan')  
  
Cooling_Valve: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cooling_Valve')  
  
Copy_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Copy_Room')  
  
Core_Temperature_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Core_Temperature_Sensor')  
  
Core_Temperature_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Core_Temperature_Setpoint')  
  
Cubicle: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cubicle')  
  
Current_Imbalance_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Current_Imbalance_Sensor')  
  
Current_Limit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Current_Limit')  
  
Current_Output_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Current_Output_Sensor')  
  
Current_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Current_Sensor')  
  
Curtailment_Override_Command: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Curtailment_Override_Command')
```

```
Cycle_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Cycle_Alarm')  
  
DC_Bus_Voltage_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#DC_Bus_Voltage_Sensor')  
  
DOAS: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#DOAS')  
  
Damper: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Damper')  
  
Damper_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Damper_Command')  
  
Damper_Position_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Damper_Position_Command')  
  
Damper_Position_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Damper_Position_Sensor')  
  
Damper_Position_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Damper_Position_Setpoint')  
  
Deadband_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Deadband_Setpoint')  
  
Deceleration_Time_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Deceleration_Time_Setpoint')  
  
Dedicated_Outdoor_Air_System_Unit: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Dedicated_Outdoor_Air_System_Unit')  
  
Dehumidification_Start_Stop_Status: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Dehumidification_Start_Stop_Status')  
  
Deionised_Water_Conductivity_Sensor: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Deionised_Water_Conductivity_Sensor')  
  
Deionised_Water_Level_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Deionised_Water_Level_Sensor')  
  
Deionized_Water: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Deionized_Water')  
  
Deionized_Water_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Deionized_Water_Alarm')  
  
Delay_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Delay_Parameter')  
  
Demand_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Demand_Sensor')  
  
Demand_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Demand_Setpoint')  
  
Derivative_Gain_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Derivative_Gain_Parameter')
```



```
Derivative_Time_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Derivative_Time_Parameter')  
  
Detention_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Detention_Room')  
  
Dew_Point_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Dew_Point_Setpoint')  
  
Dewpoint_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Dewpoint_Sensor')  
  
Differential_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Differential_Air_Temperature_Setpoint')  
  
Differential_Pressure_Bypass_Valve: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Differential_Pressure_Bypass_Valve')  
  
Differential_Pressure_Deadband_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Differential_Pressure_Deadband_Setpoint')  
  
Differential_Pressure_Integral_Time_Parameter: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Differential_Pressure_Integral_Time_Parameter')  
  
Differential_Pressure_Load_Shed_Status: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Differential_Pressure_Load_Shed_Status')  
  
Differential_Pressure_Proportional_Band: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Differential_Pressure_Proportional_Band')  
  
Differential_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Differential_Pressure_Sensor')  
  
Differential_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Differential_Pressure_Setpoint')  
  
Differential_Pressure_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Differential_Pressure_Setpoint_Limit')  
  
Differential_Pressure_Step_Parameter: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Differential_Pressure_Step_Parameter')  
  
Differential_Speed_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Differential_Speed_Sensor')  
  
Differential_Speed_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Differential_Speed_Setpoint')  
  
Differential_Supply_Return_Water_Temperature_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Differential_Supply_Return_Water_Temperature_Sensor')  
  
Dimmer: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Dimmer')  
  
Direct_Expansion_Cooling_Coil: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Direct_Expansion_Cooling_Coil')
```



```
Direct_Expansion_Heating_Coil: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Direct_Expansion_Heating_Coil')

Direction_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Direction_Command')

Direction_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Direction_Sensor')

Direction_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Direction_Status')

Disable_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Disable_Command')

Disable_Differential_Enthalpy_Command: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Disable_Differential_Enthalpy_Command')

Disable_Differential_Temperature_Command: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Disable_Differential_Temperature_Command')

Disable_Fixed_Enthalpy_Command: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Disable_Fixed_Enthalpy_Command')

Disable_Fixed_Temperature_Command: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Disable_Fixed_Temperature_Command')

Disable_Hot_Water_System_Outside_Air_Temperature_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Disable_Hot_Water_System_Outside_Air_Temperature_Setpoint')

Disable_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Disable_Status')

Discharge_Air: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Air')

Discharge_Air_Dewpoint_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Air_Dewpoint_Sensor')

Discharge_Air_Duct_Pressure_Status: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Air_Duct_Pressure_Status')

Discharge_Air_Flow_Demand_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Air_Flow_Demand_Setpoint')

Discharge_Air_Flow_High_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Air_Flow_High_Reset_Setpoint')

Discharge_Air_Flow_Low_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Air_Flow_Low_Reset_Setpoint')

Discharge_Air_Flow_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Air_Flow_Reset_Setpoint')
```

```

Discharge_Air_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Air_Flow_Sensor')

Discharge_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Discharge_Air_Flow_Setpoint')

Discharge_Air_Humidity_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Discharge_Air_Humidity_Sensor')

Discharge_Air_Humidity_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Discharge_Air_Humidity_Setpoint')

Discharge_Air_Smoke_Detection_Alarm: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Smoke_Detection_Alarm')

Discharge_Air_Static_Pressure_Deadband_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Discharge_Air_Static_Pressure_Deadband_Setpoint')

Discharge_Air_Static_Pressure_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Discharge_Air_Static_Pressure_Integral_Time_Parameter')

Discharge_Air_Static_Pressure_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Discharge_Air_Static_Pressure_Proportional_Band_Parameter')

Discharge_Air_Static_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Static_Pressure_Sensor')

Discharge_Air_Static_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Static_Pressure_Setpoint')

Discharge_Air_Static_Pressure_Step_Parameter: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Static_Pressure_Step_Parameter')

Discharge_Air_Temperature_Alarm: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Discharge_Air_Temperature_Alarm')

Discharge_Air_Temperature_Cooling_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Temperature_Cooling_Setpoint')

Discharge_Air_Temperature_Deadband_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Temperature_Deadband_Setpoint')

Discharge_Air_Temperature_Heating_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Temperature_Heating_Setpoint')

Discharge_Air_Temperature_High_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Temperature_High_Reset_Setpoint')

Discharge_Air_Temperature_Low_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Temperature_Low_Reset_Setpoint')

Discharge_Air_Temperature_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Discharge_Air_Temperature_Proportional_Band_Parameter')

```

```

Discharge_Air_Temperature_Reset_Differential_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Discharge_Air_Temperature_Reset_Differential_Setpoint')

Discharge_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Discharge_Air_Temperature_Sensor')

Discharge_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Temperature_Setpoint')

Discharge_Air_Temperature_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Temperature_Setpoint_Limit')

Discharge_Air_Temperature_Step_Parameter: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Temperature_Step_Parameter')

Discharge_Air_Velocity_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Air_Velocity_Pressure_Sensor')

Discharge_Chilled_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Chilled_Water')

Discharge_Fan: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Fan')

Discharge_Hot_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Hot_Water')

Discharge_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Discharge_Water')

Discharge_Water_Differential_Pressure_Deadband_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Discharge_Water_Differential_Pressure_Deadband_Setpoint')

Discharge_Water_Differential_Pressure_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Discharge_Water_Differential_Pressure_Integral_Time_Parameter')

Discharge_Water_Differential_Pressure_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Discharge_Water_Differential_Pressure_Proportional_Band_Parameter')

Discharge_Water_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Discharge_Water_Flow_Sensor')

Discharge_Water_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Discharge_Water_Flow_Setpoint')

Discharge_Water_Temperature_Alarm: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Water_Temperature_Alarm')

Discharge_Water_Temperature_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Discharge_Water_Temperature_Proportional_Band_Parameter')

```

```

Discharge_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Water_Temperature_Sensor')

Discharge_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Discharge_Water_Temperature_Setpoint')

Disconnect_Switch: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Disconnect_Switch')

Displacement_Flow_Air_Diffuser: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Displacement_Flow_Air_Diffuser')

Distribution_Frame: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Distribution_Frame')

Domestic_Hot_Water_Supply_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Domestic_Hot_Water_Supply_Temperature_Sensor')

Domestic_Hot_Water_Supply_Temperature_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Domestic_Hot_Water_Supply_Temperature_Setpoint')

Domestic_Hot_Water_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Domestic_Hot_Water_System')

Domestic_Hot_Water_System_Enable_Command: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Domestic_Hot_Water_System_Enable_Command')

Domestic_Hot_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Domestic_Hot_Water_Temperature_Setpoint')

Domestic_Hot_Water_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Domestic_Hot_Water_Valve')

Domestic_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Domestic_Water')

Domestic_Water_Loop: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Domestic_Water_Loop')

Drench_Hose: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Drench_Hose')

Drive_Ready_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Drive_Ready_Status')

Duration_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Duration_Sensor')

ESS_Panel: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#ESS_Panel')

EconCycle_Start_Stop_Status: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#EconCycle_Start_Stop_Status')

```

```

Economizer: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Economizer')

Economizer_Damper: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Economizer_Damper')

Effective_Air_Temperature_Cooling_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Effective_Air_Temperature_Cooling_Setpoint')

Effective_Air_Temperature_Heating_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Effective_Air_Temperature_Heating_Setpoint')

Effective_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Effective_Air_Temperature_Setpoint')

Effective_Discharge_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Effective_Discharge_Air_Temperature_Setpoint')

Effective_Return_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Effective_Return_Air_Temperature_Setpoint')

Effective_Room_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Effective_Room_Air_Temperature_Setpoint')

Effective_Supply_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Effective_Supply_Air_Temperature_Setpoint')

Effective_Zone_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Effective_Zone_Air_Temperature_Setpoint')

Electric_Baseboard_Radiator: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Electric_Baseboard_Radiator')

Electric_Boiler: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Electric_Boiler')

Electric_Radiator: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Electric_Radiator')

Electrical_Equipment: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Electrical_Equipment')

Electrical_Meter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Electrical_Meter')

Electrical_Power_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Electrical_Power_Sensor')

Electrical_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Electrical_Room')

Electrical_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Electrical_System')

Elevator: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Elevator')

```

```
Elevator_Shift: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Elevator_Shift')  
  
Elevator_Space: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Elevator_Space')  
  
Embedded_Surface_System_Panel: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Embedded_Surface_System_Panel')  
  
Embedded_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Embedded_Temperature_Sensor')  
  
Embedded_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Embedded_Temperature_Setpoint')  
  
Emergency_Air_Flow_System: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Emergency_Air_Flow_System')  
  
Emergency_Air_Flow_System_Status: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Emergency_Air_Flow_System_Status')  
  
Emergency_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Emergency_Alarm')  
  
Emergency_Generator_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Emergency_Generator_Alarm')  
  
Emergency_Generator_Status: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Emergency_Generator_Status')  
  
Emergency_Phone: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Emergency_Phone')  
  
Emergency_Power_Off_System: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Emergency_Power_Off_System')  
  
Emergency_Power_Off_System_Activated_By_High_Temperature_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Emergency_Power_Off_System_Activated_By_High_Temperature_Status')  
  
Emergency_Power_Off_System_Activated_By_Leak_Detection_System_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Emergency_Power_Off_System_Activated_By_Leak_Detection_System_Status')  
  
Emergency_Power_Off_System_Status: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Emergency_Power_Off_System_Status')  
  
Emergency_Push_Button_Status: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Emergency_Push_Button_Status')  
  
Emergency_Wash_Station: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Emergency_Wash_Station')  
  
Employee_Entrance_Lobby: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Employee_Entrance_Lobby')
```

```
Enable_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Enable_Command')  
  
Enable_Differential_Enthalpy_Command: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Enable_Differential_Enthalpy_Command')  
  
Enable_Differential_Temperature_Command: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Enable_Differential_Temperature_Command')  
  
Enable_Fixed_Enthalpy_Command: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Enable_Fixed_Enthalpy_Command')  
  
Enable_Fixed_Temperature_Command: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Enable_Fixed_Temperature_Command')  
  
Enable_Hot_Water_System_Outside_Air_Temperature_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Enable_Hot_Water_System_Outside_Air_Temperature_Setpoint')  
  
Enable_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Enable_Status')  
  
Enclosed_Office: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Enclosed_Office')  
  
Energy_Generation_System: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Energy_Generation_System')  
  
Energy_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Energy_Sensor')  
  
Energy_Storage: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Energy_Storage')  
  
Energy_Storage_System: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Energy_Storage_System')  
  
Energy_System: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Energy_System')  
  
Energy_Usage_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Energy_Usage_Sensor')  
  
Energy_Zone: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Energy_Zone')  
  
Entering_Water: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Entering_Water')  
  
Entering_Water_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Entering_Water_Flow_Sensor')  
  
Entering_Water_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Entering_Water_Flow_Setpoint')
```



```
Entering_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Entering_Water_Temperature_Sensor')

Entering_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Entering_Water_Temperature_Setpoint')

Enthalpy_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Enthalpy_Sensor')

Enthalpy_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Enthalpy_Setpoint')

Entrance: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Entrance')

Environment_Box: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Environment_Box')

Equipment: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Equipment')

Equipment_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Equipment_Room')

Evaporative_Heat_Exchanger: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Evaporative_Heat_Exchanger')

Even_Month_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Even_Month_Status')

Exercise_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exercise_Room')

Exhaust_Air: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Air')

Exhaust_Air_Dewpoint_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Air_Dewpoint_Sensor')

Exhaust_Air_Differential_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Air_Differential_Pressure_Sensor')

Exhaust_Air_Differential_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Air_Differential_Pressure_Setpoint')

Exhaust_Air_Flow_Integral_Time_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Air_Flow_Integral_Time_Parameter')

Exhaust_Air_Flow_Proportional_Band_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Air_Flow_Proportional_Band_Parameter')

Exhaust_Air_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Air_Flow_Sensor')

Exhaust_Air_Flow_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Air_Flow_Setpoint')
```



```

Exhaust_Air_Humidity_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Exhaust_Air_Humidity_Sensor')

Exhaust_Air_Humidity_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Exhaust_Air_Humidity_Setpoint')

Exhaust_Air_Stack_Flow_Deadband_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Exhaust_Air_Stack_Flow_Deadband_Setpoint')

Exhaust_Air_Stack_Flow_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Exhaust_Air_Stack_Flow_Integral_Time_Parameter')

Exhaust_Air_Stack_Flow_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Exhaust_Air_Stack_Flow_Proportional_Band_Parameter')

Exhaust_Air_Stack_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Exhaust_Air_Stack_Flow_Sensor')

Exhaust_Air_Stack_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Exhaust_Air_Stack_Flow_Setpoint')

Exhaust_Air_Static_Pressure_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Exhaust_Air_Static_Pressure_Proportional_Band_Parameter')

Exhaust_Air_Static_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Exhaust_Air_Static_Pressure_Sensor')

Exhaust_Air_Static_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Exhaust_Air_Static_Pressure_Setpoint')

Exhaust_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Exhaust_Air_Temperature_Sensor')

Exhaust_Air_Velocity_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Exhaust_Air_Velocity_Pressure_Sensor')

Exhaust_Damper: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Damper')

Exhaust_Fan: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Exhaust_Fan')

Exhaust_Fan_Disable_Command: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Exhaust_Fan_Disable_Command')

Exhaust_Fan_Enable_Command: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Exhaust_Fan_Enable_Command')

Eye_Wash_Station: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Eye_Wash_Station')

FCU: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#FCU')

```

```
Failure_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Failure_Alarm')  
  
Fan: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fan')  
  
Fan_Coil_Unit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fan_Coil_Unit')  
  
Fan_On_Off_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fan_On_Off_Status')  
  
Fan_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fan_Status')  
  
Fan_VFD: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fan_VFD')  
  
Fault_Reset_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fault_Reset_Command')  
  
Fault_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fault_Status')  
  
Field_Of_Play: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Field_Of_Play')  
  
Filter: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Filter')  
  
Filter_Differential_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Filter_Differential_Pressure_Sensor')  
  
Filter_Reset_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Filter_Reset_Command')  
  
Filter_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Filter_Status')  
  
Final_Filter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Final_Filter')  
  
Fire_Control_Panel: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fire_Control_Panel')  
  
Fire_Safety_Equipment: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fire_Safety_Equipment')  
  
Fire_Safety_System: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fire_Safety_System')  
  
Fire_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fire_Sensor')  
  
Fire_Zone: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fire_Zone')  
  
First_Aid_Kit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#First_Aid_Kit')
```

```
First_Aid_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#First_Aid_Room')  
  
Floor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Floor')  
  
Flow_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Flow_Sensor')  
  
Flow_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Flow_Setpoint')  
  
Fluid: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fluid')  
  
Food_Service_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Food_Service_Room')  
  
Formaldehyde_Level_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Formaldehyde_Level_Sensor')  
  
Freeze_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Freeze_Status')  
  
Freezer: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Freezer')  
  
Frequency_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Frequency_Command')  
  
Frequency_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Frequency_Sensor')  
  
Fresh_Air_Fan: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fresh_Air_Fan')  
  
Fresh_Air_Setpoint_Limit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fresh_Air_Setpoint_Limit')  
  
Frost: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Frost')  
  
Frost_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Frost_Sensor')  
  
Fuel_Oil: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fuel_Oil')  
  
Fume_Hood: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fume_Hood')  
  
Fume_Hood_Air_Flow_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Fume_Hood_Air_Flow_Sensor')  
  
Furniture: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Furniture')  
  
Gain_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Gain_Parameter')
```

```
Gas: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Gas')

Gas_Distribution: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Gas_Distribution')

Gas_Meter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Gas_Meter')

Gas_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Gas_Sensor')

Gas_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Gas_System')

Gas_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Gas_Valve')

Gasoline: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Gasoline')

Gatehouse: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Gatehouse')

Generator_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Generator_Room')

Glycol: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Glycol')

HVAC_Equipment: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#HVAC_Equipment')

HVAC_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#HVAC_System')

HVAC_Zone: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#HVAC_Zone')

HX: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#HX')

Hail: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hail')

Hail_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hail_Sensor')

Hallway: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hallway')

Hazardous_Materials_Storage: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Hazardous_Materials_Storage')

Heat_Exchanger: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heat_Exchanger')

Heat_Exchanger_Supply_Water_Temperature_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Heat_Exchanger_Supply_Water_Temperature_Sensor')
```

```

Heat_Exchanger_System_Enable_Status: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Heat_Exchanger_System_Enable_Status')

Heat_Recovery_Hot_Water_System: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Heat_Recovery_Hot_Water_System')

Heat_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heat_Sensor')

Heat_Wheel: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heat_Wheel')

Heat_Wheel_VFD: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heat_Wheel_VFD')

Heating_Coil: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heating_Coil')

Heating_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heating_Command')

Heating_Demand_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heating_Demand_Sensor')

Heating_Demand_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heating_Demand_Setpoint')

Heating_Discharge_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Heating_Discharge_Air_Flow_Setpoint')

Heating_Discharge_Air_Temperature_Deadband_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Heating_Discharge_Air_Temperature_Deadband_Setpoint')

Heating_Discharge_Air_Temperature_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Heating_Discharge_Air_Temperature_Integral_Time_Parameter')

Heating_Discharge_Air_Temperature_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Heating_Discharge_Air_Temperature_Proportional_Band_Parameter')

Heating_Start_Stop_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heating_Start_Stop_Status')

Heating_Supply_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Heating_Supply_Air_Flow_Setpoint')

Heating_Supply_Air_Temperature_Deadband_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Heating_Supply_Air_Temperature_Deadband_Setpoint')

Heating_Supply_Air_Temperature_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Heating_Supply_Air_Temperature_Integral_Time_Parameter')

```

```

Heating_Supply_Air_Temperature_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Heating_Supply_Air_Temperature_Proportional_Band_Parameter')

Heating_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Heating_Temperature_Setpoint')

Heating_Thermal_Power_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Heating_Thermal_Power_Sensor')

Heating_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Heating_Valve')

Heating_Ventilation_Air_Conditioning_System: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Heating_Ventilation_Air_Conditioning_System')

High_CO2_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#High_CO2_Alarm')

High_Discharge_Air_Temperature_Alarm: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#High_Discharge_Air_Temperature_Alarm')

High_Head_Pressure_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#High_Head_Pressure_Alarm')

High_Humidity_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#High_Humidity_Alarm')

High_Humidity_Alarm_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#High_Humidity_Alarm_Parameter')

High_Outside_Air_Lockout_Temperature_Differential_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#High_Outside_Air_Lockout_Temperature_Differential_Parameter')

High_Return_Air_Temperature_Alarm: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#High_Return_Air_Temperature_Alarm')

High_Static_Pressure_Cutout_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#High_Static_Pressure_Cutout_Setpoint_Limit')

High_Temperature_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#High_Temperature_Alarm')

High_Temperature_Alarm_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#High_Temperature_Alarm_Parameter')

High_Temperature_Hot_Water_Return_Temperature_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#High_Temperature_Hot_Water_Return_Temperature_Sensor')

High_Temperature_Hot_Water_Supply_Temperature_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#High_Temperature_Hot_Water_Supply_Temperature_Sensor')

```

```

Hold_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hold_Status')

Hospitality_Box: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hospitality_Box')

Hot_Box: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Box')

Hot_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water')

Hot_Water_Baseboard_Radiator: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Hot_Water_Baseboard_Radiator')

Hot_Water_Coil: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_Coil')

Hot_Water_Differential_Pressure_Deadband_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Hot_Water_Differential_Pressure_Deadband_Setpoint')

Hot_Water_Differential_Pressure_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Hot_Water_Differential_Pressure_Integral_Time_Parameter')

Hot_Water_Differential_Pressure_Load_Shed_Reset_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Hot_Water_Differential_Pressure_Load_Shed_Reset_Status')

Hot_Water_Differential_Pressure_Load_Shed_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Hot_Water_Differential_Pressure_Load_Shed_Status')

Hot_Water_Differential_Pressure_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Hot_Water_Differential_Pressure_Proportional_Band_Parameter')

Hot_Water_Differential_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Hot_Water_Differential_Pressure_Sensor')

Hot_Water_Differential_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Hot_Water_Differential_Pressure_Setpoint')

Hot_Water_Differential_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Hot_Water_Differential_Temperature_Sensor')

Hot_Water_Discharge_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Hot_Water_Discharge_Flow_Sensor')

Hot_Water_Discharge_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Hot_Water_Discharge_Flow_Setpoint')

Hot_Water_Discharge_Temperature_Load_Shed_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Hot_Water_Discharge_Temperature_Load_Shed_Status')

```



```
Hot_Water_Flow_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_Flow_Sensor')  
  
Hot_Water_Flow_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_Flow_Setpoint')  
  
Hot_Water_Loop: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_Loop')  
  
Hot_Water_Meter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_Meter')  
  
Hot_Water_Pump: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_Pump')  
  
Hot_Water_Radiator: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_Radiator')  
  
Hot_Water_Return_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Hot_Water_Return_Flow_Sensor')  
  
Hot_Water_Return_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Hot_Water_Return_Temperature_Sensor')  
  
Hot_Water_Static_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Hot_Water_Static_Pressure_Setpoint')  
  
Hot_Water_Supply_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Hot_Water_Supply_Flow_Sensor')  
  
Hot_Water_Supply_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Hot_Water_Supply_Flow_Setpoint')  
  
Hot_Water_Supply_Temperature_High_Reset_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Hot_Water_Supply_Temperature_High_Reset_Setpoint')  
  
Hot_Water_Supply_Temperature_Load_Shed_Status: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Hot_Water_Supply_Temperature_Load_Shed_Status')  
  
Hot_Water_Supply_Temperature_Low_Reset_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Hot_Water_Supply_Temperature_Low_Reset_Setpoint')  
  
Hot_Water_Supply_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Hot_Water_Supply_Temperature_Sensor')  
  
Hot_Water_System: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_System')  
  
Hot_Water_System_Enable_Command: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Hot_Water_System_Enable_Command')  
  
Hot_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Hot_Water_Temperature_Setpoint')
```



```
Hot_Water_Usage_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_Usage_Sensor')  
  
Hot_Water_Valve: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Hot_Water_Valve')  
  
Humidification_Start_Stop_Status: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Humidification_Start_Stop_Status')  
  
Humidifier: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Humidifier')  
  
Humidifier_Fault_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Humidifier_Fault_Status')  
  
Humidify_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Humidify_Command')  
  
Humidity_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Humidity_Alarm')  
  
Humidity_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Humidity_Parameter')  
  
Humidity_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Humidity_Sensor')  
  
Humidity_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Humidity_Setpoint')  
  
Humidity_Tolerance_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Humidity_Tolerance_Parameter')  
  
IDF: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#IDF')  
  
Ice: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Ice')  
  
Ice_Tank_Leaving_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Ice_Tank_Leaving_Water_Temperature_Sensor')  
  
Illuminance_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Illuminance_Sensor')  
  
Imbalance_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Imbalance_Sensor')  
  
Induction_Unit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Induction_Unit')  
  
Information_Area: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Information_Area')  
  
Inside_Face_Surface_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Inside_Face_Surface_Temperature_Sensor')  
  
Inside_Face_Surface_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Inside_Face_Surface_Temperature_Setpoint')
```

```
Intake_Air_Filter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Intake_Air_Filter')  
  
Intake_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Intake_Air_Temperature_Sensor')  
  
Integral_Gain_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Integral_Gain_Parameter')  
  
Integral_Time_Parameter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Integral_Time_Parameter')  
  
Intercom_Equipment: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Intercom_Equipment')  
  
Interface: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Interface')  
  
Intrusion_Detection_Equipment: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Intrusion_Detection_Equipment')  
  
Inverter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Inverter')  
  
Isolation_Valve: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Isolation_Valve')  
  
Janitor_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Janitor_Room')  
  
Jet_Nozzle_Air_Diffuser: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Jet_Nozzle_Air_Diffuser')  
  
Laboratory: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Laboratory')  
  
Laminar_Flow_Air_Diffuser: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Laminar_Flow_Air_Diffuser')  
  
Last_Fault_Code_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Last_Fault_Code_Status')  
  
Lead_Lag_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lead_Lag_Command')  
  
Lead_Lag_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lead_Lag_Status')  
  
Lead_On_Off_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lead_On_Off_Command')  
  
Leak_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Leak_Alarm')  
  
Leaving_Water: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Leaving_Water')
```

```

Leaving_Water_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Leaving_Water_Flow_Sensor')

Leaving_Water_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Leaving_Water_Flow_Setpoint')

Leaving_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Leaving_Water_Temperature_Sensor')

Leaving_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Leaving_Water_Temperature_Setpoint')

Library: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Library')

Lighting: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lighting')

Lighting_Equipment: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lighting_Equipment')

Lighting_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lighting_System')

Lighting_Zone: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lighting_Zone')

Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Limit')

Liquid: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Liquid')

Liquid_CO2: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Liquid_CO2')

Liquid_Detection_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Liquid_Detection_Alarm')

Load_Current_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Load_Current_Sensor')

Load_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Load_Parameter')

Load_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Load_Setpoint')

Load_Shed_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Load_Shed_Command')

Load_Shed_Differential_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Load_Shed_Differential_Pressure_Setpoint')

Load_Shed_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Load_Shed_Setpoint')

Load_Shed_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Load_Shed_Status')

```

```

Loading_Dock: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Loading_Dock')

Lobby: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lobby')

Locally_On_Off_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Locally_On_Off_Status')

Location: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Location')

Lockout_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lockout_Status')

Lockout_Temperature_Differential_Parameter: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Lockout_Temperature_Differential_Parameter')

Loop: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Loop')

Lounge: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Lounge')

Louver: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Louver')

Low_Freeze_Protect_Temperature_Parameter: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Low_Freeze_Protect_Temperature_Parameter')

Low_Humidity_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Low_Humidity_Alarm')

Low_Humidity_Alarm_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Low_Humidity_Alarm_Parameter')

Low_Outside_Air_Lockout_Temperature_Differential_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Low_Outside_Air_Lockout_Temperature_Differential_Parameter')

Low_Outside_Air_Temperature_Enable_Differential_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Low_Outside_Air_Temperature_Enable_Differential_Sensor')

Low_Outside_Air_Temperature_Enable_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Low_Outside_Air_Temperature_Enable_Setpoint')

Low_Return_Air_Temperature_Alarm: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Low_Return_Air_Temperature_Alarm')

Low_Suction_Pressure_Alarm: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Low_Suction_Pressure_Alarm')

Low_Temperature_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Low_Temperature_Alarm')

Low_Temperature_Alarm_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Low_Temperature_Alarm_Parameter')

Lowest_Exhaust_Air_Static_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Lowest_Exhaust_Air_Static_Pressure_Sensor')

```

```
Luminaire: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Luminaire')

Luminaire_Driver: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Luminaire_Driver')

Luminance_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Luminance_Alarm')

Luminance_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Luminance_Command')

Luminance_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Luminance_Sensor')

Luminance_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Luminance_Setpoint')

MAU: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#MAU')

MDF: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#MDF')

Mail_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Mail_Room')

Maintenance_Mode_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Maintenance_Mode_Command')

Maintenance_Required_Alarm: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Maintenance_Required_Alarm')

Majlis: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Majlis')

Makeup_Air_Unit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Makeup_Air_Unit')

Makeup_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Makeup_Water')

Makeup_Water_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Makeup_Water_Valve')

Manual_Auto_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Manual_Auto_Status')

Massage_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Massage_Room')

Max_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Max_Air_Flow_Setpoint_Limit')

Max_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Max_Air_Temperature_Setpoint')

Max_Chilled_Water_Differential_Pressure_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Max_Chilled_Water_Differential_Pressure_Setpoint_Limit')
```

```
Max_Cooling_Discharge_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Cooling_Discharge_Air_Flow_Setpoint_Limit')

Max_Cooling_Supply_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Cooling_Supply_Air_Flow_Setpoint_Limit')

Max_Discharge_Air_Static_Pressure_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Discharge_Air_Static_Pressure_Setpoint_Limit')

Max_Discharge_Air_Temperature_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Discharge_Air_Temperature_Setpoint_Limit')

Max_Frequency_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Frequency_Command')

Max_Heating_Discharge_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Heating_Discharge_Air_Flow_Setpoint_Limit')

Max_Heating_Supply_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Heating_Supply_Air_Flow_Setpoint_Limit')

Max_Hot_Water_Differential_Pressure_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Hot_Water_Differential_Pressure_Setpoint_Limit')

Max_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Limit')

Max_Load_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Load_Setpoint')

Max_Occupied_Cooling_Discharge_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Occupied_Cooling_Discharge_Air_Flow_Setpoint_Limit')

Max_Occupied_Cooling_Supply_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Occupied_Cooling_Supply_Air_Flow_Setpoint_Limit')

Max_Occupied_Heating_Discharge_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Occupied_Heating_Discharge_Air_Flow_Setpoint_Limit')

Max_Occupied_Heating_Supply_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Occupied_Heating_Supply_Air_Flow_Setpoint_Limit')

Max_Position_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Position_Setpoint_Limit')

Max_Speed_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Speed_Setpoint_Limit')

Max_Static_Pressure_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Static_Pressure_Setpoint_Limit')
```

```

Max_Supply_Air_Static_Pressure_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Supply_Air_Static_Pressure_Setpoint_Limit')

Max_Temperature_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Temperature_Setpoint_Limit')

Max_Unoccupied_Cooling_Discharge_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Unoccupied_Cooling_Discharge_Air_Flow_Setpoint_Limit')

Max_Unoccupied_Cooling_Supply_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Unoccupied_Cooling_Supply_Air_Flow_Setpoint_Limit')

Max_Unoccupied_Heating_Discharge_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Unoccupied_Heating_Discharge_Air_Flow_Setpoint_Limit')

Max_Unoccupied_Heating_Supply_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Unoccupied_Heating_Supply_Air_Flow_Setpoint_Limit')

Max_Water_Level_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Water_Level_Alarm')

Max_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Max_Water_Temperature_Setpoint')

Measurable: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Measurable')

Mechanical_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Mechanical_Room')

Media_Hot_Desk: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Media_Hot_Desk')

Media_Production_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Media_Production_Room')

Media_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Media_Room')

Medical_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Medical_Room')

Medium_Temperature_Hot_Water_Differential_Pressure_Load_Shed_Reset_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Medium_Temperature_Hot_Water_Differential_Pressure_Load_Shed_Reset_Status')

Medium_Temperature_Hot_Water_Differential_Pressure_Load_Shed_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Medium_Temperature_Hot_Water_Differential_Pressure_Load_Shed_Setpoint')

```



```

Medium_Temperature_Hot_Water_Differential_Pressure_Load_Shed_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Differential_Pressure_Load_Shed_Status')

Medium_Temperature_Hot_Water_Differential_Pressure_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Differential_Pressure_Sensor')

Medium_Temperature_Hot_Water_Differential_Pressure_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Differential_Pressure_Setpoint')

Medium_Temperature_Hot_Water_Discharge_Temperature_High_Reset_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Discharge_Temperature_High_Reset_Setpoint')

Medium_Temperature_Hot_Water_Discharge_Temperature_Low_Reset_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Discharge_Temperature_Low_Reset_Setpoint')

Medium_Temperature_Hot_Water_Return_Temperature_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Return_Temperature_Sensor')

Medium_Temperature_Hot_Water_Supply_Temperature_High_Reset_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Supply_Temperature_High_Reset_Setpoint')

Medium_Temperature_Hot_Water_Supply_Temperature_Load_Shed_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Supply_Temperature_Load_Shed_Setpoint')

Medium_Temperature_Hot_Water_Supply_Temperature_Load_Shed_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Supply_Temperature_Load_Shed_Status')

Medium_Temperature_Hot_Water_Supply_Temperature_Low_Reset_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Supply_Temperature_Low_Reset_Setpoint')

Medium_Temperature_Hot_Water_Supply_Temperature_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Medium_Temperature_Hot_Water_Supply_Temperature_Sensor')

Meter: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Meter')

Methane_Level_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Methane_Level_Sensor')

Min_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Min_Air_Flow_Setpoint_Limit')

Min_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Min_Air_Temperature_Setpoint')

```



```

Min_Chilled_Water_Differential_Pressure_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Chilled_Water_Differential_Pressure_Setpoint_Limit')

Min_Cooling_Discharge_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
/brickschema.org/schema/Brick#Min_Cooling_Discharge_Air_Flow_Setpoint_Limit')

Min_Cooling_Supply_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Min_Cooling_Supply_Air_Flow_Setpoint_Limit')

Min_Discharge_Air_Static_Pressure_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Discharge_Air_Static_Pressure_Setpoint_Limit')

Min_Discharge_Air_Temperature_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Min_Discharge_Air_Temperature_Setpoint_Limit')

Min_Fresh_Air_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Min_Fresh_Air_Setpoint_Limit')

Min_Heating_Discharge_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
/brickschema.org/schema/Brick#Min_Heating_Discharge_Air_Flow_Setpoint_Limit')

Min_Heating_Supply_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Min_Heating_Supply_Air_Flow_Setpoint_Limit')

Min_Hot_Water_Differential_Pressure_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Hot_Water_Differential_Pressure_Setpoint_Limit')

Min_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Min_Limit')

Min_Occupied_Cooling_Discharge_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Occupied_Cooling_Discharge_Air_Flow_Setpoint_Limit')

Min_Occupied_Cooling_Supply_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Occupied_Cooling_Supply_Air_Flow_Setpoint_Limit')

Min_Occupied_Heating_Discharge_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Occupied_Heating_Discharge_Air_Flow_Setpoint_Limit')

Min_Occupied_Heating_Supply_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Occupied_Heating_Supply_Air_Flow_Setpoint_Limit')

Min_Outside_Air_Flow_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Min_Outside_Air_Flow_Setpoint_Limit')

Min_Position_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Min_Position_Setpoint_Limit')

```

```

Min_Speed_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Min_Speed_Setpoint_Limit')

Min_Static_Pressure_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Min_Static_Pressure_Setpoint_Limit')

Min_Supply_Air_Static_Pressure_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://
/brickschema.org/schema/Brick#Min_Supply_Air_Static_Pressure_Setpoint_Limit')

Min_Temperature_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Min_Temperature_Setpoint_Limit')

Min_Unoccupied_Cooling_Discharge_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Unoccupied_Cooling_Discharge_Air_Flow_Setpoint_Limit')

Min_Unoccupied_Cooling_Supply_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Unoccupied_Cooling_Supply_Air_Flow_Setpoint_Limit')

Min_Unoccupied_Heating_Discharge_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Unoccupied_Heating_Discharge_Air_Flow_Setpoint_Limit')

Min_Unoccupied_Heating_Supply_Air_Flow_Setpoint_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Min_Unoccupied_Heating_Supply_Air_Flow_Setpoint_Limit')

Min_Water_Level_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Min_Water_Level_Alarm')

Min_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Min_Water_Temperature_Setpoint')

Mixed_Air: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Mixed_Air')

Mixed_Air_Filter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Mixed_Air_Filter')

Mixed_Air_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Mixed_Air_Flow_Sensor')

Mixed_Air_Humidity_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Mixed_Air_Humidity_Sensor')

Mixed_Air_Humidity_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Mixed_Air_Humidity_Setpoint')

Mixed_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Mixed_Air_Temperature_Sensor')

Mixed_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Mixed_Air_Temperature_Setpoint')

```

```
Mixed_Damper: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Mixed_Damper')  
  
Mode_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Mode_Command')  
  
Mode_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Mode_Status')  
  
Motion_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Motion_Sensor')  
  
Motor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Motor')  
  
Motor_Control_Center: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Motor_Control_Center')  
  
Motor_Current_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Motor_Current_Sensor')  
  
Motor_Direction_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Motor_Direction_Status')  
  
Motor_On_Off_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Motor_On_Off_Status')  
  
Motor_Speed_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Motor_Speed_Sensor')  
  
Motor_Torque_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Motor_Torque_Sensor')  
  
NO2_Level_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#NO2_Level_Sensor')  
  
NVR: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#NVR')  
  
Natural_Gas: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Natural_Gas')  
  
Natural_Gas_Boiler: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Natural_Gas_Boiler')  
  
Network_Video_Recorder: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Network_Video_Recorder')  
  
No_Water_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#No_Water_Alarm')  
  
Noncondensing_Natural_Gas_Boiler: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Noncondensing_Natural_Gas_Boiler')  
  
Occupancy_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Occupancy_Command')  
  
Occupancy_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Occupancy_Sensor')
```

```
Occupancy_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Occupancy_Status')  
  
Occupied_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Air_Temperature_Setpoint')  
  
Occupied_Cooling_Discharge_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Cooling_Discharge_Air_Flow_Setpoint')  
  
Occupied_Cooling_Supply_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Cooling_Supply_Air_Flow_Setpoint')  
  
Occupied_Cooling_Temperature_Deadband_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Occupied_Cooling_Temperature_Deadband_Setpoint')  
  
Occupied_Discharge_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Discharge_Air_Flow_Setpoint')  
  
Occupied_Discharge_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Discharge_Air_Temperature_Setpoint')  
  
Occupied_Heating_Discharge_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Heating_Discharge_Air_Flow_Setpoint')  
  
Occupied_Heating_Supply_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Heating_Supply_Air_Flow_Setpoint')  
  
Occupied_Heating_Temperature_Deadband_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/  
Brick#Occupied_Heating_Temperature_Deadband_Setpoint')  
  
Occupied_Mode_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Occupied_Mode_Status')  
  
Occupied_Return_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Return_Air_Temperature_Setpoint')  
  
Occupied_Room_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Room_Air_Temperature_Setpoint')  
  
Occupied_Supply_Air_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Supply_Air_Flow_Setpoint')  
  
Occupied_Supply_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Supply_Air_Temperature_Setpoint')  
  
Occupied_Zone_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Occupied_Zone_Air_Temperature_Setpoint')  
  
Off_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Off_Command')  
  
Off_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Off_Status')
```

```
Office: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Office')

Office_Kitchen: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Office_Kitchen')

Oil: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Oil')

On_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#On_Command')

On_Off_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#On_Off_Command')

On_Off_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#On_Off_Status')

On_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#On_Status')

On_Timer_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#On_Timer_Sensor')

Open_Close_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Open_Close_Status')

Open_Heating_Valve_Outside_Air_Temperature_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Open_Heating_Valve_Outside_Air_Temperature_Setpoint')

Open_Office: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Open_Office')

Operating_Mode_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Operating_Mode_Status')

Outdoor_Area: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outdoor_Area')

Output_Frequency_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Output_Frequency_Sensor')

Output_Voltage_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Output_Voltage_Sensor')

Outside: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside')

Outside_Air: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air')

Outside_Air_CO2_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_CO2_Sensor')

Outside_Air_CO_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_CO_Sensor')
```

Outside\_Air\_Dewpoint\_Sensor: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Dewpoint_Sensor')`

Outside\_Air\_Enthalpy\_Sensor: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Enthalpy_Sensor')`

Outside\_Air\_Flow\_Sensor: `URIRef` =  
`rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Flow_Sensor')`

Outside\_Air\_Flow\_Setpoint: `URIRef` =  
`rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Flow_Setpoint')`

Outside\_Air\_Grains\_Sensor: `URIRef` =  
`rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Grains_Sensor')`

Outside\_Air\_Humidity\_Sensor: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Humidity_Sensor')`

Outside\_Air\_Humidity\_Setpoint: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Humidity_Setpoint')`

Outside\_Air\_Lockout\_Temperature\_Differential\_Parameter: `URIRef` =  
`rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Lockout_Temperature_Differential_Parameter')`

Outside\_Air\_Lockout\_Temperature\_Setpoint: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Lockout_Temperature_Setpoint')`

Outside\_Air\_Temperature\_Enable\_Differential\_Sensor: `URIRef` =  
`rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Temperature_Enable_Differential_Sensor')`

Outside\_Air\_Temperature\_High\_Reset\_Setpoint: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Temperature_High_Reset_Setpoint')`

Outside\_Air\_Temperature\_Low\_Reset\_Setpoint: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Temperature_Low_Reset_Setpoint')`

Outside\_Air\_Temperature\_Sensor: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Temperature_Sensor')`

Outside\_Air\_Temperature\_Setpoint: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Temperature_Setpoint')`

Outside\_Air\_Wet\_Bulb\_Temperature\_Sensor: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Air_Wet_Bulb_Temperature_Sensor')`

Outside\_Damper: `URIRef` =  
`rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Damper')`

Outside\_Face\_Surface\_Temperature\_Sensor: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Face_Surface_Temperature_Sensor')`

Outside\_Face\_Surface\_Temperature\_Setpoint: `URIRef` = `rdflib.term.URIRef('https://brickschema.org/schema/Brick#Outside_Face_Surface_Temperature_Setpoint')`

```
Outside_Illuminance_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Outside_Illuminance_Sensor')

Overload_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Overload_Alarm')

Overridden_Off_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Overridden_Off_Status')

Overridden_On_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Overridden_On_Status')

Overridden_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Overridden_Status')

Override_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Override_Command')

Ozone_Level_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Ozone_Level_Sensor')

PAU: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#PAU')

PID_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PID_Parameter')

PIR_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PIR_Sensor')

PM10_Level_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PM10_Level_Sensor')

PM10_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PM10_Sensor')

PM1_Level_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PM1_Level_Sensor')

PM1_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PM1_Sensor')

PVT_Panel: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PVT_Panel')

PV_Array: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PV_Array')

PV_Current_Output_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PV_Current_Output_Sensor')

PV_Generation_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PV_Generation_System')

PV_Panel: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PV_Panel')
```



```

Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Parameter')

Parking_Level: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Parking_Level')

Parking_Space: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Parking_Space')

Parking_Structure: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Parking_Structure')

Particulate_Matter_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Particulate_Matter_Sensor')

Passive_Chilled_Beam: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Passive_Chilled_Beam')

Peak_Power_Demand_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Peak_Power_Demand_Sensor')

Photovoltaic_Array: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Photovoltaic_Array')

Photovoltaic_Current_Output_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Photovoltaic_Current_Output_Sensor')

Piezoelectric_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Piezoelectric_Sensor')

PlugStrip: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#PlugStrip')

Plumbing_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Plumbing_Room')

Point: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Point')

Portfolio: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Portfolio')

Position_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Position_Command')

Position_Limit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Position_Limit')

Position_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Position_Sensor')

Potable_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Potable_Water')

Power_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Power_Alarm')

```



```

Power_Loss_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Power_Loss_Alarm')

Power_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Power_Sensor')

Prayer_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Prayer_Room')

Pre_Filter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pre_Filter')

Pre_Filter_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pre_Filter_Status')

Preheat_Demand_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Preheat_Demand_Setpoint')

Preheat_Discharge_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Preheat_Discharge_Air_Temperature_Sensor')

Preheat_Hot_Water_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Preheat_Hot_Water_System')

Preheat_Hot_Water_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Preheat_Hot_Water_Valve')

Preheat_Supply_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Preheat_Supply_Air_Temperature_Sensor')

Pressure_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pressure_Alarm')

Pressure_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pressure_Sensor')

Pressure_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pressure_Setpoint')

Pressure_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pressure_Status')

Private_Office: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Private_Office')

Proportional_Band_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Proportional_Band_Parameter')

Proportional_Gain_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Proportional_Gain_Parameter')

Pump: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pump')

Pump_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pump_Command')

```

```
Pump_On_Off_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pump_On_Off_Status')  
  
Pump_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pump_Room')  
  
Pump_VFD: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Pump_VFD')  
  
Quantity: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Quantity')  
  
RC_Panel: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#RC_Panel')  
  
RTU: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#RTU')  
  
RVAV: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#RVAV')  
  
Radiant_Ceiling_Panel: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Radiant_Ceiling_Panel')  
  
Radiant_Panel: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Radiant_Panel')  
  
Radiant_Panel_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.  
org/schema/Brick#Radiant_Panel_Temperature_Sensor')  
  
Radiant_Panel_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Radiant_Panel_Temperature_Setpoint')  
  
Radiation_Hot_Water_System: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Radiation_Hot_Water_System')  
  
Radiator: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Radiator')  
  
Radioactivity_Concentration_Sensor: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Radioactivity_Concentration_Sensor')  
  
Radon_Concentration_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Radon_Concentration_Sensor')  
  
Rain_Duration_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Rain_Duration_Sensor')  
  
Rain_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Rain_Sensor')  
  
Rated_Speed_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Rated_Speed_Setpoint')  
  
Reactive_Power_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Reactive_Power_Sensor')  
  
Reception: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Reception')
```

```
Region: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Region')

Reheat_Hot_Water_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Reheat_Hot_Water_System')

Reheat_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Reheat_Valve')

Relative_Humidity_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Relative_Humidity_Sensor')

Relief_Damper: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Relief_Damper')

Relief_Fan: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Relief_Fan')

Remotely_On_Off_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Remotely_On_Off_Status')

Reset_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Reset_Command')

Reset_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Reset_Setpoint')

Rest_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Rest_Room')

Restroom: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Restroom')

Retail_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Retail_Room')

Return_Air: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Air')

Return_Air_CO2_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Air_CO2_Sensor')

Return_Air_CO2_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Air_CO2_Setpoint')

Return_Air_CO_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Air_CO_Sensor')

Return_Air_Dewpoint_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Return_Air_Dewpoint_Sensor')

Return_Air_Differential_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Air_Differential_Pressure_Sensor')

Return_Air_Differential_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Air_Differential_Pressure_Setpoint')
```

```
Return_Air_Enthalpy_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Return_Air_Enthalpy_Sensor')

Return_Air_Filter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Air_Filter')

Return_Air_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Air_Flow_Sensor')

Return_Air_Grains_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Air_Grains_Sensor')

Return_Air_Humidity_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Return_Air_Humidity_Sensor')

Return_Air_Humidity_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Return_Air_Humidity_Setpoint')

Return_Air_Plenum: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Air_Plenum')

Return_Air_Temperature_Alarm: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Return_Air_Temperature_Alarm')

Return_Air_Temperature_High_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Air_Temperature_High_Reset_Setpoint')

Return_Air_Temperature_Low_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Air_Temperature_Low_Reset_Setpoint')

Return_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Return_Air_Temperature_Sensor')

Return_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Return_Air_Temperature_Setpoint')

Return_Chilled_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Chilled_Water_Temperature_Setpoint')

Return_Condenser_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Condenser_Water')

Return_Condenser_Water_Flow_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Condenser_Water_Flow_Sensor')

Return_Condenser_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Condenser_Water_Temperature_Sensor')

Return_Condenser_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Condenser_Water_Temperature_Setpoint')

Return_Damper: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Damper')

Return_Fan: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Fan')
```

```

Return_Heating_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Heating_Valve')

Return_Hot_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Hot_Water')

Return_Hot_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Hot_Water_Temperature_Setpoint')

Return_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Water')

Return_Water_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Return_Water_Flow_Sensor')

Return_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Return_Water_Temperature_Sensor')

Return_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Return_Water_Temperature_Setpoint')

Riser: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Riser')

Rooftop: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Rooftop')

Rooftop_Unit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Rooftop_Unit')

Room: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Room')

Room_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Room_Air_Temperature_Setpoint')

Run_Enable_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Run_Enable_Command')

Run_Request_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Run_Request_Status')

Run_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Run_Status')

Run_Time_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Run_Time_Sensor')

Safety_Equipment: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Safety_Equipment')

Safety_Shower: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Safety_Shower')

Safety_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Safety_System')

Sash_Position_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Sash_Position_Sensor')

```

```
Schedule_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Schedule_Temperature_Setpoint')

Security_Equipment: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Security_Equipment')

Security_Service_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Security_Service_Room')

Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Sensor')

Server_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Server_Room')

Service_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Service_Room')

Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Setpoint')

Shading_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Shading_System')

Shared_Office: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Shared_Office')

Short_Cycle_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Short_Cycle_Alarm')

Shower: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Shower')

Site: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Site')

Smoke_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Smoke_Alarm')

Smoke_Detection_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Smoke_Detection_Alarm')

Solar_Azimuth_Angle_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Solar_Azimuth_Angle_Sensor')

Solar_Radiance_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Solar_Radiance_Sensor')

Solar_Thermal_Collector: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Solar_Thermal_Collector')

Solar_Zenith_Angle_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Solar_Zenith_Angle_Sensor')

Solid: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Solid')

Space: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Space')

Space_Heater: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Space_Heater')
```

```
Speed_Reset_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Speed_Reset_Command')  
  
Speed_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Speed_Sensor')  
  
Speed_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Speed_Setpoint')  
  
Speed_Setpoint_Limit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Speed_Setpoint_Limit')  
  
Speed_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Speed_Status')  
  
Sports_Service_Room: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Sports_Service_Room')  
  
Stage_Enable_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Stage_Enable_Command')  
  
Stage_Riser: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Stage_Riser')  
  
Stages_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Stages_Status')  
  
Staircase: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Staircase')  
  
Standby_CRAC: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Standby_CRAC')  
  
Standby_Fan: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Standby_Fan')  
  
Standby_Glycool_Unit_On_Off_Status: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Standby_Glycool_Unit_On_Off_Status')  
  
Standby_Load_Shed_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Standby_Load_Shed_Command')  
  
Standby_Unit_On_Off_Status: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Standby_Unit_On_Off_Status')  
  
Start_Stop_Command: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Start_Stop_Command')  
  
Start_Stop_Status: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Start_Stop_Status')  
  
Static_Pressure_Deadband_Setpoint: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Static_Pressure_Deadband_Setpoint')  
  
Static_Pressure_Integral_Time_Parameter: URIRef = rdflib.term.URIRef('https://  
brickschema.org/schema/Brick#Static_Pressure_Integral_Time_Parameter')
```



```
Static_Pressure_Proportional_Band_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Static_Pressure_Proportional_Band_Parameter')

Static_Pressure_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Static_Pressure_Sensor')

Static_Pressure_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Static_Pressure_Setpoint')

Static_Pressure_Setpoint_Limit: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Static_Pressure_Setpoint_Limit')

Static_Pressure_Step_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Static_Pressure_Step_Parameter')

Status: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Status')

Steam: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Steam')

Steam_Baseboard_Radiator: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Steam_Baseboard_Radiator')

Steam_Distribution: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Steam_Distribution')

Steam_On_Off_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Steam_On_Off_Command')

Steam_Radiator: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Steam_Radiator')

Steam_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Steam_System')

Steam_Usage_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Steam_Usage_Sensor')

Steam_Valve: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Steam_Valve')

Step_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Step_Parameter')

Storage_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Storage_Room')

Storey: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Storey')

Studio: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Studio')

Substance: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Substance')

Supply_Air: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air')
```



```
Supply_Air_Differential_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Differential_Pressure_Sensor')

Supply_Air_Differential_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Differential_Pressure_Setpoint')

Supply_Air_Duct_Pressure_Status: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Duct_Pressure_Status')

Supply_Air_Flow_Demand_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Flow_Demand_Setpoint')

Supply_Air_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Flow_Sensor')

Supply_Air_Flow_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Flow_Setpoint')

Supply_Air_Humidity_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Humidity_Sensor')

Supply_Air_Humidity_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Humidity_Setpoint')

Supply_Air_Integral_Gain_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Integral_Gain_Parameter')

Supply_Air_Plenum: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Plenum')

Supply_Air_Proportional_Gain_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Proportional_Gain_Parameter')

Supply_Air_Static_Pressure_Deadband_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Static_Pressure_Deadband_Setpoint')

Supply_Air_Static_Pressure_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Static_Pressure_Integral_Time_Parameter')

Supply_Air_Static_Pressure_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Static_Pressure_Proportional_Band_Parameter')

Supply_Air_Static_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Static_Pressure_Sensor')

Supply_Air_Static_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Static_Pressure_Setpoint')

Supply_Air_Temperature_Alarm: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Temperature_Alarm')

Supply_Air_Temperature_Deadband_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Temperature_Deadband_Setpoint')
```

```
Supply_Air_Temperature_High_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Temperature_High_Reset_Setpoint')

Supply_Air_Temperature_Low_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Temperature_Low_Reset_Setpoint')

Supply_Air_Temperature_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Temperature_Proportional_Band_Parameter')

Supply_Air_Temperature_Reset_Differential_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Temperature_Reset_Differential_Setpoint')

Supply_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Temperature_Sensor')

Supply_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Temperature_Setpoint')

Supply_Air_Temperature_Step_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Temperature_Step_Parameter')

Supply_Air_Velocity_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Air_Velocity_Pressure_Sensor')

Supply_Chilled_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Chilled_Water')

Supply_Chilled_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Chilled_Water_Temperature_Setpoint')

Supply_Condenser_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Condenser_Water')

Supply_Condenser_Water_Flow_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Condenser_Water_Flow_Sensor')

Supply_Condenser_Water_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Condenser_Water_Temperature_Sensor')

Supply_Condenser_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Condenser_Water_Temperature_Setpoint')

Supply_Fan: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Fan')

Supply_Hot_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Hot_Water')

Supply_Hot_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Hot_Water_Temperature_Setpoint')

Supply_Water: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Water')
```

```

Supply_Water_Differential_Pressure_Deadband_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Supply_Water_Differential_Pressure_Deadband_Setpoint')

Supply_Water_Differential_Pressure_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Supply_Water_Differential_Pressure_Integral_Time_Parameter')

Supply_Water_Differential_Pressure_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Supply_Water_Differential_Pressure_Proportional_Band_Parameter')

Supply_Water_Flow_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Supply_Water_Flow_Sensor')

Supply_Water_Flow_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Supply_Water_Flow_Setpoint')

Supply_Water_Temperature_Alarm: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Supply_Water_Temperature_Alarm')

Supply_Water_Temperature_Deadband_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Supply_Water_Temperature_Deadband_Setpoint')

Supply_Water_Temperature_Integral_Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Supply_Water_Temperature_Integral_Time_Parameter')

Supply_Water_Temperature_Proportional_Band_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Supply_Water_Temperature_Proportional_Band_Parameter')

Supply_Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Supply_Water_Temperature_Setpoint')

Surveillance_Camera: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Surveillance_Camera')

Switch: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Switch')

Switch_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Switch_Room')

Switchgear: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Switchgear')

System: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#System')

System_Enable_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#System_Enable_Command')

System_Shutdown_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#System_Shutdown_Status')

System_Status: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#System_Status')

```

```

TABS_Panel: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#TABS_Panel')

TETRA_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#TETRA_Room')

TVOC_Level_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#TVOC_Level_Sensor')

TVOC_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#TVOC_Sensor')

Team_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Team_Room')

Telecom_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Telecom_Room')

Temperature_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Temperature_Alarm')

Temperature_Deadband_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Temperature_Deadband_Setpoint')

Temperature_Differential_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Temperature_Differential_Reset_Setpoint')

Temperature_High_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Temperature_High_Reset_Setpoint')

Temperature_Low_Reset_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Temperature_Low_Reset_Setpoint')

Temperature_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Temperature_Parameter')

Temperature_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Temperature_Sensor')

Temperature_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Temperature_Setpoint')

Temperature_Step_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Temperature_Step_Parameter')

Temperature_Tolerance_Parameter: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Temperature_Tolerance_Parameter')

Temporary_Occupancy_Status: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Temporary_Occupancy_Status')

Terminal_Unit: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Terminal_Unit')

Thermal_Power_Meter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Thermal_Power_Meter')

```

```

Thermal_Power_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Thermal_Power_Sensor')

Thermally_Activated_Building_System_Panel: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Thermally_Activated_Building_System_Panel')

Thermostat: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Thermostat')

Ticketing_Booth: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Ticketing_Booth')

Time_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Time_Parameter')

Time_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Time_Setpoint')

Tolerance_Parameter: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Tolerance_Parameter')

Torque_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Torque_Sensor')

Touchpanel: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Touchpanel')

Trace_Heat_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Trace_Heat_Sensor')

Transformer: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Transformer')

Transformer_Room: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Transformer_Room')

Tunnel: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Tunnel')

Underfloor_Air_Plenum: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Underfloor_Air_Plenum')

Underfloor_Air_Plenum_Static_Pressure_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Underfloor_Air_Plenum_Static_Pressure_Sensor')

Underfloor_Air_Plenum_Static_Pressure_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/
Brick#Underfloor_Air_Plenum_Static_Pressure_Setpoint')

Underfloor_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Underfloor_Air_Temperature_Sensor')

Unit_Failure_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unit_Failure_Alarm')

Unoccupied_Air_Temperature_Cooling_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Unoccupied_Air_Temperature_Cooling_Setpoint')

```

```
Unoccupied_Air_Temperature_Heating_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unoccupied_Air_Temperature_Heating_Setpoint')

Unoccupied_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unoccupied_Air_Temperature_Setpoint')

Unoccupied_Cooling_Discharge_Air_Flow_Setpoint: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unoccupied_Cooling_Discharge_Air_Flow_Setpoint')

Unoccupied_Discharge_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unoccupied_Discharge_Air_Temperature_Setpoint')

Unoccupied_Load_Shed_Command: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unoccupied_Load_Shed_Command')

Unoccupied_Return_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unoccupied_Return_Air_Temperature_Setpoint')

Unoccupied_Room_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unoccupied_Room_Air_Temperature_Setpoint')

Unoccupied_Supply_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unoccupied_Supply_Air_Temperature_Setpoint')

Unoccupied_Zone_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Unoccupied_Zone_Air_Temperature_Setpoint')

Usage_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Usage_Sensor')

VAV: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#VAV')

VFD: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#VFD')

VFD_Enable_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#VFD_Enable_Command')

Valve: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Valve')

Valve_Command: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Valve_Command')

Valve_Position_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Valve_Position_Sensor')

Variable_Air_Volume_Box: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Variable_Air_Volume_Box')

Variable_Air_Volume_Box_With_Reheat: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Variable_Air_Volume_Box_With_Reheat')

Variable_Frequency_Drive: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Variable_Frequency_Drive')

Velocity_Pressure_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Velocity_Pressure_Sensor')
```

```
Velocity_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Velocity_Pressure_Setpoint')

Vent_Operating_Mode_Status: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Vent_Operating_Mode_Status')

Ventilation_Air_Flow_Ratio_Limit: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Ventilation_Air_Flow_Ratio_Limit')

Ventilation_Air_System: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Ventilation_Air_System')

Vertical_Space: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Vertical_Space')

Video_Intercom: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Video_Intercom')

Video_Surveillance_Equipment: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Video_Surveillance_Equipment')

Visitor_Lobby: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Visitor_Lobby')

Voltage_Imbalance_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Voltage_Imbalance_Sensor')

Voltage_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Voltage_Sensor')

Wardrobe: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Wardrobe')

Warm_Cool_Adjust_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Warm_Cool_Adjust_Sensor')

Warmest_Zone_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Warmest_Zone_Air_Temperature_Sensor')

Waste_Storage: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Waste_Storage')

Water: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water')

Water_Alarm: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Alarm')

Water_Differential_Pressure_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Water_Differential_Pressure_Setpoint')

Water_Differential_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Water_Differential_Temperature_Sensor')

Water_Differential_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Water_Differential_Temperature_Setpoint')
```



```
Water_Distribution: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Distribution')  
  
Water_Flow_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Flow_Sensor')  
  
Water_Flow_Setpoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Flow_Setpoint')  
  
Water_Heater: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Heater')  
  
Water_Level_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Level_Alarm')  
  
Water_Level_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Level_Sensor')  
  
Water_Loop: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Loop')  
  
Water_Loss_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Loss_Alarm')  
  
Water_Meter: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Meter')  
  
Water_Pump: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Pump')  
  
Water_System: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_System')  
  
Water_Tank: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Tank')  
  
Water_Temperature_Alarm: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Temperature_Alarm')  
  
Water_Temperature_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Temperature_Sensor')  
  
Water_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/  
schema/Brick#Water_Temperature_Setpoint')  
  
Water_Usage_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Usage_Sensor')  
  
Water_Valve: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Water_Valve')  
  
Weather_Station: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Weather_Station')  
  
Wind_Direction_Sensor: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Wind_Direction_Sensor')
```



```

Wind_Speed_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Wind_Speed_Sensor')

Wing: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Wing')

Workshop: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Workshop')

Zone: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#Zone')

Zone_Air: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Zone_Air')

Zone_Air_Cooling_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Zone_Air_Cooling_Temperature_Setpoint')

Zone_Air_Dewpoint_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Zone_Air_Dewpoint_Sensor')

Zone_Air_Heating_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Zone_Air_Heating_Temperature_Setpoint')

Zone_Air_Humidity_Sensor: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#Zone_Air_Humidity_Sensor')

Zone_Air_Humidity_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Zone_Air_Humidity_Setpoint')

Zone_Air_Temperature_Sensor: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#Zone_Air_Temperature_Sensor')

Zone_Air_Temperature_Setpoint: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Zone_Air_Temperature_Setpoint')

Zone_Standby_Load_Shed_Command: URIRef = rdflib.term.URIRef('https://brickschema.
org/schema/Brick#Zone_Standby_Load_Shed_Command')

Zone_Unoccupied_Load_Shed_Command: URIRef = rdflib.term.URIRef('https://
brickschema.org/schema/Brick#Zone_Unoccupied_Load_Shed_Command')

aggregate: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#aggregate')

area: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#area')

azimuth: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#azimuth')

buildingPrimaryFunction: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#buildingPrimaryFunction')

buildingThermalTransmittance: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#buildingThermalTransmittance')

conversionEfficiency: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#conversionEfficiency')

```

```
coolingCapacity: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#coolingCapacity')  
  
coordinates: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#coordinates')  
  
currentFlowType: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#currentFlowType')  
  
electricalPhaseCount: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#electricalPhaseCount')  
  
electricalPhases: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#electricalPhases')  
  
feeds: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#feeds')  
  
feedsAir: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#feedsAir')  
  
grossArea: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#grossArea')  
  
hasAddress: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasAddress')  
  
hasAssociatedTag: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasAssociatedTag')  
  
hasInputSubstance: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasInputSubstance')  
  
hasLocation: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasLocation')  
  
hasOutputSubstance: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasOutputSubstance')  
  
hasPart: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasPart')  
  
hasPoint: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasPoint')  
  
hasQUDTReference: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasQUDTReference')  
  
hasTag: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasTag')  
  
hasTimeseriesId: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasTimeseriesId')  
  
hasUnit: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#hasUnit')  
  
isAssociatedWith: URIRef =  
rdflib.term.URIRef('https://brickschema.org/schema/Brick#isAssociatedWith')
```

```
isFedBy: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#isFedBy')

isLocationOf: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#isLocationOf')

isMeasuredBy: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#isMeasuredBy')

isPartOf: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#isPartOf')

isPointOf: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#isPointOf')

isRegulatedBy: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#isRegulatedBy')

isTagOf: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#isTagOf')

latitude: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#latitude')

longitude: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#longitude')

measuredModuleConversionEfficiency: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#measuredModuleConversionEfficiency')

measuredPowerOutput: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#measuredPowerOutput')

measures: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#measures')

netArea: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#netArea')

operationalStage: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#operationalStage')

operationalStageCount: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#operationalStageCount')

panelArea: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#panelArea')

powerComplexity: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#powerComplexity')

powerFlow: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#powerFlow')

ratedModuleConversionEfficiency: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#ratedModuleConversionEfficiency')
```

```

ratedPowerOutput: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#ratedPowerOutput')

regulates: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#regulates')

storedAt: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#storedAt')

temperatureCoefficientofPmax: URIRef = rdflib.term.URIRef('https://brickschema.org/
schema/Brick#temperatureCoefficientofPmax')

thermalTransmittance: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#thermalTransmittance')

tilt: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#tilt')

timeseries: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#timeseries')

value: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#value')

volume: URIRef = rdflib.term.URIRef('https://brickschema.org/schema/Brick#volume')

yearBuilt: URIRef =
rdflib.term.URIRef('https://brickschema.org/schema/Brick#yearBuilt')

```

class rdflib.CSVW

Bases: *DefinedNamespace*

CSVW Namespace Vocabulary Terms

This document describes the RDFS vocabulary description used in the Metadata Vocabulary for Tabular Data [[tabular-metadata]] along with the default JSON-LD Context.

Generated from: <http://www.w3.org/ns/csvw> Date: 2020-05-26 14:19:58.184766

Cell: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#Cell')

Column: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#Column')

Datatype: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#Datatype')

Dialect: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#Dialect')

Direction: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#Direction')

ForeignKey: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#ForeignKey')

JSON: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#JSON')

NumericFormat: *URIRef* =  
rdflib.term.URIRef('http://www.w3.org/ns/csvw#NumericFormat')

Row: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#Row')

Schema: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#Schema')

Table: *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/csvw#Table')

```

TableGroup: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#TableGroup')

TableReference: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#TableReference')

Transformation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#Transformation')

aboutUrl: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#aboutUrl')

auto: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#auto')

base: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#base')

column: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#column')

columnReference: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#columnReference')

commentPrefix: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#commentPrefix')

csvEncodedTabularData: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#csvEncodedTabularData')

datatype: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#datatype')

decimalChar: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#decimalChar')

default: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#default')

delimiter: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#delimiter')

describes: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#describes')

dialect: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#dialect')

doubleQuote: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#doubleQuote')

encoding: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#encoding')

foreignKey: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#foreignKey')

format: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#format')

groupChar: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#groupChar')

header: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#header')

headerRowCount: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#headerRowCount')

inherit: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#inherit')

lang: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#lang')

length: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#length')

```

```
lineTerminators: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/csvw#lineTerminators')  
  
ltr: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#ltr')  
  
maxExclusive: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#maxExclusive')  
maxInclusive: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#maxInclusive')  
maxLength: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#maxLength')  
minExclusive: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#minExclusive')  
minInclusive: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#minInclusive')  
minLength: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#minLength')  
name: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#name')  
note: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#note')  
null: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#null')  
ordered: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#ordered')  
pattern: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#pattern')  
primaryKey: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#primaryKey')  
propertyUrl: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#propertyUrl')  
quoteChar: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#quoteChar')  
reference: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#reference')  
  
referencedRow: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/csvw#referencedRow')  
  
required: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#required')  
resource: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#resource')  
row: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#row')  
rowTitle: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#rowTitle')  
rownum: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#rownum')  
rtl: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#rtl')  
  
schemaReference: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/csvw#schemaReference')  
  
scriptFormat: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#scriptFormat')  
separator: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#separator')  
  
skipBlankRows: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/csvw#skipBlankRows')
```

```

skipColumns: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#skipColumns')

skipInitialSpace: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#skipInitialSpace')

skipRows: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#skipRows')

source: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#source')

suppressOutput: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#suppressOutput')

table: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#table')

tableDirection: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#tableDirection')

tableSchema: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#tableSchema')

tabularMetadata: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#tabularMetadata')

targetFormat: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#targetFormat')

textDirection: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#textDirection')

title: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#title')

transformations: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/csvw#transformations')

trim: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#trim')

uriTemplate: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#uriTemplate')

url: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#url')

valueUrl: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#valueUrl')

virtual: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/csvw#virtual')

```

```
class rdflib.ConjunctiveGraph(store='default', identifier=None, default_graph_base=None)
```

Bases: *Graph*

A ConjunctiveGraph is an (unnamed) aggregation of all the named graphs in a store.

It has a default graph, whose name is associated with the graph throughout its life. `__init__()` can take an identifier to use as the name of this default graph or it will assign a BNode.

All methods that add triples work against this default graph.

All queries are carried out against the union of all graphs.

#### Parameters

- **store** (*Union*[*Store*, *str*]) –
- **identifier** (*Union*[*IdentifiedNode*, *str*, *None*]) –
- **default\_graph\_base** (*Optional*[*str*]) –

```
__annotations__ = {'__identifier': '_ContextIdentifierType', '__store': 'Store',  
'default_context': '_ContextType'}
```

```
__contains__(triple_or_quad)
```

Support for ‘triple/quad in graph’ syntax

**Parameters**

**triple\_or\_quad** (Union[Tuple[Optional[Node], Optional[Node], Optional[Node]],  
Tuple[Optional[Node], Optional[Node], Optional[Node], Optional[Graph]]) –

**Return type**

bool

```
__init__(store='default', identifier=None, default_graph_base=None)
```

**Parameters**

- **store** (Union[Store, str]) –
- **identifier** (Union[IdentifiedNode, str, None]) –
- **default\_graph\_base** (Optional[str]) –

```
__len__()
```

Number of triples in the entire conjunctive graph

**Return type**

int

```
__module__ = 'rdflib.graph'
```

```
__reduce__()
```

Helper for pickle.

**Return type**

Tuple[Type[Graph], Tuple[Store, IdentifiedNode]]

```
__str__()
```

Return str(self).

**Return type**

str

```
add(triple_or_quad)
```

Add a triple or quad to the store.

if a triple is given it is added to the default context

**Parameters**

- **self** (TypeVar(\_ConjunctiveGraphT, bound= ConjunctiveGraph)) –
- **triple\_or\_quad** (Union[Tuple[Node, Node, Node], Tuple[Node, Node, Node,  
Optional[Graph]]) –

**Return type**

TypeVar(\_ConjunctiveGraphT, bound= ConjunctiveGraph)

```
addN(quads)
```

Add a sequence of triples with context

**Parameters**

- **self** (TypeVar(\_ConjunctiveGraphT, bound= ConjunctiveGraph)) –



- **quads** (`Iterable[Tuple[Node, Node, Node, Graph]]`) –

**Return type**

`TypeVar(_ConjunctiveGraphT, bound= ConjunctiveGraph)`

**context\_id**(*uri*, *context\_id=None*)

URI#context

**Parameters**

- **uri** (`str`) –
- **context\_id** (`Optional[str]`) –

**Return type**

`URIRef`

**contexts**(*triple=None*)

Iterate over all contexts in the graph

If triple is specified, iterate over all contexts the triple is in.

**Parameters**

**triple** (`Optional[Tuple[Node, Node, Node]]`) –

**Return type**

`Generator[Graph, None, None]`

**default\_context:** `Graph`

**get\_context**(*identifier*, *quoted=False*, *base=None*)

Return a context graph for the given identifier

identifier must be a URIRef or BNode.

**Parameters**

- **identifier** (`Union[IdentifiedNode, str, None]`) –
- **quoted** (`bool`) –
- **base** (`Optional[str]`) –

**Return type**

`Graph`

**get\_graph**(*identifier*)

Returns the graph identified by given identifier

**Parameters**

**identifier** (`IdentifiedNode`) –

**Return type**

`Optional[Graph]`

**parse**(*source=None*, *publicID=None*, *format=None*, *location=None*, *file=None*, *data=None*, *\*\*args*)

Parse source adding the resulting triples to its own context (sub graph of this graph).

See `rdflib.graph.Graph.parse()` for documentation on arguments.

**Returns**

The graph into which the source was parsed. In the case of n3 it returns the root context.

**Caution:** This method can access directly or indirectly requested network or file resources, for example, when parsing JSON-LD documents with `@context` directives that point to a network location.

When processing untrusted or potentially malicious documents, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

#### Parameters

- **source** (`Union[IO[bytes], TextIO, InputSource, str, bytes, PurePath, None]`) –
- **publicID** (`Optional[str]`) –
- **format** (`Optional[str]`) –
- **location** (`Optional[str]`) –
- **file** (`Union[BinaryIO, TextIO, None]`) –
- **data** (`Union[str, bytes, None]`) –
- **args** (`Any`) –

#### Return type

*Graph*

**quads**(*triple\_or\_quad=None*)

Iterate over all the quads in the entire conjunctive graph

#### Parameters

**triple\_or\_quad** (`Union[Tuple[Optional[Node], Optional[Node], Optional[Node]], Tuple[Optional[Node], Optional[Node], Optional[Node], Optional[Graph]], None]`) –

#### Return type

`Generator[Tuple[Node, Node, Node, Optional[Graph]], None, None]`

**remove**(*triple\_or\_quad*)

Removes a triple or quads

if a triple is given it is removed from all contexts

a quad is removed from the given context only

#### Parameters

- **self** (`TypeVar(_ConjunctiveGraphT, bound= ConjunctiveGraph)`) –
- **triple\_or\_quad** (`Union[Tuple[Node, Node, Node], Tuple[Node, Node, Node, Optional[Graph]]]`) –

#### Return type

`TypeVar(_ConjunctiveGraphT, bound= ConjunctiveGraph)`

**remove\_context**(*context*)

Removes the given context from the graph

#### Parameters

**context** (*Graph*) –

**Return type**`None`**triples**(*triple\_or\_quad*, *context=None*)

Iterate over all the triples in the entire conjunctive graph

For legacy reasons, this can take the context to query either as a fourth element of the quad, or as the explicit context keyword parameter. The kw param takes precedence.

**Parameters**

- **triple\_or\_quad** (`Union[Tuple[Optional[Node], Union[Path, Node, None], Optional[Node]], Tuple[Optional[Node], Union[Path, Node, None], Optional[Node], Optional[Graph]]]`) –
- **context** (`Optional[Graph]`) –

**Return type**`Generator[Union[Tuple[Node, Node, Node], Tuple[Node, Path, Node]], None, None]`**triples\_choices**(*triple*, *context=None*)

Iterate over all the triples in the entire conjunctive graph

**Parameters**

- **triple** (`Union[Tuple[List[Node], Node, Node], Tuple[Node, List[Node], Node], Tuple[Node, Node, List[Node]]]`) –
- **context** (`Optional[Graph]`) –

**Return type**`Generator[Tuple[Node, Node, Node], None, None]`**class** `rdflib.DC`Bases: `DefinedNamespace`

Dublin Core Metadata Element Set, Version 1.1

Generated from: [https://www.dublincore.org/specifications/dublin-core/dcmi-terms/dublin\\_core\\_elements.ttl](https://www.dublincore.org/specifications/dublin-core/dcmi-terms/dublin_core_elements.ttl)

Date: 2020-05-26 14:19:58.671906

**contributor:** `URIRef` =`rdflib.term.URIRef('http://purl.org/dc/elements/1.1/contributor')`**coverage:** `URIRef` = `rdflib.term.URIRef('http://purl.org/dc/elements/1.1/coverage')`**creator:** `URIRef` = `rdflib.term.URIRef('http://purl.org/dc/elements/1.1/creator')`**date:** `URIRef` = `rdflib.term.URIRef('http://purl.org/dc/elements/1.1/date')`**description:** `URIRef` =`rdflib.term.URIRef('http://purl.org/dc/elements/1.1/description')`**format:** `URIRef` = `rdflib.term.URIRef('http://purl.org/dc/elements/1.1/format')`**identifier:** `URIRef` =`rdflib.term.URIRef('http://purl.org/dc/elements/1.1/identifier')`**language:** `URIRef` = `rdflib.term.URIRef('http://purl.org/dc/elements/1.1/language')`**publisher:** `URIRef` = `rdflib.term.URIRef('http://purl.org/dc/elements/1.1/publisher')`

```
relation: URIRef = rdflib.term.URIRef('http://purl.org/dc/elements/1.1/relation')
rights: URIRef = rdflib.term.URIRef('http://purl.org/dc/elements/1.1/rights')
source: URIRef = rdflib.term.URIRef('http://purl.org/dc/elements/1.1/source')
subject: URIRef = rdflib.term.URIRef('http://purl.org/dc/elements/1.1/subject')
title: URIRef = rdflib.term.URIRef('http://purl.org/dc/elements/1.1/title')
type: URIRef = rdflib.term.URIRef('http://purl.org/dc/elements/1.1/type')
```

```
class rdflib.DCAT
```

```
    Bases: DefinedNamespace
```

```
    The data catalog vocabulary
```

DCAT is an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web. By using DCAT to describe datasets in data catalogs, publishers increase discoverability and enable applications easily to consume metadata from multiple catalogs. It further enables decentralized publishing of catalogs and facilitates federated dataset search across sites. Aggregated DCAT metadata can serve as a manifest file to facilitate digital preservation. DCAT is defined at <http://www.w3.org/TR/vocab-dcat/>. Any variance between that normative document and this schema is an error in this schema.

Generated from: <https://www.w3.org/ns/dcat2.ttl> Date: 2020-05-26 14:19:59.985854

```
Catalog: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#Catalog')
```

```
CatalogRecord: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#CatalogRecord')
```

```
DataService: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#DataService')
```

```
Dataset: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#Dataset')
```

```
Distribution: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#Distribution')
```

```
Relationship: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#Relationship')
```

```
Resource: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#Resource')
```

```
Role: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#Role')
```

```
accessService: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#accessService')
```

```
accessURL: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#accessURL')
```

```
bbox: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#bbox')
```

```
byteSize: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#byteSize')
```

```
catalog: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#catalog')
```

```
centroid: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#centroid')
```

```
compressFormat: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#compressFormat')
```

```
contactPoint: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#contactPoint')
```

```

dataset: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#dataset')
distribution: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#distribution')
downloadURL: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#downloadURL')
endDate: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#endDate')
endpointDescription: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#endpointDescription')
endpointURL: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#endpointURL')
hadRole: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#hadRole')
keyword: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#keyword')
landingPage: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#landingPage')
mediaType: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#mediaType')
packageFormat: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#packageFormat')
qualifiedRelation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#qualifiedRelation')
record: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#record')
servesDataset: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#servesDataset')
service: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#service')
spatialResolutionInMeters: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#spatialResolutionInMeters')
startDate: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#startDate')
temporalResolution: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#temporalResolution')
theme: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/dcat#theme')
themeTaxonomy: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/dcat#themeTaxonomy')

```

```
class rdflib.DCMITYPE
```

```
    Bases: DefinedNamespace
```

```
    DCMI Type Vocabulary
```

```
    Generated from: https://www.dublincore.org/specifications/dublin-core/dcmi-terms/dublin\_core\_type.ttl Date:
    2020-05-26 14:19:59.084150
```

```
    Collection: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/Collection')
```

```
    Dataset: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/Dataset')
```

```
Event: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/Event')
Image: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/Image')
InteractiveResource: URIRef =
rdflib.term.URIRef('http://purl.org/dc/dcmitype/InteractiveResource')
MovingImage: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/MovingImage')
PhysicalObject: URIRef =
rdflib.term.URIRef('http://purl.org/dc/dcmitype/PhysicalObject')
Service: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/Service')
Software: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/Software')
Sound: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/Sound')
StillImage: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/StillImage')
Text: URIRef = rdflib.term.URIRef('http://purl.org/dc/dcmitype/Text')
```

```
class rdflib.DCTERMS
```

```
    Bases: DefinedNamespace
```

```
    DCMI Metadata Terms - other
```

```
    Generated from: http://www.dublincore.org/specifications/dublin-core/dcmi-terms/dublin\_core\_terms.ttl
```

```
    Date: 2020-05-26 14:20:00.590514
```

```
    Agent: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/Agent')
```

```
    AgentClass: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/AgentClass')
```

```
    BibliographicResource: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/BibliographicResource')
```

```
    Box: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/Box')
```

```
    DCMIType: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/DCMIType')
```

```
    DDC: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/DDC')
```

```
    FileFormat: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/FileFormat')
```

```
    Frequency: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/Frequency')
```

```
    IMT: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/IMT')
```

```
    ISO3166: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/ISO3166')
```

```
    Jurisdiction: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/Jurisdiction')
```

```
    LCC: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/LCC')
```

```
    LCSH: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/LCSH')
```

```
    LicenseDocument: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/LicenseDocument')
```

```
LinguisticSystem: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/LinguisticSystem')  
  
Location: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/Location')  
  
LocationPeriodOrJurisdiction: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/LocationPeriodOrJurisdiction')  
  
MESH: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/MESH')  
  
MediaType: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/MediaType')  
  
MediaTypeOrExtent: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/MediaTypeOrExtent')  
  
MethodOfAccrual: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/MethodOfAccrual')  
  
MethodOfInstruction: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/MethodOfInstruction')  
  
NLM: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/NLM')  
  
Period: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/Period')  
  
PeriodOfTime: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/PeriodOfTime')  
  
PhysicalMedium: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/PhysicalMedium')  
  
PhysicalResource: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/PhysicalResource')  
  
Point: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/Point')  
  
Policy: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/Policy')  
  
ProvenanceStatement: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/ProvenanceStatement')  
  
RFC1766: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/RFC1766')  
  
RFC3066: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/RFC3066')  
  
RFC4646: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/RFC4646')  
  
RFC5646: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/RFC5646')  
  
RightsStatement: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/RightsStatement')  
  
SizeOrDuration: URIRef =  
rdflib.term.URIRef('http://purl.org/dc/terms/SizeOrDuration')  
  
Standard: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/Standard')  
  
TGN: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/TGN')  
  
UDC: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/UDC')
```

```
URI: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/URI')

W3CDTF: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/W3CDTF')

abstract: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/abstract')

accessRights: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/accessRights')

accrualMethod: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/accrualMethod')

accrualPeriodicity: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/accrualPeriodicity')

accrualPolicy: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/accrualPolicy')

alternative: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/alternative')

audience: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/audience')

available: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/available')

bibliographicCitation: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/bibliographicCitation')

conformsTo: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/conformsTo')

contributor: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/contributor')

coverage: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/coverage')

created: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/created')

creator: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/creator')

date: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/date')

dateAccepted: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/dateAccepted')

dateCopyrighted: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/dateCopyrighted')

dateSubmitted: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/dateSubmitted')

description: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/description')

educationLevel: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/educationLevel')

extent: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/extent')

format: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/format')

hasFormat: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/hasFormat')

hasPart: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/hasPart')
```



```
hasVersion: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/hasVersion')
identifier: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/identifier')
instructionalMethod: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/instructionalMethod')
isFormatOf: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/isFormatOf')
isPartOf: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/isPartOf')
isReferencedBy: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/isReferencedBy')
isReplacedBy: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/isReplacedBy')
isRequiredBy: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/isRequiredBy')
isVersionOf: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/isVersionOf')
issued: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/issued')
language: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/language')
license: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/license')
mediator: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/mediator')
medium: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/medium')
modified: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/modified')
provenance: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/provenance')
publisher: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/publisher')
references: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/references')
relation: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/relation')
replaces: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/replaces')
requires: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/requires')
rights: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/rights')
rightsHolder: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/rightsHolder')
source: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/source')
spatial: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/spatial')
subject: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/subject')
tableOfContents: URIRef =
rdflib.term.URIRef('http://purl.org/dc/terms/tableOfContents')
temporal: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/temporal')
title: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/title')
```

```
type: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/type')
valid: URIRef = rdflib.term.URIRef('http://purl.org/dc/terms/valid')

class rdflib.DOAP
    Bases: DefinedNamespace
    Description of a Project (DOAP) vocabulary
    The Description of a Project (DOAP) vocabulary, described using W3C RDF Schema and the Web Ontology Language.
    Generated from: http://usefulinc.com/ns/doap Date: 2020-05-26 14:20:01.307972
    ArchRepository: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#ArchRepository')

    BKRepository: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#BKRepository')

    BazaarBranch: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#BazaarBranch')

    CVSRepository: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#CVSRepository')

    DarcsRepository: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#DarcsRepository')

    GitBranch: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#GitBranch')

    GitRepository: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#GitRepository')

    HgRepository: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#HgRepository')

    Project: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#Project')

    Repository: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#Repository')

    SVNRepository: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#SVNRepository')

    Specification: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#Specification')

    Version: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#Version')

    audience: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#audience')

    blog: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#blog')

    browse: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#browse')

    category: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#category')

    created: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#created')
```

```

description: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#description')

developer: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#developer')

documenter: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#documenter')

helper: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#helper')

homepage: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#homepage')

implements: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#implements')

language: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#language')

license: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#license')

location: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#location')

maintainer: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#maintainer')

module: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#module')

name: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#name')

os: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#os')

platform: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#platform')

release: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#release')

repository: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#repository')

repositoryOf: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#repositoryOf')

revision: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#revision')

screenshots: URIRef =
rdflib.term.URIRef('http://usefulinc.com/ns/doap#screenshots')

shortdesc: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#shortdesc')

tester: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#tester')

translator: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#translator')

vendor: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#vendor')

wiki: URIRef = rdflib.term.URIRef('http://usefulinc.com/ns/doap#wiki')

```

```
class rdflib.Dataset(store='default', default_union=False, default_graph_base=None)
```

Bases: *ConjunctiveGraph*

RDF 1.1 Dataset. Small extension to the Conjunctive Graph: - the primary term is graphs in the datasets and not contexts with quads, so there is a separate method to set/retrieve a graph in a dataset and operate with graphs - graphs cannot be identified with blank nodes - added a method to directly add a single quad

Examples of usage:

```

>>> # Create a new Dataset
>>> ds = Dataset()
>>> # simple triples goes to default graph
>>> ds.add((URIRef("http://example.org/a"),
...         URIRef("http://www.example.org/b"),
...         Literal("foo")))
<Graph identifier=... (<class 'rdflib.graph.Dataset'>)>
>>>
>>> # Create a graph in the dataset, if the graph name has already been
>>> # used, the corresponding graph will be returned
>>> # (ie, the Dataset keeps track of the constituent graphs)
>>> g = ds.graph(URIRef("http://www.example.com/gr"))
>>>
>>> # add triples to the new graph as usual
>>> g.add(
...     (URIRef("http://example.org/x"),
...      URIRef("http://example.org/y"),
...      Literal("bar")))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> # alternatively: add a quad to the dataset -> goes to the graph
>>> ds.add(
...     (URIRef("http://example.org/x"),
...      URIRef("http://example.org/z"),
...      Literal("foo-bar"), g))
<Graph identifier=... (<class 'rdflib.graph.Dataset'>)>
>>>
>>> # querying triples return them all regardless of the graph
>>> for t in ds.triples((None, None, None)):
...     print(t)
(rdflib.term.URIRef("http://example.org/a"),
 rdflib.term.URIRef("http://www.example.org/b"),
 rdflib.term.Literal("foo"))
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/z"),
 rdflib.term.Literal("foo-bar"))
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/y"),
 rdflib.term.Literal("bar"))
>>>
>>> # querying quads() return quads; the fourth argument can be unrestricted
>>> # (None) or restricted to a graph
>>> for q in ds.quads((None, None, None, None)):
...     print(q)
(rdflib.term.URIRef("http://example.org/a"),
 rdflib.term.URIRef("http://www.example.org/b"),
 rdflib.term.Literal("foo"),
 None)
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/y"),
 rdflib.term.Literal("bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/z"),

```

(continues on next page)

(continued from previous page)

```

rdflib.term.Literal("foo-bar"),
rdflib.term.URIRef("http://www.example.com/gr"))
>>>
>>> # unrestricted looping is equivalent to iterating over the entire Dataset
>>> for q in ds:
...     print(q)
(rdflib.term.URIRef("http://example.org/a"),
 rdflib.term.URIRef("http://www.example.org/b"),
 rdflib.term.Literal("foo"),
 None)
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/y"),
 rdflib.term.Literal("bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/z"),
 rdflib.term.Literal("foo-bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
>>>
>>> # restricting iteration to a graph:
>>> for q in ds.quads((None, None, None, g)):
...     print(q)
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/y"),
 rdflib.term.Literal("bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
(rdflib.term.URIRef("http://example.org/x"),
 rdflib.term.URIRef("http://example.org/z"),
 rdflib.term.Literal("foo-bar"),
 rdflib.term.URIRef("http://www.example.com/gr"))
>>> # Note that in the call above -
>>> # ds.quads((None, None, None, "http://www.example.com/gr"))
>>> # would have been accepted, too
>>>
>>> # graph names in the dataset can be queried:
>>> for c in ds.graphs():
...     print(c) # doctest:
DEFAULT
http://www.example.com/gr
>>> # A graph can be created without specifying a name; a skolemized genid
>>> # is created on the fly
>>> h = ds.graph()
>>> for c in ds.graphs():
...     print(c)
DEFAULT
https://rdflib.github.io/.well-known/genid/rdflib/N...
http://www.example.com/gr
>>> # Note that the Dataset.graphs() call returns names of empty graphs,
>>> # too. This can be restricted:
>>> for c in ds.graphs(empty=False):
...     print(c)
DEFAULT

```

(continues on next page)

(continued from previous page)

```
http://www.example.com/gr
>>>
>>> # a graph can also be removed from a dataset via ds.remove_graph(g)
```

New in version 4.0.

#### Parameters

- **store** (`Union[Store, str]`) –
- **default\_union** (`bool`) –
- **default\_graph\_base** (`Optional[str]`) –

```
__annotations__ = {'__identifier': '_ContextIdentifierType', '__store': 'Store',
'default_context': '_ContextType'}
```

```
__getstate__()
```

#### Return type

```
Tuple[Store, IdentifiedNode, Graph, bool]
```

```
__init__(store='default', default_union=False, default_graph_base=None)
```

#### Parameters

- **store** (`Union[Store, str]`) –
- **default\_union** (`bool`) –
- **default\_graph\_base** (`Optional[str]`) –

```
__iter__()
```

Iterates over all quads in the store

#### Return type

```
Generator[Tuple[Node, Node, Node, Optional[IdentifiedNode]], None, None]
```

```
__module__ = 'rdflib.graph'
```

```
__reduce__()
```

Helper for pickle.

#### Return type

```
Tuple[Type[Dataset], Tuple[Store, bool]]
```

```
__setstate__(state)
```

#### Parameters

```
state (Tuple[Store, IdentifiedNode, Graph, bool]) –
```

#### Return type

```
None
```

```
__str__()
```

Return str(self).

#### Return type

```
str
```

**add\_graph**(*g*)

alias of graph for consistency

**Parameters**

**g** (Union[*IdentifiedNode*, *Graph*, str, None]) –

**Return type**

*Graph*

**contexts**(*triple=None*)

Iterate over all contexts in the graph

If triple is specified, iterate over all contexts the triple is in.

**Parameters**

**triple** (Optional[Tuple[*Node*, *Node*, *Node*]]) –

**Return type**

Generator[*Graph*, None, None]

**default\_context:** *Graph*

**graph**(*identifier=None*, *base=None*)

**Parameters**

- **identifier** (Union[*IdentifiedNode*, *Graph*, str, None]) –
- **base** (Optional[str]) –

**Return type**

*Graph*

**graphs**(*triple=None*)

Iterate over all contexts in the graph

If triple is specified, iterate over all contexts the triple is in.

**Parameters**

**triple** (Optional[Tuple[*Node*, *Node*, *Node*]]) –

**Return type**

Generator[*Graph*, None, None]

**parse**(*source=None*, *publicID=None*, *format=None*, *location=None*, *file=None*, *data=None*, *\*\*args*)

**Caution:** This method can access directly or indirectly requested network or file resources, for example, when parsing JSON-LD documents with @context directives that point to a network location.

When processing untrusted or potentially malicious documents, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

**Parameters**

- **source** (Union[IO[bytes], TextIO, *InputSource*, str, bytes, PurePath, None]) –
- **publicID** (Optional[str]) –
- **format** (Optional[str]) –

- **location** (`Optional[str]`) –
- **file** (`Union[BinaryIO, TextIO, None]`) –
- **data** (`Union[str, bytes, None]`) –
- **args** (`Any`) –

**Return type**  
`Graph`

**quads**(*quad=None*)

Iterate over all the quads in the entire conjunctive graph

**Parameters**

**quad** (`Union[Tuple[Optional[Node], Optional[Node], Optional[Node]], Tuple[Optional[Node], Optional[Node], Optional[Node], Optional[Graph]], None]`) –

**Return type**

`Generator[Tuple[Node, Node, Node, Optional[IdentifiedNode]], None, None]`

**remove\_graph**(*g*)

**Parameters**

- **self** (`TypeVar(_DatasetT, bound= Dataset)`) –
- **g** (`Union[IdentifiedNode, Graph, str, None]`) –

**Return type**

`TypeVar(_DatasetT, bound= Dataset)`

**class** `rdflib.FOAF`

Bases: `DefinedNamespace`

Friend of a Friend (FOAF) vocabulary

The Friend of a Friend (FOAF) RDF vocabulary, described using W3C RDF Schema and the Web Ontology Language.

Generated from: <http://xmlns.com/foaf/spec/index.rdf> Date: 2020-05-26 14:20:01.597998

**Agent:** `URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/Agent')`

**Document:** `URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/Document')`

**Group:** `URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/Group')`

**Image:** `URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/Image')`

**LabelProperty:** `URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/LabelProperty')`

**OnlineAccount:** `URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/OnlineAccount')`

**OnlineChatAccount:** `URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/OnlineChatAccount')`

**OnlineEcommerceAccount:** `URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/OnlineEcommerceAccount')`



```

OnlineGamingAccount: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/OnlineGamingAccount')

Organization: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/Organization')

Person: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/Person')

PersonalProfileDocument: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/PersonalProfileDocument')

Project: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/Project')

account: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/account')

accountName: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/accountName')

accountServiceHomepage: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/accountServiceHomepage')

age: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/age')

aimChatID: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/aimChatID')

based_near: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/based_near')

birthday: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/birthday')

currentProject: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/currentProject')

depiction: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/depiction')

depicts: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/depicts')

dnaChecksum: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/dnaChecksum')

familyName: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/familyName')

family_name: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/family_name')

firstName: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/firstName')

focus: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/focus')

fundedBy: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/fundedBy')

geekcode: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/geekcode')

gender: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/gender')

givenName: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/givenName')

givenname: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/givenname')

holdsAccount: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/holdsAccount')

homepage: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/homepage')

icqChatID: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/icqChatID')

```

```
img: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/img')

interest: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/interest')

isPrimaryTopicOf: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/isPrimaryTopicOf')

jabberID: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/jabberID')

knows: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/knows')

lastName: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/lastName')

logo: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/logo')

made: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/made')

maker: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/maker')

mbox: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/mbox')

mbox_sha1sum: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/mbox_sha1sum')

member: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/member')

membershipClass: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/membershipClass')

msnChatID: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/msnChatID')

myersBriggs: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/myersBriggs')

name: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/name')

nick: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/nick')

openid: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/openid')

page: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/page')

pastProject: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/pastProject')

phone: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/phone')

plan: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/plan')

primaryTopic: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/primaryTopic')

publications: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/publications')

schoolHomepage: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/schoolHomepage')

sha1: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/sha1')

skypeID: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/skypeID')

status: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/status')

surname: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/surname')
```

```

theme: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/theme')

thumbnail: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/thumbnail')

tipjar: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/tipjar')

title: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/title')

topic: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/topic')

topic_interest: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/topic_interest')

weblog: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/weblog')

workInfoHomepage: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/workInfoHomepage')

workplaceHomepage: URIRef =
rdflib.term.URIRef('http://xmlns.com/foaf/0.1/workplaceHomepage')

yahooChatID: URIRef = rdflib.term.URIRef('http://xmlns.com/foaf/0.1/yahooChatID')

class rdflib.Graph(store='default', identifier=None, namespace_manager=None, base=None,
                    bind_namespaces='rdflib')

```

Bases: *Node*

An RDF Graph

The constructor accepts one argument, the “store” that will be used to store the graph data (see the “store” package for stores currently shipped with rdflib).

Stores can be context-aware or unaware. Unaware stores take up (some) less space but cannot support features that require context, such as true merging/demerging of sub-graphs and provenance.

Even if used with a context-aware store, Graph will only expose the quads which belong to the default graph. To access the rest of the data, *ConjunctiveGraph* or *Dataset* classes can be used instead.

The Graph constructor can take an identifier which identifies the Graph by name. If none is given, the graph is assigned a BNode for its identifier.

For more on named graphs, see: <http://www.w3.org/2004/03/trix/>

#### Parameters

- **store** (*Union*[*Store*, *str*]) –
- **identifier** (*Union*[*IdentifiedNode*, *str*, *None*]) –
- **namespace\_manager** (*Optional*[*NamespaceManager*]) –
- **base** (*Optional*[*str*]) –
- **bind\_namespaces** (*Literal*['core', 'rdflib', 'none']) –

**\_\_add\_\_** (*other*)

Set-theoretic union BNode IDs are not changed.

#### Parameters

**other** (*Graph*) –

#### Return type

*Graph*

**\_\_and\_\_**(*other*)

Set-theoretic intersection. BNode IDs are not changed.

**Parameters**

**other** (*Graph*) –

**Return type**

*Graph*

**\_\_annotations\_\_** = {'\_\_identifier': '\_ContextIdentifierType', '\_\_store': 'Store'}

**\_\_cmp\_\_**(*other*)

**Return type**

*int*

**\_\_contains\_\_**(*triple*)

Support for ‘triple in graph’ syntax

**Parameters**

**triple** (*Tuple*[*Optional*[*Node*], *Optional*[*Node*], *Optional*[*Node*]]) –

**Return type**

*bool*

```

__dict__ = mappingproxy({'__module__': 'rdflib.graph', '__doc__': 'An RDF
Graph\n\n The constructor accepts one argument, the "store"\n that will be used to
store the graph data (see the "store"\n package for stores currently shipped with
rdflib).\n\n Stores can be context-aware or unaware. Unaware stores take up\n (some)
less space but cannot support features that require\n context, such as true
merging/demerging of sub-graphs and\n provenance.\n\n Even if used with a
context-aware store, Graph will only expose the quads which\n belong to the default
graph. To access the rest of the data, `ConjunctiveGraph` or\n `Dataset` classes can
be used instead.\n\n The Graph constructor can take an identifier which identifies
the Graph\n by name. If none is given, the graph is assigned a BNode for its\n
identifier.\n\n For more on named graphs, see: http://www.w3.org/2004/03/trix/\n ',
'__init__': <function Graph.__init__>, 'store': <property object>, 'identifier':
<property object>, 'namespace_manager': <property object>, '__repr__': <function
Graph.__repr__>, '__str__': <function Graph.__str__>, 'toPython': <function
Graph.toPython>, 'destroy': <function Graph.destroy>, 'commit': <function
Graph.commit>, 'rollback': <function Graph.rollback>, 'open': <function
Graph.open>, 'close': <function Graph.close>, 'add': <function Graph.add>, 'addN':
<function Graph.addN>, 'remove': <function Graph.remove>, 'triples': <function
Graph.triples>, '__getitem__': <function Graph.__getitem__>, '__len__': <function
Graph.__len__>, '__iter__': <function Graph.__iter__>, '__contains__': <function
Graph.__contains__>, '__hash__': <function Graph.__hash__>, '__cmp__': <function
Graph.__cmp__>, '__eq__': <function Graph.__eq__>, '__lt__': <function
Graph.__lt__>, '__le__': <function Graph.__le__>, '__gt__': <function
Graph.__gt__>, '__ge__': <function Graph.__ge__>, '__iadd__': <function
Graph.__iadd__>, '__isub__': <function Graph.__isub__>, '__add__': <function
Graph.__add__>, '__mul__': <function Graph.__mul__>, '__sub__': <function
Graph.__sub__>, '__xor__': <function Graph.__xor__>, '__or__': <function
Graph.__add__>, '__and__': <function Graph.__mul__>, 'set': <function Graph.set>,
'subjects': <function Graph.subjects>, 'predicates': <function Graph.predicates>,
'objects': <function Graph.objects>, 'subject_predicates': <function
Graph.subject_predicates>, 'subject_objects': <function Graph.subject_objects>,
'predicate_objects': <function Graph.predicate_objects>, 'triples_choices':
<function Graph.triples_choices>, 'value': <function Graph.value>, 'items':
<function Graph.items>, 'transitiveClosure': <function Graph.transitiveClosure>,
'transitive_objects': <function Graph.transitive_objects>, 'transitive_subjects':
<function Graph.transitive_subjects>, 'qname': <function Graph.qname>,
'compute_qname': <function Graph.compute_qname>, 'bind': <function Graph.bind>,
'namespaces': <function Graph.namespaces>, 'absolutize': <function
Graph.absolutize>, 'serialize': <function Graph.serialize>, 'print': <function
Graph.print>, 'parse': <function Graph.parse>, 'query': <function Graph.query>,
'update': <function Graph.update>, 'n3': <function Graph.n3>, '__reduce__':
<function Graph.__reduce__>, 'isomorphic': <function Graph.isomorphic>,
'connected': <function Graph.connected>, 'all_nodes': <function Graph.all_nodes>,
'collection': <function Graph.collection>, 'resource': <function Graph.resource>,
'_process_skolem_tuples': <function Graph._process_skolem_tuples>, 'skolemize':
<function Graph.skolemize>, 'de_skolemize': <function Graph.de_skolemize>, 'cbd':
<function Graph.cbd>, '__dict__': <attribute '__dict__' of 'Graph' objects>,
'__weakref__': <attribute '__weakref__' of 'Graph' objects>, '__annotations__':
{'__identifier': '_ContextIdentifierType', '__store': 'Store'}})

```

`__eq__(other)`

Return self==value.

Return type

`bool``__ge__(other)`

Return self&gt;=value.

**Parameters****other** (*Graph*) –**Return type**`bool``__getitem__(item)`

A graph can be “sliced” as a shortcut for the triples method. The python slice syntax is (ab)used for specifying triples. A generator over matches is returned, the returned tuples include only the parts not given

```
>>> import rdflib
>>> g = rdflib.Graph()
>>> g.add((rdflib.URIRef("urn:bob"), namespace.RDFS.label, rdflib.Literal("Bob"
↪)))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
```

```
>>> list(g[rdflib.URIRef("urn:bob")]) # all triples about bob
[(rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#label'), rdflib.term.
↪Literal('Bob'))]
```

```
>>> list(g[:namespace.RDFS.label]) # all label triples
[(rdflib.term.URIRef('urn:bob'), rdflib.term.Literal('Bob'))]
```

```
>>> list(g[:,rdflib.Literal("Bob")]) # all triples with bob as object
[(rdflib.term.URIRef('urn:bob'), rdflib.term.URIRef('http://www.w3.org/2000/01/
↪rdf-schema#label'))]
```

Combined with SPARQL paths, more complex queries can be written concisely:

Name of all Bobs friends:

```
g[bob : FOAF.knows/FOAF.name ]
```

Some label for Bob:

```
g[bob : DC.title|FOAF.name|RDFS.label]
```

All friends and friends of friends of Bob

```
g[bob : FOAF.knows * "+"]
```

etc.

New in version 4.0.

`__gt__(other)`

Return self&gt;value.

**Return type**`bool``__hash__()`

Return hash(self).

**Return type**`int`

**\_\_iadd\_\_**(*other*)

Add all triples in Graph *other* to Graph. BNode IDs are not changed.

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **other** (`Iterable[Tuple[Node, Node, Node]]`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

**\_\_init\_\_**(*store='default', identifier=None, namespace\_manager=None, base=None, bind\_namespaces='rdflib'*)

**Parameters**

- **store** (`Union[Store, str]`) –
- **identifier** (`Union[IdentifiedNode, str, None]`) –
- **namespace\_manager** (`Optional[NamespaceManager]`) –
- **base** (`Optional[str]`) –
- **bind\_namespaces** (`Literal['core', 'rdflib', 'none']`) –

**\_\_isub\_\_**(*other*)

Subtract all triples in Graph *other* from Graph. BNode IDs are not changed.

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **other** (`Iterable[Tuple[Node, Node, Node]]`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

**\_\_iter\_\_**()

Iterates over all triples in the store

**Return type**

`Generator[Tuple[Node, Node, Node], None, None]`

**\_\_le\_\_**(*other*)

Return self<=value.

**Parameters**

- **other** (`Graph`) –

**Return type**

`bool`

**\_\_len\_\_**()

Returns the number of triples in the graph

If context is specified then the number of triples in the context is returned instead.

**Return type**

`int`

**\_\_lt\_\_**(*other*)  
Return self<value.  
**Return type**  
*bool*

**\_\_module\_\_** = 'rdflib.graph'

**\_\_mul\_\_**(*other*)  
Set-theoretic intersection. BNode IDs are not changed.  
**Parameters**  
**other** (*Graph*) –  
**Return type**  
*Graph*

**\_\_or\_\_**(*other*)  
Set-theoretic union BNode IDs are not changed.  
**Parameters**  
**other** (*Graph*) –  
**Return type**  
*Graph*

**\_\_reduce\_\_**()  
Helper for pickle.  
**Return type**  
*Tuple*[*Type*[*Graph*], *Tuple*[*Store*, *IdentifiedNode*]]

**\_\_repr\_\_**()  
Return repr(self).  
**Return type**  
*str*

**\_\_str\_\_**()  
Return str(self).  
**Return type**  
*str*

**\_\_sub\_\_**(*other*)  
Set-theoretic difference. BNode IDs are not changed.  
**Parameters**  
**other** (*Graph*) –  
**Return type**  
*Graph*

**\_\_weakref\_\_**  
list of weak references to the object (if defined)

**\_\_xor\_\_**(*other*)  
Set-theoretic XOR. BNode IDs are not changed.  
**Parameters**  
**other** (*Graph*) –



**Return type***Graph***absolutize**(*uri*, *defrag=1*)

Turn uri into an absolute URI if it's not one already

**Parameters**

- **uri** (*str*) –
- **defrag** (*int*) –

**Return type***URIRef***add**(*triple*)

Add a triple with self as context

**Parameters**

- **self** (*TypeVar*(*\_GraphT*, bound= *Graph*)) –
- **triple** (*Tuple*[*Node*, *Node*, *Node*]) –

**Return type***TypeVar*(*\_GraphT*, bound= *Graph*)**addN**(*quads*)

Add a sequence of triple with context

**Parameters**

- **self** (*TypeVar*(*\_GraphT*, bound= *Graph*)) –
- **quads** (*Iterable*[*Tuple*[*Node*, *Node*, *Node*, *Graph*]]) –

**Return type***TypeVar*(*\_GraphT*, bound= *Graph*)**all\_nodes**()**Return type***Set*[*Node*]**bind**(*prefix*, *namespace*, *override=True*, *replace=False*)

Bind prefix to namespace

If override is True will bind namespace to given prefix even if namespace was already bound to a different prefix.

if replace, replace any existing prefix with the new namespace

for example: graph.bind("foaf", "http://xmlns.com/foaf/0.1/")

**Parameters**

- **prefix** (*Optional*[*str*]) –
- **namespace** (*Any*) –
- **override** (*bool*) –
- **replace** (*bool*) –

**Return type***None*

**cbd(resource)**

Retrieves the Concise Bounded Description of a Resource from a Graph

Concise Bounded Description (CBD) is defined in [1] as:

Given a particular node (the starting node) in a particular RDF graph (the source graph), a subgraph of that particular graph, taken to comprise a concise bounded description of the resource denoted by the starting node, can be identified as follows:

1. **Include in the subgraph all statements in the source graph where the subject of the statement is the**  
starting node;
2. **Recursively, for all statements identified in the subgraph thus far having a blank node object, include**  
in the subgraph all statements in the source graph where the subject of the statement is the blank node in question and which are not already included in the subgraph.
3. **Recursively, for all statements included in the subgraph thus far, for all reifications of each statement**  
in the source graph, include the concise bounded description beginning from the `rdf:Statement` node of each reification.

This results in a subgraph where the object nodes are either URI references, literals, or blank nodes not serving as the subject of any statement in the graph.

[1] <https://www.w3.org/Submission/CBD/>

**Parameters**

**resource** (*Node*) – a `URIRef` object, of the Resource for queried for

**Return type**

*Graph*

**Returns**

a `Graph`, subgraph of self

**close(commit\_pending\_transaction=False)**

Close the graph store

Might be necessary for stores that require closing a connection to a database or releasing some resource.

**Parameters**

**commit\_pending\_transaction** (*bool*) –

**Return type**

*None*

**collection(identifier)**

Create a new `Collection` instance.

Parameters:

- **identifier**: a `URIRef` or `BNode` instance.

Example:

```
>>> graph = Graph()
>>> uri = URIRef("http://example.org/resource")
>>> collection = graph.collection(uri)
>>> assert isinstance(collection, Collection)
>>> assert collection.uri is uri
```

(continues on next page)

(continued from previous page)

```
>>> assert collection.graph is graph
>>> collection += [ Literal(1), Literal(2) ]
```

**Parameters****identifier** (*Node*) –**Return type***Collection***commit()**

Commits active transactions

**Parameters****self** (*TypeVar*(*\_GraphT*, bound= *Graph*)) –**Return type***TypeVar*(*\_GraphT*, bound= *Graph*)**compute\_qname(uri, generate=True)****Parameters**

- **uri** (*str*) –
- **generate** (*bool*) –

**Return type***Tuple*[*str*, *URIRef*, *str*]**connected()**

Check if the Graph is connected

The Graph is considered undirectional.

Performs a search on the Graph, starting from a random node. Then iteratively goes depth-first through the triplets where the node is subject and object. Return True if all nodes have been visited and False if it cannot continue and there are still unvisited nodes left.

**Return type***bool***de\_skolemize(new\_graph=None, uriref=None)****Parameters**

- **new\_graph** (*Optional*[*Graph*]) –
- **uriref** (*Optional*[*URIRef*]) –

**Return type***Graph***destroy(configuration)**

Destroy the store identified by configuration if supported

**Parameters**

- **self** (*TypeVar*(*\_GraphT*, bound= *Graph*)) –
- **configuration** (*str*) –

**Return type**`TypeVar(_GraphT, bound= Graph)`**property identifier:** `IdentifiedNode`**isomorphic**(*other*)

does a very basic check if these graphs are the same If no BNodes are involved, this is accurate.

See `rdflib.compare` for a correct implementation of isomorphism checks

**Parameters****other** (`Graph`) –**Return type**`bool`**items**(*list*)

Generator over all items in the resource specified by list

list is an RDF collection.

**Parameters****list** (`Node`) –**Return type**`Generator[Node, None, None]`**n3**()

Return an n3 identifier for the Graph

**Return type**`str`**property namespace\_manager:** `NamespaceManager`

this graph's namespace-manager

**namespaces**()

Generator over all the prefix, namespace tuples

**Return type**`Generator[Tuple[str, URIRef], None, None]`**objects**(*subject=None, predicate=None, unique=False*)

A generator of (optionally unique) objects with the given subject and predicate

**Parameters**

- **subject** (`Optional[Node]`) –
- **predicate** (`Union[None, Path, Node]`) –
- **unique** (`bool`) –

**Return type**`Generator[Node, None, None]`**open**(*configuration, create=False*)

Open the graph store

Might be necessary for stores that require opening a connection to a database or acquiring some resource.

**Parameters**

- **configuration** (`str`) –

- **create**(bool) –

**Return type**

Optional[int]

**parse**(source=None, publicID=None, format=None, location=None, file=None, data=None, \*\*args)

Parse an RDF source adding the resulting triples to the Graph.

The source is specified using one of source, location, file or data.

**Caution:** This method can access directly or indirectly requested network or file resources, for example, when parsing JSON-LD documents with `@context` directives that point to a network location.

When processing untrusted or potentially malicious documents, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib [Security Considerations](#) documentation.

**Parameters**

- **source:** An InputSource, file-like object, or string. In the case of a string the string is the location of the source.
- **location:** A string indicating the relative or absolute URL of the source. Graph's `absolutize` method is used if a relative location is specified.
- **file:** A file-like object.
- **data:** A string containing the data to be parsed.
- **format:** Used if format can not be determined from source, e.g. file extension or Media Type. Defaults to `text/turtle`. Format support can be extended with plugins, but `"xml"`, `"n3"` (use for `turtle`), `"nt"` & `"trix"` are built in.
- **publicID:** the logical URI to use as the document base. If None specified the document location is used (at least in the case where there is a document location).

**Returns**

- `self`, the graph instance.

Examples:

```
>>> my_data = '''
... <rdf:RDF
...   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
...   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
... >
...   <rdf:Description>
...     <rdfs:label>Example</rdfs:label>
...     <rdfs:comment>This is really just an example.</rdfs:comment>
...   </rdf:Description>
... </rdf:RDF>
... '''
>>> import os, tempfile
>>> fd, file_name = tempfile.mkstemp()
>>> f = os.fdopen(fd, "w")
```

(continues on next page)

(continued from previous page)

```
>>> dummy = f.write(my_data) # Returns num bytes written
>>> f.close()
```

```
>>> g = Graph()
>>> result = g.parse(data=my_data, format="application/rdf+xml")
>>> len(g)
2
```

```
>>> g = Graph()
>>> result = g.parse(location=file_name, format="application/rdf+xml")
>>> len(g)
2
```

```
>>> g = Graph()
>>> with open(file_name, "r") as f:
...     result = g.parse(f, format="application/rdf+xml")
>>> len(g)
2
```

```
>>> os.remove(file_name)
```

```
>>> # default turtle parsing
>>> result = g.parse(data="<http://example.com/a> <http://example.com/a> <http://
↳ /example.com/a> .")
>>> len(g)
3
```

### Parameters

- **source** (`Union[IO[bytes], TextIO, InputSource, str, bytes, PurePath, None]`) –
- **publicID** (`Optional[str]`) –
- **format** (`Optional[str]`) –
- **location** (`Optional[str]`) –
- **file** (`Union[BinaryIO, TextIO, None]`) –
- **data** (`Union[str, bytes, None]`) –
- **args** (`Any`) –

### Return type

*Graph*

### **predicate\_objects**(*subject=None, unique=False*)

A generator of (optionally unique) (predicate, object) tuples for the given subject

### Parameters

- **subject** (`Optional[Node]`) –
- **unique** (`bool`) –

### Return type

`Generator[Tuple[Node, Node], None, None]`

**predicates**(*subject=None, object=None, unique=False*)

A generator of (optionally unique) predicates with the given subject and object

**Parameters**

- **subject** (*Optional[Node]*) –
- **object** (*Optional[Node]*) –
- **unique** (*bool*) –

**Return type**

*Generator[Node, None, None]*

**print**(*format='turtle', encoding='utf-8', out=None*)

**Parameters**

- **format** (*str*) –
- **encoding** (*str*) –
- **out** (*Optional[TextIO]*) –

**Return type**

*None*

**qname**(*uri*)

**Parameters**

**uri** (*str*) –

**Return type**

*str*

**query**(*query\_object, processor='sparql', result='sparql', initNs=None, initBindings=None, use\_store\_provided=True, \*\*kwargs*)

Query this graph.

A type of ‘prepared queries’ can be realised by providing initial variable bindings with *initBindings*

Initial namespaces are used to resolve prefixes used in the query, if none are given, the namespaces from the graph’s namespace manager are used.

**Caution:** This method can access indirectly requested network endpoints, for example, query processing will attempt to access network endpoints specified in *SERVICE* directives.

When processing untrusted or potentially malicious queries, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

**Returntype**

*Result*

**Parameters**

- **query\_object** (*Union[str, Query]*) –
- **processor** (*Union[str, Processor]*) –
- **result** (*Union[str, Type[Result]]*) –

- **initNs** (`Optional[Mapping[str, Any]]`) –
- **initBindings** (`Optional[Mapping[str, Identifier]]`) –
- **use\_store\_provided** (`bool`) –
- **kwargs** (`Any`) –

**Return type**

`Result`

**remove**(*triple*)

Remove a triple from the graph

If the triple does not provide a context attribute, removes the triple from all contexts.

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **triple** (`Tuple[Optional[Node], Optional[Node], Optional[Node]]`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

**resource**(*identifier*)

Create a new Resource instance.

Parameters:

- **identifier**: a URIRef or BNode instance.

Example:

```
>>> graph = Graph()
>>> uri = URIRef("http://example.org/resource")
>>> resource = graph.resource(uri)
>>> assert isinstance(resource, Resource)
>>> assert resource.identifier is uri
>>> assert resource.graph is graph
```

**Parameters**

**identifier** (`Union[Node, str]`) –

**Return type**

`Resource`

**rollback**()

Rollback active transactions

**Parameters**

**self** (`TypeVar(_GraphT, bound= Graph)`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

**serialize**(*destination=None, format='turtle', base=None, encoding=None, \*\*args*)

Serialize the graph.

**Parameters**



- **destination** (`Union[str, PurePath, IO[bytes], None]`) – The destination to serialize the graph to. This can be a path as a `str` or `PurePath` object, or it can be a `IO[bytes]` like object. If this parameter is not supplied the serialized graph will be returned.
- **format** (`str`) – The format that the output should be written in. This value references a `Serializer` plugin. Format support can be extended with plugins, but "xml", "n3", "turtle", "nt", "pretty-xml", "trix", "trig", "nquads", "json-ld" and "hex" are built in. Defaults to "turtle".
- **base** (`Optional[str]`) – The base IRI for formats that support it. For the turtle format this will be used as the @base directive.
- **encoding** (`Optional[str]`) – Encoding of output.
- **args** (`Any`) – Additional arguments to pass to the `Serializer` that will be used.
- **self** (`TypeVar(_GraphT, bound= Graph)`) –

**Returns**

The serialized graph if `destination` is `None`. The serialized graph is returned as `str` if no encoding is specified, and as `bytes` if an encoding is specified.

**Return type**

`bytes` if `destination` is `None` and encoding is not `None`.

**Return type**

`str` if `destination` is `None` and encoding is `None`.

**Returns**

`self` (i.e. the `Graph` instance) if `destination` is not `None`.

**Return type**

`Graph` if `destination` is not `None`.

**set(triple)**

Convenience method to update the value of object

Remove any existing triples for subject and predicate before adding (subject, predicate, object).

**Parameters**

- **self** (`TypeVar(_GraphT, bound= Graph)`) –
- **triple** (`Tuple[Node, Node, Node]`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

**skolemize(new\_graph=None, bnode=None, authority=None, basepath=None)****Parameters**

- **new\_graph** (`Optional[Graph]`) –
- **bnode** (`Optional[BNode]`) –
- **authority** (`Optional[str]`) –
- **basepath** (`Optional[str]`) –

**Return type**

`Graph`

property store: `Store`

**subject\_objects**(*predicate=None, unique=False*)

A generator of (optionally unique) (subject, object) tuples for the given predicate

**Parameters**

- **predicate** (`Union[None, Path, Node]`) –
- **unique** (`bool`) –

**Return type**

`Generator[Tuple[Node, Node], None, None]`

**subject\_predicates**(*object=None, unique=False*)

A generator of (optionally unique) (subject, predicate) tuples for the given object

**Parameters**

- **object** (`Optional[Node]`) –
- **unique** (`bool`) –

**Return type**

`Generator[Tuple[Node, Node], None, None]`

**subjects**(*predicate=None, object=None, unique=False*)

A generator of (optionally unique) subjects with the given predicate and object

**Parameters**

- **predicate** (`Union[None, Path, Node]`) –
- **object** (`Optional[Node]`) –
- **unique** (`bool`) –

**Return type**

`Generator[Node, None, None]`

**toPython()**

**Parameters**

**self** (`TypeVar(_GraphT, bound= Graph)`) –

**Return type**

`TypeVar(_GraphT, bound= Graph)`

**transitiveClosure**(*func, arg, seen=None*)

Generates transitive closure of a user-defined function against the graph

```
>>> from rdflib.collection import Collection
>>> g=Graph()
>>> a=BNode("foo")
>>> b=BNode("bar")
>>> c=BNode("baz")
>>> g.add((a,RDF.first,RDF.type))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((a,RDF.rest,b))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((b,RDF.first,namespace.RDFS.label))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((b,RDF.rest,c))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
```

(continues on next page)

(continued from previous page)

```

>>> g.add((c,RDF.first,namespace.RDFS.comment))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> g.add((c,RDF.rest,RDF.nil))
<Graph identifier=... (<class 'rdflib.graph.Graph'>)>
>>> def topList(node,g):
...     for s in g.subjects(RDF.rest, node):
...         yield s
>>> def reverseList(node,g):
...     for f in g.objects(node, RDF.first):
...         print(f)
...     for s in g.subjects(RDF.rest, node):
...         yield s

```

```

>>> [rt for rt in g.transitiveClosure(
...     topList,RDF.nil)]
[rdflib.term.BNode('baz'),
 rdflib.term.BNode('bar'),
 rdflib.term.BNode('foo')]

```

```

>>> [rt for rt in g.transitiveClosure(
...     reverseList,RDF.nil)]
http://www.w3.org/2000/01/rdf-schema#comment
http://www.w3.org/2000/01/rdf-schema#label
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
[rdflib.term.BNode('baz'),
 rdflib.term.BNode('bar'),
 rdflib.term.BNode('foo')]

```

**Parameters**

- **func** (Callable[[TypeVar(\_TCArgT), *Graph*], Iterable[TypeVar(\_TCArgT)]] –
- **arg** (TypeVar(\_TCArgT)) –
- **seen** (Optional[Dict[TypeVar(\_TCArgT), int]]) –

**transitive\_objects**(*subject*, *predicate*, *remember=None*)

Transitively generate objects for the *predicate* relationship

Generated objects belong to the depth first transitive closure of the *predicate* relationship starting at *subject*.

**Parameters**

- **subject** (Optional[*Node*]) –
- **predicate** (Optional[*Node*]) –
- **remember** (Optional[Dict[Optional[*Node*], int]]) –

**Return type**

Generator[Optional[*Node*], None, None]

**transitive\_subjects**(*predicate*, *object*, *remember=None*)

Transitively generate subjects for the *predicate* relationship

Generated subjects belong to the depth first transitive closure of the predicate relationship starting at object.

**Parameters**

- **predicate** (`Optional[Node]`) –
- **object** (`Optional[Node]`) –
- **remember** (`Optional[Dict[Optional[Node], int]]`) –

**Return type**

`Generator[Optional[Node], None, None]`

**triples**(*triple*)

Generator over the triple store

Returns triples that match the given triple pattern. If triple pattern does not provide a context, all contexts will be searched.

**Parameters**

**triple** (`Tuple[Optional[Node], Union[Path, Node, None], Optional[Node]]`) –

**Return type**

`Generator[Union[Tuple[Node, Node, Node], Tuple[Node, Path, Node]], None, None]`

**triples\_choices**(*triple*, *context=None*)

**Parameters**

- **triple** (`Union[Tuple[List[Node], Node, Node], Tuple[Node, List[Node], Node], Tuple[Node, Node, List[Node]]]`) –
- **context** (`Optional[Graph]`) –

**Return type**

`Generator[Tuple[Node, Node, Node], None, None]`

**update**(*update\_object*, *processor='sparql'*, *initNs=None*, *initBindings=None*, *use\_store\_provided=True*, *\*\*kwargs*)

Update this graph with the given update query.

**Caution:** This method can access indirectly requested network endpoints, for example, query processing will attempt to access network endpoints specified in SERVICE directives.

When processing untrusted or potentially malicious queries, measures should be taken to restrict network and file access.

For information on available security measures, see the RDFLib *Security Considerations* documentation.

**Parameters**

- **update\_object** (`Union[Update, str]`) –
- **processor** (`Union[str, UpdateProcessor]`) –
- **initNs** (`Optional[Mapping[str, Any]]`) –
- **initBindings** (`Optional[Mapping[str, Identifier]]`) –
- **use\_store\_provided** (`bool`) –

- **kwargs** (*Any*) –

**Return type***None*

**value**(*subject=None, predicate=rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#value'), object=None, default=None, any=True*)

Get a value for a pair of two criteria

Exactly one of subject, predicate, object must be None. Useful if one knows that there may only be one value.

It is one of those situations that occur a lot, hence this ‘macro’ like utility

Parameters: subject, predicate, object – exactly one must be None default – value to be returned if no values found any – if True, return any value in the case there is more than one, else, raise UniquenessError

**Parameters**

- **subject** (*Optional[Node]*) –
- **predicate** (*Optional[Node]*) –
- **object** (*Optional[Node]*) –
- **default** (*Optional[Node]*) –
- **any** (*bool*) –

**Return type***Optional[Node]*

**class** `rdflib IdentifiedNode`(*value: str*)

Bases: *Identifier*

An abstract class, primarily defined to identify Nodes that are not Literals.

The name “Identified Node” is not explicitly defined in the RDF specification, but can be drawn from this section: <https://www.w3.org/TR/rdf-concepts/#section-URI-Vocabulary>

```
__annotations__ = {}
```

```
__dict__ = mappingproxy({'__module__': 'rdflib.term', '__doc__': '\n An abstract class, primarily defined to identify Nodes that are not Literals.\n\n The name "Identified Node" is not explicitly defined in the RDF specification, but can be drawn from this section:
```

```
https://www.w3.org/TR/rdf-concepts/#section-URI-Vocabulary\n ', '__getnewargs__': <function IdentifiedNode.__getnewargs__>, 'toPython': <function IdentifiedNode.toPython>, '__dict__': <attribute '__dict__' of 'IdentifiedNode' objects>, '__weakref__': <attribute '__weakref__' of 'IdentifiedNode' objects>, '__annotations__': {}})
```

```
__getnewargs__()
```

**Return type***Tuple[str]*

```
__module__ = 'rdflib.term'
```

```
__weakref__
```

list of weak references to the object (if defined)

**toPython()**

**Return type**

`str`

**class** `rdflib.Literal`(*lexical\_or\_value: Any*, *lang: Optional[str] = None*, *datatype: Optional[str] = None*, *normalize: Optional[bool] = None*)

Bases: *Identifier*

RDF 1.1's Literals Section: <http://www.w3.org/TR/rdf-concepts/#section-Graph-Literal>

Literals are used for values such as strings, numbers, and dates.

A literal in an RDF graph consists of two or three elements:

- a lexical form, being a Unicode string, which SHOULD be in Normal Form C
- a datatype IRI, being an IRI identifying a datatype that determines how the lexical form maps to a literal value, and
- if and only if the datatype IRI is <http://www.w3.org/1999/02/22-rdf-syntax-ns#langString>, a non-empty language tag. The language tag MUST be well-formed according to section 2.2.9 of [Tags for identifying languages](#).

A literal is a language-tagged string if the third element is present. Lexical representations of language tags MAY be converted to lower case. The value space of language tags is always in lower case.

—

For valid XSD datatypes, the lexical form is optionally normalized at construction time. Default behaviour is set by `rdflib.NORMALIZE_LITERALS` and can be overridden by the `normalize` parameter to `__new__`

Equality and hashing of Literals are done based on the lexical form, i.e.:

```
>>> from rdflib.namespace import XSD
```

```
>>> Literal('01') != Literal('1') # clear - strings differ
True
```

but with data-type they get normalized:

```
>>> Literal('01', datatype=XSD.integer) != Literal('1', datatype=XSD.integer)
False
```

unless disabled:

```
>>> Literal('01', datatype=XSD.integer, normalize=False) != Literal('1',
↳datatype=XSD.integer)
True
```

Value based comparison is possible:

```
>>> Literal('01', datatype=XSD.integer).eq(Literal('1', datatype=XSD.float))
True
```

The `eq` method also provides limited support for basic python types:

```
>>> Literal(1).eq(1) # fine - int compatible with xsd:integer
True
```

(continues on next page)

(continued from previous page)

```

>>> Literal('a').eq('b') # fine - str compatible with plain-lit
False
>>> Literal('a', datatype=XSD.string).eq('a') # fine - str compatible with
↳xsd:string
True
>>> Literal('a').eq(1) # not fine, int incompatible with plain-lit
NotImplemented

```

Greater-than/less-than ordering comparisons are also done in value space, when compatible datatypes are used. Incompatible datatypes are ordered by DT, or by lang-tag. For other nodes the ordering is None < BNode < URIRef < Literal

Any comparison with non-rdflib Node are “NotImplemented” In PY3 this is an error.

```

>>> from rdflib import Literal, XSD
>>> lit2006 = Literal('2006-01-01',datatype=XSD.date)
>>> lit2006.toPython()
datetime.date(2006, 1, 1)
>>> lit2006 < Literal('2007-01-01',datatype=XSD.date)
True
>>> Literal(datetime.utcnow()).datatype
rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#dateTime')
>>> Literal(1) > Literal(2) # by value
False
>>> Literal(1) > Literal(2.0) # by value
False
>>> Literal('1') > Literal(1) # by DT
True
>>> Literal('1') < Literal('1') # by lexical form
False
>>> Literal('a', lang='en') > Literal('a', lang='fr') # by lang-tag
False
>>> Literal(1) > URIRef('foo') # by node-type
True

```

The > < operators will eat this NotImplemented and throw a TypeError (py3k):

```

>>> Literal(1).__gt__(2.0)
NotImplemented

```

`__abs__()`

**Return type**  
*Literal*

```

>>> abs(Literal(-1))
rdflib.term.Literal(u'1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))

```

```

>>> from rdflib.namespace import XSD
>>> abs( Literal("-1", datatype=XSD.integer))
rdflib.term.Literal(u'1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))

```

```
>>> abs(Literal("1"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Not a number; rdflib.term.Literal(u'1')
```

`__add__(val)`

```
>>> from rdflib.namespace import XSD
>>> Literal(1) + 1
rdflib.term.Literal(u'2', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#integer'))
>>> Literal("1") + "1"
rdflib.term.Literal(u'11')
```

```
# Handling dateTime/date/time based operations in Literals >>> a = Literal('2006-01-01T20:50:00',
datatype=XSD.dateTime) >>> b = Literal('P31D', datatype=XSD.duration) >>> (a + b)
rdflib.term.Literal('2006-02-01T20:50:00', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#dateTime'))
>>> from rdflib.namespace import XSD >>> a = Literal('2006-07-01T20:52:00', datatype=XSD.dateTime) >>> b = Literal('P122DT15H58M', datatype=XSD.duration)
>>> (a + b) rdflib.term.Literal('2006-11-01T12:50:00', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#dateTime'))
```

#### Parameters

**val** (*Any*) –

#### Return type

*Literal*

```
__annotations__ = {'_datatype': typing.Optional[rdflib.term.URIRef], '_ill_typed':
typing.Optional[bool], '_language': typing.Optional[str], '_value': typing.Any}
```

`__bool__()`

Is the Literal “True” This is used for if statements, `bool(literal)`, etc.

#### Return type

*bool*

`__eq__(other)`

Literals are only equal to other literals.

“Two literals are equal if and only if all of the following hold: \* The strings of the two lexical forms compare equal, character by character. \* Either both or neither have language tags. \* The language tags, if any, compare equal. \* Either both or neither have datatype URIs. \* The two datatype URIs, if any, compare equal, character by character.” – 6.5.1 Literal Equality (RDF: Concepts and Abstract Syntax)

```
>>> Literal("1", datatype=URIRef("foo")) == Literal("1", datatype=URIRef("foo"))
True
>>> Literal("1", datatype=URIRef("foo")) == Literal("1", datatype=URIRef("foo2
↪"))
False
```

```
>>> Literal("1", datatype=URIRef("foo")) == Literal("2", datatype=URIRef("foo"))
False
>>> Literal("1", datatype=URIRef("foo")) == "asdf"
False
```

(continues on next page)



(continued from previous page)

```

>>> from rdflib import XSD
>>> Literal('2007-01-01', datatype=XSD.date) == Literal('2007-01-01',
↳datatype=XSD.date)
True
>>> Literal('2007-01-01', datatype=XSD.date) == date(2007, 1, 1)
False
>>> Literal("one", lang="en") == Literal("one", lang="en")
True
>>> Literal("hast", lang='en') == Literal("hast", lang='de')
False
>>> Literal("1", datatype=XSD.integer) == Literal(1)
True
>>> Literal("1", datatype=XSD.integer) == Literal("01", datatype=XSD.integer)
True

```

**Parameters****other** (*Any*) –**Return type***bool***\_\_ge\_\_**(*other*)

Return self&gt;=value.

**Parameters****other** (*Any*) –**Return type***bool***\_\_getstate\_\_**()**Return type***Tuple*[*None*, *Dict*[*str*, *Optional*[*str*]]]**\_\_gt\_\_**(*other*)

This implements ordering for Literals, the other comparison methods delegate here

This tries to implement this: <http://www.w3.org/TR/sparql11-query/#modOrderBy>

In short, Literals with compatible data-types are ordered in value space, i.e. &gt;&gt;&gt; from rdflib import XSD

```

>>> Literal(1) > Literal(2) # int/int
False
>>> Literal(2.0) > Literal(1) # double/int
True
>>> from decimal import Decimal
>>> Literal(Decimal("3.3")) > Literal(2.0) # decimal/double
True
>>> Literal(Decimal("3.3")) < Literal(4.0) # decimal/double
True
>>> Literal('b') > Literal('a') # plain lit/plain lit
True
>>> Literal('b') > Literal('a', datatype=XSD.string) # plain lit/xsd:str
True

```

Incompatible datatype mismatches ordered by DT

```
>>> Literal(1) > Literal("2") # int>string
False
```

Langtagged literals by lang tag >>> Literal("a", lang="en") > Literal("a", lang="fr") False

#### Parameters

**other** (*Any*) –

#### Return type

*bool*

**\_\_hash\_\_()**

```
>>> from rdflib.namespace import XSD
>>> a = {Literal('1', datatype=XSD.integer): 'one'}
:rtype: :py:class:`int`
```

```
>>> Literal('1', datatype=XSD.double) in a
False
```

“Called for the key object for dictionary operations, and by the built-in function hash(). Should return a 32-bit integer usable as a hash value for dictionary operations. The only required property is that objects which compare equal have the same hash value; it is advised to somehow mix together (e.g., using exclusive or) the hash values for the components of the object that also play a part in comparison of objects.” – 3.4.1 Basic customization (Python)

“Two literals are equal if and only if all of the following hold: \* The strings of the two lexical forms compare equal, character by character. \* Either both or neither have language tags. \* The language tags, if any, compare equal. \* Either both or neither have datatype URIs. \* The two datatype URIs, if any, compare equal, character by character.” – 6.5.1 Literal Equality (RDF: Concepts and Abstract Syntax)

**\_\_invert\_\_()**

#### Return type

*Literal*

```
>>> ~(Literal(-1))
rdflib.term.Literal(u'0', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#integer'))
```

```
>>> from rdflib.namespace import XSD
>>> ~(Literal("-1", datatype=XSD.integer))
rdflib.term.Literal(u'0', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/XMLSchema#integer'))
```

Not working:

```
>>> ~(Literal("1"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Not a number; rdflib.term.Literal(u'1')
```

**\_\_le\_\_(other)**

```
>>> from rdflib.namespace import XSD
>>> Literal('2007-01-01T10:00:00', datatype=XSD.dateTime
...         ) <= Literal('2007-01-01T10:00:00', datatype=XSD.dateTime)
True
```

**Parameters**

**other** (*Any*) –

**Return type**

*bool*

`__lt__(other)`

Return self<value.

**Parameters**

**other** (*Any*) –

**Return type**

*bool*

`__module__` = 'rdflib.term'

`__neg__()`

```
>>> (- Literal(1))
rdflib.term.Literal(u'-1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
>>> (- Literal(10.5))
rdflib.term.Literal(u'-10.5', datatype=rdflib.term.URIRef(u'http://www.w3.org/
↳2001/XMLSchema#double'))
>>> from rdflib.namespace import XSD
:rtype: :py:class:`~rdflib.term.Literal`
```

```
>>> (- Literal("1", datatype=XSD.integer))
rdflib.term.Literal(u'-1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
```

```
>>> (- Literal("1"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Not a number; rdflib.term.Literal(u'1')
>>>
```

**static** `__new__(cls, lexical_or_value, lang=None, datatype=None, normalize=None)`

**Parameters**

- **lexical\_or\_value** (*Any*) –
- **lang** (*Optional*[*str*]) –
- **datatype** (*Optional*[*str*]) –
- **normalize** (*Optional*[*bool*]) –

**Return type**

*Literal*

`__pos__()`

```
>>> (+ Literal(1))
rdflib.term.Literal(u'1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
>>> (+ Literal(-1))
rdflib.term.Literal(u'-1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
>>> from rdflib.namespace import XSD
:rtype: :py:class:`~rdflib.term.Literal`
```

```
>>> (+ Literal("-1", datatype=XSD.integer))
rdflib.term.Literal(u'-1', datatype=rdflib.term.URIRef(u'http://www.w3.org/2001/
↳XMLSchema#integer'))
```

```
>>> (+ Literal("1"))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Not a number; rdflib.term.Literal(u'1')
```

`__reduce__()`

Helper for pickle.

**Return type**

`Tuple[Type[Literal], Tuple[str, Optional[str], Optional[str]]]`

`__repr__()`

Return repr(self).

**Return type**

`str`

`__setstate__(arg)`

**Parameters**

**arg** (`Tuple[Any, Dict[str, Any]]`) –

**Return type**

`None`

`__slots__ = ('_language', '_datatype', '_value', '_ill_typed')`

`__sub__(val)`

```
>>> from rdflib.namespace import XSD
>>> Literal(2) - 1
rdflib.term.Literal('1', datatype=rdflib.term.URIRef('http://www.w3.org/2001/
↳XMLSchema#integer'))
>>> Literal(1.1) - 1.0
rdflib.term.Literal('0.100000000000000009', datatype=rdflib.term.URIRef('http://
↳www.w3.org/2001/XMLSchema#double'))
>>> Literal(1.1) - 1
rdflib.term.Literal('0.1', datatype=rdflib.term.URIRef('http://www.w3.org/2001/
↳XMLSchema#decimal'))
>>> Literal(1.1, datatype=XSD.float) - Literal(1.0, datatype=XSD.float)
rdflib.term.Literal('0.100000000000000009', datatype=rdflib.term.URIRef('http://
```

(continues on next page)

(continued from previous page)

```

↳www.w3.org/2001/XMLSchema#float'))
>>> Literal("1.1") - 1.0
Traceback (most recent call last):
...
TypeError: Not a number; rdflib.term.Literal('1.1')
>>> Literal(1.1, datatype=XSD.integer) - Literal(1.0, datatype=XSD.integer)
rdflib.term.Literal('0.100000000000000009', datatype=rdflib.term.URIRef('http://
↳www.w3.org/2001/XMLSchema#integer'))

```

```

# Handling dateTime/date/time based operations in Literals >>> a = Literal('2006-01-01T20:50:00',
datatype=XSD.dateTime) >>> b = Literal('2006-02-01T20:50:00', datatype=XSD.dateTime) >>> (b -
a) rdflib.term.Literal('P31D', datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#
duration')) >>> from rdflib.namespace import XSD >>> a = Literal('2006-07-01T20:52:00',
datatype=XSD.dateTime) >>> b = Literal('2006-11-01T12:50:00', datatype=XSD.dateTime)
>>> (a - b) rdflib.term.Literal('-P122DT15H58M', datatype=rdflib.term.URIRef('http://www.
w3.org/2001/XMLSchema#duration')) >>> (b - a) rdflib.term.Literal('P122DT15H58M',
datatype=rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#duration'))

```

**Parameters****val** (*Any*) –**Return type***Literal***property datatype:** *Optional*[*URIRef*]**eq**(*other*)

Compare the value of this literal with something else

Either, with the value of another literal comparisons are then done in literal “value space”, and according to the rules of XSD subtype-substitution/type-promotion

OR, with a python object:

basestring objects can be compared with plain-literals, or those with datatype xsd:string

bool objects with xsd:boolean

a int, long or float with numeric xsd types

isodate date,time,datetime objects with xsd:date,xsd:time or xsd:datetime

Any other operations returns NotImplemented

**Parameters****other** (*Any*) –**Return type***bool***property ill\_typed:** *Optional*[*bool*]

For *recognized datatype IRIs*, this value will be *True* if the literal is ill formed, otherwise it will be *False*. *Literal.value* (i.e. the *literal value*) should always be defined if this property is *False*, but should not be considered reliable if this property is *True*.

If the literal’s datatype is *None* or not in the set of *recognized datatype IRIs* this value will be *None*.

**property language:** *Optional*[*str*]

**n3**(*namespace\_manager=None*)

Returns a representation in the N3 format.

Examples:

```
>>> Literal("foo").n3()
u'"foo"'
```

Strings with newlines or triple-quotes:

```
>>> Literal("foo\nbar").n3()
u'"""foo\nbar"""'
```

```
>>> Literal("'\"").n3()
u'"\'\\"'
```

```
>>> Literal('""').n3()
u'"\\"'\\"'
```

Language:

```
>>> Literal("hello", lang="en").n3()
u'"hello"@en'
```

Datatypes:

```
>>> Literal(1).n3()
u'"1"^^<http://www.w3.org/2001/XMLSchema#integer>'
```

```
>>> Literal(1.0).n3()
u'"1.0"^^<http://www.w3.org/2001/XMLSchema#double>'
```

```
>>> Literal(True).n3()
u'"true"^^<http://www.w3.org/2001/XMLSchema#boolean>'
```

Datatype and language isn't allowed (datatype takes precedence):

```
>>> Literal(1, lang="en").n3()
u'"1"^^<http://www.w3.org/2001/XMLSchema#integer>'
```

Custom datatype:

```
>>> footype = URIRef("http://example.org/ns#foo")
>>> Literal("1", datatype=footype).n3()
u'"1"^^<http://example.org/ns#foo>'
```

Passing a namespace-manager will use it to abbreviate datatype URIs:

```
>>> from rdflib import Graph
>>> Literal(1).n3(Graph().namespace_manager)
u'"1"^^xsd:integer'
```

### Parameters

**namespace\_manager** (*Optional[[NamespaceManager](#)]*) –

**Return type**`str`**neq(*other*)**

A “semantic”/interpreted not equal function, by default, same as `__ne__`

**Parameters**

**other** (*Any*) –

**Return type**`bool`**normalize()**

Returns a new literal with a normalised lexical representation of this literal >>> from rdflib import XSD >>> Literal(“01”, datatype=XSD.integer, normalize=False).normalize() rdflib.term.Literal(u’1’, datatype=rdflib.term.URIRef(u’http://www.w3.org/2001/XMLSchema#integer’))

Illegal lexical forms for the datatype given are simply passed on >>> Literal(“a”, datatype=XSD.integer, normalize=False) rdflib.term.Literal(u’a’, datatype=rdflib.term.URIRef(u’http://www.w3.org/2001/XMLSchema#integer’))

**Return type**`Literal`**toPython()**

Returns an appropriate python datatype derived from this RDF Literal

**Return type**`Any`

**property value:** `Any`

**class** rdflib.Namespace(*value: Union[str, bytes]*)

Bases: `str`

Utility class for quickly generating URIRefs with a common prefix

```
>>> from rdflib.namespace import Namespace
>>> n = Namespace("http://example.org/")
>>> n.Person # as attribute
rdflib.term.URIRef('http://example.org/Person')
>>> n['first-name'] # as item - for things that are not valid python identifiers
rdflib.term.URIRef('http://example.org/first-name')
>>> n.Person in n
True
>>> n2 = Namespace("http://example2.org/")
>>> n.Person in n2
False
```

**\_\_contains\_\_(*ref*)**

Allows to check if a URI is within (starts with) this Namespace.

```
>>> from rdflib import URIRef
>>> namespace = Namespace('http://example.org/')
>>> uri = URIRef('http://example.org/foo')
>>> uri in namespace
True
```

(continues on next page)

(continued from previous page)

```

>>> person_class = namespace['Person']
>>> person_class in namespace
True
>>> obj = URIRef('http://not.example.org/bar')
>>> obj in namespace
False

```

**Parameters****ref** (*str*) –**Return type***bool*

```

__dict__ = mappingproxy({'__module__': 'rdflib.namespace', '__doc__': '\n Utility
class for quickly generating URIRefs with a common prefix\n\n >>> from
rdflib.namespace import Namespace\n >>> n = Namespace("http://example.org/")\n >>>
n.Person # as attribute\n rdflib.term.URIRef(\'http://example.org/Person\')\n >>>
n[\'first-name\'] # as item - for things that are not valid python identifiers\n
rdflib.term.URIRef(\'http://example.org/first-name\')\n >>> n.Person in n\n True\n
>>> n2 = Namespace("http://example2.org/")\n >>> n.Person in n2\n False\n ',
'__new__': <staticmethod object>, 'title': <property object>, 'term': <function
Namespace.term>, '__getitem__': <function Namespace.__getitem__>, '__getattr__':
<function Namespace.__getattr__>, '__repr__': <function Namespace.__repr__>,
'__contains__': <function Namespace.__contains__>, '__dict__': <attribute
'__dict__' of 'Namespace' objects>, '__weakref__': <attribute '__weakref__' of
'Namespace' objects>, '__annotations__': {}})

```

**\_\_getattr\_\_** (*name*)**Parameters****name** (*str*) –**Return type***URIRef***\_\_getitem\_\_** (*key*)

Return self[key].

**Parameters****key** (*str*) –**Return type***URIRef***\_\_module\_\_** = 'rdflib.namespace'**static** **\_\_new\_\_** (*cls, value*)**Parameters****value** (*Union[str, bytes]*) –**Return type***Namespace***\_\_repr\_\_** ()

Return repr(self).



**Return type***str***\_\_weakref\_\_**

list of weak references to the object (if defined)

**term**(*name*)**Parameters****name** (*str*) –**Return type***URIRef***property title:** *URIRef*

Return a version of the string where each word is titlecased.

More specifically, words start with uppercased characters and all remaining cased characters have lower case.

**class** rdflib.ODRL2Bases: *DefinedNamespace*

ODRL Version 2.2

The ODRL Vocabulary and Expression defines a set of concepts and terms (the vocabulary) and encoding mechanism (the expression) for permissions and obligations statements describing digital content usage based on the ODRL Information Model.

Generated from: <https://www.w3.org/ns/odrl/2/ODRL22.ttl> Date: 2020-05-26 14:20:02.352356**Action:** *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Action')**Agreement:** *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Agreement')**All:** *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/All')**All2ndConnections:** *URIRef* =  
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/All2ndConnections')**AllConnections:** *URIRef* =  
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/AllConnections')**AllGroups:** *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/AllGroups')**Assertion:** *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Assertion')**Asset:** *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Asset')**AssetCollection:** *URIRef* =  
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/AssetCollection')**AssetScope:** *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/AssetScope')**ConflictTerm:** *URIRef* =  
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/ConflictTerm')**Constraint:** *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Constraint')**Duty:** *URIRef* = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Duty')

Group: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Group')`

Individual: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Individual')`

LeftOperand: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/LeftOperand')`

LogicalConstraint: `URIRef` =  
`rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/LogicalConstraint')`

Offer: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Offer')`

Operator: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Operator')`

Party: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Party')`

PartyCollection: `URIRef` =  
`rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/PartyCollection')`

PartyScope: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/PartyScope')`

Permission: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Permission')`

Policy: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Policy')`

Privacy: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Privacy')`

Prohibition: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Prohibition')`

Request: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Request')`

RightOperand: `URIRef` =  
`rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/RightOperand')`

Rule: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Rule')`

Set: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Set')`

Ticket: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/Ticket')`

UndefinedTerm: `URIRef` =  
`rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/UndefinedTerm')`

absolutePosition: `URIRef` =  
`rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/absolutePosition')`

absoluteSize: `URIRef` =  
`rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/absoluteSize')`

absoluteSpatialPosition: `URIRef` =  
`rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/absoluteSpatialPosition')`

absoluteTemporalPosition: `URIRef` =  
`rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/absoluteTemporalPosition')`

acceptTracking: `URIRef` =  
`rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/acceptTracking')`

action: `URIRef` = `rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/action')`

```
adHocShare: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/adHocShare')
aggregate: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/aggregate')
andSequence: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/andSequence')
annotate: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/annotate')
anonymize: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/anonymize')
append: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/append')
appendTo: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/appendTo')
archive: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/archive')
assignee: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/assignee')
assigneeOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/assigneeOf')
assigner: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/assigner')
assignerOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/assignerOf')
attachPolicy: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/attachPolicy')
attachSource: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/attachSource')
attribute: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/attribute')
attributedParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/attributedParty')
attributingParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/attributingParty')
commercialize: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/commercialize')
compensate: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/compensate')
compensatedParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/compensatedParty')
compensatingParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/compensatingParty')
concurrentUse: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/concurrentUse')
conflict: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/conflict')
consentedParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/consentedParty')
consentingParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/consentingParty')
```

```
consequence: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/consequence')
constraint: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/constraint')
contractedParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/contractedParty')
contractingParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/contractingParty')
copy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/copy')
core: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/core')
count: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/count')
dataType: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/dataType')
dateTime: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/dateTime')
delayPeriod: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/delayPeriod')
delete: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/delete')
deliveryChannel: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/deliveryChannel')
derive: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/derive')
device: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/device')
digitize: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/digitize')
display: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/display')
distribute: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/distribute')
duty: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/duty')
elapsedTime: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/elapsedTime')
ensureExclusivity: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/ensureExclusivity')
eq: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/eq')
event: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/event')
execute: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/execute')
export: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/export')
extract: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/extract')
extractChar: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/extractChar')
extractPage: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/extractPage')
extractWord: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/extractWord')
```

```

failure: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/failure')
fileFormat: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/fileFormat')
function: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/function')
give: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/give')
grantUse: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/grantUse')
gt: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/gt')
gteq: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/gteq')
hasPart: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/hasPart')
hasPolicy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/hasPolicy')
ignore: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/ignore')
implies: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/implies')
include: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/include')
includedIn: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/includedIn')
index: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/index')
industry: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/industry')
inform: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/inform')
informedParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/informedParty')
informingParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/informingParty')
inheritAllowed: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/inheritAllowed')
inheritFrom: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/inheritFrom')
inheritRelation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/inheritRelation')
install: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/install')
invalid: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/invalid')
isA: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/isA')
isAllOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/isAllOf')
isAnyOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/isAnyOf')
isNoneOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/isNoneOf')
isPartOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/isPartOf')

```

```
language: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/language')
lease: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/lease')
leftOperand: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/leftOperand')
lend: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/lend')
license: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/license')
lt: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/lt')
lteq: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/lteq')
media: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/media')
meteredTime: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/meteredTime')
modify: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/modify')
move: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/move')
neq: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/neq')
nextPolicy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/nextPolicy')
obligation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/obligation')
obtainConsent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/obtainConsent')
operand: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/operand')
operator: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/operator')
output: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/output')
partOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/partOf')
pay: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/pay')
payAmount: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/payAmount')
payeeParty: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/payeeParty')
percentage: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/percentage')
perm: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/perm')
permission: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/permission')
play: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/play')
policyUsage: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/policyUsage')
present: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/present')
preview: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/preview')
print: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/print')
```

```

product: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/product')
profile: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/profile')
prohibit: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/prohibit')
prohibition: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/prohibition')
proximity: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/proximity')
purpose: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/purpose')
read: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/read')
recipient: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/recipient')
refinement: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/refinement')
relation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/relation')
relativePosition: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/relativePosition')
relativeSize: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/relativeSize')
relativeSpatialPosition: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/relativeSpatialPosition')
relativeTemporalPosition: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/relativeTemporalPosition')
remedy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/remedy')
reproduce: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/reproduce')
resolution: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/resolution')
reviewPolicy: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/reviewPolicy')
rightOperand: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/rightOperand')
rightOperandReference: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/rightOperandReference')
scope: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/scope')
secondaryUse: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/secondaryUse')
sell: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/sell')
share: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/share')
shareAlike: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/shareAlike')
source: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/source')

```



```
spatial: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/spatial')

spatialCoordinates: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/spatialCoordinates')

status: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/status')

stream: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/stream')

support: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/support')

synchronize: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/synchronize')

system: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/system')

systemDevice: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/systemDevice')

target: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/target')

textToSpeech: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/textToSpeech')

timeInterval: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/timeInterval')

timedCount: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/timedCount')

trackedParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/trackedParty')

trackingParty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/trackingParty')

transfer: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/transfer')

transform: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/transform')

translate: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/translate')

uid: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/uid')

undefined: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/undefined')

uninstall: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/uninstall')

unit: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/unit')

unitOfCount: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/unitOfCount')

use: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/use')

version: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/version')

virtualLocation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/virtualLocation')

watermark: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/watermark')
```



```

write: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/write')
writeTo: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/writeTo')
xone: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/odrl/2/xone')

class rdflib.ORG
    Bases: DefinedNamespace
    Core organization ontology
    Vocabulary for describing organizational structures, specializable to a broad variety of types of organization.
    Generated from: http://www.w3.org/ns/org# Date: 2020-05-26 14:20:02.908408
    ChangeEvent: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#ChangeEvent')
    FormalOrganization: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#FormalOrganization')
    Head: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#Head')
    Membership: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#Membership')
    Organization: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#Organization')
    OrganizationalCollaboration: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#OrganizationalCollaboration')
    OrganizationalUnit: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#OrganizationalUnit')
    Post: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#Post')
    Role: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#Role')
    Site: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#Site')
    basedAt: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#basedAt')
    changedBy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#changedBy')
    classification: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#classification')
    hasMember: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#hasMember')
    hasMembership: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#hasMembership')
    hasPost: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#hasPost')
    hasPrimarySite: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#hasPrimarySite')
    hasRegisteredSite: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#hasRegisteredSite')
    hasSite: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#hasSite')

```

```

hasSubOrganization: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#hasSubOrganization')

hasUnit: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#hasUnit')

headOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#headOf')

heldBy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#heldBy')

holds: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#holds')

identifier: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#identifier')

linkedTo: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#linkedTo')

location: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#location')

member: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#member')

memberDuring: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#memberDuring')

memberOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#memberOf')

organization: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#organization')

originalOrganization: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#originalOrganization')

postIn: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#postIn')

purpose: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#purpose')

remuneration: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#remuneration')

reportsTo: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#reportsTo')

resultedFrom: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#resultedFrom')

resultingOrganization: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#resultingOrganization')

role: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#role')

roleProperty: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#roleProperty')

siteAddress: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#siteAddress')

siteOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#siteOf')

subOrganizationOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#subOrganizationOf')

transitiveSubOrganizationOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/org#transitiveSubOrganizationOf')

unitOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/org#unitOf')

```

**class rdflib.OWL**

Bases: *DefinedNamespace*

The OWL 2 Schema vocabulary (OWL 2)

This ontology partially describes the built-in classes and properties that together form the basis of the RDF/XML syntax of OWL 2. The content of this ontology is based on Tables 6.1 and 6.2 in Section 6.4 of the OWL 2 RDF-Based Semantics specification, available at <http://www.w3.org/TR/owl2-rdf-based-semantics/>. Please note that those tables do not include the different annotations (labels, comments and `rdfs:isDefinedBy` links) used in this file. Also note that the descriptions provided in this ontology do not provide a complete and correct formal description of either the syntax or the semantics of the introduced terms (please see the OWL 2 recommendations for the complete and normative specifications). Furthermore, the information provided by this ontology may be misleading if not used with care. This ontology SHOULD NOT be imported into OWL ontologies. Importing this file into an OWL 2 DL ontology will cause it to become an OWL 2 Full ontology and may have other, unexpected, consequences.

Generated from: <http://www.w3.org/2002/07/owl#> Date: 2020-05-26 14:20:03.193795

**AllDifferent:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#AllDifferent')`

**AllDisjointClasses:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#AllDisjointClasses')`

**AllDisjointProperties:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#AllDisjointProperties')`

**Annotation:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2002/07/owl#Annotation')`

**AnnotationProperty:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#AnnotationProperty')`

**AsymmetricProperty:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#AsymmetricProperty')`

**Axiom:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2002/07/owl#Axiom')`

**Class:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2002/07/owl#Class')`

**DataRange:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2002/07/owl#DataRange')`

**DatatypeProperty:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#DatatypeProperty')`

**DeprecatedClass:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#DeprecatedClass')`

**DeprecatedProperty:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#DeprecatedProperty')`

**FunctionalProperty:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#FunctionalProperty')`

**InverseFunctionalProperty:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#InverseFunctionalProperty')`

**IrreflexiveProperty:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2002/07/owl#IrreflexiveProperty')`

```
NamedIndividual: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#NamedIndividual')  
  
NegativePropertyAssertion: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#NegativePropertyAssertion')  
  
Nothing: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#Nothing')  
  
ObjectProperty: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#ObjectProperty')  
  
Ontology: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#Ontology')  
  
OntologyProperty: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#OntologyProperty')  
  
ReflexiveProperty: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#ReflexiveProperty')  
  
Restriction: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#Restriction')  
  
SymmetricProperty: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#SymmetricProperty')  
  
Thing: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#Thing')  
  
TransitiveProperty: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#TransitiveProperty')  
  
allValuesFrom: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#allValuesFrom')  
  
annotatedProperty: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#annotatedProperty')  
  
annotatedSource: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#annotatedSource')  
  
annotatedTarget: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#annotatedTarget')  
  
assertionProperty: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#assertionProperty')  
  
backwardCompatibleWith: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#backwardCompatibleWith')  
  
bottomDataProperty: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#bottomDataProperty')  
  
bottomObjectProperty: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#bottomObjectProperty')  
  
cardinality: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#cardinality')
```

```

complementOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#complementOf')

datatypeComplementOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#datatypeComplementOf')

deprecated: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#deprecated')

differentFrom: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#differentFrom')

disjointUnionOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#disjointUnionOf')

disjointWith: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#disjointWith')

distinctMembers: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#distinctMembers')

equivalentClass: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#equivalentClass')

equivalentProperty: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#equivalentProperty')

hasKey: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#hasKey')

hasSelf: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#hasSelf')

hasValue: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#hasValue')

imports: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#imports')

incompatibleWith: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#incompatibleWith')

intersectionOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#intersectionOf')

inverseOf: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#inverseOf')

maxCardinality: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#maxCardinality')

maxQualifiedCardinality: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#maxQualifiedCardinality')

members: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#members')

minCardinality: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#minCardinality')

minQualifiedCardinality: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#minQualifiedCardinality')

onClass: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#onClass')

```

```

onDataRange: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#onDataRange')

onDatatype: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#onDatatype')

onProperties: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#onProperties')

onProperty: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#onProperty')

oneOf: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#oneOf')

priorVersion: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#priorVersion')

propertyChainAxiom: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#propertyChainAxiom')

propertyDisjointWith: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#propertyDisjointWith')

qualifiedCardinality: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#qualifiedCardinality')

rational: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#rational')

real: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#real')

sameAs: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#sameAs')

someValuesFrom: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#someValuesFrom')

sourceIndividual: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#sourceIndividual')

targetIndividual: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#targetIndividual')

targetValue: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#targetValue')

topDataProperty: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#topDataProperty')

topObjectProperty: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#topObjectProperty')

unionOf: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#unionOf')

versionIRI: URIRef = rdflib.term.URIRef('http://www.w3.org/2002/07/owl#versionIRI')

versionInfo: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#versionInfo')

withRestrictions: URIRef =
rdflib.term.URIRef('http://www.w3.org/2002/07/owl#withRestrictions')

```

**class rdflib.PROF**Bases: *DefinedNamespace*

Profiles Vocabulary

This vocabulary is for describing relationships between standards/specifications, profiles of them and supporting artifacts such as validating resources. This model starts with [[\(http://dublincore.org/2012/06/14/dcterms#Standard{}\)](http://dublincore.org/2012/06/14/dcterms#Standard{})](*dct:Standard*) entities which can either be Base Specifications (a standard not profiling any other Standard) or Profiles (Standards which do profile others). Base Specifications or Profiles can have Resource Descriptors associated with them that defines implementing rules for the it. Resource Descriptors must indicate the role they play (to guide, to validate etc.) and the formalism they adhere to (*dct:format*) to allow for content negotiation. A vocabulary of Resource Roles are provided alongside this vocabulary but that list is extensible.

Generated from: <https://www.w3.org/ns/dx/prof/profilesont.ttl> Date: 2020-05-26 14:20:03.542924

Profile: *URIRef* = `rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/Profile')`

ResourceDescriptor: *URIRef* =  
`rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/ResourceDescriptor')`

ResourceRole: *URIRef* =  
`rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/ResourceRole')`

hasArtifact: *URIRef* =  
`rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/hasArtifact')`

hasResource: *URIRef* =  
`rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/hasResource')`

hasRole: *URIRef* = `rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/hasRole')`

hasToken: *URIRef* = `rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/hasToken')`

isInheritedFrom: *URIRef* =  
`rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/isInheritedFrom')`

isProfileOf: *URIRef* =  
`rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/isProfileOf')`

isTransitiveProfileOf: *URIRef* =  
`rdflib.term.URIRef('http://www.w3.org/ns/dx/prof/isTransitiveProfileOf')`

**class rdflib.PROV**Bases: *DefinedNamespace*

W3C PROVenance Interchange Ontology (PROV-O)

This document is published by the Provenance Working Group ([http://www.w3.org/2011/prov/wiki/Main\\_Page](http://www.w3.org/2011/prov/wiki/Main_Page)). If you wish to make comments regarding this document, please send them to [public-prov-comments@w3.org](mailto:public-prov-comments@w3.org) (subscribe [public-prov-comments-request@w3.org](mailto:public-prov-comments-request@w3.org), archives <http://lists.w3.org/Archives/Public/public-prov-comments/>). All feedback is welcome.

PROV Access and Query Ontology

This document is published by the Provenance Working Group ([http://www.w3.org/2011/prov/wiki/Main\\_Page](http://www.w3.org/2011/prov/wiki/Main_Page)). If you wish to make comments regarding this document, please send them to [public-prov-comments@w3.org](mailto:public-prov-comments@w3.org) (subscribe [public-prov-comments-request@w3.org](mailto:public-prov-comments-request@w3.org), archives <http://lists.w3.org/Archives/Public/public-prov-comments/>). All feedback is welcome.

Dublin Core extensions of the W3C PROVenance Interchange Ontology (PROV-O)



This document is published by the Provenance Working Group ([http://www.w3.org/2011/prov/wiki/Main\\_Page](http://www.w3.org/2011/prov/wiki/Main_Page)). If you wish to make comments regarding this document, please send them to [public-prov-comments@w3.org](mailto:public-prov-comments@w3.org) (subscribe [public-prov-comments-request@w3.org](mailto:public-prov-comments-request@w3.org), archives <http://lists.w3.org/Archives/Public/public-prov-comments/>). All feedback is welcome.

#### W3C PROV Linking Across Provenance Bundles Ontology (PROV-LINKS)

This document is published by the Provenance Working Group ([http://www.w3.org/2011/prov/wiki/Main\\_Page](http://www.w3.org/2011/prov/wiki/Main_Page)). If you wish to make comments regarding this document, please send them to [public-prov-comments@w3.org](mailto:public-prov-comments@w3.org) (subscribe [public-prov-comments-request@w3.org](mailto:public-prov-comments-request@w3.org), archives <http://lists.w3.org/Archives/Public/public-prov-comments/>). All feedback is welcome.

#### W3C PROVenance Interchange Ontology (PROV-O) Dictionary Extension

This document is published by the Provenance Working Group ([http://www.w3.org/2011/prov/wiki/Main\\_Page](http://www.w3.org/2011/prov/wiki/Main_Page)). If you wish to make comments regarding this document, please send them to [public-prov-comments@w3.org](mailto:public-prov-comments@w3.org) (subscribe [public-prov-comments-request@w3.org](mailto:public-prov-comments-request@w3.org), archives <http://lists.w3.org/Archives/Public/public-prov-comments/>). All feedback is welcome.

#### W3C PROVenance Interchange

This document is published by the Provenance Working Group ([http://www.w3.org/2011/prov/wiki/Main\\_Page](http://www.w3.org/2011/prov/wiki/Main_Page)). If you wish to make comments regarding this document, please send them to [public-prov-comments@w3.org](mailto:public-prov-comments@w3.org) (subscribe [public-prov-comments-request@w3.org](mailto:public-prov-comments-request@w3.org), archives <http://lists.w3.org/Archives/Public/public-prov-comments/>). All feedback is welcome.

Generated from: <http://www.w3.org/ns/prov> Date: 2020-05-26 14:20:04.650279

**Accept:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Accept')`

**Activity:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Activity')`

**ActivityInfluence:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#ActivityInfluence')`

**Agent:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Agent')`

**AgentInfluence:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#AgentInfluence')`

**Association:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Association')`

**Attribution:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Attribution')`

**Bundle:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Bundle')`

**Collection:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Collection')`

**Communication:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Communication')`

**Contribute:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Contribute')`

**Contributor:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Contributor')`

**Copyright:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Copyright')`

**Create:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Create')`

**Creator:** `URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Creator')`



```
Delegation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Delegation')
Derivation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Derivation')
Dictionary: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Dictionary')
DirectQueryService: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#DirectQueryService')
EmptyCollection: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#EmptyCollection')
EmptyDictionary: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#EmptyDictionary')
End: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#End')
Entity: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Entity')
EntityInfluence: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#EntityInfluence')
Generation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Generation')
Influence: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Influence')
Insertion: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Insertion')
InstantaneousEvent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#InstantaneousEvent')
Invalidation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Invalidation')
KeyEntityPair: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#KeyEntityPair')
Location: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Location')
Modify: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Modify')
Organization: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Organization')
Person: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Person')
Plan: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Plan')
PrimarySource: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#PrimarySource')
Publish: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Publish')
Publisher: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Publisher')
Quotation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Quotation')
Removal: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Removal')
Replace: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Replace')
```

```
Revision: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Revision')

RightsAssignment: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#RightsAssignment')

RightsHolder: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#RightsHolder')

Role: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Role')

ServiceDescription: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#ServiceDescription')

SoftwareAgent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#SoftwareAgent')

Start: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Start')

Submit: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Submit')

Usage: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#Usage')

actedOnBehalfOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#actedOnBehalfOf')

activity: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#activity')

activityOfInfluence: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#activityOfInfluence')

agent: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#agent')

agentOfInfluence: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#agentOfInfluence')

alternateOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#alternateOf')

aq: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#aq')

asInBundle: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#asInBundle')

atLocation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#atLocation')

atTime: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#atTime')

category: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#category')

component: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#component')

constraints: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#constraints')

contributed: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#contributed')

definition: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#definition')

derivedByInsertionFrom: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#derivedByInsertionFrom')

derivedByRemovalFrom: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#derivedByRemovalFrom')
```

```
describesService: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#describesService')  
  
dictionary: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#dictionary')  
  
dm: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#dm')  
  
editorialNote: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#editorialNote')  
  
editorsDefinition: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#editorsDefinition')  
  
ended: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#ended')  
  
endedAtTime: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#endedAtTime')  
  
entity: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#entity')  
  
entityOfInfluence: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#entityOfInfluence')  
  
generalizationOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#generalizationOf')  
  
generated: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#generated')  
  
generatedAsDerivation: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#generatedAsDerivation')  
  
generatedAtTime: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#generatedAtTime')  
  
hadActivity: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#hadActivity')  
  
hadDelegate: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#hadDelegate')  
  
hadDerivation: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#hadDerivation')  
  
hadDictionaryMember: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#hadDictionaryMember')  
  
hadGeneration: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#hadGeneration')  
  
hadInfluence: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#hadInfluence')  
  
hadMember: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#hadMember')  
  
hadPlan: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#hadPlan')  
  
hadPrimarySource: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#hadPrimarySource')  
  
hadRevision: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#hadRevision')  
  
hadRole: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#hadRole')
```

```
hadUsage: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#hadUsage')
has_anchor: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#has_anchor')
has_provenance: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#has_provenance')
has_query_service: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#has_query_service')
influenced: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#influenced')
influencer: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#influencer')
informed: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#informed')
insertedKeyEntityPair: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#insertedKeyEntityPair')
invalidated: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#invalidated')
invalidatedAtTime: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#invalidatedAtTime')
inverse: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#inverse')
locationOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#locationOf')
mentionOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#mentionOf')
n: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#n')
order: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#order')
pairEntity: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#pairEntity')
pairKey: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#pairKey')
pingback: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#pingback')
provenanceUriTemplate: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#provenanceUriTemplate')
qualifiedAssociation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedAssociation')
qualifiedAssociationOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedAssociationOf')
qualifiedAttribution: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedAttribution')
qualifiedAttributionOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedAttributionOf')
qualifiedCommunication: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedCommunication')
```

```
qualifiedCommunicationOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedCommunicationOf')  
  
qualifiedDelegation: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedDelegation')  
  
qualifiedDelegationOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedDelegationOf')  
  
qualifiedDerivation: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedDerivation')  
  
qualifiedDerivationOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedDerivationOf')  
  
qualifiedEnd: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedEnd')  
  
qualifiedEndOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedEndOf')  
  
qualifiedForm: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedForm')  
  
qualifiedGeneration: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedGeneration')  
  
qualifiedGenerationOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedGenerationOf')  
  
qualifiedInfluence: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedInfluence')  
  
qualifiedInfluenceOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedInfluenceOf')  
  
qualifiedInsertion: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedInsertion')  
  
qualifiedInvalidation: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedInvalidation')  
  
qualifiedInvalidationOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedInvalidationOf')  
  
qualifiedPrimarySource: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedPrimarySource')  
  
qualifiedQuotation: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedQuotation')  
  
qualifiedQuotationOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedQuotationOf')  
  
qualifiedRemoval: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedRemoval')
```

```
qualifiedRevision: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedRevision')  
  
qualifiedSourceOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedSourceOf')  
  
qualifiedStart: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedStart')  
  
qualifiedStartOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedStartOf')  
  
qualifiedUsage: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedUsage')  
  
qualifiedUsingActivity: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#qualifiedUsingActivity')  
  
quotedAs: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#quotedAs')  
  
removedKey: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#removedKey')  
  
revisedEntity: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#revisedEntity')  
  
sharesDefinitionWith: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#sharesDefinitionWith')  
  
specializationOf: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#specializationOf')  
  
started: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#started')  
  
startedAtTime: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#startedAtTime')  
  
todo: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#todo')  
  
unqualifiedForm: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#unqualifiedForm')  
  
used: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#used')  
  
value: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#value')  
  
wasActivityOfInfluence: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasActivityOfInfluence')  
  
wasAssociateFor: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasAssociateFor')  
  
wasAssociatedWith: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasAssociatedWith')  
  
wasAttributedTo: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasAttributedTo')
```

```

wasDerivedFrom: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasDerivedFrom')

wasEndedBy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#wasEndedBy')

wasGeneratedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasGeneratedBy')

wasInfluencedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasInfluencedBy')

wasInformedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasInformedBy')

wasInvalidatedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasInvalidatedBy')

wasMemberOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#wasMemberOf')

wasPlanOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#wasPlanOf')

wasPrimarySourceOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasPrimarySourceOf')

wasQuotedFrom: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasQuotedFrom')

wasRevisionOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasRevisionOf')

wasRoleIn: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#wasRoleIn')

wasStartedBy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#wasStartedBy')

wasUsedBy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/prov#wasUsedBy')

wasUsedInDerivation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/prov#wasUsedInDerivation')

```

```
class rdflib.QB
```

Bases: *DefinedNamespace*

Vocabulary for multi-dimensional (e.g. statistical) data publishing

This vocabulary allows multi-dimensional data, such as statistics, to be published in RDF. It is based on the core information model from SDMX (and thus also DDI).

Generated from: <http://purl.org/linked-data/cube#> Date: 2020-05-26 14:20:05.485176

```

Attachable: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#Attachable')

```

```

AttributeProperty: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#AttributeProperty')

```

```

CodedProperty: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#CodedProperty')

```



```
ComponentProperty: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#ComponentProperty')  
  
ComponentSet: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#ComponentSet')  
  
ComponentSpecification: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#ComponentSpecification')  
  
DataSet: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#DataSet')  
  
DataStructureDefinition: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#DataStructureDefinition')  
  
DimensionProperty: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#DimensionProperty')  
  
HierarchicalCodeList: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#HierarchicalCodeList')  
  
MeasureProperty: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#MeasureProperty')  
  
Observation: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#Observation')  
  
ObservationGroup: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#ObservationGroup')  
  
Slice: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#Slice')  
  
SliceKey: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#SliceKey')  
  
attribute: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#attribute')  
  
codeList: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#codeList')  
  
component: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#component')  
  
componentAttachment: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#componentAttachment')  
  
componentProperty: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#componentProperty')  
  
componentRequired: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#componentRequired')  
  
concept: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#concept')  
  
dataSet: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#dataSet')  
  
dimension: URIRef =  
rdflib.term.URIRef('http://purl.org/linked-data/cube#dimension')
```



```

hierarchyRoot: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#hierarchyRoot')

measure: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#measure')

measureDimension: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#measureDimension')

measureType: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#measureType')

observation: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#observation')

observationGroup: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#observationGroup')

order: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#order')

parentChildProperty: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#parentChildProperty')

slice: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#slice')

sliceKey: URIRef = rdflib.term.URIRef('http://purl.org/linked-data/cube#sliceKey')

sliceStructure: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#sliceStructure')

structure: URIRef =
rdflib.term.URIRef('http://purl.org/linked-data/cube#structure')

```

```
class rdflib.RDF
```

Bases: *DefinedNamespace*

The RDF Concepts Vocabulary (RDF)

This is the RDF Schema for the RDF vocabulary terms in the RDF Namespace, defined in RDF 1.1 Concepts.

Generated from: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> Date: 2020-05-26 14:20:05.642859

dc:date "2019-12-16"

```
Alt: URIRef = rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#Alt')
```

```
Bag: URIRef = rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag')
```

```
CompoundLiteral: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#CompoundLiteral')
```

```
HTML: URIRef = rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#HTML')
```

```
JSON: URIRef = rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#JSON')
```

```
List: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#List')
```

```
PlainLiteral: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#PlainLiteral')
```

```
Property: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#Property')

Seq: URIRef = rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq')

Statement: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement')

XMLLiteral: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#XMLLiteral')

direction: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#direction')

first: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#first')

langString: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#langString')

language: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#language')

nil: URIRef = rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#nil')

object: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#object')

predicate: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#predicate')

rest: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#rest')

subject: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#subject')

type: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#type')

value: URIRef =
rdflib.term.URIRef('http://www.w3.org/1999/02/22-rdf-syntax-ns#value')
```

```
class rdflib.RDFS
```

```
    Bases: DefinedNamespace
```

```
    The RDF Schema vocabulary (RDFS)
```

```
    Generated from: http://www.w3.org/2000/01/rdf-schema# Date: 2020-05-26 14:20:05.794866
```

```
    Class: URIRef = rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#Class')
```

```
    Container: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#Container')
```

```
    ContainerMembershipProperty: URIRef = rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#ContainerMembershipProperty')
```

```

Datatype: URIRef =
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#Datatype')

Literal: URIRef =
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#Literal')

Resource: URIRef =
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#Resource')

comment: URIRef =
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#comment')

domain: URIRef = rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#domain')

isDefinedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#isDefinedBy')

label: URIRef = rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#label')

member: URIRef = rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#member')

range: URIRef = rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#range')

seeAlso: URIRef =
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#seeAlso')

subClassOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#subClassOf')

subPropertyOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/2000/01/rdf-schema#subPropertyOf')

```

```
class rdflib.SDO
```

```
    Bases: DefinedNamespace
```

```
    schema.org namespace elements
```

```
    3DModel, True, False & yield are not available as they collide with Python terms
```

```
    Generated from: https://schema.org/version/latest/schemaorg-current-https.jsonld Date: 2021-12-01 By:
    Nicholas J. Car
```

```
    AMRadioChannel: URIRef = rdflib.term.URIRef('https://schema.org/AMRadioChannel')
```

```
    APIReference: URIRef = rdflib.term.URIRef('https://schema.org/APIReference')
```

```
    Abdomen: URIRef = rdflib.term.URIRef('https://schema.org/Abdomen')
```

```
    AboutPage: URIRef = rdflib.term.URIRef('https://schema.org/AboutPage')
```

```
    AcceptAction: URIRef = rdflib.term.URIRef('https://schema.org/AcceptAction')
```

```
    Accommodation: URIRef = rdflib.term.URIRef('https://schema.org/Accommodation')
```

```
    AccountingService: URIRef =
rdflib.term.URIRef('https://schema.org/AccountingService')
```

```
    AchieveAction: URIRef = rdflib.term.URIRef('https://schema.org/AchieveAction')
```

```
Action: URIRef = rdflib.term.URIRef('https://schema.org/Action')

ActionAccessSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/ActionAccessSpecification')

ActionStatusType: URIRef =
rdflib.term.URIRef('https://schema.org/ActionStatusType')

ActivateAction: URIRef = rdflib.term.URIRef('https://schema.org/ActivateAction')

ActivationFee: URIRef = rdflib.term.URIRef('https://schema.org/ActivationFee')

ActiveActionStatus: URIRef =
rdflib.term.URIRef('https://schema.org/ActiveActionStatus')

ActiveNotRecruiting: URIRef =
rdflib.term.URIRef('https://schema.org/ActiveNotRecruiting')

AddAction: URIRef = rdflib.term.URIRef('https://schema.org/AddAction')

AdministrativeArea: URIRef =
rdflib.term.URIRef('https://schema.org/AdministrativeArea')

AdultEntertainment: URIRef =
rdflib.term.URIRef('https://schema.org/AdultEntertainment')

AdvertiserContentArticle: URIRef =
rdflib.term.URIRef('https://schema.org/AdvertiserContentArticle')

AerobicActivity: URIRef = rdflib.term.URIRef('https://schema.org/AerobicActivity')

AggregateOffer: URIRef = rdflib.term.URIRef('https://schema.org/AggregateOffer')

AggregateRating: URIRef = rdflib.term.URIRef('https://schema.org/AggregateRating')

AgreeAction: URIRef = rdflib.term.URIRef('https://schema.org/AgreeAction')

Airline: URIRef = rdflib.term.URIRef('https://schema.org/Airline')

Airport: URIRef = rdflib.term.URIRef('https://schema.org/Airport')

AlbumRelease: URIRef = rdflib.term.URIRef('https://schema.org/AlbumRelease')

AlignmentObject: URIRef = rdflib.term.URIRef('https://schema.org/AlignmentObject')

AllWheelDriveConfiguration: URIRef =
rdflib.term.URIRef('https://schema.org/AllWheelDriveConfiguration')

AllergiesHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/AllergiesHealthAspect')

AllocateAction: URIRef = rdflib.term.URIRef('https://schema.org/AllocateAction')

AmpStory: URIRef = rdflib.term.URIRef('https://schema.org/AmpStory')

AmusementPark: URIRef = rdflib.term.URIRef('https://schema.org/AmusementPark')

AnaerobicActivity: URIRef =
rdflib.term.URIRef('https://schema.org/AnaerobicActivity')
```

```
AnalysisNewsArticle: URIRef =  
rdflib.term.URIRef('https://schema.org/AnalysisNewsArticle')  
  
AnatomicalStructure: URIRef =  
rdflib.term.URIRef('https://schema.org/AnatomicalStructure')  
  
AnatomicalSystem: URIRef =  
rdflib.term.URIRef('https://schema.org/AnatomicalSystem')  
  
Anesthesia: URIRef = rdflib.term.URIRef('https://schema.org/Anesthesia')  
  
AnimalShelter: URIRef = rdflib.term.URIRef('https://schema.org/AnimalShelter')  
  
Answer: URIRef = rdflib.term.URIRef('https://schema.org/Answer')  
  
Apartment: URIRef = rdflib.term.URIRef('https://schema.org/Apartment')  
  
ApartmentComplex: URIRef =  
rdflib.term.URIRef('https://schema.org/ApartmentComplex')  
  
Appearance: URIRef = rdflib.term.URIRef('https://schema.org/Appearance')  
  
AppendAction: URIRef = rdflib.term.URIRef('https://schema.org/AppendAction')  
  
ApplyAction: URIRef = rdflib.term.URIRef('https://schema.org/ApplyAction')  
  
ApprovedIndication: URIRef =  
rdflib.term.URIRef('https://schema.org/ApprovedIndication')  
  
Aquarium: URIRef = rdflib.term.URIRef('https://schema.org/Aquarium')  
  
ArchiveComponent: URIRef =  
rdflib.term.URIRef('https://schema.org/ArchiveComponent')  
  
ArchiveOrganization: URIRef =  
rdflib.term.URIRef('https://schema.org/ArchiveOrganization')  
  
ArriveAction: URIRef = rdflib.term.URIRef('https://schema.org/ArriveAction')  
  
ArtGallery: URIRef = rdflib.term.URIRef('https://schema.org/ArtGallery')  
  
Artery: URIRef = rdflib.term.URIRef('https://schema.org/Artery')  
  
Article: URIRef = rdflib.term.URIRef('https://schema.org/Article')  
  
AskAction: URIRef = rdflib.term.URIRef('https://schema.org/AskAction')  
  
AskPublicNewsArticle: URIRef =  
rdflib.term.URIRef('https://schema.org/AskPublicNewsArticle')  
  
AssessAction: URIRef = rdflib.term.URIRef('https://schema.org/AssessAction')  
  
AssignAction: URIRef = rdflib.term.URIRef('https://schema.org/AssignAction')  
  
Atlas: URIRef = rdflib.term.URIRef('https://schema.org/Atlas')  
  
Attorney: URIRef = rdflib.term.URIRef('https://schema.org/Attorney')  
  
Audience: URIRef = rdflib.term.URIRef('https://schema.org/Audience')
```

```
AudioObject: URIRef = rdflib.term.URIRef('https://schema.org/AudioObject')

AudioObjectSnapshot: URIRef =
rdflib.term.URIRef('https://schema.org/AudioObjectSnapshot')

Audiobook: URIRef = rdflib.term.URIRef('https://schema.org/Audiobook')

AudiobookFormat: URIRef = rdflib.term.URIRef('https://schema.org/AudiobookFormat')

AuthoritativeLegalValue: URIRef =
rdflib.term.URIRef('https://schema.org/AuthoritativeLegalValue')

AuthorizeAction: URIRef = rdflib.term.URIRef('https://schema.org/AuthorizeAction')

AutoBodyShop: URIRef = rdflib.term.URIRef('https://schema.org/AutoBodyShop')

AutoDealer: URIRef = rdflib.term.URIRef('https://schema.org/AutoDealer')

AutoPartsStore: URIRef = rdflib.term.URIRef('https://schema.org/AutoPartsStore')

AutoRental: URIRef = rdflib.term.URIRef('https://schema.org/AutoRental')

AutoRepair: URIRef = rdflib.term.URIRef('https://schema.org/AutoRepair')

AutoWash: URIRef = rdflib.term.URIRef('https://schema.org/AutoWash')

AutomatedTeller: URIRef = rdflib.term.URIRef('https://schema.org/AutomatedTeller')

AutomotiveBusiness: URIRef =
rdflib.term.URIRef('https://schema.org/AutomotiveBusiness')

Ayurvedic: URIRef = rdflib.term.URIRef('https://schema.org/Ayurvedic')

BackOrder: URIRef = rdflib.term.URIRef('https://schema.org/BackOrder')

BackgroundNewsArticle: URIRef =
rdflib.term.URIRef('https://schema.org/BackgroundNewsArticle')

Bacteria: URIRef = rdflib.term.URIRef('https://schema.org/Bacteria')

Bakery: URIRef = rdflib.term.URIRef('https://schema.org/Bakery')

Balance: URIRef = rdflib.term.URIRef('https://schema.org/Balance')

BankAccount: URIRef = rdflib.term.URIRef('https://schema.org/BankAccount')

BankOrCreditUnion: URIRef =
rdflib.term.URIRef('https://schema.org/BankOrCreditUnion')

BarOrPub: URIRef = rdflib.term.URIRef('https://schema.org/BarOrPub')

Barcode: URIRef = rdflib.term.URIRef('https://schema.org/Barcode')

BasicIncome: URIRef = rdflib.term.URIRef('https://schema.org/BasicIncome')

Beach: URIRef = rdflib.term.URIRef('https://schema.org/Beach')

BeautySalon: URIRef = rdflib.term.URIRef('https://schema.org/BeautySalon')

BedAndBreakfast: URIRef = rdflib.term.URIRef('https://schema.org/BedAndBreakfast')
```

```
BedDetails: URIRef = rdflib.term.URIRef('https://schema.org/BedDetails')
BedType: URIRef = rdflib.term.URIRef('https://schema.org/BedType')
BefriendAction: URIRef = rdflib.term.URIRef('https://schema.org/BefriendAction')
BenefitsHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/BenefitsHealthAspect')
BikeStore: URIRef = rdflib.term.URIRef('https://schema.org/BikeStore')
BioChemEntity: URIRef = rdflib.term.URIRef('https://schema.org/BioChemEntity')
Blog: URIRef = rdflib.term.URIRef('https://schema.org/Blog')
BlogPosting: URIRef = rdflib.term.URIRef('https://schema.org/BlogPosting')
BloodTest: URIRef = rdflib.term.URIRef('https://schema.org/BloodTest')
BoardingPolicyType: URIRef =
rdflib.term.URIRef('https://schema.org/BoardingPolicyType')
BoatReservation: URIRef = rdflib.term.URIRef('https://schema.org/BoatReservation')
BoatTerminal: URIRef = rdflib.term.URIRef('https://schema.org/BoatTerminal')
BoatTrip: URIRef = rdflib.term.URIRef('https://schema.org/BoatTrip')
BodyMeasurementArm: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementArm')
BodyMeasurementBust: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementBust')
BodyMeasurementChest: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementChest')
BodyMeasurementFoot: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementFoot')
BodyMeasurementHand: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementHand')
BodyMeasurementHead: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementHead')
BodyMeasurementHeight: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementHeight')
BodyMeasurementHips: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementHips')
BodyMeasurementInsideLeg: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementInsideLeg')
BodyMeasurementNeck: URIRef =
rdflib.term.URIRef('https://schema.org/BodyMeasurementNeck')
```



```
BodyMeasurementTypeEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/BodyMeasurementTypeEnumeration')  
  
BodyMeasurementUnderbust: URIRef =  
rdflib.term.URIRef('https://schema.org/BodyMeasurementUnderbust')  
  
BodyMeasurementWaist: URIRef =  
rdflib.term.URIRef('https://schema.org/BodyMeasurementWaist')  
  
BodyMeasurementWeight: URIRef =  
rdflib.term.URIRef('https://schema.org/BodyMeasurementWeight')  
  
BodyOfWater: URIRef = rdflib.term.URIRef('https://schema.org/BodyOfWater')  
  
Bone: URIRef = rdflib.term.URIRef('https://schema.org/Bone')  
  
Book: URIRef = rdflib.term.URIRef('https://schema.org/Book')  
  
BookFormatType: URIRef = rdflib.term.URIRef('https://schema.org/BookFormatType')  
  
BookSeries: URIRef = rdflib.term.URIRef('https://schema.org/BookSeries')  
  
BookStore: URIRef = rdflib.term.URIRef('https://schema.org/BookStore')  
  
BookmarkAction: URIRef = rdflib.term.URIRef('https://schema.org/BookmarkAction')  
  
Boolean: URIRef = rdflib.term.URIRef('https://schema.org/Boolean')  
  
BorrowAction: URIRef = rdflib.term.URIRef('https://schema.org/BorrowAction')  
  
BowlingAlley: URIRef = rdflib.term.URIRef('https://schema.org/BowlingAlley')  
  
BrainStructure: URIRef = rdflib.term.URIRef('https://schema.org/BrainStructure')  
  
Brand: URIRef = rdflib.term.URIRef('https://schema.org/Brand')  
  
BreadcrumbList: URIRef = rdflib.term.URIRef('https://schema.org/BreadcrumbList')  
  
Brewery: URIRef = rdflib.term.URIRef('https://schema.org/Brewery')  
  
Bridge: URIRef = rdflib.term.URIRef('https://schema.org/Bridge')  
  
BroadcastChannel: URIRef =  
rdflib.term.URIRef('https://schema.org/BroadcastChannel')  
  
BroadcastEvent: URIRef = rdflib.term.URIRef('https://schema.org/BroadcastEvent')  
  
BroadcastFrequencySpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/BroadcastFrequencySpecification')  
  
BroadcastRelease: URIRef =  
rdflib.term.URIRef('https://schema.org/BroadcastRelease')  
  
BroadcastService: URIRef =  
rdflib.term.URIRef('https://schema.org/BroadcastService')  
  
BrokerageAccount: URIRef =  
rdflib.term.URIRef('https://schema.org/BrokerageAccount')
```



```
BuddhistTemple: URIRef = rdflib.term.URIRef('https://schema.org/BuddhistTemple')
BusOrCoach: URIRef = rdflib.term.URIRef('https://schema.org/BusOrCoach')
BusReservation: URIRef = rdflib.term.URIRef('https://schema.org/BusReservation')
BusStation: URIRef = rdflib.term.URIRef('https://schema.org/BusStation')
BusStop: URIRef = rdflib.term.URIRef('https://schema.org/BusStop')
BusTrip: URIRef = rdflib.term.URIRef('https://schema.org/BusTrip')
BusinessAudience: URIRef =
rdflib.term.URIRef('https://schema.org/BusinessAudience')
BusinessEntityType: URIRef =
rdflib.term.URIRef('https://schema.org/BusinessEntityType')
BusinessEvent: URIRef = rdflib.term.URIRef('https://schema.org/BusinessEvent')
BusinessFunction: URIRef =
rdflib.term.URIRef('https://schema.org/BusinessFunction')
BusinessSupport: URIRef = rdflib.term.URIRef('https://schema.org/BusinessSupport')
BuyAction: URIRef = rdflib.term.URIRef('https://schema.org/BuyAction')
CDCPMDRecord: URIRef = rdflib.term.URIRef('https://schema.org/CDCPMDRecord')
CDFormat: URIRef = rdflib.term.URIRef('https://schema.org/CDFormat')
CT: URIRef = rdflib.term.URIRef('https://schema.org/CT')
CableOrSatelliteService: URIRef =
rdflib.term.URIRef('https://schema.org/CableOrSatelliteService')
CafeOrCoffeeShop: URIRef =
rdflib.term.URIRef('https://schema.org/CafeOrCoffeeShop')
Campground: URIRef = rdflib.term.URIRef('https://schema.org/Campground')
CampingPitch: URIRef = rdflib.term.URIRef('https://schema.org/CampingPitch')
Canal: URIRef = rdflib.term.URIRef('https://schema.org/Canal')
CancelAction: URIRef = rdflib.term.URIRef('https://schema.org/CancelAction')
Car: URIRef = rdflib.term.URIRef('https://schema.org/Car')
CarUsageType: URIRef = rdflib.term.URIRef('https://schema.org/CarUsageType')
Cardiovascular: URIRef = rdflib.term.URIRef('https://schema.org/Cardiovascular')
CardiovascularExam: URIRef =
rdflib.term.URIRef('https://schema.org/CardiovascularExam')
CaseSeries: URIRef = rdflib.term.URIRef('https://schema.org/CaseSeries')
Casino: URIRef = rdflib.term.URIRef('https://schema.org/Casino')
```

```
CassetteFormat: URIRef = rdflib.term.URIRef('https://schema.org/CassetteFormat')
CategoryCode: URIRef = rdflib.term.URIRef('https://schema.org/CategoryCode')
CategoryCodeSet: URIRef = rdflib.term.URIRef('https://schema.org/CategoryCodeSet')
CatholicChurch: URIRef = rdflib.term.URIRef('https://schema.org/CatholicChurch')
CausesHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/CausesHealthAspect')
Cemetery: URIRef = rdflib.term.URIRef('https://schema.org/Cemetery')
Chapter: URIRef = rdflib.term.URIRef('https://schema.org/Chapter')
CharitableIncorporatedOrganization: URIRef =
rdflib.term.URIRef('https://schema.org/CharitableIncorporatedOrganization')
CheckAction: URIRef = rdflib.term.URIRef('https://schema.org/CheckAction')
CheckInAction: URIRef = rdflib.term.URIRef('https://schema.org/CheckInAction')
CheckOutAction: URIRef = rdflib.term.URIRef('https://schema.org/CheckOutAction')
CheckoutPage: URIRef = rdflib.term.URIRef('https://schema.org/CheckoutPage')
ChemicalSubstance: URIRef =
rdflib.term.URIRef('https://schema.org/ChemicalSubstance')
ChildCare: URIRef = rdflib.term.URIRef('https://schema.org/ChildCare')
ChildrensEvent: URIRef = rdflib.term.URIRef('https://schema.org/ChildrensEvent')
Chiropractic: URIRef = rdflib.term.URIRef('https://schema.org/Chiropractic')
ChooseAction: URIRef = rdflib.term.URIRef('https://schema.org/ChooseAction')
Church: URIRef = rdflib.term.URIRef('https://schema.org/Church')
City: URIRef = rdflib.term.URIRef('https://schema.org/City')
CityHall: URIRef = rdflib.term.URIRef('https://schema.org/CityHall')
CivicStructure: URIRef = rdflib.term.URIRef('https://schema.org/CivicStructure')
Claim: URIRef = rdflib.term.URIRef('https://schema.org/Claim')
ClaimReview: URIRef = rdflib.term.URIRef('https://schema.org/ClaimReview')
Class: URIRef = rdflib.term.URIRef('https://schema.org/Class')
CleaningFee: URIRef = rdflib.term.URIRef('https://schema.org/CleaningFee')
Clinician: URIRef = rdflib.term.URIRef('https://schema.org/Clinician')
Clip: URIRef = rdflib.term.URIRef('https://schema.org/Clip')
ClothingStore: URIRef = rdflib.term.URIRef('https://schema.org/ClothingStore')
CoOp: URIRef = rdflib.term.URIRef('https://schema.org/CoOp')
```

```
Code: URIRef = rdflib.term.URIRef('https://schema.org/Code')
CohortStudy: URIRef = rdflib.term.URIRef('https://schema.org/CohortStudy')
Collection: URIRef = rdflib.term.URIRef('https://schema.org/Collection')
CollectionPage: URIRef = rdflib.term.URIRef('https://schema.org/CollectionPage')
CollegeOrUniversity: URIRef =
rdflib.term.URIRef('https://schema.org/CollegeOrUniversity')
ComedyClub: URIRef = rdflib.term.URIRef('https://schema.org/ComedyClub')
ComedyEvent: URIRef = rdflib.term.URIRef('https://schema.org/ComedyEvent')
ComicCoverArt: URIRef = rdflib.term.URIRef('https://schema.org/ComicCoverArt')
ComicIssue: URIRef = rdflib.term.URIRef('https://schema.org/ComicIssue')
ComicSeries: URIRef = rdflib.term.URIRef('https://schema.org/ComicSeries')
ComicStory: URIRef = rdflib.term.URIRef('https://schema.org/ComicStory')
Comment: URIRef = rdflib.term.URIRef('https://schema.org/Comment')
CommentAction: URIRef = rdflib.term.URIRef('https://schema.org/CommentAction')
CommentPermission: URIRef =
rdflib.term.URIRef('https://schema.org/CommentPermission')
CommunicateAction: URIRef =
rdflib.term.URIRef('https://schema.org/CommunicateAction')
CommunityHealth: URIRef = rdflib.term.URIRef('https://schema.org/CommunityHealth')
CompilationAlbum: URIRef =
rdflib.term.URIRef('https://schema.org/CompilationAlbum')
CompleteDataFeed: URIRef =
rdflib.term.URIRef('https://schema.org/CompleteDataFeed')
Completed: URIRef = rdflib.term.URIRef('https://schema.org/Completed')
CompletedActionStatus: URIRef =
rdflib.term.URIRef('https://schema.org/CompletedActionStatus')
CompoundPriceSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/CompoundPriceSpecification')
ComputerLanguage: URIRef =
rdflib.term.URIRef('https://schema.org/ComputerLanguage')
ComputerStore: URIRef = rdflib.term.URIRef('https://schema.org/ComputerStore')
ConfirmAction: URIRef = rdflib.term.URIRef('https://schema.org/ConfirmAction')
Consortium: URIRef = rdflib.term.URIRef('https://schema.org/Consortium')
ConsumeAction: URIRef = rdflib.term.URIRef('https://schema.org/ConsumeAction')
```

```
ContactPage: URIRef = rdflib.term.URIRef('https://schema.org/ContactPage')
ContactPoint: URIRef = rdflib.term.URIRef('https://schema.org/ContactPoint')
ContactPointOption: URIRef =
rdflib.term.URIRef('https://schema.org/ContactPointOption')
ContagiousnessHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/ContagiousnessHealthAspect')
Continent: URIRef = rdflib.term.URIRef('https://schema.org/Continent')
ControlAction: URIRef = rdflib.term.URIRef('https://schema.org/ControlAction')
ConvenienceStore: URIRef =
rdflib.term.URIRef('https://schema.org/ConvenienceStore')
Conversation: URIRef = rdflib.term.URIRef('https://schema.org/Conversation')
CookAction: URIRef = rdflib.term.URIRef('https://schema.org/CookAction')
Corporation: URIRef = rdflib.term.URIRef('https://schema.org/Corporation')
CorrectionComment: URIRef =
rdflib.term.URIRef('https://schema.org/CorrectionComment')
Country: URIRef = rdflib.term.URIRef('https://schema.org/Country')
Course: URIRef = rdflib.term.URIRef('https://schema.org/Course')
CourseInstance: URIRef = rdflib.term.URIRef('https://schema.org/CourseInstance')
Courthouse: URIRef = rdflib.term.URIRef('https://schema.org/Courthouse')
CoverArt: URIRef = rdflib.term.URIRef('https://schema.org/CoverArt')
CovidTestingFacility: URIRef =
rdflib.term.URIRef('https://schema.org/CovidTestingFacility')
CreateAction: URIRef = rdflib.term.URIRef('https://schema.org/CreateAction')
CreativeWork: URIRef = rdflib.term.URIRef('https://schema.org/CreativeWork')
CreativeWorkSeason: URIRef =
rdflib.term.URIRef('https://schema.org/CreativeWorkSeason')
CreativeWorkSeries: URIRef =
rdflib.term.URIRef('https://schema.org/CreativeWorkSeries')
CreditCard: URIRef = rdflib.term.URIRef('https://schema.org/CreditCard')
Crematorium: URIRef = rdflib.term.URIRef('https://schema.org/Crematorium')
CriticReview: URIRef = rdflib.term.URIRef('https://schema.org/CriticReview')
CrossSectional: URIRef = rdflib.term.URIRef('https://schema.org/CrossSectional')
CssSelectorType: URIRef = rdflib.term.URIRef('https://schema.org/CssSelectorType')
```

```
CurrencyConversionService: URIRef =  
rdflib.term.URIRef('https://schema.org/CurrencyConversionService')  
  
DDxElement: URIRef = rdflib.term.URIRef('https://schema.org/DDxElement')  
  
DJMixAlbum: URIRef = rdflib.term.URIRef('https://schema.org/DJMixaAlbum')  
  
DVDFormat: URIRef = rdflib.term.URIRef('https://schema.org/DVDFormat')  
  
DamagedCondition: URIRef =  
rdflib.term.URIRef('https://schema.org/DamagedCondition')  
  
DanceEvent: URIRef = rdflib.term.URIRef('https://schema.org/DanceEvent')  
  
DanceGroup: URIRef = rdflib.term.URIRef('https://schema.org/DanceGroup')  
  
DataCatalog: URIRef = rdflib.term.URIRef('https://schema.org/DataCatalog')  
  
DataDownload: URIRef = rdflib.term.URIRef('https://schema.org/DataDownload')  
  
DataFeed: URIRef = rdflib.term.URIRef('https://schema.org/DataFeed')  
  
DataFeedItem: URIRef = rdflib.term.URIRef('https://schema.org/DataFeedItem')  
  
DataType: URIRef = rdflib.term.URIRef('https://schema.org/DataType')  
  
Dataset: URIRef = rdflib.term.URIRef('https://schema.org/Dataset')  
  
Date: URIRef = rdflib.term.URIRef('https://schema.org/Date')  
  
DateTime: URIRef = rdflib.term.URIRef('https://schema.org/DateTime')  
  
DatedMoneySpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/DatedMoneySpecification')  
  
DayOfWeek: URIRef = rdflib.term.URIRef('https://schema.org/DayOfWeek')  
  
DaySpa: URIRef = rdflib.term.URIRef('https://schema.org/DaySpa')  
  
DeactivateAction: URIRef =  
rdflib.term.URIRef('https://schema.org/DeactivateAction')  
  
DecontextualizedContent: URIRef =  
rdflib.term.URIRef('https://schema.org/DecontextualizedContent')  
  
DefenceEstablishment: URIRef =  
rdflib.term.URIRef('https://schema.org/DefenceEstablishment')  
  
DefinedRegion: URIRef = rdflib.term.URIRef('https://schema.org/DefinedRegion')  
  
DefinedTerm: URIRef = rdflib.term.URIRef('https://schema.org/DefinedTerm')  
  
DefinedTermSet: URIRef = rdflib.term.URIRef('https://schema.org/DefinedTermSet')  
  
DefinitiveLegalValue: URIRef =  
rdflib.term.URIRef('https://schema.org/DefinitiveLegalValue')  
  
DeleteAction: URIRef = rdflib.term.URIRef('https://schema.org/DeleteAction')
```

```
DeliveryChargeSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/DeliveryChargeSpecification')  
  
DeliveryEvent: URIRef = rdflib.term.URIRef('https://schema.org/DeliveryEvent')  
  
DeliveryMethod: URIRef = rdflib.term.URIRef('https://schema.org/DeliveryMethod')  
  
DeliveryTimeSettings: URIRef =  
rdflib.term.URIRef('https://schema.org/DeliveryTimeSettings')  
  
Demand: URIRef = rdflib.term.URIRef('https://schema.org/Demand')  
  
DemoAlbum: URIRef = rdflib.term.URIRef('https://schema.org/DemoAlbum')  
  
Dentist: URIRef = rdflib.term.URIRef('https://schema.org/Dentist')  
  
Dentistry: URIRef = rdflib.term.URIRef('https://schema.org/Dentistry')  
  
DepartAction: URIRef = rdflib.term.URIRef('https://schema.org/DepartAction')  
  
DepartmentStore: URIRef = rdflib.term.URIRef('https://schema.org/DepartmentStore')  
  
DepositAccount: URIRef = rdflib.term.URIRef('https://schema.org/DepositAccount')  
  
Dermatologic: URIRef = rdflib.term.URIRef('https://schema.org/Dermatologic')  
  
Dermatology: URIRef = rdflib.term.URIRef('https://schema.org/Dermatology')  
  
DiabeticDiet: URIRef = rdflib.term.URIRef('https://schema.org/DiabeticDiet')  
  
Diagnostic: URIRef = rdflib.term.URIRef('https://schema.org/Diagnostic')  
  
DiagnosticLab: URIRef = rdflib.term.URIRef('https://schema.org/DiagnosticLab')  
  
DiagnosticProcedure: URIRef =  
rdflib.term.URIRef('https://schema.org/DiagnosticProcedure')  
  
Diet: URIRef = rdflib.term.URIRef('https://schema.org/Diet')  
  
DietNutrition: URIRef = rdflib.term.URIRef('https://schema.org/DietNutrition')  
  
DietarySupplement: URIRef =  
rdflib.term.URIRef('https://schema.org/DietarySupplement')  
  
DigitalAudioTapeFormat: URIRef =  
rdflib.term.URIRef('https://schema.org/DigitalAudioTapeFormat')  
  
DigitalDocument: URIRef = rdflib.term.URIRef('https://schema.org/DigitalDocument')  
  
DigitalDocumentPermission: URIRef =  
rdflib.term.URIRef('https://schema.org/DigitalDocumentPermission')  
  
DigitalDocumentPermissionType: URIRef =  
rdflib.term.URIRef('https://schema.org/DigitalDocumentPermissionType')  
  
DigitalFormat: URIRef = rdflib.term.URIRef('https://schema.org/DigitalFormat')  
  
DisabilitySupport: URIRef =  
rdflib.term.URIRef('https://schema.org/DisabilitySupport')
```

```
DisagreeAction: URIRef = rdflib.term.URIRef('https://schema.org/DisagreeAction')
Discontinued: URIRef = rdflib.term.URIRef('https://schema.org/Discontinued')
DiscoverAction: URIRef = rdflib.term.URIRef('https://schema.org/DiscoverAction')
DiscussionForumPosting: URIRef =
rdflib.term.URIRef('https://schema.org/DiscussionForumPosting')
DislikeAction: URIRef = rdflib.term.URIRef('https://schema.org/DislikeAction')
Distance: URIRef = rdflib.term.URIRef('https://schema.org/Distance')
DistanceFee: URIRef = rdflib.term.URIRef('https://schema.org/DistanceFee')
Distillery: URIRef = rdflib.term.URIRef('https://schema.org/Distillery')
DonateAction: URIRef = rdflib.term.URIRef('https://schema.org/DonateAction')
DoseSchedule: URIRef = rdflib.term.URIRef('https://schema.org/DoseSchedule')
DoubleBlindedTrial: URIRef =
rdflib.term.URIRef('https://schema.org/DoubleBlindedTrial')
DownloadAction: URIRef = rdflib.term.URIRef('https://schema.org/DownloadAction')
Downpayment: URIRef = rdflib.term.URIRef('https://schema.org/Downpayment')
DrawAction: URIRef = rdflib.term.URIRef('https://schema.org/DrawAction')
Drawing: URIRef = rdflib.term.URIRef('https://schema.org/Drawing')
DrinkAction: URIRef = rdflib.term.URIRef('https://schema.org/DrinkAction')
DriveWheelConfigurationValue: URIRef =
rdflib.term.URIRef('https://schema.org/DriveWheelConfigurationValue')
DrivingSchoolVehicleUsage: URIRef =
rdflib.term.URIRef('https://schema.org/DrivingSchoolVehicleUsage')
Drug: URIRef = rdflib.term.URIRef('https://schema.org/Drug')
DrugClass: URIRef = rdflib.term.URIRef('https://schema.org/DrugClass')
DrugCost: URIRef = rdflib.term.URIRef('https://schema.org/DrugCost')
DrugCostCategory: URIRef =
rdflib.term.URIRef('https://schema.org/DrugCostCategory')
DrugLegalStatus: URIRef = rdflib.term.URIRef('https://schema.org/DrugLegalStatus')
DrugPregnancyCategory: URIRef =
rdflib.term.URIRef('https://schema.org/DrugPregnancyCategory')
DrugPrescriptionStatus: URIRef =
rdflib.term.URIRef('https://schema.org/DrugPrescriptionStatus')
DrugStrength: URIRef = rdflib.term.URIRef('https://schema.org/DrugStrength')
```



```
DryCleaningOrLaundry: URIRef =  
rdflib.term.URIRef('https://schema.org/DryCleaningOrLaundry')  
  
Duration: URIRef = rdflib.term.URIRef('https://schema.org/Duration')  
  
EBook: URIRef = rdflib.term.URIRef('https://schema.org/EBook')  
  
EPRelease: URIRef = rdflib.term.URIRef('https://schema.org/EPRelease')  
  
EUEnergyEfficiencyCategoryA: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryA')  
  
EUEnergyEfficiencyCategoryA1Plus: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryA1Plus')  
  
EUEnergyEfficiencyCategoryA2Plus: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryA2Plus')  
  
EUEnergyEfficiencyCategoryA3Plus: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryA3Plus')  
  
EUEnergyEfficiencyCategoryB: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryB')  
  
EUEnergyEfficiencyCategoryC: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryC')  
  
EUEnergyEfficiencyCategoryD: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryD')  
  
EUEnergyEfficiencyCategoryE: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryE')  
  
EUEnergyEfficiencyCategoryF: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryF')  
  
EUEnergyEfficiencyCategoryG: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyCategoryG')  
  
EUEnergyEfficiencyEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/EUEnergyEfficiencyEnumeration')  
  
Ear: URIRef = rdflib.term.URIRef('https://schema.org/Ear')  
  
EatAction: URIRef = rdflib.term.URIRef('https://schema.org/EatAction')  
  
EditedOrCroppedContent: URIRef =  
rdflib.term.URIRef('https://schema.org/EditedOrCroppedContent')  
  
EducationEvent: URIRef = rdflib.term.URIRef('https://schema.org/EducationEvent')  
  
EducationalAudience: URIRef =  
rdflib.term.URIRef('https://schema.org/EducationalAudience')  
  
EducationalOccupationalCredential: URIRef =  
rdflib.term.URIRef('https://schema.org/EducationalOccupationalCredential')
```



```
EducationalOccupationalProgram: URIRef =  
rdflib.term.URIRef('https://schema.org/EducationalOccupationalProgram')  
  
EducationalOrganization: URIRef =  
rdflib.term.URIRef('https://schema.org/EducationalOrganization')  
  
EffectivenessHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/EffectivenessHealthAspect')  
  
Electrician: URIRef = rdflib.term.URIRef('https://schema.org/Electrician')  
  
ElectronicsStore: URIRef =  
rdflib.term.URIRef('https://schema.org/ElectronicsStore')  
  
ElementarySchool: URIRef =  
rdflib.term.URIRef('https://schema.org/ElementarySchool')  
  
EmailMessage: URIRef = rdflib.term.URIRef('https://schema.org/EmailMessage')  
  
Embassy: URIRef = rdflib.term.URIRef('https://schema.org/Embassy')  
  
Emergency: URIRef = rdflib.term.URIRef('https://schema.org/Emergency')  
  
EmergencyService: URIRef =  
rdflib.term.URIRef('https://schema.org/EmergencyService')  
  
EmployeeRole: URIRef = rdflib.term.URIRef('https://schema.org/EmployeeRole')  
  
EmployerAggregateRating: URIRef =  
rdflib.term.URIRef('https://schema.org/EmployerAggregateRating')  
  
EmployerReview: URIRef = rdflib.term.URIRef('https://schema.org/EmployerReview')  
  
EmploymentAgency: URIRef =  
rdflib.term.URIRef('https://schema.org/EmploymentAgency')  
  
Endocrine: URIRef = rdflib.term.URIRef('https://schema.org/Endocrine')  
  
EndorseAction: URIRef = rdflib.term.URIRef('https://schema.org/EndorseAction')  
  
EndorsementRating: URIRef =  
rdflib.term.URIRef('https://schema.org/EndorsementRating')  
  
Energy: URIRef = rdflib.term.URIRef('https://schema.org/Energy')  
  
EnergyConsumptionDetails: URIRef =  
rdflib.term.URIRef('https://schema.org/EnergyConsumptionDetails')  
  
EnergyEfficiencyEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/EnergyEfficiencyEnumeration')  
  
EnergyStarCertified: URIRef =  
rdflib.term.URIRef('https://schema.org/EnergyStarCertified')  
  
EnergyStarEnergyEfficiencyEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/EnergyStarEnergyEfficiencyEnumeration')
```

```
EngineSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/EngineSpecification')  
  
EnrollingByInvitation: URIRef =  
rdflib.term.URIRef('https://schema.org/EnrollingByInvitation')  
  
EntertainmentBusiness: URIRef =  
rdflib.term.URIRef('https://schema.org/EntertainmentBusiness')  
  
EntryPoint: URIRef = rdflib.term.URIRef('https://schema.org/EntryPoint')  
  
Enumeration: URIRef = rdflib.term.URIRef('https://schema.org/Enumeration')  
  
Episode: URIRef = rdflib.term.URIRef('https://schema.org/Episode')  
  
Event: URIRef = rdflib.term.URIRef('https://schema.org/Event')  
  
EventAttendanceModeEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/EventAttendanceModeEnumeration')  
  
EventCancelled: URIRef = rdflib.term.URIRef('https://schema.org/EventCancelled')  
  
EventMovedOnline: URIRef =  
rdflib.term.URIRef('https://schema.org/EventMovedOnline')  
  
EventPostponed: URIRef = rdflib.term.URIRef('https://schema.org/EventPostponed')  
  
EventRescheduled: URIRef =  
rdflib.term.URIRef('https://schema.org/EventRescheduled')  
  
EventReservation: URIRef =  
rdflib.term.URIRef('https://schema.org/EventReservation')  
  
EventScheduled: URIRef = rdflib.term.URIRef('https://schema.org/EventScheduled')  
  
EventSeries: URIRef = rdflib.term.URIRef('https://schema.org/EventSeries')  
  
EventStatusType: URIRef = rdflib.term.URIRef('https://schema.org/EventStatusType')  
  
EventVenue: URIRef = rdflib.term.URIRef('https://schema.org/EventVenue')  
  
EvidenceLevelA: URIRef = rdflib.term.URIRef('https://schema.org/EvidenceLevelA')  
  
EvidenceLevelB: URIRef = rdflib.term.URIRef('https://schema.org/EvidenceLevelB')  
  
EvidenceLevelC: URIRef = rdflib.term.URIRef('https://schema.org/EvidenceLevelC')  
  
ExchangeRateSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/ExchangeRateSpecification')  
  
ExchangeRefund: URIRef = rdflib.term.URIRef('https://schema.org/ExchangeRefund')  
  
ExerciseAction: URIRef = rdflib.term.URIRef('https://schema.org/ExerciseAction')  
  
ExerciseGym: URIRef = rdflib.term.URIRef('https://schema.org/ExerciseGym')  
  
ExercisePlan: URIRef = rdflib.term.URIRef('https://schema.org/ExercisePlan')  
  
ExhibitionEvent: URIRef = rdflib.term.URIRef('https://schema.org/ExhibitionEvent')
```

```
Eye: URIRef = rdflib.term.URIRef('https://schema.org/Eye')
FAQPage: URIRef = rdflib.term.URIRef('https://schema.org/FAQPage')
FDACategoryA: URIRef = rdflib.term.URIRef('https://schema.org/FDACategoryA')
FDACategoryB: URIRef = rdflib.term.URIRef('https://schema.org/FDACategoryB')
FDACategoryC: URIRef = rdflib.term.URIRef('https://schema.org/FDACategoryC')
FDACategoryD: URIRef = rdflib.term.URIRef('https://schema.org/FDACategoryD')
FDACategoryX: URIRef = rdflib.term.URIRef('https://schema.org/FDACategoryX')
FDAnotEvaluated: URIRef = rdflib.term.URIRef('https://schema.org/FDAnotEvaluated')
FMRadioChannel: URIRef = rdflib.term.URIRef('https://schema.org/FMRadioChannel')
FailedActionStatus: URIRef =
rdflib.term.URIRef('https://schema.org/FailedActionStatus')
FastFoodRestaurant: URIRef =
rdflib.term.URIRef('https://schema.org/FastFoodRestaurant')
Female: URIRef = rdflib.term.URIRef('https://schema.org/Female')
Festival: URIRef = rdflib.term.URIRef('https://schema.org/Festival')
FilmAction: URIRef = rdflib.term.URIRef('https://schema.org/FilmAction')
FinancialProduct: URIRef =
rdflib.term.URIRef('https://schema.org/FinancialProduct')
FinancialService: URIRef =
rdflib.term.URIRef('https://schema.org/FinancialService')
FindAction: URIRef = rdflib.term.URIRef('https://schema.org/FindAction')
FireStation: URIRef = rdflib.term.URIRef('https://schema.org/FireStation')
Flexibility: URIRef = rdflib.term.URIRef('https://schema.org/Flexibility')
Flight: URIRef = rdflib.term.URIRef('https://schema.org/Flight')
FlightReservation: URIRef =
rdflib.term.URIRef('https://schema.org/FlightReservation')
Float: URIRef = rdflib.term.URIRef('https://schema.org/Float')
FloorPlan: URIRef = rdflib.term.URIRef('https://schema.org/FloorPlan')
Florist: URIRef = rdflib.term.URIRef('https://schema.org/Florist')
FollowAction: URIRef = rdflib.term.URIRef('https://schema.org/FollowAction')
FoodEstablishment: URIRef =
rdflib.term.URIRef('https://schema.org/FoodEstablishment')
```

```

FoodEstablishmentReservation: URIRef =
rdflib.term.URIRef('https://schema.org/FoodEstablishmentReservation')

FoodEvent: URIRef = rdflib.term.URIRef('https://schema.org/FoodEvent')

FoodService: URIRef = rdflib.term.URIRef('https://schema.org/FoodService')

FourWheelDriveConfiguration: URIRef =
rdflib.term.URIRef('https://schema.org/FourWheelDriveConfiguration')

FreeReturn: URIRef = rdflib.term.URIRef('https://schema.org/FreeReturn')

Friday: URIRef = rdflib.term.URIRef('https://schema.org/Friday')

FrontWheelDriveConfiguration: URIRef =
rdflib.term.URIRef('https://schema.org/FrontWheelDriveConfiguration')

FullRefund: URIRef = rdflib.term.URIRef('https://schema.org/FullRefund')

FundingAgency: URIRef = rdflib.term.URIRef('https://schema.org/FundingAgency')

FundingScheme: URIRef = rdflib.term.URIRef('https://schema.org/FundingScheme')

Fungus: URIRef = rdflib.term.URIRef('https://schema.org/Fungus')

FurnitureStore: URIRef = rdflib.term.URIRef('https://schema.org/FurnitureStore')

Game: URIRef = rdflib.term.URIRef('https://schema.org/Game')

GamePlayMode: URIRef = rdflib.term.URIRef('https://schema.org/GamePlayMode')

GameServer: URIRef = rdflib.term.URIRef('https://schema.org/GameServer')

GameServerStatus: URIRef =
rdflib.term.URIRef('https://schema.org/GameServerStatus')

GardenStore: URIRef = rdflib.term.URIRef('https://schema.org/GardenStore')

GasStation: URIRef = rdflib.term.URIRef('https://schema.org/GasStation')

Gastroenterologic: URIRef =
rdflib.term.URIRef('https://schema.org/Gastroenterologic')

GatedResidenceCommunity: URIRef =
rdflib.term.URIRef('https://schema.org/GatedResidenceCommunity')

GenderType: URIRef = rdflib.term.URIRef('https://schema.org/GenderType')

Gene: URIRef = rdflib.term.URIRef('https://schema.org/Gene')

GeneralContractor: URIRef =
rdflib.term.URIRef('https://schema.org/GeneralContractor')

Genetic: URIRef = rdflib.term.URIRef('https://schema.org/Genetic')

Genitourinary: URIRef = rdflib.term.URIRef('https://schema.org/Genitourinary')

GeoCircle: URIRef = rdflib.term.URIRef('https://schema.org/GeoCircle')

```

```
GeoCoordinates: URIRef = rdflib.term.URIRef('https://schema.org/GeoCoordinates')

GeoShape: URIRef = rdflib.term.URIRef('https://schema.org/GeoShape')

GeospatialGeometry: URIRef =
rdflib.term.URIRef('https://schema.org/GeospatialGeometry')

Geriatric: URIRef = rdflib.term.URIRef('https://schema.org/Geriatric')

GettingAccessHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/GettingAccessHealthAspect')

GiveAction: URIRef = rdflib.term.URIRef('https://schema.org/GiveAction')

GlutenFreeDiet: URIRef = rdflib.term.URIRef('https://schema.org/GlutenFreeDiet')

GolfCourse: URIRef = rdflib.term.URIRef('https://schema.org/GolfCourse')

GovernmentBenefitsType: URIRef =
rdflib.term.URIRef('https://schema.org/GovernmentBenefitsType')

GovernmentBuilding: URIRef =
rdflib.term.URIRef('https://schema.org/GovernmentBuilding')

GovernmentOffice: URIRef =
rdflib.term.URIRef('https://schema.org/GovernmentOffice')

GovernmentOrganization: URIRef =
rdflib.term.URIRef('https://schema.org/GovernmentOrganization')

GovernmentPermit: URIRef =
rdflib.term.URIRef('https://schema.org/GovernmentPermit')

GovernmentService: URIRef =
rdflib.term.URIRef('https://schema.org/GovernmentService')

Grant: URIRef = rdflib.term.URIRef('https://schema.org/Grant')

GraphicNovel: URIRef = rdflib.term.URIRef('https://schema.org/GraphicNovel')

GroceryStore: URIRef = rdflib.term.URIRef('https://schema.org/GroceryStore')

GroupBoardingPolicy: URIRef =
rdflib.term.URIRef('https://schema.org/GroupBoardingPolicy')

Guide: URIRef = rdflib.term.URIRef('https://schema.org/Guide')

Gynecologic: URIRef = rdflib.term.URIRef('https://schema.org/Gynecologic')

HVACBusiness: URIRef = rdflib.term.URIRef('https://schema.org/HVACBusiness')

Hackathon: URIRef = rdflib.term.URIRef('https://schema.org/Hackathon')

HairSalon: URIRef = rdflib.term.URIRef('https://schema.org/HairSalon')

HalalDiet: URIRef = rdflib.term.URIRef('https://schema.org/HalalDiet')

Hardcover: URIRef = rdflib.term.URIRef('https://schema.org/Hardcover')
```

```
HardwareStore: URIRef = rdflib.term.URIRef('https://schema.org/HardwareStore')

Head: URIRef = rdflib.term.URIRef('https://schema.org/Head')

HealthAndBeautyBusiness: URIRef =
rdflib.term.URIRef('https://schema.org/HealthAndBeautyBusiness')

HealthAspectEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/HealthAspectEnumeration')

HealthCare: URIRef = rdflib.term.URIRef('https://schema.org/HealthCare')

HealthClub: URIRef = rdflib.term.URIRef('https://schema.org/HealthClub')

HealthInsurancePlan: URIRef =
rdflib.term.URIRef('https://schema.org/HealthInsurancePlan')

HealthPlanCostSharingSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/HealthPlanCostSharingSpecification')

HealthPlanFormulary: URIRef =
rdflib.term.URIRef('https://schema.org/HealthPlanFormulary')

HealthPlanNetwork: URIRef =
rdflib.term.URIRef('https://schema.org/HealthPlanNetwork')

HealthTopicContent: URIRef =
rdflib.term.URIRef('https://schema.org/HealthTopicContent')

HearingImpairedSupported: URIRef =
rdflib.term.URIRef('https://schema.org/HearingImpairedSupported')

Hematologic: URIRef = rdflib.term.URIRef('https://schema.org/Hematologic')

HighSchool: URIRef = rdflib.term.URIRef('https://schema.org/HighSchool')

HinduDiet: URIRef = rdflib.term.URIRef('https://schema.org/HinduDiet')

HinduTemple: URIRef = rdflib.term.URIRef('https://schema.org/HinduTemple')

HobbyShop: URIRef = rdflib.term.URIRef('https://schema.org/HobbyShop')

HomeAndConstructionBusiness: URIRef =
rdflib.term.URIRef('https://schema.org/HomeAndConstructionBusiness')

HomeGoodsStore: URIRef = rdflib.term.URIRef('https://schema.org/HomeGoodsStore')

Homeopathic: URIRef = rdflib.term.URIRef('https://schema.org/Homeopathic')

Hospital: URIRef = rdflib.term.URIRef('https://schema.org/Hospital')

Hostel: URIRef = rdflib.term.URIRef('https://schema.org/Hostel')

Hotel: URIRef = rdflib.term.URIRef('https://schema.org/Hotel')

HotelRoom: URIRef = rdflib.term.URIRef('https://schema.org/HotelRoom')

House: URIRef = rdflib.term.URIRef('https://schema.org/House')
```

```
HousePainter: URIRef = rdflib.term.URIRef('https://schema.org/HousePainter')

HowItWorksHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/HowItWorksHealthAspect')

HowOrWhereHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/HowOrWhereHealthAspect')

HowTo: URIRef = rdflib.term.URIRef('https://schema.org/HowTo')

HowToDirection: URIRef = rdflib.term.URIRef('https://schema.org/HowToDirection')

HowToItem: URIRef = rdflib.term.URIRef('https://schema.org/HowToItem')

HowToSection: URIRef = rdflib.term.URIRef('https://schema.org/HowToSection')

HowToStep: URIRef = rdflib.term.URIRef('https://schema.org/HowToStep')

HowToSupply: URIRef = rdflib.term.URIRef('https://schema.org/HowToSupply')

HowToTip: URIRef = rdflib.term.URIRef('https://schema.org/HowToTip')

HowToTool: URIRef = rdflib.term.URIRef('https://schema.org/HowToTool')

HyperToc: URIRef = rdflib.term.URIRef('https://schema.org/HyperToc')

HyperTocEntry: URIRef = rdflib.term.URIRef('https://schema.org/HyperTocEntry')

IceCreamShop: URIRef = rdflib.term.URIRef('https://schema.org/IceCreamShop')

IgnoreAction: URIRef = rdflib.term.URIRef('https://schema.org/IgnoreAction')

ImageGallery: URIRef = rdflib.term.URIRef('https://schema.org/ImageGallery')

ImageObject: URIRef = rdflib.term.URIRef('https://schema.org/ImageObject')

ImageObjectSnapshot: URIRef =
rdflib.term.URIRef('https://schema.org/ImageObjectSnapshot')

ImagingTest: URIRef = rdflib.term.URIRef('https://schema.org/ImagingTest')

InForce: URIRef = rdflib.term.URIRef('https://schema.org/InForce')

InStock: URIRef = rdflib.term.URIRef('https://schema.org/InStock')

InStoreOnly: URIRef = rdflib.term.URIRef('https://schema.org/InStoreOnly')

IndividualProduct: URIRef =
rdflib.term.URIRef('https://schema.org/IndividualProduct')

Infectious: URIRef = rdflib.term.URIRef('https://schema.org/Infectious')

InfectiousAgentClass: URIRef =
rdflib.term.URIRef('https://schema.org/InfectiousAgentClass')

InfectiousDisease: URIRef =
rdflib.term.URIRef('https://schema.org/InfectiousDisease')

InformAction: URIRef = rdflib.term.URIRef('https://schema.org/InformAction')
```



```
IngredientsHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/IngredientsHealthAspect')  
  
InsertAction: URIRef = rdflib.term.URIRef('https://schema.org/InsertAction')  
  
InstallAction: URIRef = rdflib.term.URIRef('https://schema.org/InstallAction')  
  
Installment: URIRef = rdflib.term.URIRef('https://schema.org/Installment')  
  
InsuranceAgency: URIRef = rdflib.term.URIRef('https://schema.org/InsuranceAgency')  
  
Intangible: URIRef = rdflib.term.URIRef('https://schema.org/Intangible')  
  
Integer: URIRef = rdflib.term.URIRef('https://schema.org/Integer')  
  
InteractAction: URIRef = rdflib.term.URIRef('https://schema.org/InteractAction')  
  
InteractionCounter: URIRef =  
rdflib.term.URIRef('https://schema.org/InteractionCounter')  
  
InternationalTrial: URIRef =  
rdflib.term.URIRef('https://schema.org/InternationalTrial')  
  
InternetCafe: URIRef = rdflib.term.URIRef('https://schema.org/InternetCafe')  
  
InvestmentFund: URIRef = rdflib.term.URIRef('https://schema.org/InvestmentFund')  
  
InvestmentOrDeposit: URIRef =  
rdflib.term.URIRef('https://schema.org/InvestmentOrDeposit')  
  
InviteAction: URIRef = rdflib.term.URIRef('https://schema.org/InviteAction')  
  
Invoice: URIRef = rdflib.term.URIRef('https://schema.org/Invoice')  
  
InvoicePrice: URIRef = rdflib.term.URIRef('https://schema.org/InvoicePrice')  
  
ItemAvailability: URIRef =  
rdflib.term.URIRef('https://schema.org/ItemAvailability')  
  
ItemList: URIRef = rdflib.term.URIRef('https://schema.org/ItemList')  
  
ItemListOrderAscending: URIRef =  
rdflib.term.URIRef('https://schema.org/ItemListOrderAscending')  
  
ItemListOrderDescending: URIRef =  
rdflib.term.URIRef('https://schema.org/ItemListOrderDescending')  
  
ItemListOrderType: URIRef =  
rdflib.term.URIRef('https://schema.org/ItemListOrderType')  
  
ItemListUnordered: URIRef =  
rdflib.term.URIRef('https://schema.org/ItemListUnordered')  
  
ItemPage: URIRef = rdflib.term.URIRef('https://schema.org/ItemPage')  
  
JewelryStore: URIRef = rdflib.term.URIRef('https://schema.org/JewelryStore')  
  
JobPosting: URIRef = rdflib.term.URIRef('https://schema.org/JobPosting')
```



```
JoinAction: URIRef = rdflib.term.URIRef('https://schema.org/JoinAction')
Joint: URIRef = rdflib.term.URIRef('https://schema.org/Joint')
KosherDiet: URIRef = rdflib.term.URIRef('https://schema.org/KosherDiet')
LaboratoryScience: URIRef =
rdflib.term.URIRef('https://schema.org/LaboratoryScience')
LakeBodyOfWater: URIRef = rdflib.term.URIRef('https://schema.org/LakeBodyOfWater')
Landform: URIRef = rdflib.term.URIRef('https://schema.org/Landform')
LandmarksOrHistoricalBuildings: URIRef =
rdflib.term.URIRef('https://schema.org/LandmarksOrHistoricalBuildings')
Language: URIRef = rdflib.term.URIRef('https://schema.org/Language')
LaserDiscFormat: URIRef = rdflib.term.URIRef('https://schema.org/LaserDiscFormat')
LearningResource: URIRef =
rdflib.term.URIRef('https://schema.org/LearningResource')
LeaveAction: URIRef = rdflib.term.URIRef('https://schema.org/LeaveAction')
LeftHandDriving: URIRef = rdflib.term.URIRef('https://schema.org/LeftHandDriving')
LegalForceStatus: URIRef =
rdflib.term.URIRef('https://schema.org/LegalForceStatus')
LegalService: URIRef = rdflib.term.URIRef('https://schema.org/LegalService')
LegalValueLevel: URIRef = rdflib.term.URIRef('https://schema.org/LegalValueLevel')
Legislation: URIRef = rdflib.term.URIRef('https://schema.org/Legislation')
LegislationObject: URIRef =
rdflib.term.URIRef('https://schema.org/LegislationObject')
LegislativeBuilding: URIRef =
rdflib.term.URIRef('https://schema.org/LegislativeBuilding')
LeisureTimeActivity: URIRef =
rdflib.term.URIRef('https://schema.org/LeisureTimeActivity')
LendAction: URIRef = rdflib.term.URIRef('https://schema.org/LendAction')
Library: URIRef = rdflib.term.URIRef('https://schema.org/Library')
LibrarySystem: URIRef = rdflib.term.URIRef('https://schema.org/LibrarySystem')
LifestyleModification: URIRef =
rdflib.term.URIRef('https://schema.org/LifestyleModification')
Ligament: URIRef = rdflib.term.URIRef('https://schema.org/Ligament')
LikeAction: URIRef = rdflib.term.URIRef('https://schema.org/LikeAction')
```

```
LimitedAvailability: URIRef =  
rdflib.term.URIRef('https://schema.org/LimitedAvailability')  
  
LimitedByGuaranteeCharity: URIRef =  
rdflib.term.URIRef('https://schema.org/LimitedByGuaranteeCharity')  
  
LinkRole: URIRef = rdflib.term.URIRef('https://schema.org/LinkRole')  
  
LiquorStore: URIRef = rdflib.term.URIRef('https://schema.org/LiquorStore')  
  
ListItem: URIRef = rdflib.term.URIRef('https://schema.org/ListItem')  
  
ListPrice: URIRef = rdflib.term.URIRef('https://schema.org/ListPrice')  
  
ListenAction: URIRef = rdflib.term.URIRef('https://schema.org/ListenAction')  
  
LiteraryEvent: URIRef = rdflib.term.URIRef('https://schema.org/LiteraryEvent')  
  
LiveAlbum: URIRef = rdflib.term.URIRef('https://schema.org/LiveAlbum')  
  
LiveBlogPosting: URIRef = rdflib.term.URIRef('https://schema.org/LiveBlogPosting')  
  
LivingWithHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/LivingWithHealthAspect')  
  
LoanOrCredit: URIRef = rdflib.term.URIRef('https://schema.org/LoanOrCredit')  
  
LocalBusiness: URIRef = rdflib.term.URIRef('https://schema.org/LocalBusiness')  
  
LocationFeatureSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/LocationFeatureSpecification')  
  
LockerDelivery: URIRef = rdflib.term.URIRef('https://schema.org/LockerDelivery')  
  
Locksmith: URIRef = rdflib.term.URIRef('https://schema.org/Locksmith')  
  
LodgingBusiness: URIRef = rdflib.term.URIRef('https://schema.org/LodgingBusiness')  
  
LodgingReservation: URIRef =  
rdflib.term.URIRef('https://schema.org/LodgingReservation')  
  
Longitudinal: URIRef = rdflib.term.URIRef('https://schema.org/Longitudinal')  
  
LoseAction: URIRef = rdflib.term.URIRef('https://schema.org/LoseAction')  
  
LowCalorieDiet: URIRef = rdflib.term.URIRef('https://schema.org/LowCalorieDiet')  
  
LowFatDiet: URIRef = rdflib.term.URIRef('https://schema.org/LowFatDiet')  
  
LowLactoseDiet: URIRef = rdflib.term.URIRef('https://schema.org/LowLactoseDiet')  
  
LowSaltDiet: URIRef = rdflib.term.URIRef('https://schema.org/LowSaltDiet')  
  
Lung: URIRef = rdflib.term.URIRef('https://schema.org/Lung')  
  
LymphaticVessel: URIRef = rdflib.term.URIRef('https://schema.org/LymphaticVessel')  
  
MRI: URIRef = rdflib.term.URIRef('https://schema.org/MRI')  
  
MSRP: URIRef = rdflib.term.URIRef('https://schema.org/MSRP')
```

```
Male: URIRef = rdflib.term.URIRef('https://schema.org/Male')
Manuscript: URIRef = rdflib.term.URIRef('https://schema.org/Manuscript')
Map: URIRef = rdflib.term.URIRef('https://schema.org/Map')
MapCategoryType: URIRef = rdflib.term.URIRef('https://schema.org/MapCategoryType')
MarryAction: URIRef = rdflib.term.URIRef('https://schema.org/MarryAction')
Mass: URIRef = rdflib.term.URIRef('https://schema.org/Mass')
MathSolver: URIRef = rdflib.term.URIRef('https://schema.org/MathSolver')
MaximumDoseSchedule: URIRef =
rdflib.term.URIRef('https://schema.org/MaximumDoseSchedule')
MayTreatHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/MayTreatHealthAspect')
MeasurementTypeEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/MeasurementTypeEnumeration')
MediaGallery: URIRef = rdflib.term.URIRef('https://schema.org/MediaGallery')
MediaManipulationRatingEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/MediaManipulationRatingEnumeration')
MediaObject: URIRef = rdflib.term.URIRef('https://schema.org/MediaObject')
MediaReview: URIRef = rdflib.term.URIRef('https://schema.org/MediaReview')
MediaReviewItem: URIRef = rdflib.term.URIRef('https://schema.org/MediaReviewItem')
MediaSubscription: URIRef =
rdflib.term.URIRef('https://schema.org/MediaSubscription')
MedicalAudience: URIRef = rdflib.term.URIRef('https://schema.org/MedicalAudience')
MedicalAudienceType: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalAudienceType')
MedicalBusiness: URIRef = rdflib.term.URIRef('https://schema.org/MedicalBusiness')
MedicalCause: URIRef = rdflib.term.URIRef('https://schema.org/MedicalCause')
MedicalClinic: URIRef = rdflib.term.URIRef('https://schema.org/MedicalClinic')
MedicalCode: URIRef = rdflib.term.URIRef('https://schema.org/MedicalCode')
MedicalCondition: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalCondition')
MedicalConditionStage: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalConditionStage')
MedicalContraindication: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalContraindication')
```

```
MedicalDevice: URIRef = rdflib.term.URIRef('https://schema.org/MedicalDevice')

MedicalDevicePurpose: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalDevicePurpose')

MedicalEntity: URIRef = rdflib.term.URIRef('https://schema.org/MedicalEntity')

MedicalEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalEnumeration')

MedicalEvidenceLevel: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalEvidenceLevel')

MedicalGuideline: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalGuideline')

MedicalGuidelineContraindication: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalGuidelineContraindication')

MedicalGuidelineRecommendation: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalGuidelineRecommendation')

MedicalImagingTechnique: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalImagingTechnique')

MedicalIndication: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalIndication')

MedicalIntangible: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalIntangible')

MedicalObservationalStudy: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalObservationalStudy')

MedicalObservationalStudyDesign: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalObservationalStudyDesign')

MedicalOrganization: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalOrganization')

MedicalProcedure: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalProcedure')

MedicalProcedureType: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalProcedureType')

MedicalResearcher: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalResearcher')

MedicalRiskCalculator: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalRiskCalculator')

MedicalRiskEstimator: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalRiskEstimator')

MedicalRiskFactor: URIRef =
rdflib.term.URIRef('https://schema.org/MedicalRiskFactor')
```

```
MedicalRiskScore: URIRef =  
rdflib.term.URIRef('https://schema.org/MedicalRiskScore')  
  
MedicalScholarlyArticle: URIRef =  
rdflib.term.URIRef('https://schema.org/MedicalScholarlyArticle')  
  
MedicalSign: URIRef = rdflib.term.URIRef('https://schema.org/MedicalSign')  
  
MedicalSignOrSymptom: URIRef =  
rdflib.term.URIRef('https://schema.org/MedicalSignOrSymptom')  
  
MedicalSpecialty: URIRef =  
rdflib.term.URIRef('https://schema.org/MedicalSpecialty')  
  
MedicalStudy: URIRef = rdflib.term.URIRef('https://schema.org/MedicalStudy')  
  
MedicalStudyStatus: URIRef =  
rdflib.term.URIRef('https://schema.org/MedicalStudyStatus')  
  
MedicalSymptom: URIRef = rdflib.term.URIRef('https://schema.org/MedicalSymptom')  
  
MedicalTest: URIRef = rdflib.term.URIRef('https://schema.org/MedicalTest')  
  
MedicalTestPanel: URIRef =  
rdflib.term.URIRef('https://schema.org/MedicalTestPanel')  
  
MedicalTherapy: URIRef = rdflib.term.URIRef('https://schema.org/MedicalTherapy')  
  
MedicalTrial: URIRef = rdflib.term.URIRef('https://schema.org/MedicalTrial')  
  
MedicalTrialDesign: URIRef =  
rdflib.term.URIRef('https://schema.org/MedicalTrialDesign')  
  
MedicalWebPage: URIRef = rdflib.term.URIRef('https://schema.org/MedicalWebPage')  
  
MedicineSystem: URIRef = rdflib.term.URIRef('https://schema.org/MedicineSystem')  
  
MeetingRoom: URIRef = rdflib.term.URIRef('https://schema.org/MeetingRoom')  
  
MensClothingStore: URIRef =  
rdflib.term.URIRef('https://schema.org/MensClothingStore')  
  
Menu: URIRef = rdflib.term.URIRef('https://schema.org/Menu')  
  
MenuItem: URIRef = rdflib.term.URIRef('https://schema.org/MenuItem')  
  
MenuSection: URIRef = rdflib.term.URIRef('https://schema.org/MenuSection')  
  
MerchantReturnEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/MerchantReturnEnumeration')  
  
MerchantReturnFiniteReturnWindow: URIRef =  
rdflib.term.URIRef('https://schema.org/MerchantReturnFiniteReturnWindow')  
  
MerchantReturnNotPermitted: URIRef =  
rdflib.term.URIRef('https://schema.org/MerchantReturnNotPermitted')
```

```
MerchantReturnPolicy: URIRef =  
rdflib.term.URIRef('https://schema.org/MerchantReturnPolicy')  
  
MerchantReturnPolicySeasonalOverride: URIRef =  
rdflib.term.URIRef('https://schema.org/MerchantReturnPolicySeasonalOverride')  
  
MerchantReturnUnlimitedWindow: URIRef =  
rdflib.term.URIRef('https://schema.org/MerchantReturnUnlimitedWindow')  
  
MerchantReturnUnspecified: URIRef =  
rdflib.term.URIRef('https://schema.org/MerchantReturnUnspecified')  
  
Message: URIRef = rdflib.term.URIRef('https://schema.org/Message')  
  
MiddleSchool: URIRef = rdflib.term.URIRef('https://schema.org/MiddleSchool')  
  
Midwifery: URIRef = rdflib.term.URIRef('https://schema.org/Midwifery')  
  
MinimumAdvertisedPrice: URIRef =  
rdflib.term.URIRef('https://schema.org/MinimumAdvertisedPrice')  
  
MisconceptionsHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/MisconceptionsHealthAspect')  
  
MixedEventAttendanceMode: URIRef =  
rdflib.term.URIRef('https://schema.org/MixedEventAttendanceMode')  
  
MixtapeAlbum: URIRef = rdflib.term.URIRef('https://schema.org/MixtapeAlbum')  
  
MobileApplication: URIRef =  
rdflib.term.URIRef('https://schema.org/MobileApplication')  
  
MobilePhoneStore: URIRef =  
rdflib.term.URIRef('https://schema.org/MobilePhoneStore')  
  
MolecularEntity: URIRef = rdflib.term.URIRef('https://schema.org/MolecularEntity')  
  
Monday: URIRef = rdflib.term.URIRef('https://schema.org/Monday')  
  
MonetaryAmount: URIRef = rdflib.term.URIRef('https://schema.org/MonetaryAmount')  
  
MonetaryAmountDistribution: URIRef =  
rdflib.term.URIRef('https://schema.org/MonetaryAmountDistribution')  
  
MonetaryGrant: URIRef = rdflib.term.URIRef('https://schema.org/MonetaryGrant')  
  
MoneyTransfer: URIRef = rdflib.term.URIRef('https://schema.org/MoneyTransfer')  
  
MortgageLoan: URIRef = rdflib.term.URIRef('https://schema.org/MortgageLoan')  
  
Mosque: URIRef = rdflib.term.URIRef('https://schema.org/Mosque')  
  
Motel: URIRef = rdflib.term.URIRef('https://schema.org/Motel')  
  
Motorcycle: URIRef = rdflib.term.URIRef('https://schema.org/Motorcycle')  
  
MotorcycleDealer: URIRef =  
rdflib.term.URIRef('https://schema.org/MotorcycleDealer')
```

```
MotorcycleRepair: URIRef =  
rdflib.term.URIRef('https://schema.org/MotorcycleRepair')  
  
MotorizedBicycle: URIRef =  
rdflib.term.URIRef('https://schema.org/MotorizedBicycle')  
  
Mountain: URIRef = rdflib.term.URIRef('https://schema.org/Mountain')  
  
MoveAction: URIRef = rdflib.term.URIRef('https://schema.org/MoveAction')  
  
Movie: URIRef = rdflib.term.URIRef('https://schema.org/Movie')  
  
MovieClip: URIRef = rdflib.term.URIRef('https://schema.org/MovieClip')  
  
MovieRentalStore: URIRef =  
rdflib.term.URIRef('https://schema.org/MovieRentalStore')  
  
MovieSeries: URIRef = rdflib.term.URIRef('https://schema.org/MovieSeries')  
  
MovieTheater: URIRef = rdflib.term.URIRef('https://schema.org/MovieTheater')  
  
MovingCompany: URIRef = rdflib.term.URIRef('https://schema.org/MovingCompany')  
  
MultiCenterTrial: URIRef =  
rdflib.term.URIRef('https://schema.org/MultiCenterTrial')  
  
MultiPlayer: URIRef = rdflib.term.URIRef('https://schema.org/MultiPlayer')  
  
MulticellularParasite: URIRef =  
rdflib.term.URIRef('https://schema.org/MulticellularParasite')  
  
Muscle: URIRef = rdflib.term.URIRef('https://schema.org/Muscle')  
  
Musculoskeletal: URIRef = rdflib.term.URIRef('https://schema.org/Musculoskeletal')  
  
MusculoskeletalExam: URIRef =  
rdflib.term.URIRef('https://schema.org/MusculoskeletalExam')  
  
Museum: URIRef = rdflib.term.URIRef('https://schema.org/Museum')  
  
MusicAlbum: URIRef = rdflib.term.URIRef('https://schema.org/MusicAlbum')  
  
MusicAlbumProductionType: URIRef =  
rdflib.term.URIRef('https://schema.org/MusicAlbumProductionType')  
  
MusicAlbumReleaseType: URIRef =  
rdflib.term.URIRef('https://schema.org/MusicAlbumReleaseType')  
  
MusicComposition: URIRef =  
rdflib.term.URIRef('https://schema.org/MusicComposition')  
  
MusicEvent: URIRef = rdflib.term.URIRef('https://schema.org/MusicEvent')  
  
MusicGroup: URIRef = rdflib.term.URIRef('https://schema.org/MusicGroup')  
  
MusicPlaylist: URIRef = rdflib.term.URIRef('https://schema.org/MusicPlaylist')  
  
MusicRecording: URIRef = rdflib.term.URIRef('https://schema.org/MusicRecording')
```



```
MusicRelease: URIRef = rdflib.term.URIRef('https://schema.org/MusicRelease')

MusicReleaseFormatType: URIRef =
rdflib.term.URIRef('https://schema.org/MusicReleaseFormatType')

MusicStore: URIRef = rdflib.term.URIRef('https://schema.org/MusicStore')

MusicVenue: URIRef = rdflib.term.URIRef('https://schema.org/MusicVenue')

MusicVideoObject: URIRef =
rdflib.term.URIRef('https://schema.org/MusicVideoObject')

NGO: URIRef = rdflib.term.URIRef('https://schema.org/NGO')

NLNonprofitType: URIRef = rdflib.term.URIRef('https://schema.org/NLNonprofitType')

NailSalon: URIRef = rdflib.term.URIRef('https://schema.org/NailSalon')

Neck: URIRef = rdflib.term.URIRef('https://schema.org/Neck')

Nerve: URIRef = rdflib.term.URIRef('https://schema.org/Nerve')

Neuro: URIRef = rdflib.term.URIRef('https://schema.org/Neuro')

Neurologic: URIRef = rdflib.term.URIRef('https://schema.org/Neurologic')

NewCondition: URIRef = rdflib.term.URIRef('https://schema.org/NewCondition')

NewsArticle: URIRef = rdflib.term.URIRef('https://schema.org/NewsArticle')

NewsMediaOrganization: URIRef =
rdflib.term.URIRef('https://schema.org/NewsMediaOrganization')

Newspaper: URIRef = rdflib.term.URIRef('https://schema.org/Newspaper')

NightClub: URIRef = rdflib.term.URIRef('https://schema.org/NightClub')

NoninvasiveProcedure: URIRef =
rdflib.term.URIRef('https://schema.org/NoninvasiveProcedure')

Nonprofit501a: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501a')

Nonprofit501c1: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c1')

Nonprofit501c10: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c10')

Nonprofit501c11: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c11')

Nonprofit501c12: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c12')

Nonprofit501c13: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c13')

Nonprofit501c14: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c14')

Nonprofit501c15: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c15')

Nonprofit501c16: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c16')

Nonprofit501c17: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c17')
```



```
Nonprofit501c18: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c18')
Nonprofit501c19: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c19')
Nonprofit501c2: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c2')
Nonprofit501c20: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c20')
Nonprofit501c21: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c21')
Nonprofit501c22: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c22')
Nonprofit501c23: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c23')
Nonprofit501c24: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c24')
Nonprofit501c25: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c25')
Nonprofit501c26: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c26')
Nonprofit501c27: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c27')
Nonprofit501c28: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c28')
Nonprofit501c3: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c3')
Nonprofit501c4: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c4')
Nonprofit501c5: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c5')
Nonprofit501c6: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c6')
Nonprofit501c7: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c7')
Nonprofit501c8: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c8')
Nonprofit501c9: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501c9')
Nonprofit501d: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501d')
Nonprofit501e: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501e')
Nonprofit501f: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501f')
Nonprofit501k: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501k')
Nonprofit501n: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501n')
Nonprofit501q: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit501q')
Nonprofit527: URIRef = rdflib.term.URIRef('https://schema.org/Nonprofit527')
NonprofitANBI: URIRef = rdflib.term.URIRef('https://schema.org/NonprofitANBI')
NonprofitSBBi: URIRef = rdflib.term.URIRef('https://schema.org/NonprofitSBBi')
NonprofitType: URIRef = rdflib.term.URIRef('https://schema.org/NonprofitType')
Nose: URIRef = rdflib.term.URIRef('https://schema.org/Nose')
NotInForce: URIRef = rdflib.term.URIRef('https://schema.org/NotInForce')
```

```
NotYetRecruiting: URIRef =  
rdflib.term.URIRef('https://schema.org/NotYetRecruiting')  
  
Notary: URIRef = rdflib.term.URIRef('https://schema.org/Notary')  
  
NoteDigitalDocument: URIRef =  
rdflib.term.URIRef('https://schema.org/NoteDigitalDocument')  
  
Number: URIRef = rdflib.term.URIRef('https://schema.org/Number')  
  
Nursing: URIRef = rdflib.term.URIRef('https://schema.org/Nursing')  
  
NutritionInformation: URIRef =  
rdflib.term.URIRef('https://schema.org/NutritionInformation')  
  
OTC: URIRef = rdflib.term.URIRef('https://schema.org/OTC')  
  
Observation: URIRef = rdflib.term.URIRef('https://schema.org/Observation')  
  
Observational: URIRef = rdflib.term.URIRef('https://schema.org/Observational')  
  
Obstetric: URIRef = rdflib.term.URIRef('https://schema.org/Obstetric')  
  
Occupation: URIRef = rdflib.term.URIRef('https://schema.org/Occupation')  
  
OccupationalActivity: URIRef =  
rdflib.term.URIRef('https://schema.org/OccupationalActivity')  
  
OccupationalExperienceRequirements: URIRef =  
rdflib.term.URIRef('https://schema.org/OccupationalExperienceRequirements')  
  
OccupationalTherapy: URIRef =  
rdflib.term.URIRef('https://schema.org/OccupationalTherapy')  
  
OceanBodyOfWater: URIRef =  
rdflib.term.URIRef('https://schema.org/OceanBodyOfWater')  
  
Offer: URIRef = rdflib.term.URIRef('https://schema.org/Offer')  
  
OfferCatalog: URIRef = rdflib.term.URIRef('https://schema.org/OfferCatalog')  
  
OfferForLease: URIRef = rdflib.term.URIRef('https://schema.org/OfferForLease')  
  
OfferForPurchase: URIRef =  
rdflib.term.URIRef('https://schema.org/OfferForPurchase')  
  
OfferItemCondition: URIRef =  
rdflib.term.URIRef('https://schema.org/OfferItemCondition')  
  
OfferShippingDetails: URIRef =  
rdflib.term.URIRef('https://schema.org/OfferShippingDetails')  
  
OfficeEquipmentStore: URIRef =  
rdflib.term.URIRef('https://schema.org/OfficeEquipmentStore')  
  
OfficialLegalValue: URIRef =  
rdflib.term.URIRef('https://schema.org/OfficialLegalValue')
```

```
OfflineEventAttendanceMode: URIRef =  
rdflib.term.URIRef('https://schema.org/OfflineEventAttendanceMode')  
  
OfflinePermanently: URIRef =  
rdflib.term.URIRef('https://schema.org/OfflinePermanently')  
  
OfflineTemporarily: URIRef =  
rdflib.term.URIRef('https://schema.org/OfflineTemporarily')  
  
OnDemandEvent: URIRef = rdflib.term.URIRef('https://schema.org/OnDemandEvent')  
  
OnSitePickup: URIRef = rdflib.term.URIRef('https://schema.org/OnSitePickup')  
  
Oncologic: URIRef = rdflib.term.URIRef('https://schema.org/Oncologic')  
  
OneTimePayments: URIRef = rdflib.term.URIRef('https://schema.org/OneTimePayments')  
  
Online: URIRef = rdflib.term.URIRef('https://schema.org/Online')  
  
OnlineEventAttendanceMode: URIRef =  
rdflib.term.URIRef('https://schema.org/OnlineEventAttendanceMode')  
  
OnlineFull: URIRef = rdflib.term.URIRef('https://schema.org/OnlineFull')  
  
OnlineOnly: URIRef = rdflib.term.URIRef('https://schema.org/OnlineOnly')  
  
OpenTrial: URIRef = rdflib.term.URIRef('https://schema.org/OpenTrial')  
  
OpeningHoursSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/OpeningHoursSpecification')  
  
OpinionNewsArticle: URIRef =  
rdflib.term.URIRef('https://schema.org/OpinionNewsArticle')  
  
Optician: URIRef = rdflib.term.URIRef('https://schema.org/Optician')  
  
Optometric: URIRef = rdflib.term.URIRef('https://schema.org/Optometric')  
  
Order: URIRef = rdflib.term.URIRef('https://schema.org/Order')  
  
OrderAction: URIRef = rdflib.term.URIRef('https://schema.org/OrderAction')  
  
OrderCancelled: URIRef = rdflib.term.URIRef('https://schema.org/OrderCancelled')  
  
OrderDelivered: URIRef = rdflib.term.URIRef('https://schema.org/OrderDelivered')  
  
OrderInTransit: URIRef = rdflib.term.URIRef('https://schema.org/OrderInTransit')  
  
OrderItem: URIRef = rdflib.term.URIRef('https://schema.org/OrderItem')  
  
OrderPaymentDue: URIRef = rdflib.term.URIRef('https://schema.org/OrderPaymentDue')  
  
OrderPickupAvailable: URIRef =  
rdflib.term.URIRef('https://schema.org/OrderPickupAvailable')  
  
OrderProblem: URIRef = rdflib.term.URIRef('https://schema.org/OrderProblem')  
  
OrderProcessing: URIRef = rdflib.term.URIRef('https://schema.org/OrderProcessing')
```

```
OrderReturned: URIRef = rdflib.term.URIRef('https://schema.org/OrderReturned')
OrderStatus: URIRef = rdflib.term.URIRef('https://schema.org/OrderStatus')
Organization: URIRef = rdflib.term.URIRef('https://schema.org/Organization')
OrganizationRole: URIRef =
rdflib.term.URIRef('https://schema.org/OrganizationRole')
OrganizeAction: URIRef = rdflib.term.URIRef('https://schema.org/OrganizeAction')
OriginalMediaContent: URIRef =
rdflib.term.URIRef('https://schema.org/OriginalMediaContent')
OriginalShippingFees: URIRef =
rdflib.term.URIRef('https://schema.org/OriginalShippingFees')
Osteopathic: URIRef = rdflib.term.URIRef('https://schema.org/Osteopathic')
Otolaryngologic: URIRef = rdflib.term.URIRef('https://schema.org/Otolaryngologic')
OutOfStock: URIRef = rdflib.term.URIRef('https://schema.org/OutOfStock')
OutletStore: URIRef = rdflib.term.URIRef('https://schema.org/OutletStore')
OverviewHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/OverviewHealthAspect')
OwnershipInfo: URIRef = rdflib.term.URIRef('https://schema.org/OwnershipInfo')
PET: URIRef = rdflib.term.URIRef('https://schema.org/PET')
PaidLeave: URIRef = rdflib.term.URIRef('https://schema.org/PaidLeave')
PaintAction: URIRef = rdflib.term.URIRef('https://schema.org/PaintAction')
Painting: URIRef = rdflib.term.URIRef('https://schema.org/Painting')
PalliativeProcedure: URIRef =
rdflib.term.URIRef('https://schema.org/PalliativeProcedure')
Paperback: URIRef = rdflib.term.URIRef('https://schema.org/Paperback')
ParcelDelivery: URIRef = rdflib.term.URIRef('https://schema.org/ParcelDelivery')
ParcelService: URIRef = rdflib.term.URIRef('https://schema.org/ParcelService')
ParentAudience: URIRef = rdflib.term.URIRef('https://schema.org/ParentAudience')
ParentalSupport: URIRef = rdflib.term.URIRef('https://schema.org/ParentalSupport')
Park: URIRef = rdflib.term.URIRef('https://schema.org/Park')
ParkingFacility: URIRef = rdflib.term.URIRef('https://schema.org/ParkingFacility')
ParkingMap: URIRef = rdflib.term.URIRef('https://schema.org/ParkingMap')
PartiallyInForce: URIRef =
rdflib.term.URIRef('https://schema.org/PartiallyInForce')
```

```
Pathology: URIRef = rdflib.term.URIRef('https://schema.org/Pathology')
PathologyTest: URIRef = rdflib.term.URIRef('https://schema.org/PathologyTest')
Patient: URIRef = rdflib.term.URIRef('https://schema.org/Patient')
PatientExperienceHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/PatientExperienceHealthAspect')
PawnShop: URIRef = rdflib.term.URIRef('https://schema.org/PawnShop')
PayAction: URIRef = rdflib.term.URIRef('https://schema.org/PayAction')
PaymentAutomaticallyApplied: URIRef =
rdflib.term.URIRef('https://schema.org/PaymentAutomaticallyApplied')
PaymentCard: URIRef = rdflib.term.URIRef('https://schema.org/PaymentCard')
PaymentChargeSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/PaymentChargeSpecification')
PaymentComplete: URIRef = rdflib.term.URIRef('https://schema.org/PaymentComplete')
PaymentDeclined: URIRef = rdflib.term.URIRef('https://schema.org/PaymentDeclined')
PaymentDue: URIRef = rdflib.term.URIRef('https://schema.org/PaymentDue')
PaymentMethod: URIRef = rdflib.term.URIRef('https://schema.org/PaymentMethod')
PaymentPastDue: URIRef = rdflib.term.URIRef('https://schema.org/PaymentPastDue')
PaymentService: URIRef = rdflib.term.URIRef('https://schema.org/PaymentService')
PaymentStatusType: URIRef =
rdflib.term.URIRef('https://schema.org/PaymentStatusType')
Pediatric: URIRef = rdflib.term.URIRef('https://schema.org/Pediatric')
PeopleAudience: URIRef = rdflib.term.URIRef('https://schema.org/PeopleAudience')
PercutaneousProcedure: URIRef =
rdflib.term.URIRef('https://schema.org/PercutaneousProcedure')
PerformAction: URIRef = rdflib.term.URIRef('https://schema.org/PerformAction')
PerformanceRole: URIRef = rdflib.term.URIRef('https://schema.org/PerformanceRole')
PerformingArtsTheater: URIRef =
rdflib.term.URIRef('https://schema.org/PerformingArtsTheater')
PerformingGroup: URIRef = rdflib.term.URIRef('https://schema.org/PerformingGroup')
Periodical: URIRef = rdflib.term.URIRef('https://schema.org/Periodical')
Permit: URIRef = rdflib.term.URIRef('https://schema.org/Permit')
Person: URIRef = rdflib.term.URIRef('https://schema.org/Person')
PetStore: URIRef = rdflib.term.URIRef('https://schema.org/PetStore')
```

```
Pharmacy: URIRef = rdflib.term.URIRef('https://schema.org/Pharmacy')

PharmacySpecialty: URIRef =
rdflib.term.URIRef('https://schema.org/PharmacySpecialty')

Photograph: URIRef = rdflib.term.URIRef('https://schema.org/Photograph')

PhotographAction: URIRef =
rdflib.term.URIRef('https://schema.org/PhotographAction')

PhysicalActivity: URIRef =
rdflib.term.URIRef('https://schema.org/PhysicalActivity')

PhysicalActivityCategory: URIRef =
rdflib.term.URIRef('https://schema.org/PhysicalActivityCategory')

PhysicalExam: URIRef = rdflib.term.URIRef('https://schema.org/PhysicalExam')

PhysicalTherapy: URIRef = rdflib.term.URIRef('https://schema.org/PhysicalTherapy')

Physician: URIRef = rdflib.term.URIRef('https://schema.org/Physician')

Physiotherapy: URIRef = rdflib.term.URIRef('https://schema.org/Physiotherapy')

Place: URIRef = rdflib.term.URIRef('https://schema.org/Place')

PlaceOfWorship: URIRef = rdflib.term.URIRef('https://schema.org/PlaceOfWorship')

PlaceboControlledTrial: URIRef =
rdflib.term.URIRef('https://schema.org/PlaceboControlledTrial')

PlanAction: URIRef = rdflib.term.URIRef('https://schema.org/PlanAction')

PlasticSurgery: URIRef = rdflib.term.URIRef('https://schema.org/PlasticSurgery')

Play: URIRef = rdflib.term.URIRef('https://schema.org/Play')

PlayAction: URIRef = rdflib.term.URIRef('https://schema.org/PlayAction')

Playground: URIRef = rdflib.term.URIRef('https://schema.org/Playground')

Plumber: URIRef = rdflib.term.URIRef('https://schema.org/Plumber')

PodcastEpisode: URIRef = rdflib.term.URIRef('https://schema.org/PodcastEpisode')

PodcastSeason: URIRef = rdflib.term.URIRef('https://schema.org/PodcastSeason')

PodcastSeries: URIRef = rdflib.term.URIRef('https://schema.org/PodcastSeries')

Podiatric: URIRef = rdflib.term.URIRef('https://schema.org/Podiatric')

PoliceStation: URIRef = rdflib.term.URIRef('https://schema.org/PoliceStation')

Pond: URIRef = rdflib.term.URIRef('https://schema.org/Pond')

PostOffice: URIRef = rdflib.term.URIRef('https://schema.org/PostOffice')

PostalAddress: URIRef = rdflib.term.URIRef('https://schema.org/PostalAddress')
```

```
PostalCodeRangeSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/PostalCodeRangeSpecification')  
  
Poster: URIRef = rdflib.term.URIRef('https://schema.org/Poster')  
  
PotentialActionStatus: URIRef =  
rdflib.term.URIRef('https://schema.org/PotentialActionStatus')  
  
PreOrder: URIRef = rdflib.term.URIRef('https://schema.org/PreOrder')  
  
PreOrderAction: URIRef = rdflib.term.URIRef('https://schema.org/PreOrderAction')  
  
PreSale: URIRef = rdflib.term.URIRef('https://schema.org/PreSale')  
  
PregnancyHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/PregnancyHealthAspect')  
  
PrependAction: URIRef = rdflib.term.URIRef('https://schema.org/PrependAction')  
  
Preschool: URIRef = rdflib.term.URIRef('https://schema.org/Preschool')  
  
PrescriptionOnly: URIRef =  
rdflib.term.URIRef('https://schema.org/PrescriptionOnly')  
  
PresentationDigitalDocument: URIRef =  
rdflib.term.URIRef('https://schema.org/PresentationDigitalDocument')  
  
PreventionHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/PreventionHealthAspect')  
  
PreventionIndication: URIRef =  
rdflib.term.URIRef('https://schema.org/PreventionIndication')  
  
PriceComponentTypeEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/PriceComponentTypeEnumeration')  
  
PriceSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/PriceSpecification')  
  
PriceTypeEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/PriceTypeEnumeration')  
  
PrimaryCare: URIRef = rdflib.term.URIRef('https://schema.org/PrimaryCare')  
  
Prion: URIRef = rdflib.term.URIRef('https://schema.org/Prion')  
  
Product: URIRef = rdflib.term.URIRef('https://schema.org/Product')  
  
ProductCollection: URIRef =  
rdflib.term.URIRef('https://schema.org/ProductCollection')  
  
ProductGroup: URIRef = rdflib.term.URIRef('https://schema.org/ProductGroup')  
  
ProductModel: URIRef = rdflib.term.URIRef('https://schema.org/ProductModel')  
  
ProfessionalService: URIRef =  
rdflib.term.URIRef('https://schema.org/ProfessionalService')
```



```

ProfilePage: URIRef = rdflib.term.URIRef('https://schema.org/ProfilePage')

PrognosisHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/PrognosisHealthAspect')

ProgramMembership: URIRef =
rdflib.term.URIRef('https://schema.org/ProgramMembership')

Project: URIRef = rdflib.term.URIRef('https://schema.org/Project')

PronounceableText: URIRef =
rdflib.term.URIRef('https://schema.org/PronounceableText')

Property: URIRef = rdflib.term.URIRef('https://schema.org/Property')

PropertyValue: URIRef = rdflib.term.URIRef('https://schema.org/PropertyValue')

PropertyValueSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/PropertyValueSpecification')

Protein: URIRef = rdflib.term.URIRef('https://schema.org/Protein')

Protozoa: URIRef = rdflib.term.URIRef('https://schema.org/Protozoa')

Psychiatric: URIRef = rdflib.term.URIRef('https://schema.org/Psychiatric')

PsychologicalTreatment: URIRef =
rdflib.term.URIRef('https://schema.org/PsychologicalTreatment')

PublicHealth: URIRef = rdflib.term.URIRef('https://schema.org/PublicHealth')

PublicHolidays: URIRef = rdflib.term.URIRef('https://schema.org/PublicHolidays')

PublicSwimmingPool: URIRef =
rdflib.term.URIRef('https://schema.org/PublicSwimmingPool')

PublicToilet: URIRef = rdflib.term.URIRef('https://schema.org/PublicToilet')

PublicationEvent: URIRef =
rdflib.term.URIRef('https://schema.org/PublicationEvent')

PublicationIssue: URIRef =
rdflib.term.URIRef('https://schema.org/PublicationIssue')

PublicationVolume: URIRef =
rdflib.term.URIRef('https://schema.org/PublicationVolume')

Pulmonary: URIRef = rdflib.term.URIRef('https://schema.org/Pulmonary')

QAPage: URIRef = rdflib.term.URIRef('https://schema.org/QAPage')

QualitativeValue: URIRef =
rdflib.term.URIRef('https://schema.org/QualitativeValue')

QuantitativeValue: URIRef =
rdflib.term.URIRef('https://schema.org/QuantitativeValue')

```



```
QuantitativeValueDistribution: URIRef =  
rdflib.term.URIRef('https://schema.org/QuantitativeValueDistribution')  
  
Quantity: URIRef = rdflib.term.URIRef('https://schema.org/Quantity')  
  
Question: URIRef = rdflib.term.URIRef('https://schema.org/Question')  
  
Quiz: URIRef = rdflib.term.URIRef('https://schema.org/Quiz')  
  
Quotation: URIRef = rdflib.term.URIRef('https://schema.org/Quotation')  
  
QuoteAction: URIRef = rdflib.term.URIRef('https://schema.org/QuoteAction')  
  
RVPark: URIRef = rdflib.term.URIRef('https://schema.org/RVPark')  
  
RadiationTherapy: URIRef =  
rdflib.term.URIRef('https://schema.org/RadiationTherapy')  
  
RadioBroadcastService: URIRef =  
rdflib.term.URIRef('https://schema.org/RadioBroadcastService')  
  
RadioChannel: URIRef = rdflib.term.URIRef('https://schema.org/RadioChannel')  
  
RadioClip: URIRef = rdflib.term.URIRef('https://schema.org/RadioClip')  
  
RadioEpisode: URIRef = rdflib.term.URIRef('https://schema.org/RadioEpisode')  
  
RadioSeason: URIRef = rdflib.term.URIRef('https://schema.org/RadioSeason')  
  
RadioSeries: URIRef = rdflib.term.URIRef('https://schema.org/RadioSeries')  
  
RadioStation: URIRef = rdflib.term.URIRef('https://schema.org/RadioStation')  
  
Radiography: URIRef = rdflib.term.URIRef('https://schema.org/Radiography')  
  
RandomizedTrial: URIRef = rdflib.term.URIRef('https://schema.org/RandomizedTrial')  
  
Rating: URIRef = rdflib.term.URIRef('https://schema.org/Rating')  
  
ReactAction: URIRef = rdflib.term.URIRef('https://schema.org/ReactAction')  
  
ReadAction: URIRef = rdflib.term.URIRef('https://schema.org/ReadAction')  
  
ReadPermission: URIRef = rdflib.term.URIRef('https://schema.org/ReadPermission')  
  
RealEstateAgent: URIRef = rdflib.term.URIRef('https://schema.org/RealEstateAgent')  
  
RealEstateListing: URIRef =  
rdflib.term.URIRef('https://schema.org/RealEstateListing')  
  
RearWheelDriveConfiguration: URIRef =  
rdflib.term.URIRef('https://schema.org/RearWheelDriveConfiguration')  
  
ReceiveAction: URIRef = rdflib.term.URIRef('https://schema.org/ReceiveAction')  
  
Recipe: URIRef = rdflib.term.URIRef('https://schema.org/Recipe')  
  
Recommendation: URIRef = rdflib.term.URIRef('https://schema.org/Recommendation')
```

```
RecommendedDoseSchedule: URIRef =  
rdflib.term.URIRef('https://schema.org/RecommendedDoseSchedule')  
  
Recruiting: URIRef = rdflib.term.URIRef('https://schema.org/Recruiting')  
  
RecyclingCenter: URIRef = rdflib.term.URIRef('https://schema.org/RecyclingCenter')  
  
RefundTypeEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/RefundTypeEnumeration')  
  
RefurbishedCondition: URIRef =  
rdflib.term.URIRef('https://schema.org/RefurbishedCondition')  
  
RegisterAction: URIRef = rdflib.term.URIRef('https://schema.org/RegisterAction')  
  
Registry: URIRef = rdflib.term.URIRef('https://schema.org/Registry')  
  
ReimbursementCap: URIRef =  
rdflib.term.URIRef('https://schema.org/ReimbursementCap')  
  
RejectAction: URIRef = rdflib.term.URIRef('https://schema.org/RejectAction')  
  
RelatedTopicsHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/RelatedTopicsHealthAspect')  
  
RemixAlbum: URIRef = rdflib.term.URIRef('https://schema.org/RemixAlbum')  
  
Renal: URIRef = rdflib.term.URIRef('https://schema.org/Renal')  
  
RentAction: URIRef = rdflib.term.URIRef('https://schema.org/RentAction')  
  
RentalCarReservation: URIRef =  
rdflib.term.URIRef('https://schema.org/RentalCarReservation')  
  
RentalVehicleUsage: URIRef =  
rdflib.term.URIRef('https://schema.org/RentalVehicleUsage')  
  
RepaymentSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/RepaymentSpecification')  
  
ReplaceAction: URIRef = rdflib.term.URIRef('https://schema.org/ReplaceAction')  
  
ReplyAction: URIRef = rdflib.term.URIRef('https://schema.org/ReplyAction')  
  
Report: URIRef = rdflib.term.URIRef('https://schema.org/Report')  
  
ReportageNewsArticle: URIRef =  
rdflib.term.URIRef('https://schema.org/ReportageNewsArticle')  
  
ReportedDoseSchedule: URIRef =  
rdflib.term.URIRef('https://schema.org/ReportedDoseSchedule')  
  
ResearchOrganization: URIRef =  
rdflib.term.URIRef('https://schema.org/ResearchOrganization')  
  
ResearchProject: URIRef = rdflib.term.URIRef('https://schema.org/ResearchProject')  
  
Researcher: URIRef = rdflib.term.URIRef('https://schema.org/Researcher')
```

```
Reservation: URIRef = rdflib.term.URIRef('https://schema.org/Reservation')

ReservationCancelled: URIRef =
rdflib.term.URIRef('https://schema.org/ReservationCancelled')

ReservationConfirmed: URIRef =
rdflib.term.URIRef('https://schema.org/ReservationConfirmed')

ReservationHold: URIRef = rdflib.term.URIRef('https://schema.org/ReservationHold')

ReservationPackage: URIRef =
rdflib.term.URIRef('https://schema.org/ReservationPackage')

ReservationPending: URIRef =
rdflib.term.URIRef('https://schema.org/ReservationPending')

ReservationStatusType: URIRef =
rdflib.term.URIRef('https://schema.org/ReservationStatusType')

ReserveAction: URIRef = rdflib.term.URIRef('https://schema.org/ReserveAction')

Reservoir: URIRef = rdflib.term.URIRef('https://schema.org/Reservoir')

Residence: URIRef = rdflib.term.URIRef('https://schema.org/Residence')

Resort: URIRef = rdflib.term.URIRef('https://schema.org/Resort')

RespiratoryTherapy: URIRef =
rdflib.term.URIRef('https://schema.org/RespiratoryTherapy')

Restaurant: URIRef = rdflib.term.URIRef('https://schema.org/Restaurant')

RestockingFees: URIRef = rdflib.term.URIRef('https://schema.org/RestockingFees')

RestrictedDiet: URIRef = rdflib.term.URIRef('https://schema.org/RestrictedDiet')

ResultsAvailable: URIRef =
rdflib.term.URIRef('https://schema.org/ResultsAvailable')

ResultsNotAvailable: URIRef =
rdflib.term.URIRef('https://schema.org/ResultsNotAvailable')

ResumeAction: URIRef = rdflib.term.URIRef('https://schema.org/ResumeAction')

Retail: URIRef = rdflib.term.URIRef('https://schema.org/Retail')

ReturnAction: URIRef = rdflib.term.URIRef('https://schema.org/ReturnAction')

ReturnAtKiosk: URIRef = rdflib.term.URIRef('https://schema.org/ReturnAtKiosk')

ReturnByMail: URIRef = rdflib.term.URIRef('https://schema.org/ReturnByMail')

ReturnFeesCustomerResponsibility: URIRef =
rdflib.term.URIRef('https://schema.org/ReturnFeesCustomerResponsibility')

ReturnFeesEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/ReturnFeesEnumeration')

ReturnInStore: URIRef = rdflib.term.URIRef('https://schema.org/ReturnInStore')
```

```
ReturnLabelCustomerResponsibility: URIRef =  
rdflib.term.URIRef('https://schema.org/ReturnLabelCustomerResponsibility')  
  
ReturnLabelDownloadAndPrint: URIRef =  
rdflib.term.URIRef('https://schema.org/ReturnLabelDownloadAndPrint')  
  
ReturnLabelInBox: URIRef =  
rdflib.term.URIRef('https://schema.org/ReturnLabelInBox')  
  
ReturnLabelSourceEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/ReturnLabelSourceEnumeration')  
  
ReturnMethodEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/ReturnMethodEnumeration')  
  
ReturnShippingFees: URIRef =  
rdflib.term.URIRef('https://schema.org/ReturnShippingFees')  
  
Review: URIRef = rdflib.term.URIRef('https://schema.org/Review')  
  
ReviewAction: URIRef = rdflib.term.URIRef('https://schema.org/ReviewAction')  
  
ReviewNewsArticle: URIRef =  
rdflib.term.URIRef('https://schema.org/ReviewNewsArticle')  
  
Rheumatologic: URIRef = rdflib.term.URIRef('https://schema.org/Rheumatologic')  
  
RightHandDriving: URIRef =  
rdflib.term.URIRef('https://schema.org/RightHandDriving')  
  
RisksOrComplicationsHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/RisksOrComplicationsHealthAspect')  
  
RiverBodyOfWater: URIRef =  
rdflib.term.URIRef('https://schema.org/RiverBodyOfWater')  
  
Role: URIRef = rdflib.term.URIRef('https://schema.org/Role')  
  
RoofingContractor: URIRef =  
rdflib.term.URIRef('https://schema.org/RoofingContractor')  
  
Room: URIRef = rdflib.term.URIRef('https://schema.org/Room')  
  
RsvpAction: URIRef = rdflib.term.URIRef('https://schema.org/RsvpAction')  
  
RsvpResponseMaybe: URIRef =  
rdflib.term.URIRef('https://schema.org/RsvpResponseMaybe')  
  
RsvpResponseNo: URIRef = rdflib.term.URIRef('https://schema.org/RsvpResponseNo')  
  
RsvpResponseType: URIRef =  
rdflib.term.URIRef('https://schema.org/RsvpResponseType')  
  
RsvpResponseYes: URIRef = rdflib.term.URIRef('https://schema.org/RsvpResponseYes')  
  
SRP: URIRef = rdflib.term.URIRef('https://schema.org/SRP')
```

```
SafetyHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/SafetyHealthAspect')  
  
SaleEvent: URIRef = rdflib.term.URIRef('https://schema.org/SaleEvent')  
  
SalePrice: URIRef = rdflib.term.URIRef('https://schema.org/SalePrice')  
  
SatireOrParodyContent: URIRef =  
rdflib.term.URIRef('https://schema.org/SatireOrParodyContent')  
  
SatiricalArticle: URIRef =  
rdflib.term.URIRef('https://schema.org/SatiricalArticle')  
  
Saturday: URIRef = rdflib.term.URIRef('https://schema.org/Saturday')  
  
Schedule: URIRef = rdflib.term.URIRef('https://schema.org/Schedule')  
  
ScheduleAction: URIRef = rdflib.term.URIRef('https://schema.org/ScheduleAction')  
  
ScholarlyArticle: URIRef =  
rdflib.term.URIRef('https://schema.org/ScholarlyArticle')  
  
School: URIRef = rdflib.term.URIRef('https://schema.org/School')  
  
SchoolDistrict: URIRef = rdflib.term.URIRef('https://schema.org/SchoolDistrict')  
  
ScreeningEvent: URIRef = rdflib.term.URIRef('https://schema.org/ScreeningEvent')  
  
ScreeningHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/ScreeningHealthAspect')  
  
Sculpture: URIRef = rdflib.term.URIRef('https://schema.org/Sculpture')  
  
SeaBodyOfWater: URIRef = rdflib.term.URIRef('https://schema.org/SeaBodyOfWater')  
  
SearchAction: URIRef = rdflib.term.URIRef('https://schema.org/SearchAction')  
  
SearchResultsPage: URIRef =  
rdflib.term.URIRef('https://schema.org/SearchResultsPage')  
  
Season: URIRef = rdflib.term.URIRef('https://schema.org/Season')  
  
Seat: URIRef = rdflib.term.URIRef('https://schema.org/Seat')  
  
SeatingMap: URIRef = rdflib.term.URIRef('https://schema.org/SeatingMap')  
  
SeeDoctorHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/SeeDoctorHealthAspect')  
  
SeekToAction: URIRef = rdflib.term.URIRef('https://schema.org/SeekToAction')  
  
SelfCareHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/SelfCareHealthAspect')  
  
SelfStorage: URIRef = rdflib.term.URIRef('https://schema.org/SelfStorage')  
  
SellAction: URIRef = rdflib.term.URIRef('https://schema.org/SellAction')  
  
SendAction: URIRef = rdflib.term.URIRef('https://schema.org/SendAction')
```

```
Series: URIRef = rdflib.term.URIRef('https://schema.org/Series')
Service: URIRef = rdflib.term.URIRef('https://schema.org/Service')
ServiceChannel: URIRef = rdflib.term.URIRef('https://schema.org/ServiceChannel')
ShareAction: URIRef = rdflib.term.URIRef('https://schema.org/ShareAction')
SheetMusic: URIRef = rdflib.term.URIRef('https://schema.org/SheetMusic')
ShippingDeliveryTime: URIRef =
rdflib.term.URIRef('https://schema.org/ShippingDeliveryTime')
ShippingRateSettings: URIRef =
rdflib.term.URIRef('https://schema.org/ShippingRateSettings')
ShoeStore: URIRef = rdflib.term.URIRef('https://schema.org/ShoeStore')
ShoppingCenter: URIRef = rdflib.term.URIRef('https://schema.org/ShoppingCenter')
ShortStory: URIRef = rdflib.term.URIRef('https://schema.org/ShortStory')
SideEffectsHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/SideEffectsHealthAspect')
SingleBlindedTrial: URIRef =
rdflib.term.URIRef('https://schema.org/SingleBlindedTrial')
SingleCenterTrial: URIRef =
rdflib.term.URIRef('https://schema.org/SingleCenterTrial')
SingleFamilyResidence: URIRef =
rdflib.term.URIRef('https://schema.org/SingleFamilyResidence')
SinglePlayer: URIRef = rdflib.term.URIRef('https://schema.org/SinglePlayer')
SingleRelease: URIRef = rdflib.term.URIRef('https://schema.org/SingleRelease')
SiteNavigationElement: URIRef =
rdflib.term.URIRef('https://schema.org/SiteNavigationElement')
SizeGroupEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/SizeGroupEnumeration')
SizeSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/SizeSpecification')
SizeSystemEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/SizeSystemEnumeration')
SizeSystemImperial: URIRef =
rdflib.term.URIRef('https://schema.org/SizeSystemImperial')
SizeSystemMetric: URIRef =
rdflib.term.URIRef('https://schema.org/SizeSystemMetric')
SkiResort: URIRef = rdflib.term.URIRef('https://schema.org/SkiResort')
```

```
Skin: URIRef = rdflib.term.URIRef('https://schema.org/Skin')

SocialEvent: URIRef = rdflib.term.URIRef('https://schema.org/SocialEvent')

SocialMediaPosting: URIRef =
rdflib.term.URIRef('https://schema.org/SocialMediaPosting')

SoftwareApplication: URIRef =
rdflib.term.URIRef('https://schema.org/SoftwareApplication')

SoftwareSourceCode: URIRef =
rdflib.term.URIRef('https://schema.org/SoftwareSourceCode')

SoldOut: URIRef = rdflib.term.URIRef('https://schema.org/SoldOut')

SolveMathAction: URIRef = rdflib.term.URIRef('https://schema.org/SolveMathAction')

SomeProducts: URIRef = rdflib.term.URIRef('https://schema.org/SomeProducts')

SoundtrackAlbum: URIRef = rdflib.term.URIRef('https://schema.org/SoundtrackAlbum')

SpeakableSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/SpeakableSpecification')

SpecialAnnouncement: URIRef =
rdflib.term.URIRef('https://schema.org/SpecialAnnouncement')

Specialty: URIRef = rdflib.term.URIRef('https://schema.org/Specialty')

SpeechPathology: URIRef = rdflib.term.URIRef('https://schema.org/SpeechPathology')

SpokenWordAlbum: URIRef = rdflib.term.URIRef('https://schema.org/SpokenWordAlbum')

SportingGoodsStore: URIRef =
rdflib.term.URIRef('https://schema.org/SportingGoodsStore')

SportsActivityLocation: URIRef =
rdflib.term.URIRef('https://schema.org/SportsActivityLocation')

SportsClub: URIRef = rdflib.term.URIRef('https://schema.org/SportsClub')

SportsEvent: URIRef = rdflib.term.URIRef('https://schema.org/SportsEvent')

SportsOrganization: URIRef =
rdflib.term.URIRef('https://schema.org/SportsOrganization')

SportsTeam: URIRef = rdflib.term.URIRef('https://schema.org/SportsTeam')

SpreadsheetDigitalDocument: URIRef =
rdflib.term.URIRef('https://schema.org/SpreadsheetDigitalDocument')

StadiumOrArena: URIRef = rdflib.term.URIRef('https://schema.org/StadiumOrArena')

StagedContent: URIRef = rdflib.term.URIRef('https://schema.org/StagedContent')

StagesHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/StagesHealthAspect')

State: URIRef = rdflib.term.URIRef('https://schema.org/State')
```



```
Statement: URIRef = rdflib.term.URIRef('https://schema.org/Statement')

StatisticalPopulation: URIRef =
rdflib.term.URIRef('https://schema.org/StatisticalPopulation')

StatusEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/StatusEnumeration')

SteeringPositionValue: URIRef =
rdflib.term.URIRef('https://schema.org/SteeringPositionValue')

Store: URIRef = rdflib.term.URIRef('https://schema.org/Store')

StoreCreditRefund: URIRef =
rdflib.term.URIRef('https://schema.org/StoreCreditRefund')

StrengthTraining: URIRef =
rdflib.term.URIRef('https://schema.org/StrengthTraining')

StructuredValue: URIRef = rdflib.term.URIRef('https://schema.org/StructuredValue')

StudioAlbum: URIRef = rdflib.term.URIRef('https://schema.org/StudioAlbum')

SubscribeAction: URIRef = rdflib.term.URIRef('https://schema.org/SubscribeAction')

Subscription: URIRef = rdflib.term.URIRef('https://schema.org/Subscription')

Substance: URIRef = rdflib.term.URIRef('https://schema.org/Substance')

SubwayStation: URIRef = rdflib.term.URIRef('https://schema.org/SubwayStation')

Suite: URIRef = rdflib.term.URIRef('https://schema.org/Suite')

Sunday: URIRef = rdflib.term.URIRef('https://schema.org/Sunday')

SuperficialAnatomy: URIRef =
rdflib.term.URIRef('https://schema.org/SuperficialAnatomy')

Surgical: URIRef = rdflib.term.URIRef('https://schema.org/Surgical')

SurgicalProcedure: URIRef =
rdflib.term.URIRef('https://schema.org/SurgicalProcedure')

SuspendAction: URIRef = rdflib.term.URIRef('https://schema.org/SuspendAction')

Suspended: URIRef = rdflib.term.URIRef('https://schema.org/Suspended')

SymptomsHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/SymptomsHealthAspect')

Synagogue: URIRef = rdflib.term.URIRef('https://schema.org/Synagogue')

TVClip: URIRef = rdflib.term.URIRef('https://schema.org/TVClip')

TVEpisode: URIRef = rdflib.term.URIRef('https://schema.org/TVEpisode')

TVSeason: URIRef = rdflib.term.URIRef('https://schema.org/TVSeason')

TVSeries: URIRef = rdflib.term.URIRef('https://schema.org/TVSeries')
```



```

Table: URIRef = rdflib.term.URIRef('https://schema.org/Table')
TakeAction: URIRef = rdflib.term.URIRef('https://schema.org/TakeAction')
TattooParlor: URIRef = rdflib.term.URIRef('https://schema.org/TattooParlor')
Taxi: URIRef = rdflib.term.URIRef('https://schema.org/Taxi')
TaxiReservation: URIRef = rdflib.term.URIRef('https://schema.org/TaxiReservation')
TaxiService: URIRef = rdflib.term.URIRef('https://schema.org/TaxiService')
TaxiStand: URIRef = rdflib.term.URIRef('https://schema.org/TaxiStand')
TaxiVehicleUsage: URIRef =
rdflib.term.URIRef('https://schema.org/TaxiVehicleUsage')
Taxon: URIRef = rdflib.term.URIRef('https://schema.org/Taxon')
TechArticle: URIRef = rdflib.term.URIRef('https://schema.org/TechArticle')
TelevisionChannel: URIRef =
rdflib.term.URIRef('https://schema.org/TelevisionChannel')
TelevisionStation: URIRef =
rdflib.term.URIRef('https://schema.org/TelevisionStation')
TennisComplex: URIRef = rdflib.term.URIRef('https://schema.org/TennisComplex')
Terminated: URIRef = rdflib.term.URIRef('https://schema.org/Terminated')
Text: URIRef = rdflib.term.URIRef('https://schema.org/Text')
TextDigitalDocument: URIRef =
rdflib.term.URIRef('https://schema.org/TextDigitalDocument')
TheaterEvent: URIRef = rdflib.term.URIRef('https://schema.org/TheaterEvent')
TheaterGroup: URIRef = rdflib.term.URIRef('https://schema.org/TheaterGroup')
Therapeutic: URIRef = rdflib.term.URIRef('https://schema.org/Therapeutic')
TherapeuticProcedure: URIRef =
rdflib.term.URIRef('https://schema.org/TherapeuticProcedure')
Thesis: URIRef = rdflib.term.URIRef('https://schema.org/Thesis')
Thing: URIRef = rdflib.term.URIRef('https://schema.org/Thing')
Throat: URIRef = rdflib.term.URIRef('https://schema.org/Throat')
Thursday: URIRef = rdflib.term.URIRef('https://schema.org/Thursday')
Ticket: URIRef = rdflib.term.URIRef('https://schema.org/Ticket')
TieAction: URIRef = rdflib.term.URIRef('https://schema.org/TieAction')
Time: URIRef = rdflib.term.URIRef('https://schema.org/Time')
TipAction: URIRef = rdflib.term.URIRef('https://schema.org/TipAction')

```

```
TireShop: URIRef = rdflib.term.URIRef('https://schema.org/TireShop')
TollFree: URIRef = rdflib.term.URIRef('https://schema.org/TollFree')
TouristAttraction: URIRef =
rdflib.term.URIRef('https://schema.org/TouristAttraction')
TouristDestination: URIRef =
rdflib.term.URIRef('https://schema.org/TouristDestination')
TouristInformationCenter: URIRef =
rdflib.term.URIRef('https://schema.org/TouristInformationCenter')
TouristTrip: URIRef = rdflib.term.URIRef('https://schema.org/TouristTrip')
Toxicologic: URIRef = rdflib.term.URIRef('https://schema.org/Toxicologic')
ToyStore: URIRef = rdflib.term.URIRef('https://schema.org/ToyStore')
TrackAction: URIRef = rdflib.term.URIRef('https://schema.org/TrackAction')
TradeAction: URIRef = rdflib.term.URIRef('https://schema.org/TradeAction')
TraditionalChinese: URIRef =
rdflib.term.URIRef('https://schema.org/TraditionalChinese')
TrainReservation: URIRef =
rdflib.term.URIRef('https://schema.org/TrainReservation')
TrainStation: URIRef = rdflib.term.URIRef('https://schema.org/TrainStation')
TrainTrip: URIRef = rdflib.term.URIRef('https://schema.org/TrainTrip')
TransferAction: URIRef = rdflib.term.URIRef('https://schema.org/TransferAction')
TransformedContent: URIRef =
rdflib.term.URIRef('https://schema.org/TransformedContent')
TransitMap: URIRef = rdflib.term.URIRef('https://schema.org/TransitMap')
TravelAction: URIRef = rdflib.term.URIRef('https://schema.org/TravelAction')
TravelAgency: URIRef = rdflib.term.URIRef('https://schema.org/TravelAgency')
TreatmentIndication: URIRef =
rdflib.term.URIRef('https://schema.org/TreatmentIndication')
TreatmentsHealthAspect: URIRef =
rdflib.term.URIRef('https://schema.org/TreatmentsHealthAspect')
Trip: URIRef = rdflib.term.URIRef('https://schema.org/Trip')
TripleBlindedTrial: URIRef =
rdflib.term.URIRef('https://schema.org/TripleBlindedTrial')
Tuesday: URIRef = rdflib.term.URIRef('https://schema.org/Tuesday')
TypeAndQuantityNode: URIRef =
rdflib.term.URIRef('https://schema.org/TypeAndQuantityNode')
```

```
TypesHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/TypesHealthAspect')  
  
UKNonprofitType: URIRef = rdflib.term.URIRef('https://schema.org/UKNonprofitType')  
  
UKTrust: URIRef = rdflib.term.URIRef('https://schema.org/UKTrust')  
  
URL: URIRef = rdflib.term.URIRef('https://schema.org/URL')  
  
USNonprofitType: URIRef = rdflib.term.URIRef('https://schema.org/USNonprofitType')  
  
Ultrasound: URIRef = rdflib.term.URIRef('https://schema.org/Ultrasound')  
  
UnRegisterAction: URIRef =  
rdflib.term.URIRef('https://schema.org/UnRegisterAction')  
  
UnemploymentSupport: URIRef =  
rdflib.term.URIRef('https://schema.org/UnemploymentSupport')  
  
UnincorporatedAssociationCharity: URIRef =  
rdflib.term.URIRef('https://schema.org/UnincorporatedAssociationCharity')  
  
UnitPriceSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/UnitPriceSpecification')  
  
UnofficialLegalValue: URIRef =  
rdflib.term.URIRef('https://schema.org/UnofficialLegalValue')  
  
UpdateAction: URIRef = rdflib.term.URIRef('https://schema.org/UpdateAction')  
  
Urologic: URIRef = rdflib.term.URIRef('https://schema.org/Urologic')  
  
UsageOrScheduleHealthAspect: URIRef =  
rdflib.term.URIRef('https://schema.org/UsageOrScheduleHealthAspect')  
  
UseAction: URIRef = rdflib.term.URIRef('https://schema.org/UseAction')  
  
UsedCondition: URIRef = rdflib.term.URIRef('https://schema.org/UsedCondition')  
  
UserBlocks: URIRef = rdflib.term.URIRef('https://schema.org/UserBlocks')  
  
UserCheckins: URIRef = rdflib.term.URIRef('https://schema.org/UserCheckins')  
  
UserComments: URIRef = rdflib.term.URIRef('https://schema.org/UserComments')  
  
UserDownloads: URIRef = rdflib.term.URIRef('https://schema.org/UserDownloads')  
  
UserInteraction: URIRef = rdflib.term.URIRef('https://schema.org/UserInteraction')  
  
UserLikes: URIRef = rdflib.term.URIRef('https://schema.org/UserLikes')  
  
UserPageVisits: URIRef = rdflib.term.URIRef('https://schema.org/UserPageVisits')  
  
UserPlays: URIRef = rdflib.term.URIRef('https://schema.org/UserPlays')  
  
UserPlusOnes: URIRef = rdflib.term.URIRef('https://schema.org/UserPlusOnes')  
  
UserReview: URIRef = rdflib.term.URIRef('https://schema.org/UserReview')
```

```
UserTweets: URIRef = rdflib.term.URIRef('https://schema.org/UserTweets')
VeganDiet: URIRef = rdflib.term.URIRef('https://schema.org/VeganDiet')
VegetarianDiet: URIRef = rdflib.term.URIRef('https://schema.org/VegetarianDiet')
Vehicle: URIRef = rdflib.term.URIRef('https://schema.org/Vehicle')
Vein: URIRef = rdflib.term.URIRef('https://schema.org/Vein')
VenueMap: URIRef = rdflib.term.URIRef('https://schema.org/VenueMap')
Vessel: URIRef = rdflib.term.URIRef('https://schema.org/Vessel')
VeterinaryCare: URIRef = rdflib.term.URIRef('https://schema.org/VeterinaryCare')
VideoGallery: URIRef = rdflib.term.URIRef('https://schema.org/VideoGallery')
VideoGame: URIRef = rdflib.term.URIRef('https://schema.org/VideoGame')
VideoGameClip: URIRef = rdflib.term.URIRef('https://schema.org/VideoGameClip')
VideoGameSeries: URIRef = rdflib.term.URIRef('https://schema.org/VideoGameSeries')
VideoObject: URIRef = rdflib.term.URIRef('https://schema.org/VideoObject')
VideoObjectSnapshot: URIRef =
rdflib.term.URIRef('https://schema.org/VideoObjectSnapshot')
ViewAction: URIRef = rdflib.term.URIRef('https://schema.org/ViewAction')
VinylFormat: URIRef = rdflib.term.URIRef('https://schema.org/VinylFormat')
VirtualLocation: URIRef = rdflib.term.URIRef('https://schema.org/VirtualLocation')
Virus: URIRef = rdflib.term.URIRef('https://schema.org/Virus')
VisualArtsEvent: URIRef = rdflib.term.URIRef('https://schema.org/VisualArtsEvent')
VisualArtwork: URIRef = rdflib.term.URIRef('https://schema.org/VisualArtwork')
VitalSign: URIRef = rdflib.term.URIRef('https://schema.org/VitalSign')
Volcano: URIRef = rdflib.term.URIRef('https://schema.org/Volcano')
VoteAction: URIRef = rdflib.term.URIRef('https://schema.org/VoteAction')
WPAdblock: URIRef = rdflib.term.URIRef('https://schema.org/WPAdblock')
WPFooter: URIRef = rdflib.term.URIRef('https://schema.org/WPFooter')
WPHeader: URIRef = rdflib.term.URIRef('https://schema.org/WPHeader')
WPSideBar: URIRef = rdflib.term.URIRef('https://schema.org/WPSideBar')
WantAction: URIRef = rdflib.term.URIRef('https://schema.org/WantAction')
WarrantyPromise: URIRef = rdflib.term.URIRef('https://schema.org/WarrantyPromise')
WarrantyScope: URIRef = rdflib.term.URIRef('https://schema.org/WarrantyScope')
```

```
WatchAction: URIRef = rdflib.term.URIRef('https://schema.org/WatchAction')

Waterfall: URIRef = rdflib.term.URIRef('https://schema.org/Waterfall')

WearAction: URIRef = rdflib.term.URIRef('https://schema.org/WearAction')

WearableMeasurementBack: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementBack')

WearableMeasurementChestOrBust: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementChestOrBust')

WearableMeasurementCollar: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementCollar')

WearableMeasurementCup: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementCup')

WearableMeasurementHeight: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementHeight')

WearableMeasurementHips: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementHips')

WearableMeasurementInseam: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementInseam')

WearableMeasurementLength: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementLength')

WearableMeasurementOutsideLeg: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementOutsideLeg')

WearableMeasurementSleeve: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementSleeve')

WearableMeasurementTypeEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementTypeEnumeration')

WearableMeasurementWaist: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementWaist')

WearableMeasurementWidth: URIRef =
rdflib.term.URIRef('https://schema.org/WearableMeasurementWidth')

WearableSizeGroupBig: URIRef =
rdflib.term.URIRef('https://schema.org/WearableSizeGroupBig')

WearableSizeGroupBoys: URIRef =
rdflib.term.URIRef('https://schema.org/WearableSizeGroupBoys')

WearableSizeGroupEnumeration: URIRef =
rdflib.term.URIRef('https://schema.org/WearableSizeGroupEnumeration')

WearableSizeGroupExtraShort: URIRef =
rdflib.term.URIRef('https://schema.org/WearableSizeGroupExtraShort')
```

```
WearableSizeGroupExtraTall: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupExtraTall')  
  
WearableSizeGroupGirls: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupGirls')  
  
WearableSizeGroupHusky: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupHusky')  
  
WearableSizeGroupInfants: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupInfants')  
  
WearableSizeGroupJuniors: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupJuniors')  
  
WearableSizeGroupMaternity: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupMaternity')  
  
WearableSizeGroupMens: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupMens')  
  
WearableSizeGroupMisses: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupMisses')  
  
WearableSizeGroupPetite: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupPetite')  
  
WearableSizeGroupPlus: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupPlus')  
  
WearableSizeGroupRegular: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupRegular')  
  
WearableSizeGroupShort: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupShort')  
  
WearableSizeGroupTall: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupTall')  
  
WearableSizeGroupWomens: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeGroupWomens')  
  
WearableSizeSystemAU: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemAU')  
  
WearableSizeSystemBR: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemBR')  
  
WearableSizeSystemCN: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemCN')  
  
WearableSizeSystemContinental: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemContinental')  
  
WearableSizeSystemDE: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemDE')
```

```
WearableSizeSystemEN13402: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemEN13402')  
  
WearableSizeSystemEnumeration: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemEnumeration')  
  
WearableSizeSystemEurope: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemEurope')  
  
WearableSizeSystemFR: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemFR')  
  
WearableSizeSystemGS1: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemGS1')  
  
WearableSizeSystemIT: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemIT')  
  
WearableSizeSystemJP: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemJP')  
  
WearableSizeSystemMX: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemMX')  
  
WearableSizeSystemUK: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemUK')  
  
WearableSizeSystemUS: URIRef =  
rdflib.term.URIRef('https://schema.org/WearableSizeSystemUS')  
  
WebAPI: URIRef = rdflib.term.URIRef('https://schema.org/WebAPI')  
  
WebApplication: URIRef = rdflib.term.URIRef('https://schema.org/WebApplication')  
  
WebContent: URIRef = rdflib.term.URIRef('https://schema.org/WebContent')  
  
WebPage: URIRef = rdflib.term.URIRef('https://schema.org/WebPage')  
  
WebPageElement: URIRef = rdflib.term.URIRef('https://schema.org/WebPageElement')  
  
WebSite: URIRef = rdflib.term.URIRef('https://schema.org/WebSite')  
  
Wednesday: URIRef = rdflib.term.URIRef('https://schema.org/Wednesday')  
  
WesternConventional: URIRef =  
rdflib.term.URIRef('https://schema.org/WesternConventional')  
  
Wholesale: URIRef = rdflib.term.URIRef('https://schema.org/Wholesale')  
  
WholesaleStore: URIRef = rdflib.term.URIRef('https://schema.org/WholesaleStore')  
  
WinAction: URIRef = rdflib.term.URIRef('https://schema.org/WinAction')  
  
Winery: URIRef = rdflib.term.URIRef('https://schema.org/Winery')  
  
Withdrawn: URIRef = rdflib.term.URIRef('https://schema.org/Withdrawn')
```



```
WorkBasedProgram: URIRef =  
rdflib.term.URIRef('https://schema.org/WorkBasedProgram')  
  
WorkersUnion: URIRef = rdflib.term.URIRef('https://schema.org/WorkersUnion')  
  
WriteAction: URIRef = rdflib.term.URIRef('https://schema.org/WriteAction')  
  
WritePermission: URIRef = rdflib.term.URIRef('https://schema.org/WritePermission')  
  
XPathType: URIRef = rdflib.term.URIRef('https://schema.org/XPathType')  
  
XRay: URIRef = rdflib.term.URIRef('https://schema.org/XRay')  
  
ZoneBoardingPolicy: URIRef =  
rdflib.term.URIRef('https://schema.org/ZoneBoardingPolicy')  
  
Zoo: URIRef = rdflib.term.URIRef('https://schema.org/Zoo')  
  
about: URIRef = rdflib.term.URIRef('https://schema.org/about')  
  
abridged: URIRef = rdflib.term.URIRef('https://schema.org/abridged')  
  
abstract: URIRef = rdflib.term.URIRef('https://schema.org/abstract')  
  
accelerationTime: URIRef =  
rdflib.term.URIRef('https://schema.org/accelerationTime')  
  
acceptedAnswer: URIRef = rdflib.term.URIRef('https://schema.org/acceptedAnswer')  
  
acceptedOffer: URIRef = rdflib.term.URIRef('https://schema.org/acceptedOffer')  
  
acceptedPaymentMethod: URIRef =  
rdflib.term.URIRef('https://schema.org/acceptedPaymentMethod')  
  
acceptsReservations: URIRef =  
rdflib.term.URIRef('https://schema.org/acceptsReservations')  
  
accessCode: URIRef = rdflib.term.URIRef('https://schema.org/accessCode')  
  
accessMode: URIRef = rdflib.term.URIRef('https://schema.org/accessMode')  
  
accessModeSufficient: URIRef =  
rdflib.term.URIRef('https://schema.org/accessModeSufficient')  
  
accessibilityAPI: URIRef = rdflib.term.URIRef('https://schema.org/accessibilityAPI')  
  
accessibilityControl: URIRef =  
rdflib.term.URIRef('https://schema.org/accessibilityControl')  
  
accessibilityFeature: URIRef =  
rdflib.term.URIRef('https://schema.org/accessibilityFeature')  
  
accessibilityHazard: URIRef =  
rdflib.term.URIRef('https://schema.org/accessibilityHazard')  
  
accessibilitySummary: URIRef =  
rdflib.term.URIRef('https://schema.org/accessibilitySummary')
```



```
accommodationCategory: URIRef =  
rdflib.term.URIRef('https://schema.org/accommodationCategory')  
  
accommodationFloorPlan: URIRef =  
rdflib.term.URIRef('https://schema.org/accommodationFloorPlan')  
  
accountId: URIRef = rdflib.term.URIRef('https://schema.org/accountId')  
  
accountMinimumInflow: URIRef =  
rdflib.term.URIRef('https://schema.org/accountMinimumInflow')  
  
accountOverdraftLimit: URIRef =  
rdflib.term.URIRef('https://schema.org/accountOverdraftLimit')  
  
accountablePerson: URIRef =  
rdflib.term.URIRef('https://schema.org/accountablePerson')  
  
acquireLicensePage: URIRef =  
rdflib.term.URIRef('https://schema.org/acquireLicensePage')  
  
acquiredFrom: URIRef = rdflib.term.URIRef('https://schema.org/acquiredFrom')  
  
acrissCode: URIRef = rdflib.term.URIRef('https://schema.org/acrissCode')  
  
actionAccessibilityRequirement: URIRef =  
rdflib.term.URIRef('https://schema.org/actionAccessibilityRequirement')  
  
actionApplication: URIRef =  
rdflib.term.URIRef('https://schema.org/actionApplication')  
  
actionOption: URIRef = rdflib.term.URIRef('https://schema.org/actionOption')  
  
actionPlatform: URIRef = rdflib.term.URIRef('https://schema.org/actionPlatform')  
  
actionStatus: URIRef = rdflib.term.URIRef('https://schema.org/actionStatus')  
  
actionableFeedbackPolicy: URIRef =  
rdflib.term.URIRef('https://schema.org/actionableFeedbackPolicy')  
  
activeIngredient: URIRef =  
rdflib.term.URIRef('https://schema.org/activeIngredient')  
  
activityDuration: URIRef =  
rdflib.term.URIRef('https://schema.org/activityDuration')  
  
activityFrequency: URIRef =  
rdflib.term.URIRef('https://schema.org/activityFrequency')  
  
actor: URIRef = rdflib.term.URIRef('https://schema.org/actor')  
  
actors: URIRef = rdflib.term.URIRef('https://schema.org/actors')  
  
addOn: URIRef = rdflib.term.URIRef('https://schema.org/addOn')  
  
additionalName: URIRef = rdflib.term.URIRef('https://schema.org/additionalName')  
  
additionalNumberOfGuests: URIRef =  
rdflib.term.URIRef('https://schema.org/additionalNumberOfGuests')
```

```
additionalProperty: URIRef =  
rdflib.term.URIRef('https://schema.org/additionalProperty')  
  
additionalType: URIRef = rdflib.term.URIRef('https://schema.org/additionalType')  
  
additionalVariable: URIRef =  
rdflib.term.URIRef('https://schema.org/additionalVariable')  
  
address: URIRef = rdflib.term.URIRef('https://schema.org/address')  
  
addressCountry: URIRef = rdflib.term.URIRef('https://schema.org/addressCountry')  
  
addressLocality: URIRef = rdflib.term.URIRef('https://schema.org/addressLocality')  
  
addressRegion: URIRef = rdflib.term.URIRef('https://schema.org/addressRegion')  
  
administrationRoute: URIRef =  
rdflib.term.URIRef('https://schema.org/administrationRoute')  
  
advanceBookingRequirement: URIRef =  
rdflib.term.URIRef('https://schema.org/advanceBookingRequirement')  
  
adverseOutcome: URIRef = rdflib.term.URIRef('https://schema.org/adverseOutcome')  
  
affectedBy: URIRef = rdflib.term.URIRef('https://schema.org/affectedBy')  
  
affiliation: URIRef = rdflib.term.URIRef('https://schema.org/affiliation')  
  
afterMedia: URIRef = rdflib.term.URIRef('https://schema.org/afterMedia')  
  
agent: URIRef = rdflib.term.URIRef('https://schema.org/agent')  
  
aggregateRating: URIRef = rdflib.term.URIRef('https://schema.org/aggregateRating')  
  
aircraft: URIRef = rdflib.term.URIRef('https://schema.org/aircraft')  
  
album: URIRef = rdflib.term.URIRef('https://schema.org/album')  
  
albumProductionType: URIRef =  
rdflib.term.URIRef('https://schema.org/albumProductionType')  
  
albumRelease: URIRef = rdflib.term.URIRef('https://schema.org/albumRelease')  
  
albumReleaseType: URIRef =  
rdflib.term.URIRef('https://schema.org/albumReleaseType')  
  
albums: URIRef = rdflib.term.URIRef('https://schema.org/albums')  
  
alcoholWarning: URIRef = rdflib.term.URIRef('https://schema.org/alcoholWarning')  
  
algorithm: URIRef = rdflib.term.URIRef('https://schema.org/algorithm')  
  
alignmentType: URIRef = rdflib.term.URIRef('https://schema.org/alignmentType')  
  
alternateName: URIRef = rdflib.term.URIRef('https://schema.org/alternateName')  
  
alternativeHeadline: URIRef =  
rdflib.term.URIRef('https://schema.org/alternativeHeadline')
```

```
alternativeOf: URIRef = rdflib.term.URIRef('https://schema.org/alternativeOf')
alumni: URIRef = rdflib.term.URIRef('https://schema.org/alumni')
alumniOf: URIRef = rdflib.term.URIRef('https://schema.org/alumniOf')
amenityFeature: URIRef = rdflib.term.URIRef('https://schema.org/amenityFeature')
amount: URIRef = rdflib.term.URIRef('https://schema.org/amount')
amountOfThisGood: URIRef =
rdflib.term.URIRef('https://schema.org/amountOfThisGood')
announcementLocation: URIRef =
rdflib.term.URIRef('https://schema.org/announcementLocation')
annualPercentageRate: URIRef =
rdflib.term.URIRef('https://schema.org/annualPercentageRate')
answerCount: URIRef = rdflib.term.URIRef('https://schema.org/answerCount')
answerExplanation: URIRef =
rdflib.term.URIRef('https://schema.org/answerExplanation')
antagonist: URIRef = rdflib.term.URIRef('https://schema.org/antagonist')
appearance: URIRef = rdflib.term.URIRef('https://schema.org/appearance')
applicableLocation: URIRef =
rdflib.term.URIRef('https://schema.org/applicableLocation')
applicantLocationRequirements: URIRef =
rdflib.term.URIRef('https://schema.org/applicantLocationRequirements')
application: URIRef = rdflib.term.URIRef('https://schema.org/application')
applicationCategory: URIRef =
rdflib.term.URIRef('https://schema.org/applicationCategory')
applicationContact: URIRef =
rdflib.term.URIRef('https://schema.org/applicationContact')
applicationDeadline: URIRef =
rdflib.term.URIRef('https://schema.org/applicationDeadline')
applicationStartDate: URIRef =
rdflib.term.URIRef('https://schema.org/applicationStartDate')
applicationSubCategory: URIRef =
rdflib.term.URIRef('https://schema.org/applicationSubCategory')
applicationSuite: URIRef =
rdflib.term.URIRef('https://schema.org/applicationSuite')
appliesToDeliveryMethod: URIRef =
rdflib.term.URIRef('https://schema.org/appliesToDeliveryMethod')
```

```
appliesToPaymentMethod: URIRef =  
rdflib.term.URIRef('https://schema.org/appliesToPaymentMethod')  
  
archiveHeld: URIRef = rdflib.term.URIRef('https://schema.org/archiveHeld')  
  
archivedAt: URIRef = rdflib.term.URIRef('https://schema.org/archivedAt')  
  
area: URIRef = rdflib.term.URIRef('https://schema.org/area')  
  
areaServed: URIRef = rdflib.term.URIRef('https://schema.org/areaServed')  
  
arrivalAirport: URIRef = rdflib.term.URIRef('https://schema.org/arrivalAirport')  
  
arrivalBoatTerminal: URIRef =  
rdflib.term.URIRef('https://schema.org/arrivalBoatTerminal')  
  
arrivalBusStop: URIRef = rdflib.term.URIRef('https://schema.org/arrivalBusStop')  
  
arrivalGate: URIRef = rdflib.term.URIRef('https://schema.org/arrivalGate')  
  
arrivalPlatform: URIRef = rdflib.term.URIRef('https://schema.org/arrivalPlatform')  
  
arrivalStation: URIRef = rdflib.term.URIRef('https://schema.org/arrivalStation')  
  
arrivalTerminal: URIRef = rdflib.term.URIRef('https://schema.org/arrivalTerminal')  
  
arrivalTime: URIRef = rdflib.term.URIRef('https://schema.org/arrivalTime')  
  
artEdition: URIRef = rdflib.term.URIRef('https://schema.org/artEdition')  
  
artMedium: URIRef = rdflib.term.URIRef('https://schema.org/artMedium')  
  
arterialBranch: URIRef = rdflib.term.URIRef('https://schema.org/arterialBranch')  
  
artform: URIRef = rdflib.term.URIRef('https://schema.org/artform')  
  
articleBody: URIRef = rdflib.term.URIRef('https://schema.org/articleBody')  
  
articleSection: URIRef = rdflib.term.URIRef('https://schema.org/articleSection')  
  
artist: URIRef = rdflib.term.URIRef('https://schema.org/artist')  
  
artworkSurface: URIRef = rdflib.term.URIRef('https://schema.org/artworkSurface')  
  
aspect: URIRef = rdflib.term.URIRef('https://schema.org/aspect')  
  
assembly: URIRef = rdflib.term.URIRef('https://schema.org/assembly')  
  
assemblyVersion: URIRef = rdflib.term.URIRef('https://schema.org/assemblyVersion')  
  
assesses: URIRef = rdflib.term.URIRef('https://schema.org/assesses')  
  
associatedAnatomy: URIRef =  
rdflib.term.URIRef('https://schema.org/associatedAnatomy')  
  
associatedArticle: URIRef =  
rdflib.term.URIRef('https://schema.org/associatedArticle')  
  
associatedClaimReview: URIRef =  
rdflib.term.URIRef('https://schema.org/associatedClaimReview')
```

```
associatedDisease: URIRef =  
rdflib.term.URIRef('https://schema.org/associatedDisease')  
  
associatedMedia: URIRef = rdflib.term.URIRef('https://schema.org/associatedMedia')  
  
associatedMediaReview: URIRef =  
rdflib.term.URIRef('https://schema.org/associatedMediaReview')  
  
associatedPathophysiology: URIRef =  
rdflib.term.URIRef('https://schema.org/associatedPathophysiology')  
  
associatedReview: URIRef =  
rdflib.term.URIRef('https://schema.org/associatedReview')  
  
athlete: URIRef = rdflib.term.URIRef('https://schema.org/athlete')  
  
attendee: URIRef = rdflib.term.URIRef('https://schema.org/attendee')  
  
attendees: URIRef = rdflib.term.URIRef('https://schema.org/attendees')  
  
audience: URIRef = rdflib.term.URIRef('https://schema.org/audience')  
  
audienceType: URIRef = rdflib.term.URIRef('https://schema.org/audienceType')  
  
audio: URIRef = rdflib.term.URIRef('https://schema.org/audio')  
  
authenticator: URIRef = rdflib.term.URIRef('https://schema.org/authenticator')  
  
author: URIRef = rdflib.term.URIRef('https://schema.org/author')  
  
availability: URIRef = rdflib.term.URIRef('https://schema.org/availability')  
  
availabilityEnds: URIRef =  
rdflib.term.URIRef('https://schema.org/availabilityEnds')  
  
availabilityStarts: URIRef =  
rdflib.term.URIRef('https://schema.org/availabilityStarts')  
  
availableAtOrFrom: URIRef =  
rdflib.term.URIRef('https://schema.org/availableAtOrFrom')  
  
availableChannel: URIRef =  
rdflib.term.URIRef('https://schema.org/availableChannel')  
  
availableDeliveryMethod: URIRef =  
rdflib.term.URIRef('https://schema.org/availableDeliveryMethod')  
  
availableFrom: URIRef = rdflib.term.URIRef('https://schema.org/availableFrom')  
  
availableIn: URIRef = rdflib.term.URIRef('https://schema.org/availableIn')  
  
availableLanguage: URIRef =  
rdflib.term.URIRef('https://schema.org/availableLanguage')  
  
availableOnDevice: URIRef =  
rdflib.term.URIRef('https://schema.org/availableOnDevice')
```

```
availableService: URIRef =  
rdflib.term.URIRef('https://schema.org/availableService')  
  
availableStrength: URIRef =  
rdflib.term.URIRef('https://schema.org/availableStrength')  
  
availableTest: URIRef = rdflib.term.URIRef('https://schema.org/availableTest')  
  
availableThrough: URIRef =  
rdflib.term.URIRef('https://schema.org/availableThrough')  
  
award: URIRef = rdflib.term.URIRef('https://schema.org/award')  
  
awards: URIRef = rdflib.term.URIRef('https://schema.org/awards')  
  
awayTeam: URIRef = rdflib.term.URIRef('https://schema.org/awayTeam')  
  
backstory: URIRef = rdflib.term.URIRef('https://schema.org/backstory')  
  
bankAccountType: URIRef = rdflib.term.URIRef('https://schema.org/bankAccountType')  
  
baseSalary: URIRef = rdflib.term.URIRef('https://schema.org/baseSalary')  
  
bccRecipient: URIRef = rdflib.term.URIRef('https://schema.org/bccRecipient')  
  
bed: URIRef = rdflib.term.URIRef('https://schema.org/bed')  
  
beforeMedia: URIRef = rdflib.term.URIRef('https://schema.org/beforeMedia')  
  
beneficiaryBank: URIRef = rdflib.term.URIRef('https://schema.org/beneficiaryBank')  
  
benefits: URIRef = rdflib.term.URIRef('https://schema.org/benefits')  
  
benefitsSummaryUrl: URIRef =  
rdflib.term.URIRef('https://schema.org/benefitsSummaryUrl')  
  
bestRating: URIRef = rdflib.term.URIRef('https://schema.org/bestRating')  
  
billingAddress: URIRef = rdflib.term.URIRef('https://schema.org/billingAddress')  
  
billingDuration: URIRef = rdflib.term.URIRef('https://schema.org/billingDuration')  
  
billingIncrement: URIRef =  
rdflib.term.URIRef('https://schema.org/billingIncrement')  
  
billingPeriod: URIRef = rdflib.term.URIRef('https://schema.org/billingPeriod')  
  
billingStart: URIRef = rdflib.term.URIRef('https://schema.org/billingStart')  
  
bioChemInteraction: URIRef =  
rdflib.term.URIRef('https://schema.org/bioChemInteraction')  
  
bioChemSimilarity: URIRef =  
rdflib.term.URIRef('https://schema.org/bioChemSimilarity')  
  
biologicalRole: URIRef = rdflib.term.URIRef('https://schema.org/biologicalRole')  
  
biomechanicalClass: URIRef =  
rdflib.term.URIRef('https://schema.org/biomechanicalClass')
```

```
birthDate: URIRef = rdflib.term.URIRef('https://schema.org/birthDate')
birthPlace: URIRef = rdflib.term.URIRef('https://schema.org/birthPlace')
bitrate: URIRef = rdflib.term.URIRef('https://schema.org/bitrate')
blogPost: URIRef = rdflib.term.URIRef('https://schema.org/blogPost')
blogPosts: URIRef = rdflib.term.URIRef('https://schema.org/blogPosts')
bloodSupply: URIRef = rdflib.term.URIRef('https://schema.org/bloodSupply')
boardingGroup: URIRef = rdflib.term.URIRef('https://schema.org/boardingGroup')
boardingPolicy: URIRef = rdflib.term.URIRef('https://schema.org/boardingPolicy')
bodyLocation: URIRef = rdflib.term.URIRef('https://schema.org/bodyLocation')
bodyType: URIRef = rdflib.term.URIRef('https://schema.org/bodyType')
bookEdition: URIRef = rdflib.term.URIRef('https://schema.org/bookEdition')
bookFormat: URIRef = rdflib.term.URIRef('https://schema.org/bookFormat')
bookingAgent: URIRef = rdflib.term.URIRef('https://schema.org/bookingAgent')
bookingTime: URIRef = rdflib.term.URIRef('https://schema.org/bookingTime')
borrower: URIRef = rdflib.term.URIRef('https://schema.org/borrower')
box: URIRef = rdflib.term.URIRef('https://schema.org/box')
branch: URIRef = rdflib.term.URIRef('https://schema.org/branch')
branchCode: URIRef = rdflib.term.URIRef('https://schema.org/branchCode')
branchOf: URIRef = rdflib.term.URIRef('https://schema.org/branchOf')
brand: URIRef = rdflib.term.URIRef('https://schema.org/brand')
breadcrumb: URIRef = rdflib.term.URIRef('https://schema.org/breadcrumb')
breastfeedingWarning: URIRef =
rdflib.term.URIRef('https://schema.org/breastfeedingWarning')
broadcastAffiliateOf: URIRef =
rdflib.term.URIRef('https://schema.org/broadcastAffiliateOf')
broadcastChannelId: URIRef =
rdflib.term.URIRef('https://schema.org/broadcastChannelId')
broadcastDisplayName: URIRef =
rdflib.term.URIRef('https://schema.org/broadcastDisplayName')
broadcastFrequency: URIRef =
rdflib.term.URIRef('https://schema.org/broadcastFrequency')
broadcastFrequencyValue: URIRef =
rdflib.term.URIRef('https://schema.org/broadcastFrequencyValue')
```

```
broadcastOfEvent: URIRef =  
rdflib.term.URIRef('https://schema.org/broadcastOfEvent')  
  
broadcastServiceTier: URIRef =  
rdflib.term.URIRef('https://schema.org/broadcastServiceTier')  
  
broadcastSignalModulation: URIRef =  
rdflib.term.URIRef('https://schema.org/broadcastSignalModulation')  
  
broadcastSubChannel: URIRef =  
rdflib.term.URIRef('https://schema.org/broadcastSubChannel')  
  
broadcastTimezone: URIRef =  
rdflib.term.URIRef('https://schema.org/broadcastTimezone')  
  
broadcaster: URIRef = rdflib.term.URIRef('https://schema.org/broadcaster')  
  
broker: URIRef = rdflib.term.URIRef('https://schema.org/broker')  
  
browserRequirements: URIRef =  
rdflib.term.URIRef('https://schema.org/browserRequirements')  
  
busName: URIRef = rdflib.term.URIRef('https://schema.org/busName')  
  
busNumber: URIRef = rdflib.term.URIRef('https://schema.org/busNumber')  
  
businessDays: URIRef = rdflib.term.URIRef('https://schema.org/businessDays')  
  
businessFunction: URIRef =  
rdflib.term.URIRef('https://schema.org/businessFunction')  
  
buyer: URIRef = rdflib.term.URIRef('https://schema.org/buyer')  
  
byArtist: URIRef = rdflib.term.URIRef('https://schema.org/byArtist')  
  
byDay: URIRef = rdflib.term.URIRef('https://schema.org/byDay')  
  
byMonth: URIRef = rdflib.term.URIRef('https://schema.org/byMonth')  
  
byMonthDay: URIRef = rdflib.term.URIRef('https://schema.org/byMonthDay')  
  
byMonthWeek: URIRef = rdflib.term.URIRef('https://schema.org/byMonthWeek')  
  
callSign: URIRef = rdflib.term.URIRef('https://schema.org/callSign')  
  
calories: URIRef = rdflib.term.URIRef('https://schema.org/calories')  
  
candidate: URIRef = rdflib.term.URIRef('https://schema.org/candidate')  
  
caption: URIRef = rdflib.term.URIRef('https://schema.org/caption')  
  
carbohydrateContent: URIRef =  
rdflib.term.URIRef('https://schema.org/carbohydrateContent')  
  
cargoVolume: URIRef = rdflib.term.URIRef('https://schema.org/cargoVolume')  
  
carrier: URIRef = rdflib.term.URIRef('https://schema.org/carrier')
```



```
carrierRequirements: URIRef =  
rdflib.term.URIRef('https://schema.org/carrierRequirements')  
  
cashBack: URIRef = rdflib.term.URIRef('https://schema.org/cashBack')  
  
catalog: URIRef = rdflib.term.URIRef('https://schema.org/catalog')  
  
catalogNumber: URIRef = rdflib.term.URIRef('https://schema.org/catalogNumber')  
  
category: URIRef = rdflib.term.URIRef('https://schema.org/category')  
  
causeOf: URIRef = rdflib.term.URIRef('https://schema.org/causeOf')  
  
ccRecipient: URIRef = rdflib.term.URIRef('https://schema.org/ccRecipient')  
  
character: URIRef = rdflib.term.URIRef('https://schema.org/character')  
  
characterAttribute: URIRef =  
rdflib.term.URIRef('https://schema.org/characterAttribute')  
  
characterName: URIRef = rdflib.term.URIRef('https://schema.org/characterName')  
  
cheatCode: URIRef = rdflib.term.URIRef('https://schema.org/cheatCode')  
  
checkinTime: URIRef = rdflib.term.URIRef('https://schema.org/checkinTime')  
  
checkoutTime: URIRef = rdflib.term.URIRef('https://schema.org/checkoutTime')  
  
chemicalComposition: URIRef =  
rdflib.term.URIRef('https://schema.org/chemicalComposition')  
  
chemicalRole: URIRef = rdflib.term.URIRef('https://schema.org/chemicalRole')  
  
childMaxAge: URIRef = rdflib.term.URIRef('https://schema.org/childMaxAge')  
  
childMinAge: URIRef = rdflib.term.URIRef('https://schema.org/childMinAge')  
  
childTaxon: URIRef = rdflib.term.URIRef('https://schema.org/childTaxon')  
  
children: URIRef = rdflib.term.URIRef('https://schema.org/children')  
  
cholesterolContent: URIRef =  
rdflib.term.URIRef('https://schema.org/cholesterolContent')  
  
circle: URIRef = rdflib.term.URIRef('https://schema.org/circle')  
  
citation: URIRef = rdflib.term.URIRef('https://schema.org/citation')  
  
claimInterpreter: URIRef =  
rdflib.term.URIRef('https://schema.org/claimInterpreter')  
  
claimReviewed: URIRef = rdflib.term.URIRef('https://schema.org/claimReviewed')  
  
clinicalPharmacology: URIRef =  
rdflib.term.URIRef('https://schema.org/clinicalPharmacology')  
  
clinicalPharmacology: URIRef =  
rdflib.term.URIRef('https://schema.org/clinicalPharmacology')
```

```
clipNumber: URIRef = rdflib.term.URIRef('https://schema.org/clipNumber')
closes: URIRef = rdflib.term.URIRef('https://schema.org/closes')
coach: URIRef = rdflib.term.URIRef('https://schema.org/coach')
code: URIRef = rdflib.term.URIRef('https://schema.org/code')
codeRepository: URIRef = rdflib.term.URIRef('https://schema.org/codeRepository')
codeSampleType: URIRef = rdflib.term.URIRef('https://schema.org/codeSampleType')
codeValue: URIRef = rdflib.term.URIRef('https://schema.org/codeValue')
codingSystem: URIRef = rdflib.term.URIRef('https://schema.org/codingSystem')
colleague: URIRef = rdflib.term.URIRef('https://schema.org/colleague')
colleagues: URIRef = rdflib.term.URIRef('https://schema.org/colleagues')
collection: URIRef = rdflib.term.URIRef('https://schema.org/collection')
collectionSize: URIRef = rdflib.term.URIRef('https://schema.org/collectionSize')
color: URIRef = rdflib.term.URIRef('https://schema.org/color')
colorist: URIRef = rdflib.term.URIRef('https://schema.org/colorist')
comment: URIRef = rdflib.term.URIRef('https://schema.org/comment')
commentCount: URIRef = rdflib.term.URIRef('https://schema.org/commentCount')
commentText: URIRef = rdflib.term.URIRef('https://schema.org/commentText')
commentTime: URIRef = rdflib.term.URIRef('https://schema.org/commentTime')
competencyRequired: URIRef =
rdflib.term.URIRef('https://schema.org/competencyRequired')
competitor: URIRef = rdflib.term.URIRef('https://schema.org/competitor')
composer: URIRef = rdflib.term.URIRef('https://schema.org/composer')
comprisedOf: URIRef = rdflib.term.URIRef('https://schema.org/comprisedOf')
conditionsOfAccess: URIRef =
rdflib.term.URIRef('https://schema.org/conditionsOfAccess')
confirmationNumber: URIRef =
rdflib.term.URIRef('https://schema.org/confirmationNumber')
connectedTo: URIRef = rdflib.term.URIRef('https://schema.org/connectedTo')
constrainingProperty: URIRef =
rdflib.term.URIRef('https://schema.org/constrainingProperty')
contactOption: URIRef = rdflib.term.URIRef('https://schema.org/contactOption')
contactPoint: URIRef = rdflib.term.URIRef('https://schema.org/contactPoint')
```

```
contactPoints: URIRef = rdflib.term.URIRef('https://schema.org/contactPoints')

contactType: URIRef = rdflib.term.URIRef('https://schema.org/contactType')

contactlessPayment: URIRef =
rdflib.term.URIRef('https://schema.org/contactlessPayment')

containedIn: URIRef = rdflib.term.URIRef('https://schema.org/containedIn')

containedInPlace: URIRef =
rdflib.term.URIRef('https://schema.org/containedInPlace')

containsPlace: URIRef = rdflib.term.URIRef('https://schema.org/containsPlace')

containsSeason: URIRef = rdflib.term.URIRef('https://schema.org/containsSeason')

contentLocation: URIRef = rdflib.term.URIRef('https://schema.org/contentLocation')

contentRating: URIRef = rdflib.term.URIRef('https://schema.org/contentRating')

contentReferenceTime: URIRef =
rdflib.term.URIRef('https://schema.org/contentReferenceTime')

contentSize: URIRef = rdflib.term.URIRef('https://schema.org/contentSize')

contentType: URIRef = rdflib.term.URIRef('https://schema.org/contentType')

contentUrl: URIRef = rdflib.term.URIRef('https://schema.org/contentUrl')

contraindication: URIRef =
rdflib.term.URIRef('https://schema.org/contraindication')

contributor: URIRef = rdflib.term.URIRef('https://schema.org/contributor')

cookTime: URIRef = rdflib.term.URIRef('https://schema.org/cookTime')

cookingMethod: URIRef = rdflib.term.URIRef('https://schema.org/cookingMethod')

copyrightHolder: URIRef = rdflib.term.URIRef('https://schema.org/copyrightHolder')

copyrightNotice: URIRef = rdflib.term.URIRef('https://schema.org/copyrightNotice')

copyrightYear: URIRef = rdflib.term.URIRef('https://schema.org/copyrightYear')

correction: URIRef = rdflib.term.URIRef('https://schema.org/correction')

correctionsPolicy: URIRef =
rdflib.term.URIRef('https://schema.org/correctionsPolicy')

costCategory: URIRef = rdflib.term.URIRef('https://schema.org/costCategory')

costCurrency: URIRef = rdflib.term.URIRef('https://schema.org/costCurrency')

costOrigin: URIRef = rdflib.term.URIRef('https://schema.org/costOrigin')

costPerUnit: URIRef = rdflib.term.URIRef('https://schema.org/costPerUnit')

countriesNotSupported: URIRef =
rdflib.term.URIRef('https://schema.org/countriesNotSupported')
```

```
countriesSupported: URIRef =  
rdflib.term.URIRef('https://schema.org/countriesSupported')  
  
countryOfAssembly: URIRef =  
rdflib.term.URIRef('https://schema.org/countryOfAssembly')  
  
countryOfLastProcessing: URIRef =  
rdflib.term.URIRef('https://schema.org/countryOfLastProcessing')  
  
countryOfOrigin: URIRef = rdflib.term.URIRef('https://schema.org/countryOfOrigin')  
  
course: URIRef = rdflib.term.URIRef('https://schema.org/course')  
  
courseCode: URIRef = rdflib.term.URIRef('https://schema.org/courseCode')  
  
courseMode: URIRef = rdflib.term.URIRef('https://schema.org/courseMode')  
  
coursePrerequisites: URIRef =  
rdflib.term.URIRef('https://schema.org/coursePrerequisites')  
  
courseWorkload: URIRef = rdflib.term.URIRef('https://schema.org/courseWorkload')  
  
coverageEndTime: URIRef = rdflib.term.URIRef('https://schema.org/coverageEndTime')  
  
coverageStartTime: URIRef =  
rdflib.term.URIRef('https://schema.org/coverageStartTime')  
  
creativeWorkStatus: URIRef =  
rdflib.term.URIRef('https://schema.org/creativeWorkStatus')  
  
creator: URIRef = rdflib.term.URIRef('https://schema.org/creator')  
  
credentialCategory: URIRef =  
rdflib.term.URIRef('https://schema.org/credentialCategory')  
  
creditText: URIRef = rdflib.term.URIRef('https://schema.org/creditText')  
  
creditedTo: URIRef = rdflib.term.URIRef('https://schema.org/creditedTo')  
  
cssSelector: URIRef = rdflib.term.URIRef('https://schema.org/cssSelector')  
  
currenciesAccepted: URIRef =  
rdflib.term.URIRef('https://schema.org/currenciesAccepted')  
  
currency: URIRef = rdflib.term.URIRef('https://schema.org/currency')  
  
currentExchangeRate: URIRef =  
rdflib.term.URIRef('https://schema.org/currentExchangeRate')  
  
customer: URIRef = rdflib.term.URIRef('https://schema.org/customer')  
  
customerRemorseReturnFees: URIRef =  
rdflib.term.URIRef('https://schema.org/customerRemorseReturnFees')  
  
customerRemorseReturnLabelSource: URIRef =  
rdflib.term.URIRef('https://schema.org/customerRemorseReturnLabelSource')
```

```

customerRemorseReturnShippingFeesAmount: URIRef =
rdflib.term.URIRef('https://schema.org/customerRemorseReturnShippingFeesAmount')

cutoffTime: URIRef = rdflib.term.URIRef('https://schema.org/cutoffTime')

cvdCollectionDate: URIRef =
rdflib.term.URIRef('https://schema.org/cvdCollectionDate')

cvdFacilityCounty: URIRef =
rdflib.term.URIRef('https://schema.org/cvdFacilityCounty')

cvdFacilityId: URIRef = rdflib.term.URIRef('https://schema.org/cvdFacilityId')

cvdNumBeds: URIRef = rdflib.term.URIRef('https://schema.org/cvdNumBeds')

cvdNumBedsOcc: URIRef = rdflib.term.URIRef('https://schema.org/cvdNumBedsOcc')

cvdNumC19Died: URIRef = rdflib.term.URIRef('https://schema.org/cvdNumC19Died')

cvdNumC19HOPats: URIRef = rdflib.term.URIRef('https://schema.org/cvdNumC19HOPats')

cvdNumC19HospPats: URIRef =
rdflib.term.URIRef('https://schema.org/cvdNumC19HospPats')

cvdNumC19MechVentPats: URIRef =
rdflib.term.URIRef('https://schema.org/cvdNumC19MechVentPats')

cvdNumC19OFMechVentPats: URIRef =
rdflib.term.URIRef('https://schema.org/cvdNumC19OFMechVentPats')

cvdNumC19OverflowPats: URIRef =
rdflib.term.URIRef('https://schema.org/cvdNumC19OverflowPats')

cvdNumICUBeds: URIRef = rdflib.term.URIRef('https://schema.org/cvdNumICUBeds')

cvdNumICUBedsOcc: URIRef =
rdflib.term.URIRef('https://schema.org/cvdNumICUBedsOcc')

cvdNumTotBeds: URIRef = rdflib.term.URIRef('https://schema.org/cvdNumTotBeds')

cvdNumVent: URIRef = rdflib.term.URIRef('https://schema.org/cvdNumVent')

cvdNumVentUse: URIRef = rdflib.term.URIRef('https://schema.org/cvdNumVentUse')

dataFeedElement: URIRef = rdflib.term.URIRef('https://schema.org/dataFeedElement')

dataset: URIRef = rdflib.term.URIRef('https://schema.org/dataset')

datasetTimeInterval: URIRef =
rdflib.term.URIRef('https://schema.org/datasetTimeInterval')

dateCreated: URIRef = rdflib.term.URIRef('https://schema.org/dateCreated')

dateDeleted: URIRef = rdflib.term.URIRef('https://schema.org/dateDeleted')

dateIssued: URIRef = rdflib.term.URIRef('https://schema.org/dateIssued')

dateModified: URIRef = rdflib.term.URIRef('https://schema.org/dateModified')

```

```
datePosted: URIRef = rdflib.term.URIRef('https://schema.org/datePosted')
datePublished: URIRef = rdflib.term.URIRef('https://schema.org/datePublished')
dateRead: URIRef = rdflib.term.URIRef('https://schema.org/dateRead')
dateReceived: URIRef = rdflib.term.URIRef('https://schema.org/dateReceived')
dateSent: URIRef = rdflib.term.URIRef('https://schema.org/dateSent')
dateVehicleFirstRegistered: URIRef =
rdflib.term.URIRef('https://schema.org/dateVehicleFirstRegistered')
dateline: URIRef = rdflib.term.URIRef('https://schema.org/dateline')
dayOfWeek: URIRef = rdflib.term.URIRef('https://schema.org/dayOfWeek')
deathDate: URIRef = rdflib.term.URIRef('https://schema.org/deathDate')
deathPlace: URIRef = rdflib.term.URIRef('https://schema.org/deathPlace')
defaultValue: URIRef = rdflib.term.URIRef('https://schema.org/defaultValue')
deliveryAddress: URIRef = rdflib.term.URIRef('https://schema.org/deliveryAddress')
deliveryLeadTime: URIRef =
rdflib.term.URIRef('https://schema.org/deliveryLeadTime')
deliveryMethod: URIRef = rdflib.term.URIRef('https://schema.org/deliveryMethod')
deliveryStatus: URIRef = rdflib.term.URIRef('https://schema.org/deliveryStatus')
deliveryTime: URIRef = rdflib.term.URIRef('https://schema.org/deliveryTime')
department: URIRef = rdflib.term.URIRef('https://schema.org/department')
departureAirport: URIRef =
rdflib.term.URIRef('https://schema.org/departureAirport')
departureBoatTerminal: URIRef =
rdflib.term.URIRef('https://schema.org/departureBoatTerminal')
departureBusStop: URIRef =
rdflib.term.URIRef('https://schema.org/departureBusStop')
departureGate: URIRef = rdflib.term.URIRef('https://schema.org/departureGate')
departurePlatform: URIRef =
rdflib.term.URIRef('https://schema.org/departurePlatform')
departureStation: URIRef =
rdflib.term.URIRef('https://schema.org/departureStation')
departureTerminal: URIRef =
rdflib.term.URIRef('https://schema.org/departureTerminal')
departureTime: URIRef = rdflib.term.URIRef('https://schema.org/departureTime')
dependencies: URIRef = rdflib.term.URIRef('https://schema.org/dependencies')
```

```

depth: URIRef = rdflib.term.URIRef('https://schema.org/depth')
description: URIRef = rdflib.term.URIRef('https://schema.org/description')
device: URIRef = rdflib.term.URIRef('https://schema.org/device')
diagnosis: URIRef = rdflib.term.URIRef('https://schema.org/diagnosis')
diagram: URIRef = rdflib.term.URIRef('https://schema.org/diagram')
diet: URIRef = rdflib.term.URIRef('https://schema.org/diet')
dietFeatures: URIRef = rdflib.term.URIRef('https://schema.org/dietFeatures')
differentialDiagnosis: URIRef =
rdflib.term.URIRef('https://schema.org/differentialDiagnosis')
directApply: URIRef = rdflib.term.URIRef('https://schema.org/directApply')
director: URIRef = rdflib.term.URIRef('https://schema.org/director')
directors: URIRef = rdflib.term.URIRef('https://schema.org/directors')
disambiguatingDescription: URIRef =
rdflib.term.URIRef('https://schema.org/disambiguatingDescription')
discount: URIRef = rdflib.term.URIRef('https://schema.org/discount')
discountCode: URIRef = rdflib.term.URIRef('https://schema.org/discountCode')
discountCurrency: URIRef =
rdflib.term.URIRef('https://schema.org/discountCurrency')
discusses: URIRef = rdflib.term.URIRef('https://schema.org/discusses')
discussionUrl: URIRef = rdflib.term.URIRef('https://schema.org/discussionUrl')
diseasePreventionInfo: URIRef =
rdflib.term.URIRef('https://schema.org/diseasePreventionInfo')
diseaseSpreadStatistics: URIRef =
rdflib.term.URIRef('https://schema.org/diseaseSpreadStatistics')
dissolutionDate: URIRef = rdflib.term.URIRef('https://schema.org/dissolutionDate')
distance: URIRef = rdflib.term.URIRef('https://schema.org/distance')
distinguishingSign: URIRef =
rdflib.term.URIRef('https://schema.org/distinguishingSign')
distribution: URIRef = rdflib.term.URIRef('https://schema.org/distribution')
diversityPolicy: URIRef = rdflib.term.URIRef('https://schema.org/diversityPolicy')
diversityStaffingReport: URIRef =
rdflib.term.URIRef('https://schema.org/diversityStaffingReport')
documentation: URIRef = rdflib.term.URIRef('https://schema.org/documentation')

```



```
doesNotShip: URIRef = rdflib.term.URIRef('https://schema.org/doesNotShip')

domainIncludes: URIRef = rdflib.term.URIRef('https://schema.org/domainIncludes')

domiciledMortgage: URIRef =
rdflib.term.URIRef('https://schema.org/domiciledMortgage')

doorTime: URIRef = rdflib.term.URIRef('https://schema.org/doorTime')

dosageForm: URIRef = rdflib.term.URIRef('https://schema.org/dosageForm')

doseSchedule: URIRef = rdflib.term.URIRef('https://schema.org/doseSchedule')

doseUnit: URIRef = rdflib.term.URIRef('https://schema.org/doseUnit')

doseValue: URIRef = rdflib.term.URIRef('https://schema.org/doseValue')

downPayment: URIRef = rdflib.term.URIRef('https://schema.org/downPayment')

downloadUrl: URIRef = rdflib.term.URIRef('https://schema.org/downloadUrl')

downvoteCount: URIRef = rdflib.term.URIRef('https://schema.org/downvoteCount')

drainsTo: URIRef = rdflib.term.URIRef('https://schema.org/drainsTo')

driveWheelConfiguration: URIRef =
rdflib.term.URIRef('https://schema.org/driveWheelConfiguration')

dropoffLocation: URIRef = rdflib.term.URIRef('https://schema.org/dropoffLocation')

dropoffTime: URIRef = rdflib.term.URIRef('https://schema.org/dropoffTime')

drug: URIRef = rdflib.term.URIRef('https://schema.org/drug')

drugClass: URIRef = rdflib.term.URIRef('https://schema.org/drugClass')

drugUnit: URIRef = rdflib.term.URIRef('https://schema.org/drugUnit')

duns: URIRef = rdflib.term.URIRef('https://schema.org/duns')

duplicateTherapy: URIRef =
rdflib.term.URIRef('https://schema.org/duplicateTherapy')

duration: URIRef = rdflib.term.URIRef('https://schema.org/duration')

durationOfWarranty: URIRef =
rdflib.term.URIRef('https://schema.org/durationOfWarranty')

duringMedia: URIRef = rdflib.term.URIRef('https://schema.org/duringMedia')

earlyPrepaymentPenalty: URIRef =
rdflib.term.URIRef('https://schema.org/earlyPrepaymentPenalty')

editEIDR: URIRef = rdflib.term.URIRef('https://schema.org/editEIDR')

editor: URIRef = rdflib.term.URIRef('https://schema.org/editor')

eduQuestionType: URIRef = rdflib.term.URIRef('https://schema.org/eduQuestionType')
```



```
educationRequirements: URIRef =  
rdflib.term.URIRef('https://schema.org/educationRequirements')  
  
educationalAlignment: URIRef =  
rdflib.term.URIRef('https://schema.org/educationalAlignment')  
  
educationalCredentialAwarded: URIRef =  
rdflib.term.URIRef('https://schema.org/educationalCredentialAwarded')  
  
educationalFramework: URIRef =  
rdflib.term.URIRef('https://schema.org/educationalFramework')  
  
educationalLevel: URIRef =  
rdflib.term.URIRef('https://schema.org/educationalLevel')  
  
educationalProgramMode: URIRef =  
rdflib.term.URIRef('https://schema.org/educationalProgramMode')  
  
educationalRole: URIRef = rdflib.term.URIRef('https://schema.org/educationalRole')  
  
educationalUse: URIRef = rdflib.term.URIRef('https://schema.org/educationalUse')  
  
elevation: URIRef = rdflib.term.URIRef('https://schema.org/elevation')  
  
eligibilityToWorkRequirement: URIRef =  
rdflib.term.URIRef('https://schema.org/eligibilityToWorkRequirement')  
  
eligibleCustomerType: URIRef =  
rdflib.term.URIRef('https://schema.org/eligibleCustomerType')  
  
eligibleDuration: URIRef =  
rdflib.term.URIRef('https://schema.org/eligibleDuration')  
  
eligibleQuantity: URIRef =  
rdflib.term.URIRef('https://schema.org/eligibleQuantity')  
  
eligibleRegion: URIRef = rdflib.term.URIRef('https://schema.org/eligibleRegion')  
  
eligibleTransactionVolume: URIRef =  
rdflib.term.URIRef('https://schema.org/eligibleTransactionVolume')  
  
email: URIRef = rdflib.term.URIRef('https://schema.org/email')  
  
embedUrl: URIRef = rdflib.term.URIRef('https://schema.org/embedUrl')  
  
embeddedTextCaption: URIRef =  
rdflib.term.URIRef('https://schema.org/embeddedTextCaption')  
  
emissionsCO2: URIRef = rdflib.term.URIRef('https://schema.org/emissionsCO2')  
  
employee: URIRef = rdflib.term.URIRef('https://schema.org/employee')  
  
employees: URIRef = rdflib.term.URIRef('https://schema.org/employees')  
  
employerOverview: URIRef =  
rdflib.term.URIRef('https://schema.org/employerOverview')  
  
employmentType: URIRef = rdflib.term.URIRef('https://schema.org/employmentType')
```

```
employmentUnit: URIRef = rdflib.term.URIRef('https://schema.org/employmentUnit')

encodesBioChemEntity: URIRef =
rdflib.term.URIRef('https://schema.org/encodesBioChemEntity')

encodesCreativeWork: URIRef =
rdflib.term.URIRef('https://schema.org/encodesCreativeWork')

encoding: URIRef = rdflib.term.URIRef('https://schema.org/encoding')

encodingFormat: URIRef = rdflib.term.URIRef('https://schema.org/encodingFormat')

encodingType: URIRef = rdflib.term.URIRef('https://schema.org/encodingType')

encodings: URIRef = rdflib.term.URIRef('https://schema.org/encodings')

endDate: URIRef = rdflib.term.URIRef('https://schema.org/endDate')

endOffset: URIRef = rdflib.term.URIRef('https://schema.org/endOffset')

endTime: URIRef = rdflib.term.URIRef('https://schema.org/endTime')

endorsee: URIRef = rdflib.term.URIRef('https://schema.org/endorsee')

endorsers: URIRef = rdflib.term.URIRef('https://schema.org/endorsers')

energyEfficiencyScaleMax: URIRef =
rdflib.term.URIRef('https://schema.org/energyEfficiencyScaleMax')

energyEfficiencyScaleMin: URIRef =
rdflib.term.URIRef('https://schema.org/energyEfficiencyScaleMin')

engineDisplacement: URIRef =
rdflib.term.URIRef('https://schema.org/engineDisplacement')

enginePower: URIRef = rdflib.term.URIRef('https://schema.org/enginePower')

engineType: URIRef = rdflib.term.URIRef('https://schema.org/engineType')

entertainmentBusiness: URIRef =
rdflib.term.URIRef('https://schema.org/entertainmentBusiness')

epidemiology: URIRef = rdflib.term.URIRef('https://schema.org/epidemiology')

episode: URIRef = rdflib.term.URIRef('https://schema.org/episode')

episodeNumber: URIRef = rdflib.term.URIRef('https://schema.org/episodeNumber')

episodes: URIRef = rdflib.term.URIRef('https://schema.org/episodes')

equal: URIRef = rdflib.term.URIRef('https://schema.org/equal')

error: URIRef = rdflib.term.URIRef('https://schema.org/error')

estimatedCost: URIRef = rdflib.term.URIRef('https://schema.org/estimatedCost')

estimatedFlightDuration: URIRef =
rdflib.term.URIRef('https://schema.org/estimatedFlightDuration')
```

```

estimatedSalary: URIRef = rdflib.term.URIRef('https://schema.org/estimatedSalary')
estimatesRiskOf: URIRef = rdflib.term.URIRef('https://schema.org/estimatesRiskOf')
ethicsPolicy: URIRef = rdflib.term.URIRef('https://schema.org/ethicsPolicy')
event: URIRef = rdflib.term.URIRef('https://schema.org/event')
eventAttendanceMode: URIRef =
rdflib.term.URIRef('https://schema.org/eventAttendanceMode')
eventSchedule: URIRef = rdflib.term.URIRef('https://schema.org/eventSchedule')
eventStatus: URIRef = rdflib.term.URIRef('https://schema.org/eventStatus')
events: URIRef = rdflib.term.URIRef('https://schema.org/events')
evidenceLevel: URIRef = rdflib.term.URIRef('https://schema.org/evidenceLevel')
evidenceOrigin: URIRef = rdflib.term.URIRef('https://schema.org/evidenceOrigin')
exampleOfWork: URIRef = rdflib.term.URIRef('https://schema.org/exampleOfWork')
exceptDate: URIRef = rdflib.term.URIRef('https://schema.org/exceptDate')
exchangeRateSpread: URIRef =
rdflib.term.URIRef('https://schema.org/exchangeRateSpread')
executableLibraryName: URIRef =
rdflib.term.URIRef('https://schema.org/executableLibraryName')
exerciseCourse: URIRef = rdflib.term.URIRef('https://schema.org/exerciseCourse')
exercisePlan: URIRef = rdflib.term.URIRef('https://schema.org/exercisePlan')
exerciseRelatedDiet: URIRef =
rdflib.term.URIRef('https://schema.org/exerciseRelatedDiet')
exerciseType: URIRef = rdflib.term.URIRef('https://schema.org/exerciseType')
exifData: URIRef = rdflib.term.URIRef('https://schema.org/exifData')
expectedArrivalFrom: URIRef =
rdflib.term.URIRef('https://schema.org/expectedArrivalFrom')
expectedArrivalUntil: URIRef =
rdflib.term.URIRef('https://schema.org/expectedArrivalUntil')
expectedPrognosis: URIRef =
rdflib.term.URIRef('https://schema.org/expectedPrognosis')
expectsAcceptanceOf: URIRef =
rdflib.term.URIRef('https://schema.org/expectsAcceptanceOf')
experienceInPlaceOfEducation: URIRef =
rdflib.term.URIRef('https://schema.org/experienceInPlaceOfEducation')
experienceRequirements: URIRef =
rdflib.term.URIRef('https://schema.org/experienceRequirements')

```

```

expertConsiderations: URIRef =
rdflib.term.URIRef('https://schema.org/expertConsiderations')

expires: URIRef = rdflib.term.URIRef('https://schema.org/expires')

expressedIn: URIRef = rdflib.term.URIRef('https://schema.org/expressedIn')

familyName: URIRef = rdflib.term.URIRef('https://schema.org/familyName')

fatContent: URIRef = rdflib.term.URIRef('https://schema.org/fatContent')

faxNumber: URIRef = rdflib.term.URIRef('https://schema.org/faxNumber')

featureList: URIRef = rdflib.term.URIRef('https://schema.org/featureList')

feesAndCommissionsSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/feesAndCommissionsSpecification')

fiberContent: URIRef = rdflib.term.URIRef('https://schema.org/fiberContent')

fileFormat: URIRef = rdflib.term.URIRef('https://schema.org/fileFormat')

fileSize: URIRef = rdflib.term.URIRef('https://schema.org/fileSize')

financialAidEligible: URIRef =
rdflib.term.URIRef('https://schema.org/financialAidEligible')

firstAppearance: URIRef = rdflib.term.URIRef('https://schema.org/firstAppearance')

firstPerformance: URIRef =
rdflib.term.URIRef('https://schema.org/firstPerformance')

flightDistance: URIRef = rdflib.term.URIRef('https://schema.org/flightDistance')

flightNumber: URIRef = rdflib.term.URIRef('https://schema.org/flightNumber')

floorLevel: URIRef = rdflib.term.URIRef('https://schema.org/floorLevel')

floorLimit: URIRef = rdflib.term.URIRef('https://schema.org/floorLimit')

floorSize: URIRef = rdflib.term.URIRef('https://schema.org/floorSize')

followee: URIRef = rdflib.term.URIRef('https://schema.org/followee')

follows: URIRef = rdflib.term.URIRef('https://schema.org/follows')

followup: URIRef = rdflib.term.URIRef('https://schema.org/followup')

foodEstablishment: URIRef =
rdflib.term.URIRef('https://schema.org/foodEstablishment')

foodEvent: URIRef = rdflib.term.URIRef('https://schema.org/foodEvent')

foodWarning: URIRef = rdflib.term.URIRef('https://schema.org/foodWarning')

founder: URIRef = rdflib.term.URIRef('https://schema.org/founder')

founders: URIRef = rdflib.term.URIRef('https://schema.org/founders')

foundingDate: URIRef = rdflib.term.URIRef('https://schema.org/foundingDate')

```

```
foundingLocation: URIRef =  
rdflib.term.URIRef('https://schema.org/foundingLocation')  
  
free: URIRef = rdflib.term.URIRef('https://schema.org/free')  
  
freeShippingThreshold: URIRef =  
rdflib.term.URIRef('https://schema.org/freeShippingThreshold')  
  
frequency: URIRef = rdflib.term.URIRef('https://schema.org/frequency')  
  
fromLocation: URIRef = rdflib.term.URIRef('https://schema.org/fromLocation')  
  
fuelCapacity: URIRef = rdflib.term.URIRef('https://schema.org/fuelCapacity')  
  
fuelConsumption: URIRef = rdflib.term.URIRef('https://schema.org/fuelConsumption')  
  
fuelEfficiency: URIRef = rdflib.term.URIRef('https://schema.org/fuelEfficiency')  
  
fuelType: URIRef = rdflib.term.URIRef('https://schema.org/fuelType')  
  
functionalClass: URIRef = rdflib.term.URIRef('https://schema.org/functionalClass')  
  
fundedItem: URIRef = rdflib.term.URIRef('https://schema.org/fundedItem')  
  
funder: URIRef = rdflib.term.URIRef('https://schema.org/funder')  
  
game: URIRef = rdflib.term.URIRef('https://schema.org/game')  
  
gameItem: URIRef = rdflib.term.URIRef('https://schema.org/gameItem')  
  
gameLocation: URIRef = rdflib.term.URIRef('https://schema.org/gameLocation')  
  
gamePlatform: URIRef = rdflib.term.URIRef('https://schema.org/gamePlatform')  
  
gameServer: URIRef = rdflib.term.URIRef('https://schema.org/gameServer')  
  
gameTip: URIRef = rdflib.term.URIRef('https://schema.org/gameTip')  
  
gender: URIRef = rdflib.term.URIRef('https://schema.org/gender')  
  
genre: URIRef = rdflib.term.URIRef('https://schema.org/genre')  
  
geo: URIRef = rdflib.term.URIRef('https://schema.org/geo')  
  
geoContains: URIRef = rdflib.term.URIRef('https://schema.org/geoContains')  
  
geoCoveredBy: URIRef = rdflib.term.URIRef('https://schema.org/geoCoveredBy')  
  
geoCovers: URIRef = rdflib.term.URIRef('https://schema.org/geoCovers')  
  
geoCrosses: URIRef = rdflib.term.URIRef('https://schema.org/geoCrosses')  
  
geoDisjoint: URIRef = rdflib.term.URIRef('https://schema.org/geoDisjoint')  
  
geoEquals: URIRef = rdflib.term.URIRef('https://schema.org/geoEquals')  
  
geoIntersects: URIRef = rdflib.term.URIRef('https://schema.org/geoIntersects')  
  
geoMidpoint: URIRef = rdflib.term.URIRef('https://schema.org/geoMidpoint')
```

```
geoOverlaps: URIRef = rdflib.term.URIRef('https://schema.org/geoOverlaps')
geoRadius: URIRef = rdflib.term.URIRef('https://schema.org/geoRadius')
geoTouches: URIRef = rdflib.term.URIRef('https://schema.org/geoTouches')
geoWithin: URIRef = rdflib.term.URIRef('https://schema.org/geoWithin')
geographicArea: URIRef = rdflib.term.URIRef('https://schema.org/geographicArea')
gettingTestedInfo: URIRef =
rdflib.term.URIRef('https://schema.org/gettingTestedInfo')
givenName: URIRef = rdflib.term.URIRef('https://schema.org/givenName')
globalLocationNumber: URIRef =
rdflib.term.URIRef('https://schema.org/globalLocationNumber')
governmentBenefitsInfo: URIRef =
rdflib.term.URIRef('https://schema.org/governmentBenefitsInfo')
gracePeriod: URIRef = rdflib.term.URIRef('https://schema.org/gracePeriod')
grantee: URIRef = rdflib.term.URIRef('https://schema.org/grantee')
greater: URIRef = rdflib.term.URIRef('https://schema.org/greater')
greaterOrEqual: URIRef = rdflib.term.URIRef('https://schema.org/greaterOrEqual')
gtin: URIRef = rdflib.term.URIRef('https://schema.org/gtin')
gtin12: URIRef = rdflib.term.URIRef('https://schema.org/gtin12')
gtin13: URIRef = rdflib.term.URIRef('https://schema.org/gtin13')
gtin14: URIRef = rdflib.term.URIRef('https://schema.org/gtin14')
gtin8: URIRef = rdflib.term.URIRef('https://schema.org/gtin8')
guideline: URIRef = rdflib.term.URIRef('https://schema.org/guideline')
guidelineDate: URIRef = rdflib.term.URIRef('https://schema.org/guidelineDate')
guidelineSubject: URIRef =
rdflib.term.URIRef('https://schema.org/guidelineSubject')
handlingTime: URIRef = rdflib.term.URIRef('https://schema.org/handlingTime')
hasBioChemEntityPart: URIRef =
rdflib.term.URIRef('https://schema.org/hasBioChemEntityPart')
hasBioPolymerSequence: URIRef =
rdflib.term.URIRef('https://schema.org/hasBioPolymerSequence')
hasBroadcastChannel: URIRef =
rdflib.term.URIRef('https://schema.org/hasBroadcastChannel')
hasCategoryCode: URIRef = rdflib.term.URIRef('https://schema.org/hasCategoryCode')
```

```
hasCourse: URIRef = rdflib.term.URIRef('https://schema.org/hasCourse')

hasCourseInstance: URIRef =
rdflib.term.URIRef('https://schema.org/hasCourseInstance')

hasCredential: URIRef = rdflib.term.URIRef('https://schema.org/hasCredential')

hasDefinedTerm: URIRef = rdflib.term.URIRef('https://schema.org/hasDefinedTerm')

hasDeliveryMethod: URIRef =
rdflib.term.URIRef('https://schema.org/hasDeliveryMethod')

hasDigitalDocumentPermission: URIRef =
rdflib.term.URIRef('https://schema.org/hasDigitalDocumentPermission')

hasDriveThroughService: URIRef =
rdflib.term.URIRef('https://schema.org/hasDriveThroughService')

hasEnergyConsumptionDetails: URIRef =
rdflib.term.URIRef('https://schema.org/hasEnergyConsumptionDetails')

hasEnergyEfficiencyCategory: URIRef =
rdflib.term.URIRef('https://schema.org/hasEnergyEfficiencyCategory')

hasHealthAspect: URIRef = rdflib.term.URIRef('https://schema.org/hasHealthAspect')

hasMap: URIRef = rdflib.term.URIRef('https://schema.org/hasMap')

hasMeasurement: URIRef = rdflib.term.URIRef('https://schema.org/hasMeasurement')

hasMenu: URIRef = rdflib.term.URIRef('https://schema.org/hasMenu')

hasMenuItem: URIRef = rdflib.term.URIRef('https://schema.org/hasMenuItem')

hasMenuSection: URIRef = rdflib.term.URIRef('https://schema.org/hasMenuSection')

hasMerchantReturnPolicy: URIRef =
rdflib.term.URIRef('https://schema.org/hasMerchantReturnPolicy')

hasMolecularFunction: URIRef =
rdflib.term.URIRef('https://schema.org/hasMolecularFunction')

hasOccupation: URIRef = rdflib.term.URIRef('https://schema.org/hasOccupation')

hasOfferCatalog: URIRef = rdflib.term.URIRef('https://schema.org/hasOfferCatalog')

hasPOS: URIRef = rdflib.term.URIRef('https://schema.org/hasPOS')

hasPart: URIRef = rdflib.term.URIRef('https://schema.org/hasPart')

hasRepresentation: URIRef =
rdflib.term.URIRef('https://schema.org/hasRepresentation')

hasVariant: URIRef = rdflib.term.URIRef('https://schema.org/hasVariant')

headline: URIRef = rdflib.term.URIRef('https://schema.org/headline')

healthCondition: URIRef = rdflib.term.URIRef('https://schema.org/healthCondition')
```



```

healthPlanCoinsuranceOption: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanCoinsuranceOption')

healthPlanCoinsuranceRate: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanCoinsuranceRate')

healthPlanCopay: URIRef = rdflib.term.URIRef('https://schema.org/healthPlanCopay')

healthPlanCopayOption: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanCopayOption')

healthPlanCostSharing: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanCostSharing')

healthPlanDrugOption: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanDrugOption')

healthPlanDrugTier: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanDrugTier')

healthPlanId: URIRef = rdflib.term.URIRef('https://schema.org/healthPlanId')

healthPlanMarketingUrl: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanMarketingUrl')

healthPlanNetworkId: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanNetworkId')

healthPlanNetworkTier: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanNetworkTier')

healthPlanPharmacyCategory: URIRef =
rdflib.term.URIRef('https://schema.org/healthPlanPharmacyCategory')

healthcareReportingData: URIRef =
rdflib.term.URIRef('https://schema.org/healthcareReportingData')

height: URIRef = rdflib.term.URIRef('https://schema.org/height')

highPrice: URIRef = rdflib.term.URIRef('https://schema.org/highPrice')

hiringOrganization: URIRef =
rdflib.term.URIRef('https://schema.org/hiringOrganization')

holdingArchive: URIRef = rdflib.term.URIRef('https://schema.org/holdingArchive')

homeLocation: URIRef = rdflib.term.URIRef('https://schema.org/homeLocation')

homeTeam: URIRef = rdflib.term.URIRef('https://schema.org/homeTeam')

honorificPrefix: URIRef = rdflib.term.URIRef('https://schema.org/honorificPrefix')

honorificSuffix: URIRef = rdflib.term.URIRef('https://schema.org/honorificSuffix')

hospitalAffiliation: URIRef =
rdflib.term.URIRef('https://schema.org/hospitalAffiliation')

```



```

hostingOrganization: URIRef =
rdflib.term.URIRef('https://schema.org/hostingOrganization')

hoursAvailable: URIRef = rdflib.term.URIRef('https://schema.org/hoursAvailable')

howPerformed: URIRef = rdflib.term.URIRef('https://schema.org/howPerformed')

httpMethod: URIRef = rdflib.term.URIRef('https://schema.org/httpMethod')

iataCode: URIRef = rdflib.term.URIRef('https://schema.org/iataCode')

icaoCode: URIRef = rdflib.term.URIRef('https://schema.org/icaoCode')

identifier: URIRef = rdflib.term.URIRef('https://schema.org/identifier')

identifyingExam: URIRef = rdflib.term.URIRef('https://schema.org/identifyingExam')

identifyingTest: URIRef = rdflib.term.URIRef('https://schema.org/identifyingTest')

illustrator: URIRef = rdflib.term.URIRef('https://schema.org/illustrator')

image: URIRef = rdflib.term.URIRef('https://schema.org/image')

imagingTechnique: URIRef =
rdflib.term.URIRef('https://schema.org/imagingTechnique')

inAlbum: URIRef = rdflib.term.URIRef('https://schema.org/inAlbum')

inBroadcastLineup: URIRef =
rdflib.term.URIRef('https://schema.org/inBroadcastLineup')

inChI: URIRef = rdflib.term.URIRef('https://schema.org/inChI')

inChIKey: URIRef = rdflib.term.URIRef('https://schema.org/inChIKey')

inCodeSet: URIRef = rdflib.term.URIRef('https://schema.org/inCodeSet')

inDefinedTermSet: URIRef =
rdflib.term.URIRef('https://schema.org/inDefinedTermSet')

inLanguage: URIRef = rdflib.term.URIRef('https://schema.org/inLanguage')

inPlaylist: URIRef = rdflib.term.URIRef('https://schema.org/inPlaylist')

inProductGroupWithID: URIRef =
rdflib.term.URIRef('https://schema.org/inProductGroupWithID')

inStoreReturnsOffered: URIRef =
rdflib.term.URIRef('https://schema.org/inStoreReturnsOffered')

inSupportOf: URIRef = rdflib.term.URIRef('https://schema.org/inSupportOf')

incentiveCompensation: URIRef =
rdflib.term.URIRef('https://schema.org/incentiveCompensation')

incentives: URIRef = rdflib.term.URIRef('https://schema.org/incentives')

includedComposition: URIRef =
rdflib.term.URIRef('https://schema.org/includedComposition')

```

```
includedDataCatalog: URIRef =  
rdflib.term.URIRef('https://schema.org/includedDataCatalog')  
  
includedInDataCatalog: URIRef =  
rdflib.term.URIRef('https://schema.org/includedInDataCatalog')  
  
includedInHealthInsurancePlan: URIRef =  
rdflib.term.URIRef('https://schema.org/includedInHealthInsurancePlan')  
  
includedRiskFactor: URIRef =  
rdflib.term.URIRef('https://schema.org/includedRiskFactor')  
  
includesAttraction: URIRef =  
rdflib.term.URIRef('https://schema.org/includesAttraction')  
  
includesHealthPlanFormulary: URIRef =  
rdflib.term.URIRef('https://schema.org/includesHealthPlanFormulary')  
  
includesHealthPlanNetwork: URIRef =  
rdflib.term.URIRef('https://schema.org/includesHealthPlanNetwork')  
  
includesObject: URIRef = rdflib.term.URIRef('https://schema.org/includesObject')  
  
increasesRiskOf: URIRef = rdflib.term.URIRef('https://schema.org/increasesRiskOf')  
  
industry: URIRef = rdflib.term.URIRef('https://schema.org/industry')  
  
ineligibleRegion: URIRef =  
rdflib.term.URIRef('https://schema.org/ineligibleRegion')  
  
infectiousAgent: URIRef = rdflib.term.URIRef('https://schema.org/infectiousAgent')  
  
infectiousAgentClass: URIRef =  
rdflib.term.URIRef('https://schema.org/infectiousAgentClass')  
  
ingredients: URIRef = rdflib.term.URIRef('https://schema.org/ingredients')  
  
inker: URIRef = rdflib.term.URIRef('https://schema.org/inker')  
  
insertion: URIRef = rdflib.term.URIRef('https://schema.org/insertion')  
  
installUrl: URIRef = rdflib.term.URIRef('https://schema.org/installUrl')  
  
instructor: URIRef = rdflib.term.URIRef('https://schema.org/instructor')  
  
instrument: URIRef = rdflib.term.URIRef('https://schema.org/instrument')  
  
intensity: URIRef = rdflib.term.URIRef('https://schema.org/intensity')  
  
interactingDrug: URIRef = rdflib.term.URIRef('https://schema.org/interactingDrug')  
  
interactionCount: URIRef =  
rdflib.term.URIRef('https://schema.org/interactionCount')  
  
interactionService: URIRef =  
rdflib.term.URIRef('https://schema.org/interactionService')
```

```
interactionStatistic: URIRef =  
rdflib.term.URIRef('https://schema.org/interactionStatistic')  
  
interactionType: URIRef = rdflib.term.URIRef('https://schema.org/interactionType')  
  
interactivityType: URIRef =  
rdflib.term.URIRef('https://schema.org/interactivityType')  
  
interestRate: URIRef = rdflib.term.URIRef('https://schema.org/interestRate')  
  
interpretedAsClaim: URIRef =  
rdflib.term.URIRef('https://schema.org/interpretedAsClaim')  
  
inventoryLevel: URIRef = rdflib.term.URIRef('https://schema.org/inventoryLevel')  
  
inverseOf: URIRef = rdflib.term.URIRef('https://schema.org/inverseOf')  
  
isAcceptingNewPatients: URIRef =  
rdflib.term.URIRef('https://schema.org/isAcceptingNewPatients')  
  
isAccessibleForFree: URIRef =  
rdflib.term.URIRef('https://schema.org/isAccessibleForFree')  
  
isAccessoryOrSparePartFor: URIRef =  
rdflib.term.URIRef('https://schema.org/isAccessoryOrSparePartFor')  
  
isAvailableGenerically: URIRef =  
rdflib.term.URIRef('https://schema.org/isAvailableGenerically')  
  
isBasedOn: URIRef = rdflib.term.URIRef('https://schema.org/isBasedOn')  
  
isBasedOnUrl: URIRef = rdflib.term.URIRef('https://schema.org/isBasedOnUrl')  
  
isConsumableFor: URIRef = rdflib.term.URIRef('https://schema.org/isConsumableFor')  
  
isEncodedByBioChemEntity: URIRef =  
rdflib.term.URIRef('https://schema.org/isEncodedByBioChemEntity')  
  
isFamilyFriendly: URIRef =  
rdflib.term.URIRef('https://schema.org/isFamilyFriendly')  
  
isGift: URIRef = rdflib.term.URIRef('https://schema.org/isGift')  
  
isInvolvedInBiologicalProcess: URIRef =  
rdflib.term.URIRef('https://schema.org/isInvolvedInBiologicalProcess')  
  
isLiveBroadcast: URIRef = rdflib.term.URIRef('https://schema.org/isLiveBroadcast')  
  
isLocatedInSubcellularLocation: URIRef =  
rdflib.term.URIRef('https://schema.org/isLocatedInSubcellularLocation')  
  
isPartOf: URIRef = rdflib.term.URIRef('https://schema.org/isPartOf')  
  
isPartOfBioChemEntity: URIRef =  
rdflib.term.URIRef('https://schema.org/isPartOfBioChemEntity')  
  
isPlanForApartment: URIRef =  
rdflib.term.URIRef('https://schema.org/isPlanForApartment')
```

```
isProprietary: URIRef = rdflib.term.URIRef('https://schema.org/isProprietary')
isRelatedTo: URIRef = rdflib.term.URIRef('https://schema.org/isRelatedTo')
isResizable: URIRef = rdflib.term.URIRef('https://schema.org/isResizable')
isSimilarTo: URIRef = rdflib.term.URIRef('https://schema.org/isSimilarTo')
isUnlabelledFallback: URIRef =
rdflib.term.URIRef('https://schema.org/isUnlabelledFallback')
isVariantOf: URIRef = rdflib.term.URIRef('https://schema.org/isVariantOf')
isbn: URIRef = rdflib.term.URIRef('https://schema.org/isbn')
isicV4: URIRef = rdflib.term.URIRef('https://schema.org/isicV4')
isrcCode: URIRef = rdflib.term.URIRef('https://schema.org/isrcCode')
issn: URIRef = rdflib.term.URIRef('https://schema.org/issn')
issueNumber: URIRef = rdflib.term.URIRef('https://schema.org/issueNumber')
issuedBy: URIRef = rdflib.term.URIRef('https://schema.org/issuedBy')
issuedThrough: URIRef = rdflib.term.URIRef('https://schema.org/issuedThrough')
iswcCode: URIRef = rdflib.term.URIRef('https://schema.org/iswcCode')
item: URIRef = rdflib.term.URIRef('https://schema.org/item')
itemCondition: URIRef = rdflib.term.URIRef('https://schema.org/itemCondition')
itemDefectReturnFees: URIRef =
rdflib.term.URIRef('https://schema.org/itemDefectReturnFees')
itemDefectReturnLabelSource: URIRef =
rdflib.term.URIRef('https://schema.org/itemDefectReturnLabelSource')
itemDefectReturnShippingFeesAmount: URIRef =
rdflib.term.URIRef('https://schema.org/itemDefectReturnShippingFeesAmount')
itemListElement: URIRef = rdflib.term.URIRef('https://schema.org/itemListElement')
itemListOrder: URIRef = rdflib.term.URIRef('https://schema.org/itemListOrder')
itemLocation: URIRef = rdflib.term.URIRef('https://schema.org/itemLocation')
itemOffered: URIRef = rdflib.term.URIRef('https://schema.org/itemOffered')
itemReviewed: URIRef = rdflib.term.URIRef('https://schema.org/itemReviewed')
itemShipped: URIRef = rdflib.term.URIRef('https://schema.org/itemShipped')
itinerary: URIRef = rdflib.term.URIRef('https://schema.org/itinerary')
iupacName: URIRef = rdflib.term.URIRef('https://schema.org/iupacName')
jobBenefits: URIRef = rdflib.term.URIRef('https://schema.org/jobBenefits')
```

```
jobImmediateStart: URIRef =  
rdflib.term.URIRef('https://schema.org/jobImmediateStart')  
  
jobLocation: URIRef = rdflib.term.URIRef('https://schema.org/jobLocation')  
  
jobLocationType: URIRef = rdflib.term.URIRef('https://schema.org/jobLocationType')  
  
jobStartDate: URIRef = rdflib.term.URIRef('https://schema.org/jobStartDate')  
  
jobTitle: URIRef = rdflib.term.URIRef('https://schema.org/jobTitle')  
  
jurisdiction: URIRef = rdflib.term.URIRef('https://schema.org/jurisdiction')  
  
keywords: URIRef = rdflib.term.URIRef('https://schema.org/keywords')  
  
knownVehicleDamages: URIRef =  
rdflib.term.URIRef('https://schema.org/knownVehicleDamages')  
  
knows: URIRef = rdflib.term.URIRef('https://schema.org/knows')  
  
knowsAbout: URIRef = rdflib.term.URIRef('https://schema.org/knowsAbout')  
  
knowsLanguage: URIRef = rdflib.term.URIRef('https://schema.org/knowsLanguage')  
  
labelDetails: URIRef = rdflib.term.URIRef('https://schema.org/labelDetails')  
  
landlord: URIRef = rdflib.term.URIRef('https://schema.org/landlord')  
  
language: URIRef = rdflib.term.URIRef('https://schema.org/language')  
  
lastReviewed: URIRef = rdflib.term.URIRef('https://schema.org/lastReviewed')  
  
latitude: URIRef = rdflib.term.URIRef('https://schema.org/latitude')  
  
layoutImage: URIRef = rdflib.term.URIRef('https://schema.org/layoutImage')  
  
learningResourceType: URIRef =  
rdflib.term.URIRef('https://schema.org/learningResourceType')  
  
leaseLength: URIRef = rdflib.term.URIRef('https://schema.org/leaseLength')  
  
legalName: URIRef = rdflib.term.URIRef('https://schema.org/legalName')  
  
legalStatus: URIRef = rdflib.term.URIRef('https://schema.org/legalStatus')  
  
legislationApplies: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationApplies')  
  
legislationChanges: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationChanges')  
  
legislationConsolidates: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationConsolidates')  
  
legislationDate: URIRef = rdflib.term.URIRef('https://schema.org/legislationDate')  
  
legislationDateVersion: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationDateVersion')
```

```
legislationIdentifier: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationIdentifier')  
  
legislationJurisdiction: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationJurisdiction')  
  
legislationLegalForce: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationLegalForce')  
  
legislationLegalValue: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationLegalValue')  
  
legislationPassedBy: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationPassedBy')  
  
legislationResponsible: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationResponsible')  
  
legislationTransposes: URIRef =  
rdflib.term.URIRef('https://schema.org/legislationTransposes')  
  
legislationType: URIRef = rdflib.term.URIRef('https://schema.org/legislationType')  
  
leiCode: URIRef = rdflib.term.URIRef('https://schema.org/leiCode')  
  
lender: URIRef = rdflib.term.URIRef('https://schema.org/lender')  
  
lesser: URIRef = rdflib.term.URIRef('https://schema.org/lesser')  
  
lesserOrEqual: URIRef = rdflib.term.URIRef('https://schema.org/lesserOrEqual')  
  
letterer: URIRef = rdflib.term.URIRef('https://schema.org/letterer')  
  
license: URIRef = rdflib.term.URIRef('https://schema.org/license')  
  
line: URIRef = rdflib.term.URIRef('https://schema.org/line')  
  
linkRelationship: URIRef =  
rdflib.term.URIRef('https://schema.org/linkRelationship')  
  
liveBlogUpdate: URIRef = rdflib.term.URIRef('https://schema.org/liveBlogUpdate')  
  
loanMortgageMandateAmount: URIRef =  
rdflib.term.URIRef('https://schema.org/loanMortgageMandateAmount')  
  
loanPaymentAmount: URIRef =  
rdflib.term.URIRef('https://schema.org/loanPaymentAmount')  
  
loanPaymentFrequency: URIRef =  
rdflib.term.URIRef('https://schema.org/loanPaymentFrequency')  
  
loanRepaymentForm: URIRef =  
rdflib.term.URIRef('https://schema.org/loanRepaymentForm')  
  
loanTerm: URIRef = rdflib.term.URIRef('https://schema.org/loanTerm')  
  
loanType: URIRef = rdflib.term.URIRef('https://schema.org/loanType')
```

```

location: URIRef = rdflib.term.URIRef('https://schema.org/location')
locationCreated: URIRef = rdflib.term.URIRef('https://schema.org/locationCreated')
lodgingUnitDescription: URIRef =
rdflib.term.URIRef('https://schema.org/lodgingUnitDescription')
lodgingUnitType: URIRef = rdflib.term.URIRef('https://schema.org/lodgingUnitType')
logo: URIRef = rdflib.term.URIRef('https://schema.org/logo')
longitude: URIRef = rdflib.term.URIRef('https://schema.org/longitude')
loser: URIRef = rdflib.term.URIRef('https://schema.org/loser')
lowPrice: URIRef = rdflib.term.URIRef('https://schema.org/lowPrice')
lyricist: URIRef = rdflib.term.URIRef('https://schema.org/lyricist')
lyrics: URIRef = rdflib.term.URIRef('https://schema.org/lyrics')
mainContentOfPage: URIRef =
rdflib.term.URIRef('https://schema.org/mainContentOfPage')
mainEntity: URIRef = rdflib.term.URIRef('https://schema.org/mainEntity')
mainEntityOfPage: URIRef =
rdflib.term.URIRef('https://schema.org/mainEntityOfPage')
maintainer: URIRef = rdflib.term.URIRef('https://schema.org/maintainer')
makesOffer: URIRef = rdflib.term.URIRef('https://schema.org/makesOffer')
manufacturer: URIRef = rdflib.term.URIRef('https://schema.org/manufacturer')
map: URIRef = rdflib.term.URIRef('https://schema.org/map')
mapType: URIRef = rdflib.term.URIRef('https://schema.org/mapType')
maps: URIRef = rdflib.term.URIRef('https://schema.org/maps')
marginOfError: URIRef = rdflib.term.URIRef('https://schema.org/marginOfError')
masthead: URIRef = rdflib.term.URIRef('https://schema.org/masthead')
material: URIRef = rdflib.term.URIRef('https://schema.org/material')
materialExtent: URIRef = rdflib.term.URIRef('https://schema.org/materialExtent')
mathExpression: URIRef = rdflib.term.URIRef('https://schema.org/mathExpression')
maxPrice: URIRef = rdflib.term.URIRef('https://schema.org/maxPrice')
maxValue: URIRef = rdflib.term.URIRef('https://schema.org/maxValue')
maximumAttendeeCapacity: URIRef =
rdflib.term.URIRef('https://schema.org/maximumAttendeeCapacity')
maximumEnrollment: URIRef =
rdflib.term.URIRef('https://schema.org/maximumEnrollment')

```



```
maximumIntake: URIRef = rdflib.term.URIRef('https://schema.org/maximumIntake')

maximumPhysicalAttendeeCapacity: URIRef =
rdflib.term.URIRef('https://schema.org/maximumPhysicalAttendeeCapacity')

maximumVirtualAttendeeCapacity: URIRef =
rdflib.term.URIRef('https://schema.org/maximumVirtualAttendeeCapacity')

mealService: URIRef = rdflib.term.URIRef('https://schema.org/mealService')

measuredProperty: URIRef =
rdflib.term.URIRef('https://schema.org/measuredProperty')

measuredValue: URIRef = rdflib.term.URIRef('https://schema.org/measuredValue')

measurementTechnique: URIRef =
rdflib.term.URIRef('https://schema.org/measurementTechnique')

mechanismOfAction: URIRef =
rdflib.term.URIRef('https://schema.org/mechanismOfAction')

mediaAuthenticityCategory: URIRef =
rdflib.term.URIRef('https://schema.org/mediaAuthenticityCategory')

mediaItemAppearance: URIRef =
rdflib.term.URIRef('https://schema.org/mediaItemAppearance')

median: URIRef = rdflib.term.URIRef('https://schema.org/median')

medicalAudience: URIRef = rdflib.term.URIRef('https://schema.org/medicalAudience')

medicalSpecialty: URIRef =
rdflib.term.URIRef('https://schema.org/medicalSpecialty')

medicineSystem: URIRef = rdflib.term.URIRef('https://schema.org/medicineSystem')

meetsEmissionStandard: URIRef =
rdflib.term.URIRef('https://schema.org/meetsEmissionStandard')

member: URIRef = rdflib.term.URIRef('https://schema.org/member')

memberOf: URIRef = rdflib.term.URIRef('https://schema.org/memberOf')

members: URIRef = rdflib.term.URIRef('https://schema.org/members')

membershipNumber: URIRef =
rdflib.term.URIRef('https://schema.org/membershipNumber')

membershipPointsEarned: URIRef =
rdflib.term.URIRef('https://schema.org/membershipPointsEarned')

memoryRequirements: URIRef =
rdflib.term.URIRef('https://schema.org/memoryRequirements')

mentions: URIRef = rdflib.term.URIRef('https://schema.org/mentions')

menu: URIRef = rdflib.term.URIRef('https://schema.org/menu')
```



```
menuAddOn: URIRef = rdflib.term.URIRef('https://schema.org/menuAddOn')

merchant: URIRef = rdflib.term.URIRef('https://schema.org/merchant')

merchantReturnDays: URIRef =
rdflib.term.URIRef('https://schema.org/merchantReturnDays')

merchantReturnLink: URIRef =
rdflib.term.URIRef('https://schema.org/merchantReturnLink')

messageAttachment: URIRef =
rdflib.term.URIRef('https://schema.org/messageAttachment')

mileageFromOdometer: URIRef =
rdflib.term.URIRef('https://schema.org/mileageFromOdometer')

minPrice: URIRef = rdflib.term.URIRef('https://schema.org/minPrice')

minValue: URIRef = rdflib.term.URIRef('https://schema.org/minValue')

minimumPaymentDue: URIRef =
rdflib.term.URIRef('https://schema.org/minimumPaymentDue')

missionCoveragePrioritiesPolicy: URIRef =
rdflib.term.URIRef('https://schema.org/missionCoveragePrioritiesPolicy')

model: URIRef = rdflib.term.URIRef('https://schema.org/model')

modelDate: URIRef = rdflib.term.URIRef('https://schema.org/modelDate')

modifiedTime: URIRef = rdflib.term.URIRef('https://schema.org/modifiedTime')

molecularFormula: URIRef =
rdflib.term.URIRef('https://schema.org/molecularFormula')

molecularWeight: URIRef = rdflib.term.URIRef('https://schema.org/molecularWeight')

monoisotopicMolecularWeight: URIRef =
rdflib.term.URIRef('https://schema.org/monoisotopicMolecularWeight')

monthlyMinimumRepaymentAmount: URIRef =
rdflib.term.URIRef('https://schema.org/monthlyMinimumRepaymentAmount')

monthsOfExperience: URIRef =
rdflib.term.URIRef('https://schema.org/monthsOfExperience')

mpn: URIRef = rdflib.term.URIRef('https://schema.org/mpn')

multipleValues: URIRef = rdflib.term.URIRef('https://schema.org/multipleValues')

muscleAction: URIRef = rdflib.term.URIRef('https://schema.org/muscleAction')

musicArrangement: URIRef =
rdflib.term.URIRef('https://schema.org/musicArrangement')

musicBy: URIRef = rdflib.term.URIRef('https://schema.org/musicBy')
```

```
musicCompositionForm: URIRef =  
rdflib.term.URIRef('https://schema.org/musicCompositionForm')  
  
musicGroupMember: URIRef =  
rdflib.term.URIRef('https://schema.org/musicGroupMember')  
  
musicReleaseFormat: URIRef =  
rdflib.term.URIRef('https://schema.org/musicReleaseFormat')  
  
musicalKey: URIRef = rdflib.term.URIRef('https://schema.org/musicalKey')  
  
naics: URIRef = rdflib.term.URIRef('https://schema.org/naics')  
  
name: URIRef = rdflib.term.URIRef('https://schema.org/name')  
  
namedPosition: URIRef = rdflib.term.URIRef('https://schema.org/namedPosition')  
  
nationality: URIRef = rdflib.term.URIRef('https://schema.org/nationality')  
  
naturalProgression: URIRef =  
rdflib.term.URIRef('https://schema.org/naturalProgression')  
  
negativeNotes: URIRef = rdflib.term.URIRef('https://schema.org/negativeNotes')  
  
nerve: URIRef = rdflib.term.URIRef('https://schema.org/nerve')  
  
nerveMotor: URIRef = rdflib.term.URIRef('https://schema.org/nerveMotor')  
  
netWorth: URIRef = rdflib.term.URIRef('https://schema.org/netWorth')  
  
newsUpdatesAndGuidelines: URIRef =  
rdflib.term.URIRef('https://schema.org/newsUpdatesAndGuidelines')  
  
nextItem: URIRef = rdflib.term.URIRef('https://schema.org/nextItem')  
  
noBylinesPolicy: URIRef = rdflib.term.URIRef('https://schema.org/noBylinesPolicy')  
  
nonEqual: URIRef = rdflib.term.URIRef('https://schema.org/nonEqual')  
  
nonProprietaryName: URIRef =  
rdflib.term.URIRef('https://schema.org/nonProprietaryName')  
  
nonprofitStatus: URIRef = rdflib.term.URIRef('https://schema.org/nonprofitStatus')  
  
normalRange: URIRef = rdflib.term.URIRef('https://schema.org/normalRange')  
  
nsn: URIRef = rdflib.term.URIRef('https://schema.org/nsn')  
  
numAdults: URIRef = rdflib.term.URIRef('https://schema.org/numAdults')  
  
numChildren: URIRef = rdflib.term.URIRef('https://schema.org/numChildren')  
  
numConstraints: URIRef = rdflib.term.URIRef('https://schema.org/numConstraints')  
  
numTracks: URIRef = rdflib.term.URIRef('https://schema.org/numTracks')  
  
numberOfAccommodationUnits: URIRef =  
rdflib.term.URIRef('https://schema.org/numberOfAccommodationUnits')
```

```
numberOfAirbags: URIRef = rdflib.term.URIRef('https://schema.org/numberOfAirbags')

numberOfAvailableAccommodationUnits: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfAvailableAccommodationUnits')

numberOfAxles: URIRef = rdflib.term.URIRef('https://schema.org/numberOfAxles')

numberOfBathroomsTotal: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfBathroomsTotal')

numberOfBedrooms: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfBedrooms')

numberOfBeds: URIRef = rdflib.term.URIRef('https://schema.org/numberOfBeds')

numberOfCredits: URIRef = rdflib.term.URIRef('https://schema.org/numberOfCredits')

numberOfDoors: URIRef = rdflib.term.URIRef('https://schema.org/numberOfDoors')

numberOfEmployees: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfEmployees')

numberOfEpisodes: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfEpisodes')

numberOfForwardGears: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfForwardGears')

numberOfFullBathrooms: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfFullBathrooms')

numberOfItems: URIRef = rdflib.term.URIRef('https://schema.org/numberOfItems')

numberOfLoanPayments: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfLoanPayments')

numberOfPages: URIRef = rdflib.term.URIRef('https://schema.org/numberOfPages')

numberOfPartialBathrooms: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfPartialBathrooms')

numberOfPlayers: URIRef = rdflib.term.URIRef('https://schema.org/numberOfPlayers')

numberOfPreviousOwners: URIRef =
rdflib.term.URIRef('https://schema.org/numberOfPreviousOwners')

numberOfRooms: URIRef = rdflib.term.URIRef('https://schema.org/numberOfRooms')

numberOfSeasons: URIRef = rdflib.term.URIRef('https://schema.org/numberOfSeasons')

numberedPosition: URIRef =
rdflib.term.URIRef('https://schema.org/numberedPosition')

nutrition: URIRef = rdflib.term.URIRef('https://schema.org/nutrition')

object: URIRef = rdflib.term.URIRef('https://schema.org/object')

observationDate: URIRef = rdflib.term.URIRef('https://schema.org/observationDate')
```

```
observedNode: URIRef = rdflib.term.URIRef('https://schema.org/observedNode')

occupancy: URIRef = rdflib.term.URIRef('https://schema.org/occupancy')

occupationLocation: URIRef =
rdflib.term.URIRef('https://schema.org/occupationLocation')

occupationalCategory: URIRef =
rdflib.term.URIRef('https://schema.org/occupationalCategory')

occupationalCredentialAwarded: URIRef =
rdflib.term.URIRef('https://schema.org/occupationalCredentialAwarded')

offerCount: URIRef = rdflib.term.URIRef('https://schema.org/offerCount')

offeredBy: URIRef = rdflib.term.URIRef('https://schema.org/offeredBy')

offers: URIRef = rdflib.term.URIRef('https://schema.org/offers')

offersPrescriptionByMail: URIRef =
rdflib.term.URIRef('https://schema.org/offersPrescriptionByMail')

openingHours: URIRef = rdflib.term.URIRef('https://schema.org/openingHours')

openingHoursSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/openingHoursSpecification')

opens: URIRef = rdflib.term.URIRef('https://schema.org/opens')

operatingSystem: URIRef = rdflib.term.URIRef('https://schema.org/operatingSystem')

opponent: URIRef = rdflib.term.URIRef('https://schema.org/opponent')

option: URIRef = rdflib.term.URIRef('https://schema.org/option')

orderDate: URIRef = rdflib.term.URIRef('https://schema.org/orderDate')

orderDelivery: URIRef = rdflib.term.URIRef('https://schema.org/orderDelivery')

orderItemNumber: URIRef = rdflib.term.URIRef('https://schema.org/orderItemNumber')

orderItemStatus: URIRef = rdflib.term.URIRef('https://schema.org/orderItemStatus')

orderNumber: URIRef = rdflib.term.URIRef('https://schema.org/orderNumber')

orderQuantity: URIRef = rdflib.term.URIRef('https://schema.org/orderQuantity')

orderStatus: URIRef = rdflib.term.URIRef('https://schema.org/orderStatus')

orderedItem: URIRef = rdflib.term.URIRef('https://schema.org/orderedItem')

organizer: URIRef = rdflib.term.URIRef('https://schema.org/organizer')

originAddress: URIRef = rdflib.term.URIRef('https://schema.org/originAddress')

originalMediaContextDescription: URIRef =
rdflib.term.URIRef('https://schema.org/originalMediaContextDescription')
```

```
originalMediaLink: URIRef =  
rdflib.term.URIRef('https://schema.org/originalMediaLink')  
  
originatesFrom: URIRef = rdflib.term.URIRef('https://schema.org/originatesFrom')  
  
overdosage: URIRef = rdflib.term.URIRef('https://schema.org/overdosage')  
  
ownedFrom: URIRef = rdflib.term.URIRef('https://schema.org/ownedFrom')  
  
ownedThrough: URIRef = rdflib.term.URIRef('https://schema.org/ownedThrough')  
  
ownershipFundingInfo: URIRef =  
rdflib.term.URIRef('https://schema.org/ownershipFundingInfo')  
  
owns: URIRef = rdflib.term.URIRef('https://schema.org/owns')  
  
pageEnd: URIRef = rdflib.term.URIRef('https://schema.org/pageEnd')  
  
pageStart: URIRef = rdflib.term.URIRef('https://schema.org/pageStart')  
  
pagination: URIRef = rdflib.term.URIRef('https://schema.org/pagination')  
  
parent: URIRef = rdflib.term.URIRef('https://schema.org/parent')  
  
parentItem: URIRef = rdflib.term.URIRef('https://schema.org/parentItem')  
  
parentOrganization: URIRef =  
rdflib.term.URIRef('https://schema.org/parentOrganization')  
  
parentService: URIRef = rdflib.term.URIRef('https://schema.org/parentService')  
  
parentTaxon: URIRef = rdflib.term.URIRef('https://schema.org/parentTaxon')  
  
parents: URIRef = rdflib.term.URIRef('https://schema.org/parents')  
  
partOfEpisode: URIRef = rdflib.term.URIRef('https://schema.org/partOfEpisode')  
  
partOfInvoice: URIRef = rdflib.term.URIRef('https://schema.org/partOfInvoice')  
  
partOfOrder: URIRef = rdflib.term.URIRef('https://schema.org/partOfOrder')  
  
partOfSeason: URIRef = rdflib.term.URIRef('https://schema.org/partOfSeason')  
  
partOfSeries: URIRef = rdflib.term.URIRef('https://schema.org/partOfSeries')  
  
partOfSystem: URIRef = rdflib.term.URIRef('https://schema.org/partOfSystem')  
  
partOfTVSeries: URIRef = rdflib.term.URIRef('https://schema.org/partOfTVSeries')  
  
partOfTrip: URIRef = rdflib.term.URIRef('https://schema.org/partOfTrip')  
  
participant: URIRef = rdflib.term.URIRef('https://schema.org/participant')  
  
partySize: URIRef = rdflib.term.URIRef('https://schema.org/partySize')  
  
passengerPriorityStatus: URIRef =  
rdflib.term.URIRef('https://schema.org/passengerPriorityStatus')  
  
passengerSequenceNumber: URIRef =  
rdflib.term.URIRef('https://schema.org/passengerSequenceNumber')
```

```
pathophysiology: URIRef = rdflib.term.URIRef('https://schema.org/pathophysiology')
pattern: URIRef = rdflib.term.URIRef('https://schema.org/pattern')
payload: URIRef = rdflib.term.URIRef('https://schema.org/payload')
paymentAccepted: URIRef = rdflib.term.URIRef('https://schema.org/paymentAccepted')
paymentDue: URIRef = rdflib.term.URIRef('https://schema.org/paymentDue')
paymentDueDate: URIRef = rdflib.term.URIRef('https://schema.org/paymentDueDate')
paymentMethod: URIRef = rdflib.term.URIRef('https://schema.org/paymentMethod')
paymentMethodId: URIRef = rdflib.term.URIRef('https://schema.org/paymentMethodId')
paymentStatus: URIRef = rdflib.term.URIRef('https://schema.org/paymentStatus')
paymentUrl: URIRef = rdflib.term.URIRef('https://schema.org/paymentUrl')
penciler: URIRef = rdflib.term.URIRef('https://schema.org/penciler')
percentile10: URIRef = rdflib.term.URIRef('https://schema.org/percentile10')
percentile25: URIRef = rdflib.term.URIRef('https://schema.org/percentile25')
percentile75: URIRef = rdflib.term.URIRef('https://schema.org/percentile75')
percentile90: URIRef = rdflib.term.URIRef('https://schema.org/percentile90')
performTime: URIRef = rdflib.term.URIRef('https://schema.org/performTime')
performer: URIRef = rdflib.term.URIRef('https://schema.org/performer')
performerIn: URIRef = rdflib.term.URIRef('https://schema.org/performerIn')
performers: URIRef = rdflib.term.URIRef('https://schema.org/performers')
permissionType: URIRef = rdflib.term.URIRef('https://schema.org/permissionType')
permissions: URIRef = rdflib.term.URIRef('https://schema.org/permissions')
permitAudience: URIRef = rdflib.term.URIRef('https://schema.org/permitAudience')
permittedUsage: URIRef = rdflib.term.URIRef('https://schema.org/permittedUsage')
petsAllowed: URIRef = rdflib.term.URIRef('https://schema.org/petsAllowed')
phoneticText: URIRef = rdflib.term.URIRef('https://schema.org/phoneticText')
photo: URIRef = rdflib.term.URIRef('https://schema.org/photo')
photos: URIRef = rdflib.term.URIRef('https://schema.org/photos')
physicalRequirement: URIRef =
rdflib.term.URIRef('https://schema.org/physicalRequirement')
physiologicalBenefits: URIRef =
rdflib.term.URIRef('https://schema.org/physiologicalBenefits')
```

```

pickupLocation: URIRef = rdflib.term.URIRef('https://schema.org/pickupLocation')
pickupTime: URIRef = rdflib.term.URIRef('https://schema.org/pickupTime')
playMode: URIRef = rdflib.term.URIRef('https://schema.org/playMode')
playerType: URIRef = rdflib.term.URIRef('https://schema.org/playerType')
playersOnline: URIRef = rdflib.term.URIRef('https://schema.org/playersOnline')
polygon: URIRef = rdflib.term.URIRef('https://schema.org/polygon')
populationType: URIRef = rdflib.term.URIRef('https://schema.org/populationType')
position: URIRef = rdflib.term.URIRef('https://schema.org/position')
positiveNotes: URIRef = rdflib.term.URIRef('https://schema.org/positiveNotes')
possibleComplication: URIRef =
rdflib.term.URIRef('https://schema.org/possibleComplication')
possibleTreatment: URIRef =
rdflib.term.URIRef('https://schema.org/possibleTreatment')
postOfficeBoxNumber: URIRef =
rdflib.term.URIRef('https://schema.org/postOfficeBoxNumber')
postOp: URIRef = rdflib.term.URIRef('https://schema.org/postOp')
postalCode: URIRef = rdflib.term.URIRef('https://schema.org/postalCode')
postalCodeBegin: URIRef = rdflib.term.URIRef('https://schema.org/postalCodeBegin')
postalCodeEnd: URIRef = rdflib.term.URIRef('https://schema.org/postalCodeEnd')
postalCodePrefix: URIRef =
rdflib.term.URIRef('https://schema.org/postalCodePrefix')
postalCodeRange: URIRef = rdflib.term.URIRef('https://schema.org/postalCodeRange')
potentialAction: URIRef = rdflib.term.URIRef('https://schema.org/potentialAction')
potentialUse: URIRef = rdflib.term.URIRef('https://schema.org/potentialUse')
preOp: URIRef = rdflib.term.URIRef('https://schema.org/preOp')
predecessorOf: URIRef = rdflib.term.URIRef('https://schema.org/predecessorOf')
pregnancyCategory: URIRef =
rdflib.term.URIRef('https://schema.org/pregnancyCategory')
pregnancyWarning: URIRef =
rdflib.term.URIRef('https://schema.org/pregnancyWarning')
prepTime: URIRef = rdflib.term.URIRef('https://schema.org/prepTime')
preparation: URIRef = rdflib.term.URIRef('https://schema.org/preparation')
prescribingInfo: URIRef = rdflib.term.URIRef('https://schema.org/prescribingInfo')

```



```
prescriptionStatus: URIRef =  
rdflib.term.URIRef('https://schema.org/prescriptionStatus')  
  
previousItem: URIRef = rdflib.term.URIRef('https://schema.org/previousItem')  
  
previousStartDate: URIRef =  
rdflib.term.URIRef('https://schema.org/previousStartDate')  
  
price: URIRef = rdflib.term.URIRef('https://schema.org/price')  
  
priceComponent: URIRef = rdflib.term.URIRef('https://schema.org/priceComponent')  
  
priceComponentType: URIRef =  
rdflib.term.URIRef('https://schema.org/priceComponentType')  
  
priceCurrency: URIRef = rdflib.term.URIRef('https://schema.org/priceCurrency')  
  
priceRange: URIRef = rdflib.term.URIRef('https://schema.org/priceRange')  
  
priceSpecification: URIRef =  
rdflib.term.URIRef('https://schema.org/priceSpecification')  
  
priceType: URIRef = rdflib.term.URIRef('https://schema.org/priceType')  
  
priceValidUntil: URIRef = rdflib.term.URIRef('https://schema.org/priceValidUntil')  
  
primaryImageOfPage: URIRef =  
rdflib.term.URIRef('https://schema.org/primaryImageOfPage')  
  
primaryPrevention: URIRef =  
rdflib.term.URIRef('https://schema.org/primaryPrevention')  
  
printColumn: URIRef = rdflib.term.URIRef('https://schema.org/printColumn')  
  
printEdition: URIRef = rdflib.term.URIRef('https://schema.org/printEdition')  
  
printPage: URIRef = rdflib.term.URIRef('https://schema.org/printPage')  
  
printSection: URIRef = rdflib.term.URIRef('https://schema.org/printSection')  
  
procedure: URIRef = rdflib.term.URIRef('https://schema.org/procedure')  
  
procedureType: URIRef = rdflib.term.URIRef('https://schema.org/procedureType')  
  
processingTime: URIRef = rdflib.term.URIRef('https://schema.org/processingTime')  
  
processorRequirements: URIRef =  
rdflib.term.URIRef('https://schema.org/processorRequirements')  
  
producer: URIRef = rdflib.term.URIRef('https://schema.org/producer')  
  
produces: URIRef = rdflib.term.URIRef('https://schema.org/produces')  
  
productGroupID: URIRef = rdflib.term.URIRef('https://schema.org/productGroupID')  
  
productID: URIRef = rdflib.term.URIRef('https://schema.org/productID')  
  
productSupported: URIRef =  
rdflib.term.URIRef('https://schema.org/productSupported')
```



```
productionCompany: URIRef =  
rdflib.term.URIRef('https://schema.org/productionCompany')  
  
productionDate: URIRef = rdflib.term.URIRef('https://schema.org/productionDate')  
  
proficiencyLevel: URIRef =  
rdflib.term.URIRef('https://schema.org/proficiencyLevel')  
  
programMembershipUsed: URIRef =  
rdflib.term.URIRef('https://schema.org/programMembershipUsed')  
  
programName: URIRef = rdflib.term.URIRef('https://schema.org/programName')  
  
programPrerequisites: URIRef =  
rdflib.term.URIRef('https://schema.org/programPrerequisites')  
  
programType: URIRef = rdflib.term.URIRef('https://schema.org/programType')  
  
programmingLanguage: URIRef =  
rdflib.term.URIRef('https://schema.org/programmingLanguage')  
  
programmingModel: URIRef =  
rdflib.term.URIRef('https://schema.org/programmingModel')  
  
propertyID: URIRef = rdflib.term.URIRef('https://schema.org/propertyID')  
  
proprietaryName: URIRef = rdflib.term.URIRef('https://schema.org/proprietaryName')  
  
proteinContent: URIRef = rdflib.term.URIRef('https://schema.org/proteinContent')  
  
provider: URIRef = rdflib.term.URIRef('https://schema.org/provider')  
  
providerMobility: URIRef =  
rdflib.term.URIRef('https://schema.org/providerMobility')  
  
providesBroadcastService: URIRef =  
rdflib.term.URIRef('https://schema.org/providesBroadcastService')  
  
providesService: URIRef = rdflib.term.URIRef('https://schema.org/providesService')  
  
publicAccess: URIRef = rdflib.term.URIRef('https://schema.org/publicAccess')  
  
publicTransportClosuresInfo: URIRef =  
rdflib.term.URIRef('https://schema.org/publicTransportClosuresInfo')  
  
publication: URIRef = rdflib.term.URIRef('https://schema.org/publication')  
  
publicationType: URIRef = rdflib.term.URIRef('https://schema.org/publicationType')  
  
publishedBy: URIRef = rdflib.term.URIRef('https://schema.org/publishedBy')  
  
publishedOn: URIRef = rdflib.term.URIRef('https://schema.org/publishedOn')  
  
publisher: URIRef = rdflib.term.URIRef('https://schema.org/publisher')  
  
publisherImprint: URIRef =  
rdflib.term.URIRef('https://schema.org/publisherImprint')
```

```
publishingPrinciples: URIRef =  
rdflib.term.URIRef('https://schema.org/publishingPrinciples')  
  
purchaseDate: URIRef = rdflib.term.URIRef('https://schema.org/purchaseDate')  
  
qualifications: URIRef = rdflib.term.URIRef('https://schema.org/qualifications')  
  
quarantineGuidelines: URIRef =  
rdflib.term.URIRef('https://schema.org/quarantineGuidelines')  
  
query: URIRef = rdflib.term.URIRef('https://schema.org/query')  
  
quest: URIRef = rdflib.term.URIRef('https://schema.org/quest')  
  
question: URIRef = rdflib.term.URIRef('https://schema.org/question')  
  
rangeIncludes: URIRef = rdflib.term.URIRef('https://schema.org/rangeIncludes')  
  
ratingCount: URIRef = rdflib.term.URIRef('https://schema.org/ratingCount')  
  
ratingExplanation: URIRef =  
rdflib.term.URIRef('https://schema.org/ratingExplanation')  
  
ratingValue: URIRef = rdflib.term.URIRef('https://schema.org/ratingValue')  
  
readBy: URIRef = rdflib.term.URIRef('https://schema.org/readBy')  
  
readonlyValue: URIRef = rdflib.term.URIRef('https://schema.org/readonlyValue')  
  
realEstateAgent: URIRef = rdflib.term.URIRef('https://schema.org/realEstateAgent')  
  
recipe: URIRef = rdflib.term.URIRef('https://schema.org/recipe')  
  
recipeCategory: URIRef = rdflib.term.URIRef('https://schema.org/recipeCategory')  
  
recipeCuisine: URIRef = rdflib.term.URIRef('https://schema.org/recipeCuisine')  
  
recipeIngredient: URIRef =  
rdflib.term.URIRef('https://schema.org/recipeIngredient')  
  
recipeInstructions: URIRef =  
rdflib.term.URIRef('https://schema.org/recipeInstructions')  
  
recipeYield: URIRef = rdflib.term.URIRef('https://schema.org/recipeYield')  
  
recipient: URIRef = rdflib.term.URIRef('https://schema.org/recipient')  
  
recognizedBy: URIRef = rdflib.term.URIRef('https://schema.org/recognizedBy')  
  
recognizingAuthority: URIRef =  
rdflib.term.URIRef('https://schema.org/recognizingAuthority')  
  
recommendationStrength: URIRef =  
rdflib.term.URIRef('https://schema.org/recommendationStrength')  
  
recommendedIntake: URIRef =  
rdflib.term.URIRef('https://schema.org/recommendedIntake')  
  
recordLabel: URIRef = rdflib.term.URIRef('https://schema.org/recordLabel')
```

```
recordedAs: URIRef = rdflib.term.URIRef('https://schema.org/recordedAs')
recordedAt: URIRef = rdflib.term.URIRef('https://schema.org/recordedAt')
recordedIn: URIRef = rdflib.term.URIRef('https://schema.org/recordedIn')
recordingOf: URIRef = rdflib.term.URIRef('https://schema.org/recordingOf')
recourseLoan: URIRef = rdflib.term.URIRef('https://schema.org/recourseLoan')
referenceQuantity: URIRef =
rdflib.term.URIRef('https://schema.org/referenceQuantity')
referencesOrder: URIRef = rdflib.term.URIRef('https://schema.org/referencesOrder')
refundType: URIRef = rdflib.term.URIRef('https://schema.org/refundType')
regionDrained: URIRef = rdflib.term.URIRef('https://schema.org/regionDrained')
regionsAllowed: URIRef = rdflib.term.URIRef('https://schema.org/regionsAllowed')
relatedAnatomy: URIRef = rdflib.term.URIRef('https://schema.org/relatedAnatomy')
relatedCondition: URIRef =
rdflib.term.URIRef('https://schema.org/relatedCondition')
relatedDrug: URIRef = rdflib.term.URIRef('https://schema.org/relatedDrug')
relatedLink: URIRef = rdflib.term.URIRef('https://schema.org/relatedLink')
relatedStructure: URIRef =
rdflib.term.URIRef('https://schema.org/relatedStructure')
relatedTherapy: URIRef = rdflib.term.URIRef('https://schema.org/relatedTherapy')
relatedTo: URIRef = rdflib.term.URIRef('https://schema.org/relatedTo')
releaseDate: URIRef = rdflib.term.URIRef('https://schema.org/releaseDate')
releaseNotes: URIRef = rdflib.term.URIRef('https://schema.org/releaseNotes')
releaseOf: URIRef = rdflib.term.URIRef('https://schema.org/releaseOf')
releasedEvent: URIRef = rdflib.term.URIRef('https://schema.org/releasedEvent')
relevantOccupation: URIRef =
rdflib.term.URIRef('https://schema.org/relevantOccupation')
relevantSpecialty: URIRef =
rdflib.term.URIRef('https://schema.org/relevantSpecialty')
remainingAttendeeCapacity: URIRef =
rdflib.term.URIRef('https://schema.org/remainingAttendeeCapacity')
renegotiableLoan: URIRef =
rdflib.term.URIRef('https://schema.org/renegotiableLoan')
repeatCount: URIRef = rdflib.term.URIRef('https://schema.org/repeatCount')
```

```
repeatFrequency: URIRef = rdflib.term.URIRef('https://schema.org/repeatFrequency')
repetitions: URIRef = rdflib.term.URIRef('https://schema.org/repetitions')
replacee: URIRef = rdflib.term.URIRef('https://schema.org/replacee')
replacer: URIRef = rdflib.term.URIRef('https://schema.org/replacer')
replyToUrl: URIRef = rdflib.term.URIRef('https://schema.org/replyToUrl')
reportNumber: URIRef = rdflib.term.URIRef('https://schema.org/reportNumber')
representativeOfPage: URIRef =
rdflib.term.URIRef('https://schema.org/representativeOfPage')
requiredCollateral: URIRef =
rdflib.term.URIRef('https://schema.org/requiredCollateral')
requiredGender: URIRef = rdflib.term.URIRef('https://schema.org/requiredGender')
requiredMaxAge: URIRef = rdflib.term.URIRef('https://schema.org/requiredMaxAge')
requiredMinAge: URIRef = rdflib.term.URIRef('https://schema.org/requiredMinAge')
requiredQuantity: URIRef =
rdflib.term.URIRef('https://schema.org/requiredQuantity')
requirements: URIRef = rdflib.term.URIRef('https://schema.org/requirements')
requiresSubscription: URIRef =
rdflib.term.URIRef('https://schema.org/requiresSubscription')
reservationFor: URIRef = rdflib.term.URIRef('https://schema.org/reservationFor')
reservationId: URIRef = rdflib.term.URIRef('https://schema.org/reservationId')
reservationStatus: URIRef =
rdflib.term.URIRef('https://schema.org/reservationStatus')
reservedTicket: URIRef = rdflib.term.URIRef('https://schema.org/reservedTicket')
responsibilities: URIRef =
rdflib.term.URIRef('https://schema.org/responsibilities')
restPeriods: URIRef = rdflib.term.URIRef('https://schema.org/restPeriods')
restockingFee: URIRef = rdflib.term.URIRef('https://schema.org/restockingFee')
result: URIRef = rdflib.term.URIRef('https://schema.org/result')
resultComment: URIRef = rdflib.term.URIRef('https://schema.org/resultComment')
resultReview: URIRef = rdflib.term.URIRef('https://schema.org/resultReview')
returnFees: URIRef = rdflib.term.URIRef('https://schema.org/returnFees')
returnLabelSource: URIRef =
rdflib.term.URIRef('https://schema.org/returnLabelSource')
```

```
returnMethod: URIRef = rdflib.term.URIRef('https://schema.org/returnMethod')

returnPolicyCategory: URIRef =
rdflib.term.URIRef('https://schema.org/returnPolicyCategory')

returnPolicyCountry: URIRef =
rdflib.term.URIRef('https://schema.org/returnPolicyCountry')

returnPolicySeasonalOverride: URIRef =
rdflib.term.URIRef('https://schema.org/returnPolicySeasonalOverride')

returnShippingFeesAmount: URIRef =
rdflib.term.URIRef('https://schema.org/returnShippingFeesAmount')

review: URIRef = rdflib.term.URIRef('https://schema.org/review')

reviewAspect: URIRef = rdflib.term.URIRef('https://schema.org/reviewAspect')

reviewBody: URIRef = rdflib.term.URIRef('https://schema.org/reviewBody')

reviewCount: URIRef = rdflib.term.URIRef('https://schema.org/reviewCount')

reviewRating: URIRef = rdflib.term.URIRef('https://schema.org/reviewRating')

reviewedBy: URIRef = rdflib.term.URIRef('https://schema.org/reviewedBy')

reviews: URIRef = rdflib.term.URIRef('https://schema.org/reviews')

riskFactor: URIRef = rdflib.term.URIRef('https://schema.org/riskFactor')

risks: URIRef = rdflib.term.URIRef('https://schema.org/risks')

roleName: URIRef = rdflib.term.URIRef('https://schema.org/roleName')

roofLoad: URIRef = rdflib.term.URIRef('https://schema.org/roofLoad')

rsvpResponse: URIRef = rdflib.term.URIRef('https://schema.org/rsvpResponse')

runsTo: URIRef = rdflib.term.URIRef('https://schema.org/runsTo')

runtime: URIRef = rdflib.term.URIRef('https://schema.org/runtime')

runtimePlatform: URIRef = rdflib.term.URIRef('https://schema.org/runtimePlatform')

rxcul: URIRef = rdflib.term.URIRef('https://schema.org/rxcui')

safetyConsideration: URIRef =
rdflib.term.URIRef('https://schema.org/safetyConsideration')

salaryCurrency: URIRef = rdflib.term.URIRef('https://schema.org/salaryCurrency')

salaryUponCompletion: URIRef =
rdflib.term.URIRef('https://schema.org/salaryUponCompletion')

sameAs: URIRef = rdflib.term.URIRef('https://schema.org/sameAs')

sampleType: URIRef = rdflib.term.URIRef('https://schema.org/sampleType')
```

```
saturatedFatContent: URIRef =  
rdflib.term.URIRef('https://schema.org/saturatedFatContent')  
  
scheduleTimezone: URIRef =  
rdflib.term.URIRef('https://schema.org/scheduleTimezone')  
  
scheduledPaymentDate: URIRef =  
rdflib.term.URIRef('https://schema.org/scheduledPaymentDate')  
  
scheduledTime: URIRef = rdflib.term.URIRef('https://schema.org/scheduledTime')  
  
schemaVersion: URIRef = rdflib.term.URIRef('https://schema.org/schemaVersion')  
  
schoolClosuresInfo: URIRef =  
rdflib.term.URIRef('https://schema.org/schoolClosuresInfo')  
  
screenCount: URIRef = rdflib.term.URIRef('https://schema.org/screenCount')  
  
screenshot: URIRef = rdflib.term.URIRef('https://schema.org/screenshot')  
  
sdDatePublished: URIRef = rdflib.term.URIRef('https://schema.org/sdDatePublished')  
  
sdLicense: URIRef = rdflib.term.URIRef('https://schema.org/sdLicense')  
  
sdPublisher: URIRef = rdflib.term.URIRef('https://schema.org/sdPublisher')  
  
season: URIRef = rdflib.term.URIRef('https://schema.org/season')  
  
seasonNumber: URIRef = rdflib.term.URIRef('https://schema.org/seasonNumber')  
  
seasons: URIRef = rdflib.term.URIRef('https://schema.org/seasons')  
  
seatNumber: URIRef = rdflib.term.URIRef('https://schema.org/seatNumber')  
  
seatRow: URIRef = rdflib.term.URIRef('https://schema.org/seatRow')  
  
seatSection: URIRef = rdflib.term.URIRef('https://schema.org/seatSection')  
  
seatingCapacity: URIRef = rdflib.term.URIRef('https://schema.org/seatingCapacity')  
  
seatingType: URIRef = rdflib.term.URIRef('https://schema.org/seatingType')  
  
secondaryPrevention: URIRef =  
rdflib.term.URIRef('https://schema.org/secondaryPrevention')  
  
securityClearanceRequirement: URIRef =  
rdflib.term.URIRef('https://schema.org/securityClearanceRequirement')  
  
securityScreening: URIRef =  
rdflib.term.URIRef('https://schema.org/securityScreening')  
  
seeks: URIRef = rdflib.term.URIRef('https://schema.org/seeks')  
  
seller: URIRef = rdflib.term.URIRef('https://schema.org/seller')  
  
sender: URIRef = rdflib.term.URIRef('https://schema.org/sender')  
  
sensoryRequirement: URIRef =  
rdflib.term.URIRef('https://schema.org/sensoryRequirement')
```

```
sensoryUnit: URIRef = rdflib.term.URIRef('https://schema.org/sensoryUnit')
serialNumber: URIRef = rdflib.term.URIRef('https://schema.org/serialNumber')
seriousAdverseOutcome: URIRef =
rdflib.term.URIRef('https://schema.org/seriousAdverseOutcome')
serverStatus: URIRef = rdflib.term.URIRef('https://schema.org/serverStatus')
servesCuisine: URIRef = rdflib.term.URIRef('https://schema.org/servesCuisine')
serviceArea: URIRef = rdflib.term.URIRef('https://schema.org/serviceArea')
serviceAudience: URIRef = rdflib.term.URIRef('https://schema.org/serviceAudience')
serviceLocation: URIRef = rdflib.term.URIRef('https://schema.org/serviceLocation')
serviceOperator: URIRef = rdflib.term.URIRef('https://schema.org/serviceOperator')
serviceOutput: URIRef = rdflib.term.URIRef('https://schema.org/serviceOutput')
servicePhone: URIRef = rdflib.term.URIRef('https://schema.org/servicePhone')
servicePostalAddress: URIRef =
rdflib.term.URIRef('https://schema.org/servicePostalAddress')
serviceSmsNumber: URIRef =
rdflib.term.URIRef('https://schema.org/serviceSmsNumber')
serviceType: URIRef = rdflib.term.URIRef('https://schema.org/serviceType')
serviceUrl: URIRef = rdflib.term.URIRef('https://schema.org/serviceUrl')
servingSize: URIRef = rdflib.term.URIRef('https://schema.org/servingSize')
sha256: URIRef = rdflib.term.URIRef('https://schema.org/sha256')
sharedContent: URIRef = rdflib.term.URIRef('https://schema.org/sharedContent')
shippingDestination: URIRef =
rdflib.term.URIRef('https://schema.org/shippingDestination')
shippingDetails: URIRef = rdflib.term.URIRef('https://schema.org/shippingDetails')
shippingLabel: URIRef = rdflib.term.URIRef('https://schema.org/shippingLabel')
shippingRate: URIRef = rdflib.term.URIRef('https://schema.org/shippingRate')
shippingSettingsLink: URIRef =
rdflib.term.URIRef('https://schema.org/shippingSettingsLink')
sibling: URIRef = rdflib.term.URIRef('https://schema.org/sibling')
siblings: URIRef = rdflib.term.URIRef('https://schema.org/siblings')
signDetected: URIRef = rdflib.term.URIRef('https://schema.org/signDetected')
signOrSymptom: URIRef = rdflib.term.URIRef('https://schema.org/signOrSymptom')
significance: URIRef = rdflib.term.URIRef('https://schema.org/significance')
```



```
significantLink: URIRef = rdflib.term.URIRef('https://schema.org/significantLink')
significantLinks: URIRef =
rdflib.term.URIRef('https://schema.org/significantLinks')

size: URIRef = rdflib.term.URIRef('https://schema.org/size')

sizeGroup: URIRef = rdflib.term.URIRef('https://schema.org/sizeGroup')

sizeSystem: URIRef = rdflib.term.URIRef('https://schema.org/sizeSystem')

skills: URIRef = rdflib.term.URIRef('https://schema.org/skills')

sku: URIRef = rdflib.term.URIRef('https://schema.org/sku')

slogan: URIRef = rdflib.term.URIRef('https://schema.org/slogan')

smiles: URIRef = rdflib.term.URIRef('https://schema.org/smiles')

smokingAllowed: URIRef = rdflib.term.URIRef('https://schema.org/smokingAllowed')

sodiumContent: URIRef = rdflib.term.URIRef('https://schema.org/sodiumContent')

softwareAddOn: URIRef = rdflib.term.URIRef('https://schema.org/softwareAddOn')

softwareHelp: URIRef = rdflib.term.URIRef('https://schema.org/softwareHelp')

softwareRequirements: URIRef =
rdflib.term.URIRef('https://schema.org/softwareRequirements')

softwareVersion: URIRef = rdflib.term.URIRef('https://schema.org/softwareVersion')

sourceOrganization: URIRef =
rdflib.term.URIRef('https://schema.org/sourceOrganization')

sourcedFrom: URIRef = rdflib.term.URIRef('https://schema.org/sourcedFrom')

spatial: URIRef = rdflib.term.URIRef('https://schema.org/spatial')

spatialCoverage: URIRef = rdflib.term.URIRef('https://schema.org/spatialCoverage')

speakable: URIRef = rdflib.term.URIRef('https://schema.org/speakable')

specialCommitments: URIRef =
rdflib.term.URIRef('https://schema.org/specialCommitments')

specialOpeningHoursSpecification: URIRef =
rdflib.term.URIRef('https://schema.org/specialOpeningHoursSpecification')

specialty: URIRef = rdflib.term.URIRef('https://schema.org/specialty')

speechToTextMarkup: URIRef =
rdflib.term.URIRef('https://schema.org/speechToTextMarkup')

speed: URIRef = rdflib.term.URIRef('https://schema.org/speed')

spokenByCharacter: URIRef =
rdflib.term.URIRef('https://schema.org/spokenByCharacter')
```



```
sponsor: URIRef = rdflib.term.URIRef('https://schema.org/sponsor')
sport: URIRef = rdflib.term.URIRef('https://schema.org/sport')
sportsActivityLocation: URIRef =
rdflib.term.URIRef('https://schema.org/sportsActivityLocation')
sportsEvent: URIRef = rdflib.term.URIRef('https://schema.org/sportsEvent')
sportsTeam: URIRef = rdflib.term.URIRef('https://schema.org/sportsTeam')
spouse: URIRef = rdflib.term.URIRef('https://schema.org/spouse')
stage: URIRef = rdflib.term.URIRef('https://schema.org/stage')
stageAsNumber: URIRef = rdflib.term.URIRef('https://schema.org/stageAsNumber')
starRating: URIRef = rdflib.term.URIRef('https://schema.org/starRating')
startDate: URIRef = rdflib.term.URIRef('https://schema.org/startDate')
startOffset: URIRef = rdflib.term.URIRef('https://schema.org/startOffset')
startTime: URIRef = rdflib.term.URIRef('https://schema.org/startTime')
status: URIRef = rdflib.term.URIRef('https://schema.org/status')
steeringPosition: URIRef =
rdflib.term.URIRef('https://schema.org/steeringPosition')
step: URIRef = rdflib.term.URIRef('https://schema.org/step')
stepValue: URIRef = rdflib.term.URIRef('https://schema.org/stepValue')
steps: URIRef = rdflib.term.URIRef('https://schema.org/steps')
storageRequirements: URIRef =
rdflib.term.URIRef('https://schema.org/storageRequirements')
streetAddress: URIRef = rdflib.term.URIRef('https://schema.org/streetAddress')
strengthUnit: URIRef = rdflib.term.URIRef('https://schema.org/strengthUnit')
strengthValue: URIRef = rdflib.term.URIRef('https://schema.org/strengthValue')
structuralClass: URIRef = rdflib.term.URIRef('https://schema.org/structuralClass')
study: URIRef = rdflib.term.URIRef('https://schema.org/study')
studyDesign: URIRef = rdflib.term.URIRef('https://schema.org/studyDesign')
studyLocation: URIRef = rdflib.term.URIRef('https://schema.org/studyLocation')
studySubject: URIRef = rdflib.term.URIRef('https://schema.org/studySubject')
subEvent: URIRef = rdflib.term.URIRef('https://schema.org/subEvent')
subEvents: URIRef = rdflib.term.URIRef('https://schema.org/subEvents')
subOrganization: URIRef = rdflib.term.URIRef('https://schema.org/subOrganization')
```

```
subReservation: URIRef = rdflib.term.URIRef('https://schema.org/subReservation')
subStageSuffix: URIRef = rdflib.term.URIRef('https://schema.org/subStageSuffix')
subStructure: URIRef = rdflib.term.URIRef('https://schema.org/subStructure')
subTest: URIRef = rdflib.term.URIRef('https://schema.org/subTest')
subTrip: URIRef = rdflib.term.URIRef('https://schema.org/subTrip')
subjectOf: URIRef = rdflib.term.URIRef('https://schema.org/subjectOf')
subtitleLanguage: URIRef =
rdflib.term.URIRef('https://schema.org/subtitleLanguage')
successorOf: URIRef = rdflib.term.URIRef('https://schema.org/successorOf')
sugarContent: URIRef = rdflib.term.URIRef('https://schema.org/sugarContent')
suggestedAge: URIRef = rdflib.term.URIRef('https://schema.org/suggestedAge')
suggestedAnswer: URIRef = rdflib.term.URIRef('https://schema.org/suggestedAnswer')
suggestedGender: URIRef = rdflib.term.URIRef('https://schema.org/suggestedGender')
suggestedMaxAge: URIRef = rdflib.term.URIRef('https://schema.org/suggestedMaxAge')
suggestedMeasurement: URIRef =
rdflib.term.URIRef('https://schema.org/suggestedMeasurement')
suggestedMinAge: URIRef = rdflib.term.URIRef('https://schema.org/suggestedMinAge')
suitableForDiet: URIRef = rdflib.term.URIRef('https://schema.org/suitableForDiet')
superEvent: URIRef = rdflib.term.URIRef('https://schema.org/superEvent')
supersededBy: URIRef = rdflib.term.URIRef('https://schema.org/supersededBy')
supply: URIRef = rdflib.term.URIRef('https://schema.org/supply')
supplyTo: URIRef = rdflib.term.URIRef('https://schema.org/supplyTo')
supportingData: URIRef = rdflib.term.URIRef('https://schema.org/supportingData')
surface: URIRef = rdflib.term.URIRef('https://schema.org/surface')
target: URIRef = rdflib.term.URIRef('https://schema.org/target')
targetCollection: URIRef =
rdflib.term.URIRef('https://schema.org/targetCollection')
targetDescription: URIRef =
rdflib.term.URIRef('https://schema.org/targetDescription')
targetName: URIRef = rdflib.term.URIRef('https://schema.org/targetName')
targetPlatform: URIRef = rdflib.term.URIRef('https://schema.org/targetPlatform')
targetPopulation: URIRef =
rdflib.term.URIRef('https://schema.org/targetPopulation')
```

```

targetProduct: URIRef = rdflib.term.URIRef('https://schema.org/targetProduct')
targetUrl: URIRef = rdflib.term.URIRef('https://schema.org/targetUrl')
taxID: URIRef = rdflib.term.URIRef('https://schema.org/taxID')
taxonRank: URIRef = rdflib.term.URIRef('https://schema.org/taxonRank')
taxonomicRange: URIRef = rdflib.term.URIRef('https://schema.org/taxonomicRange')
teaches: URIRef = rdflib.term.URIRef('https://schema.org/teaches')
telephone: URIRef = rdflib.term.URIRef('https://schema.org/telephone')
temporal: URIRef = rdflib.term.URIRef('https://schema.org/temporal')
temporalCoverage: URIRef =
rdflib.term.URIRef('https://schema.org/temporalCoverage')
termCode: URIRef = rdflib.term.URIRef('https://schema.org/termCode')
termDuration: URIRef = rdflib.term.URIRef('https://schema.org/termDuration')
termsOfService: URIRef = rdflib.term.URIRef('https://schema.org/termsOfService')
termsPerYear: URIRef = rdflib.term.URIRef('https://schema.org/termsPerYear')
text: URIRef = rdflib.term.URIRef('https://schema.org/text')
textValue: URIRef = rdflib.term.URIRef('https://schema.org/textValue')
thumbnail: URIRef = rdflib.term.URIRef('https://schema.org/thumbnail')
thumbnailUrl: URIRef = rdflib.term.URIRef('https://schema.org/thumbnailUrl')
tickerSymbol: URIRef = rdflib.term.URIRef('https://schema.org/tickerSymbol')
ticketNumber: URIRef = rdflib.term.URIRef('https://schema.org/ticketNumber')
ticketToken: URIRef = rdflib.term.URIRef('https://schema.org/ticketToken')
ticketedSeat: URIRef = rdflib.term.URIRef('https://schema.org/ticketedSeat')
timeOfDay: URIRef = rdflib.term.URIRef('https://schema.org/timeOfDay')
timeRequired: URIRef = rdflib.term.URIRef('https://schema.org/timeRequired')
timeToComplete: URIRef = rdflib.term.URIRef('https://schema.org/timeToComplete')
tissueSample: URIRef = rdflib.term.URIRef('https://schema.org/tissueSample')
title: URIRef = rdflib.term.URIRef('https://schema.org/title')
titleEIDR: URIRef = rdflib.term.URIRef('https://schema.org/titleEIDR')
toLocation: URIRef = rdflib.term.URIRef('https://schema.org/toLocation')
toRecipient: URIRef = rdflib.term.URIRef('https://schema.org/toRecipient')
tocContinuation: URIRef = rdflib.term.URIRef('https://schema.org/tocContinuation')

```

```
tocEntry: URIRef = rdflib.term.URIRef('https://schema.org/tocEntry')
tongueWeight: URIRef = rdflib.term.URIRef('https://schema.org/tongueWeight')
tool: URIRef = rdflib.term.URIRef('https://schema.org/tool')
torque: URIRef = rdflib.term.URIRef('https://schema.org/torque')
totalJobOpenings: URIRef =
rdflib.term.URIRef('https://schema.org/totalJobOpenings')
totalPaymentDue: URIRef = rdflib.term.URIRef('https://schema.org/totalPaymentDue')
totalPrice: URIRef = rdflib.term.URIRef('https://schema.org/totalPrice')
totalTime: URIRef = rdflib.term.URIRef('https://schema.org/totalTime')
tourBookingPage: URIRef = rdflib.term.URIRef('https://schema.org/tourBookingPage')
touristType: URIRef = rdflib.term.URIRef('https://schema.org/touristType')
track: URIRef = rdflib.term.URIRef('https://schema.org/track')
trackingNumber: URIRef = rdflib.term.URIRef('https://schema.org/trackingNumber')
trackingUrl: URIRef = rdflib.term.URIRef('https://schema.org/trackingUrl')
tracks: URIRef = rdflib.term.URIRef('https://schema.org/tracks')
trailer: URIRef = rdflib.term.URIRef('https://schema.org/trailer')
trailerWeight: URIRef = rdflib.term.URIRef('https://schema.org/trailerWeight')
trainName: URIRef = rdflib.term.URIRef('https://schema.org/trainName')
trainNumber: URIRef = rdflib.term.URIRef('https://schema.org/trainNumber')
trainingSalary: URIRef = rdflib.term.URIRef('https://schema.org/trainingSalary')
transFatContent: URIRef = rdflib.term.URIRef('https://schema.org/transFatContent')
transcript: URIRef = rdflib.term.URIRef('https://schema.org/transcript')
transitTime: URIRef = rdflib.term.URIRef('https://schema.org/transitTime')
transitTimeLabel: URIRef =
rdflib.term.URIRef('https://schema.org/transitTimeLabel')
translationOfWork: URIRef =
rdflib.term.URIRef('https://schema.org/translationOfWork')
translator: URIRef = rdflib.term.URIRef('https://schema.org/translator')
transmissionMethod: URIRef =
rdflib.term.URIRef('https://schema.org/transmissionMethod')
travelBans: URIRef = rdflib.term.URIRef('https://schema.org/travelBans')
trialDesign: URIRef = rdflib.term.URIRef('https://schema.org/trialDesign')
```

```

tributary: URIRef = rdflib.term.URIRef('https://schema.org/tributary')
typeOfBed: URIRef = rdflib.term.URIRef('https://schema.org/typeOfBed')
typeOfGood: URIRef = rdflib.term.URIRef('https://schema.org/typeOfGood')
typicalAgeRange: URIRef = rdflib.term.URIRef('https://schema.org/typicalAgeRange')
typicalCreditsPerTerm: URIRef =
rdflib.term.URIRef('https://schema.org/typicalCreditsPerTerm')
typicalTest: URIRef = rdflib.term.URIRef('https://schema.org/typicalTest')
underName: URIRef = rdflib.term.URIRef('https://schema.org/underName')
unitCode: URIRef = rdflib.term.URIRef('https://schema.org/unitCode')
unitText: URIRef = rdflib.term.URIRef('https://schema.org/unitText')
unnamedSourcesPolicy: URIRef =
rdflib.term.URIRef('https://schema.org/unnamedSourcesPolicy')
unsaturatedFatContent: URIRef =
rdflib.term.URIRef('https://schema.org/unsaturatedFatContent')
uploadDate: URIRef = rdflib.term.URIRef('https://schema.org/uploadDate')
upvoteCount: URIRef = rdflib.term.URIRef('https://schema.org/upvoteCount')
url: URIRef = rdflib.term.URIRef('https://schema.org/url')
urlTemplate: URIRef = rdflib.term.URIRef('https://schema.org/urlTemplate')
usageInfo: URIRef = rdflib.term.URIRef('https://schema.org/usageInfo')
usedToDiagnose: URIRef = rdflib.term.URIRef('https://schema.org/usedToDiagnose')
userInteractionCount: URIRef =
rdflib.term.URIRef('https://schema.org/userInteractionCount')
usesDevice: URIRef = rdflib.term.URIRef('https://schema.org/usesDevice')
usesHealthPlanIdStandard: URIRef =
rdflib.term.URIRef('https://schema.org/usesHealthPlanIdStandard')
utterances: URIRef = rdflib.term.URIRef('https://schema.org/utterances')
validFor: URIRef = rdflib.term.URIRef('https://schema.org/validFor')
validFrom: URIRef = rdflib.term.URIRef('https://schema.org/validFrom')
validIn: URIRef = rdflib.term.URIRef('https://schema.org/validIn')
validThrough: URIRef = rdflib.term.URIRef('https://schema.org/validThrough')
validUntil: URIRef = rdflib.term.URIRef('https://schema.org/validUntil')
value: URIRef = rdflib.term.URIRef('https://schema.org/value')

```

```
valueAddedTaxIncluded: URIRef =  
rdflib.term.URIRef('https://schema.org/valueAddedTaxIncluded')  
  
valueMaxLength: URIRef = rdflib.term.URIRef('https://schema.org/valueMaxLength')  
  
valueMinLength: URIRef = rdflib.term.URIRef('https://schema.org/valueMinLength')  
  
valueName: URIRef = rdflib.term.URIRef('https://schema.org/valueName')  
  
valuePattern: URIRef = rdflib.term.URIRef('https://schema.org/valuePattern')  
  
valueReference: URIRef = rdflib.term.URIRef('https://schema.org/valueReference')  
  
valueRequired: URIRef = rdflib.term.URIRef('https://schema.org/valueRequired')  
  
variableMeasured: URIRef =  
rdflib.term.URIRef('https://schema.org/variableMeasured')  
  
variantCover: URIRef = rdflib.term.URIRef('https://schema.org/variantCover')  
  
variesBy: URIRef = rdflib.term.URIRef('https://schema.org/variesBy')  
  
vatID: URIRef = rdflib.term.URIRef('https://schema.org/vatID')  
  
vehicleConfiguration: URIRef =  
rdflib.term.URIRef('https://schema.org/vehicleConfiguration')  
  
vehicleEngine: URIRef = rdflib.term.URIRef('https://schema.org/vehicleEngine')  
  
vehicleIdentificationNumber: URIRef =  
rdflib.term.URIRef('https://schema.org/vehicleIdentificationNumber')  
  
vehicleInteriorColor: URIRef =  
rdflib.term.URIRef('https://schema.org/vehicleInteriorColor')  
  
vehicleInteriorType: URIRef =  
rdflib.term.URIRef('https://schema.org/vehicleInteriorType')  
  
vehicleModelDate: URIRef =  
rdflib.term.URIRef('https://schema.org/vehicleModelDate')  
  
vehicleSeatingCapacity: URIRef =  
rdflib.term.URIRef('https://schema.org/vehicleSeatingCapacity')  
  
vehicleSpecialUsage: URIRef =  
rdflib.term.URIRef('https://schema.org/vehicleSpecialUsage')  
  
vehicleTransmission: URIRef =  
rdflib.term.URIRef('https://schema.org/vehicleTransmission')  
  
vendor: URIRef = rdflib.term.URIRef('https://schema.org/vendor')  
  
verificationFactCheckingPolicy: URIRef =  
rdflib.term.URIRef('https://schema.org/verificationFactCheckingPolicy')  
  
version: URIRef = rdflib.term.URIRef('https://schema.org/version')  
  
video: URIRef = rdflib.term.URIRef('https://schema.org/video')
```

```
videoFormat: URIRef = rdflib.term.URIRef('https://schema.org/videoFormat')
videoFrameSize: URIRef = rdflib.term.URIRef('https://schema.org/videoFrameSize')
videoQuality: URIRef = rdflib.term.URIRef('https://schema.org/videoQuality')
volumeNumber: URIRef = rdflib.term.URIRef('https://schema.org/volumeNumber')
warning: URIRef = rdflib.term.URIRef('https://schema.org/warning')
warranty: URIRef = rdflib.term.URIRef('https://schema.org/warranty')
warrantyPromise: URIRef = rdflib.term.URIRef('https://schema.org/warrantyPromise')
warrantyScope: URIRef = rdflib.term.URIRef('https://schema.org/warrantyScope')
webCheckinTime: URIRef = rdflib.term.URIRef('https://schema.org/webCheckinTime')
webFeed: URIRef = rdflib.term.URIRef('https://schema.org/webFeed')
weight: URIRef = rdflib.term.URIRef('https://schema.org/weight')
weightTotal: URIRef = rdflib.term.URIRef('https://schema.org/weightTotal')
wheelbase: URIRef = rdflib.term.URIRef('https://schema.org/wheelbase')
width: URIRef = rdflib.term.URIRef('https://schema.org/width')
winner: URIRef = rdflib.term.URIRef('https://schema.org/winner')
wordCount: URIRef = rdflib.term.URIRef('https://schema.org/wordCount')
workExample: URIRef = rdflib.term.URIRef('https://schema.org/workExample')
workFeatured: URIRef = rdflib.term.URIRef('https://schema.org/workFeatured')
workHours: URIRef = rdflib.term.URIRef('https://schema.org/workHours')
workLocation: URIRef = rdflib.term.URIRef('https://schema.org/workLocation')
workPerformed: URIRef = rdflib.term.URIRef('https://schema.org/workPerformed')
workPresented: URIRef = rdflib.term.URIRef('https://schema.org/workPresented')
workTranslation: URIRef = rdflib.term.URIRef('https://schema.org/workTranslation')
workload: URIRef = rdflib.term.URIRef('https://schema.org/workload')
worksFor: URIRef = rdflib.term.URIRef('https://schema.org/worksFor')
worstRating: URIRef = rdflib.term.URIRef('https://schema.org/worstRating')
xpath: URIRef = rdflib.term.URIRef('https://schema.org/xpath')
yearBuilt: URIRef = rdflib.term.URIRef('https://schema.org/yearBuilt')
yearlyRevenue: URIRef = rdflib.term.URIRef('https://schema.org/yearlyRevenue')
yearsInOperation: URIRef =
rdflib.term.URIRef('https://schema.org/yearsInOperation')
```



```
class rdflib.SH
```

```
    Bases: DefinedNamespace
```

```
    W3C Shapes Constraint Language (SHACL) Vocabulary
```

```
    This vocabulary defines terms used in SHACL, the W3C Shapes Constraint Language.
```

```
    Generated from: https://www.w3.org/ns/shacl.ttl Date: 2020-05-26 14:20:08.041103
```

```
    AbstractResult: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#AbstractResult')
```

```
    AndConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#AndConstraintComponent')
```

```
    BlankNode: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#BlankNode')
```

```
    BlankNodeOrIRI: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#BlankNodeOrIRI')
```

```
    BlankNodeOrLiteral: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#BlankNodeOrLiteral')
```

```
    ClassConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#ClassConstraintComponent')
```

```
    ClosedConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#ClosedConstraintComponent')
```

```
    ConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#ConstraintComponent')
```

```
    DatatypeConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#DatatypeConstraintComponent')
```

```
    DisjointConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#DisjointConstraintComponent')
```

```
    EqualsConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#EqualsConstraintComponent')
```

```
    ExpressionConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#ExpressionConstraintComponent')
```

```
    Function: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Function')
```

```
    HasValueConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#HasValueConstraintComponent')
```

```
    IRI: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#IRI')
```

```
    IRIOrLiteral: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#IRIOrLiteral')
```

```
    InConstraintComponent: URIRef =
```

```
    rdflib.term.URIRef('http://www.w3.org/ns/shacl#InConstraintComponent')
```

```
    Info: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Info')
```



```
JSConstraint: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#JSConstraint')  
  
JSConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#JSConstraintComponent')  
  
JSExecutable: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#JSExecutable')  
  
JSFunction: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#JSFunction')  
  
JSLibrary: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#JSLibrary')  
  
JSRule: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#JSRule')  
  
JSTarget: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#JSTarget')  
  
JSTargetType: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#JSTargetType')  
  
JSValidator: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#JSValidator')  
  
LanguageInConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#LanguageInConstraintComponent')  
  
LessThanConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#LessThanConstraintComponent')  
  
LessThanOrEqualsConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#LessThanOrEqualsConstraintComponent')  
  
Literal: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Literal')  
  
MaxCountConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#MaxCountConstraintComponent')  
  
MaxExclusiveConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#MaxExclusiveConstraintComponent')  
  
MaxInclusiveConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#MaxInclusiveConstraintComponent')  
  
MaxLengthConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#MaxLengthConstraintComponent')  
  
MinCountConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#MinCountConstraintComponent')  
  
MinExclusiveConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#MinExclusiveConstraintComponent')  
  
MinInclusiveConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#MinInclusiveConstraintComponent')  
  
MinLengthConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#MinLengthConstraintComponent')
```

```

NodeConstraintComponent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#NodeConstraintComponent')

NodeKind: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#NodeKind')

NodeKindConstraintComponent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#NodeKindConstraintComponent')

NodeShape: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#NodeShape')

NotConstraintComponent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#NotConstraintComponent')

OrConstraintComponent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#OrConstraintComponent')

Parameter: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Parameter')

Parameterizable: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#Parameterizable')

PatternConstraintComponent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#PatternConstraintComponent')

PrefixDeclaration: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#PrefixDeclaration')

PropertyConstraintComponent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#PropertyConstraintComponent')

PropertyGroup: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#PropertyGroup')

PropertyShape: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#PropertyShape')

QualifiedMaxCountConstraintComponent: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#QualifiedMaxCountConstraintComponent')

QualifiedMinCountConstraintComponent: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#QualifiedMinCountConstraintComponent')

ResultAnnotation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#ResultAnnotation')

Rule: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Rule')

SPARQLAskExecutable: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLAskExecutable')

SPARQLAskValidator: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLAskValidator')

SPARQLConstraint: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLConstraint')

```

```
SPARQLConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLConstraintComponent')  
  
SPARQLConstructExecutable: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLConstructExecutable')  
  
SPARQLExecutable: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLExecutable')  
  
SPARQLFunction: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLFunction')  
  
SPARQLRule: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLRule')  
  
SPARQLSelectExecutable: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLSelectExecutable')  
  
SPARQLSelectValidator: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLSelectValidator')  
  
SPARQLTarget: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLTarget')  
  
SPARQLTargetType: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLTargetType')  
  
SPARQLUpdateExecutable: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#SPARQLUpdateExecutable')  
  
Severity: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Severity')  
  
Shape: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Shape')  
  
Target: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Target')  
  
TargetType: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#TargetType')  
  
TripleRule: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#TripleRule')  
  
UniqueLangConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#UniqueLangConstraintComponent')  
  
ValidationReport: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#ValidationReport')  
  
ValidationResult: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#ValidationResult')  
  
Validator: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Validator')  
  
Violation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Violation')  
  
Warning: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#Warning')  
  
XoneConstraintComponent: URIRef =  
rdflib.term.URIRef('http://www.w3.org/ns/shacl#XoneConstraintComponent')
```

```

alternativePath: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#alternativePath')

annotationProperty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#annotationProperty')

annotationValue: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#annotationValue')

annotationVarName: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#annotationVarName')

ask: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#ask')

closed: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#closed')

condition: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#condition')

conforms: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#conforms')

construct: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#construct')

datatype: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#datatype')

deactivated: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#deactivated')

declare: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#declare')

defaultValue: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#defaultValue')

description: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#description')

detail: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#detail')

disjoint: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#disjoint')

entailment: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#entailment')

equals: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#equals')

expression: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#expression')

filterShape: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#filterShape')

flags: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#flags')

focusNode: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#focusNode')

group: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#group')

hasValue: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#hasValue')

ignoredProperties: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#ignoredProperties')

intersection: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#intersection')

```

```

inversePath: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#inversePath')

js: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#js')

jsFunctionName: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#jsFunctionName')

jsLibrary: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#jsLibrary')

jsLibraryURL: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#jsLibraryURL')

labelTemplate: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#labelTemplate')

languageIn: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#languageIn')

lessThan: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#lessThan')

lessThanOrEquals: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#lessThanOrEquals')

maxCount: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#maxCount')

maxExclusive: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#maxExclusive')

maxInclusive: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#maxInclusive')

maxLength: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#maxLength')

message: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#message')

minCount: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#minCount')

minExclusive: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#minExclusive')

minInclusive: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#minInclusive')

minLength: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#minLength')

name: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#name')

namespace: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#namespace')

node: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#node')

nodeKind: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#nodeKind')

nodeValidator: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#nodeValidator')

nodes: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#nodes')

object: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#object')

```

```
oneOrMorePath: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#oneOrMorePath')

optional: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#optional')

order: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#order')

parameter: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#parameter')

path: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#path')

pattern: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#pattern')

predicate: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#predicate')

prefix: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#prefix')

prefixes: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#prefixes')

property: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#property')

propertyValidator: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#propertyValidator')

qualifiedMaxCount: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#qualifiedMaxCount')

qualifiedMinCount: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#qualifiedMinCount')

qualifiedValueShape: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#qualifiedValueShape')

qualifiedValueShapesDisjoint: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#qualifiedValueShapesDisjoint')

result: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#result')

resultAnnotation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#resultAnnotation')

resultMessage: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#resultMessage')

resultPath: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#resultPath')

resultSeverity: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#resultSeverity')

returnType: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#returnType')

rule: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#rule')

select: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#select')

severity: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#severity')

shapesGraph: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#shapesGraph')
```

```

shapesGraphWellFormed: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#shapesGraphWellFormed')

sourceConstraint: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#sourceConstraint')

sourceConstraintComponent: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#sourceConstraintComponent')

sourceShape: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#sourceShape')

sparql: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#sparql')

subject: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#subject')

suggestedShapesGraph: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#suggestedShapesGraph')

target: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#target')

targetClass: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#targetClass')

targetNode: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#targetNode')

targetObjectsOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#targetObjectsOf')

targetSubjectsOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#targetSubjectsOf')

this: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#this')

union: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#union')

uniqueLang: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#uniqueLang')

update: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#update')

validator: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#validator')

value: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#value')

xone: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/shacl#xone')

zeroOrMorePath: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#zeroOrMorePath')

zeroOrOnePath: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/shacl#zeroOrOnePath')

```

```
class rdflib.SKOS
```

Bases: *DefinedNamespace*

SKOS Vocabulary

An RDF vocabulary for describing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, ‘folksonomies’, other types of controlled vocabulary, and also concept schemes embedded in glossaries and terminologies.

Generated from: <https://www.w3.org/2009/08/skos-reference/skos.rdf> Date: 2020-05-26 14:20:08.489187



```
Collection: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#Collection')  
  
Concept: URIRef = rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#Concept')  
  
ConceptScheme: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#ConceptScheme')  
  
OrderedCollection: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#OrderedCollection')  
  
altLabel: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#altLabel')  
  
broadMatch: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#broadMatch')  
  
broader: URIRef = rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#broader')  
  
broaderTransitive: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#broaderTransitive')  
  
changeNote: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#changeNote')  
  
closeMatch: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#closeMatch')  
  
definition: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#definition')  
  
editorialNote: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#editorialNote')  
  
exactMatch: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#exactMatch')  
  
example: URIRef = rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#example')  
  
hasTopConcept: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#hasTopConcept')  
  
hiddenLabel: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#hiddenLabel')  
  
historyNote: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#historyNote')  
  
inScheme: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#inScheme')  
  
mappingRelation: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#mappingRelation')  
  
member: URIRef = rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#member')
```



```

memberList: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#memberList')

narrowMatch: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#narrowMatch')

narrower: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#narrower')

narrowerTransitive: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#narrowerTransitive')

notation: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#notation')

note: URIRef = rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#note')

prefLabel: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#prefLabel')

related: URIRef = rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#related')

relatedMatch: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#relatedMatch')

scopeNote: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#scopeNote')

semanticRelation: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#semanticRelation')

topConceptOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/2004/02/skos/core#topConceptOf')

```

class rdflib.SOSA

Bases: *DefinedNamespace*

Sensor, Observation, Sample, and Actuator (SOSA) Ontology

This ontology is based on the SSN Ontology by the W3C Semantic Sensor Networks Incubator Group (SSN-XG), together with considerations from the W3C/OGC Spatial Data on the Web Working Group.

Generated from: <http://www.w3.org/ns/sosa/> Date: 2020-05-26 14:20:08.792504

```

ActuatableProperty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/ActuatableProperty')

Actuation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Actuation')

Actuator: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Actuator')

FeatureOfInterest: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/FeatureOfInterest')

ObservableProperty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/ObservableProperty')

Observation: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Observation')

```

```
Platform: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Platform')
Procedure: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Procedure')
Result: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Result')
Sample: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Sample')
Sampler: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Sampler')
Sampling: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Sampling')
Sensor: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/Sensor')

actsOnProperty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/actsOnProperty')

hasFeatureOfInterest: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/hasFeatureOfInterest')

hasResult: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/hasResult')
hasSample: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/hasSample')
hasSimpleResult: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/hasSimpleResult')

hosts: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/hosts')

isActedOnBy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/isActedOnBy')

isFeatureOfInterestOf: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/isFeatureOfInterestOf')

isHostedBy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/isHostedBy')
isObservedBy: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/isObservedBy')
isResultOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/isResultOf')
isSampleOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/isSampleOf')

madeActuation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/madeActuation')

madeByActuator: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/madeByActuator')

madeBySampler: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/madeBySampler')

madeBySensor: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/madeBySensor')

madeObservation: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/madeObservation')

madeSampling: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/madeSampling')
```

```

observedProperty: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/observedProperty')

observes: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/observes')

phenomenonTime: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/phenomenonTime')

resultTime: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/sosa/resultTime')

usedProcedure: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/sosa/usedProcedure')

```

```
class rdflib.SSN
```

```
    Bases: DefinedNamespace
```

```
    Semantic Sensor Network Ontology
```

This ontology describes sensors, actuators and observations, and related concepts. It does not describe domain concepts, time, locations, etc. these are intended to be included from other ontologies via OWL imports.

Generated from: <http://www.w3.org/ns/ssn/> Date: 2020-05-26 14:20:09.068204

```
Deployment: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/Deployment')
```

```
Input: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/Input')
```

```
Output: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/Output')
```

```
Property: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/Property')
```

```
Stimulus: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/Stimulus')
```

```
System: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/System')
```

```
deployedOnPlatform: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/ssn/deployedOnPlatform')
```

```
deployedSystem: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/ssn/deployedSystem')
```

```
detects: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/detects')
```

```
forProperty: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/forProperty')
```

```
hasDeployment: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/ssn/hasDeployment')
```

```
hasInput: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/hasInput')
```

```
hasOutput: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/hasOutput')
```

```
hasProperty: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/hasProperty')
```

```
hasSubSystem: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/hasSubSystem')
```

```
implementedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/ssn/implementedBy')
```

```
implements: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/implements')
```

```

inDeployment: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/inDeployment')
isPropertyOf: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/isPropertyOf')
isProxyFor: URIRef = rdflib.term.URIRef('http://www.w3.org/ns/ssn/isProxyFor')
wasOriginatedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/ns/ssn/wasOriginatedBy')

```

```
class rdflib.TIME
```

```
    Bases: DefinedNamespace
```

```
    OWL-Time
```

```
    Generated from: http://www.w3.org/2006/time# Date: 2020-05-26 14:20:10.531265
```

```

DateTimeDescription: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#DateTimeDescription')

```

```

DateTimeInterval: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#DateTimeInterval')

```

```
DayOfWeek: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#DayOfWeek')
```

```
Duration: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Duration')
```

```

DurationDescription: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#DurationDescription')

```

```
Friday: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Friday')
```

```

GeneralDateTimeDescription: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#GeneralDateTimeDescription')

```

```

GeneralDurationDescription: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#GeneralDurationDescription')

```

```
Instant: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Instant')
```

```
Interval: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Interval')
```

```
January: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#January')
```

```
Monday: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Monday')
```

```
MonthOfYear: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#MonthOfYear')
```

```

ProperInterval: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#ProperInterval')

```

```
Saturday: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Saturday')
```

```
Sunday: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Sunday')
```

```
TRS: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#TRS')
```

```

TemporalDuration: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#TemporalDuration')

```

```

TemporalEntity: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#TemporalEntity')

TemporalPosition: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#TemporalPosition')

TemporalUnit: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#TemporalUnit')

Thursday: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Thursday')

TimePosition: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#TimePosition')

TimeZone: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#TimeZone')

Tuesday: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Tuesday')

Wednesday: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Wednesday')

Year: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#Year')

after: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#after')

before: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#before')

day: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#day')

dayOfWeek: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#dayOfWeek')

dayOfYear: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#dayOfYear')

days: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#days')

generalDay: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#generalDay')

generalMonth: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#generalMonth')

generalYear: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#generalYear')

hasBeginning: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#hasBeginning')

hasDateTimeDescription: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#hasDateTimeDescription')

hasDuration: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#hasDuration')

hasDurationDescription: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#hasDurationDescription')

hasEnd: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#hasEnd')

hasTRS: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#hasTRS')

hasTemporalDuration: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#hasTemporalDuration')

```

```

hasTime: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#hasTime')

hasXSDDuration: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#hasXSDDuration')

hour: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#hour')

hours: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#hours')

inDateTime: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#inDateTime')

inTemporalPosition: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#inTemporalPosition')

inTimePosition: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#inTimePosition')

inXSDDate: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#inXSDDate')

inXSDDateTime: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#inXSDDateTime')

inXSDDateTimeStamp: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#inXSDDateTimeStamp')

inXSDgYear: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#inXSDgYear')

inXSDgYearMonth: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#inXSDgYearMonth')

inside: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#inside')

intervalAfter: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalAfter')

intervalBefore: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalBefore')

intervalContains: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalContains')

intervalDisjoint: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalDisjoint')

intervalDuring: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalDuring')

intervalEquals: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalEquals')

intervalFinishedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalFinishedBy')

intervalFinishes: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalFinishes')

intervalIn: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#intervalIn')

```

```

intervalMeets: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalMeets')

intervalMetBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalMetBy')

intervalOverlappedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalOverlappedBy')

intervalOverlaps: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalOverlaps')

intervalStartedBy: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalStartedBy')

intervalStarts: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#intervalStarts')

minute: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#minute')

minutes: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#minutes')

month: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#month')

monthOfYear: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#monthOfYear')

months: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#months')

nominalPosition: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#nominalPosition')

numericDuration: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#numericDuration')

numericPosition: URIRef =
rdflib.term.URIRef('http://www.w3.org/2006/time#numericPosition')

second: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#second')

seconds: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#seconds')

timeZone: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#timeZone')

unitDay: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#unitDay')

unitHour: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#unitHour')

unitMinute: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#unitMinute')

unitMonth: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#unitMonth')

unitSecond: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#unitSecond')

unitType: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#unitType')

unitWeek: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#unitWeek')

unitYear: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#unitYear')

```



```

week: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#week')
weeks: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#weeks')
xsdDateTime: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#xsdDateTime')
year: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#year')
years: URIRef = rdflib.term.URIRef('http://www.w3.org/2006/time#years')

```

```
class rdflib.URIRef(value: str, base: Optional[str] = None)
```

Bases: *IdentifiedNode*

RDF 1.1's IRI Section <https://www.w3.org/TR/rdf11-concepts/#section-IRIs>

---

**Note:** Documentation on RDF outside of RDFLib uses the term IRI or URI whereas this class is called *URIRef*. This is because it was made when the first version of the RDF specification was current, and it used the term *URIRef*, see [RDF 1.0 URIRef](#)

---

An IRI (Internationalized Resource Identifier) within an RDF graph is a Unicode string that conforms to the syntax defined in RFC 3987.

IRIs in the RDF abstract syntax MUST be absolute, and MAY contain a fragment identifier.

IRIs are a generalization of URIs [RFC3986] that permits a wider range of Unicode characters.

**\_\_add\_\_**(*other*)

Return self+value.

**Return type**

*URIRef*

```

__annotations__ = {'__invert__': typing.Callable[[ForwardRef('URIRef')],
ForwardRef('InvPath')], '__neg__': typing.Callable[[ForwardRef('URIRef')],
ForwardRef('NegatedPath')], '__or__': typing.Callable[[ForwardRef('URIRef'),
typing.Union[ForwardRef('URIRef'), ForwardRef('Path')]],
ForwardRef('AlternativePath')], '__truediv__':
typing.Callable[[ForwardRef('URIRef'), typing.Union[ForwardRef('URIRef'),
ForwardRef('Path')]], ForwardRef('SequencePath')]}

```

**\_\_invert\_\_**()

inverse path

**Parameters**

**p** (*Union*[*URIRef*, *Path*]) –

**Return type**

*InvPath*

**\_\_mod\_\_**(*other*)

Return self%value.

**Return type**

*URIRef*

**\_\_module\_\_** = 'rdflib.term'



**\_\_mul\_\_**(*mul*)  
cardinality path

**Parameters**

- **p** (`Union[URIRef, Path]`) –
- **mul** (`Literal['*', '+', '?']`) –

**Return type**  
`MulPath`

**\_\_neg\_\_**()  
negated path

**Parameters**

**p** (`Union[URIRef, AlternativePath, InvPath]`) –

**Return type**  
`NegatedPath`

**static** **\_\_new\_\_**(*cls, value, base=None*)

**Parameters**

- **value** (`str`) –
- **base** (`Optional[str]`) –

**Return type**  
`URIRef`

**\_\_or\_\_**(*other*)  
alternative path

**Parameters**

- **self** (`Union[URIRef, Path]`) –
- **other** (`Union[URIRef, Path]`) –

**\_\_radd\_\_**(*other*)

**Return type**  
`URIRef`

**\_\_reduce\_\_**()  
Helper for pickle.

**Return type**  
`Tuple[Type[URIRef], Tuple[str]]`

**\_\_repr\_\_**()  
Return repr(self).

**Return type**  
`str`

**\_\_slots\_\_** = ()

**\_\_truediv\_\_**(*other*)  
sequence path

**Parameters**

- **self** ([Union](#)[[URIRef](#), [Path](#)]) –
- **other** ([Union](#)[[URIRef](#), [Path](#)]) –

**de\_skolemize()**

Create a Blank Node from a skolem URI, in accordance with <http://www.w3.org/TR/rdf11-concepts/#section-skolemization>. This function accepts only rdflib type skolemization, to provide a round-tripping within the system. :rtype: [BNode](#)

New in version 4.0.

**defrag()**

**Return type**

[URIRef](#)

**property fragment:** [str](#)

Return the URL Fragment

```
>>> URIRef("http://example.com/some/path/#some-fragment").fragment
'some-fragment'
>>> URIRef("http://example.com/some/path/").fragment
''
```

**n3(namespace\_manager=None)**

This will do a limited check for valid URIs, essentially just making sure that the string includes no illegal characters (<, >, ", {, }, |, \, `, ^)

**Parameters**

**namespace\_manager** ([Optional](#)[[NamespaceManager](#)]) – if not None, will be used to make up a prefixed name

**Return type**

[str](#)

**class rdflib.VANN**

Bases: [DefinedNamespace](#)

VANN: A vocabulary for annotating vocabulary descriptions

This document describes a vocabulary for annotating descriptions of vocabularies with examples and usage notes.

Generated from: <https://vocab.org/vann/vann-vocab-20100607.rdf> Date: 2020-05-26 14:21:15.580430

**changes:** [URIRef](#) = `rdflib.term.URIRef('http://purl.org/vocab/vann/changes')`

**example:** [URIRef](#) = `rdflib.term.URIRef('http://purl.org/vocab/vann/example')`

**preferredNamespacePrefix:** [URIRef](#) =  
`rdflib.term.URIRef('http://purl.org/vocab/vann/preferredNamespacePrefix')`

**preferredNamespaceUri:** [URIRef](#) =  
`rdflib.term.URIRef('http://purl.org/vocab/vann/preferredNamespaceUri')`

**termGroup:** [URIRef](#) = `rdflib.term.URIRef('http://purl.org/vocab/vann/termGroup')`

**usageNote:** [URIRef](#) = `rdflib.term.URIRef('http://purl.org/vocab/vann/usageNote')`

**class rdflib.VOID**Bases: *DefinedNamespace*

Vocabulary of Interlinked Datasets (VoID)

The Vocabulary of Interlinked Datasets (VoID) is an RDF Schema vocabulary for expressing metadata about RDF datasets. It is intended as a bridge between the publishers and users of RDF data, with applications ranging from data discovery to cataloging and archiving of datasets. This document provides a formal definition of the new RDF classes and properties introduced for VoID. It is a companion to the main specification document for VoID, <http://www.w3.org/TR/void/> Describing Linked Datasets with the VoID Vocabulary.

Generated from: <http://rdfs.org/ns/void#> Date: 2020-05-26 14:20:11.911298Dataset: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#Dataset')DatasetDescription: *URIRef* =  
rdflib.term.URIRef('http://rdfs.org/ns/void#DatasetDescription')Linkset: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#Linkset')TechnicalFeature: *URIRef* =  
rdflib.term.URIRef('http://rdfs.org/ns/void#TechnicalFeature')classPartition: *URIRef* =  
rdflib.term.URIRef('http://rdfs.org/ns/void#classPartition')classes: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#classes')dataDump: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#dataDump')distinctObjects: *URIRef* =  
rdflib.term.URIRef('http://rdfs.org/ns/void#distinctObjects')distinctSubjects: *URIRef* =  
rdflib.term.URIRef('http://rdfs.org/ns/void#distinctSubjects')documents: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#documents')entities: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#entities')exampleResource: *URIRef* =  
rdflib.term.URIRef('http://rdfs.org/ns/void#exampleResource')feature: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#feature')inDataset: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#inDataset')linkPredicate: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#linkPredicate')objectsTarget: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#objectsTarget')openSearchDescription: *URIRef* =  
rdflib.term.URIRef('http://rdfs.org/ns/void#openSearchDescription')properties: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#properties')property: *URIRef* = rdflib.term.URIRef('http://rdfs.org/ns/void#property')

```
propertyPartition: URIRef =  
rdflib.term.URIRef('http://rdfs.org/ns/void#propertyPartition')  
  
rootResource: URIRef = rdflib.term.URIRef('http://rdfs.org/ns/void#rootResource')  
  
sparqlEndpoint: URIRef =  
rdflib.term.URIRef('http://rdfs.org/ns/void#sparqlEndpoint')  
  
subjectsTarget: URIRef =  
rdflib.term.URIRef('http://rdfs.org/ns/void#subjectsTarget')  
  
subset: URIRef = rdflib.term.URIRef('http://rdfs.org/ns/void#subset')  
  
target: URIRef = rdflib.term.URIRef('http://rdfs.org/ns/void#target')  
  
triples: URIRef = rdflib.term.URIRef('http://rdfs.org/ns/void#triples')  
  
uriLookupEndpoint: URIRef =  
rdflib.term.URIRef('http://rdfs.org/ns/void#uriLookupEndpoint')  
  
uriRegexPattern: URIRef =  
rdflib.term.URIRef('http://rdfs.org/ns/void#uriRegexPattern')  
  
uriSpace: URIRef = rdflib.term.URIRef('http://rdfs.org/ns/void#uriSpace')  
  
vocabulary: URIRef = rdflib.term.URIRef('http://rdfs.org/ns/void#vocabulary')
```

```
class rdflib.Variable(value: str)
```

Bases: *Identifier*

A Variable - this is used for querying, or in Formula aware graphs, where Variables can be stored

```
__annotations__ = {}
```

```
__module__ = 'rdflib.term'
```

```
static __new__(cls, value)
```

Parameters

value (*str*) –

Return type

*Variable*

```
__reduce__()
```

Helper for pickle.

Return type

*Tuple*[*Type*[*Variable*], *Tuple*[*str*]]

```
__repr__()
```

Return repr(self).

Return type

*str*

```
__slots__ = ()
```

`n3(namespace_manager=None)`

**Parameters**

`namespace_manager` (Optional[*NamespaceManager*]) –

**Return type**

`str`

`toPython()`

**Return type**

`str`

`class rdflib.XSD`

Bases: *DefinedNamespace*

W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes

Generated from: ../schemas/datatypes.xsd Date: 2021-09-05 20:37+10

**Assertions:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#Assertions')`

**ENTITIES:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#ENTITIES')`

**ENTITY:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#ENTITY')`

**ID:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#ID')`

**IDREF:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#IDREF')`

**IDREFS:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#IDREFS')`

**NCName:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#NCName')`

**NMTOKEN:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#NMTOKEN')`

**NMTOKENS:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#NMTOKENS')`

**NOTATION:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#NOTATION')`

**Name:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#Name')`

**QName:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#QName')`

**anyURI:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#anyURI')`

**base64Binary:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#base64Binary')`

**boolean:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#boolean')`

**bounded:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#bounded')`

**byte:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#byte')`

**cardinality:** *URIRef* =

`rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#cardinality')`

**date:** *URIRef* = `rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#date')`

```
dateTime: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#dateTime')
dateTimeStamp: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#dateTimeStamp')
day: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#day')
dayTimeDuration: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#dayTimeDuration')
decimal: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#decimal')
double: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#double')
duration: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#duration')
enumeration: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#enumeration')
explicitTimezone: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#explicitTimezone')
float: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#float')
fractionDigits: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#fractionDigits')
gDay: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#gDay')
gMonth: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#gMonth')
gMonthDay: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#gMonthDay')
gYear: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#gYear')
gYearMonth: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#gYearMonth')
hexBinary: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#hexBinary')
hour: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#hour')
int: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#int')
integer: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#integer')
language: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#language')
length: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#length')
long: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#long')
maxExclusive: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#maxExclusive')
maxInclusive: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#maxInclusive')
```

```

maxLength: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#maxLength')

minExclusive: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#minExclusive')

minInclusive: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#minInclusive')

minLength: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#minLength')

minute: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#minute')

month: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#month')

negativeInteger: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#negativeInteger')

nonNegativeInteger: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#nonNegativeInteger')

nonPositiveInteger: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#nonPositiveInteger')

normalizedString: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#normalizedString')

numeric: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#numeric')

ordered: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#ordered')

pattern: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#pattern')

positiveInteger: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#positiveInteger')

second: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#second')

short: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#short')

string: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#string')

time: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#time')

timezoneOffset: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#timezoneOffset')

token: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#token')

totalDigits: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#totalDigits')

unsignedByte: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#unsignedByte')

unsignedInt: URIRef =
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#unsignedInt')

```

```
unsignedLong: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#unsignedLong')  
  
unsignedShort: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#unsignedShort')  
  
whiteSpace: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#whiteSpace')  
  
year: URIRef = rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#year')  
  
yearMonthDuration: URIRef =  
rdflib.term.URIRef('http://www.w3.org/2001/XMLSchema#yearMonthDuration')
```

## 3.2 Plugins

Many parts of RDFLib are extensible with plugins, see [setuptools](#)' 'Creating and discovering plugins'. These pages list the plugins included in RDFLib core.

### 3.2.1 Plugin parsers

These serializers are available in default RDFLib, you can use them by passing the name to graph's [parse\(\)](#) method:

```
graph.parse(my_url, format='n3')
```

The html parser will auto-detect RDFa, HTurtle or Microdata.

It is also possible to pass a mime-type for the `format` parameter:

```
graph.parse(my_url, format='application/rdf+xml')
```

If you are not sure what format your file will be, you can use `rdflib.util.guess_format()` which will guess based on the file extension.

Name	Class
json-ld	<i>JsonLDParser</i>
hext	<i>HextuplesParser</i>
n3	<i>N3Parser</i>
nquads	<i>NQuadsParser</i>
nt	<i>NTParser</i>
trix	<i>TriXParser</i>
turtle	<i>TurtleParser</i>
xml	<i>RDFXMLParser</i>



## Multi-graph IDs

Note that for correct parsing of multi-graph data, e.g. Trig, HexT, etc., into a `ConjunctiveGraph` or a `Dataset`, as opposed to a context-unaware `Graph`, you will need to set the `publicID` of the `ConjunctiveGraph` a `Dataset` to the identifier of the `default_context` (default graph), for example:

```
d = Dataset()
d.parse(
    data="""" ... """,
    format="trig",
    publicID=d.default_context.identifier
)
```

(from the file `tests/test_serializer_hext.py`)

## 3.2.2 Plugin serializers

These serializers are available in default RDFLib, you can use them by passing the name to a graph's `serialize()` method:

```
print graph.serialize(format='n3')
```

It is also possible to pass a mime-type for the `format` parameter:

```
graph.serialize(my_url, format='application/rdf+xml')
```

Name	Class
json-ld	<i>JsonLDSerializer</i>
n3	<i>N3Serializer</i>
nquads	<i>NQuadsSerializer</i>
nt	<i>NTSerializer</i>
hext	<i>HextuplesSerializer</i>
pretty-xml	<i>PrettyXMLSerializer</i>
trig	<i>TrigSerializer</i>
trix	<i>TriXSerializer</i>
turtle	<i>TurtleSerializer</i>
longturtle	<i>LongTurtleSerializer</i>
xml	<i>XMLSerializer</i>

## JSON-LD

JSON-LD - 'json-ld' - has been incorporated in rdflib since v6.0.0.

## HexTuples

The HexTuples Serializer - ‘hext’ - uses the HexTuples format defined at <https://github.com/ontola/hextuples>.

For serialization of non-context-aware data sources, e.g. a single `Graph`, the ‘graph’ field (6th variable in the Hextuple) will be an empty string.

For context-aware (multi-graph) serialization, the ‘graph’ field of the default graph will be an empty string and the values for other graphs will be Blank Node IDs or IRIs.

## 3.2.3 Plugin stores

### Built In

The following Stores are contained within the rdflib core package:

Name	Class
Auditable	<i>AuditableStore</i>
Concurrent	<i>ConcurrentStore</i>
SimpleMemory	<i>SimpleMemory</i>
Memory	<i>Memory</i>
SPARQLStore	<i>SPARQLStore</i>
SPARQLUpdateStore	<i>SPARQLUpdateStore</i>
BerkeleyDB	<i>BerkeleyDB</i>
default	<i>Memory</i>

### External

The following Stores are defined externally to rdflib’s core package, so look to their documentation elsewhere for specific details of use.

Name	Repository	Notes
SQLAlchemy	<a href="https://github.com/RDFLib/rdflib-sqlalchemy">https://github.com/RDFLib/rdflib-sqlalchemy</a>	An SQLAlchemy-backed, formula-aware RDFLib Store. Tested dialects are: SQLite, MySQL & PostgreSQL
leveldb	<a href="https://github.com/RDFLib/rdflib-leveldb">https://github.com/RDFLib/rdflib-leveldb</a>	An adaptation of RDFLib BerkeleyDB Store’s key-value approach, using LevelDB as a back-end
Kyoto Cabinet	<a href="https://github.com/RDFLib/rdflib-kyotocabinet">https://github.com/RDFLib/rdflib-kyotocabinet</a>	An adaptation of RDFLib BerkeleyDB Store’s key-value approach, using Kyoto Cabinet as a back-end
HDT	<a href="https://github.com/RDFLib/rdflib-hdt">https://github.com/RDFLib/rdflib-hdt</a>	A Store back-end for rdflib to allow for reading and querying HDT documents
Oxi-graph	<a href="https://github.com/oxigraph/oxrdflib">https://github.com/oxigraph/oxrdflib</a>	Works with the <a href="#">Pyoxigraph</a> Python graph database library

*If you have, or know of a Store implementation and would like it listed here, please submit a Pull Request!*

## Use

You can use these stores like this:

```
from rdflib import Graph

# use the default memory Store
graph = Graph()

# use the BerkeleyDB Store
graph = Graph(store="BerkeleyDB")
```

In some cases, you must explicitly *open* and *close* a store, for example:

```
from rdflib import Graph

# use the BerkeleyDB Store
graph = Graph(store="BerkeleyDB")
graph.open("/some/folder/location")
# do things ...
graph.close()
```

### 3.2.4 Plugin query results

Plugins for reading and writing of (SPARQL) *Result* - pass name to either *parse()* or *serialize()*

#### Parsers

Name	Class
csv	<i>CSVResultParser</i>
json	<i>JSONResultParser</i>
tsv	<i>TSVResultParser</i>
xml	<i>XMLResultParser</i>

#### Serializers

Name	Class
csv	<i>CSVResultSerializer</i>
json	<i>JSONResultSerializer</i>
txt	<i>TXTResultSerializer</i>
xml	<i>XMLResultSerializer</i>



## FOR DEVELOPERS

### 4.1 RDFLib developers guide

#### 4.1.1 Introduction

This document describes the process and conventions to follow when developing RDFLib code.

- Please be as Pythonic as possible ([PEP 8](#)).
- Code should be formatted using [black](#) and we use Black v23.1.0, with the black config in `pyproject.toml`.
- Code should also pass [flake8](#) linting and [mypy](#) type checking.
- You must supply tests for new code.
- RDFLib uses [Poetry](#) for dependency management and packaging.

If you add a new cool feature, consider also adding an example in `./examples`

#### 4.1.2 Pull Requests Guidelines

Contributions to RDFLib are made through pull requests (PRs).

In general, maintainers will only merge PRs if the following conditions are met:

- The PR has been sufficiently reviewed.

Each PR should be reviewed and approved by at least two people other than the author of the PR before it is merged and PRs will be processed faster if they are easier to review and approve of.

Reviews are open to everyone, but the weight assigned to any particular review is at the discretion of maintainers.
- Changes that have a runtime impact are covered by unit tests.

There should either be existing tests that cover the changed code and behaviour, or the PR should include tests. For more information about what is considered adequate testing see the [Tests section](#).
- Documentation that covers something that changed has been updated.
- Type checks and unit tests that are part of our continuous integration workflow pass.

In addition to these conditions, PRs that are easier to review and approve will be processed quicker. The primary factors that determine this is the scope and size of a PR. If there are few changes and the scope is limited then there is less that a reviewer has to understand and less that they can disagree with. It is thus important to try and split up your changes into multiple independent PRs if possible. No PR is too small.

For PRs that introduce breaking changes, it is even more critical that they are limited in size and scope, as they will likely have to be kept up to date with the `main` branch of this project for some time before they are merged.

It is also critical that your PR is understandable both in what it does and why it does it, and how the change will impact the users of this project, for this reason it is essential that your PR's description explains the nature of the PR, what the PR intends to do, why this is desirable, and how this will affect the users of this project.

Please note that while we would like all PRs to follow the guidelines given here, we will not reject a PR just because it does not.

### 4.1.3 Tests

Any new functionality being added to RDFLib *must* have unit tests and should have doc tests supplied.

Typically, you should add your functionality and new tests to a branch of RDFLib and run all tests locally and see them pass. There are currently close to 4,000 tests with a few extra expected failures and skipped tests. We won't allow Pull Requests that break any of the existing tests.

Tests that you add should show how your new feature or bug fix is doing what you say it is doing: if you remove your enhancement, your new tests should fail!

Finally, please consider adding simple and more complex tests. It's good to see the basic functionality of your feature tests and then also any tricky bits or edge cases.

#### Testing framework

RDFLib uses the [pytest](#) testing framework.

#### Running tests

To run RDFLib's test suite with [pytest](#):

```
$ poetry install
$ poetry run pytest
```

Specific tests can be run by file name. For example:

```
$ poetry run pytest test/test_graph/test_graph.py
```

For more extensive tests, including tests for the [berkeleydb](#) backend, install extra requirements before executing the tests.

```
$ poetry install --all-extras
$ poetry run pytest
```

#### Writing tests

New tests should be written for [pytest](#) instead of for python's built-in [unittest](#) module as pytest provides advanced features such as parameterization and more flexibility in writing expected failure tests than [unittest](#).

A primer on how to write tests for pytest can be found [here](#).

The existing tests that use [unittest](#) work well with pytest, but they should ideally be updated to the pytest test-style when they are touched.

Test should go into the `test/` directory, either into an existing test file with a name that is applicable to the test being written, or into a new test file with a name that is descriptive of the tests placed in it. Test files should be named `test_*.py` so that [pytest](#) can discover them.

#### 4.1.4 Running static checks

Check formatting with `black`, making sure you use our `black.toml` config file:

```
poetry run black .
```

Check style and conventions with `flake8`:

```
poetry run flake8 rdflib
```

We also provide a `flakeheaven` baseline that ignores existing `flake8` errors and only reports on newly introduced `flake8` errors:

```
poetry run flakeheaven
```

Check types with `mypy`:

```
poetry run mypy --show-error-context --show-error-codes
```

#### 4.1.5 pre-commit and pre-commit ci

We have `pre-commit` configured with `black` for formatting code.

Some useful commands for using `pre-commit`:

```
# Install pre-commit.
pip install --user --upgrade pre-commit

# Install pre-commit hooks, this will run pre-commit
# every time you make a git commit.
pre-commit install

# Run pre-commit on changed files.
pre-commit run

# Run pre-commit on all files.
pre-commit run --all-files
```

There is also two tox environments for `pre-commit`:

```
# run pre-commit on changed files.
tox -e precommit

# run pre-commit on all files.
tox -e precommitall
```

There is no hard requirement for pull requests to be processed with `pre-commit` (or the underlying processors), however doing this makes for a less noisy codebase with cleaner history.

We have enabled <https://pre-commit.ci/> and this can be used to automatically fix pull requests by commenting `pre-commit.ci autofix` on a pull request.

### 4.1.6 Using tox

RDFLib has a `tox` config file that makes it easier to run validation on all supported python versions.

```
# Install tox.
pip install tox

# List the tox environments that run by default.
tox -e

# Run the default environments.
tox

# List all tox environments, including ones that don't run by default.
tox -a

# Run a specific environment.
tox -e py37 # default environment with py37
tox -e py39-extra # extra tests with py39

# Override the test command.
# the below command will run `pytest test/test_translate_algebra.py`
# instead of the default pytest command.
tox -e py37,py39 -- pytest test/test_translate_algebra.py
```

### 4.1.7 go-task and Taskfile.yml

A `Taskfile.yml` is provided for `go-task` with various commands that facilitate development.

Instructions for installing `go-task` can be seen in the [go-task installation guide](#).

Some useful commands for working with the task in the taskfile is given below:

```
# List available tasks.
task -l

# Configure the environment for development
task configure

# Run basic validation
task validate

# Build docs
task docs:build

# Run live-preview on the docs
task docs:live-server

# Run the py310 tox environment
task tox -- -e py310
```

The [Taskfile usage documentation](#) provides more information on how to work with taskfiles.



### 4.1.8 Development container

To simplify the process of getting a working development environment to develop rdflib in we provide a [Development Container](#) (*devcontainer*) that is configured in [Docker Compose](#). This container can be used directly to run various commands, or it can be used with [editors that support Development Containers](#).

**Important:** The devcontainer is intended to run with a [rootless docker](#) daemon so it can edit files owned by the invoking user without an involved configuration process.

Using a rootless docker daemon also has general security benefits.

To use the development container directly:

```
# Build the devcontainer docker image.
docker-compose build

# Configure the system for development.
docker-compose run --rm run task configure

# Run the validate task inside the devtools container.
docker-compose run --rm run task validate

# Run extensive tests inside the devtools container.
docker-compose run --rm run task EXTENSIVE=true test

# To get a shell into the devcontainer docker image.
docker-compose run --rm run bash
```

The devcontainer also works with [Podman Compose](#).

Details on how to use the development container with [VSCode](#) can found in the [Developing inside a Container](#) page. With the VSCode [development container CLI](#) installed the following command can be used to open the repository inside the development container:

```
# Inside the repository base directory
cd ./rdflib/

# Build the development container.
devcontainer build .

# Open the code inside the development container.
devcontainer open .
```

### 4.1.9 Writing documentation

We use sphinx for generating HTML docs, see [Writing RDFLib Documentation](#).

### 4.1.10 Continuous Integration

We used GitHub Actions for CI, see:

<https://github.com/RDFLib/rdflib/actions>

If you make a pull-request to RDFLib on GitHub, GitHub Actions will automatically test your code and we will only merge code passing all tests.

Please do *not* commit tests you know will fail, even if you're just pointing out a bug. If you commit such tests, flag them as expecting to fail.

### 4.1.11 Compatibility

RDFLib 6.0.0 release and later only support Python 3.7 and newer.

RDFLib 5.0.0 maintained compatibility with Python versions 2.7, 3.4, 3.5, 3.6, 3.7.

### 4.1.12 Releasing

Create a release-preparation pull request with the following changes:

- Updated copyright year in the LICENSE file.
- Updated copyright year in the docs/conf.py file.
- Updated main branch version and current version in the README.md file. The main branch version should be the next major version with an `a0` suffix to indicate it is alpha 0. When releasing 6.3.1, the main branch version in the README should be 6.4.0a0.
- Updated version in the pyproject.toml file.
- Updated `__date__` in the rdflib/\_\_init\_\_.py file.
- Accurate CHANGELOG.md entry for the release.

Once the PR is merged, switch to the main branch, build the release and upload it to PyPI:

```
# Clean up any previous builds
\rm -vf dist/*

# Build artifacts
poetry build

# Check that the built wheel works correctly:
pipx run --spec "$ (readlink -f dist/rdflib*.whl)" rdfpipe --version

# Publish to PyPI
poetry publish
```

Once this is done, create a release tag from [GitHub releases](#). For a release of version 6.3.1 the tag should be 6.3.1 (without a “v” prefix), and the release title should be “RDFLib 6.3.1”. The release notes for the latest version be added to the release description. The artifacts built with `poetry build` should be uploaded to the release as release artifacts.

The resulting release will be available at <https://github.com/RDFLib/rdflib/releases/tag/6.3.1>

Once this is done announce the release at the following locations:

- Twitter: Just make a tweet from your own account linking to the latest release.

- RDFLib mailing list.
- RDFLib Gitter / matrix.org chat room.

Once this is all done, create another post-release pull request with the following changes:

- Set the just released version in `docker/latest/requirements.in` and run `task docker:prepare` to update the `docker/latest/requirements.txt` file.
- Set the version in the `pyproject.toml` file to the next minor release with a `a0` suffix to indicate alpha 0.

## 4.2 Contributor Covenant Code of Conduct

### 4.2.1 Our Pledge

We as members, contributors, and leaders pledge to make participation in our community a harassment-free experience for everyone, regardless of age, body size, visible or invisible disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, caste, color, religion, or sexual identity and orientation.

We pledge to act and interact in ways that contribute to an open, welcoming, diverse, inclusive, and healthy community.

### 4.2.2 Our Standards

Examples of behavior that contributes to a positive environment for our community include:

- Demonstrating empathy and kindness toward other people
- Being respectful of differing opinions, viewpoints, and experiences
- Giving and gracefully accepting constructive feedback
- Accepting responsibility and apologizing to those affected by our mistakes, and learning from the experience
- Focusing on what is best not just for us as individuals, but for the overall community

Examples of unacceptable behavior include:

- The use of sexualized language or imagery, and sexual attention or advances of any kind
- Trolling, insulting or derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or email address, without their explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

### 4.2.3 Enforcement Responsibilities

Community leaders are responsible for clarifying and enforcing our standards of acceptable behavior and will take appropriate and fair corrective action in response to any behavior that they deem inappropriate, threatening, offensive, or harmful.

Community leaders have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, and will communicate reasons for moderation decisions when appropriate.

## 4.2.4 Scope

This Code of Conduct applies within all community spaces, and also applies when an individual is officially representing the community in public spaces. Examples of representing our community include using an official e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event.

## 4.2.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported to the community leaders responsible for enforcement at <https://github.com/RDFLib/rdflib/discussions>. All complaints will be reviewed and investigated promptly and fairly.

All community leaders are obligated to respect the privacy and security of the reporter of any incident.

## 4.2.6 Enforcement Guidelines

Community leaders will follow these Community Impact Guidelines in determining the consequences for any action they deem in violation of this Code of Conduct:

### 1. Correction

**Community Impact:** Use of inappropriate language or other behavior deemed unprofessional or unwelcome in the community.

**Consequence:** A private, written warning from community leaders, providing clarity around the nature of the violation and an explanation of why the behavior was inappropriate. A public apology may be requested.

### 2. Warning

**Community Impact:** A violation through a single incident or series of actions.

**Consequence:** A warning with consequences for continued behavior. No interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, for a specified period of time. This includes avoiding interactions in community spaces as well as external channels like social media. Violating these terms may lead to a temporary or permanent ban.

### 3. Temporary Ban

**Community Impact:** A serious violation of community standards, including sustained inappropriate behavior.

**Consequence:** A temporary ban from any sort of interaction or public communication with the community for a specified period of time. No public or private interaction with the people involved, including unsolicited interaction with those enforcing the Code of Conduct, is allowed during this period. Violating these terms may lead to a permanent ban.

## 4. Permanent Ban

**Community Impact:** Demonstrating a pattern of violation of community standards, including sustained inappropriate behavior, harassment of an individual, or aggression toward or disparagement of classes of individuals.

**Consequence:** A permanent ban from any sort of public interaction within the community.

### 4.2.7 Attribution

This Code of Conduct is adapted from the [Contributor Covenant](https://www.contributor-covenant.org/version/2/1/code_of_conduct.html), version 2.1, available at [https://www.contributor-covenant.org/version/2/1/code\\_of\\_conduct.html](https://www.contributor-covenant.org/version/2/1/code_of_conduct.html).

Community Impact Guidelines were inspired by [Mozilla's code of conduct enforcement ladder](#).

For answers to common questions about this code of conduct, see the FAQ at <https://www.contributor-covenant.org/faq>. Translations are available at <https://www.contributor-covenant.org/translations>.

## 4.3 Writing RDFLib Documentation

These docs are generated with Sphinx.

Sphinx makes it very easy to pull in doc-strings from modules, classes, methods, etc. When writing doc-strings, special reST fields can be used to annotate parameters, return-types, etc. This makes for pretty API docs. See [here](#) for the Sphinx documentation about these fields.

### 4.3.1 Building

To build the documentation you can use Sphinx from within the poetry environment. To do this, run the following commands:

```
# Install poetry venv
poetry install

# Build the sphinx docs
poetry run sphinx-build -b html -d docs/_build/doctrees docs docs/_build/html
```

Docs will be generated in docs/\_build/html and API documentation, generated from doc-strings, will be placed in docs/apidocs/.

There is also a `tox` environment for building documentation:

```
tox -e docs
```

### 4.3.2 API Docs

API Docs are automatically generated with `sphinx-apidoc`:

```
poetry run sphinx-apidoc -f -d 10 -o docs/apidocs/ rdflib examples
```

**Note that `rdflib.rst` was manually tweaked so as to not include all imports in `rdflib/__init__.py`.**

### 4.3.3 Tables

The tables in `plugin_*.rst` were generated with `plugintable.py`

## 4.4 Persisting Notation 3 Terms

### 4.4.1 Using N3 Syntax for Persistence

Blank Nodes, Literals, URI References, and Variables can be distinguished in persistence by relying on Notation 3 syntax convention.

All URI References can be expanded and persisted as:

```
<..URI..>
```

All Literals can be expanded and persisted as:

```
"..value.."@lang or "..value.."^^dtype_uri
```

---

**Note:** `@lang` is a language tag and `^^dtype_uri` is the URI of a data type associated with the Literal

---

Blank Nodes can be expanded and persisted as:

```
_:Id
```

---

**Note:** where `Id` is an identifier as determined by skolemization. Skolemization is a syntactic transformation routinely used in automatic inference systems in which existential variables are replaced by ‘new’ functions - function names not used elsewhere - applied to any enclosing universal variables. In RDF, Skolemization amounts to replacing every blank node in a graph by a ‘new’ name, i.e. a URI reference which is guaranteed to not occur anywhere else. In effect, it gives ‘arbitrary’ names to the anonymous entities whose existence was asserted by the use of blank nodes: the arbitrariness of the names ensures that nothing can be inferred that would not follow from the bare assertion of existence represented by the blank node. (Using a literal would not do. Literals are never ‘new’ in the required sense.)

---

Variables can be persisted as they appear in their serialization (`?varName`) - since they only need be unique within their scope (the context of their associated statements)

These syntactic conventions can facilitate term round-tripping.

### 4.4.2 Variables by Scope

Would an interface be needed in order to facilitate a quick way to aggregate all the variables in a scope (given by a formula identifier)? An interface such as:

```
def variables(formula_identifier)
```

### 4.4.3 The Need to Skolemize Formula Identifiers

It would seem reasonable to assume that a formula-aware store would assign Blank Node identifiers as names of formulae that appear in a N3 serialization. So for instance, the following bit of N3:

```
{?x a :N3Programmer} => {?x :has :Migrane}
```

Could be interpreted as the assertion of the following statement:

```
_:a log:implies _:b
```

However, how are `_:a` and `_:b` distinguished from other Blank Nodes? A formula-aware store would be expected to persist the first set of statements as quoted statements in a formula named `_:a` and the second set as quoted statements in a formula named `_:b`, but it would not be cost-effective for a serializer to have to query the store for all statements in a context named `_:a` in order to determine if `_:a` was associated with a formula (so that it could be serialized properly).

### 4.4.4 Relying on log:Formula Membership

The store could rely on explicit `log:Formula` membership (via `rdf:type` statements) to model the distinction of Blank Nodes associated with formulae. However, would these statements be expected from an N3 parser or known implicitly by the store? i.e., would all such Blank Nodes match the following pattern:

```
?formula rdf:type log:Formula
```

### 4.4.5 Relying on an Explicit Interface

A formula-aware store could also support the persistence of this distinction by implementing a method that returns an iterator over all the formulae in the store:

```
def formulae(triple=None)
```

This function would return all the Blank Node identifiers assigned to formulae or just those that contain statements matching the given triple pattern and would be the way a serializer determines if a term refers to a formula (in order to properly serialize it).

How much would such an interface reduce the need to model formulae terms as first class objects (perhaps to be returned by the `triples()` function)? Would it be more useful for the *Graph* (or the store itself) to return a Context object in place of a formula term (using the formulae interface to make this determination)?

Conversely, would these interfaces (variables and formulae) be considered optimizations only since you have the distinction by the kinds of terms triples returns (which would be expanded to include variables and formulae)?

### 4.4.6 Persisting Formula Identifiers

This is the most straight forward way to maintain this distinction - without relying on extra interfaces. Formula identifiers could be persisted distinctly from other terms by using the following notation:

`{_:bnode} or {<.. URI ..>}`

This would facilitate their persistence round-trip - same as the other terms that rely on N3 syntax to distinguish between each other.

## 4.5 Type Hints

This document provides some details about the type hints for RDFLib. More information about type hints can be found [here](#)

### 4.5.1 Rationale for Type Hints

Type hints are code annotations that describe the types of variables, function parameters and function return value types in a way that can be understood by humans, static type checkers like [mypy](#), code editors like VSCode, documentation generators like Sphinx, and other tools.

Static type checkers can use type hints to detect certain classes of errors by inspection. Code editors and IDEs can use type hints to provide better auto-completion and documentation generators can use type hints to generate better documentation.

These capabilities make it easier to develop a defect-free RDFLib and they also make it easier for users of RDFLib who can now use static type checkers to detect type errors in code that uses RDFLib.

### 4.5.2 Gradual Typing Process

Type hints are being added to RDFLib through a process called [gradual typing](#). This process involves adding type hints to some parts of RDFLib while leaving the rest without type hints. Gradual typing is being applied to many, long-lived, Python code bases.

This process is beneficial in that we can realize some of the benefits of type hints without requiring that the whole codebase have type hints.

### 4.5.3 Intended Type Hints

The intent is to have type hints in place for all of RDFLib and to have these type hints be as accurate as possible.

The accuracy of type hints is determined by both the standards that RDFLib aims to conform to, like RDF 1.1, and the deliberate choices that are made when implementing RDFLib. For example, given that the RDF 1.1 specification stipulates that the subject of an RDF triple cannot be a literal, all functions that accept an *RDF term* to be used as the subject of a triple should have type hints which excludes values that are literals.

There may be cases where some functionality of RDFLib may work perfectly well with values of types that are excluded by the type hints, but if these additional types violate the relevant standards we will consider the correct type hints to be those that exclude values of these types.



### 4.5.4 Public Type Aliases

In python, type hints are specified in annotations. Type hints are different from type aliases which are normal python variables that are not intended to provide runtime utility and are instead intended for use in static type checking.

For clarity, the following is an example of a function `foo` with type hints:

```
def foo(a: int) -> int:
    return a + 1
```

In the function `foo`, the input variable `a` is indicated to be of type `int` and the function is indicated to return an `int`.

The following is an example of a type alias `Bar`:

```
from typing import Tuple

Bar = Tuple[int, str]
```

RDFLib will provide public type aliases under the `rdflib.typing` package, for example, `rdflib.typing.Triple`, `rdflib.typing.Quad`. Type aliases in the rest of RDFLib should be private (i.e. being with an underscore).

### 4.5.5 Versioning, Compatibility and Stability

RDFLib attempts to adhere to [semver 2.0](#) which is concerned with the public API of software.

Ignoring type hints, the public API of RDFLib exists implicitly as a consequence of the code of RDFLib and the actual behaviour this entails, the relevant standards that RDFLib is trying to implement, and the documentation of RDFLib, with some interplay between all three of these. RDFLib's public API includes public type aliases, as these are normal python variables and not annotations.

Type hints attempt to formally document RDFLib's implicitly-defined public API in a machine-readable fashion as accurately and correctly as possible within the framework outline earlier in this document.

Type hints do not affect the runtime API or behaviour of RDFLib. In this way then, they are somewhat outside of the scope of semver, however, they still have an impact on the users of RDFLib, even if this impact is not at runtime, but during development. This necessitates some clarity as to what users of RDFLib should expect regarding type hints in RDFLib releases.

Changes to type hints can broadly be classified as follow:

#### Type Declaration

Adding type hints to existing code that had no explicit type hints, for example, changing

```
def foo(val):
    return val + 1
```

to

```
def foo(val: int) -> int:
    return val + 1
```

#### Type Refinement

Refining existing type hints to be narrower, for example, changing a type hint of `typing.Collection` to `typing.Sequence`.

#### Type Corrections

Correcting existing type hints which contradict the behaviour of the code or relevant specifications, for example, changing `typing.Sequence` from `typing.Set`

Given semver version components MAJOR.MINOR.PATCH, RDFLib will attempt to constrain type hint changes as follow:

Version Component	Type Declaration	Type Refinement	Type Corrections
MAJOR	YES	YES	YES
MINOR	YES	YES	YES
PATCH	NO	NO	YES

**Caution:** A caveat worth nothing here is that code that passed type validation on one version of RDFLib can fail type validation on a later version of RDFLib that only differs in PATCH version component. This is as a consequence of potential *Type Corrections*.

## 4.6 RFLib Contributing Guide

Thank you for considering contributing to RFLib. This project has no formal funding or full-time maintainers and relies entirely on independent contributors to keep it alive and relevant.

### 4.6.1 Ways to contribute

Some ways in which you can contribute to RFLib are:

- Address open issues:
- Fix `expected failure` tests.
- Add additional `expected failure` tests for open issues:
- Add tests for untested code:
- Review pull requests marked with the
- Answer questions on Stack Overflow:
- Convert `unittest` based tests to `pytest` based tests:
- Add, correct or improve docstrings:
- Update the RFLib Wikipedia entry:
- Update the RFLib Wikidata entry:
- Participate on Gitter/Matrix chat:
- Participate in GitHub discussions:
- Fix flake8 failures.

## 4.6.2 Pull Requests

Contributions that involve changes to the RDFLib repository have to be made with pull requests and should follow the *RDFLib developers guide*.

## 4.6.3 Code of Conduct

All contributions to the project should be consistent with the *code of conduct* adopted by RDFLib.

## 4.7 Decision Records

To ensure that significant changes to RDFLib are made with sufficient consultation, consideration and planning they should be preceded by a decision record that captures the particulars of the decision that lead to the change.

Decision records present the users and maintainers of RDFLib with an opportunity to review decisions before effort is expended to implement the decision in code, and it also makes it possible to review decisions without having to reconstruct them from the code changes that implement them.

Whether a change is significant is hard to measure objectively, but some characteristics that may indicate that a change is significant include:

- It will require changes to code that use RDFLib.
- It cannot be reversed without requiring changes to code that use RDFLib.
- It is onerous to reverse later.
- It increases the maintenance burden of RDFLib.
- It is very large.

Some of these characteristics are not binary but measured in degrees, so some discretion is required when determining if an architectural decision record is appropriate.

Decision records may also be used for changes that do not have any of the listed characteristics if a decision record would be otherwise helpful, for example to capture a decision to change the maintenance process of RDFLib.

Changes not preceded by decision records won't be rejected solely on this basis even if they are deemed significant, and decision records may also be created retrospectively for changes.

Decision records as described here are similar to the concept of [Architectural Decision Records](#), though it is slightly broader as it could include decisions which are not classified as architectural.

### 4.7.1 Creating a decision record

Decision records should be added to the RDFLib repository in the `./docs/decisions/` directory with a name `{YYYYmmdd}-{title}.rst`.

The content of the decision record should succinctly describe the context of the decision, the decision itself, and the status of the decision.

Decision records should preferably follow [Michael Nygard decision record template](#) that he described in a [2011 article](#) on documenting architecture decisions.

For questions about decision records please reach out to the RDFLib maintainers and community using the options given in [Further help & Contact](#).

## 4.7.2 Decision list

### Default Branch Name

---

#### Status

Accepted

---

#### Context

In recent years usage of the word `master` has become somewhat controversial [[SFC-BNAMING](#)] and consequently default branch name of Git repos has become `main`, both in Git itself [[SFC-BNAMING](#)] and in Git hosting solutions such as GitHub [[GH-BRANCHES](#)].

#### Decision

RDFLib's default branch will be renamed from `master` to `main`. This is primarily to stay in line with modern conventions and to adhere to the principle of least surprise.

#### Consequences

Anticipated negative consequences:

- Some links to old code will be broken.
- Some people's workflow may break unexpectedly and need adjusting.
- Any code and systems reliant on the old default branch name will fail.

Anticipated positive consequences:

- It will become a bit easier to work with RDFLib for developers that are used to `main` as the default branch.

#### References

## **SOURCE CODE**

The rdflib source code is hosted on GitHub at <https://github.com/RDFLib/rdflib> where you can lodge Issues and create Pull Requests to help improve this community project!

The RDFlib organisation on GitHub at <https://github.com/RDFLib> maintains this package and a number of other RDF and RDFlib-related packages that you might also find useful.



## **FURTHER HELP & CONTACT**

If you would like help with using RDFlib, rather than developing it, please post a question on StackOverflow using the tag `[rdflib]`. A list of existing `[rdflib]` tagged questions can be found [here](#).

You might also like to join RDFlib's [dev mailing list](#) or use RDFLib's [GitHub discussions section](#).

The chat is available at [gitter](#) or via matrix `#RDFLib_rdfli:gitter.im`.





## BIBLIOGRAPHY

[GH-BRANCHES] [GitHub: About the default branch](#)

[SFC-BNAMING] [Regarding Git and Branch Naming](#)



## PYTHON MODULE INDEX

### e

- `examples.berkeleydb_example`, 23
- `examples.conjunctive_graphs`, 22
- `examples.custom_datatype`, 22
- `examples.custom_eval`, 22
- `examples.foafpaths`, 22
- `examples.prepared_query`, 23
- `examples.resource_example`, 23
- `examples.secure_with_audit`, 25
- `examples.secure_with_urlopen`, 26
- `examples.slice`, 24
- `examples.smushing`, 24
- `examples.sparql_query_example`, 24
- `examples.sparql_update_example`, 24
- `examples.sparqlstore_example`, 24
- `examples.swap_primer`, 25
- `examples.transitive`, 25

### r

- `rdflib`, 334
- `rdflib.collection`, 221
- `rdflib.compare`, 226
- `rdflib.compat`, 230
- `rdflib.container`, 230
- `rdflib.events`, 233
- `rdflib.exceptions`, 235
- `rdflib.extras`, 69
  - `rdflib.extras.cmdlineutils`, 47
  - `rdflib.extras.describer`, 47
  - `rdflib.extras.external_graph_libs`, 51
  - `rdflib.extras.infixowl`, 55
- `rdflib.graph`, 236
- `rdflib.namespaces`, 69
- `rdflib.parser`, 276
- `rdflib.paths`, 279
- `rdflib.plugin`, 288
- `rdflib.plugins`, 218
  - `rdflib.plugins.parsers`, 113
    - `rdflib.plugins.parsers.hexst`, 79
    - `rdflib.plugins.parsers.jsonld`, 80
    - `rdflib.plugins.parsers.notation3`, 81
    - `rdflib.plugins.parsers.nquads`, 97

- `rdflib.plugins.parsers.ntriples`, 98
- `rdflib.plugins.parsers.RDFVOC`, 79
- `rdflib.plugins.parsers.rdfxml`, 102
- `rdflib.plugins.parsers.trig`, 108
- `rdflib.plugins.parsers.trix`, 110
- `rdflib.plugins.serializers`, 124
  - `rdflib.plugins.serializers.hexst`, 113
  - `rdflib.plugins.serializers.jsonld`, 114
  - `rdflib.plugins.serializers.longturtle`, 115
  - `rdflib.plugins.serializers.n3`, 116
  - `rdflib.plugins.serializers.nquads`, 117
  - `rdflib.plugins.serializers.nt`, 118
  - `rdflib.plugins.serializers.rdfxml`, 118
  - `rdflib.plugins.serializers.trig`, 120
  - `rdflib.plugins.serializers.trix`, 120
  - `rdflib.plugins.serializers.turtle`, 121
  - `rdflib.plugins.serializers.xmlwriter`, 123
- `rdflib.plugins.shared`, 131
  - `rdflib.plugins.shared.jsonld`, 131
    - `rdflib.plugins.shared.jsonld.context`, 124
    - `rdflib.plugins.shared.jsonld.errors`, 130
    - `rdflib.plugins.shared.jsonld.keys`, 130
    - `rdflib.plugins.shared.jsonld.util`, 130
- `rdflib.plugins.sparql`, 188
  - `rdflib.plugins.sparql.aggregates`, 139
  - `rdflib.plugins.sparql.algebra`, 145
  - `rdflib.plugins.sparql.datatypes`, 152
  - `rdflib.plugins.sparql.evaluate`, 152
  - `rdflib.plugins.sparql.evalutils`, 156
  - `rdflib.plugins.sparql.operators`, 156
  - `rdflib.plugins.sparql.parser`, 168
  - `rdflib.plugins.sparql.parserutils`, 169
  - `rdflib.plugins.sparql.processor`, 174
  - `rdflib.plugins.sparql.results`, 139
    - `rdflib.plugins.sparql.results.csvresults`, 131
    - `rdflib.plugins.sparql.results.graph`, 133
    - `rdflib.plugins.sparql.results.jsonresults`, 133
    - `rdflib.plugins.sparql.results.rdfresults`, 135
    - `rdflib.plugins.sparql.results.tsvresults`, 135
    - `rdflib.plugins.sparql.results.txtresults`, 136
    - `rdflib.plugins.sparql.results.xmlresults`, 136

- `rdflib.plugins.sparql.sparql`, 176
- `rdflib.plugins.sparql.update`, 185
- `rdflib.plugins.stores`, 218
  - `rdflib.plugins.stores.auditable`, 189
  - `rdflib.plugins.stores.berkeleydb`, 192
  - `rdflib.plugins.stores.concurrent`, 195
  - `rdflib.plugins.stores.memory`, 196
  - `rdflib.plugins.stores.regexmatching`, 201
  - `rdflib.plugins.stores.sparqlconnector`, 203
  - `rdflib.plugins.stores.sparqlstore`, 205
- `rdflib.query`, 291
- `rdflib.resource`, 298
- `rdflib.serializer`, 305
- `rdflib.store`, 306
- `rdflib.term`, 313
- `rdflib.tools`, 221
  - `rdflib.tools.chunk_serializer`, 218
  - `rdflib.tools.csv2rdf`, 219
  - `rdflib.tools.defined_namespace_creator`, 219
  - `rdflib.tools.graphisomorphism`, 220
  - `rdflib.tools.rdf2dot`, 221
  - `rdflib.tools.rdfpipe`, 221
  - `rdflib.tools.rdfs2dot`, 221
- `rdflib.util`, 330
- `rdflib.void`, 334

## Symbols

- `__abs__()` (*rdflib.Literal* method), 443
- `__abs__()` (*rdflib.term.Literal* method), 319
- `__abstractmethods__` (*rdflib.plugins.sparql.parserutils.Comp* attribute), 169
- `__abstractmethods__` (*rdflib.plugins.sparql.parserutils.Param* attribute), 171
- `__abstractmethods__` (*rdflib.plugins.sparql.parserutils.ParamList* attribute), 172
- `__abstractmethods__` (*rdflib.plugins.sparql.sparql.Bindings* attribute), 176
- `__abstractmethods__` (*rdflib.plugins.sparql.sparql.FrozenBindings* attribute), 178
- `__abstractmethods__` (*rdflib.plugins.sparql.sparql.FrozenDict* attribute), 179
- `__add__()` (*rdflib.Graph* method), 423
- `__add__()` (*rdflib.Literal* method), 444
- `__add__()` (*rdflib.URIRef* method), 604
- `__add__()` (*rdflib.graph.Graph* method), 250
- `__add__()` (*rdflib.term.Literal* method), 319
- `__add__()` (*rdflib.term.URIRef* method), 327
- `__and__()` (*rdflib.Graph* method), 423
- `__and__()` (*rdflib.extras.infixowl.Class* method), 60
- `__and__()` (*rdflib.graph.Graph* method), 250
- `__annotations__` (*rdflib.BNode* attribute), 335
- `__annotations__` (*rdflib.ConjunctiveGraph* attribute), 403
- `__annotations__` (*rdflib.Dataset* attribute), 418
- `__annotations__` (*rdflib.Graph* attribute), 424
- `__annotations__` (*rdflib.IdentifiedNode* attribute), 441
- `__annotations__` (*rdflib.Literal* attribute), 444
- `__annotations__` (*rdflib.URIRef* attribute), 604
- `__annotations__` (*rdflib.Variable* attribute), 608
- `__annotations__` (*rdflib.namespace.ClosedNamespace* attribute), 71
- `__annotations__` (*rdflib.namespace.Namespace* attribute), 72
- `__annotations__` (*rdflib.parser.URLInputSource* attribute), 278
- `__annotations__` (*rdflib.paths.Path* attribute), 284
- `__annotations__` (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* attribute), 192
- `__annotations__` (*rdflib.query.ResultRow* attribute), 295
- `__annotations__` (*rdflib.store.Store* attribute), 307
- `__annotations__` (*rdflib.term.Literal* attribute), 319
- `__annotations__` (*rdflib.term.URIRef* attribute), 327
- `__bool__()` (*rdflib.Literal* method), 444
- `__bool__()` (*rdflib.query.Result* method), 292
- `__bool__()` (*rdflib.term.Literal* method), 319
- `__call__()` (*rdflib.extras.infixowl.Infix* method), 65
- `__cmp__()` (*rdflib.Graph* method), 424
- `__cmp__()` (*rdflib.graph.Graph* method), 250
- `__cmp__()` (*rdflib.graph.ReadOnlyGraphAggregate* method), 269
- `__contains__()` (*rdflib.ConjunctiveGraph* method), 404
- `__contains__()` (*rdflib.Graph* method), 424
- `__contains__()` (*rdflib.Namespace* method), 451
- `__contains__()` (*rdflib.extras.infixowl.OWLRLDFListProxy* method), 66
- `__contains__()` (*rdflib.graph.ConjunctiveGraph* method), 242
- `__contains__()` (*rdflib.graph.Graph* method), 250
- `__contains__()` (*rdflib.graph.ReadOnlyGraphAggregate* method), 270
- `__contains__()` (*rdflib.namespace.ClosedNamespace* method), 71
- `__contains__()` (*rdflib.namespace.Namespace* method), 72
- `__contains__()` (*rdflib.namespace.NamespaceManager* method), 75
- `__contains__()` (*rdflib.plugins.sparql.sparql.Bindings* method), 176
- `__del__()` (*rdflib.plugins.stores.concurrent.ResponsibleGenerator* method), 196
- `__delitem__()` (*rdflib.collection.Collection* method),

- 222
- `__delitem__()` (*rdflib.container.Container* method), 231
- `__delitem__()` (*rdflib.extras.infixowl.OWL RDFListProxy* method), 66
- `__delitem__()` (*rdflib.plugins.sparql.sparql.Bindings* method), 176
- `__dict__` (*rdflib.Graph* attribute), 424
- `__dict__` (*rdflib.IdentifiedNode* attribute), 441
- `__dict__` (*rdflib.Namespace* attribute), 452
- `__dict__` (*rdflib.collection.Collection* attribute), 223
- `__dict__` (*rdflib.container.Container* attribute), 231
- `__dict__` (*rdflib.events.Dispatcher* attribute), 234
- `__dict__` (*rdflib.events.Event* attribute), 234
- `__dict__` (*rdflib.extras.describer.Describer* attribute), 49
- `__dict__` (*rdflib.extras.infixowl.Callable* attribute), 60
- `__dict__` (*rdflib.extras.infixowl.Individual* attribute), 64
- `__dict__` (*rdflib.extras.infixowl.Infix* attribute), 65
- `__dict__` (*rdflib.extras.infixowl.OWL RDFListProxy* attribute), 66
- `__dict__` (*rdflib.graph.BatchAddGraph* attribute), 240
- `__dict__` (*rdflib.graph.Graph* attribute), 251
- `__dict__` (*rdflib.graph.Seq* attribute), 274
- `__dict__` (*rdflib.namespace.Namespace* attribute), 73
- `__dict__` (*rdflib.namespace.NamespaceManager* attribute), 75
- `__dict__` (*rdflib.paths.Path* attribute), 285
- `__dict__` (*rdflib.paths.PathList* attribute), 286
- `__dict__` (*rdflib.plugin.Plugin* attribute), 289
- `__dict__` (*rdflib.plugins.parsers.hexst.HexuplesParser* attribute), 79
- `__dict__` (*rdflib.plugins.parsers.jsonld.JsonLDParse* attribute), 80
- `__dict__` (*rdflib.plugins.parsers.notation3.Formula* attribute), 82
- `__dict__` (*rdflib.plugins.parsers.notation3.RDFSink* attribute), 84
- `__dict__` (*rdflib.plugins.parsers.notation3.SinkParser* attribute), 87
- `__dict__` (*rdflib.plugins.parsers.notation3.TurtleParser* attribute), 95
- `__dict__` (*rdflib.plugins.parsers.nquads.NQuadsParser* attribute), 97
- `__dict__` (*rdflib.plugins.parsers.ntriples.DummySink* attribute), 98
- `__dict__` (*rdflib.plugins.parsers.rdfxml.RDFXMLParser* attribute), 107
- `__dict__` (*rdflib.plugins.parsers.trig.TrigParser* attribute), 108
- `__dict__` (*rdflib.plugins.parsers.trix.TriXParser* attribute), 112
- `__dict__` (*rdflib.plugins.serializers.xmlwriter.XMLWriter* attribute), 123
- `__dict__` (*rdflib.plugins.shared.jsonld.context.Context* attribute), 124
- `__dict__` (*rdflib.plugins.shared.jsonld.context.Defined* attribute), 129
- `__dict__` (*rdflib.plugins.sparql.aggregates.Accumulator* attribute), 139
- `__dict__` (*rdflib.plugins.sparql.aggregates.Aggregator* attribute), 140
- `__dict__` (*rdflib.plugins.sparql.parserutils.ParamValue* attribute), 172
- `__dict__` (*rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter* attribute), 136
- `__dict__` (*rdflib.plugins.sparql.sparql.Bindings* attribute), 176
- `__dict__` (*rdflib.plugins.sparql.sparql.FrozenDict* attribute), 179
- `__dict__` (*rdflib.plugins.sparql.sparql.Prologue* attribute), 181
- `__dict__` (*rdflib.plugins.sparql.sparql.Query* attribute), 182
- `__dict__` (*rdflib.plugins.sparql.sparql.QueryContext* attribute), 182
- `__dict__` (*rdflib.plugins.sparql.sparql.Update* attribute), 185
- `__dict__` (*rdflib.plugins.stores.concurrent.ConcurrentStore* attribute), 195
- `__dict__` (*rdflib.plugins.stores.regexmatching.REGEXTerm* attribute), 203
- `__dict__` (*rdflib.plugins.stores.sparqlconnector.SPARQLConnector* attribute), 203
- `__dict__` (*rdflib.query.EncodeOnlyUnicode* attribute), 291
- `__dict__` (*rdflib.query.Processor* attribute), 291
- `__dict__` (*rdflib.query.Result* attribute), 292
- `__dict__` (*rdflib.query.ResultParser* attribute), 294
- `__dict__` (*rdflib.query.ResultRow* attribute), 295
- `__dict__` (*rdflib.query.ResultSerializer* attribute), 297
- `__dict__` (*rdflib.query.UpdateProcessor* attribute), 297
- `__dict__` (*rdflib.resource.Resource* attribute), 303
- `__dict__` (*rdflib.serializer.Serializer* attribute), 305
- `__dict__` (*rdflib.store.NodePickler* attribute), 306
- `__dict__` (*rdflib.store.Store* attribute), 307
- `__dict__` (*rdflib.term.IdentifiedNode* attribute), 314
- `__dict__` (*rdflib.tools.csv2rdf.CSV2RDF* attribute), 219
- `__dir__()` (*rdflib.namespace.ClosedNamespace* method), 71
- `__enter__()` (*rdflib.graph.BatchAddGraph* method), 241
- `__eq__()` (*rdflib.Graph* method), 425
- `__eq__()` (*rdflib.Literal* method), 444
- `__eq__()` (*rdflib.compare.IsomorphicGraph* method), 228
- `__eq__()` (*rdflib.extras.infixowl.Class* method), 61
- `__eq__()` (*rdflib.extras.infixowl.OWL RDFListProxy* method), 66

- method), 66
- \_\_eq\_\_() (rdflib.extras.infixowl.Restriction method), 68
- \_\_eq\_\_() (rdflib.graph.Graph method), 252
- \_\_eq\_\_() (rdflib.paths.Path method), 285
- \_\_eq\_\_() (rdflib.query.Result method), 293
- \_\_eq\_\_() (rdflib.resource.Resource method), 304
- \_\_eq\_\_() (rdflib.term.Identifier method), 315
- \_\_eq\_\_() (rdflib.term.Literal method), 319
- \_\_eq\_\_() (rdflib.tools.graphisomorphism.IsomorphicTestableGraph method), 129
- method), 220
- \_\_exit\_\_() (rdflib.graph.BatchAddGraph method), 241
- \_\_ge\_\_() (rdflib.Graph method), 426
- \_\_ge\_\_() (rdflib.Literal method), 445
- \_\_ge\_\_() (rdflib.graph.Graph method), 253
- \_\_ge\_\_() (rdflib.paths.Path method), 285
- \_\_ge\_\_() (rdflib.resource.Resource method), 304
- \_\_ge\_\_() (rdflib.term.Identifier method), 315
- \_\_ge\_\_() (rdflib.term.Literal method), 320
- \_\_getattr\_\_() (rdflib.Namespace method), 452
- \_\_getattr\_\_() (rdflib.extras.infixowl.ClassNamespaceFactory method), 62
- \_\_getattr\_\_() (rdflib.namespace.ClosedNamespace method), 71
- \_\_getattr\_\_() (rdflib.namespace.Namespace method), 73
- \_\_getattr\_\_() (rdflib.plugins.sparql.parserutils.CompValue method), 170
- \_\_getattr\_\_() (rdflib.query.EncodeOnlyUnicode method), 291
- \_\_getattr\_\_() (rdflib.query.Result method), 293
- \_\_getattr\_\_() (rdflib.query.ResultRow method), 296
- \_\_getitem\_\_() (rdflib.Graph method), 426
- \_\_getitem\_\_() (rdflib.Namespace method), 452
- \_\_getitem\_\_() (rdflib.collection.Collection method), 224
- \_\_getitem\_\_() (rdflib.container.Container method), 232
- \_\_getitem\_\_() (rdflib.extras.infixowl.ClassNamespaceFactory method), 62
- \_\_getitem\_\_() (rdflib.extras.infixowl.OWLRLDFListProxy method), 66
- \_\_getitem\_\_() (rdflib.graph.Graph method), 253
- \_\_getitem\_\_() (rdflib.graph.Seq method), 274
- \_\_getitem\_\_() (rdflib.namespace.ClosedNamespace method), 71
- \_\_getitem\_\_() (rdflib.namespace.Namespace method), 73
- \_\_getitem\_\_() (rdflib.plugins.sparql.parserutils.CompValue method), 170
- \_\_getitem\_\_() (rdflib.plugins.sparql.sparql.Bindings method), 177
- \_\_getitem\_\_() (rdflib.plugins.sparql.sparql.FrozenBindings method), 178
- \_\_getitem\_\_() (rdflib.plugins.sparql.sparql.FrozenDict method), 179
- method), 179
- \_\_getitem\_\_() (rdflib.plugins.sparql.sparql.QueryContext method), 182
- \_\_getitem\_\_() (rdflib.query.ResultRow method), 296
- \_\_getitem\_\_() (rdflib.resource.Resource method), 304
- \_\_getnewargs\_\_() (rdflib.IdentifiedNode method), 441
- \_\_getnewargs\_\_() (rdflib.plugins.shared.jsonld.context.Term method), 315
- \_\_getnewargs\_\_() (rdflib.term.IdentifiedNode method), 315
- \_\_getstate\_\_() (rdflib.Dataset method), 418
- \_\_getstate\_\_() (rdflib.Literal method), 445
- \_\_getstate\_\_() (rdflib.graph.Dataset method), 247
- \_\_getstate\_\_() (rdflib.store.NodePickler method), 306
- \_\_getstate\_\_() (rdflib.term.Literal method), 320
- \_\_gt\_\_() (rdflib.Graph method), 426
- \_\_gt\_\_() (rdflib.Literal method), 445
- \_\_gt\_\_() (rdflib.graph.Graph method), 253
- \_\_gt\_\_() (rdflib.paths.Path method), 285
- \_\_gt\_\_() (rdflib.resource.Resource method), 304
- \_\_gt\_\_() (rdflib.term.Identifier method), 316
- \_\_gt\_\_() (rdflib.term.Literal method), 320
- \_\_hash\_\_() (rdflib.extras.infixowl.OWLRLDFListProxy attribute), 66
- \_\_hash\_\_() (rdflib.query.Result attribute), 293
- \_\_hash\_\_() (rdflib.tools.graphisomorphism.IsomorphicTestableGraph attribute), 220
- \_\_hash\_\_() (rdflib.Graph method), 426
- \_\_hash\_\_() (rdflib.Literal method), 446
- \_\_hash\_\_() (rdflib.compare.IsomorphicGraph method), 228
- \_\_hash\_\_() (rdflib.extras.infixowl.Class method), 61
- \_\_hash\_\_() (rdflib.extras.infixowl.Restriction method), 68
- \_\_hash\_\_() (rdflib.graph.Graph method), 253
- \_\_hash\_\_() (rdflib.graph.ReadOnlyGraphAggregate method), 270
- \_\_hash\_\_() (rdflib.paths.Path method), 285
- \_\_hash\_\_() (rdflib.plugins.sparql.sparql.FrozenDict method), 179
- \_\_hash\_\_() (rdflib.resource.Resource method), 304
- \_\_hash\_\_() (rdflib.term.Identifier method), 316
- \_\_hash\_\_() (rdflib.term.Literal method), 321
- \_\_iadd\_\_() (rdflib.Graph method), 427
- \_\_iadd\_\_() (rdflib.collection.Collection method), 224
- \_\_iadd\_\_() (rdflib.extras.infixowl.Class method), 61
- \_\_iadd\_\_() (rdflib.extras.infixowl.OWLRLDFListProxy method), 66
- \_\_iadd\_\_() (rdflib.graph.Graph method), 254
- \_\_iadd\_\_() (rdflib.graph.ReadOnlyGraphAggregate method), 270
- \_\_init\_\_() (rdflib.ConjunctiveGraph method), 404
- \_\_init\_\_() (rdflib.Dataset method), 418



- `__init__()` (*rdflib.Graph* method), 427
- `__init__()` (*rdflib.collection.Collection* method), 224
- `__init__()` (*rdflib.compare.IsomorphicGraph* method), 228
- `__init__()` (*rdflib.container.Alt* method), 230
- `__init__()` (*rdflib.container.Bag* method), 230
- `__init__()` (*rdflib.container.Container* method), 232
- `__init__()` (*rdflib.container.NoElementException* method), 233
- `__init__()` (*rdflib.container.Seq* method), 233
- `__init__()` (*rdflib.events.Event* method), 234
- `__init__()` (*rdflib.exceptions.Error* method), 235
- `__init__()` (*rdflib.exceptions.ParserError* method), 235
- `__init__()` (*rdflib.exceptions.UniquenessError* method), 235
- `__init__()` (*rdflib.extras.describer.Describer* method), 49
- `__init__()` (*rdflib.extras.infixowl.AnnotatableTerms* method), 58
- `__init__()` (*rdflib.extras.infixowl.BooleanClass* method), 59
- `__init__()` (*rdflib.extras.infixowl.Callable* method), 60
- `__init__()` (*rdflib.extras.infixowl.Class* method), 61
- `__init__()` (*rdflib.extras.infixowl.EnumeratedClass* method), 64
- `__init__()` (*rdflib.extras.infixowl.Individual* method), 65
- `__init__()` (*rdflib.extras.infixowl.Infix* method), 65
- `__init__()` (*rdflib.extras.infixowl.MalformedClassError* method), 66
- `__init__()` (*rdflib.extras.infixowl.OWL RDFListProxy* method), 66
- `__init__()` (*rdflib.extras.infixowl.Ontology* method), 67
- `__init__()` (*rdflib.extras.infixowl.Property* method), 67
- `__init__()` (*rdflib.extras.infixowl.Restriction* method), 68
- `__init__()` (*rdflib.graph.BatchAddGraph* method), 241
- `__init__()` (*rdflib.graph.ConjunctiveGraph* method), 242
- `__init__()` (*rdflib.graph.Dataset* method), 248
- `__init__()` (*rdflib.graph.Graph* method), 254
- `__init__()` (*rdflib.graph.ModificationException* method), 268
- `__init__()` (*rdflib.graph.QuotedGraph* method), 268
- `__init__()` (*rdflib.graph.ReadOnlyGraphAggregate* method), 270
- `__init__()` (*rdflib.graph.Seq* method), 275
- `__init__()` (*rdflib.graph.UnSupportedAggregateOperation* method), 275
- `__init__()` (*rdflib.namespace.NamespaceManager* method), 76
- `__init__()` (*rdflib.parser.FileInputSource* method), 276
- `__init__()` (*rdflib.parser.InputSource* method), 276
- `__init__()` (*rdflib.parser.Parser* method), 276
- `__init__()` (*rdflib.parser.PythonInputSource* method), 277
- `__init__()` (*rdflib.parser.StringInputSource* method), 278
- `__init__()` (*rdflib.parser.URLInputSource* method), 278
- `__init__()` (*rdflib.paths.AlternativePath* method), 282
- `__init__()` (*rdflib.paths.InvPath* method), 283
- `__init__()` (*rdflib.paths.MulPath* method), 283
- `__init__()` (*rdflib.paths.NegatedPath* method), 284
- `__init__()` (*rdflib.paths.SequencePath* method), 287
- `__init__()` (*rdflib.plugin.PKGPlugin* method), 289
- `__init__()` (*rdflib.plugin.Plugin* method), 289
- `__init__()` (*rdflib.plugins.parsers.hextr.HextuplesParser* method), 79
- `__init__()` (*rdflib.plugins.parsers.jsonld.JsonLdParser* method), 80
- `__init__()` (*rdflib.plugins.parsers.notation3.BadSyntax* method), 82
- `__init__()` (*rdflib.plugins.parsers.notation3.Formula* method), 82
- `__init__()` (*rdflib.plugins.parsers.notation3.N3Parser* method), 83
- `__init__()` (*rdflib.plugins.parsers.notation3.RDFSink* method), 84
- `__init__()` (*rdflib.plugins.parsers.notation3.SinkParser* method), 87
- `__init__()` (*rdflib.plugins.parsers.notation3.TurtleParser* method), 95
- `__init__()` (*rdflib.plugins.parsers.ntriples.DummySink* method), 98
- `__init__()` (*rdflib.plugins.parsers.ntriples.NTGraphSink* method), 99
- `__init__()` (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* method), 100
- `__init__()` (*rdflib.plugins.parsers.rdfxml.BagID* method), 102
- `__init__()` (*rdflib.plugins.parsers.rdfxml.ElementHandler* method), 102
- `__init__()` (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 103
- `__init__()` (*rdflib.plugins.parsers.rdfxml.RDFXMLParser* method), 108
- `__init__()` (*rdflib.plugins.parsers.trig.TrigParser* method), 108
- `__init__()` (*rdflib.plugins.parsers.trix.TriXHandler* method), 110
- `__init__()` (*rdflib.plugins.parsers.trix.TriXParser* method), 113
- `__init__()` (*rdflib.plugins.serializers.hextr.HextuplesSerializer* method), 113
- `__init__()` (*rdflib.plugins.serializers.jsonld.JsonLdSerializer* method), 114
- `__init__()` (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 114



method), 115

`__init__` () (rdflib.plugins.serializers.n3.N3Serializer method), 116

`__init__` () (rdflib.plugins.serializers.nquads.NQuadsSerializer method), 117

`__init__` () (rdflib.plugins.serializers.nt.NTSerializer method), 118

`__init__` () (rdflib.plugins.serializers.rdfxml.PrettyXMLSerializer method), 118

`__init__` () (rdflib.plugins.serializers.rdfxml.XMLSerializer method), 119

`__init__` () (rdflib.plugins.serializers.trig.TrigSerializer method), 120

`__init__` () (rdflib.plugins.serializers.trix.TriXSerializer method), 120

`__init__` () (rdflib.plugins.serializers.turtle.RecursiveSerializer method), 121

`__init__` () (rdflib.plugins.serializers.turtle.TurtleSerializer method), 122

`__init__` () (rdflib.plugins.serializers.xmlwriter.XMLWriter method), 123

`__init__` () (rdflib.plugins.shared.jsonld.context.Context method), 125

`__init__` () (rdflib.plugins.sparql.aggregates.Accumulator method), 140

`__init__` () (rdflib.plugins.sparql.aggregates.Aggregator method), 141

`__init__` () (rdflib.plugins.sparql.aggregates.Average method), 141

`__init__` () (rdflib.plugins.sparql.aggregates.Counter method), 142

`__init__` () (rdflib.plugins.sparql.aggregates.Extremum method), 143

`__init__` () (rdflib.plugins.sparql.aggregates.GroupConcat method), 143

`__init__` () (rdflib.plugins.sparql.aggregates.Sample method), 144

`__init__` () (rdflib.plugins.sparql.aggregates.Sum method), 145

`__init__` () (rdflib.plugins.sparql.algebra.StopTraversal method), 147

`__init__` () (rdflib.plugins.sparql.parserutils.Comp method), 169

`__init__` () (rdflib.plugins.sparql.parserutils.CompValue method), 170

`__init__` () (rdflib.plugins.sparql.parserutils.Expr method), 171

`__init__` () (rdflib.plugins.sparql.parserutils.Param method), 171

`__init__` () (rdflib.plugins.sparql.parserutils.ParamList method), 172

`__init__` () (rdflib.plugins.sparql.parserutils.ParamValue method), 172

`__init__` () (rdflib.plugins.sparql.processor.SPARQLProcessor method), 174

`__init__` () (rdflib.plugins.sparql.processor.SPARQLResult method), 174

`__init__` () (rdflib.plugins.sparql.processor.SPARQLUpdateProcessor method), 175

`__init__` () (rdflib.plugins.sparql.results.csvresults.CSVResultParser method), 132

`__init__` () (rdflib.plugins.sparql.results.csvresults.CSVResultSerializer method), 132

`__init__` () (rdflib.plugins.sparql.results.jsonresults.JSONResult method), 133

`__init__` () (rdflib.plugins.sparql.results.jsonresults.JSONResultSerializer method), 134

`__init__` () (rdflib.plugins.sparql.results.rdfresults.RDFResult method), 135

`__init__` () (rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter method), 137

`__init__` () (rdflib.plugins.sparql.results.xmlresults.XMLResult method), 138

`__init__` () (rdflib.plugins.sparql.results.xmlresults.XMLResultSerializer method), 138

`__init__` () (rdflib.plugins.sparql.sparql.AlreadyBound method), 176

`__init__` () (rdflib.plugins.sparql.sparql.Bindings method), 177

`__init__` () (rdflib.plugins.sparql.sparql.FrozenBindings method), 178

`__init__` () (rdflib.plugins.sparql.sparql.FrozenDict method), 179

`__init__` () (rdflib.plugins.sparql.sparql.NotBoundError method), 180

`__init__` () (rdflib.plugins.sparql.sparql.Prologue method), 181

`__init__` () (rdflib.plugins.sparql.sparql.Query method), 182

`__init__` () (rdflib.plugins.sparql.sparql.QueryContext method), 183

`__init__` () (rdflib.plugins.sparql.sparql.SPARQLError method), 184

`__init__` () (rdflib.plugins.sparql.sparql.SPARQLTypeError method), 184

`__init__` () (rdflib.plugins.sparql.sparql.Update method), 185

`__init__` () (rdflib.plugins.stores.auditable.AuditableStore method), 189

`__init__` () (rdflib.plugins.stores.berkeleydb.BerkeleyDB method), 192

`__init__` () (rdflib.plugins.stores.concurrent.ConcurrentStore method), 195

`__init__` () (rdflib.plugins.stores.concurrent.ResponsibleGenerator method), 196

`__init__` () (rdflib.plugins.stores.memory.Memory method), 196

`__init__` () (rdflib.plugins.stores.memory.SimpleMemory method), 196

method), 199  
 \_\_init\_\_ () (rdflib.plugins.stores.regexmatching.REGEXMatching method), 201  
 \_\_init\_\_ () (rdflib.plugins.stores.regexmatching.REGEXTerm method), 203  
 \_\_init\_\_ () (rdflib.plugins.stores.sparqlconnector.SPARQLConnector method), 204  
 \_\_init\_\_ () (rdflib.plugins.stores.sparqlstore.SPARQLStore method), 206  
 \_\_init\_\_ () (rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore method), 212  
 \_\_init\_\_ () (rdflib.query.EncodeOnlyUnicode method), 291  
 \_\_init\_\_ () (rdflib.query.Processor method), 292  
 \_\_init\_\_ () (rdflib.query.Result method), 293  
 \_\_init\_\_ () (rdflib.query.ResultParser method), 294  
 \_\_init\_\_ () (rdflib.query.ResultSerializer method), 297  
 \_\_init\_\_ () (rdflib.query.UpdateProcessor method), 297  
 \_\_init\_\_ () (rdflib.resource.Resource method), 304  
 \_\_init\_\_ () (rdflib.serializer.Serializer method), 305  
 \_\_init\_\_ () (rdflib.store.NodePickler method), 306  
 \_\_init\_\_ () (rdflib.store.Store method), 308  
 \_\_init\_\_ () (rdflib.tools.csv2rdf.CSV2RDF method), 219  
 \_\_init\_\_ () (rdflib.tools.graphisomorphism.IsomorphicTestableGraph method), 220  
 \_\_invert\_\_ () (rdflib.Literal method), 446  
 \_\_invert\_\_ () (rdflib.URIRef method), 604  
 \_\_invert\_\_ () (rdflib.extras.infixowl.Class method), 61  
 \_\_invert\_\_ () (rdflib.paths.Path method), 285  
 \_\_invert\_\_ () (rdflib.term.Literal method), 321  
 \_\_invert\_\_ () (rdflib.term.URIRef method), 327  
 \_\_isub\_\_ () (rdflib.Graph method), 427  
 \_\_isub\_\_ () (rdflib.extras.infixowl.Class method), 61  
 \_\_isub\_\_ () (rdflib.graph.Graph method), 254  
 \_\_isub\_\_ () (rdflib.graph.ReadOnlyGraphAggregate method), 270  
 \_\_iter\_\_ () (rdflib.Dataset method), 418  
 \_\_iter\_\_ () (rdflib.Graph method), 427  
 \_\_iter\_\_ () (rdflib.collection.Collection method), 224  
 \_\_iter\_\_ () (rdflib.extras.infixowl.OWLRLDFListProxy method), 66  
 \_\_iter\_\_ () (rdflib.graph.Dataset method), 248  
 \_\_iter\_\_ () (rdflib.graph.Graph method), 254  
 \_\_iter\_\_ () (rdflib.graph.Seq method), 275  
 \_\_iter\_\_ () (rdflib.plugins.sparql.sparql.Bindings method), 177  
 \_\_iter\_\_ () (rdflib.plugins.sparql.sparql.FrozenDict method), 179  
 \_\_iter\_\_ () (rdflib.plugins.stores.concurrent.ResponsibleGenerator method), 196  
 \_\_iter\_\_ () (rdflib.query.Result method), 293  
 \_\_iter\_\_ () (rdflib.resource.Resource method), 304  
 \_\_le\_\_ () (rdflib.Graph method), 427  
 \_\_le\_\_ () (rdflib.Literal method), 446  
 \_\_le\_\_ () (rdflib.graph.Graph method), 254  
 \_\_le\_\_ () (rdflib.paths.Path method), 285  
 \_\_le\_\_ () (rdflib.resource.Resource method), 304  
 \_\_le\_\_ () (rdflib.term.Identifier method), 316  
 \_\_le\_\_ () (rdflib.term.Literal method), 322  
 \_\_len\_\_ () (rdflib.ConjunctiveGraph method), 404  
 \_\_len\_\_ () (rdflib.Graph method), 427  
 \_\_len\_\_ () (rdflib.collection.Collection method), 225  
 \_\_len\_\_ () (rdflib.container.Container method), 232  
 \_\_len\_\_ () (rdflib.extras.infixowl.OWLRLDFListProxy method), 66  
 \_\_len\_\_ () (rdflib.graph.ConjunctiveGraph method), 242  
 \_\_len\_\_ () (rdflib.graph.Graph method), 254  
 \_\_len\_\_ () (rdflib.graph.ReadOnlyGraphAggregate method), 270  
 \_\_len\_\_ () (rdflib.graph.Seq method), 275  
 \_\_len\_\_ () (rdflib.plugins.sparql.sparql.Bindings method), 177  
 \_\_len\_\_ () (rdflib.plugins.sparql.sparql.FrozenDict method), 179  
 \_\_len\_\_ () (rdflib.plugins.stores.auditable.AuditableStore method), 189  
 \_\_len\_\_ () (rdflib.plugins.stores.berkeleydb.BerkeleyDB method), 192  
 \_\_len\_\_ () (rdflib.plugins.stores.concurrent.ConcurrentStore method), 195  
 \_\_len\_\_ () (rdflib.plugins.stores.memory.Memory method), 196  
 \_\_len\_\_ () (rdflib.plugins.stores.memory.SimpleMemory method), 199  
 \_\_len\_\_ () (rdflib.plugins.stores.regexmatching.REGEXMatching method), 202  
 \_\_len\_\_ () (rdflib.plugins.stores.sparqlstore.SPARQLStore method), 206  
 \_\_len\_\_ () (rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore method), 213  
 \_\_len\_\_ () (rdflib.query.Result method), 293  
 \_\_len\_\_ () (rdflib.store.Store method), 308  
 \_\_lt\_\_ () (rdflib.Graph method), 427  
 \_\_lt\_\_ () (rdflib.Literal method), 447  
 \_\_lt\_\_ () (rdflib.graph.Graph method), 254  
 \_\_lt\_\_ () (rdflib.paths.Path method), 285  
 \_\_lt\_\_ () (rdflib.resource.Resource method), 304  
 \_\_lt\_\_ () (rdflib.term.Identifier method), 316  
 \_\_lt\_\_ () (rdflib.term.Literal method), 322  
 \_\_matmul\_\_ () (rdflib.extras.infixowl.Infix method), 65  
 \_\_mod\_\_ () (rdflib.URIRef method), 604  
 \_\_mod\_\_ () (rdflib.term.URIRef method), 327  
 \_\_module\_\_ (examples.secure\_with\_urlopen.SecuredHTTPHandler attribute), 26  
 \_\_module\_\_ (rdflib.BNode attribute), 335

- `__module__` (*rdflib.ConjunctiveGraph* attribute), 404
- `__module__` (*rdflib.Dataset* attribute), 418
- `__module__` (*rdflib.Graph* attribute), 428
- `__module__` (*rdflib.IdentifiedNode* attribute), 441
- `__module__` (*rdflib.Literal* attribute), 447
- `__module__` (*rdflib.Namespace* attribute), 452
- `__module__` (*rdflib.URIRef* attribute), 604
- `__module__` (*rdflib.Variable* attribute), 608
- `__module__` (*rdflib.collection.Collection* attribute), 225
- `__module__` (*rdflib.compare.IsomorphicGraph* attribute), 228
- `__module__` (*rdflib.container.Alt* attribute), 230
- `__module__` (*rdflib.container.Bag* attribute), 230
- `__module__` (*rdflib.container.Container* attribute), 232
- `__module__` (*rdflib.container.NoElementException* attribute), 233
- `__module__` (*rdflib.container.Seq* attribute), 233
- `__module__` (*rdflib.events.Dispatcher* attribute), 234
- `__module__` (*rdflib.events.Event* attribute), 235
- `__module__` (*rdflib.exceptions.Error* attribute), 235
- `__module__` (*rdflib.exceptions.ParserError* attribute), 235
- `__module__` (*rdflib.exceptions.UniquenessError* attribute), 236
- `__module__` (*rdflib.extras.describer.Describer* attribute), 49
- `__module__` (*rdflib.extras.infixowl.AnnotatableTerms* attribute), 58
- `__module__` (*rdflib.extras.infixowl.BooleanClass* attribute), 59
- `__module__` (*rdflib.extras.infixowl.Callable* attribute), 60
- `__module__` (*rdflib.extras.infixowl.Class* attribute), 61
- `__module__` (*rdflib.extras.infixowl.ClassNamespaceFactory* attribute), 62
- `__module__` (*rdflib.extras.infixowl.EnumeratedClass* attribute), 64
- `__module__` (*rdflib.extras.infixowl.Individual* attribute), 65
- `__module__` (*rdflib.extras.infixowl.Infix* attribute), 65
- `__module__` (*rdflib.extras.infixowl.MalformedClass* attribute), 65
- `__module__` (*rdflib.extras.infixowl.MalformedClassError* attribute), 66
- `__module__` (*rdflib.extras.infixowl.OWLRLDListProxy* attribute), 66
- `__module__` (*rdflib.extras.infixowl.Ontology* attribute), 67
- `__module__` (*rdflib.extras.infixowl.Property* attribute), 67
- `__module__` (*rdflib.extras.infixowl.Restriction* attribute), 68
- `__module__` (*rdflib.graph.BatchAddGraph* attribute), 241
- `__module__` (*rdflib.graph.ConjunctiveGraph* attribute), 242
- `__module__` (*rdflib.graph.Dataset* attribute), 248
- `__module__` (*rdflib.graph.Graph* attribute), 255
- `__module__` (*rdflib.graph.ModificationException* attribute), 268
- `__module__` (*rdflib.graph.QuotedGraph* attribute), 269
- `__module__` (*rdflib.graph.ReadOnlyGraphAggregate* attribute), 270
- `__module__` (*rdflib.graph.Seq* attribute), 275
- `__module__` (*rdflib.graph.UnSupportedAggregateOperation* attribute), 275
- `__module__` (*rdflib.namespace.ClosedNamespace* attribute), 72
- `__module__` (*rdflib.namespace.Namespace* attribute), 73
- `__module__` (*rdflib.namespace.NamespaceManager* attribute), 76
- `__module__` (*rdflib.parser.FileInputSource* attribute), 276
- `__module__` (*rdflib.parser.InputSource* attribute), 276
- `__module__` (*rdflib.parser.Parser* attribute), 276
- `__module__` (*rdflib.parser.PythonInputSource* attribute), 277
- `__module__` (*rdflib.parser.StringInputSource* attribute), 278
- `__module__` (*rdflib.parser.URLInputSource* attribute), 278
- `__module__` (*rdflib.paths.AlternativePath* attribute), 282
- `__module__` (*rdflib.paths.InvPath* attribute), 283
- `__module__` (*rdflib.paths.MulPath* attribute), 283
- `__module__` (*rdflib.paths.NegatedPath* attribute), 284
- `__module__` (*rdflib.paths.Path* attribute), 285
- `__module__` (*rdflib.paths.PathList* attribute), 286
- `__module__` (*rdflib.paths.SequencePath* attribute), 287
- `__module__` (*rdflib.plugin.PKGPlugin* attribute), 289
- `__module__` (*rdflib.plugin.Plugin* attribute), 290
- `__module__` (*rdflib.plugin.PluginException* attribute), 290
- `__module__` (*rdflib.plugins.parsers.hexst.HexuplesParser* attribute), 80
- `__module__` (*rdflib.plugins.parsers.jsonld.JsonLDParser* attribute), 81
- `__module__` (*rdflib.plugins.parsers.notation3.BadSyntax* attribute), 82
- `__module__` (*rdflib.plugins.parsers.notation3.Formula* attribute), 82
- `__module__` (*rdflib.plugins.parsers.notation3.N3Parser* attribute), 83
- `__module__` (*rdflib.plugins.parsers.notation3.RDFSink* attribute), 84
- `__module__` (*rdflib.plugins.parsers.notation3.SinkParser* attribute), 88
- `__module__` (*rdflib.plugins.parsers.notation3.TurtleParser* attribute), 95
- `__module__` (*rdflib.plugins.parsers.nquads.NQuadsParser* attribute), 95

<code>__module__</code> ( <code>rdflib.plugins.parsers.ntriples.DummySink</code> attribute), 98	<code>__module__</code> ( <code>rdflib.plugins.shared.jsonld.context.Term</code> attribute), 129
<code>__module__</code> ( <code>rdflib.plugins.parsers.ntriples.NTGraphSink</code> attribute), 99	<code>__module__</code> ( <code>rdflib.plugins.shared.jsonld.errors.JSONLDErrorException</code> attribute), 130
<code>__module__</code> ( <code>rdflib.plugins.parsers.ntriples.NTParser</code> attribute), 99	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.Accumulator</code> attribute), 140
<code>__module__</code> ( <code>rdflib.plugins.parsers.ntriples.W3CNTriplesParser</code> attribute), 100	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.Aggregator</code> attribute), 141
<code>__module__</code> ( <code>rdflib.plugins.parsers.rdfxml.BagID</code> attribute), 102	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.Average</code> attribute), 141
<code>__module__</code> ( <code>rdflib.plugins.parsers.rdfxml.ElementHandler</code> attribute), 102	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.Counter</code> attribute), 142
<code>__module__</code> ( <code>rdflib.plugins.parsers.rdfxml.RDFXMLHandler</code> attribute), 103	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.Extremum</code> attribute), 143
<code>__module__</code> ( <code>rdflib.plugins.parsers.rdfxml.RDFXMLParser</code> attribute), 108	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.GroupConcat</code> attribute), 143
<code>__module__</code> ( <code>rdflib.plugins.parsers.trig.TrigParser</code> attribute), 108	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.Maximum</code> attribute), 144
<code>__module__</code> ( <code>rdflib.plugins.parsers.trig.TrigSinkParser</code> attribute), 109	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.Minimum</code> attribute), 144
<code>__module__</code> ( <code>rdflib.plugins.parsers.trix.TriXHandler</code> attribute), 110	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.Sample</code> attribute), 144
<code>__module__</code> ( <code>rdflib.plugins.parsers.trix.TriXParser</code> attribute), 113	<code>__module__</code> ( <code>rdflib.plugins.sparql.aggregates.Sum</code> attribute), 145
<code>__module__</code> ( <code>rdflib.plugins.serializers.hext.HextuplesSerializer</code> attribute), 113	<code>__module__</code> ( <code>rdflib.plugins.sparql.algebra.ExpressionNotCoveredException</code> attribute), 146
<code>__module__</code> ( <code>rdflib.plugins.serializers.jsonld.JsonLDSerializer</code> attribute), 115	<code>__module__</code> ( <code>rdflib.plugins.sparql.algebra.StopTraversal</code> attribute), 147
<code>__module__</code> ( <code>rdflib.plugins.serializers.longturtle.LongTurtleSerializer</code> attribute), 115	<code>__module__</code> ( <code>rdflib.plugins.sparql.parserutils.Comp</code> attribute), 169
<code>__module__</code> ( <code>rdflib.plugins.serializers.n3.N3Serializer</code> attribute), 116	<code>__module__</code> ( <code>rdflib.plugins.sparql.parserutils.CompValue</code> attribute), 170
<code>__module__</code> ( <code>rdflib.plugins.serializers.nquads.NQuadsSerializer</code> attribute), 117	<code>__module__</code> ( <code>rdflib.plugins.sparql.parserutils.Expr</code> attribute), 171
<code>__module__</code> ( <code>rdflib.plugins.serializers.nt.NTSerializer</code> attribute), 118	<code>__module__</code> ( <code>rdflib.plugins.sparql.parserutils.Param</code> attribute), 171
<code>__module__</code> ( <code>rdflib.plugins.serializers.rdfxml.PrettyXMLSerializer</code> attribute), 118	<code>__module__</code> ( <code>rdflib.plugins.sparql.parserutils.ParamList</code> attribute), 172
<code>__module__</code> ( <code>rdflib.plugins.serializers.rdfxml.XMLSerializer</code> attribute), 119	<code>__module__</code> ( <code>rdflib.plugins.sparql.parserutils.ParamValue</code> attribute), 173
<code>__module__</code> ( <code>rdflib.plugins.serializers.trig.TrigSerializer</code> attribute), 120	<code>__module__</code> ( <code>rdflib.plugins.sparql.processor.SPARQLProcessor</code> attribute), 174
<code>__module__</code> ( <code>rdflib.plugins.serializers.trix.TriXSerializer</code> attribute), 121	<code>__module__</code> ( <code>rdflib.plugins.sparql.processor.SPARQLResult</code> attribute), 174
<code>__module__</code> ( <code>rdflib.plugins.serializers.turtle.RecursiveSerializer</code> attribute), 121	<code>__module__</code> ( <code>rdflib.plugins.sparql.processor.SPARQLUpdateProcessor</code> attribute), 175
<code>__module__</code> ( <code>rdflib.plugins.serializers.turtle.TurtleSerializer</code> attribute), 122	<code>__module__</code> ( <code>rdflib.plugins.sparql.results.csvresults.CSVResultParser</code> attribute), 132
<code>__module__</code> ( <code>rdflib.plugins.serializers.xmlwriter.XMLWriter</code> attribute), 123	<code>__module__</code> ( <code>rdflib.plugins.sparql.results.csvresults.CSVResultSerializer</code> attribute), 132
<code>__module__</code> ( <code>rdflib.plugins.shared.jsonld.context.Context</code> attribute), 125	<code>__module__</code> ( <code>rdflib.plugins.sparql.results.graph.GraphResultParser</code> attribute), 133
<code>__module__</code> ( <code>rdflib.plugins.shared.jsonld.context.Defined</code> attribute), 129	<code>__module__</code> ( <code>rdflib.plugins.sparql.results.jsonresults.JSONResult</code> attribute), 133



- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.jsonresults.JSONResults attribute\), 133](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.jsonresults.JSONResults attribute\), 134](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.rdfresults.RDFResults attribute\), 135](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.rdfresults.RDFResults attribute\), 135](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.tsvresults.TSVResults attribute\), 135](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.txtresults.TXTResults attribute\), 136](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter attribute\), 137](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.xmlresults.XMLResults attribute\), 138](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.xmlresults.XMLResults attribute\), 138](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.results.xmlresults.XMLResults attribute\), 138](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.AlreadyBound attribute\), 176](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.Bindings attribute\), 177](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.FrozenBindings attribute\), 178](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.FrozenDict attribute\), 180](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.NotBoundError attribute\), 181](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.Prologue attribute\), 181](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.Query attribute\), 182](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.QueryContext attribute\), 183](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.SPARQLError attribute\), 184](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.SPARQLTypeError attribute\), 184](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.sparql.sparql.Update attribute\), 185](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.auditable.AuditableStore attribute\), 189](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.berkeleydb.BerkeleyDB attribute\), 192](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.concurrent.ConcurrentStore attribute\), 195](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.concurrent.ResponsibleGenerator attribute\), 196](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.memory.Memory attribute\), 196](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.memory.SimpleMemory attribute\), 199](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.regexmatching.REGEXMatching attribute\), 202](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.regexmatching.REGEXTerm attribute\), 203](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.sparqlconnector.SPARQLConnector attribute\), 204](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.sparqlconnector.SPARQLConnectorException attribute\), 204](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.sparqlstore.SPARQLStore attribute\), 206](#)
- [\\_\\_module\\_\\_ \(rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore attribute\), 213](#)
- [\\_\\_module\\_\\_ \(rdflib.query.EncodeOnlyUnicode attribute\), 291](#)
- [\\_\\_module\\_\\_ \(rdflib.query.Processor attribute\), 292](#)
- [\\_\\_module\\_\\_ \(rdflib.query.Result attribute\), 293](#)
- [\\_\\_module\\_\\_ \(rdflib.query.ResultException attribute\), 294](#)
- [\\_\\_module\\_\\_ \(rdflib.query.ResultParser attribute\), 295](#)
- [\\_\\_module\\_\\_ \(rdflib.query.ResultRow attribute\), 296](#)
- [\\_\\_module\\_\\_ \(rdflib.query.ResultSerializer attribute\), 297](#)
- [\\_\\_module\\_\\_ \(rdflib.query.UpdateProcessor attribute\), 298](#)
- [\\_\\_module\\_\\_ \(rdflib.resource.Resource attribute\), 304](#)
- [\\_\\_module\\_\\_ \(rdflib.serializer.Serializer attribute\), 306](#)
- [\\_\\_module\\_\\_ \(rdflib.store.NodePickler attribute\), 306](#)
- [\\_\\_module\\_\\_ \(rdflib.store.Store attribute\), 308](#)
- [\\_\\_module\\_\\_ \(rdflib.store.StoreCreatedEvent attribute\), 312](#)
- [\\_\\_module\\_\\_ \(rdflib.store.TripleAddedEvent attribute\), 313](#)
- [\\_\\_module\\_\\_ \(rdflib.store.TripleRemovedEvent attribute\), 313](#)
- [\\_\\_module\\_\\_ \(rdflib.term.BNode attribute\), 313](#)
- [\\_\\_module\\_\\_ \(rdflib.term.IdentifiedNode attribute\), 315](#)
- [\\_\\_module\\_\\_ \(rdflib.term.Identifier attribute\), 316](#)
- [\\_\\_module\\_\\_ \(rdflib.term.Literal attribute\), 322](#)
- [\\_\\_module\\_\\_ \(rdflib.term.Node attribute\), 326](#)
- [\\_\\_module\\_\\_ \(rdflib.term.URIRef attribute\), 327](#)
- [\\_\\_module\\_\\_ \(rdflib.term.Variable attribute\), 329](#)
- [\\_\\_module\\_\\_ \(rdflib.tools.csv2rdf.CSV2RDF attribute\), 219](#)
- [\\_\\_module\\_\\_ \(rdflib.tools.graphisomorphism.IsomorphicTestableGraph attribute\), 220](#)
- [\\_\\_mul\\_\\_ \(\) \(rdflib.Graph method\), 428](#)
- [\\_\\_mul\\_\\_ \(\) \(rdflib.URIRef method\), 604](#)
- [\\_\\_mul\\_\\_ \(\) \(rdflib.graph.Graph method\), 255](#)
- [\\_\\_mul\\_\\_ \(\) \(rdflib.paths.Path method\), 285](#)
- [\\_\\_mul\\_\\_ \(\) \(rdflib.term.URIRef method\), 327](#)
- [\\_\\_ne\\_\\_ \(\) \(rdflib.compare.IsomorphicGraph method\), 228](#)
- [\\_\\_ne\\_\\_ \(\) \(rdflib.resource.Resource method\), 304](#)
- [\\_\\_ne\\_\\_ \(\) \(rdflib.term.Identifier method\), 316](#)

[\\_\\_ne\\_\\_\(\) \(rdflib.tools.graphisomorphism.IsomorphicTestable method\), 220](#)  
[\\_\\_neg\\_\\_\(\) \(rdflib.Literal method\), 447](#)  
[\\_\\_neg\\_\\_\(\) \(rdflib.URIRef method\), 605](#)  
[\\_\\_neg\\_\\_\(\) \(rdflib.paths.Path method\), 286](#)  
[\\_\\_neg\\_\\_\(\) \(rdflib.term.Literal method\), 322](#)  
[\\_\\_neg\\_\\_\(\) \(rdflib.term.URIRef method\), 327](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.BNode static method\), 335](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.Literal static method\), 447](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.Namespace static method\), 452](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.URIRef static method\), 605](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.Variable static method\), 608](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.namespace.ClosedNamespace static method\), 72](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.namespace.Namespace static method\), 73](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.plugins.shared.jsonld.context.Term static method\), 129](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.query.ResultRow static method\), 296](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.term.BNode static method\), 313](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.term.Identifier static method\), 316](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.term.Literal static method\), 323](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.term.URIRef static method\), 328](#)  
[\\_\\_new\\_\\_\(\) \(rdflib.term.Variable static method\), 329](#)  
[\\_\\_next\\_\\_\(\) \(rdflib.plugins.stores.concurrent.ResponsibleGenerator method\), 196](#)  
[\\_\\_or\\_\\_\(\) \(rdflib.Graph method\), 428](#)  
[\\_\\_or\\_\\_\(\) \(rdflib.URIRef method\), 605](#)  
[\\_\\_or\\_\\_\(\) \(rdflib.extras.infixowl.BooleanClass method\), 59](#)  
[\\_\\_or\\_\\_\(\) \(rdflib.extras.infixowl.Class method\), 61](#)  
[\\_\\_or\\_\\_\(\) \(rdflib.graph.Graph method\), 255](#)  
[\\_\\_or\\_\\_\(\) \(rdflib.paths.Path method\), 286](#)  
[\\_\\_or\\_\\_\(\) \(rdflib.term.URIRef method\), 328](#)  
[\\_\\_orig\\_bases\\_\\_ \(rdflib.plugin.PKGPlugin attribute\), 289](#)  
[\\_\\_orig\\_bases\\_\\_ \(rdflib.plugin.Plugin attribute\), 290](#)  
[\\_\\_orig\\_bases\\_\\_ \(rdflib.query.ResultRow attribute\), 296](#)  
[\\_\\_parameters\\_\\_ \(rdflib.plugin.PKGPlugin attribute\), 289](#)  
[\\_\\_parameters\\_\\_ \(rdflib.plugin.Plugin attribute\), 290](#)  
[\\_\\_parameters\\_\\_ \(rdflib.query.ResultRow attribute\), 296](#)  
[\\_\\_pos\\_\\_\(\) \(rdflib.Literal method\), 447](#)  
[\\_\\_pos\\_\\_\(\) \(rdflib.term.Literal method\), 323](#)  
[\\_\\_radd\\_\\_\(\) \(rdflib.URIRef method\), 605](#)  
[\\_\\_radd\\_\\_\(\) \(rdflib.term.URIRef method\), 328](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.BNode method\), 335](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.ConjunctiveGraph method\), 404](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.Dataset method\), 418](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.Graph method\), 428](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.Literal method\), 448](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.URIRef method\), 605](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.Variable method\), 608](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.graph.ConjunctiveGraph method\), 242](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.graph.Dataset method\), 248](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.graph.Graph method\), 255](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.graph.QuotedGraph method\), 269](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.graph.ReadOnlyGraphAggregate method\), 270](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.plugins.stores.regexmatching.REGEXTerm method\), 203](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.term.BNode method\), 314](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.term.Literal method\), 323](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.term.URIRef method\), 328](#)  
[\\_\\_reduce\\_\\_\(\) \(rdflib.term.Variable method\), 329](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.BNode method\), 335](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.Graph method\), 428](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.Literal method\), 448](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.Namespace method\), 452](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.URIRef method\), 605](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.Variable method\), 608](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.events.Event method\), 235](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.extras.infixowl.BooleanClass method\), 59](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.extras.infixowl.Class method\), 61](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.extras.infixowl.EnumeratedClass method\), 68](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.extras.infixowl.MalformedClassError method\), 66](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.extras.infixowl.Property method\), 67](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.extras.infixowl.Restriction method\), 68](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.graph.Graph method\), 255](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.graph.ReadOnlyGraphAggregate method\), 271](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.namespace.ClosedNamespace method\), 72](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.namespace.Namespace method\), 74](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.parser.FileInputSource method\), 276](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.parser.URLInputSource method\), 278](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.paths.AlternativePath method\), 282](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.paths.InvPath method\), 283](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.paths.MulPath method\), 284](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.paths.NegatedPath method\), 284](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.paths.SequencePath method\), 287](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.plugins.shared.jsonld.context.Term method\), 129](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.plugins.sparql.parserutils.CompValue method\), 170](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.plugins.sparql.sparql.Bindings method\), 177](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.plugins.sparql.sparql.FrozenDict method\), 180](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.resource.Resource method\), 304](#)  
[\\_\\_repr\\_\\_\(\) \(rdflib.term.BNode method\), 314](#)

`__repr__()` (*rdflib.term.Literal* method), 323  
`__repr__()` (*rdflib.term.URIRef* method), 328  
`__repr__()` (*rdflib.term.Variable* method), 329  
`__rshift__()` (*rdflib.extras.infixowl.Infix* method), 65  
`__rmatmul__()` (*rdflib.extras.infixowl.Infix* method), 65  
`__rshift__()` (*rdflib.extras.infixowl.Infix* method), 65  
`__setitem__()` (*rdflib.collection.Collection* method), 225  
`__setitem__()` (*rdflib.container.Container* method), 232  
`__setitem__()` (*rdflib.extras.infixowl.OWL* *RDFListProxy* method), 66  
`__setitem__()` (*rdflib.plugins.sparql.sparql.Bindings* method), 177  
`__setitem__()` (*rdflib.plugins.sparql.sparql.QueryContext* method), 183  
`__setitem__()` (*rdflib.resource.Resource* method), 304  
`__setstate__()` (*rdflib.Dataset* method), 418  
`__setstate__()` (*rdflib.Literal* method), 448  
`__setstate__()` (*rdflib.graph.Dataset* method), 248  
`__setstate__()` (*rdflib.store.NodePickler* method), 306  
`__setstate__()` (*rdflib.term.Literal* method), 323  
`__slotnames__` (*rdflib.plugins.sparql.parserutils.Comp* attribute), 169  
`__slotnames__` (*rdflib.plugins.sparql.parserutils.Param* attribute), 172  
`__slotnames__` (*rdflib.plugins.sparql.parserutils.ParamList* attribute), 172  
`__slots__` (*rdflib.BNode* attribute), 336  
`__slots__` (*rdflib.Literal* attribute), 448  
`__slots__` (*rdflib.URIRef* attribute), 605  
`__slots__` (*rdflib.Variable* attribute), 608  
`__slots__` (*rdflib.parser.Parser* attribute), 276  
`__slots__` (*rdflib.plugins.parsers.ntriples.NTGraphSink* attribute), 99  
`__slots__` (*rdflib.plugins.parsers.ntriples.NTParser* attribute), 99  
`__slots__` (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* attribute), 100  
`__slots__` (*rdflib.plugins.parsers.rdfxml.BagID* attribute), 102  
`__slots__` (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 102  
`__slots__` (*rdflib.plugins.shared.jsonld.context.Term* attribute), 129  
`__slots__` (*rdflib.plugins.stores.concurrent.ResponsibleGenerator* attribute), 196  
`__slots__` (*rdflib.term.BNode* attribute), 314  
`__slots__` (*rdflib.term.Identifier* attribute), 316  
`__slots__` (*rdflib.term.Literal* attribute), 323  
`__slots__` (*rdflib.term.Node* attribute), 326  
`__slots__` (*rdflib.term.URIRef* attribute), 328  
`__slots__` (*rdflib.term.Variable* attribute), 329  
`__str__()` (*rdflib.ConjunctiveGraph* method), 404  
`__str__()` (*rdflib.Dataset* method), 418  
`__str__()` (*rdflib.Graph* method), 428  
`__str__()` (*rdflib.container.NoElementException* method), 233  
`__str__()` (*rdflib.exceptions.ParserError* method), 235  
`__str__()` (*rdflib.graph.ConjunctiveGraph* method), 242  
`__str__()` (*rdflib.graph.Dataset* method), 248  
`__str__()` (*rdflib.graph.Graph* method), 255  
`__str__()` (*rdflib.graph.ModificationException* method), 268  
`__str__()` (*rdflib.graph.QuotedGraph* method), 269  
`__str__()` (*rdflib.graph.UnsupportedAggregateOperation* method), 275  
`__str__()` (*rdflib.plugins.parsers.notation3.BadSyntax* method), 82  
`__str__()` (*rdflib.plugins.parsers.notation3.Formula* method), 83  
`__str__()` (*rdflib.plugins.sparql.parserutils.CompValue* method), 170  
`__str__()` (*rdflib.plugins.sparql.parserutils.ParamValue* method), 173  
`__str__()` (*rdflib.plugins.sparql.sparql.Bindings* method), 177  
`__str__()` (*rdflib.plugins.sparql.sparql.FrozenDict* method), 180  
`__str__()` (*rdflib.resource.Resource* method), 305  
`__sub__()` (*rdflib.Graph* method), 428  
`__sub__()` (*rdflib.Literal* method), 448  
`__sub__()` (*rdflib.graph.Graph* method), 255  
`__sub__()` (*rdflib.term.Literal* method), 324  
`__truediv__()` (*rdflib.URIRef* method), 605  
`__truediv__()` (*rdflib.paths.Path* method), 286  
`__truediv__()` (*rdflib.term.URIRef* method), 328  
`__unicode__()` (*rdflib.resource.Resource* method), 305  
`__weakref__` (*rdflib.Graph* attribute), 428  
`__weakref__` (*rdflib IdentifiedNode* attribute), 441  
`__weakref__` (*rdflib.Namespace* attribute), 453  
`__weakref__` (*rdflib.collection.Collection* attribute), 225  
`__weakref__` (*rdflib.container.Container* attribute), 232  
`__weakref__` (*rdflib.container.NoElementException* attribute), 233  
`__weakref__` (*rdflib.events.Dispatcher* attribute), 234  
`__weakref__` (*rdflib.events.Event* attribute), 235  
`__weakref__` (*rdflib.exceptions.Error* attribute), 235  
`__weakref__` (*rdflib.extras.describer.Describer* attribute), 49  
`__weakref__` (*rdflib.extras.infixowl.Callable* attribute), 60  
`__weakref__` (*rdflib.extras.infixowl.Individual* attribute), 65  
`__weakref__` (*rdflib.extras.infixowl.Infix* attribute), 65  
`__weakref__` (*rdflib.extras.infixowl.MalformedClass* attribute), 66

- `__weakref__` (`rdflib.extras.infixowl.OWL RDFListProxy attribute`), 66
  - `__weakref__` (`rdflib.graph.BatchAddGraph attribute`), 241
  - `__weakref__` (`rdflib.graph.Graph attribute`), 255
  - `__weakref__` (`rdflib.graph.ModificationException attribute`), 268
  - `__weakref__` (`rdflib.graph.Seq attribute`), 275
  - `__weakref__` (`rdflib.graph.UnSupportedAggregateOperation attribute`), 275
  - `__weakref__` (`rdflib.namespace.Namespace attribute`), 74
  - `__weakref__` (`rdflib.namespace.NamespaceManager attribute`), 76
  - `__weakref__` (`rdflib.paths.Path attribute`), 286
  - `__weakref__` (`rdflib.paths.PathList attribute`), 286
  - `__weakref__` (`rdflib.plugin.Plugin attribute`), 290
  - `__weakref__` (`rdflib.plugins.parsers.hext.HexuplesParser attribute`), 80
  - `__weakref__` (`rdflib.plugins.parsers.jsonld.JsonLDParse attribute`), 81
  - `__weakref__` (`rdflib.plugins.parsers.notation3.BadSyntax attribute`), 82
  - `__weakref__` (`rdflib.plugins.parsers.notation3.Formula attribute`), 83
  - `__weakref__` (`rdflib.plugins.parsers.notation3.RDFSink attribute`), 84
  - `__weakref__` (`rdflib.plugins.parsers.notation3.SinkParser attribute`), 88
  - `__weakref__` (`rdflib.plugins.parsers.notation3.TurtleParser attribute`), 95
  - `__weakref__` (`rdflib.plugins.parsers.nquads.NQuadsParser attribute`), 98
  - `__weakref__` (`rdflib.plugins.parsers.ntriples.DummySink attribute`), 99
  - `__weakref__` (`rdflib.plugins.parsers.rdfxml.RDFXMLParser attribute`), 108
  - `__weakref__` (`rdflib.plugins.parsers.trig.TrigParser attribute`), 108
  - `__weakref__` (`rdflib.plugins.parsers.trix.TriXParser attribute`), 113
  - `__weakref__` (`rdflib.plugins.serializers.xmlwriter.XMLWriter attribute`), 123
  - `__weakref__` (`rdflib.plugins.shared.jsonld.context.Context attribute`), 125
  - `__weakref__` (`rdflib.plugins.shared.jsonld.errors.JSONLDException attribute`), 130
  - `__weakref__` (`rdflib.plugins.sparql.aggregates.Accumulator attribute`), 140
  - `__weakref__` (`rdflib.plugins.sparql.aggregates.Aggregator attribute`), 141
  - `__weakref__` (`rdflib.plugins.sparql.algebra.ExpressionNotCoveredException attribute`), 146
  - `__weakref__` (`rdflib.plugins.sparql.algebra.StopTraversal attribute`), 147
  - `__weakref__` (`rdflib.plugins.sparql.parserutils.ParamValue attribute`), 173
  - `__weakref__` (`rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter attribute`), 137
  - `__weakref__` (`rdflib.plugins.sparql.sparql.Bindings attribute`), 177
  - `__weakref__` (`rdflib.plugins.sparql.sparql.FrozenDict attribute`), 180
  - `__weakref__` (`rdflib.plugins.sparql.sparql.Prologue attribute`), 181
  - `__weakref__` (`rdflib.plugins.sparql.sparql.Query attribute`), 182
  - `__weakref__` (`rdflib.plugins.sparql.sparql.QueryContext attribute`), 183
  - `__weakref__` (`rdflib.plugins.sparql.sparql.SPARQLError attribute`), 184
  - `__weakref__` (`rdflib.plugins.sparql.sparql.Update attribute`), 185
  - `__weakref__` (`rdflib.plugins.stores.concurrent.ConcurrentStore attribute`), 195
  - `__weakref__` (`rdflib.plugins.stores.regexmatching.REGEXTerm attribute`), 203
  - `__weakref__` (`rdflib.plugins.stores.sparqlconnector.SPARQLConnector attribute`), 204
  - `__weakref__` (`rdflib.plugins.stores.sparqlconnector.SPARQLConnectorException attribute`), 204
  - `__weakref__` (`rdflib.query.EncodeOnlyUnicode attribute`), 291
  - `__weakref__` (`rdflib.query.Processor attribute`), 292
  - `__weakref__` (`rdflib.query.Result attribute`), 293
  - `__weakref__` (`rdflib.query.ResultException attribute`), 294
  - `__weakref__` (`rdflib.query.ResultParser attribute`), 295
  - `__weakref__` (`rdflib.query.ResultSerializer attribute`), 297
  - `__weakref__` (`rdflib.query.UpdateProcessor attribute`), 298
  - `__weakref__` (`rdflib.resource.Resource attribute`), 305
  - `__weakref__` (`rdflib.serializer.Serializer attribute`), 306
  - `__weakref__` (`rdflib.store.NodePickler attribute`), 307
  - `__weakref__` (`rdflib.store.Store attribute`), 308
  - `__weakref__` (`rdflib.term.IdentifiedNode attribute`), 315
  - `__weakref__` (`rdflib.tools.csv2rdf.CSV2RDF attribute`), 219
  - `__xor__` (`rdflib.Graph method`), 428
  - `__xor__` (`rdflib.graph.Graph method`), 255
  - `castLexicalToPython()` (in module `rdflib.term`), 34
  - `castPythonToLiteral()` (in module `rdflib.term`), 33
- ## A
- `Abdomen` (`rdflib.SDO attribute`), 479
  - `Ablutions_Room` (`rdflib.BRICK attribute`), 336



- about (*rdflib.plugins.parsers.RDFVOC.RDFVOC* attribute), 79
- about (*rdflib.SDO* attribute), 532
- about() (*rdflib.extras.describer.Describer* method), 49
- AboutPage (*rdflib.SDO* attribute), 479
- aboutUrl (*rdflib.CSVW* attribute), 401
- abridged (*rdflib.SDO* attribute), 532
- absolutePosition (*rdflib.ODRL2* attribute), 454
- absoluteSize (*rdflib.ODRL2* attribute), 454
- absoluteSpatialPosition (*rdflib.ODRL2* attribute), 454
- absoluteTemporalPosition (*rdflib.ODRL2* attribute), 454
- absolutize() (*rdflib.Graph* method), 429
- absolutize() (*rdflib.graph.Graph* method), 256
- absolutize() (*rdflib.graph.ReadOnlyGraphAggregate* method), 271
- absolutize() (*rdflib.namespace.NamespaceManager* method), 76
- absolutize() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 103
- absolutize() (*rdflib.plugins.sparql.sparql.Prologue* method), 181
- Absorption\_Chiller (*rdflib.BRICK* attribute), 336
- abstract (*rdflib.DCTERMS* attribute), 412
- abstract (*rdflib.SDO* attribute), 532
- AbstractResult (*rdflib.SH* attribute), 588
- Acceleration\_Time\_Setpoint (*rdflib.BRICK* attribute), 336
- accelerationTime (*rdflib.SDO* attribute), 532
- Accept (*rdflib.PROV* attribute), 468
- AcceptAction (*rdflib.SDO* attribute), 479
- acceptedAnswer (*rdflib.SDO* attribute), 532
- acceptedOffer (*rdflib.SDO* attribute), 532
- acceptedPaymentMethod (*rdflib.SDO* attribute), 532
- acceptsReservations (*rdflib.SDO* attribute), 532
- acceptTracking (*rdflib.ODRL2* attribute), 454
- Access\_Control\_Equipment (*rdflib.BRICK* attribute), 336
- Access\_Reader (*rdflib.BRICK* attribute), 336
- accessCode (*rdflib.SDO* attribute), 532
- accessibilityAPI (*rdflib.SDO* attribute), 532
- accessibilityControl (*rdflib.SDO* attribute), 532
- accessibilityFeature (*rdflib.SDO* attribute), 532
- accessibilityHazard (*rdflib.SDO* attribute), 532
- accessibilitySummary (*rdflib.SDO* attribute), 532
- accessMode (*rdflib.SDO* attribute), 532
- accessModeSufficient (*rdflib.SDO* attribute), 532
- accessRights (*rdflib.DCTERMS* attribute), 412
- accessService (*rdflib.DCAT* attribute), 408
- accessURL (*rdflib.DCAT* attribute), 408
- Accommodation (*rdflib.SDO* attribute), 479
- accommodationCategory (*rdflib.SDO* attribute), 532
- accommodationFloorPlan (*rdflib.SDO* attribute), 532
- account (*rdflib.FOAF* attribute), 421
- accountablePerson (*rdflib.SDO* attribute), 533
- accountId (*rdflib.SDO* attribute), 533
- AccountingService (*rdflib.SDO* attribute), 479
- accountMinimumInflow (*rdflib.SDO* attribute), 533
- accountName (*rdflib.FOAF* attribute), 421
- accountOverdraftLimit (*rdflib.SDO* attribute), 533
- accountServiceHomepage (*rdflib.FOAF* attribute), 421
- accrualMethod (*rdflib.DCTERMS* attribute), 412
- accrualPeriodicity (*rdflib.DCTERMS* attribute), 412
- accrualPolicy (*rdflib.DCTERMS* attribute), 412
- Accumulator (class in *rdflib.plugins.sparql.aggregates*), 139
- accumulator\_classes (*rdflib.plugins.sparql.aggregates.Aggregator* attribute), 141
- AchieveAction (*rdflib.SDO* attribute), 479
- acquiredFrom (*rdflib.SDO* attribute), 533
- acquireLicensePage (*rdflib.SDO* attribute), 533
- alphaNumericCode (*rdflib.SDO* attribute), 533
- actedOnBehalfOf (*rdflib.PROV* attribute), 470
- Action (*rdflib.ODRL2* attribute), 453
- action (*rdflib.ODRL2* attribute), 454
- Action (*rdflib.SDO* attribute), 479
- actionableFeedbackPolicy (*rdflib.SDO* attribute), 533
- actionAccessibilityRequirement (*rdflib.SDO* attribute), 533
- ActionAccessSpecification (*rdflib.SDO* attribute), 480
- actionApplication (*rdflib.SDO* attribute), 533
- actionOption (*rdflib.SDO* attribute), 533
- actionPlatform (*rdflib.SDO* attribute), 533
- actionStatus (*rdflib.SDO* attribute), 533
- ActionStatusType (*rdflib.SDO* attribute), 480
- ActivateAction (*rdflib.SDO* attribute), 480
- ActivationFee (*rdflib.SDO* attribute), 480
- Active\_Chilled\_Beam (*rdflib.BRICK* attribute), 336
- Active\_Power\_Sensor (*rdflib.BRICK* attribute), 336
- ActiveActionStatus (*rdflib.SDO* attribute), 480
- activeIngredient (*rdflib.SDO* attribute), 533
- ActiveNotRecruiting (*rdflib.SDO* attribute), 480
- Activity (*rdflib.PROV* attribute), 468
- activity (*rdflib.PROV* attribute), 470
- activityDuration (*rdflib.SDO* attribute), 533
- activityFrequency (*rdflib.SDO* attribute), 533
- ActivityInfluence (*rdflib.PROV* attribute), 468
- activityOfInfluence (*rdflib.PROV* attribute), 470
- actor (*rdflib.SDO* attribute), 533
- actors (*rdflib.SDO* attribute), 533
- actsOnProperty (*rdflib.SOSA* attribute), 598
- ActuatableProperty (*rdflib.SOSA* attribute), 597
- Actuation (*rdflib.SOSA* attribute), 597
- Actuator (*rdflib.SOSA* attribute), 597

- `add()` (*rdflib.ConjunctiveGraph* method), 404
- `add()` (*rdflib.Graph* method), 429
- `add()` (*rdflib.graph.BatchAddGraph* method), 241
- `add()` (*rdflib.graph.ConjunctiveGraph* method), 242
- `add()` (*rdflib.graph.Graph* method), 256
- `add()` (*rdflib.graph.QuotedGraph* method), 269
- `add()` (*rdflib.graph.ReadOnlyGraphAggregate* method), 271
- `add()` (*rdflib.plugins.stores.auditable.AuditableStore* method), 189
- `add()` (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* method), 192
- `add()` (*rdflib.plugins.stores.concurrent.ConcurrentStore* method), 195
- `add()` (*rdflib.plugins.stores.memory.Memory* method), 196
- `add()` (*rdflib.plugins.stores.memory.SimpleMemory* method), 199
- `add()` (*rdflib.plugins.stores.regexmatching.REGEXMatching* method), 202
- `add()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 206
- `add()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 213
- `add()` (*rdflib.resource.Resource* method), 305
- `add()` (*rdflib.store.Store* method), 308
- `add_at_position()` (*rdflib.container.Seq* method), 233
- `add_graph()` (*rdflib.Dataset* method), 418
- `add_graph()` (*rdflib.graph.Dataset* method), 248
- `add_graph()` (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* method), 193
- `add_graph()` (*rdflib.plugins.stores.memory.Memory* method), 197
- `add_graph()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 207
- `add_graph()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 213
- `add_graph()` (*rdflib.store.Store* method), 309
- `add_reified()` (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 103
- `add_term()` (*rdflib.plugins.shared.jsonld.context.Context* method), 125
- `AddAction` (*rdflib.SDO* attribute), 480
- `additionalName` (*rdflib.SDO* attribute), 533
- `additionalNumberOfGuests` (*rdflib.SDO* attribute), 533
- `additionalProperty` (*rdflib.SDO* attribute), 533
- `additionalType` (*rdflib.SDO* attribute), 534
- `additionalVariable` (*rdflib.SDO* attribute), 534
- `AdditiveExpression()` (in module *rdflib.plugins.sparql.operators*), 156
- `addN()` (*rdflib.ConjunctiveGraph* method), 404
- `addN()` (*rdflib.Graph* method), 429
- `addN()` (*rdflib.graph.BatchAddGraph* method), 241
- `addN()` (*rdflib.graph.ConjunctiveGraph* method), 242
- `addN()` (*rdflib.graph.Graph* method), 256
- `addN()` (*rdflib.graph.QuotedGraph* method), 269
- `addN()` (*rdflib.graph.ReadOnlyGraphAggregate* method), 271
- `addN()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 206
- `addN()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 213
- `addN()` (*rdflib.store.Store* method), 308
- `addNamespace()` (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 115
- `addNamespace()` (*rdflib.plugins.serializers.turtle.RecursiveSerializer* method), 121
- `addNamespace()` (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 122
- `addOn` (*rdflib.SDO* attribute), 533
- `address` (*rdflib.SDO* attribute), 534
- `addressCountry` (*rdflib.SDO* attribute), 534
- `addressLocality` (*rdflib.SDO* attribute), 534
- `addressRegion` (*rdflib.SDO* attribute), 534
- `adHocShare` (*rdflib.ODRL2* attribute), 454
- `Adjust_Sensor` (*rdflib.BRICK* attribute), 336
- `administrationRoute` (*rdflib.SDO* attribute), 534
- `AdministrativeArea` (*rdflib.SDO* attribute), 480
- `AdultEntertainment` (*rdflib.SDO* attribute), 480
- `advanceBookingRequirement` (*rdflib.SDO* attribute), 534
- `adverseOutcome` (*rdflib.SDO* attribute), 534
- `AdvertiserContentArticle` (*rdflib.SDO* attribute), 480
- `AED` (*rdflib.BRICK* attribute), 336
- `AerobicActivity` (*rdflib.SDO* attribute), 480
- `affectedBy` (*rdflib.SDO* attribute), 534
- `affiliation` (*rdflib.SDO* attribute), 534
- `afterMedia` (*rdflib.SDO* attribute), 534
- `age` (*rdflib.FOAF* attribute), 421
- `agent` (*rdflib.DCTERMS* attribute), 410
- `Agent` (*rdflib.FOAF* attribute), 420
- `Agent` (*rdflib.PROV* attribute), 468
- `agent` (*rdflib.PROV* attribute), 470
- `agent` (*rdflib.SDO* attribute), 534
- `AgentClass` (*rdflib.DCTERMS* attribute), 410
- `AgentInfluence` (*rdflib.PROV* attribute), 468
- `agentOfInfluence` (*rdflib.PROV* attribute), 470
- `aggregate` (*rdflib.BRICK* attribute), 397
- `aggregate` (*rdflib.ODRL2* attribute), 455
- `AggregateOffer` (*rdflib.SDO* attribute), 480
- `AggregateRating` (*rdflib.SDO* attribute), 480
- `aggregateRating` (*rdflib.SDO* attribute), 534
- `Aggregator` (class in *rdflib.plugins.sparql.aggregates*), 140
- `AgreeAction` (*rdflib.SDO* attribute), 480

- Agreement (*rdflib.ODRL2 attribute*), 453
- AHU (*rdflib.BRICK attribute*), 336
- aimChatID (*rdflib.FOAF attribute*), 421
- Air (*rdflib.BRICK attribute*), 337
- Air\_Alarm (*rdflib.BRICK attribute*), 337
- Air\_Differential\_Pressure\_Sensor (*rdflib.BRICK attribute*), 337
- Air\_Differential\_Pressure\_Setpoint (*rdflib.BRICK attribute*), 337
- Air\_Diffuser (*rdflib.BRICK attribute*), 337
- Air\_Enthalpy\_Sensor (*rdflib.BRICK attribute*), 337
- Air\_Flow\_Deadband\_Setpoint (*rdflib.BRICK attribute*), 337
- Air\_Flow\_Demand\_Setpoint (*rdflib.BRICK attribute*), 337
- Air\_Flow\_Loss\_Alarm (*rdflib.BRICK attribute*), 337
- Air\_Flow\_Sensor (*rdflib.BRICK attribute*), 337
- Air\_Flow\_Setpoint (*rdflib.BRICK attribute*), 337
- Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK attribute*), 337
- Air\_Grains\_Sensor (*rdflib.BRICK attribute*), 337
- Air\_Handler\_Unit (*rdflib.BRICK attribute*), 337
- Air\_Handling\_Unit (*rdflib.BRICK attribute*), 337
- Air\_Humidity\_Setpoint (*rdflib.BRICK attribute*), 337
- Air\_Loop (*rdflib.BRICK attribute*), 337
- Air\_Plenum (*rdflib.BRICK attribute*), 337
- Air\_Quality\_Sensor (*rdflib.BRICK attribute*), 337
- Air\_Static\_Pressure\_Step\_Parameter (*rdflib.BRICK attribute*), 338
- Air\_System (*rdflib.BRICK attribute*), 338
- Air\_Temperature\_Alarm (*rdflib.BRICK attribute*), 338
- Air\_Temperature\_Integral\_Time\_Parameter (*rdflib.BRICK attribute*), 338
- Air\_Temperature\_Sensor (*rdflib.BRICK attribute*), 338
- Air\_Temperature\_Setpoint (*rdflib.BRICK attribute*), 338
- Air\_Temperature\_Setpoint\_Limit (*rdflib.BRICK attribute*), 338
- Air\_Temperature\_Step\_Parameter (*rdflib.BRICK attribute*), 338
- Air\_Wet\_Bulb\_Temperature\_Sensor (*rdflib.BRICK attribute*), 338
- aircraft (*rdflib.SDO attribute*), 534
- Airline (*rdflib.SDO attribute*), 480
- Airport (*rdflib.SDO attribute*), 480
- Alarm (*rdflib.BRICK attribute*), 338
- Alarm\_Delay\_Parameter (*rdflib.BRICK attribute*), 338
- album (*rdflib.SDO attribute*), 534
- albumProductionType (*rdflib.SDO attribute*), 534
- AlbumRelease (*rdflib.SDO attribute*), 480
- albumRelease (*rdflib.SDO attribute*), 534
- albumReleaseType (*rdflib.SDO attribute*), 534
- albums (*rdflib.SDO attribute*), 534
- alcoholWarning (*rdflib.SDO attribute*), 534
- algorithm (*rdflib.SDO attribute*), 534
- AlignmentObject (*rdflib.SDO attribute*), 480
- alignmentType (*rdflib.SDO attribute*), 534
- All (*rdflib.ODRL2 attribute*), 453
- All2ndConnections (*rdflib.ODRL2 attribute*), 453
- all\_nodes() (*rdflib.Graph method*), 429
- all\_nodes() (*rdflib.graph.Graph method*), 256
- AllClasses() (*in module rdflib.extras.infixowl*), 57
- AllConnections (*rdflib.ODRL2 attribute*), 453
- AllDifferent (*rdflib.OWL attribute*), 463
- AllDifferent() (*in module rdflib.extras.infixowl*), 57
- AllDisjointClasses (*rdflib.OWL attribute*), 463
- AllDisjointProperties (*rdflib.OWL attribute*), 463
- AllergiesHealthAspect (*rdflib.SDO attribute*), 480
- AllGroups (*rdflib.ODRL2 attribute*), 453
- AllocateAction (*rdflib.SDO attribute*), 480
- AllProperties() (*in module rdflib.extras.infixowl*), 57
- allValuesFrom (*rdflib.extras.infixowl.Restriction property*), 68
- allValuesFrom (*rdflib.OWL attribute*), 464
- AllWheelDriveConfiguration (*rdflib.SDO attribute*), 480
- AlreadyBound, 176
- Alt (*class in rdflib.container*), 230
- Alt (*rdflib.RDF attribute*), 477
- alternateName (*rdflib.SDO attribute*), 534
- alternateOf (*rdflib.PROV attribute*), 470
- alternative (*rdflib.DCTERMS attribute*), 412
- alternativeHeadline (*rdflib.SDO attribute*), 534
- alternativeOf (*rdflib.SDO attribute*), 534
- AlternativePath (*class in rdflib.paths*), 282
- alternativePath (*rdflib.SH attribute*), 591
- altLabel (*rdflib.SKOS attribute*), 596
- alumni (*rdflib.SDO attribute*), 535
- alumniOf (*rdflib.SDO attribute*), 535
- amenityFeature (*rdflib.SDO attribute*), 535
- amount (*rdflib.SDO attribute*), 535
- amountOfThisGood (*rdflib.SDO attribute*), 535
- AmpStory (*rdflib.SDO attribute*), 480
- AMRadioChannel (*rdflib.SDO attribute*), 479
- AmusementPark (*rdflib.SDO attribute*), 480
- AnaerobicActivity (*rdflib.SDO attribute*), 480
- analyse() (*in module rdflib.plugins.sparql.algebra*), 148
- AnalysisNewsArticle (*rdflib.SDO attribute*), 480
- AnatomicalStructure (*rdflib.SDO attribute*), 481
- AnatomicalSystem (*rdflib.SDO attribute*), 481
- and\_() (*in module rdflib.plugins.sparql.operators*), 165
- AndConstraintComponent (*rdflib.SH attribute*), 588
- andSequence (*rdflib.ODRL2 attribute*), 455
- Anesthesia (*rdflib.SDO attribute*), 481
- Angle\_Sensor (*rdflib.BRICK attribute*), 338
- AnimalShelter (*rdflib.SDO attribute*), 481
- AnnotatableTerms (*class in rdflib.extras.infixowl*), 57



- `annotate` (*rdflib.ODRL2 attribute*), 455
- `annotatedProperty` (*rdflib.OWL attribute*), 464
- `annotatedSource` (*rdflib.OWL attribute*), 464
- `annotatedTarget` (*rdflib.OWL attribute*), 464
- `annotation` (*rdflib.extras.infixowl.Class property*), 61
- `Annotation` (*rdflib.OWL attribute*), 463
- `AnnotationProperty` (*rdflib.OWL attribute*), 463
- `annotationProperty` (*rdflib.SH attribute*), 592
- `annotationValue` (*rdflib.SH attribute*), 592
- `annotationVarName` (*rdflib.SH attribute*), 592
- `announcementLocation` (*rdflib.SDO attribute*), 535
- `annualPercentageRate` (*rdflib.SDO attribute*), 535
- `anonymize` (*rdflib.ODRL2 attribute*), 455
- `anonymousNode()` (*rdflib.plugins.parsers.notation3.SinkParser method*), 88
- `Answer` (*rdflib.SDO attribute*), 481
- `answerCount` (*rdflib.SDO attribute*), 535
- `answerExplanation` (*rdflib.SDO attribute*), 535
- `antagonist` (*rdflib.SDO attribute*), 535
- `anyone()` (*rdflib.container.Alt method*), 230
- `anyURI` (*rdflib.XSD attribute*), 609
- `Apartment` (*rdflib.SDO attribute*), 481
- `ApartmentComplex` (*rdflib.SDO attribute*), 481
- `APIReference` (*rdflib.SDO attribute*), 479
- `Appearance` (*rdflib.SDO attribute*), 481
- `appearance` (*rdflib.SDO attribute*), 535
- `append` (*rdflib.ODRL2 attribute*), 455
- `append()` (*rdflib.collection.Collection method*), 225
- `append()` (*rdflib.container.Container method*), 232
- `append()` (*rdflib.extras.infixowl.OWLRDFListProxy method*), 66
- `append_multiple()` (*rdflib.container.Container method*), 232
- `AppendAction` (*rdflib.SDO attribute*), 481
- `appendTo` (*rdflib.ODRL2 attribute*), 455
- `applicableLocation` (*rdflib.SDO attribute*), 535
- `applicantLocationRequirements` (*rdflib.SDO attribute*), 535
- `application` (*rdflib.SDO attribute*), 535
- `applicationCategory` (*rdflib.SDO attribute*), 535
- `applicationContact` (*rdflib.SDO attribute*), 535
- `applicationDeadline` (*rdflib.SDO attribute*), 535
- `applicationStartDate` (*rdflib.SDO attribute*), 535
- `applicationSubCategory` (*rdflib.SDO attribute*), 535
- `applicationSuite` (*rdflib.SDO attribute*), 535
- `appliesToDeliveryMethod` (*rdflib.SDO attribute*), 535
- `appliesToPaymentMethod` (*rdflib.SDO attribute*), 535
- `ApplyAction` (*rdflib.SDO attribute*), 481
- `ApprovedIndication` (*rdflib.SDO attribute*), 481
- `aq` (*rdflib.PROV attribute*), 470
- `Aquarium` (*rdflib.SDO attribute*), 481
- `archive` (*rdflib.ODRL2 attribute*), 455
- `ArchiveComponent` (*rdflib.SDO attribute*), 481
- `archivedAt` (*rdflib.SDO attribute*), 536
- `archiveHeld` (*rdflib.SDO attribute*), 536
- `ArchiveOrganization` (*rdflib.SDO attribute*), 481
- `ArchRepository` (*rdflib.DOAP attribute*), 414
- `area` (*rdflib.BRICK attribute*), 397
- `area` (*rdflib.SDO attribute*), 536
- `areaServed` (*rdflib.SDO attribute*), 536
- `arrivalAirport` (*rdflib.SDO attribute*), 536
- `arrivalBoatTerminal` (*rdflib.SDO attribute*), 536
- `arrivalBusStop` (*rdflib.SDO attribute*), 536
- `arrivalGate` (*rdflib.SDO attribute*), 536
- `arrivalPlatform` (*rdflib.SDO attribute*), 536
- `arrivalStation` (*rdflib.SDO attribute*), 536
- `arrivalTerminal` (*rdflib.SDO attribute*), 536
- `arrivalTime` (*rdflib.SDO attribute*), 536
- `ArriveAction` (*rdflib.SDO attribute*), 481
- `artEdition` (*rdflib.SDO attribute*), 536
- `arterialBranch` (*rdflib.SDO attribute*), 536
- `Artery` (*rdflib.SDO attribute*), 481
- `artform` (*rdflib.SDO attribute*), 536
- `ArtGallery` (*rdflib.SDO attribute*), 481
- `Article` (*rdflib.SDO attribute*), 481
- `articleBody` (*rdflib.SDO attribute*), 536
- `articleSection` (*rdflib.SDO attribute*), 536
- `artist` (*rdflib.SDO attribute*), 536
- `artMedium` (*rdflib.SDO attribute*), 536
- `artworkSurface` (*rdflib.SDO attribute*), 536
- `ascii()` (*in module rdflib.compat*), 230
- `asdict()` (*rdflib.query.ResultRow method*), 296
- `asInBundle` (*rdflib.PROV attribute*), 470
- `ask` (*rdflib.SH attribute*), 592
- `AskAction` (*rdflib.SDO attribute*), 481
- `askAnswer` (*rdflib.plugins.sparql.processor.SPARQLResult attribute*), 174
- `askAnswer` (*rdflib.plugins.sparql.results.jsonresults.JSONResult attribute*), 133
- `askAnswer` (*rdflib.plugins.sparql.results.rdfresults.RDFResult attribute*), 135
- `askAnswer` (*rdflib.plugins.sparql.results.xmlresults.XMLResult attribute*), 138
- `AskPublicNewsArticle` (*rdflib.SDO attribute*), 481
- `aspect` (*rdflib.SDO attribute*), 536
- `assembly` (*rdflib.SDO attribute*), 536
- `assemblyVersion` (*rdflib.SDO attribute*), 536
- `Assertion` (*rdflib.ODRL2 attribute*), 453
- `assertionProperty` (*rdflib.OWL attribute*), 464
- `Assertions` (*rdflib.XSD attribute*), 609
- `AssessAction` (*rdflib.SDO attribute*), 481
- `assesses` (*rdflib.SDO attribute*), 536
- `Asset` (*rdflib.ODRL2 attribute*), 453
- `AssetCollection` (*rdflib.ODRL2 attribute*), 453
- `AssetScope` (*rdflib.ODRL2 attribute*), 453
- `AssignAction` (*rdflib.SDO attribute*), 481
- `assignee` (*rdflib.ODRL2 attribute*), 455

- assigneeOf (*rdflib.ODRL2 attribute*), 455  
 assigner (*rdflib.ODRL2 attribute*), 455  
 assignerOf (*rdflib.ODRL2 attribute*), 455  
 associatedAnatomy (*rdflib.SDO attribute*), 536  
 associatedArticle (*rdflib.SDO attribute*), 536  
 associatedClaimReview (*rdflib.SDO attribute*), 536  
 associatedDisease (*rdflib.SDO attribute*), 536  
 associatedMedia (*rdflib.SDO attribute*), 537  
 associatedMediaReview (*rdflib.SDO attribute*), 537  
 associatedPathophysiology (*rdflib.SDO attribute*), 537  
 associatedReview (*rdflib.SDO attribute*), 537  
 Association (*rdflib.PROV attribute*), 468  
 AsymmetricProperty (*rdflib.OWL attribute*), 463  
 athlete (*rdflib.SDO attribute*), 537  
 Atlas (*rdflib.SDO attribute*), 481  
 atLocation (*rdflib.PROV attribute*), 470  
 Attachable (*rdflib.QB attribute*), 475  
 attachPolicy (*rdflib.ODRL2 attribute*), 455  
 attachSource (*rdflib.ODRL2 attribute*), 455  
 attendee (*rdflib.SDO attribute*), 537  
 attendees (*rdflib.SDO attribute*), 537  
 atTime (*rdflib.PROV attribute*), 470  
 Attorney (*rdflib.SDO attribute*), 481  
 attribute (*rdflib.ODRL2 attribute*), 455  
 attribute (*rdflib.QB attribute*), 476  
 attribute() (*rdflib.plugins.serializers.xmlwriter.XMLWriter* method), 123  
 attributedParty (*rdflib.ODRL2 attribute*), 455  
 AttributeProperty (*rdflib.QB attribute*), 475  
 attributingParty (*rdflib.ODRL2 attribute*), 455  
 Attribution (*rdflib.PROV attribute*), 468  
 audience (*rdflib.DCTERMS attribute*), 412  
 audience (*rdflib.DOAP attribute*), 414  
 Audience (*rdflib.SDO attribute*), 481  
 audience (*rdflib.SDO attribute*), 537  
 audienceType (*rdflib.SDO attribute*), 537  
 audio (*rdflib.SDO attribute*), 537  
 Audiobook (*rdflib.SDO attribute*), 482  
 AudiobookFormat (*rdflib.SDO attribute*), 482  
 AudioObject (*rdflib.SDO attribute*), 481  
 AudioObjectSnapshot (*rdflib.SDO attribute*), 482  
 audit\_hook() (*in module examples.secure\_with\_audit*), 25  
 AuditableStore (class in *rdflib.plugins.stores.auditable*), 189  
 Auditorium (*rdflib.BRICK attribute*), 338  
 authenticator (*rdflib.SDO attribute*), 537  
 author (*rdflib.SDO attribute*), 537  
 AuthoritativeLegalValue (*rdflib.SDO attribute*), 482  
 AuthorizeAction (*rdflib.SDO attribute*), 482  
 auto (*rdflib.CSVW attribute*), 401  
 AutoBodyShop (*rdflib.SDO attribute*), 482  
 AutoDealer (*rdflib.SDO attribute*), 482  
 Automated\_External\_Defibrillator (*rdflib.BRICK attribute*), 338  
 AutomatedTeller (*rdflib.SDO attribute*), 482  
 Automatic\_Mode\_Command (*rdflib.BRICK attribute*), 338  
 AutomotiveBusiness (*rdflib.SDO attribute*), 482  
 AutoPartsStore (*rdflib.SDO attribute*), 482  
 AutoRental (*rdflib.SDO attribute*), 482  
 AutoRepair (*rdflib.SDO attribute*), 482  
 AutoWash (*rdflib.SDO attribute*), 482  
 availability (*rdflib.SDO attribute*), 537  
 Availability\_Status (*rdflib.BRICK attribute*), 338  
 availabilityEnds (*rdflib.SDO attribute*), 537  
 availabilityStarts (*rdflib.SDO attribute*), 537  
 available (*rdflib.DCTERMS attribute*), 412  
 availableAtOrFrom (*rdflib.SDO attribute*), 537  
 availableChannel (*rdflib.SDO attribute*), 537  
 availableDeliveryMethod (*rdflib.SDO attribute*), 537  
 availableFrom (*rdflib.SDO attribute*), 537  
 availableIn (*rdflib.SDO attribute*), 537  
 availableLanguage (*rdflib.SDO attribute*), 537  
 availableOnDevice (*rdflib.SDO attribute*), 537  
 availableService (*rdflib.SDO attribute*), 537  
 availableStrength (*rdflib.SDO attribute*), 538  
 availableTest (*rdflib.SDO attribute*), 538  
 availableThrough (*rdflib.SDO attribute*), 538  
 Average (class in *rdflib.plugins.sparql.aggregates*), 141  
 Average\_Cooling\_Demand\_Sensor (*rdflib.BRICK attribute*), 338  
 Average\_Discharge\_Air\_Flow\_Sensor (*rdflib.BRICK attribute*), 338  
 Average\_Exhaust\_Air\_Static\_Pressure\_Sensor (*rdflib.BRICK attribute*), 338  
 Average\_Heating\_Demand\_Sensor (*rdflib.BRICK attribute*), 339  
 Average\_Supply\_Air\_Flow\_Sensor (*rdflib.BRICK attribute*), 339  
 Average\_Zone\_Air\_Temperature\_Sensor (*rdflib.BRICK attribute*), 339  
 award (*rdflib.SDO attribute*), 538  
 awards (*rdflib.SDO attribute*), 538  
 awayTeam (*rdflib.SDO attribute*), 538  
 Axiom (*rdflib.OWL attribute*), 463  
 Ayurvedic (*rdflib.SDO attribute*), 482  
 azimuth (*rdflib.BRICK attribute*), 397
- ## B
- BackgroundNewsArticle (*rdflib.SDO attribute*), 482  
 BackOrder (*rdflib.SDO attribute*), 482  
 backstory (*rdflib.SDO attribute*), 538  
 backwardCompatibleWith (*rdflib.OWL attribute*), 464  
 Bacteria (*rdflib.SDO attribute*), 482  
 BadSyntax, 81

- BadSyntax() (*rdflib.plugins.parsers.notation3.SinkParser* method), 86
- Bag (*class in rdflib.container*), 230
- Bag (*rdflib.RDF* attribute), 477
- BagID (*class in rdflib.plugins.parsers.rdfxml*), 102
- Bakery (*rdflib.SDO* attribute), 482
- Balance (*rdflib.SDO* attribute), 482
- BankAccount (*rdflib.SDO* attribute), 482
- bankAccountType (*rdflib.SDO* attribute), 538
- BankOrCreditUnion (*rdflib.SDO* attribute), 482
- Barcode (*rdflib.SDO* attribute), 482
- bareWord() (*rdflib.plugins.parsers.notation3.SinkParser* method), 88
- BarOrPub (*rdflib.SDO* attribute), 482
- base (*rdflib.CSVW* attribute), 401
- base (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 102
- base (*rdflib.plugins.serializers.hext.HextuplesSerializer* attribute), 114
- base (*rdflib.plugins.serializers.jsonld.JsonLDSerializer* attribute), 115
- base (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* attribute), 115
- base (*rdflib.plugins.serializers.n3.N3Serializer* attribute), 117
- base (*rdflib.plugins.serializers.nquads.NQuadsSerializer* attribute), 117
- base (*rdflib.plugins.serializers.nt.NTSerializer* attribute), 118
- base (*rdflib.plugins.serializers.rdfxml.PrettyXMLSerializer* attribute), 118
- base (*rdflib.plugins.serializers.rdfxml.XMLSerializer* attribute), 119
- base (*rdflib.plugins.serializers.trig.TrigSerializer* attribute), 120
- base (*rdflib.plugins.serializers.trix.TriXSerializer* attribute), 121
- base (*rdflib.plugins.serializers.turtle.RecursiveSerializer* attribute), 121
- base (*rdflib.plugins.serializers.turtle.TurtleSerializer* attribute), 122
- base (*rdflib.plugins.shared.jsonld.context.Context* property), 126
- base() (*in module rdflib.plugins.parsers.notation3*), 95
- base64Binary (*rdflib.XSD* attribute), 609
- Baseboard\_Radiator (*rdflib.BRICK* attribute), 339
- based\_near (*rdflib.FOAF* attribute), 421
- basedAt (*rdflib.ORG* attribute), 461
- Basement (*rdflib.BRICK* attribute), 339
- baseSalary (*rdflib.SDO* attribute), 538
- BasicIncome (*rdflib.SDO* attribute), 482
- BatchAddGraph (*class in rdflib.graph*), 240
- Battery (*rdflib.BRICK* attribute), 339
- Battery\_Energy\_Storage\_System (*rdflib.BRICK* attribute), 339
- Battery\_Room (*rdflib.BRICK* attribute), 339
- Battery\_Voltage\_Sensor (*rdflib.BRICK* attribute), 339
- BazaarBranch (*rdflib.DOAP* attribute), 414
- bbox (*rdflib.DCAT* attribute), 408
- bccRecipient (*rdflib.SDO* attribute), 538
- Beach (*rdflib.SDO* attribute), 482
- BeautySalon (*rdflib.SDO* attribute), 482
- becauseSubGraph() (*in module rdflib.plugins.parsers.trig*), 110
- bed (*rdflib.SDO* attribute), 538
- BedAndBreakfast (*rdflib.SDO* attribute), 482
- BedDetails (*rdflib.SDO* attribute), 482
- BedType (*rdflib.SDO* attribute), 483
- before (*rdflib.TIME* attribute), 601
- beforeMedia (*rdflib.SDO* attribute), 538
- BefriendAction (*rdflib.SDO* attribute), 483
- Bench\_Space (*rdflib.BRICK* attribute), 339
- beneficiaryBank (*rdflib.SDO* attribute), 538
- benefits (*rdflib.SDO* attribute), 538
- BenefitsHealthAspect (*rdflib.SDO* attribute), 483
- benefitsSummaryUrl (*rdflib.SDO* attribute), 538
- BerkeleyDB (*class in rdflib.plugins.stores.berkeleydb*), 192
- bestRating (*rdflib.SDO* attribute), 538
- BGP() (*in module rdflib.plugins.sparql.algebra*), 145
- bibliographicCitation (*rdflib.DCTERMS* attribute), 412
- BibliographicResource (*rdflib.DCTERMS* attribute), 410
- BikeStore (*rdflib.SDO* attribute), 483
- billingAddress (*rdflib.SDO* attribute), 538
- billingDuration (*rdflib.SDO* attribute), 538
- billingIncrement (*rdflib.SDO* attribute), 538
- billingPeriod (*rdflib.SDO* attribute), 538
- billingStart (*rdflib.SDO* attribute), 538
- bind() (*in module rdflib.term*), 329
- bind() (*rdflib.Graph* method), 429
- bind() (*rdflib.graph.Graph* method), 256
- bind() (*rdflib.graph.ReadOnlyGraphAggregate* method), 271
- bind() (*rdflib.namespace.NamespaceManager* method), 77
- bind() (*rdflib.plugins.parsers.notation3.RDFSink* method), 84
- bind() (*rdflib.plugins.parsers.notation3.SinkParser* method), 88
- bind() (*rdflib.plugins.sparql.sparql.Prologue* method), 181
- bind() (*rdflib.plugins.stores.auditable.AuditableStore* method), 189



- `bind()` (`rdflib.plugins.stores.berkeleydb.BerkeleyDB` method), 193
- `bind()` (`rdflib.plugins.stores.memory.Memory` method), 197
- `bind()` (`rdflib.plugins.stores.memory.SimpleMemory` method), 200
- `bind()` (`rdflib.plugins.stores.regexmatching.REGEXMatching` method), 202
- `bind()` (`rdflib.plugins.stores.sparqlstore.SPARQLStore` method), 207
- `bind()` (`rdflib.store.Store` method), 309
- `Bindings` (class in `rdflib.plugins.sparql.sparql`), 176
- `bindings` (`rdflib.query.Result` property), 293
- `BioChemEntity` (`rdflib.SDO` attribute), 483
- `bioChemInteraction` (`rdflib.SDO` attribute), 538
- `bioChemSimilarity` (`rdflib.SDO` attribute), 538
- `biologicalRole` (`rdflib.SDO` attribute), 538
- `biomechanicalClass` (`rdflib.SDO` attribute), 538
- `birthDate` (`rdflib.SDO` attribute), 538
- `birthday` (`rdflib.FOAF` attribute), 421
- `birthPlace` (`rdflib.SDO` attribute), 539
- `bitrate` (`rdflib.SDO` attribute), 539
- `BKRepository` (`rdflib.DOAP` attribute), 414
- `BlankNode` (`rdflib.SH` attribute), 588
- `blankNode()` (`rdflib.plugins.parsers.notation3.SinkParser` method), 88
- `BlankNodeOrIRI` (`rdflib.SH` attribute), 588
- `BlankNodeOrLiteral` (`rdflib.SH` attribute), 588
- `BLOCK_END` (`rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore` attribute), 212
- `BLOCK_FINDING_PATTERN` (`rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore` attribute), 212
- `BLOCK_START` (`rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore` attribute), 212
- `BlockContent` (`rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore` attribute), 212
- `BlockFinding` (`rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore` attribute), 212
- `blog` (`rdflib.DOAP` attribute), 414
- `Blog` (`rdflib.SDO` attribute), 483
- `blogPost` (`rdflib.SDO` attribute), 539
- `BlogPosting` (`rdflib.SDO` attribute), 483
- `blogPosts` (`rdflib.SDO` attribute), 539
- `bloodSupply` (`rdflib.SDO` attribute), 539
- `BloodTest` (`rdflib.SDO` attribute), 483
- `Blowdown_Water` (`rdflib.BRICK` attribute), 339
- `BNode` (class in `rdflib`), 335
- `BNode` (class in `rdflib.term`), 313
- `bnodes` (`rdflib.plugins.sparql.sparql.FrozenBindings` property), 178
- `boardingGroup` (`rdflib.SDO` attribute), 539
- `boardingPolicy` (`rdflib.SDO` attribute), 539
- `BoardingPolicyType` (`rdflib.SDO` attribute), 483
- `BoatReservation` (`rdflib.SDO` attribute), 483
- `BoatTerminal` (`rdflib.SDO` attribute), 483
- `BoatTrip` (`rdflib.SDO` attribute), 483
- `bodyLocation` (`rdflib.SDO` attribute), 539
- `BodyMeasurementArm` (`rdflib.SDO` attribute), 483
- `BodyMeasurementBust` (`rdflib.SDO` attribute), 483
- `BodyMeasurementChest` (`rdflib.SDO` attribute), 483
- `BodyMeasurementFoot` (`rdflib.SDO` attribute), 483
- `BodyMeasurementHand` (`rdflib.SDO` attribute), 483
- `BodyMeasurementHead` (`rdflib.SDO` attribute), 483
- `BodyMeasurementHeight` (`rdflib.SDO` attribute), 483
- `BodyMeasurementHips` (`rdflib.SDO` attribute), 483
- `BodyMeasurementInsideLeg` (`rdflib.SDO` attribute), 483
- `BodyMeasurementNeck` (`rdflib.SDO` attribute), 483
- `BodyMeasurementTypeEnumeration` (`rdflib.SDO` attribute), 483
- `BodyMeasurementUnderbust` (`rdflib.SDO` attribute), 484
- `BodyMeasurementWaist` (`rdflib.SDO` attribute), 484
- `BodyMeasurementWeight` (`rdflib.SDO` attribute), 484
- `BodyOfWater` (`rdflib.SDO` attribute), 484
- `bodyType` (`rdflib.SDO` attribute), 539
- `Boiler` (`rdflib.BRICK` attribute), 339
- `Bone` (`rdflib.SDO` attribute), 484
- `Book` (`rdflib.SDO` attribute), 484
- `bookEdition` (`rdflib.SDO` attribute), 539
- `bookFormat` (`rdflib.SDO` attribute), 539
- `BookFormatType` (`rdflib.SDO` attribute), 484
- `bookingAgent` (`rdflib.SDO` attribute), 539
- `bookingTime` (`rdflib.SDO` attribute), 539
- `BookmarkAction` (`rdflib.SDO` attribute), 484
- `BookSeries` (`rdflib.SDO` attribute), 484
- `BookStore` (`rdflib.SDO` attribute), 484
- `Boolean` (`rdflib.SDO` attribute), 484
- `booleanForm` (`rdflib.XSD` attribute), 609
- `BooleanClass` (class in `rdflib.extras.infixowl`), 59
- `BoosterFan` (`rdflib.BRICK` attribute), 339
- `bopen()` (in module `rdflib.compat`), 230
- `BorrowAction` (`rdflib.SDO` attribute), 484
- `borrower` (`rdflib.SDO` attribute), 539
- `bottomDataProperty` (`rdflib.OWL` attribute), 464
- `bottomObjectProperty` (`rdflib.OWL` attribute), 464
- `bounded` (`rdflib.XSD` attribute), 609
- `BowlingAlley` (`rdflib.SDO` attribute), 484
- `Box` (`rdflib.DCTERMS` attribute), 410
- `box` (`rdflib.SDO` attribute), 539
- `Box_Mode_Command` (`rdflib.BRICK` attribute), 339
- `BrainStructure` (`rdflib.SDO` attribute), 484
- `branch` (`rdflib.SDO` attribute), 539
- `branchCode` (`rdflib.SDO` attribute), 539
- `branchOf` (`rdflib.SDO` attribute), 539
- `Brand` (`rdflib.SDO` attribute), 484
- `brand` (`rdflib.SDO` attribute), 539

- breadcrumb (*rdflib.SDO attribute*), 539
- BreadcrumbList (*rdflib.SDO attribute*), 484
- Break\_Room (*rdflib.BRICK attribute*), 339
- Breaker\_Panel (*rdflib.BRICK attribute*), 339
- Breakroom (*rdflib.BRICK attribute*), 339
- breastfeedingWarning (*rdflib.SDO attribute*), 539
- Brewery (*rdflib.SDO attribute*), 484
- BRICK (*class in rdflib*), 336
- Bridge (*rdflib.SDO attribute*), 484
- Broadcast\_Room (*rdflib.BRICK attribute*), 339
- broadcastAffiliateOf (*rdflib.SDO attribute*), 539
- BroadcastChannel (*rdflib.SDO attribute*), 484
- broadcastChannelId (*rdflib.SDO attribute*), 539
- broadcastDisplayName (*rdflib.SDO attribute*), 539
- broadcaster (*rdflib.SDO attribute*), 540
- BroadcastEvent (*rdflib.SDO attribute*), 484
- broadcastFrequency (*rdflib.SDO attribute*), 539
- BroadcastFrequencySpecification (*rdflib.SDO attribute*), 484
- broadcastFrequencyValue (*rdflib.SDO attribute*), 539
- broadcastOfEvent (*rdflib.SDO attribute*), 539
- BroadcastRelease (*rdflib.SDO attribute*), 484
- BroadcastService (*rdflib.SDO attribute*), 484
- broadcastServiceTier (*rdflib.SDO attribute*), 540
- broadcastSignalModulation (*rdflib.SDO attribute*), 540
- broadcastSubChannel (*rdflib.SDO attribute*), 540
- broadcastTimezone (*rdflib.SDO attribute*), 540
- broader (*rdflib.SKOS attribute*), 596
- broaderTransitive (*rdflib.SKOS attribute*), 596
- broadMatch (*rdflib.SKOS attribute*), 596
- broker (*rdflib.SDO attribute*), 540
- BrokerageAccount (*rdflib.SDO attribute*), 484
- browse (*rdflib.DOAP attribute*), 414
- browserRequirements (*rdflib.SDO attribute*), 540
- BuddhistTemple (*rdflib.SDO attribute*), 484
- buffer (*rdflib.plugins.parsers.nquads.NQuadsParser attribute*), 98
- buffer (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser attribute*), 100
- Building (*rdflib.BRICK attribute*), 339
- Building\_Air (*rdflib.BRICK attribute*), 340
- Building\_Air\_Humidity\_Setpoint (*rdflib.BRICK attribute*), 340
- Building\_Air\_Static\_Pressure\_Sensor (*rdflib.BRICK attribute*), 340
- Building\_Air\_Static\_Pressure\_Setpoint (*rdflib.BRICK attribute*), 340
- Building\_Chilled\_Water\_Meter (*rdflib.BRICK attribute*), 340
- Building\_Electrical\_Meter (*rdflib.BRICK attribute*), 340
- Building\_Gas\_Meter (*rdflib.BRICK attribute*), 340
- Building\_Hot\_Water\_Meter (*rdflib.BRICK attribute*), 340
- Building\_Meter (*rdflib.BRICK attribute*), 340
- Building\_Water\_Meter (*rdflib.BRICK attribute*), 340
- buildingPrimaryFunction (*rdflib.BRICK attribute*), 397
- buildingThermalTransmittance (*rdflib.BRICK attribute*), 397
- buildPredicateHash() (*rdflib.plugins.serializers.turtle.RecursiveSerializer method*), 121
- Builtin\_ABS() (*in module rdflib.plugins.sparql.operators*), 156
- Builtin\_BNODE() (*in module rdflib.plugins.sparql.operators*), 156
- Builtin\_BOUND() (*in module rdflib.plugins.sparql.operators*), 157
- Builtin\_CEIL() (*in module rdflib.plugins.sparql.operators*), 157
- Builtin\_COALESCE() (*in module rdflib.plugins.sparql.operators*), 157
- Builtin\_CONCAT() (*in module rdflib.plugins.sparql.operators*), 157
- Builtin\_CONTAINS() (*in module rdflib.plugins.sparql.operators*), 157
- Builtin\_DATATYPE() (*in module rdflib.plugins.sparql.operators*), 157
- Builtin\_DAY() (*in module rdflib.plugins.sparql.operators*), 157
- Builtin\_ENCODE\_FOR\_URI() (*in module rdflib.plugins.sparql.operators*), 157
- Builtin\_EXISTS() (*in module rdflib.plugins.sparql.operators*), 158
- Builtin\_FLOOR() (*in module rdflib.plugins.sparql.operators*), 158
- Builtin\_HOURS() (*in module rdflib.plugins.sparql.operators*), 158
- Builtin\_IF() (*in module rdflib.plugins.sparql.operators*), 158
- Builtin\_IRI() (*in module rdflib.plugins.sparql.operators*), 158
- Builtin\_isBLANK() (*in module rdflib.plugins.sparql.operators*), 163
- Builtin\_isIRI() (*in module rdflib.plugins.sparql.operators*), 163
- Builtin\_isLITERAL() (*in module rdflib.plugins.sparql.operators*), 163
- Builtin\_isNUMERIC() (*in module rdflib.plugins.sparql.operators*), 163
- Builtin\_LANG() (*in module rdflib.plugins.sparql.operators*), 158
- Builtin\_LANGMATCHES() (*in module rdflib.plugins.sparql.operators*), 159
- Builtin\_LCASE() (*in module rdflib.plugins.sparql.operators*), 159



- flib.plugins.sparql.operators*), 159
- Builtin\_MD5() (in module *flib.plugins.sparql.operators*), 159
- Builtin\_MINUTES() (in module *flib.plugins.sparql.operators*), 159
- Builtin\_MONTH() (in module *flib.plugins.sparql.operators*), 159
- Builtin\_NOW() (in module *flib.plugins.sparql.operators*), 159
- Builtin\_RAND() (in module *flib.plugins.sparql.operators*), 159
- Builtin\_REGEX() (in module *flib.plugins.sparql.operators*), 159
- Builtin\_REPLACE() (in module *flib.plugins.sparql.operators*), 160
- Builtin\_ROUND() (in module *flib.plugins.sparql.operators*), 160
- Builtin\_sameTerm() (in module *flib.plugins.sparql.operators*), 163
- Builtin\_SECONDS() (in module *flib.plugins.sparql.operators*), 160
- Builtin\_SHA1() (in module *flib.plugins.sparql.operators*), 160
- Builtin\_SHA256() (in module *flib.plugins.sparql.operators*), 160
- Builtin\_SHA384() (in module *flib.plugins.sparql.operators*), 160
- Builtin\_SHA512() (in module *flib.plugins.sparql.operators*), 160
- Builtin\_STR() (in module *flib.plugins.sparql.operators*), 161
- Builtin\_STRAFTER() (in module *flib.plugins.sparql.operators*), 161
- Builtin\_STRBEFORE() (in module *flib.plugins.sparql.operators*), 161
- Builtin\_STRDT() (in module *flib.plugins.sparql.operators*), 161
- Builtin\_STREND() (in module *flib.plugins.sparql.operators*), 161
- Builtin\_STRLANG() (in module *flib.plugins.sparql.operators*), 161
- Builtin\_STRLen() (in module *flib.plugins.sparql.operators*), 161
- Builtin\_STRSTARTS() (in module *flib.plugins.sparql.operators*), 162
- Builtin\_STRUUID() (in module *flib.plugins.sparql.operators*), 162
- Builtin\_SUBSTR() (in module *flib.plugins.sparql.operators*), 162
- Builtin\_TIMEZONE() (in module *flib.plugins.sparql.operators*), 162
- Builtin\_TZ() (in module *flib.plugins.sparql.operators*), 162
- Builtin\_UCASE() (in module *flib.plugins.sparql.operators*), 162
- rd- Builtin\_UUID() (in module *flib.plugins.sparql.operators*), 162
- rd- Builtin\_YEAR() (in module *flib.plugins.sparql.operators*), 163
- rd- Bundle (*rdflib.PROV* attribute), 468
- rd- Bus\_Riser (*rdflib.BRICK* attribute), 340
- rd- BusinessAudience (*rdflib.SDO* attribute), 485
- rd- businessDays (*rdflib.SDO* attribute), 540
- rd- BusinessEntityType (*rdflib.SDO* attribute), 485
- rd- BusinessEvent (*rdflib.SDO* attribute), 485
- rd- BusinessFunction (*rdflib.SDO* attribute), 485
- rd- businessFunction (*rdflib.SDO* attribute), 540
- rd- BusinessSupport (*rdflib.SDO* attribute), 485
- rd- busName (*rdflib.SDO* attribute), 540
- rd- busNumber (*rdflib.SDO* attribute), 540
- rd- BusOrCoach (*rdflib.SDO* attribute), 485
- rd- BusReservation (*rdflib.SDO* attribute), 485
- rd- BusStation (*rdflib.SDO* attribute), 485
- rd- BusStop (*rdflib.SDO* attribute), 485
- rd- BusTrip (*rdflib.SDO* attribute), 485
- rd- BuyAction (*rdflib.SDO* attribute), 485
- rd- buyer (*rdflib.SDO* attribute), 540
- rd- byArtist (*rdflib.SDO* attribute), 540
- rd- byDay (*rdflib.SDO* attribute), 540
- rd- byMonth (*rdflib.SDO* attribute), 540
- rd- byMonthDay (*rdflib.SDO* attribute), 540
- rd- byMonthWeek (*rdflib.SDO* attribute), 540
- rd- Bypass\_Air (*rdflib.BRICK* attribute), 340
- rd- Bypass\_Air\_Flow\_Sensor (*rdflib.BRICK* attribute), 340
- rd- Bypass\_Air\_Humidity\_Setpoint (*rdflib.BRICK* attribute), 340
- rd- Bypass\_Command (*rdflib.BRICK* attribute), 340
- rd- Bypass\_Valve (*rdflib.BRICK* attribute), 340
- rd- Bypass\_Water (*rdflib.BRICK* attribute), 340
- rd- Bypass\_Water\_Flow\_Sensor (*rdflib.BRICK* attribute), 340
- rd- Bypass\_Water\_Flow\_Setpoint (*rdflib.BRICK* attribute), 340
- rd- byte (*rdflib.XSD* attribute), 609
- rd- byteSize (*rdflib.DCAT* attribute), 408
- rd- **C**
- rd- CableOrSatelliteService (*rdflib.SDO* attribute), 485
- rd- CafeOrCoffeeShop (*rdflib.SDO* attribute), 485
- rd- Cafeteria (*rdflib.BRICK* attribute), 341
- rd- calculateDuration() (in module *flib.plugins.sparql.operators*), 165
- rd- calculateFinalDateTime() (in module *flib.plugins.sparql.operators*), 165
- rd- Callable (class in *rdflib.extras.infixowl*), 59
- rd- callSign (*rdflib.SDO* attribute), 540
- rd- calories (*rdflib.SDO* attribute), 540

- Camera (*rdflib.BRICK* attribute), 341
- Campground (*rdflib.SDO* attribute), 485
- CampingPitch (*rdflib.SDO* attribute), 485
- Canal (*rdflib.SDO* attribute), 485
- CancelAction (*rdflib.SDO* attribute), 485
- candidate (*rdflib.SDO* attribute), 540
- Capacity\_Sensor (*rdflib.BRICK* attribute), 341
- caption (*rdflib.SDO* attribute), 540
- Car (*rdflib.SDO* attribute), 485
- carbohydrateContent (*rdflib.SDO* attribute), 540
- cardinality (*rdflib.extras.infixowl.Restriction* property), 68
- cardinality (*rdflib.OWL* attribute), 464
- cardinality (*rdflib.XSD* attribute), 609
- Cardiovascular (*rdflib.SDO* attribute), 485
- CardiovascularExam (*rdflib.SDO* attribute), 485
- cargoVolume (*rdflib.SDO* attribute), 540
- carrier (*rdflib.SDO* attribute), 540
- carrierRequirements (*rdflib.SDO* attribute), 540
- CarUsageType (*rdflib.SDO* attribute), 485
- CaseSeries (*rdflib.SDO* attribute), 485
- cashBack (*rdflib.SDO* attribute), 541
- Casino (*rdflib.SDO* attribute), 485
- CassetteFormat (*rdflib.SDO* attribute), 485
- cast\_bytes() (in module *rdflib.compat*), 230
- cast\_identifier() (in module *rdflib.extras.describer*), 51
- cast\_value() (in module *rdflib.extras.describer*), 51
- CastClass() (in module *rdflib.extras.infixowl*), 60
- Catalog (*rdflib.DCAT* attribute), 408
- catalog (*rdflib.DCAT* attribute), 408
- catalog (*rdflib.SDO* attribute), 541
- catalogNumber (*rdflib.SDO* attribute), 541
- CatalogRecord (*rdflib.DCAT* attribute), 408
- category (*rdflib.DOAP* attribute), 414
- category (*rdflib.PROV* attribute), 470
- category (*rdflib.SDO* attribute), 541
- CategoryCode (*rdflib.SDO* attribute), 486
- CategoryCodeSet (*rdflib.SDO* attribute), 486
- CatholicChurch (*rdflib.SDO* attribute), 486
- causeOf (*rdflib.SDO* attribute), 541
- CausesHealthAspect (*rdflib.SDO* attribute), 486
- CAV (*rdflib.BRICK* attribute), 341
- cbd() (*rdflib.Graph* method), 429
- cbd() (*rdflib.graph.Graph* method), 256
- ccRecipient (*rdflib.SDO* attribute), 541
- CDCPMDRecord (*rdflib.SDO* attribute), 485
- CDFormat (*rdflib.SDO* attribute), 485
- Ceiling\_Fan (*rdflib.BRICK* attribute), 341
- Cell (*rdflib.CSVW* attribute), 400
- Cemetery (*rdflib.SDO* attribute), 486
- Centrifugal\_Chiller (*rdflib.BRICK* attribute), 341
- centroid (*rdflib.DCAT* attribute), 408
- Change\_Filter\_Alarm (*rdflib.BRICK* attribute), 341
- changedBy (*rdflib.ORG* attribute), 461
- ChangeEvent (*rdflib.ORG* attribute), 461
- changeNote (*rdflib.SKOS* attribute), 596
- changeOperator() (*rdflib.extras.infixowl.BooleanClass* method), 59
- changes (*rdflib.VANN* attribute), 606
- Chapter (*rdflib.SDO* attribute), 486
- char (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 102
- character (*rdflib.SDO* attribute), 541
- characterAttribute (*rdflib.SDO* attribute), 541
- characterName (*rdflib.SDO* attribute), 541
- characters() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 103
- characters() (*rdflib.plugins.parsers.trix.TriXHandler* method), 110
- CharitableIncorporatedOrganization (*rdflib.SDO* attribute), 486
- cheatCode (*rdflib.SDO* attribute), 541
- CheckAction (*rdflib.SDO* attribute), 486
- checkDot() (*rdflib.plugins.parsers.notation3.SinkParser* method), 88
- CheckInAction (*rdflib.SDO* attribute), 486
- checkinTime (*rdflib.SDO* attribute), 541
- CheckoutAction (*rdflib.SDO* attribute), 486
- CheckoutPage (*rdflib.SDO* attribute), 486
- checkoutTime (*rdflib.SDO* attribute), 541
- checkSubject() (*rdflib.plugins.serializers.turtle.RecursiveSerializer* method), 121
- chemicalComposition (*rdflib.SDO* attribute), 541
- chemicalRole (*rdflib.SDO* attribute), 541
- ChemicalSubstance (*rdflib.SDO* attribute), 486
- ChildCare (*rdflib.SDO* attribute), 486
- childMaxAge (*rdflib.SDO* attribute), 541
- childMinAge (*rdflib.SDO* attribute), 541
- children (*rdflib.SDO* attribute), 541
- ChildrensEvent (*rdflib.SDO* attribute), 486
- childTaxon (*rdflib.SDO* attribute), 541
- Chilled\_Beam (*rdflib.BRICK* attribute), 341
- Chilled\_Water (*rdflib.BRICK* attribute), 341
- Chilled\_Water\_Coil (*rdflib.BRICK* attribute), 341
- Chilled\_Water\_Differential\_Pressure\_Deadband\_Setpoint (*rdflib.BRICK* attribute), 342
- Chilled\_Water\_Differential\_Pressure\_Integral\_Time\_Parameter (*rdflib.BRICK* attribute), 342
- Chilled\_Water\_Differential\_Pressure\_Load\_Shed\_Reset\_Status (*rdflib.BRICK* attribute), 342
- Chilled\_Water\_Differential\_Pressure\_Load\_Shed\_Setpoint (*rdflib.BRICK* attribute), 342
- Chilled\_Water\_Differential\_Pressure\_Load\_Shed\_Status (*rdflib.BRICK* attribute), 342
- Chilled\_Water\_Differential\_Pressure\_Proportional\_Band\_Parameter (*rdflib.BRICK* attribute), 342

Chilled\_Water\_Differential\_Pressure\_Sensor (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Differential\_Pressure\_Setpoint (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Differential\_Pressure\_Step\_Parameter (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Differential\_Temperature\_Sensor (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Discharge\_Flow\_Sensor (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Discharge\_Flow\_Setpoint (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Flow\_Sensor (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Flow\_Setpoint (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Loop (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Meter (rdflib.BRICK attribute), 342  
 Chilled\_Water\_Pump (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Pump\_Differential\_Pressure\_Deadband\_Setpoint (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Return\_Flow\_Sensor (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Return\_Temperature\_Sensor (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Static\_Pressure\_Setpoint (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Supply\_Flow\_Sensor (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Supply\_Flow\_Setpoint (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Supply\_Temperature\_Sensor (rdflib.BRICK attribute), 343  
 Chilled\_Water\_System (rdflib.BRICK attribute), 343  
 Chilled\_Water\_System\_Enable\_Command (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Temperature\_Sensor (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Temperature\_Setpoint (rdflib.BRICK attribute), 343  
 Chilled\_Water\_Valve (rdflib.BRICK attribute), 343  
 Chiller (rdflib.BRICK attribute), 343  
 Chiropractic (rdflib.SDO attribute), 486  
 cholesterolContent (rdflib.SDO attribute), 541  
 ChooseAction (rdflib.SDO attribute), 486  
 Church (rdflib.SDO attribute), 486  
 circle (rdflib.SDO attribute), 541  
 citation (rdflib.SDO attribute), 541  
 City (rdflib.SDO attribute), 486  
 CityHall (rdflib.SDO attribute), 486  
 CivicStructure (rdflib.SDO attribute), 486  
 Claim (rdflib.SDO attribute), 486  
 claimInterpreter (rdflib.SDO attribute), 541  
 ClaimReview (rdflib.SDO attribute), 486  
 claimReviewed (rdflib.SDO attribute), 541  
 Class (class in rdflib.extras.infixowl), 60  
 Class (rdflib.BRICK attribute), 343  
 Class (rdflib.OWL attribute), 463  
 Class (rdflib.RDFS attribute), 478  
 Class (rdflib.SDO attribute), 486  
 ClassConstraintComponent (rdflib.SH attribute), 588  
 classes (rdflib.VOID attribute), 607  
 classification (rdflib.ORG attribute), 461  
 ClassNamespaceFactory (class in rdflib.extras.infixowl), 62  
 classOrIdentifier() (in module rdflib.extras.infixowl), 69  
 classOrTerm() (in module rdflib.extras.infixowl), 69  
 classPartition (rdflib.VOID attribute), 607  
 clean() (rdflib.plugins.sparql.sparql.QueryContext method), 183  
 CleaningFee (rdflib.SDO attribute), 486  
 cleanup (rdflib.plugins.stores.concurrent.ResponsibleGenerator method), 196  
 clear() (rdflib.collection.Collection method), 225  
 clear() (rdflib.container.Container method), 232  
 clear() (rdflib.extras.infixowl.OWLRLDFListProxy method), 66  
 clearInDegree() (rdflib.extras.infixowl.Individual method), 65  
 clearOutDegree() (rdflib.extras.infixowl.Individual method), 65  
 clinicalPharmacology (rdflib.SDO attribute), 541  
 clinicalPharmacology (rdflib.SDO attribute), 541  
 Clinician (rdflib.SDO attribute), 486  
 Clip (rdflib.SDO attribute), 486  
 clipNumber (rdflib.SDO attribute), 541  
 clone() (rdflib.plugins.sparql.parserutils.CompValue method), 170  
 clone() (rdflib.plugins.sparql.sparql.QueryContext method), 183  
 close() (rdflib.Graph method), 430  
 close() (rdflib.graph.Graph method), 257  
 close() (rdflib.graph.ReadOnlyGraphAggregate method), 271  
 close() (rdflib.parser.InputSource method), 276  
 close() (rdflib.parser.PythonInputSource method), 277  
 close() (rdflib.plugins.parsers.notation3.Formula method), 83  
 close() (rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter method), 137  
 close() (rdflib.plugins.stores.auditale.AuditaleStore method), 190  
 close() (rdflib.plugins.stores.berkeleydb.BerkeleyDB method), 193  
 close() (rdflib.plugins.stores.regexmatching.REGEXMatching method), 202  
 close() (rdflib.store.Store method), 309

- Close\_Limit (*rdflib.BRICK* attribute), 343
- closed (*rdflib.SH* attribute), 592
- ClosedConstraintComponent (*rdflib.SH* attribute), 588
- ClosedNamespace (*class in rdflib.namespace*), 71
- closeMatch (*rdflib.SKOS* attribute), 596
- closes (*rdflib.SDO* attribute), 542
- ClothingStore (*rdflib.SDO* attribute), 486
- C0 (*rdflib.BRICK* attribute), 341
- C02 (*rdflib.BRICK* attribute), 341
- C02\_Alarm (*rdflib.BRICK* attribute), 341
- C02\_Differential\_Sensor (*rdflib.BRICK* attribute), 341
- C02\_Level\_Sensor (*rdflib.BRICK* attribute), 341
- C02\_Sensor (*rdflib.BRICK* attribute), 341
- C02\_Setpoint (*rdflib.BRICK* attribute), 341
- C0\_Differential\_Sensor (*rdflib.BRICK* attribute), 341
- C0\_Level\_Sensor (*rdflib.BRICK* attribute), 341
- C0\_Sensor (*rdflib.BRICK* attribute), 341
- coach (*rdflib.SDO* attribute), 542
- Code (*rdflib.SDO* attribute), 486
- code (*rdflib.SDO* attribute), 542
- CodedProperty (*rdflib.QB* attribute), 475
- codeList (*rdflib.QB* attribute), 476
- codeRepository (*rdflib.SDO* attribute), 542
- codeSampleType (*rdflib.SDO* attribute), 542
- codeValue (*rdflib.SDO* attribute), 542
- codingSystem (*rdflib.SDO* attribute), 542
- CohortStudy (*rdflib.SDO* attribute), 487
- Coil (*rdflib.BRICK* attribute), 343
- Cold\_Box (*rdflib.BRICK* attribute), 343
- Coldest\_Zone\_Air\_Temperature\_Sensor (*rdflib.BRICK* attribute), 343
- colleague (*rdflib.SDO* attribute), 542
- colleagues (*rdflib.SDO* attribute), 542
- collectAndRemoveFilters() (*in module rdflib.plugins.sparql.algebra*), 148
- Collection (*class in rdflib.collection*), 221
- Collection (*rdflib.BRICK* attribute), 344
- Collection (*rdflib.DCMITYPE* attribute), 409
- Collection (*rdflib.PROV* attribute), 468
- Collection (*rdflib.SDO* attribute), 487
- collection (*rdflib.SDO* attribute), 542
- Collection (*rdflib.SKOS* attribute), 595
- collection() (*rdflib.Graph* method), 430
- collection() (*rdflib.graph.Graph* method), 257
- Collection\_Basin\_Water (*rdflib.BRICK* attribute), 344
- Collection\_Basin\_Water\_Heater (*rdflib.BRICK* attribute), 344
- Collection\_Basin\_Water\_Level\_Alarm (*rdflib.BRICK* attribute), 344
- Collection\_Basin\_Water\_Level\_Sensor (*rdflib.BRICK* attribute), 344
- Collection\_Basin\_Water\_Temperature\_Sensor (*rdflib.BRICK* attribute), 344
- CollectionPage (*rdflib.SDO* attribute), 487
- collectionSize (*rdflib.SDO* attribute), 542
- CollegeOrUniversity (*rdflib.SDO* attribute), 487
- color (*rdflib.SDO* attribute), 542
- colorist (*rdflib.SDO* attribute), 542
- Column (*rdflib.CSVW* attribute), 400
- column (*rdflib.CSVW* attribute), 401
- columnReference (*rdflib.CSVW* attribute), 401
- ComedyClub (*rdflib.SDO* attribute), 487
- ComedyEvent (*rdflib.SDO* attribute), 487
- ComicCoverArt (*rdflib.SDO* attribute), 487
- ComicIssue (*rdflib.SDO* attribute), 487
- ComicSeries (*rdflib.SDO* attribute), 487
- ComicStory (*rdflib.SDO* attribute), 487
- Command (*rdflib.BRICK* attribute), 344
- commaSeparatedList() (*rdflib.plugins.parsers.notation3.SinkParser* method), 89
- comment (*rdflib.extras.infixowl.AnnotatableTerms* property), 58
- COMMENT (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* attribute), 212
- comment (*rdflib.RDFS* attribute), 479
- Comment (*rdflib.SDO* attribute), 487
- comment (*rdflib.SDO* attribute), 542
- CommentAction (*rdflib.SDO* attribute), 487
- commentCount (*rdflib.SDO* attribute), 542
- CommentPermission (*rdflib.SDO* attribute), 487
- commentPrefix (*rdflib.CSVW* attribute), 401
- commentText (*rdflib.SDO* attribute), 542
- commentTime (*rdflib.SDO* attribute), 542
- commercialize (*rdflib.ODRL2* attribute), 455
- commit() (*rdflib.Graph* method), 431
- commit() (*rdflib.graph.Graph* method), 258
- commit() (*rdflib.graph.ReadOnlyGraphAggregate* method), 272
- commit() (*rdflib.plugins.stores.auditable.AuditableStore* method), 190
- commit() (*rdflib.plugins.stores.regexmatching.REGEXMatching* method), 202
- commit() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 207
- commit() (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 214
- commit() (*rdflib.store.Store* method), 309
- Common\_Space (*rdflib.BRICK* attribute), 344
- CommonNSBindings() (*in module rdflib.extras.infixowl*), 62
- CommunicateAction (*rdflib.SDO* attribute), 487
- Communication (*rdflib.PROV* attribute), 468



- Communication\_Loss\_Alarm (*rdflib.BRICK* attribute), 344
- CommunityHealth (*rdflib.SDO* attribute), 487
- Comp (class in *rdflib.plugins.sparql.parserutils*), 169
- compare() (*rdflib.plugins.sparql.aggregates.Maximum* method), 144
- compare() (*rdflib.plugins.sparql.aggregates.Minimum* method), 144
- compatible() (*rdflib.plugins.sparql.sparql.FrozenDict* method), 180
- compensate (*rdflib.ODRL2* attribute), 455
- compensatedParty (*rdflib.ODRL2* attribute), 455
- compensatingParty (*rdflib.ODRL2* attribute), 455
- competencyRequired (*rdflib.SDO* attribute), 542
- competitor (*rdflib.SDO* attribute), 542
- CompilationAlbum (*rdflib.SDO* attribute), 487
- complementOf (*rdflib.extras.infixowl.Class* property), 61
- complementOf (*rdflib.OWL* attribute), 464
- Completed (*rdflib.SDO* attribute), 487
- CompletedActionStatus (*rdflib.SDO* attribute), 487
- CompleteDataFeed (*rdflib.SDO* attribute), 487
- component (*rdflib.PROV* attribute), 470
- component (*rdflib.QB* attribute), 476
- componentAttachment (*rdflib.QB* attribute), 476
- ComponentProperty (*rdflib.QB* attribute), 475
- componentProperty (*rdflib.QB* attribute), 476
- componentRequired (*rdflib.QB* attribute), 476
- ComponentSet (*rdflib.QB* attribute), 476
- ComponentSpecification (*rdflib.QB* attribute), 476
- ComponentTerms() (in module *rdflib.extras.infixowl*), 62
- composer (*rdflib.SDO* attribute), 542
- CompoundLiteral (*rdflib.RDF* attribute), 477
- CompoundPriceSpecification (*rdflib.SDO* attribute), 487
- compressFormat (*rdflib.DCAT* attribute), 408
- Compressor (*rdflib.BRICK* attribute), 344
- comprisedOf (*rdflib.SDO* attribute), 542
- compute\_qname() (*rdflib.Graph* method), 431
- compute\_qname() (*rdflib.graph.Graph* method), 258
- compute\_qname() (*rdflib.graph.ReadOnlyGraphAggregate* method), 272
- compute\_qname() (*rdflib.namespace.NamespaceManager* method), 77
- compute\_qname\_strict() (*rdflib.namespace.NamespaceManager* method), 77
- Computer\_Room\_Air\_Conditioning (*rdflib.BRICK* attribute), 344
- ComputerLanguage (*rdflib.SDO* attribute), 487
- ComputerStore (*rdflib.SDO* attribute), 487
- CompValue (class in *rdflib.plugins.sparql.parserutils*), 170
- concept (*rdflib.QB* attribute), 476
- Concept (*rdflib.SKOS* attribute), 596
- ConceptScheme (*rdflib.SKOS* attribute), 596
- Concession (*rdflib.BRICK* attribute), 344
- ConcurrentStore (class in *rdflib.plugins.stores.concurrent*), 195
- concurrentUse (*rdflib.ODRL2* attribute), 455
- Condensate\_Leak\_Alarm (*rdflib.BRICK* attribute), 344
- Condenser (*rdflib.BRICK* attribute), 344
- Condenser\_Heat\_Exchanger (*rdflib.BRICK* attribute), 344
- Condenser\_Water (*rdflib.BRICK* attribute), 344
- Condenser\_Water\_Bypass\_Valve (*rdflib.BRICK* attribute), 344
- Condenser\_Water\_Isolation\_Valve (*rdflib.BRICK* attribute), 344
- Condenser\_Water\_Pump (*rdflib.BRICK* attribute), 344
- Condenser\_Water\_System (*rdflib.BRICK* attribute), 345
- Condenser\_Water\_Temperature\_Sensor (*rdflib.BRICK* attribute), 345
- Condenser\_Water\_Valve (*rdflib.BRICK* attribute), 345
- Condensing\_Natural\_Gas\_Boiler (*rdflib.BRICK* attribute), 345
- condition (*rdflib.SH* attribute), 592
- ConditionalAndExpression() (in module *rdflib.plugins.sparql.operators*), 163
- ConditionalOrExpression() (in module *rdflib.plugins.sparql.operators*), 163
- conditionsOfAccess (*rdflib.SDO* attribute), 542
- Conductivity\_Sensor (*rdflib.BRICK* attribute), 345
- Conference\_Room (*rdflib.BRICK* attribute), 345
- ConfirmAction (*rdflib.SDO* attribute), 487
- confirmationNumber (*rdflib.SDO* attribute), 542
- conflict (*rdflib.ODRL2* attribute), 455
- ConflictTerm (*rdflib.ODRL2* attribute), 453
- conforms (*rdflib.SH* attribute), 592
- conformsTo (*rdflib.DCTERMS* attribute), 412
- ConjunctiveGraph (class in *rdflib*), 403
- ConjunctiveGraph (class in *rdflib.graph*), 241
- connected() (*rdflib.Graph* method), 431
- connected() (*rdflib.graph.Graph* method), 258
- connectedTo (*rdflib.SDO* attribute), 542
- consentedParty (*rdflib.ODRL2* attribute), 455
- consentingParty (*rdflib.ODRL2* attribute), 455
- consequence (*rdflib.ODRL2* attribute), 455
- Consortium (*rdflib.SDO* attribute), 487
- Constant\_Air\_Volume\_Box (*rdflib.BRICK* attribute), 345
- constrainingProperty (*rdflib.SDO* attribute), 542
- Constraint (*rdflib.ODRL2* attribute), 453
- constraint (*rdflib.ODRL2* attribute), 456
- ConstraintComponent (*rdflib.SH* attribute), 588
- constraints (*rdflib.PROV* attribute), 470

- `construct` (*rdflib.SH* attribute), 592
- `ConsumeAction` (*rdflib.SDO* attribute), 487
- `Contact_Sensor` (*rdflib.BRICK* attribute), 345
- `contactlessPayment` (*rdflib.SDO* attribute), 543
- `contactOption` (*rdflib.SDO* attribute), 542
- `ContactPage` (*rdflib.SDO* attribute), 487
- `contactPoint` (*rdflib.DCAT* attribute), 408
- `ContactPoint` (*rdflib.SDO* attribute), 488
- `contactPoint` (*rdflib.SDO* attribute), 542
- `ContactPointOption` (*rdflib.SDO* attribute), 488
- `contactPoints` (*rdflib.SDO* attribute), 542
- `contactType` (*rdflib.SDO* attribute), 543
- `ContagiousnessHealthAspect` (*rdflib.SDO* attribute), 488
- `containedIn` (*rdflib.SDO* attribute), 543
- `containedInPlace` (*rdflib.SDO* attribute), 543
- `Container` (class in *rdflib.container*), 231
- `container` (*rdflib.plugins.shared.jsonld.context.Term* attribute), 129
- `Container` (*rdflib.RDFS* attribute), 478
- `ContainerMembershipProperty` (*rdflib.RDFS* attribute), 478
- `containsPlace` (*rdflib.SDO* attribute), 543
- `containsSeason` (*rdflib.SDO* attribute), 543
- `content_type` (*rdflib.parser.PythonInputSource* attribute), 277
- `content_type` (*rdflib.parser.StringInputSource* attribute), 278
- `contentLocation` (*rdflib.SDO* attribute), 543
- `contentRating` (*rdflib.SDO* attribute), 543
- `contentReferenceTime` (*rdflib.SDO* attribute), 543
- `contentSize` (*rdflib.SDO* attribute), 543
- `contentType` (*rdflib.SDO* attribute), 543
- `contentUrl` (*rdflib.SDO* attribute), 543
- `Context` (class in *rdflib.plugins.shared.jsonld.context*), 124
- `context` (*rdflib.plugins.shared.jsonld.context.Term* attribute), 129
- `context_aware` (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* attribute), 193
- `context_aware` (*rdflib.plugins.stores.memory.Memory* attribute), 197
- `context_aware` (*rdflib.store.Store* attribute), 309
- `context_from_urlinputsource()` (in module *rdflib.plugins.shared.jsonld.util*), 130
- `context_id()` (*rdflib.ConjunctiveGraph* method), 405
- `context_id()` (*rdflib.graph.ConjunctiveGraph* method), 243
- `contexts()` (*rdflib.ConjunctiveGraph* method), 405
- `contexts()` (*rdflib.Dataset* method), 419
- `contexts()` (*rdflib.graph.ConjunctiveGraph* method), 243
- `contexts()` (*rdflib.graph.Dataset* method), 248
- `contexts()` (*rdflib.plugins.stores.auditable.AuditableStore* method), 190
- `contexts()` (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* method), 193
- `contexts()` (*rdflib.plugins.stores.memory.Memory* method), 197
- `contexts()` (*rdflib.plugins.stores.regexmatching.REGEXMatching* method), 202
- `contexts()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 207
- `contexts()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 214
- `contexts()` (*rdflib.store.Store* method), 309
- `Continent` (*rdflib.SDO* attribute), 488
- `contractedParty` (*rdflib.ODRL2* attribute), 456
- `contractingParty` (*rdflib.ODRL2* attribute), 456
- `contraindication` (*rdflib.SDO* attribute), 543
- `Contribute` (*rdflib.PROV* attribute), 468
- `contributed` (*rdflib.PROV* attribute), 470
- `contributor` (*rdflib.DC* attribute), 407
- `contributor` (*rdflib.DCTERMS* attribute), 412
- `Contributor` (*rdflib.PROV* attribute), 468
- `contributor` (*rdflib.SDO* attribute), 543
- `Control_Room` (*rdflib.BRICK* attribute), 345
- `ControlAction` (*rdflib.SDO* attribute), 488
- `ConvenienceStore` (*rdflib.SDO* attribute), 488
- `Conversation` (*rdflib.SDO* attribute), 488
- `conversionEfficiency` (*rdflib.BRICK* attribute), 397
- `convert()` (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 103
- `convert()` (*rdflib.tools.csv2rdf.CSV2RDF* method), 219
- `convertTerm()` (*rdflib.plugins.sparql.results.csvresults.CSVResultParser* method), 132
- `convertTerm()` (*rdflib.plugins.sparql.results.tsvresults.TSVResultParser* method), 135
- `CookAction` (*rdflib.SDO* attribute), 488
- `cookingMethod` (*rdflib.SDO* attribute), 543
- `cookTime` (*rdflib.SDO* attribute), 543
- `Cooling_Coil` (*rdflib.BRICK* attribute), 345
- `Cooling_Command` (*rdflib.BRICK* attribute), 345
- `Cooling_Demand_Sensor` (*rdflib.BRICK* attribute), 345
- `Cooling_Demand_Setpoint` (*rdflib.BRICK* attribute), 345
- `Cooling_Discharge_Air_Flow_Setpoint` (*rdflib.BRICK* attribute), 345
- `Cooling_Discharge_Air_Temperature_Deadband_Setpoint` (*rdflib.BRICK* attribute), 345
- `Cooling_Discharge_Air_Temperature_Integral_Time_Parameter` (*rdflib.BRICK* attribute), 345
- `Cooling_Discharge_Air_Temperature_Proportional_Band_Parameter` (*rdflib.BRICK* attribute), 345
- `Cooling_Start_Stop_Status` (*rdflib.BRICK* attribute), 345
- `Cooling_Supply_Air_Flow_Setpoint` (*rdflib.BRICK* attribute), 345

- attribute), 346
- Cooling\_Supply\_Air\_Temperature\_Deadband\_Setpoint (rdflib.BRICK attribute), 346
- Cooling\_Supply\_Air\_Temperature\_Integral\_Time\_Parameter (rdflib.BRICK attribute), 346
- Cooling\_Supply\_Air\_Temperature\_Proportional\_Band\_Parameter (rdflib.BRICK attribute), 346
- Cooling\_Temperature\_Setpoint (rdflib.BRICK attribute), 346
- Cooling\_Tower (rdflib.BRICK attribute), 346
- Cooling\_Tower\_Fan (rdflib.BRICK attribute), 346
- Cooling\_Valve (rdflib.BRICK attribute), 346
- coolingCapacity (rdflib.BRICK attribute), 397
- CoOp (rdflib.SDO attribute), 486
- coordinates (rdflib.BRICK attribute), 398
- copy (rdflib.ODRL2 attribute), 456
- copy() (rdflib.extras.infixowl.BooleanClass method), 59
- Copy\_Room (rdflib.BRICK attribute), 346
- Copyright (rdflib.PROV attribute), 468
- copyrightHolder (rdflib.SDO attribute), 543
- copyrightNotice (rdflib.SDO attribute), 543
- copyrightYear (rdflib.SDO attribute), 543
- core (rdflib.ODRL2 attribute), 456
- Core\_Temperature\_Sensor (rdflib.BRICK attribute), 346
- Core\_Temperature\_Setpoint (rdflib.BRICK attribute), 346
- Corporation (rdflib.SDO attribute), 488
- correction (rdflib.SDO attribute), 543
- CorrectionComment (rdflib.SDO attribute), 488
- correctionsPolicy (rdflib.SDO attribute), 543
- costCategory (rdflib.SDO attribute), 543
- costCurrency (rdflib.SDO attribute), 543
- costOrigin (rdflib.SDO attribute), 543
- costPerUnit (rdflib.SDO attribute), 543
- count (rdflib.ODRL2 attribute), 456
- Counter (class in rdflib.plugins.sparql.aggregates), 142
- countriesNotSupported (rdflib.SDO attribute), 543
- countriesSupported (rdflib.SDO attribute), 543
- Country (rdflib.SDO attribute), 488
- countryOfAssembly (rdflib.SDO attribute), 544
- countryOfLastProcessing (rdflib.SDO attribute), 544
- countryOfOrigin (rdflib.SDO attribute), 544
- Course (rdflib.SDO attribute), 488
- course (rdflib.SDO attribute), 544
- courseCode (rdflib.SDO attribute), 544
- CourseInstance (rdflib.SDO attribute), 488
- courseMode (rdflib.SDO attribute), 544
- coursePrerequisites (rdflib.SDO attribute), 544
- courseWorkload (rdflib.SDO attribute), 544
- Courthouse (rdflib.SDO attribute), 488
- coverage (rdflib.DC attribute), 407
- coverage (rdflib.DCTERMS attribute), 412
- coverageEndTime (rdflib.SDO attribute), 544
- coverageStartTime (rdflib.SDO attribute), 544
- CoverArt (rdflib.SDO attribute), 488
- CovidTestingFacility (rdflib.SDO attribute), 488
- CREATE (rdflib.BRICK attribute), 341
- Create (rdflib.PROV attribute), 468
- CreateAction (rdflib.plugins.stores.sparqlstore.SPARQLStore method), 207
- create() (rdflib.store.Store method), 309
- create\_parser() (in module rdflib.plugins.parsers.rdfxml), 108
- create\_parser() (in module rdflib.plugins.parsers.trix), 113
- CreateAction (rdflib.SDO attribute), 488
- created (rdflib.DCTERMS attribute), 412
- created (rdflib.DOAP attribute), 414
- CreativeWork (rdflib.SDO attribute), 488
- CreativeWorkSeason (rdflib.SDO attribute), 488
- CreativeWorkSeries (rdflib.SDO attribute), 488
- creativeWorkStatus (rdflib.SDO attribute), 544
- creator (rdflib.DC attribute), 407
- creator (rdflib.DCTERMS attribute), 412
- Creator (rdflib.PROV attribute), 468
- creator (rdflib.SDO attribute), 544
- credentialCategory (rdflib.SDO attribute), 544
- CreditCard (rdflib.SDO attribute), 488
- creditedTo (rdflib.SDO attribute), 544
- creditText (rdflib.SDO attribute), 544
- Crematorium (rdflib.SDO attribute), 488
- CriticReview (rdflib.SDO attribute), 488
- CrossSectional (rdflib.SDO attribute), 488
- cssSelector (rdflib.SDO attribute), 544
- CssSelectorType (rdflib.SDO attribute), 488
- CSV2RDF (class in rdflib.tools.csv2rdf), 219
- csvEncodedTabularData (rdflib.CSVW attribute), 401
- CSVResultParser (class in rdflib.plugins.sparql.results.csvresults), 131
- CSVResultSerializer (class in rdflib.plugins.sparql.results.csvresults), 132
- CSVW (class in rdflib), 400
- CT (rdflib.SDO attribute), 485
- Cubicle (rdflib.BRICK attribute), 346
- currenciesAccepted (rdflib.SDO attribute), 544
- currency (rdflib.SDO attribute), 544
- CurrencyConversionService (rdflib.SDO attribute), 488
- current (rdflib.plugins.parsers.rdfxml.RDFXMLHandler property), 104
- Current\_Imbalance\_Sensor (rdflib.BRICK attribute), 346
- Current\_Limit (rdflib.BRICK attribute), 346
- Current\_Output\_Sensor (rdflib.BRICK attribute), 346
- Current\_Sensor (rdflib.BRICK attribute), 346
- currentExchangeRate (rdflib.SDO attribute), 544
- currentFlowType (rdflib.BRICK attribute), 398



- `currentProject` (*rdflib.FOAF* attribute), 421
- `Curtailment_Override_Command` (*rdflib.BRICK* attribute), 346
- `CUSTOM_EVALS` (in module *rdflib.plugins.sparql*), 188
- `custom_function()` (in module *rdflib.plugins.sparql.operators*), 166
- `customer` (*rdflib.SDO* attribute), 544
- `customerRemorseReturnFees` (*rdflib.SDO* attribute), 544
- `customerRemorseReturnLabelSource` (*rdflib.SDO* attribute), 544
- `customerRemorseReturnShippingFeesAmount` (*rdflib.SDO* attribute), 544
- `customEval()` (in module *examples.custom\_eval*), 22
- `cutoffTime` (*rdflib.SDO* attribute), 545
- `cvdCollectionDate` (*rdflib.SDO* attribute), 545
- `cvdFacilityCounty` (*rdflib.SDO* attribute), 545
- `cvdFacilityId` (*rdflib.SDO* attribute), 545
- `cvdNumBeds` (*rdflib.SDO* attribute), 545
- `cvdNumBedsOcc` (*rdflib.SDO* attribute), 545
- `cvdNumC19Died` (*rdflib.SDO* attribute), 545
- `cvdNumC19HOPats` (*rdflib.SDO* attribute), 545
- `cvdNumC19HospPats` (*rdflib.SDO* attribute), 545
- `cvdNumC19MechVentPats` (*rdflib.SDO* attribute), 545
- `cvdNumC190FMechVentPats` (*rdflib.SDO* attribute), 545
- `cvdNumC190verflowPats` (*rdflib.SDO* attribute), 545
- `cvdNumICUBeds` (*rdflib.SDO* attribute), 545
- `cvdNumICUBedsOcc` (*rdflib.SDO* attribute), 545
- `cvdNumTotBeds` (*rdflib.SDO* attribute), 545
- `cvdNumVent` (*rdflib.SDO* attribute), 545
- `cvdNumVentUse` (*rdflib.SDO* attribute), 545
- `CVSRepository` (*rdflib.DOAP* attribute), 414
- `Cycle_Alarm` (*rdflib.BRICK* attribute), 346
- D**
- `DamagedCondition` (*rdflib.SDO* attribute), 489
- `Damper` (*rdflib.BRICK* attribute), 347
- `Damper_Command` (*rdflib.BRICK* attribute), 347
- `Damper_Position_Command` (*rdflib.BRICK* attribute), 347
- `Damper_Position_Sensor` (*rdflib.BRICK* attribute), 347
- `Damper_Position_Setpoint` (*rdflib.BRICK* attribute), 347
- `DanceEvent` (*rdflib.SDO* attribute), 489
- `DanceGroup` (*rdflib.SDO* attribute), 489
- `DarcsRepository` (*rdflib.DOAP* attribute), 414
- `data` (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 102
- `DataCatalog` (*rdflib.SDO* attribute), 489
- `DataDownload` (*rdflib.SDO* attribute), 489
- `dataDump` (*rdflib.VOID* attribute), 607
- `DataFeed` (*rdflib.SDO* attribute), 489
- `dataFeedElement` (*rdflib.SDO* attribute), 545
- `DataFeedItem` (*rdflib.SDO* attribute), 489
- `DataRange` (*rdflib.OWL* attribute), 463
- `DataService` (*rdflib.DCAT* attribute), 408
- `Dataset` (class in *rdflib*), 415
- `Dataset` (class in *rdflib.graph*), 245
- `Dataset` (*rdflib.DCAT* attribute), 408
- `dataset` (*rdflib.DCAT* attribute), 408
- `Dataset` (*rdflib.DCMITYPE* attribute), 409
- `dataset` (*rdflib.plugins.sparql.sparql.QueryContext* property), 183
- `DataSet` (*rdflib.QB* attribute), 476
- `dataSet` (*rdflib.QB* attribute), 476
- `Dataset` (*rdflib.SDO* attribute), 489
- `dataset` (*rdflib.SDO* attribute), 545
- `Dataset` (*rdflib.VOID* attribute), 607
- `DatasetDescription` (*rdflib.VOID* attribute), 607
- `datasetTimeInterval` (*rdflib.SDO* attribute), 545
- `DataStructureDefinition` (*rdflib.QB* attribute), 476
- `Datatype` (*rdflib.CSVW* attribute), 400
- `datatype` (*rdflib.CSVW* attribute), 401
- `datatype` (*rdflib.Literal* property), 449
- `dataType` (*rdflib.ODRL2* attribute), 456
- `datatype` (*rdflib.plugins.parsers.RDFVOC.RDFVOC* attribute), 79
- `datatype` (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 102
- `Datatype` (*rdflib.RDFS* attribute), 478
- `DataType` (*rdflib.SDO* attribute), 489
- `datatype` (*rdflib.SH* attribute), 592
- `datatype` (*rdflib.term.Literal* property), 324
- `datatypeComplementOf` (*rdflib.OWL* attribute), 465
- `DatatypeConstraintComponent` (*rdflib.SH* attribute), 588
- `DatatypeProperty` (*rdflib.OWL* attribute), 463
- `date` (*rdflib.DC* attribute), 407
- `date` (*rdflib.DCTERMS* attribute), 412
- `Date` (*rdflib.SDO* attribute), 489
- `date` (*rdflib.XSD* attribute), 609
- `date()` (in module *rdflib.plugins.sparql.operators*), 166
- `date_time()` (in module *rdflib.util*), 330
- `dateAccepted` (*rdflib.DCTERMS* attribute), 412
- `dateCopyrighted` (*rdflib.DCTERMS* attribute), 412
- `dateCreated` (*rdflib.SDO* attribute), 545
- `dateDeleted` (*rdflib.SDO* attribute), 545
- `DatedMoneySpecification` (*rdflib.SDO* attribute), 489
- `dateIssued` (*rdflib.SDO* attribute), 545
- `dateline` (*rdflib.SDO* attribute), 546
- `dateModified` (*rdflib.SDO* attribute), 545
- `datePosted` (*rdflib.SDO* attribute), 545
- `datePublished` (*rdflib.SDO* attribute), 546
- `dateRead` (*rdflib.SDO* attribute), 546
- `dateReceived` (*rdflib.SDO* attribute), 546
- `dateSent` (*rdflib.SDO* attribute), 546
- `dateSubmitted` (*rdflib.DCTERMS* attribute), 412



- `dateTime` (*rdflib.ODRL2 attribute*), 456
- `DateTime` (*rdflib.SDO attribute*), 489
- `dateTime` (*rdflib.XSD attribute*), 609
- `datetime()` (in module *rdflib.plugins.sparql.operators*), 166
- `DateTimeDescription` (*rdflib.TIME attribute*), 600
- `DateTimeInterval` (*rdflib.TIME attribute*), 600
- `dateTimeObjects()` (in module *rdflib.plugins.sparql.operators*), 166
- `dateTimeStamp` (*rdflib.XSD attribute*), 610
- `dateVehicleFirstRegistered` (*rdflib.SDO attribute*), 546
- `day` (*rdflib.TIME attribute*), 601
- `day` (*rdflib.XSD attribute*), 610
- `DayOfWeek` (*rdflib.SDO attribute*), 489
- `dayOfWeek` (*rdflib.SDO attribute*), 546
- `DayOfWeek` (*rdflib.TIME attribute*), 600
- `dayOfWeek` (*rdflib.TIME attribute*), 601
- `dayOfYear` (*rdflib.TIME attribute*), 601
- `days` (*rdflib.TIME attribute*), 601
- `DaySpa` (*rdflib.SDO attribute*), 489
- `dayTimeDuration` (*rdflib.XSD attribute*), 610
- `db_env` (*rdflib.plugins.stores.berkeleydb.BerkeleyDB attribute*), 193
- `DC` (class in *rdflib*), 407
- `DC_Bus_Voltage_Sensor` (*rdflib.BRICK attribute*), 347
- `DCAT` (class in *rdflib*), 408
- `DCMITYPE` (class in *rdflib*), 409
- `DCMIType` (*rdflib.DCTERMS attribute*), 410
- `DCTERMS` (class in *rdflib*), 410
- `DDC` (*rdflib.DCTERMS attribute*), 410
- `DDxElement` (*rdflib.SDO attribute*), 489
- `de_skolemize()` (*rdflib.Graph method*), 431
- `de_skolemize()` (*rdflib.graph.Graph method*), 258
- `de_skolemize()` (*rdflib.term.URIRef method*), 328
- `de_skolemize()` (*rdflib.URIRef method*), 606
- `DeactivateAction` (*rdflib.SDO attribute*), 489
- `deactivated` (*rdflib.SH attribute*), 592
- `Deadband_Setpoint` (*rdflib.BRICK attribute*), 347
- `deathDate` (*rdflib.SDO attribute*), 546
- `deathPlace` (*rdflib.SDO attribute*), 546
- `Deceleration_Time_Setpoint` (*rdflib.BRICK attribute*), 347
- `decimal` (*rdflib.XSD attribute*), 610
- `decimalChar` (*rdflib.CSVW attribute*), 401
- `declare` (*rdflib.SH attribute*), 592
- `declared` (*rdflib.plugins.parsers.rdfxml.ElementHandler attribute*), 102
- `declareExistential()` (*rdflib.plugins.parsers.notation3.Formula method*), 83
- `decodeStringEscape()` (in module *rdflib.compat*), 230
- `decodeUnicodeEscape()` (in module *rdflib.compat*), 230
- `DecontextualizedContent` (*rdflib.SDO attribute*), 489
- `Dedicated_Outdoor_Air_System_Unit` (*rdflib.BRICK attribute*), 347
- `DeepClassClear()` (in module *rdflib.extras.infixowl*), 62
- `default` (*rdflib.CSVW attribute*), 401
- `default_cast()` (in module *rdflib.plugins.sparql.operators*), 166
- `default_context` (*rdflib.ConjunctiveGraph attribute*), 405
- `default_context` (*rdflib.Dataset attribute*), 419
- `default_context` (*rdflib.graph.Dataset attribute*), 248
- `default_context` (*rdflib.graph.ReadOnlyGraphAggregate attribute*), 272
- `defaultValue` (*rdflib.SDO attribute*), 546
- `defaultValue` (*rdflib.SH attribute*), 592
- `DefenceEstablishment` (*rdflib.SDO attribute*), 489
- `Defined` (class in *rdflib.plugins.shared.jsonld.context*), 129
- `DefinedNamespace` (class in *rdflib.namespace*), 72
- `DefinedRegion` (*rdflib.SDO attribute*), 489
- `DefinedTerm` (*rdflib.SDO attribute*), 489
- `DefinedTermSet` (*rdflib.SDO attribute*), 489
- `definition` (*rdflib.PROV attribute*), 470
- `definition` (*rdflib.SKOS attribute*), 596
- `DefinitiveLegalValue` (*rdflib.SDO attribute*), 489
- `defrag()` (*rdflib.term.URIRef method*), 328
- `defrag()` (*rdflib.URIRef method*), 606
- `Dehumidification_Start_Stop_Status` (*rdflib.BRICK attribute*), 347
- `Deionised_Water_Conductivity_Sensor` (*rdflib.BRICK attribute*), 347
- `Deionised_Water_Level_Sensor` (*rdflib.BRICK attribute*), 347
- `Deionized_Water` (*rdflib.BRICK attribute*), 347
- `Deionized_Water_Alarm` (*rdflib.BRICK attribute*), 347
- `Delay_Parameter` (*rdflib.BRICK attribute*), 347
- `delayPeriod` (*rdflib.ODRL2 attribute*), 456
- `Delegation` (*rdflib.PROV attribute*), 468
- `delete` (*rdflib.ODRL2 attribute*), 456
- `delete()` (*rdflib.extras.infixowl.Individual method*), 65
- `DeleteAction` (*rdflib.SDO attribute*), 489
- `delimiter` (*rdflib.CSVW attribute*), 401
- `deliveryAddress` (*rdflib.SDO attribute*), 546
- `deliveryChannel` (*rdflib.ODRL2 attribute*), 456
- `DeliveryChargeSpecification` (*rdflib.SDO attribute*), 489
- `DeliveryEvent` (*rdflib.SDO attribute*), 490
- `deliveryLeadTime` (*rdflib.SDO attribute*), 546
- `DeliveryMethod` (*rdflib.SDO attribute*), 490
- `deliveryMethod` (*rdflib.SDO attribute*), 546
- `deliveryStatus` (*rdflib.SDO attribute*), 546
- `deliveryTime` (*rdflib.SDO attribute*), 546
- `DeliveryTimeSettings` (*rdflib.SDO attribute*), 490

- Demand (*rdflib.SDO* attribute), 490
- Demand\_Sensor (*rdflib.BRICK* attribute), 347
- Demand\_Setpoint (*rdflib.BRICK* attribute), 347
- DemoAlbum (*rdflib.SDO* attribute), 490
- Dentist (*rdflib.SDO* attribute), 490
- Dentistry (*rdflib.SDO* attribute), 490
- DepartAction (*rdflib.SDO* attribute), 490
- department (*rdflib.SDO* attribute), 546
- DepartmentStore (*rdflib.SDO* attribute), 490
- departureAirport (*rdflib.SDO* attribute), 546
- departureBoatTerminal (*rdflib.SDO* attribute), 546
- departureBusStop (*rdflib.SDO* attribute), 546
- departureGate (*rdflib.SDO* attribute), 546
- departurePlatform (*rdflib.SDO* attribute), 546
- departureStation (*rdflib.SDO* attribute), 546
- departureTerminal (*rdflib.SDO* attribute), 546
- departureTime (*rdflib.SDO* attribute), 546
- dependencies (*rdflib.SDO* attribute), 546
- depiction (*rdflib.FOAF* attribute), 421
- depicts (*rdflib.FOAF* attribute), 421
- deployedOnPlatform (*rdflib.SSN* attribute), 599
- deployedSystem (*rdflib.SSN* attribute), 599
- Deployment (*rdflib.SSN* attribute), 599
- DepositAccount (*rdflib.SDO* attribute), 490
- deprecated (*rdflib.OWL* attribute), 465
- DeprecatedClass (*rdflib.OWL* attribute), 463
- DeprecatedProperty (*rdflib.OWL* attribute), 463
- depth (*rdflib.SDO* attribute), 546
- Derivation (*rdflib.PROV* attribute), 469
- Derivative\_Gain\_Parameter (*rdflib.BRICK* attribute), 347
- Derivative\_Time\_Parameter (*rdflib.BRICK* attribute), 347
- derive (*rdflib.ODRL2* attribute), 456
- derivedByInsertionFrom (*rdflib.PROV* attribute), 470
- derivedByRemovalFrom (*rdflib.PROV* attribute), 470
- Dermatologic (*rdflib.SDO* attribute), 490
- Dermatology (*rdflib.SDO* attribute), 490
- Describer (*class in rdflib.extras.describer*), 49
- describes (*rdflib.CSVW* attribute), 401
- describesService (*rdflib.PROV* attribute), 470
- description (*rdflib.DC* attribute), 407
- description (*rdflib.DCTERMS* attribute), 412
- description (*rdflib.DOAP* attribute), 414
- Description (*rdflib.plugins.parsers.RDFVOC.RDFVOC* attribute), 79
- description (*rdflib.SDO* attribute), 547
- description (*rdflib.SH* attribute), 592
- destroy() (*rdflib.Graph* method), 431
- destroy() (*rdflib.graph.Graph* method), 258
- destroy() (*rdflib.graph.ReadOnlyGraphAggregate* method), 272
- destroy() (*rdflib.plugins.stores.auditable.AuditableStore* method), 190
- destroy() (*rdflib.plugins.stores.regexmatching.REGEXMatching* method), 202
- destroy() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 207
- destroy() (*rdflib.store.Store* method), 310
- detail (*rdflib.SH* attribute), 592
- detects (*rdflib.SSN* attribute), 599
- Detention\_Room (*rdflib.BRICK* attribute), 348
- developer (*rdflib.DOAP* attribute), 415
- device (*rdflib.ODRL2* attribute), 456
- device (*rdflib.SDO* attribute), 547
- Dew\_Point\_Setpoint (*rdflib.BRICK* attribute), 348
- Dewpoint\_Sensor (*rdflib.BRICK* attribute), 348
- DiabeticDiet (*rdflib.SDO* attribute), 490
- diagnosis (*rdflib.SDO* attribute), 547
- Diagnostic (*rdflib.SDO* attribute), 490
- DiagnosticLab (*rdflib.SDO* attribute), 490
- DiagnosticProcedure (*rdflib.SDO* attribute), 490
- diagram (*rdflib.SDO* attribute), 547
- Dialect (*rdflib.CSVW* attribute), 400
- dialect (*rdflib.CSVW* attribute), 401
- Dictionary (*rdflib.PROV* attribute), 469
- dictionary (*rdflib.PROV* attribute), 471
- Diet (*rdflib.SDO* attribute), 490
- diet (*rdflib.SDO* attribute), 547
- DietarySupplement (*rdflib.SDO* attribute), 490
- dietFeatures (*rdflib.SDO* attribute), 547
- DietNutrition (*rdflib.SDO* attribute), 490
- differentFrom (*rdflib.OWL* attribute), 465
- Differential\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 348
- Differential\_Pressure\_Bypass\_Valve (*rdflib.BRICK* attribute), 348
- Differential\_Pressure\_Deadband\_Setpoint (*rdflib.BRICK* attribute), 348
- Differential\_Pressure\_Integral\_Time\_Parameter (*rdflib.BRICK* attribute), 348
- Differential\_Pressure\_Load\_Shed\_Status (*rdflib.BRICK* attribute), 348
- Differential\_Pressure\_Proportional\_Band (*rdflib.BRICK* attribute), 348
- Differential\_Pressure\_Sensor (*rdflib.BRICK* attribute), 348
- Differential\_Pressure\_Setpoint (*rdflib.BRICK* attribute), 348
- Differential\_Pressure\_Setpoint\_Limit (*rdflib.BRICK* attribute), 348
- Differential\_Pressure\_Step\_Parameter (*rdflib.BRICK* attribute), 348
- Differential\_Speed\_Sensor (*rdflib.BRICK* attribute), 348
- Differential\_Speed\_Setpoint (*rdflib.BRICK* attribute), 348

Differential_Supply_Return_Water_Temperature_Sensor (rdflib.BRICK attribute), 348	Discharge_Air_Duct_Pressure_Status (rdflib.BRICK attribute), 349
differentialDiagnosis (rdflib.SDO attribute), 547	Discharge_Air_Flow_Demand_Setpoint (rdflib.BRICK attribute), 349
DigitalAudioTapeFormat (rdflib.SDO attribute), 490	Discharge_Air_Flow_High_Reset_Setpoint (rdflib.BRICK attribute), 349
DigitalDocument (rdflib.SDO attribute), 490	Discharge_Air_Flow_Low_Reset_Setpoint (rdflib.BRICK attribute), 349
DigitalDocumentPermission (rdflib.SDO attribute), 490	Discharge_Air_Flow_Reset_Setpoint (rdflib.BRICK attribute), 349
DigitalDocumentPermissionType (rdflib.SDO attribute), 490	Discharge_Air_Flow_Sensor (rdflib.BRICK attribute), 349
DigitalFormat (rdflib.SDO attribute), 490	Discharge_Air_Flow_Setpoint (rdflib.BRICK attribute), 350
digitize (rdflib.ODRL2 attribute), 456	Discharge_Air_Humidity_Sensor (rdflib.BRICK attribute), 350
dimension (rdflib.QB attribute), 476	Discharge_Air_Humidity_Setpoint (rdflib.BRICK attribute), 350
DimensionProperty (rdflib.QB attribute), 476	Discharge_Air_Smoke_Detection_Alarm (rdflib.BRICK attribute), 350
Dimmer (rdflib.BRICK attribute), 348	Discharge_Air_Static_Pressure_Deadband_Setpoint (rdflib.BRICK attribute), 350
Direct_Expansion_Cooling_Coil (rdflib.BRICK attribute), 348	Discharge_Air_Static_Pressure_Integral_Time_Parameter (rdflib.BRICK attribute), 350
Direct_Expansion_Heating_Coil (rdflib.BRICK attribute), 348	Discharge_Air_Static_Pressure_Proportional_Band_Parameter (rdflib.BRICK attribute), 350
directApply (rdflib.SDO attribute), 547	Discharge_Air_Static_Pressure_Sensor (rdflib.BRICK attribute), 350
Direction (rdflib.CSVW attribute), 400	Discharge_Air_Static_Pressure_Setpoint (rdflib.BRICK attribute), 350
direction (rdflib.RDF attribute), 478	Discharge_Air_Static_Pressure_Step_Parameter (rdflib.BRICK attribute), 350
Direction_Command (rdflib.BRICK attribute), 349	Discharge_Air_Temperature_Alarm (rdflib.BRICK attribute), 350
Direction_Sensor (rdflib.BRICK attribute), 349	Discharge_Air_Temperature_Cooling_Setpoint (rdflib.BRICK attribute), 350
Direction_Status (rdflib.BRICK attribute), 349	Discharge_Air_Temperature_Deadband_Setpoint (rdflib.BRICK attribute), 350
directive() (rdflib.plugins.parsers.notation3.SinkParser method), 89	Discharge_Air_Temperature_Heating_Setpoint (rdflib.BRICK attribute), 350
directiveOrStatement() (rdflib.plugins.parsers.notation3.SinkParser method), 89	Discharge_Air_Temperature_High_Reset_Setpoint (rdflib.BRICK attribute), 350
directiveOrStatement() (rdflib.plugins.parsers.trig.TrigSinkParser method), 109	Discharge_Air_Temperature_Low_Reset_Setpoint (rdflib.BRICK attribute), 350
director (rdflib.SDO attribute), 547	Discharge_Air_Temperature_Proportional_Band_Parameter (rdflib.BRICK attribute), 350
directors (rdflib.SDO attribute), 547	Discharge_Air_Temperature_Reset_Differential_Setpoint (rdflib.BRICK attribute), 350
DirectQueryService (rdflib.PROV attribute), 469	Discharge_Air_Temperature_Sensor (rdflib.BRICK attribute), 351
DisabilitySupport (rdflib.SDO attribute), 490	Discharge_Air_Temperature_Setpoint (rdflib.BRICK attribute), 351
Disable_Command (rdflib.BRICK attribute), 349	Discharge_Air_Temperature_Setpoint_Limit (rdflib.BRICK attribute), 351
Disable_Differential_Enthalpy_Command (rdflib.BRICK attribute), 349	
Disable_Differential_Temperature_Command (rdflib.BRICK attribute), 349	
Disable_Fixed_Enthalpy_Command (rdflib.BRICK attribute), 349	
Disable_Fixed_Temperature_Command (rdflib.BRICK attribute), 349	
Disable_Hot_Water_System_Outside_Air_Temperature_Setpoint (rdflib.BRICK attribute), 349	
Disable_Status (rdflib.BRICK attribute), 349	
DisagreeAction (rdflib.SDO attribute), 490	
disambiguatingDescription (rdflib.SDO attribute), 547	
Discharge_Air (rdflib.BRICK attribute), 349	
Discharge_Air_Dewpoint_Sensor (rdflib.BRICK attribute), 349	

- Discharge\_Air\_Temperature\_Step\_Parameter (*rdflib.BRICK* attribute), 351
- Discharge\_Air\_Velocity\_Pressure\_Sensor (*rdflib.BRICK* attribute), 351
- Discharge\_Chilled\_Water (*rdflib.BRICK* attribute), 351
- Discharge\_Fan (*rdflib.BRICK* attribute), 351
- Discharge\_Hot\_Water (*rdflib.BRICK* attribute), 351
- Discharge\_Water (*rdflib.BRICK* attribute), 351
- Discharge\_Water\_Differential\_Pressure\_Deadband\_Setpoint (*rdflib.BRICK* attribute), 351
- Discharge\_Water\_Differential\_Pressure\_Integral\_Time\_Parameter (*rdflib.BRICK* attribute), 351
- Discharge\_Water\_Differential\_Pressure\_Proportional\_Band\_Setpoint (*rdflib.BRICK* attribute), 351
- Discharge\_Water\_Flow\_Sensor (*rdflib.BRICK* attribute), 351
- Discharge\_Water\_Flow\_Setpoint (*rdflib.BRICK* attribute), 351
- Discharge\_Water\_Temperature\_Alarm (*rdflib.BRICK* attribute), 351
- Discharge\_Water\_Temperature\_Proportional\_Band\_Setpoint (*rdflib.BRICK* attribute), 351
- Discharge\_Water\_Temperature\_Sensor (*rdflib.BRICK* attribute), 351
- Discharge\_Water\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 352
- Disconnect\_Switch (*rdflib.BRICK* attribute), 352
- Discontinued (*rdflib.SDO* attribute), 491
- discount (*rdflib.SDO* attribute), 547
- discountCode (*rdflib.SDO* attribute), 547
- discountCurrency (*rdflib.SDO* attribute), 547
- DiscoverAction (*rdflib.SDO* attribute), 491
- discusses (*rdflib.SDO* attribute), 547
- DiscussionForumPosting (*rdflib.SDO* attribute), 491
- discussionUrl (*rdflib.SDO* attribute), 547
- diseasePreventionInfo (*rdflib.SDO* attribute), 547
- diseaseSpreadStatistics (*rdflib.SDO* attribute), 547
- disjoint (*rdflib.SH* attribute), 592
- DisjointConstraintComponent (*rdflib.SH* attribute), 588
- disjointDomain() (*rdflib.plugins.sparql.sparql.FrozenDict* method), 180
- disjointUnionOf (*rdflib.OWL* attribute), 465
- disjointWith (*rdflib.extras.infixowl.Class* property), 61
- disjointWith (*rdflib.OWL* attribute), 465
- DislikeAction (*rdflib.SDO* attribute), 491
- dispatch() (*rdflib.events.Dispatcher* method), 234
- Dispatcher (class in *rdflib.events*), 234
- Displacement\_Flow\_Air\_Diffuser (*rdflib.BRICK* attribute), 352
- display (*rdflib.ODRL2* attribute), 456
- dissolutionDate (*rdflib.SDO* attribute), 547
- Distance (*rdflib.SDO* attribute), 491
- distance (*rdflib.SDO* attribute), 547
- DistanceFee (*rdflib.SDO* attribute), 491
- Distillery (*rdflib.SDO* attribute), 491
- distinctMembers (*rdflib.OWL* attribute), 465
- distinctObjects (*rdflib.VOID* attribute), 607
- distinctSubjects (*rdflib.VOID* attribute), 607
- distinguishingSign (*rdflib.SDO* attribute), 547
- distribute (*rdflib.ODRL2* attribute), 456
- Distribution (*rdflib.DCAT* attribute), 408
- distribution (*rdflib.DCAT* attribute), 409
- Distribution\_Parameter (*rdflib.SDO* attribute), 547
- Distribution\_Frame (*rdflib.BRICK* attribute), 352
- diverse\_Sign\_Parameter (*rdflib.SDO* attribute), 547
- diversityStaffingReport (*rdflib.SDO* attribute), 547
- DJMixAlbum (*rdflib.SDO* attribute), 489
- dm (*rdflib.PROV* attribute), 471
- dnaChecksum (*rdflib.FOAF* attribute), 421
- DOAP (class in *rdflib*), 414
- DOAS (*rdflib.BRICK* attribute), 347
- Document (*rdflib.FOAF* attribute), 420
- ParameterElement\_start() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 104
- documentation (*rdflib.SDO* attribute), 547
- documenter (*rdflib.DOAP* attribute), 415
- documents (*rdflib.VOID* attribute), 607
- doesNotShip (*rdflib.SDO* attribute), 547
- doList() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 115
- doList() (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 122
- domain (*rdflib.extras.infixowl.Property* property), 67
- domain (*rdflib.RDFS* attribute), 479
- domainIncludes (*rdflib.SDO* attribute), 548
- Domestic\_Hot\_Water\_Supply\_Temperature\_Sensor (*rdflib.BRICK* attribute), 352
- Domestic\_Hot\_Water\_Supply\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 352
- Domestic\_Hot\_Water\_System (*rdflib.BRICK* attribute), 352
- Domestic\_Hot\_Water\_System\_Enable\_Command (*rdflib.BRICK* attribute), 352
- Domestic\_Hot\_Water\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 352
- Domestic\_Hot\_Water\_Valve (*rdflib.BRICK* attribute), 352
- Domestic\_Water (*rdflib.BRICK* attribute), 352
- Domestic\_Water\_Loop (*rdflib.BRICK* attribute), 352
- domiciledMortgage (*rdflib.SDO* attribute), 548
- DonateAction (*rdflib.SDO* attribute), 491
- dont\_care() (*rdflib.plugins.sparql.aggregates.Accumulator* method), 140
- doorTime (*rdflib.SDO* attribute), 548



dosageForm (*rdflib.SDO attribute*), 548  
 DoseSchedule (*rdflib.SDO attribute*), 491  
 doseSchedule (*rdflib.SDO attribute*), 548  
 doseUnit (*rdflib.SDO attribute*), 548  
 doseValue (*rdflib.SDO attribute*), 548  
 double (*rdflib.XSD attribute*), 610  
 DoubleBlindedTrial (*rdflib.SDO attribute*), 491  
 doubleQuote (*rdflib.CSVW attribute*), 401  
 DownloadAction (*rdflib.SDO attribute*), 491  
 downloadURL (*rdflib.DCAT attribute*), 409  
 downloadUrl (*rdflib.SDO attribute*), 548  
 Downpayment (*rdflib.SDO attribute*), 491  
 downPayment (*rdflib.SDO attribute*), 548  
 downvoteCount (*rdflib.SDO attribute*), 548  
 drainsTo (*rdflib.SDO attribute*), 548  
 DrawAction (*rdflib.SDO attribute*), 491  
 Drawing (*rdflib.SDO attribute*), 491  
 Drench\_Hose (*rdflib.BRICK attribute*), 352  
 DrinkAction (*rdflib.SDO attribute*), 491  
 Drive\_Ready\_Status (*rdflib.BRICK attribute*), 352  
 driveWheelConfiguration (*rdflib.SDO attribute*), 548  
 DriveWheelConfigurationValue (*rdflib.SDO attribute*), 491  
 DrivingSchoolVehicleUsage (*rdflib.SDO attribute*), 491  
 dropoffLocation (*rdflib.SDO attribute*), 548  
 dropoffTime (*rdflib.SDO attribute*), 548  
 Drug (*rdflib.SDO attribute*), 491  
 drug (*rdflib.SDO attribute*), 548  
 DrugClass (*rdflib.SDO attribute*), 491  
 drugClass (*rdflib.SDO attribute*), 548  
 DrugCost (*rdflib.SDO attribute*), 491  
 DrugCostCategory (*rdflib.SDO attribute*), 491  
 DrugLegalStatus (*rdflib.SDO attribute*), 491  
 DrugPregnancyCategory (*rdflib.SDO attribute*), 491  
 DrugPrescriptionStatus (*rdflib.SDO attribute*), 491  
 DrugStrength (*rdflib.SDO attribute*), 491  
 drugUnit (*rdflib.SDO attribute*), 548  
 DryCleaningOrLaundry (*rdflib.SDO attribute*), 491  
 DummySink (*class in rdflib.plugins.parsers.ntriples*), 98  
 dumps() (*rdflib.store.NodePickler method*), 307  
 duns (*rdflib.SDO attribute*), 548  
 duplicateTherapy (*rdflib.SDO attribute*), 548  
 Duration (*rdflib.SDO attribute*), 492  
 duration (*rdflib.SDO attribute*), 548  
 Duration (*rdflib.TIME attribute*), 600  
 duration (*rdflib.XSD attribute*), 610  
 Duration\_Sensor (*rdflib.BRICK attribute*), 352  
 DurationDescription (*rdflib.TIME attribute*), 600  
 durationOfWarranty (*rdflib.SDO attribute*), 548  
 duringMedia (*rdflib.SDO attribute*), 548  
 Duty (*rdflib.ODRL2 attribute*), 453  
 duty (*rdflib.ODRL2 attribute*), 456  
 DVDFormat (*rdflib.SDO attribute*), 489

## E

Ear (*rdflib.SDO attribute*), 492  
 earlyPrepaymentPenalty (*rdflib.SDO attribute*), 548  
 eat() (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser method*), 100  
 EatAction (*rdflib.SDO attribute*), 492  
 EBook (*rdflib.SDO attribute*), 492  
 EBV() (*in module rdflib.plugins.sparql.operators*), 164  
 EconCycle\_Start\_Stop\_Status (*rdflib.BRICK attribute*), 352  
 Economizer (*rdflib.BRICK attribute*), 352  
 Economizer\_Damper (*rdflib.BRICK attribute*), 353  
 EditedOrCroppedContent (*rdflib.SDO attribute*), 492  
 editEIDR (*rdflib.SDO attribute*), 548  
 editor (*rdflib.SDO attribute*), 548  
 editorialNote (*rdflib.PROV attribute*), 471  
 editorialNote (*rdflib.SKOS attribute*), 596  
 editorsDefinition (*rdflib.PROV attribute*), 471  
 educationalAlignment (*rdflib.SDO attribute*), 549  
 EducationalAudience (*rdflib.SDO attribute*), 492  
 educationalCredentialAwarded (*rdflib.SDO attribute*), 549  
 educationalFramework (*rdflib.SDO attribute*), 549  
 educationalLevel (*rdflib.SDO attribute*), 549  
 EducationalOccupationalCredential (*rdflib.SDO attribute*), 492  
 EducationalOccupationalProgram (*rdflib.SDO attribute*), 492  
 EducationalOrganization (*rdflib.SDO attribute*), 493  
 educationalProgramMode (*rdflib.SDO attribute*), 549  
 educationalRole (*rdflib.SDO attribute*), 549  
 educationalUse (*rdflib.SDO attribute*), 549  
 EducationEvent (*rdflib.SDO attribute*), 492  
 educationLevel (*rdflib.DCTERMS attribute*), 412  
 educationRequirements (*rdflib.SDO attribute*), 548  
 eduQuestionType (*rdflib.SDO attribute*), 548  
 Effective\_Air\_Temperature\_Cooling\_Setpoint (*rdflib.BRICK attribute*), 353  
 Effective\_Air\_Temperature\_Heating\_Setpoint (*rdflib.BRICK attribute*), 353  
 Effective\_Air\_Temperature\_Setpoint (*rdflib.BRICK attribute*), 353  
 Effective\_Discharge\_Air\_Temperature\_Setpoint (*rdflib.BRICK attribute*), 353  
 Effective\_Return\_Air\_Temperature\_Setpoint (*rdflib.BRICK attribute*), 353  
 Effective\_Room\_Air\_Temperature\_Setpoint (*rdflib.BRICK attribute*), 353  
 Effective\_Supply\_Air\_Temperature\_Setpoint (*rdflib.BRICK attribute*), 353  
 Effective\_Zone\_Air\_Temperature\_Setpoint (*rdflib.BRICK attribute*), 353  
 EffectivenessHealthAspect (*rdflib.SDO attribute*), 493

- `elapsedTime` (*rdflib.ODRL2 attribute*), 456
- `Electric_Baseboard_Radiator` (*rdflib.BRICK attribute*), 353
- `Electric_Boiler` (*rdflib.BRICK attribute*), 353
- `Electric_Radiator` (*rdflib.BRICK attribute*), 353
- `Electrical_Equipment` (*rdflib.BRICK attribute*), 353
- `Electrical_Meter` (*rdflib.BRICK attribute*), 353
- `Electrical_Power_Sensor` (*rdflib.BRICK attribute*), 353
- `Electrical_Room` (*rdflib.BRICK attribute*), 353
- `Electrical_System` (*rdflib.BRICK attribute*), 353
- `electricalPhaseCount` (*rdflib.BRICK attribute*), 398
- `electricalPhases` (*rdflib.BRICK attribute*), 398
- `Electrician` (*rdflib.SDO attribute*), 493
- `ElectronicsStore` (*rdflib.SDO attribute*), 493
- `element()` (*rdflib.plugins.serializers.xmlwriter.XMLWriter method*), 123
- `ElementarySchool` (*rdflib.SDO attribute*), 493
- `ElementHandler` (*class in rdflib.plugins.parsers.rdfxml*), 102
- `elevation` (*rdflib.SDO attribute*), 549
- `Elevator` (*rdflib.BRICK attribute*), 353
- `Elevator_Shaft` (*rdflib.BRICK attribute*), 353
- `Elevator_Space` (*rdflib.BRICK attribute*), 354
- `eligibilityToWorkRequirement` (*rdflib.SDO attribute*), 549
- `eligibleCustomerType` (*rdflib.SDO attribute*), 549
- `eligibleDuration` (*rdflib.SDO attribute*), 549
- `eligibleQuantity` (*rdflib.SDO attribute*), 549
- `eligibleRegion` (*rdflib.SDO attribute*), 549
- `eligibleTransactionVolume` (*rdflib.SDO attribute*), 549
- `email` (*rdflib.SDO attribute*), 549
- `EmailMessage` (*rdflib.SDO attribute*), 493
- `Embassy` (*rdflib.SDO attribute*), 493
- `Embedded_Surface_System_Panel` (*rdflib.BRICK attribute*), 354
- `Embedded_Temperature_Sensor` (*rdflib.BRICK attribute*), 354
- `Embedded_Temperature_Setpoint` (*rdflib.BRICK attribute*), 354
- `embeddedTextCaption` (*rdflib.SDO attribute*), 549
- `embedUrl` (*rdflib.SDO attribute*), 549
- `Emergency` (*rdflib.SDO attribute*), 493
- `Emergency_Air_Flow_System` (*rdflib.BRICK attribute*), 354
- `Emergency_Air_Flow_System_Status` (*rdflib.BRICK attribute*), 354
- `Emergency_Alarm` (*rdflib.BRICK attribute*), 354
- `Emergency_Generator_Alarm` (*rdflib.BRICK attribute*), 354
- `Emergency_Generator_Status` (*rdflib.BRICK attribute*), 354
- `Emergency_Phone` (*rdflib.BRICK attribute*), 354
- `Emergency_Power_Off_System` (*rdflib.BRICK attribute*), 354
- `Emergency_Power_Off_System_Activated_By_High_Temperature_S` (*rdflib.BRICK attribute*), 354
- `Emergency_Power_Off_System_Activated_By_Leak_Detection_Sys` (*rdflib.BRICK attribute*), 354
- `Emergency_Power_Off_System_Status` (*rdflib.BRICK attribute*), 354
- `Emergency_Push_Button_Status` (*rdflib.BRICK attribute*), 354
- `Emergency_Wash_Station` (*rdflib.BRICK attribute*), 354
- `EmergencyService` (*rdflib.SDO attribute*), 493
- `emissionsCO2` (*rdflib.SDO attribute*), 549
- `employee` (*rdflib.SDO attribute*), 549
- `Employee_Entrance_Lobby` (*rdflib.BRICK attribute*), 354
- `EmployeeRole` (*rdflib.SDO attribute*), 493
- `employees` (*rdflib.SDO attribute*), 549
- `EmployerAggregateRating` (*rdflib.SDO attribute*), 493
- `employerOverview` (*rdflib.SDO attribute*), 549
- `EmployerReview` (*rdflib.SDO attribute*), 493
- `EmploymentAgency` (*rdflib.SDO attribute*), 493
- `employmentType` (*rdflib.SDO attribute*), 549
- `employmentUnit` (*rdflib.SDO attribute*), 549
- `EmptyCollection` (*rdflib.PROV attribute*), 469
- `EmptyDictionary` (*rdflib.PROV attribute*), 469
- `Enable_Command` (*rdflib.BRICK attribute*), 354
- `Enable_Differential_Enthalpy_Command` (*rdflib.BRICK attribute*), 355
- `Enable_Differential_Temperature_Command` (*rdflib.BRICK attribute*), 355
- `Enable_Fixed_Enthalpy_Command` (*rdflib.BRICK attribute*), 355
- `Enable_Fixed_Temperature_Command` (*rdflib.BRICK attribute*), 355
- `Enable_Hot_Water_System_Outside_Air_Temperature_Setpoint` (*rdflib.BRICK attribute*), 355
- `Enable_Status` (*rdflib.BRICK attribute*), 355
- `Enclosed_Office` (*rdflib.BRICK attribute*), 355
- `EncodeOnlyUnicode` (*class in rdflib.query*), 291
- `encodesBioChemEntity` (*rdflib.SDO attribute*), 550
- `encodesCreativeWork` (*rdflib.SDO attribute*), 550
- `encoding` (*rdflib.CSVW attribute*), 401
- `encoding` (*rdflib.plugins.serializers.hex.HextuplesSerializer attribute*), 114
- `encoding` (*rdflib.plugins.serializers.jsonld.JsonLDSerializer attribute*), 115
- `encoding` (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer attribute*), 116
- `encoding` (*rdflib.plugins.serializers.n3.N3Serializer attribute*), 117
- `encoding` (*rdflib.plugins.serializers.nquads.NQuadsSerializer attribute*), 117

encoding (rdflib.plugins.serializers.nt.NTSerializer attribute), 118  
 encoding (rdflib.plugins.serializers.rdfxml.PrettyXMLSerializer attribute), 118  
 encoding (rdflib.plugins.serializers.rdfxml.XMLSerializer attribute), 119  
 encoding (rdflib.plugins.serializers.trig.TrigSerializer attribute), 120  
 encoding (rdflib.plugins.serializers.trix.TriXSerializer attribute), 121  
 encoding (rdflib.plugins.serializers.turtle.RecursiveSerializer attribute), 121  
 encoding (rdflib.plugins.serializers.turtle.TurtleSerializer attribute), 122  
 encoding (rdflib.SDO attribute), 550  
 encodingFormat (rdflib.SDO attribute), 550  
 encodings (rdflib.SDO attribute), 550  
 encodingType (rdflib.SDO attribute), 550  
 end (rdflib.plugins.parsers.rdfxml.ElementHandler attribute), 102  
 End (rdflib.PROV attribute), 469  
 end() (rdflib.container.Container method), 233  
 endDate (rdflib.DCAT attribute), 409  
 endDate (rdflib.SDO attribute), 550  
 endDoc() (rdflib.plugins.parsers.notation3.RDFSink method), 84  
 endDoc() (rdflib.plugins.parsers.notation3.SinkParser method), 89  
 endDocument() (rdflib.plugins.serializers.longturtle.LongTurtleSerializer method), 116  
 endDocument() (rdflib.plugins.serializers.n3.N3Serializer method), 117  
 endDocument() (rdflib.plugins.serializers.turtle.TurtleSerializer method), 122  
 ended (rdflib.PROV attribute), 471  
 endedAtTime (rdflib.PROV attribute), 471  
 endElementNS() (rdflib.plugins.parsers.rdfxml.RDFXMLHandler method), 104  
 endElementNS() (rdflib.plugins.parsers.trix.TriXHandler method), 110  
 Endocrine (rdflib.SDO attribute), 493  
 endOffset (rdflib.SDO attribute), 550  
 EndorseAction (rdflib.SDO attribute), 493  
 endorsee (rdflib.SDO attribute), 550  
 EndorsementRating (rdflib.SDO attribute), 493  
 endorsers (rdflib.SDO attribute), 550  
 endpointDescription (rdflib.DCAT attribute), 409  
 endpointURL (rdflib.DCAT attribute), 409  
 endPrefixMapping() (rdflib.plugins.parsers.rdfxml.RDFXMLHandler method), 104  
 endPrefixMapping() (rdflib.plugins.parsers.trix.TriXHandler method), 110  
 endTime (rdflib.SDO attribute), 550  
 Energy (rdflib.SDO attribute), 493  
 Energy\_Generation\_System (rdflib.BRICK attribute), 355  
 Energy\_Sensor (rdflib.BRICK attribute), 355  
 Energy\_Storage (rdflib.BRICK attribute), 355  
 Energy\_Storage\_System (rdflib.BRICK attribute), 355  
 Energy\_System (rdflib.BRICK attribute), 355  
 Energy\_Usage\_Sensor (rdflib.BRICK attribute), 355  
 Energy\_Zone (rdflib.BRICK attribute), 355  
 EnergyConsumptionDetails (rdflib.SDO attribute), 493  
 EnergyEfficiencyEnumeration (rdflib.SDO attribute), 493  
 energyEfficiencyScaleMax (rdflib.SDO attribute), 550  
 energyEfficiencyScaleMin (rdflib.SDO attribute), 550  
 EnergyStarCertified (rdflib.SDO attribute), 493  
 EnergyStarEnergyEfficiencyEnumeration (rdflib.SDO attribute), 493  
 engineDisplacement (rdflib.SDO attribute), 550  
 enginePower (rdflib.SDO attribute), 550  
 EngineSpecification (rdflib.SDO attribute), 493  
 engineType (rdflib.SDO attribute), 550  
 EnrollingByInvitation (rdflib.SDO attribute), 494  
 ensureExclusivity (rdflib.ODRL2 attribute), 456  
 entailment (rdflib.SH attribute), 592  
 Entering\_Water (rdflib.BRICK attribute), 355  
 Entering\_Water\_Flow\_Sensor (rdflib.BRICK attribute), 355  
 Entering\_Water\_Flow\_Setpoint (rdflib.BRICK attribute), 355  
 Entering\_Water\_Temperature\_Sensor (rdflib.BRICK attribute), 355  
 Entering\_Water\_Temperature\_Setpoint (rdflib.BRICK attribute), 356  
 EntertainmentBusiness (rdflib.SDO attribute), 494  
 entertainmentBusiness (rdflib.SDO attribute), 550  
 Enthalpy\_Sensor (rdflib.BRICK attribute), 356  
 Enthalpy\_Setpoint (rdflib.BRICK attribute), 356  
 entities (rdflib.VOID attribute), 607  
 ENTITIES (rdflib.XSD attribute), 609  
 Entity (rdflib.PROV attribute), 469  
 entity (rdflib.PROV attribute), 471  
 ENTITY (rdflib.XSD attribute), 609  
 EntityInfluence (rdflib.PROV attribute), 469  
 entityOfInfluence (rdflib.PROV attribute), 471  
 Entrance (rdflib.BRICK attribute), 356  
 EntryPoint (rdflib.SDO attribute), 494  
 EnumeratedClass (class in rdflib.extras.infixowl), 63  
 Enumeration (rdflib.SDO attribute), 494  
 enumeration (rdflib.XSD attribute), 610  
 Environment\_Box (rdflib.BRICK attribute), 356



- epidemiology (*rdflib.SDO* attribute), 550
- Episode (*rdflib.SDO* attribute), 494
- episode (*rdflib.SDO* attribute), 550
- episodeNumber (*rdflib.SDO* attribute), 550
- episodes (*rdflib.SDO* attribute), 550
- EPRelease (*rdflib.SDO* attribute), 492
- eq (*rdflib.ODRL2* attribute), 456
- eq() (*rdflib.Literal* method), 449
- eq() (*rdflib.term.Identifier* method), 316
- eq() (*rdflib.term.Literal* method), 324
- equal (*rdflib.SDO* attribute), 550
- equals (*rdflib.SH* attribute), 592
- EqualsConstraintComponent (*rdflib.SH* attribute), 588
- Equipment (*rdflib.BRICK* attribute), 356
- Equipment\_Room (*rdflib.BRICK* attribute), 356
- equivalentClass (*rdflib.extras.infixowl.Class* property), 61
- equivalentClass (*rdflib.OWL* attribute), 465
- equivalentProperty (*rdflib.OWL* attribute), 465
- Error, 235
- error (*rdflib.SDO* attribute), 550
- error() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 104
- error() (*rdflib.plugins.parsers.trix.TriXHandler* method), 111
- ESCAPED (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* attribute), 212
- ESS\_Panel (*rdflib.BRICK* attribute), 352
- estimatedCost (*rdflib.SDO* attribute), 550
- estimatedFlightDuration (*rdflib.SDO* attribute), 550
- estimatedSalary (*rdflib.SDO* attribute), 550
- estimatesRiskOf (*rdflib.SDO* attribute), 551
- ethicsPolicy (*rdflib.SDO* attribute), 551
- EUEnergyEfficiencyCategoryA (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyCategoryA1Plus (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyCategoryA2Plus (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyCategoryA3Plus (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyCategoryB (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyCategoryC (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyCategoryD (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyCategoryE (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyCategoryF (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyCategoryG (*rdflib.SDO* attribute), 492
- EUEnergyEfficiencyEnumeration (*rdflib.SDO* attribute), 492
- eval() (*rdflib.paths.AlternativePath* method), 282
- eval() (*rdflib.paths.InvPath* method), 283
- eval() (*rdflib.paths.MulPath* method), 284
- eval() (*rdflib.paths.NegatedPath* method), 284
- eval() (*rdflib.paths.Path* method), 286
- eval() (*rdflib.paths.SequencePath* method), 287
- eval() (*rdflib.plugins.sparql.parserutils.Expr* method), 171
- eval\_full\_row() (*rdflib.plugins.sparql.aggregates.Counter* method), 142
- eval\_path() (in module *rdflib.paths*), 287
- eval\_row() (*rdflib.plugins.sparql.aggregates.Counter* method), 142
- evalAdd() (in module *rdflib.plugins.sparql.update*), 185
- evalAggregateJoin() (in module *rdflib.plugins.sparql.evaluate*), 152
- evalAskQuery() (in module *rdflib.plugins.sparql.evaluate*), 152
- evalBGP() (in module *rdflib.plugins.sparql.evaluate*), 152
- evalClear() (in module *rdflib.plugins.sparql.update*), 185
- evalConstructQuery() (in module *rdflib.plugins.sparql.evaluate*), 152
- evalCopy() (in module *rdflib.plugins.sparql.update*), 185
- evalCreate() (in module *rdflib.plugins.sparql.update*), 186
- evalDeleteData() (in module *rdflib.plugins.sparql.update*), 186
- evalDeleteWhere() (in module *rdflib.plugins.sparql.update*), 186
- evalDescribeQuery() (in module *rdflib.plugins.sparql.evaluate*), 152
- evalDistinct() (in module *rdflib.plugins.sparql.evaluate*), 153
- evalDrop() (in module *rdflib.plugins.sparql.update*), 186
- evalExtend() (in module *rdflib.plugins.sparql.evaluate*), 153
- evalFilter() (in module *rdflib.plugins.sparql.evaluate*), 153
- evalGraph() (in module *rdflib.plugins.sparql.evaluate*), 153
- evalGroup() (in module *rdflib.plugins.sparql.evaluate*), 153
- evalInsertData() (in module *rdflib.plugins.sparql.update*), 186
- evalJoin() (in module *rdflib.plugins.sparql.evaluate*), 153
- evalLazyJoin() (in module *rd-*



- [flib.plugins.sparql.evaluate](#)), 154
- [evalLeftJoin\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 154
- [evalLoad\(\)](#) (in module [rdflib.plugins.sparql.update](#)), 187
- [evalMinus\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 154
- [evalModify\(\)](#) (in module [rdflib.plugins.sparql.update](#)), 187
- [evalMove\(\)](#) (in module [rdflib.plugins.sparql.update](#)), 187
- [evalMultiset\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 154
- [evalOrderBy\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 154
- [evalPart\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 154
- [evalPath\(\)](#) (in module [rdflib.paths](#)), 287
- [evalProject\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 155
- [evalQuery\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 155
- [evalReduced\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 155
- [evalSelectQuery\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 155
- [evalServiceQuery\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 155
- [evalSlice\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 156
- [evalUnion\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 156
- [evalUpdate\(\)](#) (in module [rdflib.plugins.sparql.update](#)), 187
- [evalValues\(\)](#) (in module [rdflib.plugins.sparql.evaluate](#)), 156
- [Evaporative\\_Heat\\_Exchanger](#) ([rdflib.BRICK](#) attribute), 356
- [Even\\_Month\\_Status](#) ([rdflib.BRICK](#) attribute), 356
- [Event](#) (class in [rdflib.events](#)), 234
- [Event](#) ([rdflib.DCMITYPE](#) attribute), 409
- [event](#) ([rdflib.ODRL2](#) attribute), 456
- [Event](#) ([rdflib.SDO](#) attribute), 494
- [event](#) ([rdflib.SDO](#) attribute), 551
- [eventAttendanceMode](#) ([rdflib.SDO](#) attribute), 551
- [EventAttendanceModeEnumeration](#) ([rdflib.SDO](#) attribute), 494
- [EventCancelled](#) ([rdflib.SDO](#) attribute), 494
- [EventMovedOnline](#) ([rdflib.SDO](#) attribute), 494
- [EventPostponed](#) ([rdflib.SDO](#) attribute), 494
- [EventRescheduled](#) ([rdflib.SDO](#) attribute), 494
- [EventReservation](#) ([rdflib.SDO](#) attribute), 494
- [events](#) ([rdflib.SDO](#) attribute), 551
- [eventSchedule](#) ([rdflib.SDO](#) attribute), 551
- [EventScheduled](#) ([rdflib.SDO](#) attribute), 494
- [EventSeries](#) ([rdflib.SDO](#) attribute), 494
- [eventStatus](#) ([rdflib.SDO](#) attribute), 551
- [EventStatusType](#) ([rdflib.SDO](#) attribute), 494
- [EventVenue](#) ([rdflib.SDO](#) attribute), 494
- [evidenceLevel](#) ([rdflib.SDO](#) attribute), 551
- [EvidenceLevelA](#) ([rdflib.SDO](#) attribute), 494
- [EvidenceLevelB](#) ([rdflib.SDO](#) attribute), 494
- [EvidenceLevelC](#) ([rdflib.SDO](#) attribute), 494
- [evidenceOrigin](#) ([rdflib.SDO](#) attribute), 551
- [exactMatch](#) ([rdflib.SKOS](#) attribute), 596
- [example](#) ([rdflib.SKOS](#) attribute), 596
- [example](#) ([rdflib.VANN](#) attribute), 606
- [example\\_1\(\)](#) (in module [examples.berkeleydb\\_example](#)), 23
- [example\\_2\(\)](#) (in module [examples.berkeleydb\\_example](#)), 23
- [exampleOfWork](#) ([rdflib.SDO](#) attribute), 551
- [exampleResource](#) ([rdflib.VOID](#) attribute), 607
- [examples.berkeleydb\\_example](#) module, 23
- [examples.conjunctive\\_graphs](#) module, 22
- [examples.custom\\_datatype](#) module, 22
- [examples.custom\\_eval](#) module, 22
- [examples.foafpaths](#) module, 22
- [examples.prepared\\_query](#) module, 23
- [examples.resource\\_example](#) module, 23
- [examples.secure\\_with\\_audit](#) module, 25
- [examples.secure\\_with\\_urlopen](#) module, 26
- [examples.slice](#) module, 24
- [examples.smushing](#) module, 24
- [examples.sparql\\_query\\_example](#) module, 24
- [examples.sparql\\_update\\_example](#) module, 24
- [examples.sparqlstore\\_example](#) module, 24
- [examples.swap\\_primer](#) module, 25
- [examples.transitive](#) module, 25
- [exceptDate](#) ([rdflib.SDO](#) attribute), 551
- [ExchangeRateSpecification](#) ([rdflib.SDO](#) attribute), 494

- exchangeRateSpread (*rdflib.SDO* attribute), 551
  - ExchangeRefund (*rdflib.SDO* attribute), 494
  - executableLibraryName (*rdflib.SDO* attribute), 551
  - execute (*rdflib.ODRL2* attribute), 456
  - Exercise\_Room (*rdflib.BRICK* attribute), 356
  - ExerciseAction (*rdflib.SDO* attribute), 494
  - exerciseCourse (*rdflib.SDO* attribute), 551
  - ExerciseGym (*rdflib.SDO* attribute), 494
  - ExercisePlan (*rdflib.SDO* attribute), 494
  - exercisePlan (*rdflib.SDO* attribute), 551
  - exerciseRelatedDiet (*rdflib.SDO* attribute), 551
  - exerciseType (*rdflib.SDO* attribute), 551
  - Exhaust\_Air (*rdflib.BRICK* attribute), 356
  - Exhaust\_Air\_Dewpoint\_Sensor (*rdflib.BRICK* attribute), 356
  - Exhaust\_Air\_Differential\_Pressure\_Sensor (*rdflib.BRICK* attribute), 356
  - Exhaust\_Air\_Differential\_Pressure\_Setpoint (*rdflib.BRICK* attribute), 356
  - Exhaust\_Air\_Flow\_Integral\_Time\_Parameter (*rdflib.BRICK* attribute), 356
  - Exhaust\_Air\_Flow\_Proportional\_Band\_Parameter (*rdflib.BRICK* attribute), 356
  - Exhaust\_Air\_Flow\_Sensor (*rdflib.BRICK* attribute), 356
  - Exhaust\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 356
  - Exhaust\_Air\_Humidity\_Sensor (*rdflib.BRICK* attribute), 356
  - Exhaust\_Air\_Humidity\_Setpoint (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Stack\_Flow\_Deadband\_Setpoint (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Stack\_Flow\_Integral\_Time\_Parameter (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Stack\_Flow\_Proportional\_Band\_Parameter (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Stack\_Flow\_Sensor (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Stack\_Flow\_Setpoint (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Static\_Pressure\_Proportional\_Band\_Parameter (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Static\_Pressure\_Sensor (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Static\_Pressure\_Setpoint (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Temperature\_Sensor (*rdflib.BRICK* attribute), 357
  - Exhaust\_Air\_Velocity\_Pressure\_Sensor (*rdflib.BRICK* attribute), 357
  - Exhaust\_Damper (*rdflib.BRICK* attribute), 357
  - Exhaust\_Fan (*rdflib.BRICK* attribute), 357
  - Exhaust\_Fan\_Disable\_Command (*rdflib.BRICK* attribute), 357
  - Exhaust\_Fan\_Enable\_Command (*rdflib.BRICK* attribute), 357
  - ExhibitionEvent (*rdflib.SDO* attribute), 494
  - exifData (*rdflib.SDO* attribute), 551
  - expand() (*rdflib.plugins.shared.jsonld.context.Context* method), 126
  - expand\_curie() (*rdflib.namespace.NamespaceManager* method), 77
  - expandBNodeTriples() (in module *rdflib.plugins.sparql.parser*), 168
  - expandCollection() (in module *rdflib.plugins.sparql.parser*), 168
  - expandTriples() (in module *rdflib.plugins.sparql.parser*), 168
  - expandUnicodeEscapes() (in module *rdflib.plugins.sparql.parser*), 168
  - expectedArrivalFrom (*rdflib.SDO* attribute), 551
  - expectedArrivalUntil (*rdflib.SDO* attribute), 551
  - expectedPrognosis (*rdflib.SDO* attribute), 551
  - expectsAcceptanceOf (*rdflib.SDO* attribute), 551
  - experienceInPlaceOfEducation (*rdflib.SDO* attribute), 551
  - experienceRequirements (*rdflib.SDO* attribute), 551
  - expertConsiderations (*rdflib.SDO* attribute), 551
  - expires (*rdflib.SDO* attribute), 552
  - explicitTimezone (*rdflib.XSD* attribute), 610
  - export (*rdflib.ODRL2* attribute), 456
  - Expr (class in *rdflib.plugins.sparql.parserutils*), 171
  - expressedIn (*rdflib.SDO* attribute), 552
  - expression (*rdflib.SH* attribute), 592
  - ExpressionConstraintComponent (*rdflib.SH* attribute), 588
  - ExpressionNotCoveredException, 145
  - Extend() (in module *rdflib.plugins.sparql.algebra*), 146
  - extent (*rdflib.DCTERMS* attribute), 412
  - extent (*rdflib.extras.infixowl.Class* property), 61
  - extent (*rdflib.extras.infixowl.Property* property), 67
  - extentQuery (*rdflib.extras.infixowl.Class* property), 61
  - extract (*rdflib.ODRL2* attribute), 456
  - extractChar (*rdflib.ODRL2* attribute), 456
  - extractPage (*rdflib.ODRL2* attribute), 456
  - extractWord (*rdflib.ODRL2* attribute), 456
  - Extremum (class in *rdflib.plugins.sparql.aggregates*), 142
  - Eye (*rdflib.SDO* attribute), 494
  - Eye\_Wash\_Station (*rdflib.BRICK* attribute), 357
- ## F
- factoryGraph (*rdflib.extras.infixowl.Individual* attribute), 65
  - failAction (*rdflib.plugins.sparql.parserutils.ParamList* attribute), 172
  - FailedActionStatus (*rdflib.SDO* attribute), 495
  - failure (*rdflib.ODRL2* attribute), 456

- Failure\_Alarm (*rdflib.BRICK* attribute), 357  
 family\_name (*rdflib.FOAF* attribute), 421  
 familyName (*rdflib.FOAF* attribute), 421  
 familyName (*rdflib.SDO* attribute), 552  
 Fan (*rdflib.BRICK* attribute), 358  
 Fan\_Coil\_Unit (*rdflib.BRICK* attribute), 358  
 Fan\_On\_Off\_Status (*rdflib.BRICK* attribute), 358  
 Fan\_Status (*rdflib.BRICK* attribute), 358  
 Fan\_VFD (*rdflib.BRICK* attribute), 358  
 FAQPage (*rdflib.SDO* attribute), 495  
 FastFoodRestaurant (*rdflib.SDO* attribute), 495  
 fatContent (*rdflib.SDO* attribute), 552  
 Fault\_Reset\_Command (*rdflib.BRICK* attribute), 358  
 Fault\_Status (*rdflib.BRICK* attribute), 358  
 faxNumber (*rdflib.SDO* attribute), 552  
 FCU (*rdflib.BRICK* attribute), 357  
 FDACategoryA (*rdflib.SDO* attribute), 495  
 FDACategoryB (*rdflib.SDO* attribute), 495  
 FDACategoryC (*rdflib.SDO* attribute), 495  
 FDACategoryD (*rdflib.SDO* attribute), 495  
 FDACategoryX (*rdflib.SDO* attribute), 495  
 FDAnotEvaluated (*rdflib.SDO* attribute), 495  
 feature (*rdflib.VOID* attribute), 607  
 featureList (*rdflib.SDO* attribute), 552  
 FeatureOfInterest (*rdflib.SOSA* attribute), 597  
 feed() (*rdflib.plugins.parsers.notation3.SinkParser* method), 89  
 feeds (*rdflib.BRICK* attribute), 398  
 feedsAir (*rdflib.BRICK* attribute), 398  
 feesAndCommissionsSpecification (*rdflib.SDO* attribute), 552  
 Female (*rdflib.SDO* attribute), 495  
 Festival (*rdflib.SDO* attribute), 495  
 fiberContent (*rdflib.SDO* attribute), 552  
 Field\_Of\_Play (*rdflib.BRICK* attribute), 358  
 file (*rdflib.plugins.parsers.nquads.NQuadsParser* attribute), 98  
 file (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* attribute), 100  
 FileFormat (*rdflib.DCTERMS* attribute), 410  
 fileFormat (*rdflib.ODRL2* attribute), 457  
 fileFormat (*rdflib.SDO* attribute), 552  
 FileInputSource (class in *rdflib.parser*), 276  
 fileSize (*rdflib.SDO* attribute), 552  
 FilmAction (*rdflib.SDO* attribute), 495  
 Filter (*rdflib.BRICK* attribute), 358  
 Filter() (in module *rdflib.plugins.sparql.algebra*), 146  
 Filter\_Differential\_Pressure\_Sensor (*rdflib.BRICK* attribute), 358  
 Filter\_Reset\_Command (*rdflib.BRICK* attribute), 358  
 Filter\_Status (*rdflib.BRICK* attribute), 358  
 filterShape (*rdflib.SH* attribute), 592  
 Final\_Filter (*rdflib.BRICK* attribute), 358  
 financialAidEligible (*rdflib.SDO* attribute), 552  
 FinancialProduct (*rdflib.SDO* attribute), 495  
 FinancialService (*rdflib.SDO* attribute), 495  
 find\_roots() (in module *rdflib.util*), 330  
 find\_term() (*rdflib.plugins.shared.jsonld.context.Context* method), 126  
 FindAction (*rdflib.SDO* attribute), 495  
 Fire\_Control\_Panel (*rdflib.BRICK* attribute), 358  
 Fire\_Safety\_Equipment (*rdflib.BRICK* attribute), 358  
 Fire\_Safety\_System (*rdflib.BRICK* attribute), 358  
 Fire\_Sensor (*rdflib.BRICK* attribute), 358  
 Fire\_Zone (*rdflib.BRICK* attribute), 358  
 FireStation (*rdflib.SDO* attribute), 495  
 first (*rdflib.RDF* attribute), 478  
 first() (in module *rdflib.util*), 330  
 First\_Aid\_Kit (*rdflib.BRICK* attribute), 358  
 First\_Aid\_Room (*rdflib.BRICK* attribute), 358  
 firstAppearance (*rdflib.SDO* attribute), 552  
 firstName (*rdflib.FOAF* attribute), 421  
 firstPerformance (*rdflib.SDO* attribute), 552  
 fix() (in module *rdflib.plugins.serializers.rdfxml*), 119  
 flags (*rdflib.SH* attribute), 592  
 Flexibility (*rdflib.SDO* attribute), 495  
 Flight (*rdflib.SDO* attribute), 495  
 flightDistance (*rdflib.SDO* attribute), 552  
 flightNumber (*rdflib.SDO* attribute), 552  
 FlightReservation (*rdflib.SDO* attribute), 495  
 Float (*rdflib.SDO* attribute), 495  
 float (*rdflib.XSD* attribute), 610  
 Floor (*rdflib.BRICK* attribute), 359  
 floorLevel (*rdflib.SDO* attribute), 552  
 floorLimit (*rdflib.SDO* attribute), 552  
 FloorPlan (*rdflib.SDO* attribute), 495  
 floorSize (*rdflib.SDO* attribute), 552  
 Florist (*rdflib.SDO* attribute), 495  
 Flow\_Sensor (*rdflib.BRICK* attribute), 359  
 Flow\_Setpoint (*rdflib.BRICK* attribute), 359  
 Fluid (*rdflib.BRICK* attribute), 359  
 FMRadioChannel (*rdflib.SDO* attribute), 495  
 FOAF (class in *rdflib*), 420  
 focus (*rdflib.FOAF* attribute), 421  
 focusNode (*rdflib.SH* attribute), 592  
 FollowAction (*rdflib.SDO* attribute), 495  
 followee (*rdflib.SDO* attribute), 552  
 follows (*rdflib.SDO* attribute), 552  
 followup (*rdflib.SDO* attribute), 552  
 Food\_Service\_Room (*rdflib.BRICK* attribute), 359  
 FoodEstablishment (*rdflib.SDO* attribute), 495  
 foodEstablishment (*rdflib.SDO* attribute), 552  
 FoodEstablishmentReservation (*rdflib.SDO* attribute), 495  
 FoodEvent (*rdflib.SDO* attribute), 496  
 foodEvent (*rdflib.SDO* attribute), 552  
 FoodService (*rdflib.SDO* attribute), 496  
 foodWarning (*rdflib.SDO* attribute), 552

- ForeignKey (*rdflib.CSVW* attribute), 400
  - foreignKey (*rdflib.CSVW* attribute), 401
  - forget() (*rdflib.plugins.sparql.sparql.FrozenBindings* method), 178
  - Formaldehyde\_Level\_Sensor (*rdflib.BRICK* attribute), 359
  - FormalOrganization (*rdflib.ORG* attribute), 461
  - format (*rdflib.CSVW* attribute), 401
  - format (*rdflib.DC* attribute), 407
  - format (*rdflib.DCTERMS* attribute), 412
  - Formula (class in *rdflib.plugins.parsers.notation3*), 82
  - formula() (*rdflib.plugins.parsers.notation3.SinkParser* method), 89
  - formula\_aware (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* attribute), 193
  - formula\_aware (*rdflib.plugins.stores.memory.Memory* attribute), 197
  - formula\_aware (*rdflib.plugins.stores.sparqlstore.SPARQLStore* attribute), 207
  - formula\_aware (*rdflib.store.Store* attribute), 310
  - forProperty (*rdflib.SSN* attribute), 599
  - founder (*rdflib.SDO* attribute), 552
  - founders (*rdflib.SDO* attribute), 552
  - foundingDate (*rdflib.SDO* attribute), 552
  - foundingLocation (*rdflib.SDO* attribute), 552
  - FourWheelDriveConfiguration (*rdflib.SDO* attribute), 496
  - fractionDigits (*rdflib.XSD* attribute), 610
  - fragment (*rdflib.term.URIRef* property), 328
  - fragment (*rdflib.URIRef* property), 606
  - free (*rdflib.SDO* attribute), 553
  - FreeReturn (*rdflib.SDO* attribute), 496
  - freeShippingThreshold (*rdflib.SDO* attribute), 553
  - Freeze\_Status (*rdflib.BRICK* attribute), 359
  - Freezer (*rdflib.BRICK* attribute), 359
  - Frequency (*rdflib.DCTERMS* attribute), 410
  - frequency (*rdflib.SDO* attribute), 553
  - Frequency\_Command (*rdflib.BRICK* attribute), 359
  - Frequency\_Sensor (*rdflib.BRICK* attribute), 359
  - Fresh\_Air\_Fan (*rdflib.BRICK* attribute), 359
  - Fresh\_Air\_Setpoint\_Limit (*rdflib.BRICK* attribute), 359
  - Friday (*rdflib.SDO* attribute), 496
  - Friday (*rdflib.TIME* attribute), 600
  - from\_n3() (in module *rdflib.util*), 331
  - from\_rdf() (in module *rdflib.plugins.serializers.jsonld*), 115
  - fromLocation (*rdflib.SDO* attribute), 553
  - FrontWheelDriveConfiguration (*rdflib.SDO* attribute), 496
  - Frost (*rdflib.BRICK* attribute), 359
  - Frost\_Sensor (*rdflib.BRICK* attribute), 359
  - FrozenBindings (class in *rdflib.plugins.sparql.sparql*), 178
  - FrozenDict (class in *rdflib.plugins.sparql.sparql*), 179
  - Fuel\_Oil (*rdflib.BRICK* attribute), 359
  - fuelCapacity (*rdflib.SDO* attribute), 553
  - fuelConsumption (*rdflib.SDO* attribute), 553
  - fuelEfficiency (*rdflib.SDO* attribute), 553
  - fuelType (*rdflib.SDO* attribute), 553
  - FullRefund (*rdflib.SDO* attribute), 496
  - Fume\_Hood (*rdflib.BRICK* attribute), 359
  - Fume\_Hood\_Air\_Flow\_Sensor (*rdflib.BRICK* attribute), 359
  - function (*rdflib.ODRL2* attribute), 457
  - Function (*rdflib.SH* attribute), 588
  - Function() (in module *rdflib.plugins.sparql.operators*), 164
  - functionalClass (*rdflib.SDO* attribute), 553
  - FunctionalProperty (*rdflib.OWL* attribute), 463
  - fundedBy (*rdflib.FOAF* attribute), 421
  - fundedItem (*rdflib.SDO* attribute), 553
  - funder (*rdflib.SDO* attribute), 553
  - FundingAgency (*rdflib.SDO* attribute), 496
  - FundingScheme (*rdflib.SDO* attribute), 496
  - Fungus (*rdflib.SDO* attribute), 496
  - Furniture (*rdflib.BRICK* attribute), 359
  - FurnitureStore (*rdflib.SDO* attribute), 496
- ## G
- g (*rdflib.plugins.parsers.ntriples.NTGraphSink* attribute), 99
  - Gain\_Parameter (*rdflib.BRICK* attribute), 359
  - Game (*rdflib.SDO* attribute), 496
  - game (*rdflib.SDO* attribute), 553
  - gameItem (*rdflib.SDO* attribute), 553
  - gameLocation (*rdflib.SDO* attribute), 553
  - gamePlatform (*rdflib.SDO* attribute), 553
  - GamePlayMode (*rdflib.SDO* attribute), 496
  - GameServer (*rdflib.SDO* attribute), 496
  - gameServer (*rdflib.SDO* attribute), 553
  - GameServerStatus (*rdflib.SDO* attribute), 496
  - gameTip (*rdflib.SDO* attribute), 553
  - GardenStore (*rdflib.SDO* attribute), 496
  - Gas (*rdflib.BRICK* attribute), 359
  - Gas\_Distribution (*rdflib.BRICK* attribute), 360
  - Gas\_Meter (*rdflib.BRICK* attribute), 360
  - Gas\_Sensor (*rdflib.BRICK* attribute), 360
  - Gas\_System (*rdflib.BRICK* attribute), 360
  - Gas\_Valve (*rdflib.BRICK* attribute), 360
  - Gasoline (*rdflib.BRICK* attribute), 360
  - GasStation (*rdflib.SDO* attribute), 496
  - Gastroenterologic (*rdflib.SDO* attribute), 496
  - GatedResidenceCommunity (*rdflib.SDO* attribute), 496
  - Gatehouse (*rdflib.BRICK* attribute), 360
  - gc() (*rdflib.store.Store* method), 310
  - gDay (*rdflib.XSD* attribute), 610
  - geekcode (*rdflib.FOAF* attribute), 421



- `gen` (*rdflib.plugins.stores.concurrent.ResponsibleGenerator* attribute), 196
- `gender` (*rdflib.FOAF* attribute), 421
- `gender` (*rdflib.SDO* attribute), 553
- `GenderType` (*rdflib.SDO* attribute), 496
- `Gene` (*rdflib.SDO* attribute), 496
- `GeneralContractor` (*rdflib.SDO* attribute), 496
- `GeneralDateTimeDescription` (*rdflib.TIME* attribute), 600
- `generalDay` (*rdflib.TIME* attribute), 601
- `GeneralDurationDescription` (*rdflib.TIME* attribute), 600
- `generalizationOf` (*rdflib.PROV* attribute), 471
- `generalMonth` (*rdflib.TIME* attribute), 601
- `generalYear` (*rdflib.TIME* attribute), 601
- `generated` (*rdflib.PROV* attribute), 471
- `generatedAsDerivation` (*rdflib.PROV* attribute), 471
- `generatedAtTime` (*rdflib.PROV* attribute), 471
- `generateQName`() (in module *rdflib.extras.infixowl*), 69
- `generateVoID`() (in module *rdflib.void*), 334
- `Generation` (*rdflib.PROV* attribute), 469
- `Generator_Room` (*rdflib.BRICK* attribute), 360
- `Genetic` (*rdflib.SDO* attribute), 496
- `Genitourinary` (*rdflib.SDO* attribute), 496
- `genre` (*rdflib.SDO* attribute), 553
- `geo` (*rdflib.SDO* attribute), 553
- `GeoCircle` (*rdflib.SDO* attribute), 496
- `geoContains` (*rdflib.SDO* attribute), 553
- `GeoCoordinates` (*rdflib.SDO* attribute), 496
- `geoCoveredBy` (*rdflib.SDO* attribute), 553
- `geoCovers` (*rdflib.SDO* attribute), 553
- `geoCrosses` (*rdflib.SDO* attribute), 553
- `geoDisjoint` (*rdflib.SDO* attribute), 553
- `geoEquals` (*rdflib.SDO* attribute), 553
- `geographicArea` (*rdflib.SDO* attribute), 554
- `geoIntersects` (*rdflib.SDO* attribute), 553
- `geoMidpoint` (*rdflib.SDO* attribute), 553
- `geoOverlaps` (*rdflib.SDO* attribute), 553
- `geoRadius` (*rdflib.SDO* attribute), 554
- `GeoShape` (*rdflib.SDO* attribute), 497
- `GeospatialGeometry` (*rdflib.SDO* attribute), 497
- `geoTouches` (*rdflib.SDO* attribute), 554
- `geoWithin` (*rdflib.SDO* attribute), 554
- `Geriatic` (*rdflib.SDO* attribute), 497
- `get`() (in module *rdflib.plugin*), 290
- `get`() (*rdflib.plugins.sparql.parserutils.CompValue* method), 171
- `get`() (*rdflib.plugins.sparql.sparql.QueryContext* method), 183
- `get`() (*rdflib.query.ResultRow* method), 296
- `get_alternates`() (*rdflib.parser.URLInputSource* method), 279
- `get_bindings`() (*rdflib.plugins.sparql.aggregates.Aggregate* method), 141
- `get_bnode`() (*rdflib.plugins.parsers.trix.TriXHandler* method), 111
- `get_context`() (*rdflib.ConjunctiveGraph* method), 405
- `get_context`() (*rdflib.graph.ConjunctiveGraph* method), 243
- `get_context_for_term`() (*rdflib.plugins.shared.jsonld.context.Context* method), 126
- `get_context_for_type`() (*rdflib.plugins.shared.jsonld.context.Context* method), 126
- `get_current`() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 104
- `get_graph`() (*rdflib.ConjunctiveGraph* method), 405
- `get_graph`() (*rdflib.graph.ConjunctiveGraph* method), 243
- `get_graph`() (*rdflib.plugins.shared.jsonld.context.Context* method), 126
- `get_id`() (*rdflib.plugins.shared.jsonld.context.Context* method), 126
- `get_key`() (*rdflib.plugins.shared.jsonld.context.Context* method), 127
- `get_keys`() (*rdflib.plugins.shared.jsonld.context.Context* method), 127
- `get_language`() (*rdflib.plugins.shared.jsonld.context.Context* method), 127
- `get_links`() (*rdflib.parser.URLInputSource* class method), 279
- `get_list`() (*rdflib.plugins.shared.jsonld.context.Context* method), 127
- `get_map`() (*rdflib.events.Dispatcher* method), 234
- `get_next`() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 104
- `get_parent`() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 104
- `get_rev`() (*rdflib.plugins.shared.jsonld.context.Context* method), 127
- `get_set`() (*rdflib.plugins.shared.jsonld.context.Context* method), 127
- `get_target_namespace_elements`() (in module *rdflib.tools.defined\_namespace\_creator*), 219
- `get_tree`() (in module *rdflib.util*), 331
- `get_type`() (*rdflib.plugins.shared.jsonld.context.Context* method), 127
- `get_value` (*rdflib.plugins.sparql.aggregates.Extremum* attribute), 143
- `get_value` (*rdflib.plugins.sparql.aggregates.Maximum* attribute), 144
- `get_value` (*rdflib.plugins.sparql.aggregates.Minimum* attribute), 144
- `get_value`() (*rdflib.plugins.shared.jsonld.context.Context* method), 127
- `get_value`() (*rdflib.plugins.sparql.aggregates.Average* method), 141

- `get_value()` (*rdflib.plugins.sparql.aggregates.Counter* method), 142
- `get_value()` (*rdflib.plugins.sparql.aggregates.GroupConcat* method), 143
- `get_value()` (*rdflib.plugins.sparql.aggregates.Sample* method), 144
- `get_value()` (*rdflib.plugins.sparql.aggregates.Sum* method), 145
- `getallmatchingheaders()` (*rdflib.parser.URLInputSource* class method), 279
- `getClass()` (*rdflib.plugin.PKGPlugin* method), 289
- `getClass()` (*rdflib.plugin.Plugin* method), 290
- `GetIdentifiedClasses()` (in module *rdflib.extras.infixowl*), 64
- `getIntersections` (*rdflib.extras.infixowl.BooleanClass* attribute), 59
- `getPublicId()` (*rdflib.parser.PythonInputSource* method), 277
- `getQName()` (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 116
- `getQName()` (*rdflib.plugins.serializers.n3.N3Serializer* method), 117
- `getQName()` (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 122
- `getSystemId()` (*rdflib.parser.PythonInputSource* method), 277
- `GettingAccessHealthAspect` (*rdflib.SDO* attribute), 497
- `gettingTestedInfo` (*rdflib.SDO* attribute), 554
- `getUnions` (*rdflib.extras.infixowl.BooleanClass* attribute), 59
- `GitBranch` (*rdflib.DOAP* attribute), 414
- `GitRepository` (*rdflib.DOAP* attribute), 414
- `give` (*rdflib.ODRL2* attribute), 457
- `GiveAction` (*rdflib.SDO* attribute), 497
- `givenName` (*rdflib.FOAF* attribute), 421
- `givenname` (*rdflib.FOAF* attribute), 421
- `givenName` (*rdflib.SDO* attribute), 554
- `globalLocationNumber` (*rdflib.SDO* attribute), 554
- `GlutenFreeDiet` (*rdflib.SDO* attribute), 497
- `Glycol` (*rdflib.BRICK* attribute), 360
- `gMonth` (*rdflib.XSD* attribute), 610
- `gMonthDay` (*rdflib.XSD* attribute), 610
- `GolfCourse` (*rdflib.SDO* attribute), 497
- `governmentBenefitsInfo` (*rdflib.SDO* attribute), 554
- `GovernmentBenefitsType` (*rdflib.SDO* attribute), 497
- `GovernmentBuilding` (*rdflib.SDO* attribute), 497
- `GovernmentOffice` (*rdflib.SDO* attribute), 497
- `GovernmentOrganization` (*rdflib.SDO* attribute), 497
- `GovernmentPermit` (*rdflib.SDO* attribute), 497
- `GovernmentService` (*rdflib.SDO* attribute), 497
- `gracePeriod` (*rdflib.SDO* attribute), 554
- `Grant` (*rdflib.SDO* attribute), 497
- `grantee` (*rdflib.SDO* attribute), 554
- `grantUse` (*rdflib.ODRL2* attribute), 457
- `Graph` (class in *rdflib*), 423
- `Graph` (class in *rdflib.graph*), 250
- `graph` (*rdflib.plugins.sparql.processor.SPARQLResult* attribute), 174
- `graph` (*rdflib.plugins.sparql.results.jsonresults.JSONResult* attribute), 133
- `graph` (*rdflib.plugins.sparql.results.rdfresults.RDFResult* attribute), 135
- `graph` (*rdflib.plugins.sparql.results.xmlresults.XMLResult* attribute), 138
- `graph` (*rdflib.resource.Resource* property), 305
- `Graph()` (in module *rdflib.plugins.sparql.algebra*), 146
- `graph()` (*rdflib.Dataset* method), 419
- `graph()` (*rdflib.graph.Dataset* method), 248
- `graph()` (*rdflib.plugins.parsers.trig.TrigSinkParser* method), 109
- `graph_aware` (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* attribute), 194
- `graph_aware` (*rdflib.plugins.stores.memory.Memory* attribute), 197
- `graph_aware` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* attribute), 208
- `graph_aware` (*rdflib.store.Store* attribute), 310
- `graph_diff()` (in module *rdflib.compare*), 228
- `graph_digest()` (*rdflib.compare.IsomorphicGraph* method), 228
- `graph_key` (*rdflib.plugins.shared.jsonld.context.Context* property), 128
- `GraphicNovel` (*rdflib.SDO* attribute), 497
- `GraphResultParser` (class in *rdflib.plugins.sparql.results.graph*), 133
- `graphs()` (*rdflib.Dataset* method), 419
- `graphs()` (*rdflib.graph.Dataset* method), 249
- `greater` (*rdflib.SDO* attribute), 554
- `greaterOrEqual` (*rdflib.SDO* attribute), 554
- `GroceryStore` (*rdflib.SDO* attribute), 497
- `grossArea` (*rdflib.BRICK* attribute), 398
- `Group` (*rdflib.FOAF* attribute), 420
- `Group` (*rdflib.ODRL2* attribute), 453
- `group` (*rdflib.SH* attribute), 592
- `Group()` (in module *rdflib.plugins.sparql.algebra*), 146
- `GroupBoardingPolicy` (*rdflib.SDO* attribute), 497
- `groupChar` (*rdflib.CSVW* attribute), 401
- `GroupConcat` (class in *rdflib.plugins.sparql.aggregates*), 143
- `gt` (*rdflib.ODRL2* attribute), 457
- `gteq` (*rdflib.ODRL2* attribute), 457
- `gtin` (*rdflib.SDO* attribute), 554
- `gtin12` (*rdflib.SDO* attribute), 554
- `gtin13` (*rdflib.SDO* attribute), 554
- `gtin14` (*rdflib.SDO* attribute), 554
- `gtin8` (*rdflib.SDO* attribute), 554

guess\_format() (in module *rdflib.util*), 332  
 Guide (*rdflib.SDO* attribute), 497  
 guideline (*rdflib.SDO* attribute), 554  
 guidelineDate (*rdflib.SDO* attribute), 554  
 guidelineSubject (*rdflib.SDO* attribute), 554  
 gYear (*rdflib.XSD* attribute), 610  
 gYearMonth (*rdflib.XSD* attribute), 610  
 Gynecologic (*rdflib.SDO* attribute), 497

## H

Hackathon (*rdflib.SDO* attribute), 497  
 hadActivity (*rdflib.PROV* attribute), 471  
 hadDelegate (*rdflib.PROV* attribute), 471  
 hadDerivation (*rdflib.PROV* attribute), 471  
 hadDictionaryMember (*rdflib.PROV* attribute), 471  
 hadGeneration (*rdflib.PROV* attribute), 471  
 hadInfluence (*rdflib.PROV* attribute), 471  
 hadMember (*rdflib.PROV* attribute), 471  
 hadPlan (*rdflib.PROV* attribute), 471  
 hadPrimarySource (*rdflib.PROV* attribute), 471  
 hadRevision (*rdflib.PROV* attribute), 471  
 hadRole (*rdflib.DCAT* attribute), 409  
 hadRole (*rdflib.PROV* attribute), 471  
 hadUsage (*rdflib.PROV* attribute), 471  
 Hail (*rdflib.BRICK* attribute), 360  
 Hail\_Sensor (*rdflib.BRICK* attribute), 360  
 HairSalon (*rdflib.SDO* attribute), 497  
 HalalDiet (*rdflib.SDO* attribute), 497  
 Hallway (*rdflib.BRICK* attribute), 360  
 handleAnnotation() (*rdflib.extras.infixowl.AnnotatableTerms* method), 58  
 handlingTime (*rdflib.SDO* attribute), 554  
 Hardcover (*rdflib.SDO* attribute), 497  
 HardwareStore (*rdflib.SDO* attribute), 497  
 has\_anchor (*rdflib.PROV* attribute), 472  
 has\_provenance (*rdflib.PROV* attribute), 472  
 has\_query\_service (*rdflib.PROV* attribute), 472  
 hasAddress (*rdflib.BRICK* attribute), 398  
 hasArtifact (*rdflib.PROF* attribute), 467  
 hasAssociatedTag (*rdflib.BRICK* attribute), 398  
 hasBeginning (*rdflib.TIME* attribute), 601  
 hasBioChemEntityPart (*rdflib.SDO* attribute), 554  
 hasBioPolymerSequence (*rdflib.SDO* attribute), 554  
 hasBroadcastChannel (*rdflib.SDO* attribute), 554  
 hasCategoryCode (*rdflib.SDO* attribute), 554  
 hasCourse (*rdflib.SDO* attribute), 554  
 hasCourseInstance (*rdflib.SDO* attribute), 555  
 hasCredential (*rdflib.SDO* attribute), 555  
 hasDateTimeDescription (*rdflib.TIME* attribute), 601  
 hasDefinedTerm (*rdflib.SDO* attribute), 555  
 hasDeliveryMethod (*rdflib.SDO* attribute), 555  
 hasDeployment (*rdflib.SSN* attribute), 599

hasDigitalDocumentPermission (*rdflib.SDO* attribute), 555  
 hasDriveThroughService (*rdflib.SDO* attribute), 555  
 hasDuration (*rdflib.TIME* attribute), 601  
 hasDurationDescription (*rdflib.TIME* attribute), 601  
 hasEnd (*rdflib.TIME* attribute), 601  
 hasEnergyConsumptionDetails (*rdflib.SDO* attribute), 555  
 hasEnergyEfficiencyCategory (*rdflib.SDO* attribute), 555  
 hasFeatureOfInterest (*rdflib.SOSA* attribute), 598  
 hasFormat (*rdflib.DCTERMS* attribute), 412  
 hasHealthAspect (*rdflib.SDO* attribute), 555  
 hashtriples() (*rdflib.tools.graphisomorphism.IsomorphicTestableGraph* method), 220  
 hasInput (*rdflib.SSN* attribute), 599  
 hasInputSubstance (*rdflib.BRICK* attribute), 398  
 hasKey (*rdflib.OWL* attribute), 465  
 hasLocation (*rdflib.BRICK* attribute), 398  
 hasMap (*rdflib.SDO* attribute), 555  
 hasMeasurement (*rdflib.SDO* attribute), 555  
 hasMember (*rdflib.ORG* attribute), 461  
 hasMembership (*rdflib.ORG* attribute), 461  
 hasMenu (*rdflib.SDO* attribute), 555  
 hasMenuItem (*rdflib.SDO* attribute), 555  
 hasMenuSection (*rdflib.SDO* attribute), 555  
 hasMerchantReturnPolicy (*rdflib.SDO* attribute), 555  
 hasMolecularFunction (*rdflib.SDO* attribute), 555  
 hasOccupation (*rdflib.SDO* attribute), 555  
 hasOfferCatalog (*rdflib.SDO* attribute), 555  
 hasOutput (*rdflib.SSN* attribute), 599  
 hasOutputSubstance (*rdflib.BRICK* attribute), 398  
 hasPart (*rdflib.BRICK* attribute), 398  
 hasPart (*rdflib.DCTERMS* attribute), 412  
 hasPart (*rdflib.ODRL2* attribute), 457  
 hasPart (*rdflib.SDO* attribute), 555  
 hasPoint (*rdflib.BRICK* attribute), 398  
 hasPolicy (*rdflib.ODRL2* attribute), 457  
 hasPOS (*rdflib.SDO* attribute), 555  
 hasPost (*rdflib.ORG* attribute), 461  
 hasPrimarySite (*rdflib.ORG* attribute), 461  
 hasProperty (*rdflib.SSN* attribute), 599  
 hasQUOTReference (*rdflib.BRICK* attribute), 398  
 hasRegisteredSite (*rdflib.ORG* attribute), 461  
 hasRepresentation (*rdflib.SDO* attribute), 555  
 hasResource (*rdflib.PROF* attribute), 467  
 hasResult (*rdflib.SOSA* attribute), 598  
 hasRole (*rdflib.PROF* attribute), 467  
 hasSample (*rdflib.SOSA* attribute), 598  
 hasSelf (*rdflib.OWL* attribute), 465  
 hasSimpleResult (*rdflib.SOSA* attribute), 598  
 hasSite (*rdflib.ORG* attribute), 461  
 hasSubOrganization (*rdflib.ORG* attribute), 461  
 hasSubSystem (*rdflib.SSN* attribute), 599



- hasTag (*rdflib.BRICK* attribute), 398
- hasTemporalDuration (*rdflib.TIME* attribute), 601
- hasTime (*rdflib.TIME* attribute), 601
- hasTimeseriesId (*rdflib.BRICK* attribute), 398
- hasToken (*rdflib.PROF* attribute), 467
- hasTopConcept (*rdflib.SKOS* attribute), 596
- hasTRS (*rdflib.TIME* attribute), 601
- hasUnit (*rdflib.BRICK* attribute), 398
- hasUnit (*rdflib.ORG* attribute), 462
- hasValue (*rdflib.extras.infixowl.Restriction* property), 68
- hasValue (*rdflib.OWL* attribute), 465
- hasValue (*rdflib.SH* attribute), 592
- HasValueConstraintComponent (*rdflib.SH* attribute), 588
- hasVariant (*rdflib.SDO* attribute), 555
- hasVersion (*rdflib.DCTERMS* attribute), 412
- hasXSDDuration (*rdflib.TIME* attribute), 602
- Hazardous\_Materials\_Storage (*rdflib.BRICK* attribute), 360
- Head (*rdflib.ORG* attribute), 461
- Head (*rdflib.SDO* attribute), 498
- header (*rdflib.CSVW* attribute), 401
- headerRowCount (*rdflib.CSVW* attribute), 401
- headline (*rdflib.SDO* attribute), 555
- headOf (*rdflib.ORG* attribute), 462
- HealthAndBeautyBusiness (*rdflib.SDO* attribute), 498
- HealthAspectEnumeration (*rdflib.SDO* attribute), 498
- HealthCare (*rdflib.SDO* attribute), 498
- healthcareReportingData (*rdflib.SDO* attribute), 556
- HealthClub (*rdflib.SDO* attribute), 498
- healthCondition (*rdflib.SDO* attribute), 555
- HealthInsurancePlan (*rdflib.SDO* attribute), 498
- healthPlanCoinsuranceOption (*rdflib.SDO* attribute), 555
- healthPlanCoinsuranceRate (*rdflib.SDO* attribute), 556
- healthPlanCopay (*rdflib.SDO* attribute), 556
- healthPlanCopayOption (*rdflib.SDO* attribute), 556
- healthPlanCostSharing (*rdflib.SDO* attribute), 556
- HealthPlanCostSharingSpecification (*rdflib.SDO* attribute), 498
- healthPlanDrugOption (*rdflib.SDO* attribute), 556
- healthPlanDrugTier (*rdflib.SDO* attribute), 556
- HealthPlanFormulary (*rdflib.SDO* attribute), 498
- healthPlanId (*rdflib.SDO* attribute), 556
- healthPlanMarketingUrl (*rdflib.SDO* attribute), 556
- HealthPlanNetwork (*rdflib.SDO* attribute), 498
- healthPlanNetworkId (*rdflib.SDO* attribute), 556
- healthPlanNetworkTier (*rdflib.SDO* attribute), 556
- healthPlanPharmacyCategory (*rdflib.SDO* attribute), 556
- HealthTopicContent (*rdflib.SDO* attribute), 498
- HearingImpairedSupported (*rdflib.SDO* attribute), 498
- Heat\_Exchange (*rdflib.BRICK* attribute), 360
- Heat\_Exchange\_Supply\_Water\_Temperature\_Sensor (*rdflib.BRICK* attribute), 360
- Heat\_Exchange\_System\_Enable\_Status (*rdflib.BRICK* attribute), 360
- Heat\_Recovery\_Hot\_Water\_System (*rdflib.BRICK* attribute), 361
- Heat\_Sensor (*rdflib.BRICK* attribute), 361
- Heat\_Wheel (*rdflib.BRICK* attribute), 361
- Heat\_Wheel\_VFD (*rdflib.BRICK* attribute), 361
- Heating\_Coil (*rdflib.BRICK* attribute), 361
- Heating\_Command (*rdflib.BRICK* attribute), 361
- Heating\_Demand\_Sensor (*rdflib.BRICK* attribute), 361
- Heating\_Demand\_Setpoint (*rdflib.BRICK* attribute), 361
- Heating\_Discharge\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 361
- Heating\_Discharge\_Air\_Temperature\_Deadband\_Setpoint (*rdflib.BRICK* attribute), 361
- Heating\_Discharge\_Air\_Temperature\_Integral\_Time\_Parameter (*rdflib.BRICK* attribute), 361
- Heating\_Discharge\_Air\_Temperature\_Proportional\_Band\_Parameter (*rdflib.BRICK* attribute), 361
- Heating\_Start\_Stop\_Status (*rdflib.BRICK* attribute), 361
- Heating\_Supply\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 361
- Heating\_Supply\_Air\_Temperature\_Deadband\_Setpoint (*rdflib.BRICK* attribute), 361
- Heating\_Supply\_Air\_Temperature\_Integral\_Time\_Parameter (*rdflib.BRICK* attribute), 361
- Heating\_Supply\_Air\_Temperature\_Proportional\_Band\_Parameter (*rdflib.BRICK* attribute), 361
- Heating\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 362
- Heating\_Thermal\_Power\_Sensor (*rdflib.BRICK* attribute), 362
- Heating\_Valve (*rdflib.BRICK* attribute), 362
- Heating\_Ventilation\_Air\_Conditioning\_System (*rdflib.BRICK* attribute), 362
- height (*rdflib.SDO* attribute), 556
- heldBy (*rdflib.ORG* attribute), 462
- helper (*rdflib.DOAP* attribute), 415
- Hematologic (*rdflib.SDO* attribute), 498
- here() (*rdflib.plugins.parsers.notation3.SinkParser* method), 90
- hexBinary (*rdflib.XSD* attribute), 610
- hexify() (in module *rdflib.plugins.parsers.notation3*), 95
- HextuplesParser (class in *rdflib.plugins.parsers.hext*), 79
- HextuplesSerializer (class in *rdflib.plugins.serializers.hext*), 113
- HgRepository (*rdflib.DOAP* attribute), 414



hiddenLabel (rdflib.SKOS attribute), 596  
 HierarchicalCodeList (rdflib.QB attribute), 476  
 hierarchyRoot (rdflib.QB attribute), 476  
 High\_CO2\_Alarm (rdflib.BRICK attribute), 362  
 High\_Discharge\_Air\_Temperature\_Alarm (rdflib.BRICK attribute), 362  
 High\_Head\_Pressure\_Alarm (rdflib.BRICK attribute), 362  
 High\_Humidity\_Alarm (rdflib.BRICK attribute), 362  
 High\_Humidity\_Alarm\_Parameter (rdflib.BRICK attribute), 362  
 High\_Outside\_Air\_Lockout\_Temperature\_Differential\_Pressure\_Parameter (rdflib.BRICK attribute), 362  
 High\_Return\_Air\_Temperature\_Alarm (rdflib.BRICK attribute), 362  
 High\_Static\_Pressure\_Cutout\_Setpoint\_Limit (rdflib.BRICK attribute), 362  
 High\_Temperature\_Alarm (rdflib.BRICK attribute), 362  
 High\_Temperature\_Alarm\_Parameter (rdflib.BRICK attribute), 362  
 High\_Temperature\_Hot\_Water\_Return\_Temperature\_Sensor (rdflib.BRICK attribute), 362  
 High\_Temperature\_Hot\_Water\_Supply\_Temperature\_Sensor (rdflib.BRICK attribute), 362  
 highPrice (rdflib.SDO attribute), 556  
 HighSchool (rdflib.SDO attribute), 498  
 HinduDiet (rdflib.SDO attribute), 498  
 HinduTemple (rdflib.SDO attribute), 498  
 hiringOrganization (rdflib.SDO attribute), 556  
 historyNote (rdflib.SKOS attribute), 596  
 HobbyShop (rdflib.SDO attribute), 498  
 Hold\_Status (rdflib.BRICK attribute), 362  
 holdingArchive (rdflib.SDO attribute), 556  
 holds (rdflib.ORG attribute), 462  
 holdsAccount (rdflib.FOAF attribute), 421  
 HomeAndConstructionBusiness (rdflib.SDO attribute), 498  
 HomeGoodsStore (rdflib.SDO attribute), 498  
 homeLocation (rdflib.SDO attribute), 556  
 Homeopathic (rdflib.SDO attribute), 498  
 homepage (rdflib.DOAP attribute), 415  
 homepage (rdflib.FOAF attribute), 421  
 homeTeam (rdflib.SDO attribute), 556  
 honorificPrefix (rdflib.SDO attribute), 556  
 honorificSuffix (rdflib.SDO attribute), 556  
 Hospital (rdflib.SDO attribute), 498  
 hospitalAffiliation (rdflib.SDO attribute), 556  
 Hospitality\_Box (rdflib.BRICK attribute), 363  
 Hostel (rdflib.SDO attribute), 498  
 hostingOrganization (rdflib.SDO attribute), 556  
 hosts (rdflib.SOSA attribute), 598  
 Hot\_Box (rdflib.BRICK attribute), 363  
 Hot\_Water (rdflib.BRICK attribute), 363  
 Hot\_Water\_Baseboard\_Radiator (rdflib.BRICK attribute), 363  
 Hot\_Water\_Coil (rdflib.BRICK attribute), 363  
 Hot\_Water\_Differential\_Pressure\_Deadband\_Setpoint (rdflib.BRICK attribute), 363  
 Hot\_Water\_Differential\_Pressure\_Integral\_Time\_Parameter (rdflib.BRICK attribute), 363  
 Hot\_Water\_Differential\_Pressure\_Load\_Shed\_Reset\_Status (rdflib.BRICK attribute), 363  
 Hot\_Water\_Differential\_Pressure\_Load\_Shed\_Status (rdflib.BRICK attribute), 363  
 Hot\_Water\_Differential\_Pressure\_Proportional\_Band\_Parameter (rdflib.BRICK attribute), 363  
 Hot\_Water\_Differential\_Pressure\_Sensor (rdflib.BRICK attribute), 363  
 Hot\_Water\_Differential\_Pressure\_Setpoint (rdflib.BRICK attribute), 363  
 Hot\_Water\_Differential\_Temperature\_Sensor (rdflib.BRICK attribute), 363  
 Hot\_Water\_Discharge\_Flow\_Sensor (rdflib.BRICK attribute), 363  
 Hot\_Water\_Discharge\_Flow\_Setpoint (rdflib.BRICK attribute), 363  
 Hot\_Water\_Discharge\_Temperature\_Load\_Shed\_Status (rdflib.BRICK attribute), 363  
 Hot\_Water\_Flow\_Sensor (rdflib.BRICK attribute), 364  
 Hot\_Water\_Flow\_Setpoint (rdflib.BRICK attribute), 364  
 Hot\_Water\_Loop (rdflib.BRICK attribute), 364  
 Hot\_Water\_Meter (rdflib.BRICK attribute), 364  
 Hot\_Water\_Pump (rdflib.BRICK attribute), 364  
 Hot\_Water\_Radiator (rdflib.BRICK attribute), 364  
 Hot\_Water\_Return\_Flow\_Sensor (rdflib.BRICK attribute), 364  
 Hot\_Water\_Return\_Temperature\_Sensor (rdflib.BRICK attribute), 364  
 Hot\_Water\_Static\_Pressure\_Setpoint (rdflib.BRICK attribute), 364  
 Hot\_Water\_Supply\_Flow\_Sensor (rdflib.BRICK attribute), 364  
 Hot\_Water\_Supply\_Flow\_Setpoint (rdflib.BRICK attribute), 364  
 Hot\_Water\_Supply\_Temperature\_High\_Reset\_Setpoint (rdflib.BRICK attribute), 364  
 Hot\_Water\_Supply\_Temperature\_Load\_Shed\_Status (rdflib.BRICK attribute), 364  
 Hot\_Water\_Supply\_Temperature\_Low\_Reset\_Setpoint (rdflib.BRICK attribute), 364  
 Hot\_Water\_Supply\_Temperature\_Sensor (rdflib.BRICK attribute), 364  
 Hot\_Water\_System (rdflib.BRICK attribute), 364  
 Hot\_Water\_System\_Enable\_Command (rdflib.BRICK attribute), 364  
 Hot\_Water\_Temperature\_Setpoint (rdflib.BRICK attribute), 364

- tribute*), 364
- Hot\_Water\_Usage\_Sensor (*rdflib.BRICK attribute*), 364
- Hot\_Water\_Valve (*rdflib.BRICK attribute*), 365
- Hotel (*rdflib.SDO attribute*), 498
- HotelRoom (*rdflib.SDO attribute*), 498
- hour (*rdflib.TIME attribute*), 602
- hour (*rdflib.XSD attribute*), 610
- hours (*rdflib.TIME attribute*), 602
- hoursAvailable (*rdflib.SDO attribute*), 557
- House (*rdflib.SDO attribute*), 498
- HousePainter (*rdflib.SDO attribute*), 498
- HowItWorksHealthAspect (*rdflib.SDO attribute*), 499
- HowOrWhereHealthAspect (*rdflib.SDO attribute*), 499
- howPerformed (*rdflib.SDO attribute*), 557
- HowTo (*rdflib.SDO attribute*), 499
- HowToDirection (*rdflib.SDO attribute*), 499
- HowToItem (*rdflib.SDO attribute*), 499
- HowToSection (*rdflib.SDO attribute*), 499
- HowToStep (*rdflib.SDO attribute*), 499
- HowToSupply (*rdflib.SDO attribute*), 499
- HowToTip (*rdflib.SDO attribute*), 499
- HowToTool (*rdflib.SDO attribute*), 499
- HTML (*rdflib.RDF attribute*), 477
- http\_open() (*examples.secure\_with\_urlopen.SecuredHTTPHandler* method), 26
- httpMethod (*rdflib.SDO attribute*), 557
- Humidification\_Start\_Stop\_Status (*rdflib.BRICK attribute*), 365
- Humidifier (*rdflib.BRICK attribute*), 365
- Humidifier\_Fault\_Status (*rdflib.BRICK attribute*), 365
- Humidify\_Command (*rdflib.BRICK attribute*), 365
- Humidity\_Alarm (*rdflib.BRICK attribute*), 365
- Humidity\_Parameter (*rdflib.BRICK attribute*), 365
- Humidity\_Sensor (*rdflib.BRICK attribute*), 365
- Humidity\_Setpoint (*rdflib.BRICK attribute*), 365
- Humidity\_Tolerance\_Parameter (*rdflib.BRICK attribute*), 365
- HVAC\_Equipment (*rdflib.BRICK attribute*), 360
- HVAC\_System (*rdflib.BRICK attribute*), 360
- HVAC\_Zone (*rdflib.BRICK attribute*), 360
- HVACBusiness (*rdflib.SDO attribute*), 497
- HX (*rdflib.BRICK attribute*), 360
- HyperToc (*rdflib.SDO attribute*), 499
- HyperTocEntry (*rdflib.SDO attribute*), 499
- I
- iataCode (*rdflib.SDO attribute*), 557
- icaoCode (*rdflib.SDO attribute*), 557
- Ice (*rdflib.BRICK attribute*), 365
- Ice\_Tank\_Leaving\_Water\_Temperature\_Sensor (*rdflib.BRICK attribute*), 365
- IceCreamShop (*rdflib.SDO attribute*), 499
- icqChatID (*rdflib.FOAF attribute*), 421
- ID (*rdflib.plugins.parsers.RDFVOC.RDFVOC attribute*), 79
- id (*rdflib.plugins.parsers.rdfxml.ElementHandler attribute*), 102
- id (*rdflib.plugins.shared.jsonld.context.Term attribute*), 129
- ID (*rdflib.XSD attribute*), 609
- id() (*rdflib.plugins.parsers.notation3.Formula method*), 83
- id\_key (*rdflib.plugins.shared.jsonld.context.Context property*), 128
- IdentifiedNode (*class in rdflib*), 441
- IdentifiedNode (*class in rdflib.term*), 314
- Identifier (*class in rdflib.term*), 315
- identifier (*rdflib.DC attribute*), 407
- identifier (*rdflib.DCTERMS attribute*), 413
- identifier (*rdflib.extras.infixowl.Individual property*), 65
- identifier (*rdflib.Graph property*), 432
- identifier (*rdflib.graph.Graph property*), 259
- identifier (*rdflib.ORG attribute*), 462
- identifier (*rdflib.plugins.stores.berkeleydb.BerkeleyDB property*), 194
- Identifier (*rdflib.resource.Resource property*), 305
- identifier (*rdflib.SDO attribute*), 557
- identifyingExam (*rdflib.SDO attribute*), 557
- identifyingTest (*rdflib.SDO attribute*), 557
- IDF (*rdflib.BRICK attribute*), 365
- IDREF (*rdflib.XSD attribute*), 609
- IDREFS (*rdflib.XSD attribute*), 609
- ignorableWhitespace() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler method*), 104
- ignorableWhitespace() (*rdflib.plugins.parsers.trix.TriXHandler method*), 111
- ignore (*rdflib.ODRL2 attribute*), 457
- IgnoreAction (*rdflib.SDO attribute*), 499
- ignoredProperties (*rdflib.SH attribute*), 592
- ignoreExprs (*rdflib.plugins.sparql.parserutils.ParamList attribute*), 172
- ill\_typed (*rdflib.Literal property*), 449
- ill\_typed (*rdflib.term.Literal property*), 325
- Illuminance\_Sensor (*rdflib.BRICK attribute*), 365
- illustrator (*rdflib.SDO attribute*), 557
- Image (*rdflib.DCMITYPE attribute*), 410
- Image (*rdflib.FOAF attribute*), 420
- image (*rdflib.SDO attribute*), 557
- ImageGallery (*rdflib.SDO attribute*), 499
- ImageObject (*rdflib.SDO attribute*), 499
- ImageObjectSnapshot (*rdflib.SDO attribute*), 499
- imagingTechnique (*rdflib.SDO attribute*), 557
- ImagingTest (*rdflib.SDO attribute*), 499

- Imbalance\_Sensor (rdflib.BRICK attribute), 365
- img (rdflib.FOAF attribute), 421
- implementedBy (rdflib.SSN attribute), 599
- implements (rdflib.DOAP attribute), 415
- implements (rdflib.SSN attribute), 599
- implies (rdflib.ODRL2 attribute), 457
- imports (rdflib.extras.infixowl.Ontology property), 67
- imports (rdflib.OWL attribute), 465
- IMT (rdflib.DCTERMS attribute), 410
- inAlbum (rdflib.SDO attribute), 557
- inBroadcastLineup (rdflib.SDO attribute), 557
- incentiveCompensation (rdflib.SDO attribute), 557
- incentives (rdflib.SDO attribute), 557
- inChI (rdflib.SDO attribute), 557
- inChIKey (rdflib.SDO attribute), 557
- include (rdflib.ODRL2 attribute), 457
- includedComposition (rdflib.SDO attribute), 557
- includedDataCatalog (rdflib.SDO attribute), 557
- includedIn (rdflib.ODRL2 attribute), 457
- includedInDataCatalog (rdflib.SDO attribute), 558
- includedInHealthInsurancePlan (rdflib.SDO attribute), 558
- includedRiskFactor (rdflib.SDO attribute), 558
- includesAttraction (rdflib.SDO attribute), 558
- includesHealthPlanFormulary (rdflib.SDO attribute), 558
- includesHealthPlanNetwork (rdflib.SDO attribute), 558
- includesObject (rdflib.SDO attribute), 558
- inCodeSet (rdflib.SDO attribute), 557
- incompatibleWith (rdflib.OWL attribute), 465
- InConstraintComponent (rdflib.SH attribute), 588
- increasesRiskOf (rdflib.SDO attribute), 558
- inDataset (rdflib.VOID attribute), 607
- inDateTime (rdflib.TIME attribute), 602
- inDefinedTermSet (rdflib.SDO attribute), 557
- indent (rdflib.plugins.serializers.xmlwriter.XMLWriter property), 123
- indent() (rdflib.plugins.serializers.n3.N3Serializer method), 117
- indent() (rdflib.plugins.serializers.turtle.RecursiveSerializer method), 121
- indentString (rdflib.plugins.serializers.longturtle.LongTurtleSerializer attribute), 116
- indentString (rdflib.plugins.serializers.trig.TrigSerializer attribute), 120
- indentString (rdflib.plugins.serializers.turtle.RecursiveSerializer attribute), 121
- indentString (rdflib.plugins.serializers.turtle.TurtleSerializer attribute), 122
- inDeployment (rdflib.SSN attribute), 599
- index (rdflib.ODRL2 attribute), 457
- index (rdflib.plugins.shared.jsonld.context.Term attribute), 130
- index() (rdflib.collection.Collection method), 225
- index() (rdflib.container.Container method), 233
- index() (rdflib.extras.infixowl.OWLRLDListProxy method), 67
- Individual (class in rdflib.extras.infixowl), 64
- Individual (rdflib.ODRL2 attribute), 454
- IndividualProduct (rdflib.SDO attribute), 499
- Induction\_Unit (rdflib.BRICK attribute), 365
- industry (rdflib.ODRL2 attribute), 457
- industry (rdflib.SDO attribute), 558
- ineligibleRegion (rdflib.SDO attribute), 558
- Infectious (rdflib.SDO attribute), 499
- infectiousAgent (rdflib.SDO attribute), 558
- InfectiousAgentClass (rdflib.SDO attribute), 499
- infectiousAgentClass (rdflib.SDO attribute), 558
- InfectiousDisease (rdflib.SDO attribute), 499
- Infix (class in rdflib.extras.infixowl), 65
- Influence (rdflib.PROV attribute), 469
- influenced (rdflib.PROV attribute), 472
- influencer (rdflib.PROV attribute), 472
- Info (rdflib.SH attribute), 588
- InForce (rdflib.SDO attribute), 499
- inform (rdflib.ODRL2 attribute), 457
- InformAction (rdflib.SDO attribute), 499
- Information\_Area (rdflib.BRICK attribute), 365
- informed (rdflib.PROV attribute), 472
- informedParty (rdflib.ODRL2 attribute), 457
- informingParty (rdflib.ODRL2 attribute), 457
- ingredients (rdflib.SDO attribute), 558
- IngredientsHealthAspect (rdflib.SDO attribute), 499
- inherit (rdflib.CSVW attribute), 401
- inheritAllowed (rdflib.ODRL2 attribute), 457
- inheritFrom (rdflib.ODRL2 attribute), 457
- inheritRelation (rdflib.ODRL2 attribute), 457
- inker (rdflib.SDO attribute), 558
- inLanguage (rdflib.SDO attribute), 557
- inPlaylist (rdflib.SDO attribute), 557
- inProductGroupWithID (rdflib.SDO attribute), 557
- Input (rdflib.SSN attribute), 599
- InputSource (class in rdflib.parser), 276
- inScheme (rdflib.SKOS attribute), 596
- InsertAction (rdflib.SDO attribute), 500
- insertedKeyEntityPair (rdflib.PROV attribute), 472
- Insertion (rdflib.PROV attribute), 469
- insertion (rdflib.SDO attribute), 558
- inside (rdflib.TIME attribute), 602
- Inside\_Face\_Surface\_Temperature\_Sensor (rdflib.BRICK attribute), 365
- Inside\_Face\_Surface\_Temperature\_Setpoint (rdflib.BRICK attribute), 365
- install (rdflib.ODRL2 attribute), 457
- InstallAction (rdflib.SDO attribute), 500
- Installment (rdflib.SDO attribute), 500
- installUrl (rdflib.SDO attribute), 558



- Instant (*rdflib.TIME* attribute), 600
- InstantaneousEvent (*rdflib.PROV* attribute), 469
- InStock (*rdflib.SDO* attribute), 499
- InStoreOnly (*rdflib.SDO* attribute), 499
- inStoreReturnsOffered (*rdflib.SDO* attribute), 557
- instructionalMethod (*rdflib.DCTERMS* attribute), 413
- instructor (*rdflib.SDO* attribute), 558
- instrument (*rdflib.SDO* attribute), 558
- inSupportOf (*rdflib.SDO* attribute), 557
- InsuranceAgency (*rdflib.SDO* attribute), 500
- int (*rdflib.XSD* attribute), 610
- Intake\_Air\_Filter (*rdflib.BRICK* attribute), 365
- Intake\_Air\_Temperature\_Sensor (*rdflib.BRICK* attribute), 366
- Intangible (*rdflib.SDO* attribute), 500
- Integer (*rdflib.SDO* attribute), 500
- integer (*rdflib.XSD* attribute), 610
- Integral\_Gain\_Parameter (*rdflib.BRICK* attribute), 366
- Integral\_Time\_Parameter (*rdflib.BRICK* attribute), 366
- inTemporalPosition (*rdflib.TIME* attribute), 602
- intensity (*rdflib.SDO* attribute), 558
- InteractAction (*rdflib.SDO* attribute), 500
- interactingDrug (*rdflib.SDO* attribute), 558
- interactionCount (*rdflib.SDO* attribute), 558
- InteractionCounter (*rdflib.SDO* attribute), 500
- interactionService (*rdflib.SDO* attribute), 558
- interactionStatistic (*rdflib.SDO* attribute), 558
- interactionType (*rdflib.SDO* attribute), 559
- InteractiveResource (*rdflib.DCMITYPE* attribute), 410
- interactivityType (*rdflib.SDO* attribute), 559
- Intercom\_Equipment (*rdflib.BRICK* attribute), 366
- interest (*rdflib.FOAF* attribute), 422
- interestRate (*rdflib.SDO* attribute), 559
- Interface (*rdflib.BRICK* attribute), 366
- intern() (*rdflib.plugins.parsers.notation3.RDFSink* method), 84
- internal\_hash() (*rdflib.compare.IsomorphicGraph* method), 228
- internal\_hash() (*rdflib.tools.graphisomorphism.IsomorphicTestableGraph* method), 220
- InternationalTrial (*rdflib.SDO* attribute), 500
- InternetCafe (*rdflib.SDO* attribute), 500
- interpretedAsClaim (*rdflib.SDO* attribute), 559
- intersection (*rdflib.SH* attribute), 592
- intersectionOf (*rdflib.OWL* attribute), 465
- Interval (*rdflib.TIME* attribute), 600
- intervalAfter (*rdflib.TIME* attribute), 602
- intervalBefore (*rdflib.TIME* attribute), 602
- intervalContains (*rdflib.TIME* attribute), 602
- intervalDisjoint (*rdflib.TIME* attribute), 602
- intervalDuring (*rdflib.TIME* attribute), 602
- intervalEquals (*rdflib.TIME* attribute), 602
- intervalFinishedBy (*rdflib.TIME* attribute), 602
- intervalFinishes (*rdflib.TIME* attribute), 602
- intervalIn (*rdflib.TIME* attribute), 602
- intervalMeets (*rdflib.TIME* attribute), 602
- intervalMetBy (*rdflib.TIME* attribute), 603
- intervalOverlappedBy (*rdflib.TIME* attribute), 603
- intervalOverlaps (*rdflib.TIME* attribute), 603
- intervalStartedBy (*rdflib.TIME* attribute), 603
- intervalStarts (*rdflib.TIME* attribute), 603
- inTimePosition (*rdflib.TIME* attribute), 602
- Intrusion\_Detection\_Equipment (*rdflib.BRICK* attribute), 366
- inv\_path() (in module *rdflib.paths*), 287
- invalid (*rdflib.ODRL2* attribute), 457
- invalidated (*rdflib.PROV* attribute), 472
- invalidatedAtTime (*rdflib.PROV* attribute), 472
- Invalidation (*rdflib.PROV* attribute), 469
- inventoryLevel (*rdflib.SDO* attribute), 559
- inverse (*rdflib.PROV* attribute), 472
- InverseFunctionalProperty (*rdflib.OWL* attribute), 463
- inverseOf (*rdflib.extras.infixowl.Property* property), 67
- inverseOf (*rdflib.OWL* attribute), 465
- inverseOf (*rdflib.SDO* attribute), 559
- inversePath (*rdflib.SH* attribute), 592
- Inverter (*rdflib.BRICK* attribute), 366
- InvestmentFund (*rdflib.SDO* attribute), 500
- InvestmentOrDeposit (*rdflib.SDO* attribute), 500
- InviteAction (*rdflib.SDO* attribute), 500
- Invoice (*rdflib.SDO* attribute), 500
- InvoicePrice (*rdflib.SDO* attribute), 500
- InvPath (class in *rdflib.paths*), 283
- inXSDDate (*rdflib.TIME* attribute), 602
- inXSDDateTime (*rdflib.TIME* attribute), 602
- inXSDDateTimeStamp (*rdflib.TIME* attribute), 602
- inXSDgYear (*rdflib.TIME* attribute), 602
- inXSDgYearMonth (*rdflib.TIME* attribute), 602
- IRI (*rdflib.SH* attribute), 588
- IRIOrLiteral (*rdflib.SH* attribute), 588
- IRIREF (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* attribute), 212
- IrreflexiveProperty (*rdflib.OWL* attribute), 463
- is\_ncname() (in module *rdflib.namespace*), 78
- is\_open() (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* method), 194
- isA (*rdflib.ODRL2* attribute), 457
- isAcceptingNewPatients (*rdflib.SDO* attribute), 559
- isAccessibleForFree (*rdflib.SDO* attribute), 559
- isAccessoryOrSparePartFor (*rdflib.SDO* attribute), 559
- isActedOnBy (*rdflib.SOSA* attribute), 598

- `isAllOf` (*rdflib.ODRL2 attribute*), 457
- `isAnyOf` (*rdflib.ODRL2 attribute*), 457
- `isAssociatedWith` (*rdflib.BRICK attribute*), 398
- `isAvailableGenerically` (*rdflib.SDO attribute*), 559
- `isBasedOn` (*rdflib.SDO attribute*), 559
- `isBasedOnUrl` (*rdflib.SDO attribute*), 559
- `isblank()` (*rdflib.plugins.shared.jsonld.context.Context method*), 128
- `isbn` (*rdflib.SDO attribute*), 560
- `isCompatibleDateTimeDatatype()` (in module *rdflib.plugins.sparql.operators*), 166
- `isConsumableFor` (*rdflib.SDO attribute*), 559
- `isDefinedBy` (*rdflib.RDFS attribute*), 479
- `isDone()` (*rdflib.plugins.serializers.turtle.RecursiveSerializer method*), 121
- `isEncodedByBioChemEntity` (*rdflib.SDO attribute*), 559
- `isFamilyFriendly` (*rdflib.SDO attribute*), 559
- `isFeatureOfInterestOf` (*rdflib.SOSA attribute*), 598
- `isFedBy` (*rdflib.BRICK attribute*), 398
- `isFormatOf` (*rdflib.DCTERMS attribute*), 413
- `isGift` (*rdflib.SDO attribute*), 559
- `isHostedBy` (*rdflib.SOSA attribute*), 598
- `isicV4` (*rdflib.SDO attribute*), 560
- `isInheritedFrom` (*rdflib.PROF attribute*), 467
- `isInvolvedInBiologicalProcess` (*rdflib.SDO attribute*), 559
- `isLiveBroadcast` (*rdflib.SDO attribute*), 559
- `isLocatedInSubcellularLocation` (*rdflib.SDO attribute*), 559
- `isLocationOf` (*rdflib.BRICK attribute*), 399
- `isMeasuredBy` (*rdflib.BRICK attribute*), 399
- `isNoneOf` (*rdflib.ODRL2 attribute*), 457
- `ISO3166` (*rdflib.DCTERMS attribute*), 410
- `isObservedBy` (*rdflib.SOSA attribute*), 598
- `Isolation_Valve` (*rdflib.BRICK attribute*), 366
- `isomorphic()` (in module *rdflib.compare*), 228
- `isomorphic()` (*rdflib.Graph method*), 432
- `isomorphic()` (*rdflib.graph.Graph method*), 259
- `IsomorphicGraph` (class in *rdflib.compare*), 227
- `IsomorphicTestableGraph` (class in *rdflib.tools.graphisomorphism*), 220
- `isPartOf` (*rdflib.BRICK attribute*), 399
- `isPartOf` (*rdflib.DCTERMS attribute*), 413
- `isPartOf` (*rdflib.ODRL2 attribute*), 457
- `isPartOf` (*rdflib.SDO attribute*), 559
- `isPartOfBioChemEntity` (*rdflib.SDO attribute*), 559
- `isPlanForApartment` (*rdflib.SDO attribute*), 559
- `isPointOf` (*rdflib.BRICK attribute*), 399
- `isPrimaryTopicOf` (*rdflib.FOAF attribute*), 422
- `isPrimitive()` (*rdflib.extras.infixowl.BooleanClass method*), 59
- `isPrimitive()` (*rdflib.extras.infixowl.Class method*), 61
- `isPrimitive()` (*rdflib.extras.infixowl.EnumeratedClass method*), 64
- `isPrimitive()` (*rdflib.extras.infixowl.Restriction method*), 68
- `isProfileOf` (*rdflib.PROF attribute*), 467
- `isPropertyOf` (*rdflib.SSN attribute*), 600
- `isProprietary` (*rdflib.SDO attribute*), 559
- `isProxyFor` (*rdflib.SSN attribute*), 600
- `isrcCode` (*rdflib.SDO attribute*), 560
- `isReferencedBy` (*rdflib.DCTERMS attribute*), 413
- `isRegulatedBy` (*rdflib.BRICK attribute*), 399
- `isRelatedTo` (*rdflib.SDO attribute*), 560
- `isReplacedBy` (*rdflib.DCTERMS attribute*), 413
- `isRequiredBy` (*rdflib.DCTERMS attribute*), 413
- `isResizable` (*rdflib.SDO attribute*), 560
- `isResultOf` (*rdflib.SOSA attribute*), 598
- `isSampleOf` (*rdflib.SOSA attribute*), 598
- `isSimilarTo` (*rdflib.SDO attribute*), 560
- `issn` (*rdflib.SDO attribute*), 560
- `issued` (*rdflib.DCTERMS attribute*), 413
- `issuedBy` (*rdflib.SDO attribute*), 560
- `issuedThrough` (*rdflib.SDO attribute*), 560
- `issueNumber` (*rdflib.SDO attribute*), 560
- `isTagOf` (*rdflib.BRICK attribute*), 399
- `isTransitiveProfileOf` (*rdflib.PROF attribute*), 467
- `isUnlabelledFallback` (*rdflib.SDO attribute*), 560
- `isValidList()` (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer method*), 116
- `isValidList()` (*rdflib.plugins.serializers.turtle.TurtleSerializer method*), 122
- `isVariantOf` (*rdflib.SDO attribute*), 560
- `isVersionOf` (*rdflib.DCTERMS attribute*), 413
- `iswcCode` (*rdflib.SDO attribute*), 560
- `item` (*rdflib.SDO attribute*), 560
- `item()` (*rdflib.plugins.parsers.notation3.SinkParser method*), 90
- `ItemAvailability` (*rdflib.SDO attribute*), 500
- `itemCondition` (*rdflib.SDO attribute*), 560
- `itemDefectReturnFees` (*rdflib.SDO attribute*), 560
- `itemDefectReturnLabelSource` (*rdflib.SDO attribute*), 560
- `itemDefectReturnShippingFeesAmount` (*rdflib.SDO attribute*), 560
- `ItemList` (*rdflib.SDO attribute*), 500
- `itemListElement` (*rdflib.SDO attribute*), 560
- `itemListOrder` (*rdflib.SDO attribute*), 560
- `ItemListOrderAscending` (*rdflib.SDO attribute*), 500
- `ItemListOrderDescending` (*rdflib.SDO attribute*), 500
- `ItemListOrderType` (*rdflib.SDO attribute*), 500
- `ItemListUnordered` (*rdflib.SDO attribute*), 500
- `itemLocation` (*rdflib.SDO attribute*), 560
- `itemOffered` (*rdflib.SDO attribute*), 560
- `ItemPage` (*rdflib.SDO attribute*), 500
- `itemReviewed` (*rdflib.SDO attribute*), 560

items() (*rdflib.container.Container* method), 233  
 items() (*rdflib.Graph* method), 432  
 items() (*rdflib.graph.Graph* method), 259  
 items() (*rdflib.resource.Resource* method), 305  
 itemShipped (*rdflib.SDO* attribute), 560  
 itinerary (*rdflib.SDO* attribute), 560  
 iupacName (*rdflib.SDO* attribute), 560

## J

jabberID (*rdflib.FOAF* attribute), 422  
 Janitor\_Room (*rdflib.BRICK* attribute), 366  
 January (*rdflib.TIME* attribute), 600  
 Jet\_Nozzle\_Air\_Diffuser (*rdflib.BRICK* attribute), 366  
 JewelryStore (*rdflib.SDO* attribute), 500  
 jobBenefits (*rdflib.SDO* attribute), 560  
 jobImmediateStart (*rdflib.SDO* attribute), 560  
 jobLocation (*rdflib.SDO* attribute), 561  
 jobLocationType (*rdflib.SDO* attribute), 561  
 JobPosting (*rdflib.SDO* attribute), 500  
 jobStartDate (*rdflib.SDO* attribute), 561  
 jobTitle (*rdflib.SDO* attribute), 561  
 join() (in module *rdflib.plugins.parsers.notation3*), 95  
 Join() (in module *rdflib.plugins.sparql.algebra*), 146  
 JoinAction (*rdflib.SDO* attribute), 500  
 Joint (*rdflib.SDO* attribute), 501  
 js (*rdflib.SH* attribute), 593  
 JSConstraint (*rdflib.SH* attribute), 588  
 JSConstraintComponent (*rdflib.SH* attribute), 589  
 JSExecutable (*rdflib.SH* attribute), 589  
 JSFunction (*rdflib.SH* attribute), 589  
 jsFunctionName (*rdflib.SH* attribute), 593  
 JSLibrary (*rdflib.SH* attribute), 589  
 jsLibrary (*rdflib.SH* attribute), 593  
 jsLibraryURL (*rdflib.SH* attribute), 593  
 JSON (*rdflib.CSVW* attribute), 400  
 JSON (*rdflib.RDF* attribute), 477  
 JSONLDException, 130  
 JsonLDParse (class in *rdflib.plugins.parsers.jsonld*), 80  
 JsonLDSerializer (class in *rdflib.plugins.serializers.jsonld*), 114  
 JSONResult (class in *rdflib.plugins.sparql.results.jsonresults*), 133  
 JSONResultParser (class in *rdflib.plugins.sparql.results.jsonresults*), 133  
 JSONResultSerializer (class in *rdflib.plugins.sparql.results.jsonresults*), 134  
 JSRule (*rdflib.SH* attribute), 589  
 JSTarget (*rdflib.SH* attribute), 589  
 JSTargetType (*rdflib.SH* attribute), 589  
 JSValidator (*rdflib.SH* attribute), 589  
 Jurisdiction (*rdflib.DCTERMS* attribute), 410  
 jurisdiction (*rdflib.SDO* attribute), 561

## K

KeyEntityPair (*rdflib.PROV* attribute), 469  
 keyword (*rdflib.DCAT* attribute), 409  
 keywords (*rdflib.SDO* attribute), 561  
 knownVehicleDamages (*rdflib.SDO* attribute), 561  
 knows (*rdflib.FOAF* attribute), 422  
 knows (*rdflib.SDO* attribute), 561  
 knowsAbout (*rdflib.SDO* attribute), 561  
 knowsLanguage (*rdflib.SDO* attribute), 561  
 KosherDiet (*rdflib.SDO* attribute), 501

## L

label (*rdflib.extras.infixowl.AnnotatableTerms* property), 58  
 label (*rdflib.RDFS* attribute), 479  
 label() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 116  
 label() (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 122  
 labelDetails (*rdflib.SDO* attribute), 561  
 labelOrSubject() (*rdflib.plugins.parsers.trig.TrigSinkParser* method), 109  
 LabelProperty (*rdflib.FOAF* attribute), 420  
 labels (*rdflib.query.ResultRow* attribute), 296  
 labelTemplate (*rdflib.SH* attribute), 593  
 Laboratory (*rdflib.BRICK* attribute), 366  
 LaboratoryScience (*rdflib.SDO* attribute), 501  
 LakeBodyOfWater (*rdflib.SDO* attribute), 501  
 Laminar\_Flow\_Air\_Diffuser (*rdflib.BRICK* attribute), 366  
 Landform (*rdflib.SDO* attribute), 501  
 landingPage (*rdflib.DCAT* attribute), 409  
 landlord (*rdflib.SDO* attribute), 561  
 LandmarksOrHistoricalBuildings (*rdflib.SDO* attribute), 501  
 lang (*rdflib.CSVW* attribute), 401  
 lang\_key (*rdflib.plugins.shared.jsonld.context.Context* property), 128  
 langString (*rdflib.RDF* attribute), 478  
 language (*rdflib.DC* attribute), 407  
 language (*rdflib.DCTERMS* attribute), 413  
 language (*rdflib.DOAP* attribute), 415  
 language (*rdflib.Literal* property), 449  
 language (*rdflib.ODRL2* attribute), 457  
 language (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 102  
 language (*rdflib.plugins.shared.jsonld.context.Term* attribute), 130  
 language (*rdflib.RDF* attribute), 478  
 Language (*rdflib.SDO* attribute), 501  
 language (*rdflib.SDO* attribute), 561  
 language (*rdflib.term.Literal* property), 325  
 language (*rdflib.XSD* attribute), 610



- languageIn (*rdflib.SH* attribute), 593
- LanguageInConstraintComponent (*rdflib.SH* attribute), 589
- LaserDiscFormat (*rdflib.SDO* attribute), 501
- Last\_Fault\_Code\_Status (*rdflib.BRICK* attribute), 366
- lastName (*rdflib.FOAF* attribute), 422
- lastReviewed (*rdflib.SDO* attribute), 561
- latitude (*rdflib.BRICK* attribute), 399
- latitude (*rdflib.SDO* attribute), 561
- layoutImage (*rdflib.SDO* attribute), 561
- LCC (*rdflib.DCTERMS* attribute), 410
- LCSH (*rdflib.DCTERMS* attribute), 410
- Lead\_Lag\_Command (*rdflib.BRICK* attribute), 366
- Lead\_Lag\_Status (*rdflib.BRICK* attribute), 366
- Lead\_On\_Off\_Command (*rdflib.BRICK* attribute), 366
- Leak\_Alarm (*rdflib.BRICK* attribute), 366
- LearningResource (*rdflib.SDO* attribute), 501
- learningResourceType (*rdflib.SDO* attribute), 561
- lease (*rdflib.ODRL2* attribute), 458
- leaseLength (*rdflib.SDO* attribute), 561
- LeaveAction (*rdflib.SDO* attribute), 501
- Leaving\_Water (*rdflib.BRICK* attribute), 366
- Leaving\_Water\_Flow\_Sensor (*rdflib.BRICK* attribute), 366
- Leaving\_Water\_Flow\_Setpoint (*rdflib.BRICK* attribute), 367
- Leaving\_Water\_Temperature\_Sensor (*rdflib.BRICK* attribute), 367
- Leaving\_Water\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 367
- LeftHandDriving (*rdflib.SDO* attribute), 501
- LeftJoin() (in module *rdflib.plugins.sparql.algebra*), 146
- LeftOperand (*rdflib.ODRL2* attribute), 454
- leftOperand (*rdflib.ODRL2* attribute), 458
- LegalForceStatus (*rdflib.SDO* attribute), 501
- legalName (*rdflib.SDO* attribute), 561
- LegalService (*rdflib.SDO* attribute), 501
- legalStatus (*rdflib.SDO* attribute), 561
- LegalValueLevel (*rdflib.SDO* attribute), 501
- Legislation (*rdflib.SDO* attribute), 501
- legislationApplies (*rdflib.SDO* attribute), 561
- legislationChanges (*rdflib.SDO* attribute), 561
- legislationConsolidates (*rdflib.SDO* attribute), 561
- legislationDate (*rdflib.SDO* attribute), 561
- legislationDateVersion (*rdflib.SDO* attribute), 561
- legislationIdentifier (*rdflib.SDO* attribute), 561
- legislationJurisdiction (*rdflib.SDO* attribute), 562
- legislationLegalForce (*rdflib.SDO* attribute), 562
- legislationLegalValue (*rdflib.SDO* attribute), 562
- LegislationObject (*rdflib.SDO* attribute), 501
- legislationPassedBy (*rdflib.SDO* attribute), 562
- legislationResponsible (*rdflib.SDO* attribute), 562
- legislationTransposes (*rdflib.SDO* attribute), 562
- legislationType (*rdflib.SDO* attribute), 562
- LegislativeBuilding (*rdflib.SDO* attribute), 501
- leiCode (*rdflib.SDO* attribute), 562
- LeisureTimeActivity (*rdflib.SDO* attribute), 501
- lend (*rdflib.ODRL2* attribute), 458
- LendAction (*rdflib.SDO* attribute), 501
- lender (*rdflib.SDO* attribute), 562
- length (*rdflib.CSVW* attribute), 401
- length (*rdflib.XSD* attribute), 610
- lesser (*rdflib.SDO* attribute), 562
- lesserOrEqual (*rdflib.SDO* attribute), 562
- lessThan (*rdflib.SH* attribute), 593
- LessThanConstraintComponent (*rdflib.SH* attribute), 589
- lessThanOrEquals (*rdflib.SH* attribute), 593
- LessThanOrEqualsConstraintComponent (*rdflib.SH* attribute), 589
- letterer (*rdflib.SDO* attribute), 562
- li (*rdflib.plugins.parsers.RDFVOC.RDFVOC* attribute), 79
- li (*rdflib.plugins.parsers.rdfxml.BagID* attribute), 102
- li (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 102
- Library (*rdflib.BRICK* attribute), 367
- Library (*rdflib.SDO* attribute), 501
- LibrarySystem (*rdflib.SDO* attribute), 501
- license (*rdflib.DCTERMS* attribute), 413
- license (*rdflib.DOAP* attribute), 415
- license (*rdflib.ODRL2* attribute), 458
- license (*rdflib.SDO* attribute), 562
- LicenseDocument (*rdflib.DCTERMS* attribute), 410
- LifestyleModification (*rdflib.SDO* attribute), 501
- Ligament (*rdflib.SDO* attribute), 501
- Lighting (*rdflib.BRICK* attribute), 367
- Lighting\_Equipment (*rdflib.BRICK* attribute), 367
- Lighting\_System (*rdflib.BRICK* attribute), 367
- Lighting\_Zone (*rdflib.BRICK* attribute), 367
- LikeAction (*rdflib.SDO* attribute), 501
- Limit (*rdflib.BRICK* attribute), 367
- LimitedAvailability (*rdflib.SDO* attribute), 501
- LimitedByGuaranteeCharity (*rdflib.SDO* attribute), 502
- line (*rdflib.plugins.parsers.nquads.NQuadsParser* attribute), 98
- line (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* attribute), 100
- line (*rdflib.SDO* attribute), 562
- lineTerminators (*rdflib.CSVW* attribute), 401
- LinguisticSystem (*rdflib.DCTERMS* attribute), 410
- linkedTo (*rdflib.ORG* attribute), 462
- linkPredicate (*rdflib.VOID* attribute), 607
- linkRelationship (*rdflib.SDO* attribute), 562
- LinkRole (*rdflib.SDO* attribute), 502

- links (*rdflib.parser.URLInputSource* attribute), 279
- Linkset (*rdflib.VOID* attribute), 607
- Liquid (*rdflib.BRICK* attribute), 367
- Liquid\_CO2 (*rdflib.BRICK* attribute), 367
- Liquid\_Detection\_Alarm (*rdflib.BRICK* attribute), 367
- LiquorStore (*rdflib.SDO* attribute), 502
- list (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 103
- List (*rdflib.RDF* attribute), 477
- list2set() (in module *rdflib.util*), 332
- list\_key (*rdflib.plugins.shared.jsonld.context.Context* property), 128
- list\_node\_element\_end() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 105
- ListenAction (*rdflib.SDO* attribute), 502
- ListItem (*rdflib.SDO* attribute), 502
- ListPrice (*rdflib.SDO* attribute), 502
- Literal (class in *rdflib*), 442
- Literal (class in *rdflib.term*), 317
- Literal (*rdflib.RDFS* attribute), 479
- Literal (*rdflib.SH* attribute), 589
- literal() (in module *rdflib.plugins.sparql.operators*), 167
- literal() (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* method), 100
- literal\_element\_char() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 105
- literal\_element\_end() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 105
- literal\_element\_start() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 105
- LiteraryEvent (*rdflib.SDO* attribute), 502
- LiveAlbum (*rdflib.SDO* attribute), 502
- LiveBlogPosting (*rdflib.SDO* attribute), 502
- liveBlogUpdate (*rdflib.SDO* attribute), 562
- LivingWithHealthAspect (*rdflib.SDO* attribute), 502
- load() (*rdflib.plugins.shared.jsonld.context.Context* method), 128
- load() (*rdflib.plugins.sparql.sparql.QueryContext* method), 183
- Load\_Current\_Sensor (*rdflib.BRICK* attribute), 367
- Load\_Parameter (*rdflib.BRICK* attribute), 367
- Load\_Setpoint (*rdflib.BRICK* attribute), 367
- Load\_Shed\_Command (*rdflib.BRICK* attribute), 367
- Load\_Shed\_Differential\_Pressure\_Setpoint (*rdflib.BRICK* attribute), 367
- Load\_Shed\_Setpoint (*rdflib.BRICK* attribute), 367
- Load\_Shed\_Status (*rdflib.BRICK* attribute), 367
- loadBuf() (*rdflib.plugins.parsers.notation3.SinkParser* method), 90
- Loading\_Dock (*rdflib.BRICK* attribute), 367
- loads() (*rdflib.store.NodePickler* method), 307
- loadStream() (*rdflib.plugins.parsers.notation3.SinkParser* method), 90
- loanMortgageMandateAmount (*rdflib.SDO* attribute), 562
- LoanOrCredit (*rdflib.SDO* attribute), 502
- loanPaymentAmount (*rdflib.SDO* attribute), 562
- loanPaymentFrequency (*rdflib.SDO* attribute), 562
- loanRepaymentForm (*rdflib.SDO* attribute), 562
- loanTerm (*rdflib.SDO* attribute), 562
- loanType (*rdflib.SDO* attribute), 562
- Lobby (*rdflib.BRICK* attribute), 368
- LocalBusiness (*rdflib.SDO* attribute), 502
- Locally\_On\_Off\_Status (*rdflib.BRICK* attribute), 368
- Location (*rdflib.BRICK* attribute), 368
- Location (*rdflib.DCTERMS* attribute), 411
- location (*rdflib.DOAP* attribute), 415
- location (*rdflib.ORG* attribute), 462
- Location (*rdflib.PROV* attribute), 469
- location (*rdflib.SDO* attribute), 562
- locationCreated (*rdflib.SDO* attribute), 563
- LocationFeatureSpecification (*rdflib.SDO* attribute), 502
- locationOf (*rdflib.PROV* attribute), 472
- LocationPeriodOrJurisdiction (*rdflib.DCTERMS* attribute), 411
- LockerDelivery (*rdflib.SDO* attribute), 502
- Lockout\_Status (*rdflib.BRICK* attribute), 368
- Lockout\_Temperature\_Differential\_Parameter (*rdflib.BRICK* attribute), 368
- Locksmith (*rdflib.SDO* attribute), 502
- LodgingBusiness (*rdflib.SDO* attribute), 502
- LodgingReservation (*rdflib.SDO* attribute), 502
- lodgingUnitDescription (*rdflib.SDO* attribute), 563
- lodgingUnitType (*rdflib.SDO* attribute), 563
- log (in module *rdflib.plugins.sparql.results.xmlresults*), 139
- LogicalConstraint (*rdflib.ODRL2* attribute), 454
- logo (*rdflib.FOAF* attribute), 422
- logo (*rdflib.SDO* attribute), 563
- long (*rdflib.XSD* attribute), 610
- longitude (*rdflib.BRICK* attribute), 399
- longitude (*rdflib.SDO* attribute), 563
- Longitudinal (*rdflib.SDO* attribute), 502
- LongTurtleSerializer (class in *rdflib.plugins.serializers.longturtle*), 115
- Loop (*rdflib.BRICK* attribute), 368
- LoseAction (*rdflib.SDO* attribute), 502
- loser (*rdflib.SDO* attribute), 563
- Lounge (*rdflib.BRICK* attribute), 368
- Louver (*rdflib.BRICK* attribute), 368



- Low\_Freeze\_Protect\_Temperature\_Parameter (*rdflib.BRICK* attribute), 368
- Low\_Humidity\_Alarm (*rdflib.BRICK* attribute), 368
- Low\_Humidity\_Alarm\_Parameter (*rdflib.BRICK* attribute), 368
- Low\_Outside\_Air\_Lockout\_Temperature\_Differential\_Parameter (*rdflib.BRICK* attribute), 368
- Low\_Outside\_Air\_Temperature\_Enable\_Differential\_Sensor (*rdflib.BRICK* attribute), 368
- Low\_Outside\_Air\_Temperature\_Enable\_Setpoint (*rdflib.BRICK* attribute), 368
- Low\_Return\_Air\_Temperature\_Alarm (*rdflib.BRICK* attribute), 368
- Low\_Suction\_Pressure\_Alarm (*rdflib.BRICK* attribute), 368
- Low\_Temperature\_Alarm (*rdflib.BRICK* attribute), 368
- Low\_Temperature\_Alarm\_Parameter (*rdflib.BRICK* attribute), 368
- LowCalorieDiet (*rdflib.SDO* attribute), 502
- Lowest\_Exhaust\_Air\_Static\_Pressure\_Sensor (*rdflib.BRICK* attribute), 368
- LowFatDiet (*rdflib.SDO* attribute), 502
- LowLactoseDiet (*rdflib.SDO* attribute), 502
- lowPrice (*rdflib.SDO* attribute), 563
- LowSaltDiet (*rdflib.SDO* attribute), 502
- lt (*rdflib.ODRL2* attribute), 458
- lteq (*rdflib.ODRL2* attribute), 458
- ltr (*rdflib.CSVW* attribute), 402
- Luminaire (*rdflib.BRICK* attribute), 368
- Luminaire\_Driver (*rdflib.BRICK* attribute), 369
- Luminance\_Alarm (*rdflib.BRICK* attribute), 369
- Luminance\_Command (*rdflib.BRICK* attribute), 369
- Luminance\_Sensor (*rdflib.BRICK* attribute), 369
- Luminance\_Setpoint (*rdflib.BRICK* attribute), 369
- Lung (*rdflib.SDO* attribute), 502
- LymphaticVessel (*rdflib.SDO* attribute), 502
- lyricist (*rdflib.SDO* attribute), 563
- lyrics (*rdflib.SDO* attribute), 563
- ## M
- made (*rdflib.FOAF* attribute), 422
- madeActuation (*rdflib.SOSA* attribute), 598
- madeByActuator (*rdflib.SOSA* attribute), 598
- madeBySampler (*rdflib.SOSA* attribute), 598
- madeBySensor (*rdflib.SOSA* attribute), 598
- madeObservation (*rdflib.SOSA* attribute), 598
- madeSampling (*rdflib.SOSA* attribute), 598
- Mail\_Room (*rdflib.BRICK* attribute), 369
- main() (in module *examples.secure\_with\_audit*), 26
- main() (in module *examples.secure\_with\_urlopen*), 26
- main() (in module *rdflib.extras.cmdlineutils*), 47
- main() (in module *rdflib.tools.graphisomorphism*), 220
- main() (in module *rdflib.tools.rdf2dot*), 221
- main() (in module *rdflib.tools.rdfpipe*), 221
- main() (in module *rdflib.tools.rdfs2dot*), 221
- mainContentOfPage (*rdflib.SDO* attribute), 563
- mainEntity (*rdflib.SDO* attribute), 563
- mainEntityOfPage (*rdflib.SDO* attribute), 563
- maintainer (*rdflib.DOAP* attribute), 415
- mainParameter (*rdflib.SDO* attribute), 563
- Maintenance\_Mode\_Command (*rdflib.BRICK* attribute), 369
- Maintenance\_Required\_Alarm (*rdflib.BRICK* attribute), 369
- Majlis (*rdflib.BRICK* attribute), 369
- make\_dn\_file() (in module *rdflib.tools.defined\_namespace\_creator*), 219
- make\_option\_parser() (in module *rdflib.tools.rdfpipe*), 221
- maker (*rdflib.FOAF* attribute), 422
- makesOffer (*rdflib.SDO* attribute), 563
- makeStatement() (*rdflib.plugins.parsers.notation3.RDFSink* method), 85
- makeStatement() (*rdflib.plugins.parsers.notation3.SinkParser* method), 90
- Makeup\_Air\_Unit (*rdflib.BRICK* attribute), 369
- Makeup\_Water (*rdflib.BRICK* attribute), 369
- Makeup\_Water\_Valve (*rdflib.BRICK* attribute), 369
- Male (*rdflib.SDO* attribute), 502
- MalformedClass, 65
- MalformedClassError, 66
- manchesterSyntax() (in module *rdflib.extras.infixowl*), 69
- Manual\_Auto\_Status (*rdflib.BRICK* attribute), 369
- manufacturer (*rdflib.SDO* attribute), 563
- Manuscript (*rdflib.SDO* attribute), 503
- Map (*rdflib.SDO* attribute), 503
- map (*rdflib.SDO* attribute), 563
- MapCategoryType (*rdflib.SDO* attribute), 503
- mappingRelation (*rdflib.SKOS* attribute), 596
- maps (*rdflib.SDO* attribute), 563
- mapType (*rdflib.SDO* attribute), 563
- marginOfError (*rdflib.SDO* attribute), 563
- MarryAction (*rdflib.SDO* attribute), 503
- Mass (*rdflib.SDO* attribute), 503
- Massage\_Room (*rdflib.BRICK* attribute), 369
- masthead (*rdflib.SDO* attribute), 563
- material (*rdflib.SDO* attribute), 563
- materialExtent (*rdflib.SDO* attribute), 563
- mathExpression (*rdflib.SDO* attribute), 563
- MathSolver (*rdflib.SDO* attribute), 503
- MAU (*rdflib.BRICK* attribute), 369
- Max\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 369
- Max\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 369

Max_Chilled_Water_Differential_Pressure_Setpoint_Limit (rdflib.BRICK attribute), 369	maxDepth (rdflib.plugins.serializers.turtle.RecursiveSerializer attribute), 121
Max_Cooling_Discharge_Air_Flow_Setpoint_Limit (rdflib.BRICK attribute), 369	maxExclusive (rdflib.CSVW attribute), 402
Max_Cooling_Supply_Air_Flow_Setpoint_Limit (rdflib.BRICK attribute), 370	maxExclusive (rdflib.SH attribute), 593
Max_Discharge_Air_Static_Pressure_Setpoint_Limit (rdflib.BRICK attribute), 370	maxExclusive (rdflib.XSD attribute), 610
Max_Discharge_Air_Temperature_Setpoint_Limit (rdflib.BRICK attribute), 370	MaxExclusiveConstraintComponent (rdflib.SH attribute), 589
Max_Frequency_Command (rdflib.BRICK attribute), 370	Maximum (class in rdflib.plugins.sparql.aggregates), 143
Max_Heating_Discharge_Air_Flow_Setpoint_Limit (rdflib.BRICK attribute), 370	maximumAttendeeCapacity (rdflib.SDO attribute), 563
Max_Heating_Supply_Air_Flow_Setpoint_Limit (rdflib.BRICK attribute), 370	MaximumDoseSchedule (rdflib.SDO attribute), 503
Max_Hot_Water_Differential_Pressure_Setpoint_Limit (rdflib.BRICK attribute), 370	maximumEnrollment (rdflib.SDO attribute), 563
Max_Limit (rdflib.BRICK attribute), 370	maximumIntake (rdflib.SDO attribute), 563
Max_Load_Setpoint (rdflib.BRICK attribute), 370	maximumPhysicalAttendeeCapacity (rdflib.SDO attribute), 564
Max_Occupied_Cooling_Discharge_Air_Flow_Setpoint (rdflib.BRICK attribute), 370	maximumVirtualAttendeeCapacity (rdflib.SDO attribute), 564
Max_Occupied_Cooling_Supply_Air_Flow_Setpoint (rdflib.BRICK attribute), 370	maxInclusive (rdflib.CSVW attribute), 402
Max_Occupied_Heating_Discharge_Air_Flow_Setpoint (rdflib.BRICK attribute), 370	maxInclusive (rdflib.SH attribute), 593
Max_Occupied_Heating_Supply_Air_Flow_Setpoint_Limit (rdflib.BRICK attribute), 370	maxInclusive (rdflib.XSD attribute), 610
Max_Position_Setpoint_Limit (rdflib.BRICK attribute), 370	MaxInclusiveConstraintComponent (rdflib.SH attribute), 589
Max_Speed_Setpoint_Limit (rdflib.BRICK attribute), 370	maxLength (rdflib.CSVW attribute), 402
Max_Static_Pressure_Setpoint_Limit (rdflib.BRICK attribute), 370	maxLength (rdflib.SH attribute), 593
Max_Supply_Air_Static_Pressure_Setpoint_Limit (rdflib.BRICK attribute), 370	maxLength (rdflib.XSD attribute), 610
Max_Temperature_Setpoint_Limit (rdflib.BRICK attribute), 371	MaxLengthConstraintComponent (rdflib.SH attribute), 589
Max_Unoccupied_Cooling_Discharge_Air_Flow_Setpoint (rdflib.BRICK attribute), 371	maxPrice (rdflib.SDO attribute), 563
Max_Unoccupied_Cooling_Supply_Air_Flow_Setpoint (rdflib.BRICK attribute), 371	maxQualifiedCardinality (rdflib.OWL attribute), 465
Max_Unoccupied_Heating_Discharge_Air_Flow_Setpoint (rdflib.BRICK attribute), 371	maxValue (rdflib.SDO attribute), 563
Max_Unoccupied_Heating_Supply_Air_Flow_Setpoint (rdflib.BRICK attribute), 371	MayTreatHealthAspect (rdflib.SDO attribute), 503
Max_Water_Level_Alarm (rdflib.BRICK attribute), 371	mbox (rdflib.FOAF attribute), 422
Max_Water_Temperature_Setpoint (rdflib.BRICK attribute), 371	mbox_sha1sum (rdflib.FOAF attribute), 422
maxCardinality (rdflib.extras.infixowl.Restriction property), 68	MDF (rdflib.BRICK attribute), 369
maxCardinality (rdflib.OWL attribute), 465	mealService (rdflib.SDO attribute), 564
maxCount (rdflib.SH attribute), 593	Measurable (rdflib.BRICK attribute), 371
MaxCountConstraintComponent (rdflib.SH attribute), 589	measure (rdflib.QB attribute), 477
	measureDimension (rdflib.QB attribute), 477
	measureTime (rdflib.SDO attribute), 564
	MeasurementModuleConversionEfficiency (rdflib.BRICK attribute), 399
	measuredPowerOutput (rdflib.BRICK attribute), 399
	measuredProperty (rdflib.SDO attribute), 564
	measuredValue (rdflib.SDO attribute), 564
	measurementTechnique (rdflib.SDO attribute), 564
	MeasurementTypeEnumeration (rdflib.SDO attribute), 503
	MeasureProperty (rdflib.QB attribute), 476
	measures (rdflib.BRICK attribute), 399
	measureType (rdflib.QB attribute), 477
	Mechanical_Room (rdflib.BRICK attribute), 371
	mechanismOfAction (rdflib.SDO attribute), 564
	media (rdflib.ODRL2 attribute), 458
	Media_Hot_Desk (rdflib.BRICK attribute), 371
	Media_Production_Room (rdflib.BRICK attribute), 371
	Media_Room (rdflib.BRICK attribute), 371

- mediaAuthenticityCategory (*rdflib.SDO attribute*), 564
- MediaGallery (*rdflib.SDO attribute*), 503
- mediaItemAppearance (*rdflib.SDO attribute*), 564
- MediaManipulationRatingEnumeration (*rdflib.SDO attribute*), 503
- median (*rdflib.SDO attribute*), 564
- MediaObject (*rdflib.SDO attribute*), 503
- MediaReview (*rdflib.SDO attribute*), 503
- MediaReviewItem (*rdflib.SDO attribute*), 503
- MediaSubscription (*rdflib.SDO attribute*), 503
- mediator (*rdflib.DCTERMS attribute*), 413
- mediaType (*rdflib.DCAT attribute*), 409
- MediaType (*rdflib.DCTERMS attribute*), 411
- MediaTypeOrExtent (*rdflib.DCTERMS attribute*), 411
- Medical\_Room (*rdflib.BRICK attribute*), 371
- MedicalAudience (*rdflib.SDO attribute*), 503
- medicalAudience (*rdflib.SDO attribute*), 564
- MedicalAudienceType (*rdflib.SDO attribute*), 503
- MedicalBusiness (*rdflib.SDO attribute*), 503
- MedicalCause (*rdflib.SDO attribute*), 503
- MedicalClinic (*rdflib.SDO attribute*), 503
- MedicalCode (*rdflib.SDO attribute*), 503
- MedicalCondition (*rdflib.SDO attribute*), 503
- MedicalConditionStage (*rdflib.SDO attribute*), 503
- MedicalContraindication (*rdflib.SDO attribute*), 503
- MedicalDevice (*rdflib.SDO attribute*), 503
- MedicalDevicePurpose (*rdflib.SDO attribute*), 504
- MedicalEntity (*rdflib.SDO attribute*), 504
- MedicalEnumeration (*rdflib.SDO attribute*), 504
- MedicalEvidenceLevel (*rdflib.SDO attribute*), 504
- MedicalGuideline (*rdflib.SDO attribute*), 504
- MedicalGuidelineContraindication (*rdflib.SDO attribute*), 504
- MedicalGuidelineRecommendation (*rdflib.SDO attribute*), 504
- MedicalImagingTechnique (*rdflib.SDO attribute*), 504
- MedicalIndication (*rdflib.SDO attribute*), 504
- MedicalIntangible (*rdflib.SDO attribute*), 504
- MedicalObservationalStudy (*rdflib.SDO attribute*), 504
- MedicalObservationalStudyDesign (*rdflib.SDO attribute*), 504
- MedicalOrganization (*rdflib.SDO attribute*), 504
- MedicalProcedure (*rdflib.SDO attribute*), 504
- MedicalProcedureType (*rdflib.SDO attribute*), 504
- MedicalResearcher (*rdflib.SDO attribute*), 504
- MedicalRiskCalculator (*rdflib.SDO attribute*), 504
- MedicalRiskEstimator (*rdflib.SDO attribute*), 504
- MedicalRiskFactor (*rdflib.SDO attribute*), 504
- MedicalRiskScore (*rdflib.SDO attribute*), 504
- MedicalScholarlyArticle (*rdflib.SDO attribute*), 505
- MedicalSign (*rdflib.SDO attribute*), 505
- MedicalSignOrSymptom (*rdflib.SDO attribute*), 505
- MedicalSpecialty (*rdflib.SDO attribute*), 505
- medicalSpecialty (*rdflib.SDO attribute*), 564
- MedicalStudy (*rdflib.SDO attribute*), 505
- MedicalStudyStatus (*rdflib.SDO attribute*), 505
- MedicalSymptom (*rdflib.SDO attribute*), 505
- MedicalTest (*rdflib.SDO attribute*), 505
- MedicalTestPanel (*rdflib.SDO attribute*), 505
- MedicalTherapy (*rdflib.SDO attribute*), 505
- MedicalTrial (*rdflib.SDO attribute*), 505
- MedicalTrialDesign (*rdflib.SDO attribute*), 505
- MedicalWebPage (*rdflib.SDO attribute*), 505
- MedicineSystem (*rdflib.SDO attribute*), 505
- medicineSystem (*rdflib.SDO attribute*), 564
- medium (*rdflib.DCTERMS attribute*), 413
- Medium\_Temperature\_Hot\_Water\_Differential\_Pressure\_Load\_Shedding (*rdflib.BRICK attribute*), 371
- Medium\_Temperature\_Hot\_Water\_Differential\_Pressure\_Load\_SheddingRate (*rdflib.BRICK attribute*), 371
- Medium\_Temperature\_Hot\_Water\_Differential\_Pressure\_Load\_SheddingRateSetpoint (*rdflib.BRICK attribute*), 371
- Medium\_Temperature\_Hot\_Water\_Differential\_Pressure\_Sensor (*rdflib.BRICK attribute*), 372
- Medium\_Temperature\_Hot\_Water\_Differential\_Pressure\_Setpoint (*rdflib.BRICK attribute*), 372
- Medium\_Temperature\_Hot\_Water\_Discharge\_Temperature\_High\_Reset (*rdflib.BRICK attribute*), 372
- Medium\_Temperature\_Hot\_Water\_Discharge\_Temperature\_Low\_Reset (*rdflib.BRICK attribute*), 372
- Medium\_Temperature\_Hot\_Water\_Return\_Temperature\_Sensor (*rdflib.BRICK attribute*), 372
- Medium\_Temperature\_Hot\_Water\_Supply\_Temperature\_High\_Reset (*rdflib.BRICK attribute*), 372
- Medium\_Temperature\_Hot\_Water\_Supply\_Temperature\_Load\_Shedding (*rdflib.BRICK attribute*), 372
- Medium\_Temperature\_Hot\_Water\_Supply\_Temperature\_Load\_SheddingRate (*rdflib.BRICK attribute*), 372
- Medium\_Temperature\_Hot\_Water\_Supply\_Temperature\_Low\_Reset (*rdflib.BRICK attribute*), 372
- Medium\_Temperature\_Hot\_Water\_Supply\_Temperature\_Sensor (*rdflib.BRICK attribute*), 372
- MeetingRoom (*rdflib.SDO attribute*), 505
- meetsEmissionStandard (*rdflib.SDO attribute*), 564
- member (*rdflib.FOAF attribute*), 422
- member (*rdflib.ORG attribute*), 462
- member (*rdflib.RDFS attribute*), 479
- member (*rdflib.SDO attribute*), 564
- member (*rdflib.SKOS attribute*), 596
- memberDuring (*rdflib.ORG attribute*), 462
- memberList (*rdflib.SKOS attribute*), 596
- memberOf (*rdflib.ORG attribute*), 462
- memberOf (*rdflib.SDO attribute*), 564
- members (*rdflib.OWL attribute*), 465
- members (*rdflib.SDO attribute*), 564
- Membership (*rdflib.ORG attribute*), 461



- membershipClass (*rdflib.FOAF* attribute), 422
- membershipNumber (*rdflib.SDO* attribute), 564
- membershipPointsEarned (*rdflib.SDO* attribute), 564
- Memory (*class in rdflib.plugins.stores.memory*), 196
- memoryRequirements (*rdflib.SDO* attribute), 564
- MensClothingStore (*rdflib.SDO* attribute), 505
- mentionOf (*rdflib.PROV* attribute), 472
- mentions (*rdflib.SDO* attribute), 564
- Menu (*rdflib.SDO* attribute), 505
- menu (*rdflib.SDO* attribute), 564
- menuAddOn (*rdflib.SDO* attribute), 564
- MenuItem (*rdflib.SDO* attribute), 505
- MenuSection (*rdflib.SDO* attribute), 505
- merchant (*rdflib.SDO* attribute), 565
- merchantReturnDays (*rdflib.SDO* attribute), 565
- MerchantReturnEnumeration (*rdflib.SDO* attribute), 505
- MerchantReturnFiniteReturnWindow (*rdflib.SDO* attribute), 505
- merchantReturnLink (*rdflib.SDO* attribute), 565
- MerchantReturnNotPermitted (*rdflib.SDO* attribute), 505
- MerchantReturnPolicy (*rdflib.SDO* attribute), 505
- MerchantReturnPolicySeasonalOverride (*rdflib.SDO* attribute), 506
- MerchantReturnUnlimitedWindow (*rdflib.SDO* attribute), 506
- MerchantReturnUnspecified (*rdflib.SDO* attribute), 506
- merge() (*rdflib.plugins.sparql.sparql.FrozenBindings* method), 178
- merge() (*rdflib.plugins.sparql.sparql.FrozenDict* method), 180
- MESH (*rdflib.DCTERMS* attribute), 411
- message (*rdflib.plugins.parsers.notation3.BadSyntax* property), 82
- Message (*rdflib.SDO* attribute), 506
- message (*rdflib.SH* attribute), 593
- messageAttachment (*rdflib.SDO* attribute), 565
- Meter (*rdflib.BRICK* attribute), 372
- meteredTime (*rdflib.ODRL2* attribute), 458
- Methane\_Level\_Sensor (*rdflib.BRICK* attribute), 372
- method (*rdflib.plugins.stores.sparqlconnector.SPARQLConnector* property), 204
- MethodOfAccrual (*rdflib.DCTERMS* attribute), 411
- MethodOfInstruction (*rdflib.DCTERMS* attribute), 411
- MiddleSchool (*rdflib.SDO* attribute), 506
- Midwifery (*rdflib.SDO* attribute), 506
- mileageFromOdometer (*rdflib.SDO* attribute), 565
- Min\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 372
- Min\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 372
- Min\_Chilled\_Water\_Differential\_Pressure\_Setpoint\_Limit (*rdflib.BRICK* attribute), 372
- Min\_Cooling\_Discharge\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Cooling\_Supply\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Discharge\_Air\_Static\_Pressure\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Discharge\_Air\_Temperature\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Fresh\_Air\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Heating\_Discharge\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Heating\_Supply\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Hot\_Water\_Differential\_Pressure\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Occupied\_Cooling\_Discharge\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Occupied\_Cooling\_Supply\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Occupied\_Heating\_Discharge\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Occupied\_Heating\_Supply\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Outside\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Position\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Speed\_Setpoint\_Limit (*rdflib.BRICK* attribute), 373
- Min\_Static\_Pressure\_Setpoint\_Limit (*rdflib.BRICK* attribute), 374
- Min\_Supply\_Air\_Static\_Pressure\_Setpoint\_Limit (*rdflib.BRICK* attribute), 374
- Min\_Temperature\_Setpoint\_Limit (*rdflib.BRICK* attribute), 374
- Min\_Unoccupied\_Cooling\_Discharge\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 374
- Min\_Unoccupied\_Cooling\_Supply\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 374
- Min\_Unoccupied\_Heating\_Discharge\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 374
- Min\_Unoccupied\_Heating\_Supply\_Air\_Flow\_Setpoint\_Limit (*rdflib.BRICK* attribute), 374
- Min\_Water\_Level\_Alarm (*rdflib.BRICK* attribute), 374
- Min\_Water\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 374
- minCardinality (*rdflib.extras.infixowl.Restriction* property), 68
- minCardinality (*rdflib.OWL* attribute), 465
- minCount (*rdflib.SH* attribute), 593

- MinCountConstraintComponent (*rdflib.SH* attribute), 589
- minExclusive (*rdflib.CSVW* attribute), 402
- minExclusive (*rdflib.SH* attribute), 593
- minExclusive (*rdflib.XSD* attribute), 611
- MinExclusiveConstraintComponent (*rdflib.SH* attribute), 589
- Minimum (class in *rdflib.plugins.sparql.aggregates*), 144
- MinimumAdvertisedPrice (*rdflib.SDO* attribute), 506
- minimumPaymentDue (*rdflib.SDO* attribute), 565
- minInclusive (*rdflib.CSVW* attribute), 402
- minInclusive (*rdflib.SH* attribute), 593
- minInclusive (*rdflib.XSD* attribute), 611
- MinInclusiveConstraintComponent (*rdflib.SH* attribute), 589
- minLength (*rdflib.CSVW* attribute), 402
- minLength (*rdflib.SH* attribute), 593
- minLength (*rdflib.XSD* attribute), 611
- MinLengthConstraintComponent (*rdflib.SH* attribute), 589
- minPrice (*rdflib.SDO* attribute), 565
- minQualifiedCardinality (*rdflib.OWL* attribute), 465
- Minus() (in module *rdflib.plugins.sparql.algebra*), 147
- minute (*rdflib.TIME* attribute), 603
- minute (*rdflib.XSD* attribute), 611
- minutes (*rdflib.TIME* attribute), 603
- minValue (*rdflib.SDO* attribute), 565
- MisconceptionsHealthAspect (*rdflib.SDO* attribute), 506
- missionCoveragePrioritiesPolicy (*rdflib.SDO* attribute), 565
- Mixed\_Air (*rdflib.BRICK* attribute), 374
- Mixed\_Air\_Filter (*rdflib.BRICK* attribute), 374
- Mixed\_Air\_Flow\_Sensor (*rdflib.BRICK* attribute), 374
- Mixed\_Air\_Humidity\_Sensor (*rdflib.BRICK* attribute), 374
- Mixed\_Air\_Humidity\_Setpoint (*rdflib.BRICK* attribute), 374
- Mixed\_Air\_Temperature\_Sensor (*rdflib.BRICK* attribute), 374
- Mixed\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 374
- Mixed\_Damper (*rdflib.BRICK* attribute), 374
- MixedEventAttendanceMode (*rdflib.SDO* attribute), 506
- MixtapeAlbum (*rdflib.SDO* attribute), 506
- MobileApplication (*rdflib.SDO* attribute), 506
- MobilePhoneStore (*rdflib.SDO* attribute), 506
- Mode\_Command (*rdflib.BRICK* attribute), 375
- Mode\_Status (*rdflib.BRICK* attribute), 375
- model (*rdflib.SDO* attribute), 565
- modelDate (*rdflib.SDO* attribute), 565
- ModificationException, 268
- modified (*rdflib.DCTERMS* attribute), 413
- modifiedTime (*rdflib.SDO* attribute), 565
- modify (*rdflib.ODRL2* attribute), 458
- Modify (*rdflib.PROV* attribute), 469
- module
- examples.berkeleydb\_example, 23
  - examples.conjunctive\_graphs, 22
  - examples.custom\_datatype, 22
  - examples.custom\_eval, 22
  - examples.foafpaths, 22
  - examples.prepared\_query, 23
  - examples.resource\_example, 23
  - examples.secure\_with\_audit, 25
  - examples.secure\_with\_urlopen, 26
  - examples.slice, 24
  - examples.smushing, 24
  - examples.sparql\_query\_example, 24
  - examples.sparql\_update\_example, 24
  - examples.sparqlstore\_example, 24
  - examples.swap\_primer, 25
  - examples.transitive, 25
  - rdflib, 334
  - rdflib.collection, 221
  - rdflib.compare, 226
  - rdflib.compat, 230
  - rdflib.container, 230
  - rdflib.events, 233
  - rdflib.exceptions, 235
  - rdflib.extras, 69
  - rdflib.extras.cmdlineutils, 47
  - rdflib.extras.describer, 47
  - rdflib.extras.external\_graph\_libs, 51
  - rdflib.extras.infixowl, 55
  - rdflib.graph, 236
  - rdflib.namespace, 69
  - rdflib.parser, 276
  - rdflib.paths, 279
  - rdflib.plugin, 288
  - rdflib.plugins, 218
  - rdflib.plugins.parsers, 113
  - rdflib.plugins.parsers.hext, 79
  - rdflib.plugins.parsers.jsonld, 80
  - rdflib.plugins.parsers.notation3, 81
  - rdflib.plugins.parsers.nquads, 97
  - rdflib.plugins.parsers.ntriples, 98
  - rdflib.plugins.parsers.RDFVOC, 79
  - rdflib.plugins.parsers.rdfxml, 102
  - rdflib.plugins.parsers.trig, 108
  - rdflib.plugins.parsers.trix, 110
  - rdflib.plugins.serializers, 124
  - rdflib.plugins.serializers.hext, 113
  - rdflib.plugins.serializers.jsonld, 114
  - rdflib.plugins.serializers.longturtle, 115
  - rdflib.plugins.serializers.n3, 116

rdflib.plugins.serializers.nquads, 117  
 rdflib.plugins.serializers.nt, 118  
 rdflib.plugins.serializers.rdfxml, 118  
 rdflib.plugins.serializers.trig, 120  
 rdflib.plugins.serializers.trix, 120  
 rdflib.plugins.serializers.turtle, 121  
 rdflib.plugins.serializers.xmlwriter, 123  
 rdflib.plugins.shared, 131  
 rdflib.plugins.shared.jsonld, 131  
 rdflib.plugins.shared.jsonld.context, 124  
 rdflib.plugins.shared.jsonld.errors, 130  
 rdflib.plugins.shared.jsonld.keys, 130  
 rdflib.plugins.shared.jsonld.util, 130  
 rdflib.plugins.sparql, 188  
 rdflib.plugins.sparql.aggregates, 139  
 rdflib.plugins.sparql.algebra, 145  
 rdflib.plugins.sparql.datatypes, 152  
 rdflib.plugins.sparql.evaluate, 152  
 rdflib.plugins.sparql.evalutils, 156  
 rdflib.plugins.sparql.operators, 156  
 rdflib.plugins.sparql.parser, 168  
 rdflib.plugins.sparql.parserutils, 169  
 rdflib.plugins.sparql.processor, 174  
 rdflib.plugins.sparql.results, 139  
 rdflib.plugins.sparql.results.csvresults, 131  
 rdflib.plugins.sparql.results.graph, 133  
 rdflib.plugins.sparql.results.jsonresults, 133  
 rdflib.plugins.sparql.results.rdfresults, 135  
 rdflib.plugins.sparql.results.tsvresults, 135  
 rdflib.plugins.sparql.results.txtresults, 136  
 rdflib.plugins.sparql.results.xmlresults, 136  
 rdflib.plugins.sparql.sparql, 176  
 rdflib.plugins.sparql.update, 185  
 rdflib.plugins.stores, 218  
 rdflib.plugins.stores.auditable, 189  
 rdflib.plugins.stores.berkeleydb, 192  
 rdflib.plugins.stores.concurrent, 195  
 rdflib.plugins.stores.memory, 196  
 rdflib.plugins.stores.regexmatching, 201  
 rdflib.plugins.stores.sparqlconnector, 203  
 rdflib.plugins.stores.sparqlstore, 205  
 rdflib.query, 291  
 rdflib.resource, 298  
 rdflib.serializer, 305  
 rdflib.store, 306  
 rdflib.term, 313  
 rdflib.tools, 221  
 rdflib.tools.chunk\_serializer, 218  
 rdflib.tools.csv2rdf, 219  
 rdflib.tools.defined\_namespace\_creator, 219  
 rdflib.tools.graphisomorphism, 220  
 rdflib.tools.rdf2dot, 221  
 rdflib.tools.rdfpipe, 221  
 rdflib.tools.rdfs2dot, 221  
 rdflib.util, 330  
 rdflib.void, 334  
 module (*rdflib.DOAP attribute*), 415  
 MolecularEntity (*rdflib.SDO attribute*), 506  
 molecularFormula (*rdflib.SDO attribute*), 565  
 molecularWeight (*rdflib.SDO attribute*), 565  
 Monday (*rdflib.SDO attribute*), 506  
 Monday (*rdflib.TIME attribute*), 600  
 MonetaryAmount (*rdflib.SDO attribute*), 506  
 MonetaryAmountDistribution (*rdflib.SDO attribute*), 506  
 MonetaryGrant (*rdflib.SDO attribute*), 506  
 MoneyTransfer (*rdflib.SDO attribute*), 506  
 monoisotopicMolecularWeight (*rdflib.SDO attribute*), 565  
 month (*rdflib.TIME attribute*), 603  
 month (*rdflib.XSD attribute*), 611  
 monthlyMinimumRepaymentAmount (*rdflib.SDO attribute*), 565  
 MonthOfYear (*rdflib.TIME attribute*), 600  
 monthOfYear (*rdflib.TIME attribute*), 603  
 months (*rdflib.TIME attribute*), 603  
 monthsOfExperience (*rdflib.SDO attribute*), 565  
 more\_than() (in module *rdflib.util*), 333  
 MortgageLoan (*rdflib.SDO attribute*), 506  
 Mosque (*rdflib.SDO attribute*), 506  
 Motel (*rdflib.SDO attribute*), 506  
 Motion\_Sensor (*rdflib.BRICK attribute*), 375  
 Motor (*rdflib.BRICK attribute*), 375  
 Motor\_Control\_Center (*rdflib.BRICK attribute*), 375  
 Motor\_Current\_Sensor (*rdflib.BRICK attribute*), 375  
 Motor\_Direction\_Status (*rdflib.BRICK attribute*), 375  
 Motor\_On\_Off\_Status (*rdflib.BRICK attribute*), 375  
 Motor\_Speed\_Sensor (*rdflib.BRICK attribute*), 375  
 Motor\_Torque\_Sensor (*rdflib.BRICK attribute*), 375  
 Motorcycle (*rdflib.SDO attribute*), 506  
 MotorcycleDealer (*rdflib.SDO attribute*), 506  
 MotorcycleRepair (*rdflib.SDO attribute*), 506  
 MotorizedBicycle (*rdflib.SDO attribute*), 507  
 Mountain (*rdflib.SDO attribute*), 507  
 move (*rdflib.ODRL2 attribute*), 458  
 MoveAction (*rdflib.SDO attribute*), 507  
 Movie (*rdflib.SDO attribute*), 507  
 MovieClip (*rdflib.SDO attribute*), 507  
 MovieRentalStore (*rdflib.SDO attribute*), 507

- MovieSeries (*rdflib.SDO* attribute), 507  
 MovieTheater (*rdflib.SDO* attribute), 507  
 MovingCompany (*rdflib.SDO* attribute), 507  
 MovingImage (*rdflib.DCMITYPE* attribute), 410  
 mpn (*rdflib.SDO* attribute), 565  
 MRI (*rdflib.SDO* attribute), 502  
 msnChatID (*rdflib.FOAF* attribute), 422  
 MSRP (*rdflib.SDO* attribute), 502  
 mul\_path() (in module *rdflib.paths*), 288  
 MulPath (class in *rdflib.paths*), 283  
 MulticellularParasite (*rdflib.SDO* attribute), 507  
 MultiCenterTrial (*rdflib.SDO* attribute), 507  
 MultiPlayer (*rdflib.SDO* attribute), 507  
 multipleValues (*rdflib.SDO* attribute), 565  
 MultiplicativeExpression() (in module *rdflib.plugins.sparql.operators*), 164  
 Muscle (*rdflib.SDO* attribute), 507  
 muscleAction (*rdflib.SDO* attribute), 565  
 Musculoskeletal (*rdflib.SDO* attribute), 507  
 MusculoskeletalExam (*rdflib.SDO* attribute), 507  
 Museum (*rdflib.SDO* attribute), 507  
 MusicAlbum (*rdflib.SDO* attribute), 507  
 MusicAlbumProductionType (*rdflib.SDO* attribute), 507  
 MusicAlbumReleaseType (*rdflib.SDO* attribute), 507  
 musicalKey (*rdflib.SDO* attribute), 566  
 musicArrangement (*rdflib.SDO* attribute), 565  
 musicBy (*rdflib.SDO* attribute), 565  
 MusicComposition (*rdflib.SDO* attribute), 507  
 musicCompositionForm (*rdflib.SDO* attribute), 565  
 MusicEvent (*rdflib.SDO* attribute), 507  
 MusicGroup (*rdflib.SDO* attribute), 507  
 musicGroupMember (*rdflib.SDO* attribute), 566  
 MusicPlaylist (*rdflib.SDO* attribute), 507  
 MusicRecording (*rdflib.SDO* attribute), 507  
 MusicRelease (*rdflib.SDO* attribute), 507  
 musicReleaseFormat (*rdflib.SDO* attribute), 566  
 MusicReleaseFormatType (*rdflib.SDO* attribute), 508  
 MusicStore (*rdflib.SDO* attribute), 508  
 MusicVenue (*rdflib.SDO* attribute), 508  
 MusicVideoObject (*rdflib.SDO* attribute), 508  
 myersBriggs (*rdflib.FOAF* attribute), 422
- ## N
- n (*rdflib.PROV* attribute), 472  
 n3() (*rdflib.BNode* method), 336  
 n3() (*rdflib.collection.Collection* method), 225  
 n3() (*rdflib.container.Container* method), 233  
 n3() (*rdflib.Graph* method), 432  
 n3() (*rdflib.graph.Graph* method), 259  
 n3() (*rdflib.graph.QuotedGraph* method), 269  
 n3() (*rdflib.graph.ReadOnlyGraphAggregate* method), 272  
 n3() (*rdflib.Literal* method), 449  
 n3() (*rdflib.paths.AlternativePath* method), 282  
 n3() (*rdflib.paths.InvPath* method), 283  
 n3() (*rdflib.paths.MulPath* method), 284  
 n3() (*rdflib.paths.NegatedPath* method), 284  
 n3() (*rdflib.paths.SequencePath* method), 287  
 n3() (*rdflib.term.BNode* method), 314  
 n3() (*rdflib.term.Literal* method), 325  
 n3() (*rdflib.term.URIRef* method), 329  
 n3() (*rdflib.term.Variable* method), 329  
 n3() (*rdflib.URIRef* method), 606  
 n3() (*rdflib.Variable* method), 608  
 N3Parser (class in *rdflib.plugins.parsers.notation3*), 83  
 N3Serializer (class in *rdflib.plugins.serializers.n3*), 116  
 naics (*rdflib.SDO* attribute), 566  
 NailSalon (*rdflib.SDO* attribute), 508  
 name (*rdflib.CSVW* attribute), 402  
 name (*rdflib.DOAP* attribute), 415  
 name (*rdflib.FOAF* attribute), 422  
 name (*rdflib.plugins.shared.jsonld.context.Term* attribute), 130  
 name (*rdflib.SDO* attribute), 566  
 name (*rdflib.SH* attribute), 593  
 Name (*rdflib.XSD* attribute), 609  
 NamedIndividual (*rdflib.OWL* attribute), 463  
 namedPosition (*rdflib.SDO* attribute), 566  
 Namespace (class in *rdflib*), 451  
 Namespace (class in *rdflib.namespace*), 72  
 namespace (*rdflib.SH* attribute), 593  
 namespace() (*rdflib.plugins.stores.auditable.AuditableStore* method), 190  
 namespace() (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* method), 194  
 namespace() (*rdflib.plugins.stores.memory.Memory* method), 197  
 namespace() (*rdflib.plugins.stores.memory.SimpleMemory* method), 200  
 namespace() (*rdflib.plugins.stores.regexmatching.REGEXMatching* method), 202  
 namespace() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 208  
 namespace() (*rdflib.store.Store* method), 310  
 namespace\_manager (*rdflib.Graph* property), 432  
 namespace\_manager (*rdflib.graph.Graph* property), 259  
 NamespaceManager (class in *rdflib.namespace*), 74  
 namespaces() (*rdflib.Graph* method), 432  
 namespaces() (*rdflib.graph.Graph* method), 259  
 namespaces() (*rdflib.graph.ReadOnlyGraphAggregate* method), 272  
 namespaces() (*rdflib.namespace.NamespaceManager* method), 77  
 namespaces() (*rdflib.plugins.serializers.xmlwriter.XMLWriter* method), 123



[namespaces\(\)](#) ([rdflib.plugins.stores.auditable.AuditableStore](#) method), 190  
[namespaces\(\)](#) ([rdflib.plugins.stores.berkeleydb.BerkeleyDB](#) method), 194  
[namespaces\(\)](#) ([rdflib.plugins.stores.memory.Memory](#) method), 197  
[namespaces\(\)](#) ([rdflib.plugins.stores.memory.SimpleMemory](#) method), 200  
[namespaces\(\)](#) ([rdflib.plugins.stores.regexmatching.REGEXMatching](#) method), 202  
[namespaces\(\)](#) ([rdflib.plugins.stores.sparqlstore.SPARQLStore](#) method), 208  
[namespaces\(\)](#) ([rdflib.store.Store](#) method), 310  
[narrower](#) ([rdflib.SKOS](#) attribute), 597  
[narrowerTransitive](#) ([rdflib.SKOS](#) attribute), 597  
[narrowMatch](#) ([rdflib.SKOS](#) attribute), 597  
[nationality](#) ([rdflib.SDO](#) attribute), 566  
[Natural\\_Gas](#) ([rdflib.BRICK](#) attribute), 375  
[Natural\\_Gas\\_Boiler](#) ([rdflib.BRICK](#) attribute), 375  
[naturalProgression](#) ([rdflib.SDO](#) attribute), 566  
[NCName](#) ([rdflib.XSD](#) attribute), 609  
[Neck](#) ([rdflib.SDO](#) attribute), 508  
[neg\(\)](#) (in module [rdflib.plugins.sparql.parser](#)), 168  
[neg\\_path\(\)](#) (in module [rdflib.paths](#)), 288  
[NegatedPath](#) (class in [rdflib.paths](#)), 284  
[negativeInteger](#) ([rdflib.XSD](#) attribute), 611  
[negativeNotes](#) ([rdflib.SDO](#) attribute), 566  
[NegativePropertyAssertion](#) ([rdflib.OWL](#) attribute), 464  
[neq](#) ([rdflib.ODRL2](#) attribute), 458  
[neq\(\)](#) ([rdflib.Literal](#) method), 451  
[neq\(\)](#) ([rdflib.term.Identifier](#) method), 317  
[neq\(\)](#) ([rdflib.term.Literal](#) method), 326  
[Nerve](#) ([rdflib.SDO](#) attribute), 508  
[nerve](#) ([rdflib.SDO](#) attribute), 566  
[nerveMotor](#) ([rdflib.SDO](#) attribute), 566  
[netArea](#) ([rdflib.BRICK](#) attribute), 399  
[Network\\_Video\\_Recorder](#) ([rdflib.BRICK](#) attribute), 375  
[netWorth](#) ([rdflib.SDO](#) attribute), 566  
[Neuro](#) ([rdflib.SDO](#) attribute), 508  
[Neurologic](#) ([rdflib.SDO](#) attribute), 508  
[newBlankNode\(\)](#) ([rdflib.plugins.parsers.notation3.Formula](#) method), 83  
[newBlankNode\(\)](#) ([rdflib.plugins.parsers.notation3.RDFSink](#) method), 85  
[NewCondition](#) ([rdflib.SDO](#) attribute), 508  
[newFormula\(\)](#) ([rdflib.plugins.parsers.notation3.RDFSink](#) method), 85  
[newGraph\(\)](#) ([rdflib.plugins.parsers.notation3.RDFSink](#) method), 85  
[newList\(\)](#) ([rdflib.plugins.parsers.notation3.RDFSink](#) method), 85  
[newLiteral\(\)](#) ([rdflib.plugins.parsers.notation3.RDFSink](#) method), 85  
[NewsArticle](#) ([rdflib.SDO](#) attribute), 508  
[newSet\(\)](#) ([rdflib.plugins.parsers.notation3.RDFSink](#) method), 85  
[NewsMediaOrganization](#) ([rdflib.SDO](#) attribute), 508  
[Newspaper](#) ([rdflib.SDO](#) attribute), 508  
[newsUpdatesAndGuidelines](#) ([rdflib.SDO](#) attribute), 566  
[newSymbol\(\)](#) ([rdflib.plugins.parsers.notation3.RDFSink](#) method), 86  
[newUniversal\(\)](#) ([rdflib.plugins.parsers.notation3.Formula](#) method), 83  
[next](#) ([rdflib.plugins.parsers.rdfxml.RDFXMLHandler](#) property), 105  
[next\\_li\(\)](#) ([rdflib.plugins.parsers.rdfxml.BagID](#) method), 102  
[next\\_li\(\)](#) ([rdflib.plugins.parsers.rdfxml.ElementHandler](#) method), 103  
[nextItem](#) ([rdflib.SDO](#) attribute), 566  
[nextPolicy](#) ([rdflib.ODRL2](#) attribute), 458  
[NGO](#) ([rdflib.SDO](#) attribute), 508  
[nick](#) ([rdflib.FOAF](#) attribute), 422  
[NightClub](#) ([rdflib.SDO](#) attribute), 508  
[nil](#) ([rdflib.RDF](#) attribute), 478  
[NLM](#) ([rdflib.DCTERMS](#) attribute), 411  
[NLNonprofitType](#) ([rdflib.SDO](#) attribute), 508  
[NMTOKEN](#) ([rdflib.XSD](#) attribute), 609  
[NMTOKENS](#) ([rdflib.XSD](#) attribute), 609  
[NO2\\_Level\\_Sensor](#) ([rdflib.BRICK](#) attribute), 375  
[No\\_Water\\_Alarm](#) ([rdflib.BRICK](#) attribute), 375  
[noBylinesPolicy](#) ([rdflib.SDO](#) attribute), 566  
[Node](#) (class in [rdflib.term](#)), 326  
[node](#) ([rdflib.SH](#) attribute), 593  
[node\(\)](#) ([rdflib.plugins.parsers.notation3.SinkParser](#) method), 90  
[node\\_element\\_end\(\)](#) ([rdflib.plugins.parsers.rdfxml.RDFXMLHandler](#) method), 105  
[node\\_element\\_start\(\)](#) ([rdflib.plugins.parsers.rdfxml.RDFXMLHandler](#) method), 105  
[node\\_picker](#) ([rdflib.store.Store](#) property), 310  
[NodeConstraintComponent](#) ([rdflib.SH](#) attribute), 589  
[nodeID](#) ([rdflib.plugins.parsers.RDFVOC.RDFVOC](#) attribute), 79  
[nodeid\(\)](#) ([rdflib.plugins.parsers.ntriples.W3CNTriplesParser](#) method), 100  
[NodeKind](#) ([rdflib.SH](#) attribute), 590  
[nodeKind](#) ([rdflib.SH](#) attribute), 593  
[NodeKindConstraintComponent](#) ([rdflib.SH](#) attribute), 590  
[nodeOrLiteral\(\)](#) ([rdflib.plugins.parsers.notation3.SinkParser](#) method), 90



- method*), 91
- NodePickler (class in *rdflib.store*), 306
- nodes (*rdflib.SH* attribute), 593
- NodeShape (*rdflib.SH* attribute), 590
- nodeValidator (*rdflib.SH* attribute), 593
- NoElementException, 233
- nominalPosition (*rdflib.TIME* attribute), 603
- Noncondensing\_Natural\_Gas\_Boiler (*rdflib.BRICK* attribute), 375
- nonEqual (*rdflib.SDO* attribute), 566
- NoninvasiveProcedure (*rdflib.SDO* attribute), 508
- nonNegativeInteger (*rdflib.XSD* attribute), 611
- nonPositiveInteger (*rdflib.XSD* attribute), 611
- Nonprofit501a (*rdflib.SDO* attribute), 508
- Nonprofit501c1 (*rdflib.SDO* attribute), 508
- Nonprofit501c10 (*rdflib.SDO* attribute), 508
- Nonprofit501c11 (*rdflib.SDO* attribute), 508
- Nonprofit501c12 (*rdflib.SDO* attribute), 508
- Nonprofit501c13 (*rdflib.SDO* attribute), 508
- Nonprofit501c14 (*rdflib.SDO* attribute), 508
- Nonprofit501c15 (*rdflib.SDO* attribute), 508
- Nonprofit501c16 (*rdflib.SDO* attribute), 508
- Nonprofit501c17 (*rdflib.SDO* attribute), 508
- Nonprofit501c18 (*rdflib.SDO* attribute), 508
- Nonprofit501c19 (*rdflib.SDO* attribute), 509
- Nonprofit501c2 (*rdflib.SDO* attribute), 509
- Nonprofit501c20 (*rdflib.SDO* attribute), 509
- Nonprofit501c21 (*rdflib.SDO* attribute), 509
- Nonprofit501c22 (*rdflib.SDO* attribute), 509
- Nonprofit501c23 (*rdflib.SDO* attribute), 509
- Nonprofit501c24 (*rdflib.SDO* attribute), 509
- Nonprofit501c25 (*rdflib.SDO* attribute), 509
- Nonprofit501c26 (*rdflib.SDO* attribute), 509
- Nonprofit501c27 (*rdflib.SDO* attribute), 509
- Nonprofit501c28 (*rdflib.SDO* attribute), 509
- Nonprofit501c3 (*rdflib.SDO* attribute), 509
- Nonprofit501c4 (*rdflib.SDO* attribute), 509
- Nonprofit501c5 (*rdflib.SDO* attribute), 509
- Nonprofit501c6 (*rdflib.SDO* attribute), 509
- Nonprofit501c7 (*rdflib.SDO* attribute), 509
- Nonprofit501c8 (*rdflib.SDO* attribute), 509
- Nonprofit501c9 (*rdflib.SDO* attribute), 509
- Nonprofit501d (*rdflib.SDO* attribute), 509
- Nonprofit501e (*rdflib.SDO* attribute), 509
- Nonprofit501f (*rdflib.SDO* attribute), 509
- Nonprofit501k (*rdflib.SDO* attribute), 509
- Nonprofit501n (*rdflib.SDO* attribute), 509
- Nonprofit501q (*rdflib.SDO* attribute), 509
- Nonprofit527 (*rdflib.SDO* attribute), 509
- NonprofitANBI (*rdflib.SDO* attribute), 509
- NonprofitSBBi (*rdflib.SDO* attribute), 509
- nonprofitStatus (*rdflib.SDO* attribute), 566
- NonprofitType (*rdflib.SDO* attribute), 509
- nonProprietaryName (*rdflib.SDO* attribute), 566
- norm\_url() (in module *rdflib.plugins.shared.jsonld.util*), 130
- normalise() (*rdflib.plugins.parsers.notation3.RDFSink* method), 86
- normalize() (*rdflib.Literal* method), 451
- normalize() (*rdflib.term.Literal* method), 326
- normalizedString (*rdflib.XSD* attribute), 611
- normalizeUri() (*rdflib.namespace.NamespaceManager* method), 78
- normalRange (*rdflib.SDO* attribute), 566
- Nose (*rdflib.SDO* attribute), 509
- not\_() (in module *rdflib.plugins.sparql.operators*), 167
- Notary (*rdflib.SDO* attribute), 510
- notation (*rdflib.SKOS* attribute), 597
- NOTATION (*rdflib.XSD* attribute), 609
- NotBoundError, 180
- NotConstraintComponent (*rdflib.SH* attribute), 590
- note (*rdflib.CSVW* attribute), 402
- note (*rdflib.SKOS* attribute), 597
- NoteDigitalDocument (*rdflib.SDO* attribute), 510
- Nothing (*rdflib.OWL* attribute), 464
- NotInForce (*rdflib.SDO* attribute), 509
- NotYetRecruiting (*rdflib.SDO* attribute), 509
- now (*rdflib.plugins.sparql.sparql.FrozenBindings* property), 178
- now (*rdflib.plugins.sparql.sparql.QueryContext* property), 184
- NQuadsParser (class in *rdflib.plugins.parsers.nquads*), 97
- NQuadsSerializer (class in *rdflib.plugins.serializers.nquads*), 117
- nsBindings (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* attribute), 214
- nsn (*rdflib.SDO* attribute), 566
- NTGraphSink (class in *rdflib.plugins.parsers.ntriples*), 99
- NTParser (class in *rdflib.plugins.parsers.ntriples*), 99
- NTSerializer (class in *rdflib.plugins.serializers.nt*), 118
- null (*rdflib.CSVW* attribute), 402
- numAdults (*rdflib.SDO* attribute), 566
- number (*rdflib.plugins.parsers.notation3.Formula* attribute), 83
- Number (*rdflib.SDO* attribute), 510
- numberedPosition (*rdflib.SDO* attribute), 567
- numberOfAccommodationUnits (*rdflib.SDO* attribute), 566
- numberOfAirbags (*rdflib.SDO* attribute), 566
- numberOfAvailableAccommodationUnits (*rdflib.SDO* attribute), 567
- numberOfAxles (*rdflib.SDO* attribute), 567
- numberOfBathroomsTotal (*rdflib.SDO* attribute), 567
- numberOfBedrooms (*rdflib.SDO* attribute), 567
- numberOfBeds (*rdflib.SDO* attribute), 567
- numberOfCredits (*rdflib.SDO* attribute), 567

numberOfDoors (*rdflib.SDO* attribute), 567  
 numberOfEmployees (*rdflib.SDO* attribute), 567  
 numberOfEpisodes (*rdflib.SDO* attribute), 567  
 numberOfForwardGears (*rdflib.SDO* attribute), 567  
 numberOfFullBathrooms (*rdflib.SDO* attribute), 567  
 numberOfItems (*rdflib.SDO* attribute), 567  
 numberOfLoanPayments (*rdflib.SDO* attribute), 567  
 numberOfPages (*rdflib.SDO* attribute), 567  
 numberOfPartialBathrooms (*rdflib.SDO* attribute), 567  
 numberOfPlayers (*rdflib.SDO* attribute), 567  
 numberOfPreviousOwners (*rdflib.SDO* attribute), 567  
 numberOfRooms (*rdflib.SDO* attribute), 567  
 numberOfSeasons (*rdflib.SDO* attribute), 567  
 numChildren (*rdflib.SDO* attribute), 566  
 numConstraints (*rdflib.SDO* attribute), 566  
 numeric (*rdflib.XSD* attribute), 611  
 numeric() (in module *rdflib.plugins.sparql.operators*), 167  
 numericDuration (*rdflib.TIME* attribute), 603  
 NumericFormat (*rdflib.CSVW* attribute), 400  
 numericPosition (*rdflib.TIME* attribute), 603  
 numTracks (*rdflib.SDO* attribute), 566  
 Nursing (*rdflib.SDO* attribute), 510  
 nutrition (*rdflib.SDO* attribute), 567  
 NutritionInformation (*rdflib.SDO* attribute), 510  
 NVR (*rdflib.BRICK* attribute), 375

## O

object (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 103  
 object (*rdflib.RDF* attribute), 478  
 object (*rdflib.SDO* attribute), 567  
 object (*rdflib.SH* attribute), 593  
 object() (*rdflib.plugins.parsers.notation3.SinkParser* method), 91  
 object() (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* method), 100  
 objectList() (*rdflib.plugins.parsers.notation3.SinkParser* method), 91  
 objectList() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 116  
 objectList() (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 122  
 ObjectProperty (*rdflib.OWL* attribute), 464  
 objects() (*rdflib.Graph* method), 432  
 objects() (*rdflib.graph.Graph* method), 259  
 objects() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 208  
 objects() (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 214  
 objects() (*rdflib.resource.Resource* method), 305  
 objectsTarget (*rdflib.VOID* attribute), 607  
 obligation (*rdflib.ODRL2* attribute), 458

ObservableProperty (*rdflib.SOSA* attribute), 597  
 Observation (*rdflib.QB* attribute), 476  
 observation (*rdflib.QB* attribute), 477  
 Observation (*rdflib.SDO* attribute), 510  
 Observation (*rdflib.SOSA* attribute), 597  
 Observational (*rdflib.SDO* attribute), 510  
 observationDate (*rdflib.SDO* attribute), 567  
 ObservationGroup (*rdflib.QB* attribute), 476  
 observationGroup (*rdflib.QB* attribute), 477  
 observedNode (*rdflib.SDO* attribute), 567  
 observedProperty (*rdflib.SOSA* attribute), 598  
 observes (*rdflib.SOSA* attribute), 599  
 Obstetric (*rdflib.SDO* attribute), 510  
 obtainConsent (*rdflib.ODRL2* attribute), 458  
 occupancy (*rdflib.SDO* attribute), 568  
 Occupancy\_Command (*rdflib.BRICK* attribute), 375  
 Occupancy\_Sensor (*rdflib.BRICK* attribute), 375  
 Occupancy\_Status (*rdflib.BRICK* attribute), 375  
 Occupation (*rdflib.SDO* attribute), 510  
 OccupationalActivity (*rdflib.SDO* attribute), 510  
 occupationalCategory (*rdflib.SDO* attribute), 568  
 occupationalCredentialAwarded (*rdflib.SDO* attribute), 568  
 OccupationalExperienceRequirements (*rdflib.SDO* attribute), 510  
 OccupationalTherapy (*rdflib.SDO* attribute), 510  
 occupationLocation (*rdflib.SDO* attribute), 568  
 Occupied\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Cooling\_Discharge\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Cooling\_Supply\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Cooling\_Temperature\_Deadband\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Discharge\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Discharge\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Heating\_Discharge\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Heating\_Supply\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Heating\_Temperature\_Deadband\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Mode\_Status (*rdflib.BRICK* attribute), 376  
 Occupied\_Return\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Room\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Supply\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 376  
 Occupied\_Supply\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 376

- Occupied\_Zone\_Air\_Temperature\_Setpoint (*rdflib.BRICK attribute*), 376
- OceanBodyOfWater (*rdflib.SDO attribute*), 510
- ODRL2 (*class in rdflib*), 453
- Off\_Command (*rdflib.BRICK attribute*), 376
- Off\_Status (*rdflib.BRICK attribute*), 376
- Offer (*rdflib.ODRL2 attribute*), 454
- Offer (*rdflib.SDO attribute*), 510
- OfferCatalog (*rdflib.SDO attribute*), 510
- offerCount (*rdflib.SDO attribute*), 568
- offeredBy (*rdflib.SDO attribute*), 568
- OfferForLease (*rdflib.SDO attribute*), 510
- OfferForPurchase (*rdflib.SDO attribute*), 510
- OfferItemCondition (*rdflib.SDO attribute*), 510
- offers (*rdflib.SDO attribute*), 568
- OfferShippingDetails (*rdflib.SDO attribute*), 510
- offersPrescriptionByMail (*rdflib.SDO attribute*), 568
- Office (*rdflib.BRICK attribute*), 376
- Office\_Kitchen (*rdflib.BRICK attribute*), 377
- OfficeEquipmentStore (*rdflib.SDO attribute*), 510
- OfficialLegalValue (*rdflib.SDO attribute*), 510
- OfflineEventAttendanceMode (*rdflib.SDO attribute*), 510
- OfflinePermanently (*rdflib.SDO attribute*), 511
- OfflineTemporarily (*rdflib.SDO attribute*), 511
- Oil (*rdflib.BRICK attribute*), 377
- On\_Command (*rdflib.BRICK attribute*), 377
- On\_Off\_Command (*rdflib.BRICK attribute*), 377
- On\_Off\_Status (*rdflib.BRICK attribute*), 377
- On\_Status (*rdflib.BRICK attribute*), 377
- On\_Timer\_Sensor (*rdflib.BRICK attribute*), 377
- onClass (*rdflib.OWL attribute*), 465
- Oncologic (*rdflib.SDO attribute*), 511
- onDataRange (*rdflib.OWL attribute*), 465
- onDatatype (*rdflib.OWL attribute*), 466
- OnDemandEvent (*rdflib.SDO attribute*), 511
- oneOf (*rdflib.OWL attribute*), 466
- oneOrMorePath (*rdflib.SH attribute*), 593
- OneTimePayments (*rdflib.SDO attribute*), 511
- Online (*rdflib.SDO attribute*), 511
- OnlineAccount (*rdflib.FOAF attribute*), 420
- OnlineChatAccount (*rdflib.FOAF attribute*), 420
- OnlineEcommerceAccount (*rdflib.FOAF attribute*), 420
- OnlineEventAttendanceMode (*rdflib.SDO attribute*), 511
- OnlineFull (*rdflib.SDO attribute*), 511
- OnlineGamingAccount (*rdflib.FOAF attribute*), 420
- OnlineOnly (*rdflib.SDO attribute*), 511
- onProperties (*rdflib.OWL attribute*), 466
- onProperty (*rdflib.extras.infixowl.Restriction property*), 68
- onProperty (*rdflib.OWL attribute*), 466
- OnSitePickup (*rdflib.SDO attribute*), 511
- Ontology (*class in rdflib.extras.infixowl*), 67
- Ontology (*rdflib.OWL attribute*), 464
- OntologyProperty (*rdflib.OWL attribute*), 464
- open() (*rdflib.Graph method*), 432
- open() (*rdflib.graph.Graph method*), 259
- open() (*rdflib.graph.ReadOnlyGraphAggregate method*), 272
- open() (*rdflib.plugins.stores.auditable.AuditableStore method*), 190
- open() (*rdflib.plugins.stores.berkeleydb.BerkeleyDB method*), 194
- open() (*rdflib.plugins.stores.regexmatching.REGEXMatching method*), 202
- open() (*rdflib.plugins.stores.sparqlstore.SPARQLStore method*), 208
- open() (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore method*), 214
- open() (*rdflib.store.Store method*), 310
- Open\_Close\_Status (*rdflib.BRICK attribute*), 377
- Open\_Heating\_Valve\_Outside\_Air\_Temperature\_Setpoint (*rdflib.BRICK attribute*), 377
- Open\_Office (*rdflib.BRICK attribute*), 377
- openid (*rdflib.FOAF attribute*), 422
- openingHours (*rdflib.SDO attribute*), 568
- OpeningHoursSpecification (*rdflib.SDO attribute*), 511
- openingHoursSpecification (*rdflib.SDO attribute*), 568
- opens (*rdflib.SDO attribute*), 568
- openSearchDescription (*rdflib.VOID attribute*), 607
- OpenTrial (*rdflib.SDO attribute*), 511
- operand (*rdflib.ODRL2 attribute*), 458
- Operating\_Mode\_Status (*rdflib.BRICK attribute*), 377
- operatingSystem (*rdflib.SDO attribute*), 568
- operationalStage (*rdflib.BRICK attribute*), 399
- operationalStageCount (*rdflib.BRICK attribute*), 399
- Operator (*rdflib.ODRL2 attribute*), 454
- operator (*rdflib.ODRL2 attribute*), 458
- OpinionNewsArticle (*rdflib.SDO attribute*), 511
- opponent (*rdflib.SDO attribute*), 568
- Optician (*rdflib.SDO attribute*), 511
- option (*rdflib.SDO attribute*), 568
- optional (*rdflib.SH attribute*), 594
- Optometric (*rdflib.SDO attribute*), 511
- OrConstraintComponent (*rdflib.SH attribute*), 590
- order (*rdflib.PROV attribute*), 472
- order (*rdflib.QB attribute*), 477
- Order (*rdflib.SDO attribute*), 511
- order (*rdflib.SH attribute*), 594
- OrderAction (*rdflib.SDO attribute*), 511
- OrderBy() (*in module rdflib.plugins.sparql.algebra*), 147
- OrderCancelled (*rdflib.SDO attribute*), 511
- orderDate (*rdflib.SDO attribute*), 568
- OrderDelivered (*rdflib.SDO attribute*), 511



- orderDelivery (*rdflib.SDO* attribute), 568
- ordered (*rdflib.CSVW* attribute), 402
- ordered (*rdflib.XSD* attribute), 611
- OrderedCollection (*rdflib.SKOS* attribute), 596
- orderedItem (*rdflib.SDO* attribute), 568
- OrderInTransit (*rdflib.SDO* attribute), 511
- OrderItem (*rdflib.SDO* attribute), 511
- orderItemNumber (*rdflib.SDO* attribute), 568
- orderItemStatus (*rdflib.SDO* attribute), 568
- orderNumber (*rdflib.SDO* attribute), 568
- OrderPaymentDue (*rdflib.SDO* attribute), 511
- OrderPickupAvailable (*rdflib.SDO* attribute), 511
- OrderProblem (*rdflib.SDO* attribute), 511
- OrderProcessing (*rdflib.SDO* attribute), 511
- orderQuantity (*rdflib.SDO* attribute), 568
- OrderReturned (*rdflib.SDO* attribute), 511
- OrderStatus (*rdflib.SDO* attribute), 512
- orderStatus (*rdflib.SDO* attribute), 568
- orderSubjects() (*rdflib.plugins.serializers.turtle.RecursiveSerializer* method), 121
- ORG (class in *rdflib*), 461
- Organization (*rdflib.FOAF* attribute), 421
- Organization (*rdflib.ORG* attribute), 461
- organization (*rdflib.ORG* attribute), 462
- Organization (*rdflib.PROV* attribute), 469
- Organization (*rdflib.SDO* attribute), 512
- OrganizationalCollaboration (*rdflib.ORG* attribute), 461
- OrganizationalUnit (*rdflib.ORG* attribute), 461
- OrganizationRole (*rdflib.SDO* attribute), 512
- OrganizeAction (*rdflib.SDO* attribute), 512
- organizer (*rdflib.SDO* attribute), 568
- originAddress (*rdflib.SDO* attribute), 568
- OriginalMediaContent (*rdflib.SDO* attribute), 512
- originalMediaContextDescription (*rdflib.SDO* attribute), 568
- originalMediaLink (*rdflib.SDO* attribute), 568
- originalOrganization (*rdflib.ORG* attribute), 462
- OriginalShippingFees (*rdflib.SDO* attribute), 512
- originatesFrom (*rdflib.SDO* attribute), 569
- os (*rdflib.DOAP* attribute), 415
- Osteopathic (*rdflib.SDO* attribute), 512
- OTC (*rdflib.SDO* attribute), 510
- Otolaryngologic (*rdflib.SDO* attribute), 512
- Outdoor\_Area (*rdflib.BRICK* attribute), 377
- OutletStore (*rdflib.SDO* attribute), 512
- OutOfStock (*rdflib.SDO* attribute), 512
- output (*rdflib.ODRL2* attribute), 458
- Output (*rdflib.SSN* attribute), 599
- Output\_Frequency\_Sensor (*rdflib.BRICK* attribute), 377
- Output\_Voltage\_Sensor (*rdflib.BRICK* attribute), 377
- Outside (*rdflib.BRICK* attribute), 377
- Outside\_Air (*rdflib.BRICK* attribute), 377
- Outside\_Air\_CO2\_Sensor (*rdflib.BRICK* attribute), 377
- Outside\_Air\_CO\_Sensor (*rdflib.BRICK* attribute), 377
- Outside\_Air\_Dewpoint\_Sensor (*rdflib.BRICK* attribute), 377
- Outside\_Air\_Enthalpy\_Sensor (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Flow\_Sensor (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Grains\_Sensor (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Humidity\_Sensor (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Humidity\_Setpoint (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Lockout\_Temperature\_Differential\_Parameter (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Lockout\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Temperature\_Enable\_Differential\_Sensor (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Temperature\_High\_Reset\_Setpoint (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Temperature\_Low\_Reset\_Setpoint (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Temperature\_Sensor (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 378
- Outside\_Air\_Wet\_Bulb\_Temperature\_Sensor (*rdflib.BRICK* attribute), 378
- Outside\_Damper (*rdflib.BRICK* attribute), 378
- Outside\_Face\_Surface\_Temperature\_Sensor (*rdflib.BRICK* attribute), 378
- Outside\_Face\_Surface\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 378
- Outside\_Illuminance\_Sensor (*rdflib.BRICK* attribute), 378
- overdosage (*rdflib.SDO* attribute), 569
- Overload\_Alarm (*rdflib.BRICK* attribute), 379
- Overridden\_Off\_Status (*rdflib.BRICK* attribute), 379
- Overridden\_On\_Status (*rdflib.BRICK* attribute), 379
- Overridden\_Status (*rdflib.BRICK* attribute), 379
- Override\_Command (*rdflib.BRICK* attribute), 379
- OverviewHealthAspect (*rdflib.SDO* attribute), 512
- OWL (class in *rdflib*), 462
- OWL RDFListProxy (class in *rdflib.extras.infixowl*), 66
- ownedFrom (*rdflib.SDO* attribute), 569
- ownedThrough (*rdflib.SDO* attribute), 569
- ownershipFundingInfo (*rdflib.SDO* attribute), 569
- OwnershipInfo (*rdflib.SDO* attribute), 512

- owns (*rdflib.SDO* attribute), 569
- Ozone\_Level\_Sensor (*rdflib.BRICK* attribute), 379
- ## P
- p\_clause() (*rdflib.plugins.serializers.n3.N3Serializer* method), 117
- p\_default() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 116
- p\_default() (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 122
- p\_squared() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 116
- p\_squared() (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 122
- packageFormat (*rdflib.DCAT* attribute), 409
- page (*rdflib.FOAF* attribute), 422
- pageEnd (*rdflib.SDO* attribute), 569
- pageStart (*rdflib.SDO* attribute), 569
- pagination (*rdflib.SDO* attribute), 569
- PaidLeave (*rdflib.SDO* attribute), 512
- PaintAction (*rdflib.SDO* attribute), 512
- Painting (*rdflib.SDO* attribute), 512
- pairEntity (*rdflib.PROV* attribute), 472
- pairKey (*rdflib.PROV* attribute), 472
- PalliativeProcedure (*rdflib.SDO* attribute), 512
- panelArea (*rdflib.BRICK* attribute), 399
- Paperback (*rdflib.SDO* attribute), 512
- Param (class in *rdflib.plugins.sparql.parserutils*), 171
- Parameter (*rdflib.BRICK* attribute), 379
- Parameter (*rdflib.SH* attribute), 590
- parameter (*rdflib.SH* attribute), 594
- Parameterizable (*rdflib.SH* attribute), 590
- ParamList (class in *rdflib.plugins.sparql.parserutils*), 172
- ParamValue (class in *rdflib.plugins.sparql.parserutils*), 172
- ParcelDelivery (*rdflib.SDO* attribute), 512
- ParcelService (*rdflib.SDO* attribute), 512
- parent (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* property), 106
- parent (*rdflib.SDO* attribute), 569
- ParentalSupport (*rdflib.SDO* attribute), 512
- ParentAudience (*rdflib.SDO* attribute), 512
- parentChildProperty (*rdflib.QB* attribute), 477
- parentItem (*rdflib.SDO* attribute), 569
- parentOrganization (*rdflib.SDO* attribute), 569
- parents (*rdflib.extras.infixowl.Class* property), 61
- parents (*rdflib.SDO* attribute), 569
- parentService (*rdflib.SDO* attribute), 569
- parentTaxon (*rdflib.SDO* attribute), 569
- Park (*rdflib.SDO* attribute), 512
- Parking\_Level (*rdflib.BRICK* attribute), 380
- Parking\_Space (*rdflib.BRICK* attribute), 380
- Parking\_Structure (*rdflib.BRICK* attribute), 380
- ParkingFacility (*rdflib.SDO* attribute), 512
- ParkingMap (*rdflib.SDO* attribute), 512
- parse() (*rdflib.ConjunctiveGraph* method), 405
- parse() (*rdflib.Dataset* method), 419
- parse() (*rdflib.Graph* method), 433
- parse() (*rdflib.graph.ConjunctiveGraph* method), 243
- parse() (*rdflib.graph.Dataset* method), 249
- parse() (*rdflib.graph.Graph* method), 260
- parse() (*rdflib.graph.ReadOnlyGraphAggregate* method), 272
- parse() (*rdflib.parser.Parser* method), 276
- parse() (*rdflib.plugins.parsers.hexst.HexuplesParser* method), 80
- parse() (*rdflib.plugins.parsers.jsonld.JsonLDParser* method), 81
- parse() (*rdflib.plugins.parsers.notation3.N3Parser* method), 83
- parse() (*rdflib.plugins.parsers.notation3.TurtleParser* method), 95
- parse() (*rdflib.plugins.parsers.nquads.NQuadsParser* method), 98
- parse() (*rdflib.plugins.parsers.ntriples.NTParser* class method), 99
- parse() (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* method), 101
- parse() (*rdflib.plugins.parsers.rdfxml.RDFXMLParser* method), 108
- parse() (*rdflib.plugins.parsers.trig.TrigParser* method), 108
- parse() (*rdflib.plugins.parsers.trix.TriXParser* method), 113
- parse() (*rdflib.plugins.sparql.results.csvresults.CSVResultParser* method), 132
- parse() (*rdflib.plugins.sparql.results.graph.GraphResultParser* method), 133
- parse() (*rdflib.plugins.sparql.results.jsonresults.JSONResultParser* method), 133
- parse() (*rdflib.plugins.sparql.results.rdfresults.RDFResultParser* method), 135
- parse() (*rdflib.plugins.sparql.results.tsvresults.TSVResultParser* method), 136
- parse() (*rdflib.plugins.sparql.results.xmlresults.XMLResultParser* method), 138
- parse() (*rdflib.query.Result* static method), 293
- parse() (*rdflib.query.ResultParser* method), 295
- parse\_and\_serialize() (in module *rdflib.tools.rdfpipe*), 221
- parse\_date\_time() (in module *rdflib.util*), 333
- parseAction (*rdflib.plugins.sparql.parserutils.ParamList* attribute), 172
- parseJsonTerm() (in module *rdflib.plugins.sparql.results.jsonresults*), 134
- parseline() (*rdflib.plugins.parsers.nquads.NQuadsParser* method), 98

parseline() (rdflib.plugins.parsers.ntriples.W3CNTriplesParser method), 101  
 parseQuery() (in module rdflib.plugins.sparql.parser), 169  
 Parser (class in rdflib.parser), 276  
 ParserError, 235  
 parseRow() (rdflib.plugins.sparql.results.csvresults.CSVResults method), 132  
 parsestring() (rdflib.plugins.parsers.ntriples.W3CNTriplesParser method), 101  
 parseTerm() (in module rdflib.plugins.sparql.results.xmlresults), 139  
 parseType (rdflib.plugins.parsers.RDFVOC.RDFVOC attribute), 79  
 parseUpdate() (in module rdflib.plugins.sparql.parser), 169  
 PartiallyInForce (rdflib.SDO attribute), 512  
 participant (rdflib.SDO attribute), 569  
 Particulate\_Matter\_Sensor (rdflib.BRICK attribute), 380  
 partOf (rdflib.ODRL2 attribute), 458  
 partOfEpisode (rdflib.SDO attribute), 569  
 partOfInvoice (rdflib.SDO attribute), 569  
 partOfOrder (rdflib.SDO attribute), 569  
 partOfSeason (rdflib.SDO attribute), 569  
 partOfSeries (rdflib.SDO attribute), 569  
 partOfSystem (rdflib.SDO attribute), 569  
 partOfTrip (rdflib.SDO attribute), 569  
 partOfTVSeries (rdflib.SDO attribute), 569  
 Party (rdflib.ODRL2 attribute), 454  
 PartyCollection (rdflib.ODRL2 attribute), 454  
 PartyScope (rdflib.ODRL2 attribute), 454  
 partySize (rdflib.SDO attribute), 569  
 passengerPriorityStatus (rdflib.SDO attribute), 569  
 passengerSequenceNumber (rdflib.SDO attribute), 569  
 Passive\_Chilled\_Beam (rdflib.BRICK attribute), 380  
 pastProject (rdflib.FOAF attribute), 422  
 Path (class in rdflib.paths), 284  
 path (rdflib.SH attribute), 594  
 path() (rdflib.plugins.parsers.notation3.SinkParser method), 91  
 path() (rdflib.plugins.serializers.longturtle.LongTurtleSerializer method), 116  
 path() (rdflib.plugins.serializers.n3.N3Serializer method), 117  
 path() (rdflib.plugins.serializers.turtle.TurtleSerializer method), 122  
 path\_alternative() (in module rdflib.paths), 288  
 path\_sequence() (in module rdflib.paths), 288  
 PathList (class in rdflib.paths), 286  
 Pathology (rdflib.SDO attribute), 512  
 PathologyTest (rdflib.SDO attribute), 513  
 pathophysiology (rdflib.SDO attribute), 569  
 Patient (rdflib.SDO attribute), 513  
 PatientExperienceHealthAspect (rdflib.SDO attribute), 513  
 pattern (rdflib.CSVW attribute), 402  
 pattern (rdflib.SDO attribute), 570  
 pattern (rdflib.SH attribute), 594  
 pattern (rdflib.XSD attribute), 611  
 PatternConstraintComponent (rdflib.SH attribute), 590  
 PawnShop (rdflib.SDO attribute), 513  
 pay (rdflib.ODRL2 attribute), 458  
 PayAction (rdflib.SDO attribute), 513  
 payAmount (rdflib.ODRL2 attribute), 458  
 payeeParty (rdflib.ODRL2 attribute), 458  
 payload (rdflib.SDO attribute), 570  
 paymentAccepted (rdflib.SDO attribute), 570  
 PaymentAutomaticallyApplied (rdflib.SDO attribute), 513  
 PaymentCard (rdflib.SDO attribute), 513  
 PaymentChargeSpecification (rdflib.SDO attribute), 513  
 PaymentComplete (rdflib.SDO attribute), 513  
 PaymentDeclined (rdflib.SDO attribute), 513  
 PaymentDue (rdflib.SDO attribute), 513  
 paymentDue (rdflib.SDO attribute), 570  
 paymentDueDate (rdflib.SDO attribute), 570  
 PaymentMethod (rdflib.SDO attribute), 513  
 paymentMethod (rdflib.SDO attribute), 570  
 paymentMethodId (rdflib.SDO attribute), 570  
 PaymentPastDue (rdflib.SDO attribute), 513  
 PaymentService (rdflib.SDO attribute), 513  
 paymentStatus (rdflib.SDO attribute), 570  
 PaymentStatusType (rdflib.SDO attribute), 513  
 paymentUrl (rdflib.SDO attribute), 570  
 Peak\_Power\_Demand\_Sensor (rdflib.BRICK attribute), 380  
 Pediatric (rdflib.SDO attribute), 513  
 peek() (rdflib.plugins.parsers.ntriples.W3CNTriplesParser method), 101  
 penciler (rdflib.SDO attribute), 570  
 PeopleAudience (rdflib.SDO attribute), 513  
 percentage (rdflib.ODRL2 attribute), 458  
 percentile10 (rdflib.SDO attribute), 570  
 percentile25 (rdflib.SDO attribute), 570  
 percentile75 (rdflib.SDO attribute), 570  
 percentile90 (rdflib.SDO attribute), 570  
 PercutaneousProcedure (rdflib.SDO attribute), 513  
 PerformAction (rdflib.SDO attribute), 513  
 PerformanceRole (rdflib.SDO attribute), 513  
 performer (rdflib.SDO attribute), 570  
 performerIn (rdflib.SDO attribute), 570  
 performers (rdflib.SDO attribute), 570  
 PerformingArtsTheater (rdflib.SDO attribute), 513  
 PerformingGroup (rdflib.SDO attribute), 513



- performTime (*rdflib.SDO* attribute), 570
- Period (*rdflib.DCTERMS* attribute), 411
- Periodical (*rdflib.SDO* attribute), 513
- PeriodOfTime (*rdflib.DCTERMS* attribute), 411
- perm (*rdflib.ODRL2* attribute), 458
- Permission (*rdflib.ODRL2* attribute), 454
- permission (*rdflib.ODRL2* attribute), 458
- permissions (*rdflib.SDO* attribute), 570
- permissionType (*rdflib.SDO* attribute), 570
- Permit (*rdflib.SDO* attribute), 513
- permitAudience (*rdflib.SDO* attribute), 570
- permittedUsage (*rdflib.SDO* attribute), 570
- Person (*rdflib.FOAF* attribute), 421
- Person (*rdflib.PROV* attribute), 469
- Person (*rdflib.SDO* attribute), 513
- PersonalProfileDocument (*rdflib.FOAF* attribute), 421
- PET (*rdflib.SDO* attribute), 512
- petsAllowed (*rdflib.SDO* attribute), 570
- PetStore (*rdflib.SDO* attribute), 513
- Pharmacy (*rdflib.SDO* attribute), 513
- PharmacySpecialty (*rdflib.SDO* attribute), 514
- phenomenonTime (*rdflib.SOSA* attribute), 599
- phone (*rdflib.FOAF* attribute), 422
- phoneticText (*rdflib.SDO* attribute), 570
- photo (*rdflib.SDO* attribute), 570
- Photograph (*rdflib.SDO* attribute), 514
- PhotographAction (*rdflib.SDO* attribute), 514
- photos (*rdflib.SDO* attribute), 570
- Photovoltaic\_Array (*rdflib.BRICK* attribute), 380
- Photovoltaic\_Current\_Output\_Sensor (*rdflib.BRICK* attribute), 380
- PhysicalActivity (*rdflib.SDO* attribute), 514
- PhysicalActivityCategory (*rdflib.SDO* attribute), 514
- PhysicalExam (*rdflib.SDO* attribute), 514
- PhysicalMedium (*rdflib.DCTERMS* attribute), 411
- PhysicalObject (*rdflib.DCMITYPE* attribute), 410
- physicalRequirement (*rdflib.SDO* attribute), 570
- PhysicalResource (*rdflib.DCTERMS* attribute), 411
- PhysicalTherapy (*rdflib.SDO* attribute), 514
- Physician (*rdflib.SDO* attribute), 514
- physiologicalBenefits (*rdflib.SDO* attribute), 570
- Physiotherapy (*rdflib.SDO* attribute), 514
- pickupLocation (*rdflib.SDO* attribute), 570
- pickupTime (*rdflib.SDO* attribute), 571
- PID\_Parameter (*rdflib.BRICK* attribute), 379
- Piezoelectric\_Sensor (*rdflib.BRICK* attribute), 380
- pingback (*rdflib.PROV* attribute), 472
- PIR\_Sensor (*rdflib.BRICK* attribute), 379
- PKGPlugin (*class in rdflib.plugin*), 289
- Place (*rdflib.SDO* attribute), 514
- PlaceboControlledTrial (*rdflib.SDO* attribute), 514
- PlaceOfWorship (*rdflib.SDO* attribute), 514
- PlainLiteral (*rdflib.RDF* attribute), 477
- plan (*rdflib.FOAF* attribute), 422
- Plan (*rdflib.PROV* attribute), 469
- PlanAction (*rdflib.SDO* attribute), 514
- PlasticSurgery (*rdflib.SDO* attribute), 514
- platform (*rdflib.DOAP* attribute), 415
- Platform (*rdflib.SOSA* attribute), 597
- play (*rdflib.ODRL2* attribute), 458
- Play (*rdflib.SDO* attribute), 514
- PlayAction (*rdflib.SDO* attribute), 514
- playersOnline (*rdflib.SDO* attribute), 571
- playerType (*rdflib.SDO* attribute), 571
- Playground (*rdflib.SDO* attribute), 514
- playMode (*rdflib.SDO* attribute), 571
- Plugin (*class in rdflib.plugin*), 289
- PluginException, 290
- plugins() (*in module rdflib.plugin*), 290
- PluginT (*in module rdflib.plugin*), 290
- PlugStrip (*rdflib.BRICK* attribute), 380
- Plumber (*rdflib.SDO* attribute), 514
- Plumbing\_Room (*rdflib.BRICK* attribute), 380
- PM10\_Level\_Sensor (*rdflib.BRICK* attribute), 379
- PM10\_Sensor (*rdflib.BRICK* attribute), 379
- PM1\_Level\_Sensor (*rdflib.BRICK* attribute), 379
- PM1\_Sensor (*rdflib.BRICK* attribute), 379
- PodcastEpisode (*rdflib.SDO* attribute), 514
- PodcastSeason (*rdflib.SDO* attribute), 514
- PodcastSeries (*rdflib.SDO* attribute), 514
- Podiatric (*rdflib.SDO* attribute), 514
- Point (*rdflib.BRICK* attribute), 380
- Point (*rdflib.DCTERMS* attribute), 411
- PoliceStation (*rdflib.SDO* attribute), 514
- Policy (*rdflib.DCTERMS* attribute), 411
- Policy (*rdflib.ODRL2* attribute), 454
- policyUsage (*rdflib.ODRL2* attribute), 458
- polygon (*rdflib.SDO* attribute), 571
- Pond (*rdflib.SDO* attribute), 514
- pop() (*rdflib.plugins.serializers.xmlwriter.XMLWriter* method), 123
- populationType (*rdflib.SDO* attribute), 571
- Portfolio (*rdflib.BRICK* attribute), 380
- position (*rdflib.SDO* attribute), 571
- Position\_Command (*rdflib.BRICK* attribute), 380
- Position\_Limit (*rdflib.BRICK* attribute), 380
- Position\_Sensor (*rdflib.BRICK* attribute), 380
- positiveInteger (*rdflib.XSD* attribute), 611
- positiveNotes (*rdflib.SDO* attribute), 571
- possibleComplication (*rdflib.SDO* attribute), 571
- possibleTreatment (*rdflib.SDO* attribute), 571
- Post (*rdflib.ORG* attribute), 461
- PostalAddress (*rdflib.SDO* attribute), 514
- postalCode (*rdflib.SDO* attribute), 571
- postalCodeBegin (*rdflib.SDO* attribute), 571
- postalCodeEnd (*rdflib.SDO* attribute), 571

- postalCodePrefix (*rdflib.SDO* attribute), 571
- postalCodeRange (*rdflib.SDO* attribute), 571
- PostalCodeRangeSpecification (*rdflib.SDO* attribute), 514
- Poster (*rdflib.SDO* attribute), 515
- postIn (*rdflib.ORG* attribute), 462
- PostOffice (*rdflib.SDO* attribute), 514
- postOfficeBoxNumber (*rdflib.SDO* attribute), 571
- postOp (*rdflib.SDO* attribute), 571
- postParse() (*rdflib.plugins.sparql.parserutils.Comp* method), 170
- postParse2() (*rdflib.plugins.sparql.parserutils.Param* method), 172
- Potable\_Water (*rdflib.BRICK* attribute), 380
- potentialAction (*rdflib.SDO* attribute), 571
- PotentialActionStatus (*rdflib.SDO* attribute), 515
- potentialUse (*rdflib.SDO* attribute), 571
- Power\_Alarm (*rdflib.BRICK* attribute), 380
- Power\_Loss\_Alarm (*rdflib.BRICK* attribute), 380
- Power\_Sensor (*rdflib.BRICK* attribute), 381
- powerComplexity (*rdflib.BRICK* attribute), 399
- powerFlow (*rdflib.BRICK* attribute), 399
- pprintAlgebra() (in module *rdflib.plugins.sparql.algebra*), 148
- Prayer\_Room (*rdflib.BRICK* attribute), 381
- Pre\_Filter (*rdflib.BRICK* attribute), 381
- Pre\_Filter\_Status (*rdflib.BRICK* attribute), 381
- predecessorOf (*rdflib.SDO* attribute), 571
- predicate (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 103
- predicate (*rdflib.RDF* attribute), 478
- predicate (*rdflib.SH* attribute), 594
- predicate() (*rdflib.plugins.parsers.ntriples.W3CNTriples* method), 101
- predicate() (*rdflib.plugins.serializers.rdfxml.PrettyXMLSerializer* method), 118
- predicate() (*rdflib.plugins.serializers.rdfxml.XMLSerializer* method), 119
- predicate\_objects() (*rdflib.Graph* method), 434
- predicate\_objects() (*rdflib.graph.Graph* method), 261
- predicate\_objects() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 208
- predicate\_objects() (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 214
- predicate\_objects() (*rdflib.resource.Resource* method), 305
- predicateList() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 116
- predicateList() (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 122
- predicateOrder (*rdflib.plugins.serializers.turtle.RecursiveSerializer* attribute), 121
- predicates() (*rdflib.Graph* method), 435
- predicates() (*rdflib.graph.Graph* method), 262
- predicates() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 208
- predicates() (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 215
- predicates() (*rdflib.resource.Resource* method), 305
- preferredNamespacePrefix (*rdflib.VANN* attribute), 606
- preferredNamespaceUri (*rdflib.VANN* attribute), 606
- prefix (*rdflib.plugins.shared.jsonld.context.Term* attribute), 130
- prefix (*rdflib.SH* attribute), 594
- prefix() (*rdflib.plugins.stores.auditable.AuditableStore* method), 191
- prefix() (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* method), 194
- prefix() (*rdflib.plugins.stores.memory.Memory* method), 198
- prefix() (*rdflib.plugins.stores.memory.SimpleMemory* method), 200
- prefix() (*rdflib.plugins.stores.regexmatching.REGEXMatching* method), 202
- prefix() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 208
- prefix() (*rdflib.store.Store* method), 310
- PrefixDeclaration (*rdflib.SH* attribute), 590
- prefixes (*rdflib.SH* attribute), 594
- prefLabel (*rdflib.SKOS* attribute), 597
- PregnancyCategory (*rdflib.SDO* attribute), 571
- PregnancyHealthAspect (*rdflib.SDO* attribute), 515
- PregnancyWarning (*rdflib.SDO* attribute), 571
- Preheat\_Demand\_Setpoint (*rdflib.BRICK* attribute), 381
- Preheat\_Discharge\_Air\_Temperature\_Sensor (*rdflib.BRICK* attribute), 381
- Preheat\_Hot\_Water\_System (*rdflib.BRICK* attribute), 381
- Preheat\_Hot\_Water\_Valve (*rdflib.BRICK* attribute), 381
- Preheat\_Supply\_Air\_Temperature\_Sensor (*rdflib.BRICK* attribute), 381
- preOp (*rdflib.SDO* attribute), 571
- PreOrder (*rdflib.SDO* attribute), 515
- PreOrderAction (*rdflib.SDO* attribute), 515
- preparation (*rdflib.SDO* attribute), 571
- prepareQuery() (in module *rdflib.plugins.sparql*), 188
- prepareQuery() (in module *rdflib.plugins.sparql.processor*), 175
- prepareUpdate() (in module *rdflib.plugins.sparql*), 188



- prepareUpdate() (in module *rdflib.plugins.sparql.processor*), 175
- PrependAction (*rdflib.SDO* attribute), 515
- preprocess() (*rdflib.plugins.serializers.trig.TrigSerializer* method), 120
- preprocess() (*rdflib.plugins.serializers.turtle.RecursiveSerializer* method), 122
- preprocessTriple() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 116
- preprocessTriple() (*rdflib.plugins.serializers.n3.N3Serializer* method), 117
- preprocessTriple() (*rdflib.plugins.serializers.turtle.RecursiveSerializer* method), 122
- preprocessTriple() (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 122
- prepTime (*rdflib.SDO* attribute), 571
- PreSale (*rdflib.SDO* attribute), 515
- Preschool (*rdflib.SDO* attribute), 515
- prescribingInfo (*rdflib.SDO* attribute), 571
- PrescriptionOnly (*rdflib.SDO* attribute), 515
- prescriptionStatus (*rdflib.SDO* attribute), 571
- present (*rdflib.ODRL2* attribute), 458
- PresentationDigitalDocument (*rdflib.SDO* attribute), 515
- Pressure\_Alarm (*rdflib.BRICK* attribute), 381
- Pressure\_Sensor (*rdflib.BRICK* attribute), 381
- Pressure\_Setpoint (*rdflib.BRICK* attribute), 381
- Pressure\_Status (*rdflib.BRICK* attribute), 381
- prettify\_parsetree() (in module *rdflib.plugins.sparql.parserutils*), 173
- PrettyXMLSerializer (class in *rdflib.plugins.serializers.rdfxml*), 118
- PreventionHealthAspect (*rdflib.SDO* attribute), 515
- PreventionIndication (*rdflib.SDO* attribute), 515
- preview (*rdflib.ODRL2* attribute), 458
- previousItem (*rdflib.SDO* attribute), 572
- previousStartDate (*rdflib.SDO* attribute), 572
- price (*rdflib.SDO* attribute), 572
- priceComponent (*rdflib.SDO* attribute), 572
- priceComponentType (*rdflib.SDO* attribute), 572
- PriceComponentTypeEnumeration (*rdflib.SDO* attribute), 515
- priceCurrency (*rdflib.SDO* attribute), 572
- priceRange (*rdflib.SDO* attribute), 572
- PriceSpecification (*rdflib.SDO* attribute), 515
- priceSpecification (*rdflib.SDO* attribute), 572
- priceType (*rdflib.SDO* attribute), 572
- PriceTypeEnumeration (*rdflib.SDO* attribute), 515
- priceValidUntil (*rdflib.SDO* attribute), 572
- PrimaryCare (*rdflib.SDO* attribute), 515
- primaryImageOfPage (*rdflib.SDO* attribute), 572
- primaryKey (*rdflib.CSVW* attribute), 402
- primaryPrevention (*rdflib.SDO* attribute), 572
- PrimarySource (*rdflib.PROV* attribute), 469
- primaryTopic (*rdflib.FOAF* attribute), 422
- print() (*rdflib.ODRL2* attribute), 458
- print() (*rdflib.Graph* method), 435
- print() (*rdflib.graph.Graph* method), 262
- printColumn (*rdflib.SDO* attribute), 572
- printEdition (*rdflib.SDO* attribute), 572
- printPage (*rdflib.SDO* attribute), 572
- printSection (*rdflib.SDO* attribute), 572
- Prion (*rdflib.SDO* attribute), 515
- priorVersion (*rdflib.OWL* attribute), 466
- Privacy (*rdflib.ODRL2* attribute), 454
- Private\_Office (*rdflib.BRICK* attribute), 381
- procedure (*rdflib.SDO* attribute), 572
- Procedure (*rdflib.SOSA* attribute), 598
- procedureType (*rdflib.SDO* attribute), 572
- processingInstruction() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 106
- processingInstruction() (*rdflib.plugins.parsers.trix.TriXHandler* method), 111
- processingTime (*rdflib.SDO* attribute), 572
- Processor (class in *rdflib.query*), 291
- processorRequirements (*rdflib.SDO* attribute), 572
- processUpdate() (in module *rdflib.plugins.sparql*), 188
- processUpdate() (in module *rdflib.plugins.sparql.processor*), 175
- producer (*rdflib.SDO* attribute), 572
- produces (*rdflib.SDO* attribute), 572
- product (*rdflib.ODRL2* attribute), 458
- Product (*rdflib.SDO* attribute), 515
- ProductCollection (*rdflib.SDO* attribute), 515
- ProductGroup (*rdflib.SDO* attribute), 515
- productGroupID (*rdflib.SDO* attribute), 572
- productID (*rdflib.SDO* attribute), 572
- productionCompany (*rdflib.SDO* attribute), 572
- productionDate (*rdflib.SDO* attribute), 573
- ProductModel (*rdflib.SDO* attribute), 515
- productSupported (*rdflib.SDO* attribute), 572
- PROF (class in *rdflib*), 466
- ProfessionalService (*rdflib.SDO* attribute), 515
- proficiencyLevel (*rdflib.SDO* attribute), 573
- profile (*rdflib.ODRL2* attribute), 459
- Profile (*rdflib.PROF* attribute), 467
- ProfilePage (*rdflib.SDO* attribute), 515
- PrognosisHealthAspect (*rdflib.SDO* attribute), 516
- ProgramMembership (*rdflib.SDO* attribute), 516
- programMembershipUsed (*rdflib.SDO* attribute), 573
- programmingLanguage (*rdflib.SDO* attribute), 573
- programmingModel (*rdflib.SDO* attribute), 573

- `programName` (*rdflib.SDO attribute*), 573
- `programPrerequisites` (*rdflib.SDO attribute*), 573
- `programType` (*rdflib.SDO attribute*), 573
- `prohibit` (*rdflib.ODRL2 attribute*), 459
- `Prohibition` (*rdflib.ODRL2 attribute*), 454
- `prohibition` (*rdflib.ODRL2 attribute*), 459
- `Project` (*rdflib.DOAP attribute*), 414
- `Project` (*rdflib.FOAF attribute*), 421
- `Project` (*rdflib.SDO attribute*), 516
- `Project()` (in module *rdflib.plugins.sparql.algebra*), 147
- `project()` (*rdflib.plugins.sparql.sparql.FrozenBindings method*), 178
- `project()` (*rdflib.plugins.sparql.sparql.FrozenDict method*), 180
- `Prologue` (class in *rdflib.plugins.sparql.sparql*), 181
- `prologue` (*rdflib.plugins.sparql.sparql.FrozenBindings property*), 178
- `PronounceableText` (*rdflib.SDO attribute*), 516
- `prop()` (*rdflib.plugins.parsers.notation3.SinkParser method*), 91
- `ProperInterval` (*rdflib.TIME attribute*), 600
- `properties` (*rdflib.VOID attribute*), 607
- `Property` (class in *rdflib.extras.infixowl*), 67
- `Property` (*rdflib.RDF attribute*), 477
- `Property` (*rdflib.SDO attribute*), 516
- `property` (*rdflib.SH attribute*), 594
- `Property` (*rdflib.SSN attribute*), 599
- `property` (*rdflib.VOID attribute*), 607
- `property_element_char()` (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler method*), 106
- `property_element_end()` (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler method*), 106
- `property_element_start()` (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler method*), 106
- `property_list()` (*rdflib.plugins.parsers.notation3.SinkParser method*), 92
- `propertyChainAxiom` (*rdflib.OWL attribute*), 466
- `PropertyConstraintComponent` (*rdflib.SH attribute*), 590
- `propertyDisjointWith` (*rdflib.OWL attribute*), 466
- `PropertyGroup` (*rdflib.SH attribute*), 590
- `propertyID` (*rdflib.SDO attribute*), 573
- `propertyOrIdentifier()` (in module *rdflib.extras.infixowl*), 69
- `propertyPartition` (*rdflib.VOID attribute*), 607
- `PropertyShape` (*rdflib.SH attribute*), 590
- `propertyUrl` (*rdflib.CSVW attribute*), 402
- `propertyValidator` (*rdflib.SH attribute*), 594
- `PropertyValue` (*rdflib.SDO attribute*), 516
- `PropertyValueSpecification` (*rdflib.SDO attribute*), 516
- `Proportional_Band_Parameter` (*rdflib.BRICK attribute*), 381
- `Proportional_Gain_Parameter` (*rdflib.BRICK attribute*), 381
- `proprietaryName` (*rdflib.SDO attribute*), 573
- `protected` (*rdflib.plugins.shared.jsonld.context.Term attribute*), 130
- `Protein` (*rdflib.SDO attribute*), 516
- `proteinContent` (*rdflib.SDO attribute*), 573
- `Protozoa` (*rdflib.SDO attribute*), 516
- `PROV` (class in *rdflib*), 467
- `provenance` (*rdflib.DCTERMS attribute*), 413
- `ProvenanceStatement` (*rdflib.DCTERMS attribute*), 411
- `provenanceUriTemplate` (*rdflib.PROV attribute*), 472
- `provider` (*rdflib.SDO attribute*), 573
- `providerMobility` (*rdflib.SDO attribute*), 573
- `providesBroadcastService` (*rdflib.SDO attribute*), 573
- `providesService` (*rdflib.SDO attribute*), 573
- `proximity` (*rdflib.ODRL2 attribute*), 459
- `Psychiatric` (*rdflib.SDO attribute*), 516
- `PsychologicalTreatment` (*rdflib.SDO attribute*), 516
- `publicAccess` (*rdflib.SDO attribute*), 573
- `publication` (*rdflib.SDO attribute*), 573
- `PublicationEvent` (*rdflib.SDO attribute*), 516
- `PublicationIssue` (*rdflib.SDO attribute*), 516
- `publications` (*rdflib.FOAF attribute*), 422
- `publicationType` (*rdflib.SDO attribute*), 573
- `PublicationVolume` (*rdflib.SDO attribute*), 516
- `PublicHealth` (*rdflib.SDO attribute*), 516
- `PublicHolidays` (*rdflib.SDO attribute*), 516
- `PublicSwimmingPool` (*rdflib.SDO attribute*), 516
- `PublicToilet` (*rdflib.SDO attribute*), 516
- `publicTransportClosuresInfo` (*rdflib.SDO attribute*), 573
- `Publish` (*rdflib.PROV attribute*), 469
- `publishedBy` (*rdflib.SDO attribute*), 573
- `publishedOn` (*rdflib.SDO attribute*), 573
- `publisher` (*rdflib.DC attribute*), 407
- `publisher` (*rdflib.DCTERMS attribute*), 413
- `Publisher` (*rdflib.PROV attribute*), 469
- `publisher` (*rdflib.SDO attribute*), 573
- `publisherImprint` (*rdflib.SDO attribute*), 573
- `publishingPrinciples` (*rdflib.SDO attribute*), 573
- `Pulmonary` (*rdflib.SDO attribute*), 516
- `Pump` (*rdflib.BRICK attribute*), 381
- `Pump_Command` (*rdflib.BRICK attribute*), 381
- `Pump_On_Off_Status` (*rdflib.BRICK attribute*), 381
- `Pump_Room` (*rdflib.BRICK attribute*), 382
- `Pump_VFD` (*rdflib.BRICK attribute*), 382
- `purchaseDate` (*rdflib.SDO attribute*), 574

- `purpose` (*rdflib.ODRL2* attribute), 459
- `purpose` (*rdflib.ORG* attribute), 462
- `push()` (*rdflib.plugins.serializers.xmlwriter.XMLWriter* method), 123
- `push()` (*rdflib.plugins.sparql.sparql.QueryContext* method), 184
- `pushGraph()` (*rdflib.plugins.sparql.sparql.QueryContext* method), 184
- `PV_Array` (*rdflib.BRICK* attribute), 379
- `PV_Current_Output_Sensor` (*rdflib.BRICK* attribute), 379
- `PV_Generation_System` (*rdflib.BRICK* attribute), 379
- `PV_Panel` (*rdflib.BRICK* attribute), 379
- `PVT_Panel` (*rdflib.BRICK* attribute), 379
- Python Enhancement Proposals  
PEP 8, 617
- `PythonInputSource` (class in *rdflib.parser*), 277
- Q**
- `QAPage` (*rdflib.SDO* attribute), 516
- `QB` (class in *rdflib*), 475
- `QName` (*rdflib.XSD* attribute), 609
- `qname()` (*rdflib.Graph* method), 435
- `qname()` (*rdflib.graph.Graph* method), 262
- `qname()` (*rdflib.graph.ReadOnlyGraphAggregate* method), 273
- `qname()` (*rdflib.namespace.NamespaceManager* method), 78
- `qname()` (*rdflib.plugins.parsers.notation3.SinkParser* method), 92
- `qname()` (*rdflib.plugins.serializers.xmlwriter.XMLWriter* method), 123
- `qname()` (*rdflib.resource.Resource* method), 305
- `qname_strict()` (*rdflib.namespace.NamespaceManager* method), 78
- `quads()` (*rdflib.ConjunctiveGraph* method), 406
- `quads()` (*rdflib.Dataset* method), 420
- `quads()` (*rdflib.graph.ConjunctiveGraph* method), 244
- `quads()` (*rdflib.graph.Dataset* method), 249
- `quads()` (*rdflib.graph.ReadOnlyGraphAggregate* method), 273
- `qualifications` (*rdflib.SDO* attribute), 574
- `qualifiedAssociation` (*rdflib.PROV* attribute), 472
- `qualifiedAssociationOf` (*rdflib.PROV* attribute), 472
- `qualifiedAttribution` (*rdflib.PROV* attribute), 472
- `qualifiedAttributionOf` (*rdflib.PROV* attribute), 472
- `qualifiedCardinality` (*rdflib.OWL* attribute), 466
- `qualifiedCommunication` (*rdflib.PROV* attribute), 472
- `qualifiedCommunicationOf` (*rdflib.PROV* attribute), 472
- `qualifiedDelegation` (*rdflib.PROV* attribute), 473
- `qualifiedDelegationOf` (*rdflib.PROV* attribute), 473
- `qualifiedDerivation` (*rdflib.PROV* attribute), 473
- `qualifiedDerivationOf` (*rdflib.PROV* attribute), 473
- `qualifiedEnd` (*rdflib.PROV* attribute), 473
- `qualifiedEndOf` (*rdflib.PROV* attribute), 473
- `qualifiedForm` (*rdflib.PROV* attribute), 473
- `qualifiedGeneration` (*rdflib.PROV* attribute), 473
- `qualifiedGenerationOf` (*rdflib.PROV* attribute), 473
- `qualifiedInfluence` (*rdflib.PROV* attribute), 473
- `qualifiedInfluenceOf` (*rdflib.PROV* attribute), 473
- `qualifiedInsertion` (*rdflib.PROV* attribute), 473
- `qualifiedInvalidation` (*rdflib.PROV* attribute), 473
- `qualifiedInvalidationOf` (*rdflib.PROV* attribute), 473
- `qualifiedMaxCount` (*rdflib.SH* attribute), 594
- `QualifiedMaxCountConstraintComponent` (*rdflib.SH* attribute), 590
- `qualifiedMinCount` (*rdflib.SH* attribute), 594
- `QualifiedMinCountConstraintComponent` (*rdflib.SH* attribute), 590
- `qualifiedPrimarySource` (*rdflib.PROV* attribute), 473
- `qualifiedQuotation` (*rdflib.PROV* attribute), 473
- `qualifiedQuotationOf` (*rdflib.PROV* attribute), 473
- `qualifiedRelation` (*rdflib.DCAT* attribute), 409
- `qualifiedRemoval` (*rdflib.PROV* attribute), 473
- `qualifiedRevision` (*rdflib.PROV* attribute), 473
- `qualifiedSourceOf` (*rdflib.PROV* attribute), 474
- `qualifiedStart` (*rdflib.PROV* attribute), 474
- `qualifiedStartOf` (*rdflib.PROV* attribute), 474
- `qualifiedUsage` (*rdflib.PROV* attribute), 474
- `qualifiedUsingActivity` (*rdflib.PROV* attribute), 474
- `qualifiedValueShape` (*rdflib.SH* attribute), 594
- `qualifiedValueShapesDisjoint` (*rdflib.SH* attribute), 594
- `QualitativeValue` (*rdflib.SDO* attribute), 516
- `QuantitativeValue` (*rdflib.SDO* attribute), 516
- `QuantitativeValueDistribution` (*rdflib.SDO* attribute), 516
- `Quantity` (*rdflib.BRICK* attribute), 382
- `Quantity` (*rdflib.SDO* attribute), 517
- `quarantineGuidelines` (*rdflib.SDO* attribute), 574
- `Query` (class in *rdflib.plugins.sparql.sparql*), 181
- `query` (*rdflib.SDO* attribute), 574
- `query()` (*rdflib.Graph* method), 435
- `query()` (*rdflib.graph.Graph* method), 262
- `query()` (*rdflib.plugins.sparql.processor.SPARQLProcessor* method), 174
- `query()` (*rdflib.plugins.stores.auditable.AuditableStore* method), 191
- `query()` (*rdflib.plugins.stores.memory.Memory* method), 198
- `query()` (*rdflib.plugins.stores.memory.SimpleMemory* method), 200
- `query()` (*rdflib.plugins.stores.sparqlconnector.SPARQLConnector* method), 204
- `query()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 209

- `query()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 215
- `query()` (*rdflib.query.Processor* method), 292
- `query()` (*rdflib.store.Store* method), 311
- `QueryContext` (class in *rdflib.plugins.sparql.sparql*), 182
- `quest` (*rdflib.SDO* attribute), 574
- `Question` (*rdflib.SDO* attribute), 517
- `question` (*rdflib.SDO* attribute), 574
- `Quiz` (*rdflib.SDO* attribute), 517
- `Quotation` (*rdflib.PROV* attribute), 469
- `Quotation` (*rdflib.SDO* attribute), 517
- `QuoteAction` (*rdflib.SDO* attribute), 517
- `quoteChar` (*rdflib.CSVW* attribute), 402
- `quotedAs` (*rdflib.PROV* attribute), 474
- `QuotedGraph` (class in *rdflib.graph*), 268
- R**
  - `Radiant_Ceiling_Panel` (*rdflib.BRICK* attribute), 382
  - `Radiant_Panel` (*rdflib.BRICK* attribute), 382
  - `Radiant_Panel_Temperature_Sensor` (*rdflib.BRICK* attribute), 382
  - `Radiant_Panel_Temperature_Setpoint` (*rdflib.BRICK* attribute), 382
  - `Radiation_Hot_Water_System` (*rdflib.BRICK* attribute), 382
  - `RadiationTherapy` (*rdflib.SDO* attribute), 517
  - `Radiator` (*rdflib.BRICK* attribute), 382
  - `Radioactivity_Concentration_Sensor` (*rdflib.BRICK* attribute), 382
  - `RadioBroadcastService` (*rdflib.SDO* attribute), 517
  - `RadioChannel` (*rdflib.SDO* attribute), 517
  - `RadioClip` (*rdflib.SDO* attribute), 517
  - `RadioEpisode` (*rdflib.SDO* attribute), 517
  - `Radiography` (*rdflib.SDO* attribute), 517
  - `RadioSeason` (*rdflib.SDO* attribute), 517
  - `RadioSeries` (*rdflib.SDO* attribute), 517
  - `RadioStation` (*rdflib.SDO* attribute), 517
  - `Radon_Concentration_Sensor` (*rdflib.BRICK* attribute), 382
  - `Rain_Duration_Sensor` (*rdflib.BRICK* attribute), 382
  - `Rain_Sensor` (*rdflib.BRICK* attribute), 382
  - `RandomizedTrial` (*rdflib.SDO* attribute), 517
  - `range` (*rdflib.extras.infixowl.Property* property), 67
  - `range` (*rdflib.RDFS* attribute), 479
  - `rangeIncludes` (*rdflib.SDO* attribute), 574
  - `Rated_Speed_Setpoint` (*rdflib.BRICK* attribute), 382
  - `ratedModuleConversionEfficiency` (*rdflib.BRICK* attribute), 399
  - `ratedPowerOutput` (*rdflib.BRICK* attribute), 399
  - `Rating` (*rdflib.SDO* attribute), 517
  - `ratingCount` (*rdflib.SDO* attribute), 574
  - `ratingExplanation` (*rdflib.SDO* attribute), 574
  - `ratingValue` (*rdflib.SDO* attribute), 574
  - `Stational` (*rdflib.OWL* attribute), 466
  - `RC_Panel` (*rdflib.BRICK* attribute), 382
  - `RDF` (class in *rdflib*), 477
  - `RDF` (*rdflib.plugins.parsers.RDFVOC.RDFVOC* attribute), 79
  - `rdf2dot()` (in module *rdflib.tools.rdf2dot*), 221
  - `rdflib`
    - module, 334
  - `rdflib.collection`
    - module, 221
  - `rdflib.compare`
    - module, 226
  - `rdflib.compat`
    - module, 230
  - `rdflib.container`
    - module, 230
  - `rdflib.events`
    - module, 233
  - `rdflib.exceptions`
    - module, 235
  - `rdflib.extras`
    - module, 69
  - `rdflib.extras.cmdlineutils`
    - module, 47
  - `rdflib.extras.describer`
    - module, 47
  - `rdflib.extras.external_graph_libs`
    - module, 51
  - `rdflib.extras.infixowl`
    - module, 55
  - `rdflib.graph`
    - module, 236
  - `rdflib.namespace`
    - module, 69
  - `rdflib.parser`
    - module, 276
  - `rdflib.paths`
    - module, 279
  - `rdflib.plugin`
    - module, 288
  - `rdflib.plugins`
    - module, 218
  - `rdflib.plugins.parsers`
    - module, 113
  - `rdflib.plugins.parsers.hexst`
    - module, 79
  - `rdflib.plugins.parsers.jsonld`
    - module, 80
  - `rdflib.plugins.parsers.notation3`
    - module, 81
  - `rdflib.plugins.parsers.nquads`
    - module, 97
  - `rdflib.plugins.parsers.ntriples`
    - module, 98



---

rdflib.plugins.parsers.RDFVOC	rdflib.plugins.sparql.evalutils
module, 79	module, 156
rdflib.plugins.parsers.rdfxml	rdflib.plugins.sparql.operators
module, 102	module, 156
rdflib.plugins.parsers.trig	rdflib.plugins.sparql.parser
module, 108	module, 168
rdflib.plugins.parsers.trix	rdflib.plugins.sparql.parserutils
module, 110	module, 169
rdflib.plugins.serializers	rdflib.plugins.sparql.processor
module, 124	module, 174
rdflib.plugins.serializers.hext	rdflib.plugins.sparql.results
module, 113	module, 139
rdflib.plugins.serializers.jsonld	rdflib.plugins.sparql.results.csvresults
module, 114	module, 131
rdflib.plugins.serializers.longturtle	rdflib.plugins.sparql.results.graph
module, 115	module, 133
rdflib.plugins.serializers.n3	rdflib.plugins.sparql.results.jsonresults
module, 116	module, 133
rdflib.plugins.serializers.nquads	rdflib.plugins.sparql.results.rdfresults
module, 117	module, 135
rdflib.plugins.serializers.nt	rdflib.plugins.sparql.results.tsvresults
module, 118	module, 135
rdflib.plugins.serializers.rdfxml	rdflib.plugins.sparql.results.txtresults
module, 118	module, 136
rdflib.plugins.serializers.trig	rdflib.plugins.sparql.results.xmlresults
module, 120	module, 136
rdflib.plugins.serializers.trix	rdflib.plugins.sparql.sparql
module, 120	module, 176
rdflib.plugins.serializers.turtle	rdflib.plugins.sparql.update
module, 121	module, 185
rdflib.plugins.serializers.xmlwriter	rdflib.plugins.stores
module, 123	module, 218
rdflib.plugins.shared	rdflib.plugins.stores.auditable
module, 131	module, 189
rdflib.plugins.shared.jsonld	rdflib.plugins.stores.berkeleydb
module, 131	module, 192
rdflib.plugins.shared.jsonld.context	rdflib.plugins.stores.concurrent
module, 124	module, 195
rdflib.plugins.shared.jsonld.errors	rdflib.plugins.stores.memory
module, 130	module, 196
rdflib.plugins.shared.jsonld.keys	rdflib.plugins.stores.regexmatching
module, 130	module, 201
rdflib.plugins.shared.jsonld.util	rdflib.plugins.stores.sparqlconnector
module, 130	module, 203
rdflib.plugins.sparql	rdflib.plugins.stores.sparqlstore
module, 188	module, 205
rdflib.plugins.sparql.aggregates	rdflib.query
module, 139	module, 291
rdflib.plugins.sparql.algebra	rdflib.resource
module, 145	module, 298
rdflib.plugins.sparql.datatypes	rdflib.serializer
module, 152	module, 305
rdflib.plugins.sparql.evaluate	rdflib.store
module, 152	module, 306

rdflib.term  
     module, 313  
 rdflib.tools  
     module, 221  
 rdflib.tools.chunk\_serializer  
     module, 218  
 rdflib.tools.csv2rdf  
     module, 219  
 rdflib.tools.defined\_namespace\_creator  
     module, 219  
 rdflib.tools.graphisomorphism  
     module, 220  
 rdflib.tools.rdf2dot  
     module, 221  
 rdflib.tools.rdfpipe  
     module, 221  
 rdflib.tools.rdfs2dot  
     module, 221  
 rdflib.util  
     module, 330  
 rdflib.void  
     module, 334  
 rdflib\_to\_graphtool() (in module *rdflib.extras.external\_graph\_libs*), 51  
 rdflib\_to\_networkx\_digraph() (in module *rdflib.extras.external\_graph\_libs*), 52  
 rdflib\_to\_networkx\_graph() (in module *rdflib.extras.external\_graph\_libs*), 53  
 rdflib\_to\_networkx\_multidigraph() (in module *rdflib.extras.external\_graph\_libs*), 54  
 RDFResult (class in *rdflib.plugins.sparql.results.rdfresults*), 135  
 RDFResultParser (class in *rdflib.plugins.sparql.results.rdfresults*), 135  
 RDFS (class in *rdflib*), 478  
 rdfs2dot() (in module *rdflib.tools.rdfs2dot*), 221  
 RDFSink (class in *rdflib.plugins.parsers.notation3*), 84  
 rdftype() (*rdflib.extras.describer.Describer* method), 50  
 RDFVOC (class in *rdflib.plugins.parsers.RDFVOC*), 79  
 RDFXMLHandler (class in *rdflib.plugins.parsers.rdfxml*), 103  
 RDFXMLParser (class in *rdflib.plugins.parsers.rdfxml*), 107  
 ReactAction (*rdflib.SDO* attribute), 517  
 Reactive\_Power\_Sensor (*rdflib.BRICK* attribute), 382  
 read (*rdflib.ODRL2* attribute), 459  
 ReadAction (*rdflib.SDO* attribute), 517  
 readBy (*rdflib.SDO* attribute), 574  
 readline() (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* method), 101  
 ReadOnlyGraphAggregate (class in *rdflib.graph*), 269  
 readonlyValue (*rdflib.SDO* attribute), 574  
 ReadPermission (*rdflib.SDO* attribute), 517  
 real (*rdflib.OWL* attribute), 466  
 RealEstateAgent (*rdflib.SDO* attribute), 517  
 realEstateAgent (*rdflib.SDO* attribute), 574  
 RealEstateListing (*rdflib.SDO* attribute), 517  
 RearWheelDriveConfiguration (*rdflib.SDO* attribute), 517  
 ReceiveAction (*rdflib.SDO* attribute), 517  
 Reception (*rdflib.BRICK* attribute), 382  
 Recipe (*rdflib.SDO* attribute), 517  
 recipe (*rdflib.SDO* attribute), 574  
 recipeCategory (*rdflib.SDO* attribute), 574  
 recipeCuisine (*rdflib.SDO* attribute), 574  
 recipeIngredient (*rdflib.SDO* attribute), 574  
 recipeInstructions (*rdflib.SDO* attribute), 574  
 recipeYield (*rdflib.SDO* attribute), 574  
 recipient (*rdflib.ODRL2* attribute), 459  
 recipient (*rdflib.SDO* attribute), 574  
 recognizedBy (*rdflib.SDO* attribute), 574  
 recognizingAuthority (*rdflib.SDO* attribute), 574  
 Recommendation (*rdflib.SDO* attribute), 517  
 recommendationStrength (*rdflib.SDO* attribute), 574  
 RecommendedDoseSchedule (*rdflib.SDO* attribute), 517  
 recommendedIntake (*rdflib.SDO* attribute), 574  
 record (*rdflib.DCAT* attribute), 409  
 recordedAs (*rdflib.SDO* attribute), 574  
 recordedAt (*rdflib.SDO* attribute), 575  
 recordedIn (*rdflib.SDO* attribute), 575  
 recordingOf (*rdflib.SDO* attribute), 575  
 recordLabel (*rdflib.SDO* attribute), 574  
 recourseLoan (*rdflib.SDO* attribute), 575  
 Recruiting (*rdflib.SDO* attribute), 518  
 RecursiveSerializer (class in *rdflib.plugins.serializers.turtle*), 121  
 RecyclingCenter (*rdflib.SDO* attribute), 518  
 reference (*rdflib.CSVW* attribute), 402  
 referencedRow (*rdflib.CSVW* attribute), 402  
 referenceQuantity (*rdflib.SDO* attribute), 575  
 references (*rdflib.DCTERMS* attribute), 413  
 referencesOrder (*rdflib.SDO* attribute), 575  
 refinement (*rdflib.ODRL2* attribute), 459  
 ReflexiveProperty (*rdflib.OWL* attribute), 464  
 refundType (*rdflib.SDO* attribute), 575  
 RefundTypeEnumeration (*rdflib.SDO* attribute), 518  
 RefurbishedCondition (*rdflib.SDO* attribute), 518  
 regex\_matching (*rdflib.plugins.stores.sparqlstore.SPARQLStore* attribute), 209  
 regexCompareQuad() (in module *rdflib.plugins.stores.regexmatching*), 203  
 REGEXMatching (class in *rdflib.plugins.stores.regexmatching*), 201  
 REGEXTerm (class in *rdflib.plugins.stores.regexmatching*), 203  
 Region (*rdflib.BRICK* attribute), 382  
 regionDrained (*rdflib.SDO* attribute), 575

- regionsAllowed (*rdflib.SDO* attribute), 575
- register() (in module *rdflib.plugin*), 290
- register() (*rdflib.store.NodePickler* method), 307
- register\_custom\_function() (in module *rdflib.plugins.sparql.operators*), 167
- RegisterAction (*rdflib.SDO* attribute), 518
- Registry (*rdflib.SDO* attribute), 518
- regulates (*rdflib.BRICK* attribute), 400
- Reheat\_Hot\_Water\_System (*rdflib.BRICK* attribute), 383
- Reheat\_Valve (*rdflib.BRICK* attribute), 383
- ReimbursementCap (*rdflib.SDO* attribute), 518
- RejectAction (*rdflib.SDO* attribute), 518
- rel() (*rdflib.extras.describer.Describer* method), 50
- related (*rdflib.SKOS* attribute), 597
- relatedAnatomy (*rdflib.SDO* attribute), 575
- relatedCondition (*rdflib.SDO* attribute), 575
- relatedDrug (*rdflib.SDO* attribute), 575
- relatedLink (*rdflib.SDO* attribute), 575
- relatedMatch (*rdflib.SKOS* attribute), 597
- relatedStructure (*rdflib.SDO* attribute), 575
- relatedTherapy (*rdflib.SDO* attribute), 575
- relatedTo (*rdflib.SDO* attribute), 575
- RelatedTopicsHealthAspect (*rdflib.SDO* attribute), 518
- relation (*rdflib.DC* attribute), 407
- relation (*rdflib.DCTERMS* attribute), 413
- relation (*rdflib.ODRL2* attribute), 459
- RelationalExpression() (in module *rdflib.plugins.sparql.operators*), 164
- Relationship (*rdflib.DCAT* attribute), 408
- Relative\_Humidity\_Sensor (*rdflib.BRICK* attribute), 383
- relativePosition (*rdflib.ODRL2* attribute), 459
- relativeSize (*rdflib.ODRL2* attribute), 459
- relativeSpatialPosition (*rdflib.ODRL2* attribute), 459
- relativeTemporalPosition (*rdflib.ODRL2* attribute), 459
- relativize() (*rdflib.serializer.Serializer* method), 306
- release (*rdflib.DOAP* attribute), 415
- releaseDate (*rdflib.SDO* attribute), 575
- releasedEvent (*rdflib.SDO* attribute), 575
- releaseNotes (*rdflib.SDO* attribute), 575
- releaseOf (*rdflib.SDO* attribute), 575
- relevantOccupation (*rdflib.SDO* attribute), 575
- relevantSpecialty (*rdflib.SDO* attribute), 575
- Relief\_Damper (*rdflib.BRICK* attribute), 383
- Relief\_Fan (*rdflib.BRICK* attribute), 383
- remainingAttendeeCapacity (*rdflib.SDO* attribute), 575
- remedy (*rdflib.ODRL2* attribute), 459
- remember() (*rdflib.plugins.sparql.sparql.FrozenBindings* method), 178
- RemixAlbum (*rdflib.SDO* attribute), 518
- Remotely\_On\_Off\_Status (*rdflib.BRICK* attribute), 383
- Removal (*rdflib.PROV* attribute), 469
- remove() (*rdflib.ConjunctiveGraph* method), 406
- remove() (*rdflib.Graph* method), 436
- remove() (*rdflib.graph.ConjunctiveGraph* method), 244
- remove() (*rdflib.graph.Graph* method), 263
- remove() (*rdflib.graph.ReadOnlyGraphAggregate* method), 273
- remove() (*rdflib.plugins.stores.auditable.AuditableStore* method), 191
- remove() (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* method), 194
- remove() (*rdflib.plugins.stores.concurrent.ConcurrentStore* method), 195
- remove() (*rdflib.plugins.stores.memory.Memory* method), 198
- remove() (*rdflib.plugins.stores.memory.SimpleMemory* method), 201
- remove() (*rdflib.plugins.stores.regexmatching.REGEXMatching* method), 202
- remove() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 209
- remove() (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 215
- remove() (*rdflib.resource.Resource* method), 305
- remove() (*rdflib.store.Store* method), 311
- remove\_context() (*rdflib.ConjunctiveGraph* method), 406
- remove\_context() (*rdflib.graph.ConjunctiveGraph* method), 244
- remove\_context() (*rdflib.plugins.stores.regexmatching.REGEXMatching* method), 202
- remove\_graph() (*rdflib.Dataset* method), 420
- remove\_graph() (*rdflib.graph.Dataset* method), 249
- remove\_graph() (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* method), 194
- remove\_graph() (*rdflib.plugins.stores.memory.Memory* method), 198
- remove\_graph() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 209
- remove\_graph() (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 215
- remove\_graph() (*rdflib.store.Store* method), 311
- removedKey (*rdflib.PROV* attribute), 474
- remuneration (*rdflib.ORG* attribute), 462
- Renal (*rdflib.SDO* attribute), 518
- renegotiableLoan (*rdflib.SDO* attribute), 575
- RentAction (*rdflib.SDO* attribute), 518
- RentalCarReservation (*rdflib.SDO* attribute), 518
- RentalVehicleUsage (*rdflib.SDO* attribute), 518
- reorderTriples() (in module *rd-*

- flib.plugins.sparql.algebra*), 148
- RepaymentSpecification (*rdflib.SDO* attribute), 518
- repeatCount (*rdflib.SDO* attribute), 575
- repeatFrequency (*rdflib.SDO* attribute), 575
- repetitions (*rdflib.SDO* attribute), 576
- Replace (*rdflib.PROV* attribute), 469
- replace() (*rdflib.extras.infixowl.Individual* method), 65
- replace() (*rdflib.extras.infixowl.Property* method), 67
- ReplaceAction (*rdflib.SDO* attribute), 518
- replacee (*rdflib.SDO* attribute), 576
- replacer (*rdflib.SDO* attribute), 576
- replaces (*rdflib.DCTERMS* attribute), 413
- ReplyAction (*rdflib.SDO* attribute), 518
- replyToUrl (*rdflib.SDO* attribute), 576
- Report (*rdflib.SDO* attribute), 518
- ReportageNewsArticle (*rdflib.SDO* attribute), 518
- ReportedDoseSchedule (*rdflib.SDO* attribute), 518
- reportNumber (*rdflib.SDO* attribute), 576
- reportsTo (*rdflib.ORG* attribute), 462
- Repository (*rdflib.DOAP* attribute), 414
- repository (*rdflib.DOAP* attribute), 415
- repositoryOf (*rdflib.DOAP* attribute), 415
- representativeOfPage (*rdflib.SDO* attribute), 576
- reproduce (*rdflib.ODRL2* attribute), 459
- Request (*rdflib.ODRL2* attribute), 454
- required (*rdflib.CSVW* attribute), 402
- requiredCollateral (*rdflib.SDO* attribute), 576
- requiredGender (*rdflib.SDO* attribute), 576
- requiredMaxAge (*rdflib.SDO* attribute), 576
- requiredMinAge (*rdflib.SDO* attribute), 576
- requiredQuantity (*rdflib.SDO* attribute), 576
- requirements (*rdflib.SDO* attribute), 576
- requires (*rdflib.DCTERMS* attribute), 413
- requiresSubscription (*rdflib.SDO* attribute), 576
- Researcher (*rdflib.SDO* attribute), 518
- ResearchOrganization (*rdflib.SDO* attribute), 518
- ResearchProject (*rdflib.SDO* attribute), 518
- Reservation (*rdflib.SDO* attribute), 518
- ReservationCancelled (*rdflib.SDO* attribute), 519
- ReservationConfirmed (*rdflib.SDO* attribute), 519
- reservationFor (*rdflib.SDO* attribute), 576
- ReservationHold (*rdflib.SDO* attribute), 519
- reservationId (*rdflib.SDO* attribute), 576
- ReservationPackage (*rdflib.SDO* attribute), 519
- ReservationPending (*rdflib.SDO* attribute), 519
- reservationStatus (*rdflib.SDO* attribute), 576
- ReservationStatusType (*rdflib.SDO* attribute), 519
- ReserveAction (*rdflib.SDO* attribute), 519
- reservedTicket (*rdflib.SDO* attribute), 576
- Reservoir (*rdflib.SDO* attribute), 519
- reset() (*rdflib.graph.BatchAddGraph* method), 241
- reset() (*rdflib.namespace.NamespaceManager* method), 78
- reset() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 106
- reset() (*rdflib.plugins.parsers.trix.TriXHandler* method), 111
- reset() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 116
- reset() (*rdflib.plugins.serializers.n3.N3Serializer* method), 117
- reset() (*rdflib.plugins.serializers.trig.TrigSerializer* method), 120
- reset() (*rdflib.plugins.serializers.turtle.RecursiveSerializer* method), 122
- reset() (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 123
- Reset\_Command (*rdflib.BRICK* attribute), 383
- Reset\_Setpoint (*rdflib.BRICK* attribute), 383
- Residence (*rdflib.SDO* attribute), 519
- resolution (*rdflib.ODRL2* attribute), 459
- resolve() (*rdflib.plugins.shared.jsonld.context.Context* method), 128
- resolve\_iri() (*rdflib.plugins.shared.jsonld.context.Context* method), 128
- resolvePName() (*rdflib.plugins.sparql.sparql.Prologue* method), 181
- Resort (*rdflib.SDO* attribute), 519
- Resource (class in *rdflib.resource*), 303
- resource (*rdflib.CSVW* attribute), 402
- Resource (*rdflib.DCAT* attribute), 408
- resource (*rdflib.plugins.parsers.RDFVOC.RDFVOC* attribute), 79
- Resource (*rdflib.RDFS* attribute), 479
- resource() (*rdflib.Graph* method), 436
- resource() (*rdflib.graph.Graph* method), 263
- ResourceDescriptor (*rdflib.PROF* attribute), 467
- ResourceRole (*rdflib.PROF* attribute), 467
- RespiratoryTherapy (*rdflib.SDO* attribute), 519
- responsibilities (*rdflib.SDO* attribute), 576
- ResponsibleGenerator (class in *rdflib.plugins.stores.concurrent*), 196
- rest (*rdflib.RDF* attribute), 478
- Rest\_Room (*rdflib.BRICK* attribute), 383
- Restaurant (*rdflib.SDO* attribute), 519
- restockingFee (*rdflib.SDO* attribute), 576
- RestockingFees (*rdflib.SDO* attribute), 519
- restPeriods (*rdflib.SDO* attribute), 576
- RestrictedDiet (*rdflib.SDO* attribute), 519
- Restriction (class in *rdflib.extras.infixowl*), 68
- Restriction (*rdflib.OWL* attribute), 464
- restrictionKind() (*rdflib.extras.infixowl.Restriction* method), 68
- restrictionKinds (*rdflib.extras.infixowl.Restriction* attribute), 68
- Restroom (*rdflib.BRICK* attribute), 383
- Result (class in *rdflib.query*), 292



- `result` (*rdflib.SDO attribute*), 576
- `result` (*rdflib.SH attribute*), 594
- `Result` (*rdflib.SOSA attribute*), 598
- `ResultAnnotation` (*rdflib.SH attribute*), 590
- `resultAnnotation` (*rdflib.SH attribute*), 594
- `resultComment` (*rdflib.SDO attribute*), 576
- `resultedFrom` (*rdflib.ORG attribute*), 462
- `ResultException`, 294
- `resultingOrganization` (*rdflib.ORG attribute*), 462
- `resultMessage` (*rdflib.SH attribute*), 594
- `ResultParser` (*class in rdflib.query*), 294
- `resultPath` (*rdflib.SH attribute*), 594
- `resultReview` (*rdflib.SDO attribute*), 576
- `ResultRow` (*class in rdflib.query*), 295
- `ResultsAvailable` (*rdflib.SDO attribute*), 519
- `ResultSerializer` (*class in rdflib.query*), 297
- `resultSeverity` (*rdflib.SH attribute*), 594
- `ResultsNotAvailable` (*rdflib.SDO attribute*), 519
- `resultTime` (*rdflib.SOSA attribute*), 599
- `ResumeAction` (*rdflib.SDO attribute*), 519
- `Retail` (*rdflib.SDO attribute*), 519
- `Retail_Room` (*rdflib.BRICK attribute*), 383
- `Return_Air` (*rdflib.BRICK attribute*), 383
- `Return_Air_CO2_Sensor` (*rdflib.BRICK attribute*), 383
- `Return_Air_CO2_Setpoint` (*rdflib.BRICK attribute*), 383
- `Return_Air_CO_Sensor` (*rdflib.BRICK attribute*), 383
- `Return_Air_Dewpoint_Sensor` (*rdflib.BRICK attribute*), 383
- `Return_Air_Differential_Pressure_Sensor` (*rdflib.BRICK attribute*), 383
- `Return_Air_Differential_Pressure_Setpoint` (*rdflib.BRICK attribute*), 383
- `Return_Air_Enthalpy_Sensor` (*rdflib.BRICK attribute*), 383
- `Return_Air_Filter` (*rdflib.BRICK attribute*), 384
- `Return_Air_Flow_Sensor` (*rdflib.BRICK attribute*), 384
- `Return_Air_Grains_Sensor` (*rdflib.BRICK attribute*), 384
- `Return_Air_Humidity_Sensor` (*rdflib.BRICK attribute*), 384
- `Return_Air_Humidity_Setpoint` (*rdflib.BRICK attribute*), 384
- `Return_Air_Plenum` (*rdflib.BRICK attribute*), 384
- `Return_Air_Temperature_Alarm` (*rdflib.BRICK attribute*), 384
- `Return_Air_Temperature_High_Reset_Setpoint` (*rdflib.BRICK attribute*), 384
- `Return_Air_Temperature_Low_Reset_Setpoint` (*rdflib.BRICK attribute*), 384
- `Return_Air_Temperature_Sensor` (*rdflib.BRICK attribute*), 384
- `Return_Air_Temperature_Setpoint` (*rdflib.BRICK attribute*), 384
- `Return_Chilled_Water_Temperature_Setpoint` (*rdflib.BRICK attribute*), 384
- `Return_Condenser_Water` (*rdflib.BRICK attribute*), 384
- `Return_Condenser_Water_Flow_Sensor` (*rdflib.BRICK attribute*), 384
- `Return_Condenser_Water_Temperature_Sensor` (*rdflib.BRICK attribute*), 384
- `Return_Condenser_Water_Temperature_Setpoint` (*rdflib.BRICK attribute*), 384
- `Return_Damper` (*rdflib.BRICK attribute*), 384
- `Return_Fan` (*rdflib.BRICK attribute*), 384
- `Return_Heating_Valve` (*rdflib.BRICK attribute*), 384
- `Return_Hot_Water` (*rdflib.BRICK attribute*), 385
- `Return_Hot_Water_Temperature_Setpoint` (*rdflib.BRICK attribute*), 385
- `Return_Water` (*rdflib.BRICK attribute*), 385
- `Return_Water_Flow_Sensor` (*rdflib.BRICK attribute*), 385
- `Return_Water_Temperature_Sensor` (*rdflib.BRICK attribute*), 385
- `Return_Water_Temperature_Setpoint` (*rdflib.BRICK attribute*), 385
- `ReturnAction` (*rdflib.SDO attribute*), 519
- `ReturnAtKiosk` (*rdflib.SDO attribute*), 519
- `ReturnByMail` (*rdflib.SDO attribute*), 519
- `returnFees` (*rdflib.SDO attribute*), 576
- `ReturnFeesCustomerResponsibility` (*rdflib.SDO attribute*), 519
- `ReturnFeesEnumeration` (*rdflib.SDO attribute*), 519
- `ReturnInStore` (*rdflib.SDO attribute*), 519
- `ReturnLabelCustomerResponsibility` (*rdflib.SDO attribute*), 519
- `ReturnLabelDownloadAndPrint` (*rdflib.SDO attribute*), 520
- `ReturnLabelInBox` (*rdflib.SDO attribute*), 520
- `returnLabelSource` (*rdflib.SDO attribute*), 576
- `ReturnLabelSourceEnumeration` (*rdflib.SDO attribute*), 520
- `returnMethod` (*rdflib.SDO attribute*), 576
- `ReturnMethodEnumeration` (*rdflib.SDO attribute*), 520
- `returnPolicyCategory` (*rdflib.SDO attribute*), 577
- `returnPolicyCountry` (*rdflib.SDO attribute*), 577
- `returnPolicySeasonalOverride` (*rdflib.SDO attribute*), 577
- `ReturnShippingFees` (*rdflib.SDO attribute*), 520
- `returnShippingFeesAmount` (*rdflib.SDO attribute*), 577
- `returnType` (*rdflib.SH attribute*), 594
- `rev()` (*rdflib.extras.describer.Describer method*), 50
- `rev_key` (*rdflib.plugins.shared.jsonld.context.Context property*), 128

- `reverse` (*rdflib.plugins.shared.jsonld.context.Term attribute*), 130
  - `Review` (*rdflib.SDO attribute*), 520
  - `review` (*rdflib.SDO attribute*), 577
  - `ReviewAction` (*rdflib.SDO attribute*), 520
  - `reviewAspect` (*rdflib.SDO attribute*), 577
  - `reviewBody` (*rdflib.SDO attribute*), 577
  - `reviewCount` (*rdflib.SDO attribute*), 577
  - `reviewedBy` (*rdflib.SDO attribute*), 577
  - `ReviewNewsArticle` (*rdflib.SDO attribute*), 520
  - `reviewPolicy` (*rdflib.ODRL2 attribute*), 459
  - `reviewRating` (*rdflib.SDO attribute*), 577
  - `reviews` (*rdflib.SDO attribute*), 577
  - `revisedEntity` (*rdflib.PROV attribute*), 474
  - `revision` (*rdflib.DOAP attribute*), 415
  - `Revision` (*rdflib.PROV attribute*), 469
  - RFC
    - RFC 3066, 27, 31
  - `RFC1766` (*rdflib.DCTERMS attribute*), 411
  - `RFC3066` (*rdflib.DCTERMS attribute*), 411
  - `RFC4646` (*rdflib.DCTERMS attribute*), 411
  - `RFC5646` (*rdflib.DCTERMS attribute*), 411
  - `Rheumatologic` (*rdflib.SDO attribute*), 520
  - `RightHandDriving` (*rdflib.SDO attribute*), 520
  - `RightOperand` (*rdflib.ODRL2 attribute*), 454
  - `rightOperand` (*rdflib.ODRL2 attribute*), 459
  - `rightOperandReference` (*rdflib.ODRL2 attribute*), 459
  - `rights` (*rdflib.DC attribute*), 408
  - `rights` (*rdflib.DCTERMS attribute*), 413
  - `RightsAssignment` (*rdflib.PROV attribute*), 470
  - `rightsHolder` (*rdflib.DCTERMS attribute*), 413
  - `RightsHolder` (*rdflib.PROV attribute*), 470
  - `RightsStatement` (*rdflib.DCTERMS attribute*), 411
  - `Riser` (*rdflib.BRICK attribute*), 385
  - `riskFactor` (*rdflib.SDO attribute*), 577
  - `risks` (*rdflib.SDO attribute*), 577
  - `RisksOrComplicationsHealthAspect` (*rdflib.SDO attribute*), 520
  - `RiverBodyOfWater` (*rdflib.SDO attribute*), 520
  - `Role` (*rdflib.DCAT attribute*), 408
  - `Role` (*rdflib.ORG attribute*), 461
  - `role` (*rdflib.ORG attribute*), 462
  - `Role` (*rdflib.PROV attribute*), 470
  - `Role` (*rdflib.SDO attribute*), 520
  - `roleName` (*rdflib.SDO attribute*), 577
  - `roleProperty` (*rdflib.ORG attribute*), 462
  - `rollback()` (*rdflib.Graph method*), 436
  - `rollback()` (*rdflib.graph.Graph method*), 263
  - `rollback()` (*rdflib.graph.ReadOnlyGraphAggregate method*), 274
  - `rollback()` (*rdflib.plugins.stores.auditable.AuditableStore method*), 191
  - `rollback()` (*rdflib.plugins.stores.regexmatching.REGEXMatching method*), 202
  - `rollback()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore method*), 209
  - `rollback()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore method*), 215
  - `rollback()` (*rdflib.store.Store method*), 311
  - `RoofingContractor` (*rdflib.SDO attribute*), 520
  - `roofLoad` (*rdflib.SDO attribute*), 577
  - `Rooftop` (*rdflib.BRICK attribute*), 385
  - `Rooftop_Unit` (*rdflib.BRICK attribute*), 385
  - `Room` (*rdflib.BRICK attribute*), 385
  - `Room` (*rdflib.SDO attribute*), 520
  - `Room_Air_Temperature_Setpoint` (*rdflib.BRICK attribute*), 385
  - `rootResource` (*rdflib.VOID attribute*), 608
  - `roundtrip_prefixes` (*rdflib.plugins.serializers.turtle.RecursiveSerializer attribute*), 122
  - `Row` (*rdflib.CSVW attribute*), 400
  - `row` (*rdflib.CSVW attribute*), 402
  - `rownum` (*rdflib.CSVW attribute*), 402
  - `rowTitle` (*rdflib.CSVW attribute*), 402
  - `RsvpAction` (*rdflib.SDO attribute*), 520
  - `rsvpResponse` (*rdflib.SDO attribute*), 577
  - `RsvpResponseMaybe` (*rdflib.SDO attribute*), 520
  - `RsvpResponseNo` (*rdflib.SDO attribute*), 520
  - `RsvpResponseType` (*rdflib.SDO attribute*), 520
  - `RsvpResponseYes` (*rdflib.SDO attribute*), 520
  - `rtl` (*rdflib.CSVW attribute*), 402
  - `RTU` (*rdflib.BRICK attribute*), 382
  - `Rule` (*rdflib.ODRL2 attribute*), 454
  - `Rule` (*rdflib.SH attribute*), 590
  - `rule` (*rdflib.SH attribute*), 594
  - `Run_Enable_Command` (*rdflib.BRICK attribute*), 385
  - `Run_Request_Status` (*rdflib.BRICK attribute*), 385
  - `Run_Status` (*rdflib.BRICK attribute*), 385
  - `Run_Time_Sensor` (*rdflib.BRICK attribute*), 385
  - `runNamespace()` (*in module rdflib.plugins.parsers.notation3*), 96
  - `runsTo` (*rdflib.SDO attribute*), 577
  - `runtime` (*rdflib.SDO attribute*), 577
  - `runtimePlatform` (*rdflib.SDO attribute*), 577
  - `RVAV` (*rdflib.BRICK attribute*), 382
  - `RVPark` (*rdflib.SDO attribute*), 517
  - `rxcuri` (*rdflib.SDO attribute*), 577
- ## S
- `s_clause()` (*rdflib.plugins.serializers.n3.N3Serializer method*), 117
  - `s_default()` (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer method*), 116
  - `s_default()` (*rdflib.plugins.serializers.turtle.TurtleSerializer method*), 123
  - `s_squared()` (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer method*), 116

- `s_squared()` (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 123
- `Safety_Equipment` (*rdflib.BRICK* attribute), 385
- `Safety_Shower` (*rdflib.BRICK* attribute), 385
- `Safety_System` (*rdflib.BRICK* attribute), 385
- `safetyConsideration` (*rdflib.SDO* attribute), 577
- `SafetyHealthAspect` (*rdflib.SDO* attribute), 520
- `salaryCurrency` (*rdflib.SDO* attribute), 577
- `salaryUponCompletion` (*rdflib.SDO* attribute), 577
- `SaleEvent` (*rdflib.SDO* attribute), 521
- `SalePrice` (*rdflib.SDO* attribute), 521
- `sameAs` (*rdflib.extras.infixowl.Individual* property), 65
- `sameAs` (*rdflib.OWL* attribute), 466
- `sameAs` (*rdflib.SDO* attribute), 577
- `Sample` (class in *rdflib.plugins.sparql.aggregates*), 144
- `Sample` (*rdflib.SOSA* attribute), 598
- `Sampler` (*rdflib.SOSA* attribute), 598
- `sampleType` (*rdflib.SDO* attribute), 577
- `Sampling` (*rdflib.SOSA* attribute), 598
- `Sash_Position_Sensor` (*rdflib.BRICK* attribute), 385
- `SatireOrParodyContent` (*rdflib.SDO* attribute), 521
- `SatiricalArticle` (*rdflib.SDO* attribute), 521
- `saturatedFatContent` (*rdflib.SDO* attribute), 577
- `Saturday` (*rdflib.SDO* attribute), 521
- `Saturday` (*rdflib.TIME* attribute), 600
- `Schedule` (*rdflib.SDO* attribute), 521
- `Schedule_Temperature_Setpoint` (*rdflib.BRICK* attribute), 385
- `ScheduleAction` (*rdflib.SDO* attribute), 521
- `scheduledPaymentDate` (*rdflib.SDO* attribute), 578
- `scheduledTime` (*rdflib.SDO* attribute), 578
- `scheduleTimezone` (*rdflib.SDO* attribute), 578
- `Schema` (*rdflib.CSVW* attribute), 400
- `schemaReference` (*rdflib.CSVW* attribute), 402
- `schemaVersion` (*rdflib.SDO* attribute), 578
- `ScholarlyArticle` (*rdflib.SDO* attribute), 521
- `School` (*rdflib.SDO* attribute), 521
- `schoolClosuresInfo` (*rdflib.SDO* attribute), 578
- `SchoolDistrict` (*rdflib.SDO* attribute), 521
- `schoolHomepage` (*rdflib.FOAF* attribute), 422
- `scope` (*rdflib.ODRL2* attribute), 459
- `scopeNote` (*rdflib.SKOS* attribute), 597
- `screenCount` (*rdflib.SDO* attribute), 578
- `ScreeningEvent` (*rdflib.SDO* attribute), 521
- `ScreeningHealthAspect` (*rdflib.SDO* attribute), 521
- `screenshot` (*rdflib.SDO* attribute), 578
- `screenshots` (*rdflib.DOAP* attribute), 415
- `scriptFormat` (*rdflib.CSVW* attribute), 402
- `Sculpture` (*rdflib.SDO* attribute), 521
- `sdDatePublished` (*rdflib.SDO* attribute), 578
- `sdLicense` (*rdflib.SDO* attribute), 578
- `SDO` (class in *rdflib*), 479
- `sdPublisher` (*rdflib.SDO* attribute), 578
- `SeaBodyOfWater` (*rdflib.SDO* attribute), 521
- `SearchAction` (*rdflib.SDO* attribute), 521
- `SearchResultsPage` (*rdflib.SDO* attribute), 521
- `Season` (*rdflib.SDO* attribute), 521
- `season` (*rdflib.SDO* attribute), 578
- `seasonNumber` (*rdflib.SDO* attribute), 578
- `seasons` (*rdflib.SDO* attribute), 578
- `Seat` (*rdflib.SDO* attribute), 521
- `seatingCapacity` (*rdflib.SDO* attribute), 578
- `SeatingMap` (*rdflib.SDO* attribute), 521
- `seatingType` (*rdflib.SDO* attribute), 578
- `seatNumber` (*rdflib.SDO* attribute), 578
- `seatRow` (*rdflib.SDO* attribute), 578
- `seatSection` (*rdflib.SDO* attribute), 578
- `second` (*rdflib.TIME* attribute), 603
- `second` (*rdflib.XSD* attribute), 611
- `secondaryPrevention` (*rdflib.SDO* attribute), 578
- `secondaryUse` (*rdflib.ODRL2* attribute), 459
- `seconds` (*rdflib.TIME* attribute), 603
- `SecuredHTTPHandler` (class in *examples.secure\_with\_urlopen*), 26
- `Security_Equipment` (*rdflib.BRICK* attribute), 386
- `Security_Service_Room` (*rdflib.BRICK* attribute), 386
- `securityClearanceRequirement` (*rdflib.SDO* attribute), 578
- `securityScreening` (*rdflib.SDO* attribute), 578
- `seeAlso` (*rdflib.extras.infixowl.AnnotatableTerms* property), 58
- `seeAlso` (*rdflib.RDFS* attribute), 479
- `SeeDoctorHealthAspect` (*rdflib.SDO* attribute), 521
- `seeks` (*rdflib.SDO* attribute), 578
- `SeekToAction` (*rdflib.SDO* attribute), 521
- `seen` (*rdflib.plugins.sparql.aggregates.Average* attribute), 141
- `seen` (*rdflib.plugins.sparql.aggregates.Counter* attribute), 142
- `seen` (*rdflib.plugins.sparql.aggregates.Extremum* attribute), 143
- `seen` (*rdflib.plugins.sparql.aggregates.GroupConcat* attribute), 143
- `seen` (*rdflib.plugins.sparql.aggregates.Maximum* attribute), 144
- `seen` (*rdflib.plugins.sparql.aggregates.Minimum* attribute), 144
- `seen` (*rdflib.plugins.sparql.aggregates.Sample* attribute), 144
- `seen` (*rdflib.plugins.sparql.aggregates.Sum* attribute), 145
- `select` (*rdflib.SH* attribute), 594
- `SelfCareHealthAspect` (*rdflib.SDO* attribute), 521
- `SelfStorage` (*rdflib.SDO* attribute), 521
- `sell` (*rdflib.ODRL2* attribute), 459
- `SellAction` (*rdflib.SDO* attribute), 521
- `seller` (*rdflib.SDO* attribute), 578
- `semanticRelation` (*rdflib.SKOS* attribute), 597

- SendAction (rdflib.SDO attribute), 521
- sender (rdflib.SDO attribute), 578
- Sensor (rdflib.BRICK attribute), 386
- Sensor (rdflib.SOSA attribute), 598
- sensoryRequirement (rdflib.SDO attribute), 578
- sensoryUnit (rdflib.SDO attribute), 578
- separator (rdflib.CSVW attribute), 402
- Seq (class in rdflib.container), 233
- Seq (class in rdflib.graph), 274
- Seq (rdflib.RDF attribute), 478
- SequencePath (class in rdflib.paths), 286
- serialize() (rdflib.extras.infixowl.BooleanClass method), 59
- serialize() (rdflib.extras.infixowl.Class method), 62
- serialize() (rdflib.extras.infixowl.EnumeratedClass method), 64
- serialize() (rdflib.extras.infixowl.Individual method), 65
- serialize() (rdflib.extras.infixowl.Property method), 67
- serialize() (rdflib.extras.infixowl.Restriction method), 68
- serialize() (rdflib.Graph method), 436
- serialize() (rdflib.graph.Graph method), 263
- serialize() (rdflib.plugins.serializers.hex.HexTuplesSerializer method), 114
- serialize() (rdflib.plugins.serializers.jsonld.JsonLDSerializer method), 115
- serialize() (rdflib.plugins.serializers.longturtle.LongTurtleSerializer method), 116
- serialize() (rdflib.plugins.serializers.nquads.NQuadsSerializer method), 117
- serialize() (rdflib.plugins.serializers.nt.NTSerializer method), 118
- serialize() (rdflib.plugins.serializers.rdfxml.PrettyXMLSerializer method), 118
- serialize() (rdflib.plugins.serializers.rdfxml.XMLSerializer method), 119
- serialize() (rdflib.plugins.serializers.trig.TrigSerializer method), 120
- serialize() (rdflib.plugins.serializers.trix.TriXSerializer method), 121
- serialize() (rdflib.plugins.serializers.turtle.TurtleSerializer method), 123
- serialize() (rdflib.plugins.sparql.results.csvresults.CSVResultSerializer method), 132
- serialize() (rdflib.plugins.sparql.results.jsonresults.JSONResultSerializer method), 134
- serialize() (rdflib.plugins.sparql.results.txtresults.TXTResultSerializer method), 136
- serialize() (rdflib.plugins.sparql.results.xmlresults.XMLResultSerializer method), 139
- serialize() (rdflib.query.Result method), 294
- serialize() (rdflib.query.ResultSerializer method), 297
- serialize() (rdflib.serializer.Serializer method), 306
- serialize\_in\_chunks() (in module rdflib.tools.chunk\_serializer), 218
- Serializer (class in rdflib.serializer), 305
- serializeTerm() (rdflib.plugins.sparql.results.csvresults.CSVResultSerializer method), 132
- serialNumber (rdflib.SDO attribute), 579
- Series (rdflib.SDO attribute), 521
- seriousAdverseOutcome (rdflib.SDO attribute), 579
- Server\_Room (rdflib.BRICK attribute), 386
- serverStatus (rdflib.SDO attribute), 579
- servesCuisine (rdflib.SDO attribute), 579
- servesDataset (rdflib.DCAT attribute), 409
- service (rdflib.DCAT attribute), 409
- Service (rdflib.DCMITYPE attribute), 410
- Service (rdflib.SDO attribute), 522
- Service\_Room (rdflib.BRICK attribute), 386
- serviceArea (rdflib.SDO attribute), 579
- serviceAudience (rdflib.SDO attribute), 579
- ServiceChannel (rdflib.SDO attribute), 522
- ServiceDescription (rdflib.PROV attribute), 470
- serviceLocation (rdflib.SDO attribute), 579
- serviceOperator (rdflib.SDO attribute), 579
- serviceOutput (rdflib.SDO attribute), 579
- servicePhone (rdflib.SDO attribute), 579
- servicePostalAddress (rdflib.SDO attribute), 579
- serviceSmsNumber (rdflib.SDO attribute), 579
- ServiceType (rdflib.SDO attribute), 579
- serviceUrl (rdflib.SDO attribute), 579
- serviceWorkingSize (rdflib.SDO attribute), 579
- Set (rdflib.ODRL2 attribute), 454
- set() (rdflib.Graph method), 437
- set() (rdflib.graph.Graph method), 264
- set() (rdflib.resource.Resource method), 305
- set\_map() (rdflib.events.Dispatcher method), 234
- set\_value() (rdflib.plugins.sparql.aggregates.Accumulator method), 140
- set\_value() (rdflib.plugins.sparql.aggregates.Extremum method), 143
- setDataType() (in module rdflib.plugins.sparql.parser), 169
- setDefaultNamespace() (rdflib.plugins.parsers.notation3.RDFSink method), 86
- setDocumentLocator() (rdflib.plugins.parsers.rdfxml.RDFXMLHandler method), 106
- setEntityLocator() (rdflib.plugins.parsers.trix.TriXHandler method), 117
- setEvalFn() (rdflib.plugins.sparql.parserutils.Comp method), 170



- setKeywords() (*rdflib.plugins.parsers.notation3.SinkParser* attribute), 92  
 setLanguage() (*in module rdflib.plugins.sparql.parser*), 169  
 Setpoint (*rdflib.BRICK* attribute), 386  
 setPublicId() (*rdflib.parser.PythonInputSource* method), 277  
 setSystemId() (*rdflib.parser.PythonInputSource* method), 278  
 setTimeout() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 216  
 setupACEAnnotations() (*rdflib.extras.infixowl.AnnotatableTerms* method), 59  
 setupNounAnnotations() (*rdflib.extras.infixowl.Class* method), 62  
 setupVerbAnnotations() (*rdflib.extras.infixowl.Property* method), 67  
 setVersion() (*rdflib.extras.infixowl.Ontology* method), 67  
 Severity (*rdflib.SH* attribute), 591  
 severity (*rdflib.SH* attribute), 594  
 SH (*class in rdflib*), 587  
 sha1 (*rdflib.FOAF* attribute), 422  
 sha256 (*rdflib.SDO* attribute), 579  
 Shading\_System (*rdflib.BRICK* attribute), 386  
 Shape (*rdflib.SH* attribute), 591  
 shapesGraph (*rdflib.SH* attribute), 594  
 shapesGraphWellFormed (*rdflib.SH* attribute), 594  
 share (*rdflib.ODRL2* attribute), 459  
 ShareAction (*rdflib.SDO* attribute), 522  
 shareAlike (*rdflib.ODRL2* attribute), 459  
 Shared\_Office (*rdflib.BRICK* attribute), 386  
 sharedContent (*rdflib.SDO* attribute), 579  
 sharesDefinitionWith (*rdflib.PROV* attribute), 474  
 SheetMusic (*rdflib.SDO* attribute), 522  
 ShippingDeliveryTime (*rdflib.SDO* attribute), 522  
 shippingDestination (*rdflib.SDO* attribute), 579  
 shippingDetails (*rdflib.SDO* attribute), 579  
 shippingLabel (*rdflib.SDO* attribute), 579  
 shippingRate (*rdflib.SDO* attribute), 579  
 ShippingRateSettings (*rdflib.SDO* attribute), 522  
 shippingSettingsLink (*rdflib.SDO* attribute), 579  
 ShoeStore (*rdflib.SDO* attribute), 522  
 ShoppingCenter (*rdflib.SDO* attribute), 522  
 short (*rdflib.XSD* attribute), 611  
 Short\_Cycle\_Alarm (*rdflib.BRICK* attribute), 386  
 short\_name (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* attribute), 116  
 short\_name (*rdflib.plugins.serializers.n3.N3Serializer* attribute), 117  
 short\_name (*rdflib.plugins.serializers.trig.TrigSerializer* attribute), 120  
 short\_name (*rdflib.plugins.serializers.turtle.TurtleSerializer* attribute), 123  
 shortdesc (*rdflib.DOAP* attribute), 415  
 ShortStory (*rdflib.SDO* attribute), 522  
 Shower (*rdflib.BRICK* attribute), 386  
 shrink\_iri() (*rdflib.plugins.shared.jsonld.context.Context* method), 128  
 sibling (*rdflib.SDO* attribute), 579  
 siblings (*rdflib.SDO* attribute), 579  
 SideEffectsHealthAspect (*rdflib.SDO* attribute), 522  
 sign() (*in module rdflib.compat*), 230  
 signDetected (*rdflib.SDO* attribute), 579  
 significance (*rdflib.SDO* attribute), 579  
 significantLink (*rdflib.SDO* attribute), 579  
 significantLinks (*rdflib.SDO* attribute), 580  
 signOrSymptom (*rdflib.SDO* attribute), 579  
 similar() (*in module rdflib.compare*), 229  
 SimpleMemory (*class in rdflib.plugins.stores.memory*), 199  
 simplify() (*in module rdflib.plugins.sparql.algebra*), 148  
 simplify() (*in module rdflib.plugins.sparql.operators*), 167  
 SingleBlindedTrial (*rdflib.SDO* attribute), 522  
 SingleCenterTrial (*rdflib.SDO* attribute), 522  
 SingleFamilyResidence (*rdflib.SDO* attribute), 522  
 SinglePlayer (*rdflib.SDO* attribute), 522  
 SingleRelease (*rdflib.SDO* attribute), 522  
 sink (*rdflib.plugins.parsers.nquads.NQuadsParser* attribute), 98  
 sink (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* attribute), 101  
 SinkParser (*class in rdflib.plugins.parsers.notation3*), 86  
 Site (*rdflib.BRICK* attribute), 386  
 Site (*rdflib.ORG* attribute), 461  
 siteAddress (*rdflib.ORG* attribute), 462  
 SiteNavigationElement (*rdflib.SDO* attribute), 522  
 siteOf (*rdflib.ORG* attribute), 462  
 size (*rdflib.SDO* attribute), 580  
 sizeGroup (*rdflib.SDO* attribute), 580  
 SizeGroupEnumeration (*rdflib.SDO* attribute), 522  
 SizeOrDuration (*rdflib.DCTERMS* attribute), 411  
 SizeSpecification (*rdflib.SDO* attribute), 522  
 sizeSystem (*rdflib.SDO* attribute), 580  
 SizeSystemEnumeration (*rdflib.SDO* attribute), 522  
 SizeSystemImperial (*rdflib.SDO* attribute), 522  
 SizeSystemMetric (*rdflib.SDO* attribute), 522  
 skills (*rdflib.SDO* attribute), 580  
 Skin (*rdflib.SDO* attribute), 522  
 skipBlankRows (*rdflib.CSVW* attribute), 402  
 skipColumns (*rdflib.CSVW* attribute), 402  
 skipInitialSpace (*rdflib.CSVW* attribute), 403  
 skipRows (*rdflib.CSVW* attribute), 403

- `skipSpace()` (*rdflib.plugins.parsers.notation3.SinkParser* method), 92
- `SkiResort` (*rdflib.SDO* attribute), 522
- `skolemize()` (*rdflib.BNode* method), 336
- `skolemize()` (*rdflib.Graph* method), 437
- `skolemize()` (*rdflib.graph.Graph* method), 264
- `skolemize()` (*rdflib.term.BNode* method), 314
- `SKOS` (class in *rdflib*), 595
- `sku` (*rdflib.SDO* attribute), 580
- `skypeID` (*rdflib.FOAF* attribute), 422
- `Slice` (*rdflib.QB* attribute), 476
- `slice` (*rdflib.QB* attribute), 477
- `SliceKey` (*rdflib.QB* attribute), 476
- `sliceKey` (*rdflib.QB* attribute), 477
- `sliceStructure` (*rdflib.QB* attribute), 477
- `slogan` (*rdflib.SDO* attribute), 580
- `smiles` (*rdflib.SDO* attribute), 580
- `Smoke_Alarm` (*rdflib.BRICK* attribute), 386
- `Smoke_Detection_Alarm` (*rdflib.BRICK* attribute), 386
- `smokingAllowed` (*rdflib.SDO* attribute), 580
- `SocialEvent` (*rdflib.SDO* attribute), 523
- `SocialMediaPosting` (*rdflib.SDO* attribute), 523
- `sodiumContent` (*rdflib.SDO* attribute), 580
- `Software` (*rdflib.DCMITYPE* attribute), 410
- `softwareAddOn` (*rdflib.SDO* attribute), 580
- `SoftwareAgent` (*rdflib.PROV* attribute), 470
- `SoftwareApplication` (*rdflib.SDO* attribute), 523
- `softwareHelp` (*rdflib.SDO* attribute), 580
- `softwareRequirements` (*rdflib.SDO* attribute), 580
- `SoftwareSourceCode` (*rdflib.SDO* attribute), 523
- `softwareVersion` (*rdflib.SDO* attribute), 580
- `Solar_Azimuth_Angle_Sensor` (*rdflib.BRICK* attribute), 386
- `Solar_Radiance_Sensor` (*rdflib.BRICK* attribute), 386
- `Solar_Thermal_Collector` (*rdflib.BRICK* attribute), 386
- `Solar_Zenith_Angle_Sensor` (*rdflib.BRICK* attribute), 386
- `SoldOut` (*rdflib.SDO* attribute), 523
- `Solid` (*rdflib.BRICK* attribute), 386
- `solution()` (*rdflib.plugins.sparql.sparql.QueryContext* method), 184
- `SolveMathAction` (*rdflib.SDO* attribute), 523
- `SomeProducts` (*rdflib.SDO* attribute), 523
- `someValuesFrom` (*rdflib.extras.infixowl.Restriction* property), 69
- `someValuesFrom` (*rdflib.OWL* attribute), 466
- `sortProperties()` (*rdflib.plugins.serializers.turtle.RecursiveSerializer* method), 122
- `SOSA` (class in *rdflib*), 597
- `Sound` (*rdflib.DCMITYPE* attribute), 410
- `SoundtrackAlbum` (*rdflib.SDO* attribute), 523
- `source` (*rdflib.CSVW* attribute), 403
- `source` (*rdflib.DC* attribute), 408
- `source` (*rdflib.DCTERMS* attribute), 413
- `source` (*rdflib.ODRL2* attribute), 459
- `source_to_json()` (in module *rdflib.plugins.shared.jsonld.util*), 131
- `sourceConstraint` (*rdflib.SH* attribute), 595
- `sourceConstraintComponent` (*rdflib.SH* attribute), 595
- `sourcedFrom` (*rdflib.SDO* attribute), 580
- `sourceIndividual` (*rdflib.OWL* attribute), 466
- `sourceOrganization` (*rdflib.SDO* attribute), 580
- `sourceShape` (*rdflib.SH* attribute), 595
- `Space` (*rdflib.BRICK* attribute), 386
- `Space_Heater` (*rdflib.BRICK* attribute), 386
- `sparql` (*rdflib.SH* attribute), 595
- `SPARQLAskExecutable` (*rdflib.SH* attribute), 590
- `SPARQLAskValidator` (*rdflib.SH* attribute), 590
- `SPARQLConnector` (class in *rdflib.plugins.stores.sparqlconnector*), 203
- `SPARQLConnectorException`, 204
- `SPARQLConstraint` (*rdflib.SH* attribute), 590
- `SPARQLConstraintComponent` (*rdflib.SH* attribute), 590
- `SPARQLConstructExecutable` (*rdflib.SH* attribute), 591
- `sparqlDirective()` (*rdflib.plugins.parsers.notation3.SinkParser* method), 92
- `sparqlEndpoint` (*rdflib.VOID* attribute), 608
- `SPARQLError`, 184
- `SPARQLExecutable` (*rdflib.SH* attribute), 591
- `SPARQLFunction` (*rdflib.SH* attribute), 591
- `SPARQLProcessor` (class in *rdflib.plugins.sparql.processor*), 174
- `SPARQLResult` (class in *rdflib.plugins.sparql.processor*), 174
- `SPARQLRule` (*rdflib.SH* attribute), 591
- `SPARQLSelectExecutable` (*rdflib.SH* attribute), 591
- `SPARQLSelectValidator` (*rdflib.SH* attribute), 591
- `SPARQLStore` (class in *rdflib.plugins.stores.sparqlstore*), 205
- `SPARQLTarget` (*rdflib.SH* attribute), 591
- `SPARQLTargetType` (*rdflib.SH* attribute), 591
- `sparqlTok()` (*rdflib.plugins.parsers.notation3.SinkParser* method), 92
- `SPARQLTypeError`, 184
- `SPARQLUpdateExecutable` (*rdflib.SH* attribute), 591
- `SPARQLUpdateProcessor` (class in *rdflib.plugins.sparql.processor*), 174
- `SPARQLUpdateStore` (class in *rdflib.plugins.stores.sparqlstore*), 211
- `SPARQLXMLWriter` (class in *rdflib.plugins.sparql.results.xmlresults*), 136
- `spatial` (*rdflib.DCTERMS* attribute), 413

- spatial (*rdflib.ODRL2* attribute), 459
- spatial (*rdflib.SDO* attribute), 580
- spatialCoordinates (*rdflib.ODRL2* attribute), 460
- spatialCoverage (*rdflib.SDO* attribute), 580
- spatialResolutionInMeters (*rdflib.DCAT* attribute), 409
- speakable (*rdflib.SDO* attribute), 580
- SpeakableSpecification (*rdflib.SDO* attribute), 523
- SpecialAnnouncement (*rdflib.SDO* attribute), 523
- specialCommitments (*rdflib.SDO* attribute), 580
- specializationOf (*rdflib.PROV* attribute), 474
- specialOpeningHoursSpecification (*rdflib.SDO* attribute), 580
- Specialty (*rdflib.SDO* attribute), 523
- specialty (*rdflib.SDO* attribute), 580
- Specification (*rdflib.DOAP* attribute), 414
- SpeechPathology (*rdflib.SDO* attribute), 523
- speechToTextMarkup (*rdflib.SDO* attribute), 580
- speed (*rdflib.SDO* attribute), 580
- Speed\_Reset\_Command (*rdflib.BRICK* attribute), 386
- Speed\_Sensor (*rdflib.BRICK* attribute), 387
- Speed\_Setpoint (*rdflib.BRICK* attribute), 387
- Speed\_Setpoint\_Limit (*rdflib.BRICK* attribute), 387
- Speed\_Status (*rdflib.BRICK* attribute), 387
- split\_iri() (in module *rdflib.plugins.shared.jsonld.util*), 131
- split\_uri() (in module *rdflib.namespace*), 78
- splitFragPC() (in module *rdflib.plugins.parsers.notation3*), 96
- spokenByCharacter (*rdflib.SDO* attribute), 580
- SpokenWordAlbum (*rdflib.SDO* attribute), 523
- sponsor (*rdflib.SDO* attribute), 580
- sport (*rdflib.SDO* attribute), 581
- SportingGoodsStore (*rdflib.SDO* attribute), 523
- Sports\_Service\_Room (*rdflib.BRICK* attribute), 387
- SportsActivityLocation (*rdflib.SDO* attribute), 523
- sportsActivityLocation (*rdflib.SDO* attribute), 581
- SportsClub (*rdflib.SDO* attribute), 523
- SportsEvent (*rdflib.SDO* attribute), 523
- sportsEvent (*rdflib.SDO* attribute), 581
- SportsOrganization (*rdflib.SDO* attribute), 523
- SportsTeam (*rdflib.SDO* attribute), 523
- sportsTeam (*rdflib.SDO* attribute), 581
- spouse (*rdflib.SDO* attribute), 581
- SpreadsheetDigitalDocument (*rdflib.SDO* attribute), 523
- SRP (*rdflib.SDO* attribute), 520
- SSN (class in *rdflib*), 599
- StadiumOrArena (*rdflib.SDO* attribute), 523
- stage (*rdflib.SDO* attribute), 581
- Stage\_Enable\_Command (*rdflib.BRICK* attribute), 387
- Stage\_Riser (*rdflib.BRICK* attribute), 387
- stageAsNumber (*rdflib.SDO* attribute), 581
- StagedContent (*rdflib.SDO* attribute), 523
- Stages\_Status (*rdflib.BRICK* attribute), 387
- StagesHealthAspect (*rdflib.SDO* attribute), 523
- Staircase (*rdflib.BRICK* attribute), 387
- Standard (*rdflib.DCTERMS* attribute), 411
- Standby\_CRAC (*rdflib.BRICK* attribute), 387
- Standby\_Fan (*rdflib.BRICK* attribute), 387
- Standby\_Glycool\_Unit\_On\_Off\_Status (*rdflib.BRICK* attribute), 387
- Standby\_Load\_Shed\_Command (*rdflib.BRICK* attribute), 387
- Standby\_Unit\_On\_Off\_Status (*rdflib.BRICK* attribute), 387
- starRating (*rdflib.SDO* attribute), 581
- start (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 103
- Start (*rdflib.PROV* attribute), 470
- Start\_Stop\_Command (*rdflib.BRICK* attribute), 387
- Start\_Stop\_Status (*rdflib.BRICK* attribute), 387
- startDate (*rdflib.DCAT* attribute), 409
- startDate (*rdflib.SDO* attribute), 581
- startDoc() (*rdflib.plugins.parsers.notation3.RDFSink* method), 86
- startDoc() (*rdflib.plugins.parsers.notation3.SinkParser* method), 93
- startDocument() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 107
- startDocument() (*rdflib.plugins.parsers.trix.TriXHandler* method), 112
- startDocument() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer* method), 116
- startDocument() (*rdflib.plugins.serializers.turtle.TurtleSerializer* method), 123
- started (*rdflib.PROV* attribute), 474
- startedAtTime (*rdflib.PROV* attribute), 474
- startElementNS() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 107
- startElementNS() (*rdflib.plugins.parsers.trix.TriXHandler* method), 112
- startOffset (*rdflib.SDO* attribute), 581
- startPrefixMapping() (*rdflib.plugins.parsers.rdfxml.RDFXMLHandler* method), 107
- startPrefixMapping() (*rdflib.plugins.parsers.trix.TriXHandler* method), 112
- startswith() (*rdflib.term.Identifier* method), 317
- startTime (*rdflib.SDO* attribute), 581
- State (*rdflib.SDO* attribute), 523

- Statement (*rdflib.RDF attribute*), 478
- Statement (*rdflib.SDO attribute*), 523
- statement() (*rdflib.plugins.parsers.notation3.SinkParser method*), 93
- statement() (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer method*), 116
- statement() (*rdflib.plugins.serializers.n3.N3Serializer method*), 117
- statement() (*rdflib.plugins.serializers.turtle.TurtleSerializer method*), 123
- Static\_Pressure\_Deadband\_Setpoint (*rdflib.BRICK attribute*), 387
- Static\_Pressure\_Integral\_Time\_Parameter (*rdflib.BRICK attribute*), 387
- Static\_Pressure\_Proportional\_Band\_Parameter (*rdflib.BRICK attribute*), 387
- Static\_Pressure\_Sensor (*rdflib.BRICK attribute*), 388
- Static\_Pressure\_Setpoint (*rdflib.BRICK attribute*), 388
- Static\_Pressure\_Setpoint\_Limit (*rdflib.BRICK attribute*), 388
- Static\_Pressure\_Step\_Parameter (*rdflib.BRICK attribute*), 388
- StatisticalPopulation (*rdflib.SDO attribute*), 524
- Status (*rdflib.BRICK attribute*), 388
- status (*rdflib.FOAF attribute*), 422
- status (*rdflib.ODRL2 attribute*), 460
- status (*rdflib.SDO attribute*), 581
- StatusEnumeration (*rdflib.SDO attribute*), 524
- Steam (*rdflib.BRICK attribute*), 388
- Steam\_Baseboard\_Radiator (*rdflib.BRICK attribute*), 388
- Steam\_Distribution (*rdflib.BRICK attribute*), 388
- Steam\_On\_Off\_Command (*rdflib.BRICK attribute*), 388
- Steam\_Radiator (*rdflib.BRICK attribute*), 388
- Steam\_System (*rdflib.BRICK attribute*), 388
- Steam\_Usage\_Sensor (*rdflib.BRICK attribute*), 388
- Steam\_Valve (*rdflib.BRICK attribute*), 388
- steeringPosition (*rdflib.SDO attribute*), 581
- SteeringPositionValue (*rdflib.SDO attribute*), 524
- step (*rdflib.SDO attribute*), 581
- Step\_Parameter (*rdflib.BRICK attribute*), 388
- steps (*rdflib.SDO attribute*), 581
- stepValue (*rdflib.SDO attribute*), 581
- StillImage (*rdflib.DCMITYPE attribute*), 410
- Stimulus (*rdflib.SSN attribute*), 599
- StopTraversal, 147
- Storage\_Room (*rdflib.BRICK attribute*), 388
- storageRequirements (*rdflib.SDO attribute*), 581
- Store (class in *rdflib.store*), 307
- store (*rdflib.Graph property*), 437
- store (*rdflib.graph.Graph property*), 264
- store (*rdflib.namespace.NamespaceManager property*), 78
- store (*rdflib.plugins.serializers.hext.HexuplesSerializer attribute*), 114
- store (*rdflib.plugins.serializers.jsonld.JsonLDSerializer attribute*), 115
- store (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer attribute*), 116
- store (*rdflib.plugins.serializers.n3.N3Serializer attribute*), 117
- store (*rdflib.plugins.serializers.nquads.NQuadsSerializer attribute*), 117
- store (*rdflib.plugins.serializers.nt.NTSerializer attribute*), 118
- store (*rdflib.plugins.serializers.rdfxml.PrettyXMLSerializer attribute*), 119
- store (*rdflib.plugins.serializers.rdfxml.XMLSerializer attribute*), 119
- store (*rdflib.plugins.serializers.trig.TrigSerializer attribute*), 120
- store (*rdflib.plugins.serializers.trix.TriXSerializer attribute*), 121
- store (*rdflib.plugins.serializers.turtle.RecursiveSerializer attribute*), 122
- store (*rdflib.plugins.serializers.turtle.TurtleSerializer attribute*), 123
- Store (*rdflib.SDO attribute*), 524
- StoreCreatedEvent (class in *rdflib.store*), 312
- StoreCreditRefund (*rdflib.SDO attribute*), 524
- storedAt (*rdflib.BRICK attribute*), 400
- Storey (*rdflib.BRICK attribute*), 388
- strconst() (*rdflib.plugins.parsers.notation3.SinkParser method*), 93
- stream (*rdflib.ODRL2 attribute*), 460
- streetAddress (*rdflib.SDO attribute*), 581
- StrengthTraining (*rdflib.SDO attribute*), 524
- strengthUnit (*rdflib.SDO attribute*), 581
- strengthValue (*rdflib.SDO attribute*), 581
- String (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore attribute*), 212
- string (*rdflib.XSD attribute*), 611
- string() (in module *rdflib.plugins.sparql.operators*), 167
- STRING\_LITERAL1 (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore attribute*), 212
- STRING\_LITERAL2 (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore attribute*), 212
- STRING\_LITERAL\_LONG1 (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore attribute*), 212
- STRING\_LITERAL\_LONG2 (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore attribute*), 212



- attribute*), 212
- `StringInputSource` (class in *rdflib.parser*), 278
- `structuralClass` (*rdflib.SDO* attribute), 581
- `structure` (*rdflib.QB* attribute), 477
- `StructuredValue` (*rdflib.SDO* attribute), 524
- `Studio` (*rdflib.BRICK* attribute), 388
- `StudioAlbum` (*rdflib.SDO* attribute), 524
- `study` (*rdflib.SDO* attribute), 581
- `studyDesign` (*rdflib.SDO* attribute), 581
- `studyLocation` (*rdflib.SDO* attribute), 581
- `studySubject` (*rdflib.SDO* attribute), 581
- `subClassOf` (*rdflib.extras.infixowl.Class* property), 62
- `subClassOf` (*rdflib.RDFS* attribute), 479
- `subcontext()` (*rdflib.plugins.shared.jsonld.context.Context* method), 128
- `subEvent` (*rdflib.SDO* attribute), 581
- `subEvents` (*rdflib.SDO* attribute), 581
- `subject` (*rdflib.DC* attribute), 408
- `subject` (*rdflib.DCTERMS* attribute), 413
- `subject` (*rdflib.plugins.parsers.rdfxml.ElementHandler* attribute), 103
- `subject` (*rdflib.RDF* attribute), 478
- `subject` (*rdflib.SH* attribute), 595
- `subject()` (*rdflib.plugins.parsers.notation3.SinkParser* method), 93
- `subject()` (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* method), 101
- `subject()` (*rdflib.plugins.serializers.rdfxml.PrettyXMLSerializer* method), 119
- `subject()` (*rdflib.plugins.serializers.rdfxml.XMLSerializer* method), 119
- `subject_objects()` (*rdflib.Graph* method), 437
- `subject_objects()` (*rdflib.graph.Graph* method), 264
- `subject_objects()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 209
- `subject_objects()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 216
- `subject_objects()` (*rdflib.resource.Resource* method), 305
- `subject_predicates()` (*rdflib.Graph* method), 438
- `subject_predicates()` (*rdflib.graph.Graph* method), 265
- `subject_predicates()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 210
- `subject_predicates()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 216
- `subject_predicates()` (*rdflib.resource.Resource* method), 305
- `subjectDone()` (*rdflib.plugins.serializers.turtle.RecursiveSerializer* method), 122
- `subjectOf` (*rdflib.SDO* attribute), 582
- `subjects()` (*rdflib.Graph* method), 438
- `subjects()` (*rdflib.graph.Graph* method), 265
- `subjects()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 210
- `subjects()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 216
- `subjects()` (*rdflib.resource.Resource* method), 305
- `subjectsTarget` (*rdflib.VOID* attribute), 608
- `Submit` (*rdflib.PROV* attribute), 470
- `subOrganization` (*rdflib.SDO* attribute), 581
- `subOrganizationOf` (*rdflib.ORG* attribute), 462
- `subPropertyOf` (*rdflib.extras.infixowl.Property* property), 68
- `subPropertyOf` (*rdflib.RDFS* attribute), 479
- `subReservation` (*rdflib.SDO* attribute), 581
- `subscribe()` (*rdflib.events.Dispatcher* method), 234
- `SubscribeAction` (*rdflib.SDO* attribute), 524
- `Subscription` (*rdflib.SDO* attribute), 524
- `subset` (*rdflib.VOID* attribute), 608
- `subStageSuffix` (*rdflib.SDO* attribute), 582
- `Substance` (*rdflib.BRICK* attribute), 388
- `Substance` (*rdflib.SDO* attribute), 524
- `subStructure` (*rdflib.SDO* attribute), 582
- `subSumpteeIds()` (*rdflib.extras.infixowl.Class* method), 62
- `subTest` (*rdflib.SDO* attribute), 582
- `subTitleLanguage` (*rdflib.SDO* attribute), 582
- `subTrip` (*rdflib.SDO* attribute), 582
- `SubwayStation` (*rdflib.SDO* attribute), 524
- `successorOf` (*rdflib.SDO* attribute), 582
- `sugarContent` (*rdflib.SDO* attribute), 582
- `suggestedAge` (*rdflib.SDO* attribute), 582
- `suggestedAnswer` (*rdflib.SDO* attribute), 582
- `suggestedGender` (*rdflib.SDO* attribute), 582
- `suggestedMaxAge` (*rdflib.SDO* attribute), 582
- `suggestedMeasurement` (*rdflib.SDO* attribute), 582
- `suggestedMinAge` (*rdflib.SDO* attribute), 582
- `suggestedShapesGraph` (*rdflib.SH* attribute), 595
- `suitableForDiet` (*rdflib.SDO* attribute), 582
- `Suite` (*rdflib.SDO* attribute), 524
- `Sum` (class in *rdflib.plugins.sparql.aggregates*), 145
- `Sunday` (*rdflib.SDO* attribute), 524
- `Sunday` (*rdflib.TIME* attribute), 600
- `superEvent` (*rdflib.SDO* attribute), 582
- `SuperficialAnatomy` (*rdflib.SDO* attribute), 524
- `supersededBy` (*rdflib.SDO* attribute), 582
- `supply` (*rdflib.SDO* attribute), 582
- `Supply_Air` (*rdflib.BRICK* attribute), 388
- `Supply_Air_Differential_Pressure_Sensor` (*rdflib.BRICK* attribute), 388
- `Supply_Air_Differential_Pressure_Setpoint` (*rdflib.BRICK* attribute), 389

Supply\_Air\_Duct\_Pressure\_Status (rdflib.BRICK attribute), 389

Supply\_Air\_Flow\_Demand\_Setpoint (rdflib.BRICK attribute), 389

Supply\_Air\_Flow\_Sensor (rdflib.BRICK attribute), 389

Supply\_Air\_Flow\_Setpoint (rdflib.BRICK attribute), 389

Supply\_Air\_Humidity\_Sensor (rdflib.BRICK attribute), 389

Supply\_Air\_Humidity\_Setpoint (rdflib.BRICK attribute), 389

Supply\_Air\_Integral\_Gain\_Parameter (rdflib.BRICK attribute), 389

Supply\_Air\_Plenum (rdflib.BRICK attribute), 389

Supply\_Air\_Proportional\_Gain\_Parameter (rdflib.BRICK attribute), 389

Supply\_Air\_Static\_Pressure\_Deadband\_Setpoint (rdflib.BRICK attribute), 389

Supply\_Air\_Static\_Pressure\_Integral\_Time\_Parameter (rdflib.BRICK attribute), 389

Supply\_Air\_Static\_Pressure\_Proportional\_Band\_Parameter (rdflib.BRICK attribute), 389

Supply\_Air\_Static\_Pressure\_Sensor (rdflib.BRICK attribute), 389

Supply\_Air\_Static\_Pressure\_Setpoint (rdflib.BRICK attribute), 389

Supply\_Air\_Temperature\_Alarm (rdflib.BRICK attribute), 389

Supply\_Air\_Temperature\_Deadband\_Setpoint (rdflib.BRICK attribute), 389

Supply\_Air\_Temperature\_High\_Reset\_Setpoint (rdflib.BRICK attribute), 389

Supply\_Air\_Temperature\_Low\_Reset\_Setpoint (rdflib.BRICK attribute), 390

Supply\_Air\_Temperature\_Proportional\_Band\_Parameter (rdflib.BRICK attribute), 390

Supply\_Air\_Temperature\_Reset\_Differential\_Setpoint (rdflib.BRICK attribute), 390

Supply\_Air\_Temperature\_Sensor (rdflib.BRICK attribute), 390

Supply\_Air\_Temperature\_Setpoint (rdflib.BRICK attribute), 390

Supply\_Air\_Temperature\_Step\_Parameter (rdflib.BRICK attribute), 390

Supply\_Air\_Velocity\_Pressure\_Sensor (rdflib.BRICK attribute), 390

Supply\_Chilled\_Water (rdflib.BRICK attribute), 390

Supply\_Chilled\_Water\_Temperature\_Setpoint (rdflib.BRICK attribute), 390

Supply\_Condenser\_Water (rdflib.BRICK attribute), 390

Supply\_Condenser\_Water\_Flow\_Sensor (rdflib.BRICK attribute), 390

Supply\_Condenser\_Water\_Temperature\_Sensor (rdflib.BRICK attribute), 390

Supply\_Condenser\_Water\_Temperature\_Setpoint (rdflib.BRICK attribute), 390

Supply\_Fan (rdflib.BRICK attribute), 390

Supply\_Hot\_Water (rdflib.BRICK attribute), 390

Supply\_Hot\_Water\_Temperature\_Setpoint (rdflib.BRICK attribute), 390

Supply\_Water (rdflib.BRICK attribute), 390

Supply\_Water\_Differential\_Pressure\_Deadband\_Setpoint (rdflib.BRICK attribute), 390

Supply\_Water\_Differential\_Pressure\_Integral\_Time\_Parameter (rdflib.BRICK attribute), 391

Supply\_Water\_Differential\_Pressure\_Proportional\_Band\_Parameter (rdflib.BRICK attribute), 391

Supply\_Water\_Flow\_Sensor (rdflib.BRICK attribute), 391

Supply\_Water\_Flow\_Setpoint (rdflib.BRICK attribute), 391

Supply\_Water\_Temperature\_Alarm (rdflib.BRICK attribute), 391

Supply\_Water\_Temperature\_Deadband\_Setpoint (rdflib.BRICK attribute), 391

Supply\_Water\_Temperature\_Integral\_Time\_Parameter (rdflib.BRICK attribute), 391

Supply\_Water\_Temperature\_Proportional\_Band\_Parameter (rdflib.BRICK attribute), 391

Supply\_Water\_Temperature\_Setpoint (rdflib.BRICK attribute), 391

supplyTo (rdflib.SDO attribute), 582

support (rdflib.ODRL2 attribute), 460

supportingData (rdflib.SDO attribute), 582

suppress\_warnings\_ (rdflib.plugins.sparql.parserutils.ParamList attribute), 172

suppressOutput (rdflib.CSVW attribute), 403

surface (rdflib.SDO attribute), 582

surgical (rdflib.SDO attribute), 524

SurgicalProcedure (rdflib.SDO attribute), 524

surname (rdflib.FOAF attribute), 422

Surveillance\_Camera (rdflib.BRICK attribute), 391

SuspendAction (rdflib.SDO attribute), 524

Suspended (rdflib.SDO attribute), 524

SVNRepository (rdflib.DOAP attribute), 414

Switch (rdflib.BRICK attribute), 391

Switch\_Room (rdflib.BRICK attribute), 391

Switchgear (rdflib.BRICK attribute), 391

SymmetricProperty (rdflib.OWL attribute), 464

SymptomsHealthAspect (rdflib.SDO attribute), 524

Synagogue (rdflib.SDO attribute), 524

sync() (rdflib.plugins.stores.berkeleydb.BerkeleyDB method), 195

synchronize (rdflib.ODRL2 attribute), 460

System (rdflib.BRICK attribute), 391

system (*rdflib.ODRL2 attribute*), 460  
 System (*rdflib.SSN attribute*), 599  
 System\_Enable\_Command (*rdflib.BRICK attribute*), 391  
 System\_Shutdown\_Status (*rdflib.BRICK attribute*), 391  
 System\_Status (*rdflib.BRICK attribute*), 391  
 systemDevice (*rdflib.ODRL2 attribute*), 460

## T

Table (*rdflib.CSVW attribute*), 400  
 table (*rdflib.CSVW attribute*), 403  
 Table (*rdflib.SDO attribute*), 524  
 tableDirection (*rdflib.CSVW attribute*), 403  
 TableGroup (*rdflib.CSVW attribute*), 400  
 tableOfContents (*rdflib.DCTERMS attribute*), 413  
 TableReference (*rdflib.CSVW attribute*), 401  
 tableSchema (*rdflib.CSVW attribute*), 403  
 TABS\_Panel (*rdflib.BRICK attribute*), 391  
 tabularMetadata (*rdflib.CSVW attribute*), 403  
 TakeAction (*rdflib.SDO attribute*), 525  
 target (*rdflib.ODRL2 attribute*), 460  
 target (*rdflib.SDO attribute*), 582  
 Target (*rdflib.SH attribute*), 591  
 target (*rdflib.SH attribute*), 595  
 target (*rdflib.VOID attribute*), 608  
 targetClass (*rdflib.SH attribute*), 595  
 targetCollection (*rdflib.SDO attribute*), 582  
 targetDescription (*rdflib.SDO attribute*), 582  
 targetFormat (*rdflib.CSVW attribute*), 403  
 targetIndividual (*rdflib.OWL attribute*), 466  
 targetName (*rdflib.SDO attribute*), 582  
 targetNode (*rdflib.SH attribute*), 595  
 targetObjectsOf (*rdflib.SH attribute*), 595  
 targetPlatform (*rdflib.SDO attribute*), 582  
 targetPopulation (*rdflib.SDO attribute*), 582  
 targetProduct (*rdflib.SDO attribute*), 582  
 targetSubjectsOf (*rdflib.SH attribute*), 595  
 TargetType (*rdflib.SH attribute*), 591  
 targetUrl (*rdflib.SDO attribute*), 583  
 targetValue (*rdflib.OWL attribute*), 466  
 TattooParlor (*rdflib.SDO attribute*), 525  
 Taxi (*rdflib.SDO attribute*), 525  
 taxID (*rdflib.SDO attribute*), 583  
 TaxiReservation (*rdflib.SDO attribute*), 525  
 TaxiService (*rdflib.SDO attribute*), 525  
 TaxiStand (*rdflib.SDO attribute*), 525  
 TaxiVehicleUsage (*rdflib.SDO attribute*), 525  
 Taxon (*rdflib.SDO attribute*), 525  
 taxonomicRange (*rdflib.SDO attribute*), 583  
 taxonRank (*rdflib.SDO attribute*), 583  
 teaches (*rdflib.SDO attribute*), 583  
 Team\_Room (*rdflib.BRICK attribute*), 392  
 TechArticle (*rdflib.SDO attribute*), 525  
 TechnicalFeature (*rdflib.VOID attribute*), 607  
 Telecom\_Room (*rdflib.BRICK attribute*), 392  
 telephone (*rdflib.SDO attribute*), 583  
 TelevisionChannel (*rdflib.SDO attribute*), 525  
 TelevisionStation (*rdflib.SDO attribute*), 525  
 Temperature\_Alarm (*rdflib.BRICK attribute*), 392  
 Temperature\_Deadband\_Setpoint (*rdflib.BRICK attribute*), 392  
 Temperature\_Differential\_Reset\_Setpoint (*rdflib.BRICK attribute*), 392  
 Temperature\_High\_Reset\_Setpoint (*rdflib.BRICK attribute*), 392  
 Temperature\_Low\_Reset\_Setpoint (*rdflib.BRICK attribute*), 392  
 Temperature\_Parameter (*rdflib.BRICK attribute*), 392  
 Temperature\_Sensor (*rdflib.BRICK attribute*), 392  
 Temperature\_Setpoint (*rdflib.BRICK attribute*), 392  
 Temperature\_Step\_Parameter (*rdflib.BRICK attribute*), 392  
 Temperature\_Tolerance\_Parameter (*rdflib.BRICK attribute*), 392  
 temperatureCoefficientofPmax (*rdflib.BRICK attribute*), 400  
 temporal (*rdflib.DCTERMS attribute*), 413  
 temporal (*rdflib.SDO attribute*), 583  
 temporalCoverage (*rdflib.SDO attribute*), 583  
 TemporalDuration (*rdflib.TIME attribute*), 600  
 TemporalEntity (*rdflib.TIME attribute*), 600  
 TemporalPosition (*rdflib.TIME attribute*), 601  
 temporalResolution (*rdflib.DCAT attribute*), 409  
 TemporalUnit (*rdflib.TIME attribute*), 601  
 Temporary\_Occupancy\_Status (*rdflib.BRICK attribute*), 392  
 TennisComplex (*rdflib.SDO attribute*), 525  
 Term (*class in rdflib.plugins.shared.jsonld.context*), 129  
 term() (*rdflib.extras.infixowl.ClassNamespaceFactory method*), 62  
 term() (*rdflib.Namespace method*), 453  
 term() (*rdflib.namespace.ClosedNamespace method*), 72  
 term() (*rdflib.namespace.Namespace method*), 74  
 termCode (*rdflib.SDO attribute*), 583  
 termDuration (*rdflib.SDO attribute*), 583  
 termGroup (*rdflib.VANN attribute*), 606  
 Terminal\_Unit (*rdflib.BRICK attribute*), 392  
 Terminated (*rdflib.SDO attribute*), 525  
 termsOfService (*rdflib.SDO attribute*), 583  
 termsPerYear (*rdflib.SDO attribute*), 583  
 termToJSON() (*in module rdflib.plugins.sparql.results.jsonresults*), 134  
 tester (*rdflib.DOAP attribute*), 415  
 TETRA\_Room (*rdflib.BRICK attribute*), 392  
 Text (*rdflib.DCMITYPE attribute*), 410  
 Text (*rdflib.SDO attribute*), 525  
 text (*rdflib.SDO attribute*), 583



- `text()` (*rdflib.plugins.serializers.xmlwriter.XMLWriter* method), 124
- `TextDigitalDocument` (*rdflib.SDO* attribute), 525
- `textDirection` (*rdflib.CSVW* attribute), 403
- `textToSpeech` (*rdflib.ODRL2* attribute), 460
- `textValue` (*rdflib.SDO* attribute), 583
- `TGN` (*rdflib.DCTERMS* attribute), 411
- `thaw()` (*rdflib.plugins.sparql.sparql.QueryContext* method), 184
- `TheaterEvent` (*rdflib.SDO* attribute), 525
- `TheaterGroup` (*rdflib.SDO* attribute), 525
- `theme` (*rdflib.DCAT* attribute), 409
- `theme` (*rdflib.FOAF* attribute), 422
- `themeTaxonomy` (*rdflib.DCAT* attribute), 409
- `Therapeutic` (*rdflib.SDO* attribute), 525
- `TherapeuticProcedure` (*rdflib.SDO* attribute), 525
- `Thermal_Power_Meter` (*rdflib.BRICK* attribute), 392
- `Thermal_Power_Sensor` (*rdflib.BRICK* attribute), 392
- `Thermally_Activated_Building_System_Panel` (*rdflib.BRICK* attribute), 393
- `thermalTransmittance` (*rdflib.BRICK* attribute), 400
- `Thermostat` (*rdflib.BRICK* attribute), 393
- `Thesis` (*rdflib.SDO* attribute), 525
- `Thing` (*rdflib.OWL* attribute), 464
- `Thing` (*rdflib.SDO* attribute), 525
- `this` (*rdflib.SH* attribute), 595
- `Throat` (*rdflib.SDO* attribute), 525
- `thumbnail` (*rdflib.FOAF* attribute), 423
- `thumbnail` (*rdflib.SDO* attribute), 583
- `thumbnailUrl` (*rdflib.SDO* attribute), 583
- `Thursday` (*rdflib.SDO* attribute), 525
- `Thursday` (*rdflib.TIME* attribute), 601
- `tickerSymbol` (*rdflib.SDO* attribute), 583
- `Ticket` (*rdflib.ODRL2* attribute), 454
- `Ticket` (*rdflib.SDO* attribute), 525
- `ticketedSeat` (*rdflib.SDO* attribute), 583
- `Ticketing_Booth` (*rdflib.BRICK* attribute), 393
- `ticketNumber` (*rdflib.SDO* attribute), 583
- `ticketToken` (*rdflib.SDO* attribute), 583
- `TieAction` (*rdflib.SDO* attribute), 525
- `tilt` (*rdflib.BRICK* attribute), 400
- `TIME` (class in *rdflib*), 600
- `Time` (*rdflib.SDO* attribute), 525
- `time` (*rdflib.XSD* attribute), 611
- `Time_Parameter` (*rdflib.BRICK* attribute), 393
- `Time_Setpoint` (*rdflib.BRICK* attribute), 393
- `timedCount` (*rdflib.ODRL2* attribute), 460
- `timeInterval` (*rdflib.ODRL2* attribute), 460
- `timeOfDay` (*rdflib.SDO* attribute), 583
- `TimePosition` (*rdflib.TIME* attribute), 601
- `timeRequired` (*rdflib.SDO* attribute), 583
- `timeseries` (*rdflib.BRICK* attribute), 400
- `timeToComplete` (*rdflib.SDO* attribute), 583
- `TimeZone` (*rdflib.TIME* attribute), 601
- `timeZone` (*rdflib.TIME* attribute), 603
- `timezoneOffset` (*rdflib.XSD* attribute), 611
- `TipAction` (*rdflib.SDO* attribute), 525
- `tipjar` (*rdflib.FOAF* attribute), 423
- `TireShop` (*rdflib.SDO* attribute), 525
- `tissueSample` (*rdflib.SDO* attribute), 583
- `title` (*rdflib.CSVW* attribute), 403
- `title` (*rdflib.DC* attribute), 408
- `title` (*rdflib.DCTERMS* attribute), 413
- `title` (*rdflib.FOAF* attribute), 423
- `title` (*rdflib.Namespace* property), 453
- `title` (*rdflib.namespace.Namespace* property), 74
- `title` (*rdflib.SDO* attribute), 583
- `titleEIDR` (*rdflib.SDO* attribute), 583
- `to_canonical_graph()` (in module *rdflib.compare*), 229
- `to_dict()` (*rdflib.plugins.shared.jsonld.context.Context* method), 129
- `to_isomorphic()` (in module *rdflib.compare*), 229
- `to_rdf()` (in module *rdflib.plugins.parsers.jsonld*), 81
- `to_symbol()` (*rdflib.plugins.shared.jsonld.context.Context* method), 129
- `to_term()` (in module *rdflib.util*), 333
- `tocContinuation` (*rdflib.SDO* attribute), 583
- `tocEntry` (*rdflib.SDO* attribute), 583
- `todo` (*rdflib.PROV* attribute), 474
- `tok()` (*rdflib.plugins.parsers.notation3.SinkParser* method), 93
- `token` (*rdflib.XSD* attribute), 611
- `Tolerance_Parameter` (*rdflib.BRICK* attribute), 393
- `TollFree` (*rdflib.SDO* attribute), 526
- `toLocation` (*rdflib.SDO* attribute), 583
- `ToMultiSet()` (in module *rdflib.plugins.sparql.algebra*), 147
- `tongueWeight` (*rdflib.SDO* attribute), 584
- `tool` (*rdflib.SDO* attribute), 584
- `topClasses` (*rdflib.plugins.serializers.turtle.RecursiveSerializer* attribute), 122
- `topConceptOf` (*rdflib.SKOS* attribute), 597
- `topDataProperty` (*rdflib.OWL* attribute), 466
- `topic` (*rdflib.FOAF* attribute), 423
- `topic_interest` (*rdflib.FOAF* attribute), 423
- `topObjectProperty` (*rdflib.OWL* attribute), 466
- `toPython()` (*rdflib.Graph* method), 438
- `toPython()` (*rdflib.graph.Graph* method), 265
- `toPython()` (*rdflib.graph.Seq* method), 275
- `toPython()` (*rdflib IdentifiedNode* method), 441
- `toPython()` (*rdflib.Literal* method), 451
- `toPython()` (*rdflib.term IdentifiedNode* method), 315
- `toPython()` (*rdflib.term.Literal* method), 326
- `toPython()` (*rdflib.term.Variable* method), 329
- `toPython()` (*rdflib.Variable* method), 609
- `toRecipient` (*rdflib.SDO* attribute), 583
- `torque` (*rdflib.SDO* attribute), 584

- Torque\_Sensor (*rdflib.BRICK* attribute), 393  
 totalDigits (*rdflib.XSD* attribute), 611  
 totalJobOpenings (*rdflib.SDO* attribute), 584  
 totalPaymentDue (*rdflib.SDO* attribute), 584  
 totalPrice (*rdflib.SDO* attribute), 584  
 totalTime (*rdflib.SDO* attribute), 584  
 Touchpanel (*rdflib.BRICK* attribute), 393  
 tourBookingPage (*rdflib.SDO* attribute), 584  
 TouristAttraction (*rdflib.SDO* attribute), 526  
 TouristDestination (*rdflib.SDO* attribute), 526  
 TouristInformationCenter (*rdflib.SDO* attribute), 526  
 TouristTrip (*rdflib.SDO* attribute), 526  
 touristType (*rdflib.SDO* attribute), 584  
 Toxicologic (*rdflib.SDO* attribute), 526  
 ToyStore (*rdflib.SDO* attribute), 526  
 Trace\_Heat\_Sensor (*rdflib.BRICK* attribute), 393  
 track (*rdflib.SDO* attribute), 584  
 TrackAction (*rdflib.SDO* attribute), 526  
 trackedParty (*rdflib.ODRL2* attribute), 460  
 trackingNumber (*rdflib.SDO* attribute), 584  
 trackingParty (*rdflib.ODRL2* attribute), 460  
 trackingUrl (*rdflib.SDO* attribute), 584  
 tracks (*rdflib.SDO* attribute), 584  
 TradeAction (*rdflib.SDO* attribute), 526  
 TraditionalChinese (*rdflib.SDO* attribute), 526  
 trailer (*rdflib.SDO* attribute), 584  
 trailerWeight (*rdflib.SDO* attribute), 584  
 trainingSalary (*rdflib.SDO* attribute), 584  
 trainName (*rdflib.SDO* attribute), 584  
 trainNumber (*rdflib.SDO* attribute), 584  
 TrainReservation (*rdflib.SDO* attribute), 526  
 TrainStation (*rdflib.SDO* attribute), 526  
 TrainTrip (*rdflib.SDO* attribute), 526  
 transaction\_aware (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* attribute), 195  
 transaction\_aware (*rdflib.plugins.stores.sparqlstore.SPARQLStore* attribute), 210  
 transaction\_aware (*rdflib.store.Store* attribute), 311  
 transcript (*rdflib.SDO* attribute), 584  
 transFatContent (*rdflib.SDO* attribute), 584  
 transfer (*rdflib.ODRL2* attribute), 460  
 TransferAction (*rdflib.SDO* attribute), 526  
 transform (*rdflib.ODRL2* attribute), 460  
 Transformation (*rdflib.CSVW* attribute), 401  
 transformations (*rdflib.CSVW* attribute), 403  
 TransformedContent (*rdflib.SDO* attribute), 526  
 Transformer (*rdflib.BRICK* attribute), 393  
 Transformer\_Room (*rdflib.BRICK* attribute), 393  
 transitive\_objects() (*rdflib.Graph* method), 439  
 transitive\_objects() (*rdflib.graph.Graph* method), 266  
 transitive\_subjects() (*rdflib.resource.Resource* method), 305  
 transitive\_closure() (*rdflib.Graph* method), 438  
 transitive\_closure() (*rdflib.graph.Graph* method), 265  
 TransitiveProperty (*rdflib.OWL* attribute), 464  
 transitiveSubOrganizationOf (*rdflib.ORG* attribute), 462  
 TransitMap (*rdflib.SDO* attribute), 526  
 transitTime (*rdflib.SDO* attribute), 584  
 transitTimeLabel (*rdflib.SDO* attribute), 584  
 translate (*rdflib.ODRL2* attribute), 460  
 translate() (in module *rdflib.plugins.sparql.algebra*), 149  
 translateAggregates() (in module *rdflib.plugins.sparql.algebra*), 149  
 translateAlgebra() (in module *rdflib.plugins.sparql.algebra*), 149  
 translateExists() (in module *rdflib.plugins.sparql.algebra*), 149  
 translateGraphGraphPattern() (in module *rdflib.plugins.sparql.algebra*), 149  
 translateGroupGraphPattern() (in module *rdflib.plugins.sparql.algebra*), 149  
 translateGroupOrUnionGraphPattern() (in module *rdflib.plugins.sparql.algebra*), 149  
 translateInlineData() (in module *rdflib.plugins.sparql.algebra*), 150  
 translatePath() (in module *rdflib.plugins.sparql.algebra*), 150  
 translatePName() (in module *rdflib.plugins.sparql.algebra*), 150  
 translatePrologue() (in module *rdflib.plugins.sparql.algebra*), 150  
 translateQuads() (in module *rdflib.plugins.sparql.algebra*), 150  
 translateQuery() (in module *rdflib.plugins.sparql.algebra*), 150  
 translateUpdate() (in module *rdflib.plugins.sparql.algebra*), 151  
 translateUpdate1() (in module *rdflib.plugins.sparql.algebra*), 151  
 translateValues() (in module *rdflib.plugins.sparql.algebra*), 151  
 translationOfWork (*rdflib.SDO* attribute), 584  
 translator (*rdflib.DOAP* attribute), 415  
 translator (*rdflib.SDO* attribute), 584  
 transmissionMethod (*rdflib.SDO* attribute), 584  
 TravelAction (*rdflib.SDO* attribute), 526

- TravelAgency (*rdflib.SDO* attribute), 526
- travelBans (*rdflib.SDO* attribute), 584
- traverse() (in module *rdflib.plugins.sparql.algebra*), 151
- TreatmentIndication (*rdflib.SDO* attribute), 526
- TreatmentsHealthAspect (*rdflib.SDO* attribute), 526
- trialDesign (*rdflib.SDO* attribute), 584
- tributary (*rdflib.SDO* attribute), 584
- TrigParser (class in *rdflib.plugins.parsers.trig*), 108
- TrigSerializer (class in *rdflib.plugins.serializers.trig*), 120
- TrigSinkParser (class in *rdflib.plugins.parsers.trig*), 109
- trim (*rdflib.CSVW* attribute), 403
- Trip (*rdflib.SDO* attribute), 526
- triple() (*rdflib.plugins.parsers.ntriples.DummySink* method), 99
- triple() (*rdflib.plugins.parsers.ntriples.NTGraphSink* method), 99
- triple() (*rdflib.tools.csv2rdf.CSV2RDF* method), 219
- TripleAddedEvent (class in *rdflib.store*), 312
- TripleBlindedTrial (*rdflib.SDO* attribute), 526
- TripleRemovedEvent (class in *rdflib.store*), 313
- TripleRule (*rdflib.SH* attribute), 591
- triples (*rdflib.VOID* attribute), 608
- triples() (in module *rdflib.plugins.sparql.algebra*), 151
- triples() (*rdflib.ConjunctiveGraph* method), 407
- triples() (*rdflib.Graph* method), 440
- triples() (*rdflib.graph.ConjunctiveGraph* method), 245
- triples() (*rdflib.graph.Graph* method), 267
- triples() (*rdflib.graph.ReadOnlyGraphAggregate* method), 274
- triples() (*rdflib.plugins.stores.auditable.AuditableStore* method), 191
- triples() (*rdflib.plugins.stores.berkeleydb.BerkeleyDB* method), 195
- triples() (*rdflib.plugins.stores.concurrent.ConcurrentStore* method), 195
- triples() (*rdflib.plugins.stores.memory.Memory* method), 198
- triples() (*rdflib.plugins.stores.memory.SimpleMemory* method), 201
- triples() (*rdflib.plugins.stores.regexmatching.REGEXMatching* method), 202
- triples() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 210
- triples() (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* method), 216
- triples() (*rdflib.store.Store* method), 311
- triples\_choices() (*rdflib.ConjunctiveGraph* method), 407
- triples\_choices() (*rdflib.Graph* method), 440
- triples\_choices() (*rdflib.graph.ConjunctiveGraph* method), 245
- triples\_choices() (*rdflib.graph.Graph* method), 267
- triples\_choices() (*rdflib.graph.ReadOnlyGraphAggregate* method), 274
- triples\_choices() (*rdflib.plugins.stores.sparqlstore.SPARQLStore* method), 211
- triples\_choices() (*rdflib.store.Store* method), 312
- TriXHandler (class in *rdflib.plugins.parsers.trix*), 110
- TriXParser (class in *rdflib.plugins.parsers.trix*), 112
- TriXSerializer (class in *rdflib.plugins.serializers.trix*), 120
- TRS (*rdflib.TIME* attribute), 600
- TSVResultParser (class in *rdflib.plugins.sparql.results.tsvresults*), 135
- Tuesday (*rdflib.SDO* attribute), 526
- Tuesday (*rdflib.TIME* attribute), 601
- Tunnel (*rdflib.BRICK* attribute), 393
- TurtleParser (class in *rdflib.plugins.parsers.notation3*), 95
- TurtleSerializer (class in *rdflib.plugins.serializers.turtle*), 122
- TVClip (*rdflib.SDO* attribute), 524
- TVEpisode (*rdflib.SDO* attribute), 524
- TVOC\_Level\_Sensor (*rdflib.BRICK* attribute), 392
- TVOC\_Sensor (*rdflib.BRICK* attribute), 392
- TVSeason (*rdflib.SDO* attribute), 524
- TVSeries (*rdflib.SDO* attribute), 524
- TXTResultSerializer (class in *rdflib.plugins.sparql.results.txtresults*), 136
- type (*rdflib.DC* attribute), 408
- type (*rdflib.DCTERMS* attribute), 413
- type (*rdflib.extras.infixowl.Individual* property), 65
- type (*rdflib.plugins.shared.jsonld.context.Term* attribute), 130
- type (*rdflib.RDF* attribute), 478
- type\_key (*rdflib.plugins.shared.jsonld.context.Context* property), 129
- type\_of\_container() (*rdflib.container.Container* method), 233
- type\_promotion() (in module *rdflib.plugins.sparql.datatypes*), 152
- type\_safe\_numbers() (in module *rdflib.plugins.sparql.aggregates*), 145
- TypeAndQuantityNode (*rdflib.SDO* attribute), 526
- typeOfBed (*rdflib.SDO* attribute), 585
- typeOfGood (*rdflib.SDO* attribute), 585
- TypesHealthAspect (*rdflib.SDO* attribute), 526
- typicalAgeRange (*rdflib.SDO* attribute), 585
- typicalCreditsPerTerm (*rdflib.SDO* attribute), 585
- typicalTest (*rdflib.SDO* attribute), 585



## U

- UDC (*rdflib.DCTERMS* attribute), 411
- UEscape() (*rdflib.plugins.parsers.notation3.SinkParser* method), 87
- uEscape() (*rdflib.plugins.parsers.notation3.SinkParser* method), 94
- uid (*rdflib.ODRL2* attribute), 460
- UKNonprofitType (*rdflib.SDO* attribute), 527
- UKTrust (*rdflib.SDO* attribute), 527
- Ultrasound (*rdflib.SDO* attribute), 527
- UnaryMinus() (in module *rdflib.plugins.sparql.operators*), 164
- UnaryNot() (in module *rdflib.plugins.sparql.operators*), 165
- UnaryPlus() (in module *rdflib.plugins.sparql.operators*), 165
- undefined (*rdflib.ODRL2* attribute), 460
- UndefinedTerm (*rdflib.ODRL2* attribute), 454
- Underfloor\_Air\_Plenum (*rdflib.BRICK* attribute), 393
- Underfloor\_Air\_Plenum\_Static\_Pressure\_Sensor (*rdflib.BRICK* attribute), 393
- Underfloor\_Air\_Plenum\_Static\_Pressure\_Setpoint (*rdflib.BRICK* attribute), 393
- Underfloor\_Air\_Temperature\_Sensor (*rdflib.BRICK* attribute), 393
- underName (*rdflib.SDO* attribute), 585
- UnemploymentSupport (*rdflib.SDO* attribute), 527
- UnincorporatedAssociationCharity (*rdflib.SDO* attribute), 527
- uninstall (*rdflib.ODRL2* attribute), 460
- union (*rdflib.SH* attribute), 595
- Union() (in module *rdflib.plugins.sparql.algebra*), 148
- unionOf (*rdflib.OWL* attribute), 466
- uniq() (in module *rdflib.util*), 333
- uniqueLang (*rdflib.SH* attribute), 595
- UniqueLangConstraintComponent (*rdflib.SH* attribute), 591
- UniquenessError, 235
- uniqueURI() (in module *rdflib.plugins.parsers.notation3*), 97
- unit (*rdflib.ODRL2* attribute), 460
- Unit\_Failure\_Alarm (*rdflib.BRICK* attribute), 393
- unitCode (*rdflib.SDO* attribute), 585
- unitDay (*rdflib.TIME* attribute), 603
- unitHour (*rdflib.TIME* attribute), 603
- unitMinute (*rdflib.TIME* attribute), 603
- unitMonth (*rdflib.TIME* attribute), 603
- unitOf (*rdflib.ORG* attribute), 462
- unitOfCount (*rdflib.ODRL2* attribute), 460
- UnitPriceSpecification (*rdflib.SDO* attribute), 527
- unitSecond (*rdflib.TIME* attribute), 603
- unitText (*rdflib.SDO* attribute), 585
- unitType (*rdflib.TIME* attribute), 603
- unitWeek (*rdflib.TIME* attribute), 603
- unitYear (*rdflib.TIME* attribute), 603
- unnamedSourcesPolicy (*rdflib.SDO* attribute), 585
- Unoccupied\_Air\_Temperature\_Cooling\_Setpoint (*rdflib.BRICK* attribute), 393
- Unoccupied\_Air\_Temperature\_Heating\_Setpoint (*rdflib.BRICK* attribute), 393
- Unoccupied\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 394
- Unoccupied\_Cooling\_Discharge\_Air\_Flow\_Setpoint (*rdflib.BRICK* attribute), 394
- Unoccupied\_Discharge\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 394
- Unoccupied\_Load\_Shed\_Command (*rdflib.BRICK* attribute), 394
- Unoccupied\_Return\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 394
- Unoccupied\_Room\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 394
- Unoccupied\_Supply\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 394
- Unoccupied\_Zone\_Air\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 394
- UnofficialLegalValue (*rdflib.SDO* attribute), 527
- unqualifiedForm (*rdflib.PROV* attribute), 474
- unquote() (in module *rdflib.plugins.parsers.ntriples*), 101
- unregister\_custom\_function() (in module *rdflib.plugins.sparql.operators*), 168
- UnRegisterAction (*rdflib.SDO* attribute), 527
- unsaturatedFatContent (*rdflib.SDO* attribute), 585
- unsignedByte (*rdflib.XSD* attribute), 611
- unsignedInt (*rdflib.XSD* attribute), 611
- unsignedLong (*rdflib.XSD* attribute), 611
- unsignedShort (*rdflib.XSD* attribute), 612
- UnsupportedAggregateOperation, 275
- Update (class in *rdflib.plugins.sparql.sparql*), 185
- update (*rdflib.SH* attribute), 595
- update() (*rdflib.Graph* method), 440
- update() (*rdflib.graph.Graph* method), 267
- update() (*rdflib.plugins.sparql.aggregates.Aggregator* method), 141
- update() (*rdflib.plugins.sparql.aggregates.Average* method), 141
- update() (*rdflib.plugins.sparql.aggregates.Counter* method), 142
- update() (*rdflib.plugins.sparql.aggregates.Extremum* method), 143
- update() (*rdflib.plugins.sparql.aggregates.GroupConcat* method), 143
- update() (*rdflib.plugins.sparql.aggregates.Sample* method), 145
- update() (*rdflib.plugins.sparql.aggregates.Sum* method), 145
- update() (*rdflib.plugins.sparql.processor.SPARQLUpdateProcessor*

- method*), 175
  - `update()` (*rdflib.plugins.stores.memory.Memory* *method*), 199
  - `update()` (*rdflib.plugins.stores.memory.SimpleMemory* *method*), 201
  - `update()` (*rdflib.plugins.stores.sparqlconnector.SPARQLConnector* *method*), 204
  - `update()` (*rdflib.plugins.stores.sparqlstore.SPARQLStore* *method*), 211
  - `update()` (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore* *method*), 217
  - `update()` (*rdflib.query.UpdateProcessor* *method*), 298
  - `update()` (*rdflib.store.Store* *method*), 312
  - `UpdateAction` (*rdflib.SDO* attribute), 527
  - `UpdateProcessor` (*class in rdflib.query*), 297
  - `uploadDate` (*rdflib.SDO* attribute), 585
  - `upvoteCount` (*rdflib.SDO* attribute), 585
  - `URI` (*rdflib.DCTERMS* attribute), 411
  - `uri` (*rdflib.namespace.ClosedNamespace* property), 72
  - `uri_ref2()` (*rdflib.plugins.parsers.notation3.SinkParser* *method*), 94
  - `uriLookupEndpoint` (*rdflib.VOID* attribute), 608
  - `uriOf()` (*rdflib.plugins.parsers.notation3.SinkParser* *method*), 94
  - `uriquote()` (*in module rdflib.plugins.parsers.ntriples*), 102
  - `URIRef` (*class in rdflib*), 604
  - `URIRef` (*class in rdflib.term*), 326
  - `uriref()` (*rdflib.plugins.parsers.ntriples.W3CNTriplesParser* *method*), 101
  - `uriRegexPattern` (*rdflib.VOID* attribute), 608
  - `uriSpace` (*rdflib.VOID* attribute), 608
  - `uriTemplate` (*rdflib.CSVW* attribute), 403
  - `url` (*rdflib.CSVW* attribute), 403
  - `URL` (*rdflib.SDO* attribute), 527
  - `url` (*rdflib.SDO* attribute), 585
  - `URLInputSource` (*class in rdflib.parser*), 278
  - `urlTemplate` (*rdflib.SDO* attribute), 585
  - `Urologic` (*rdflib.SDO* attribute), 527
  - `Usage` (*rdflib.PROV* attribute), 470
  - `Usage_Sensor` (*rdflib.BRICK* attribute), 394
  - `usageInfo` (*rdflib.SDO* attribute), 585
  - `usageNote` (*rdflib.VANN* attribute), 606
  - `UsageOrScheduleHealthAspect` (*rdflib.SDO* attribute), 527
  - `use` (*rdflib.ODRL2* attribute), 460
  - `use_row()` (*rdflib.plugins.sparql.aggregates.Accumulator* *method*), 140
  - `use_row()` (*rdflib.plugins.sparql.aggregates.Counter* *method*), 142
  - `UseAction` (*rdflib.SDO* attribute), 527
  - `used` (*rdflib.PROV* attribute), 474
  - `UsedCondition` (*rdflib.SDO* attribute), 527
  - `usedProcedure` (*rdflib.SOSA* attribute), 599
  - `usedToDiagnose` (*rdflib.SDO* attribute), 585
  - `UserBlocks` (*rdflib.SDO* attribute), 527
  - `UserChecksins` (*rdflib.SDO* attribute), 527
  - `UserComments` (*rdflib.SDO* attribute), 527
  - `UserDownloads` (*rdflib.SDO* attribute), 527
  - `UserInteraction` (*rdflib.SDO* attribute), 527
  - `userInteractionCount` (*rdflib.SDO* attribute), 585
  - `UserLikes` (*rdflib.SDO* attribute), 527
  - `UserPageVisits` (*rdflib.SDO* attribute), 527
  - `UserPlays` (*rdflib.SDO* attribute), 527
  - `UserPlusOnes` (*rdflib.SDO* attribute), 527
  - `UserReview` (*rdflib.SDO* attribute), 527
  - `UserTweets` (*rdflib.SDO* attribute), 527
  - `usesDevice` (*rdflib.SDO* attribute), 585
  - `usesHealthPlanIdStandard` (*rdflib.SDO* attribute), 585
  - `USNonprofitType` (*rdflib.SDO* attribute), 527
  - `utterances` (*rdflib.SDO* attribute), 585
- ## V
- `valid` (*rdflib.DCTERMS* attribute), 414
  - `validate_namespace()` (*in module rdflib.tools.defined\_namespace\_creator*), 219
  - `validate_object_id()` (*in module rdflib.tools.defined\_namespace\_creator*), 220
  - `ValidationReport` (*rdflib.SH* attribute), 591
  - `ValidationResult` (*rdflib.SH* attribute), 591
  - `Validator` (*rdflib.SH* attribute), 591
  - `validator` (*rdflib.SH* attribute), 595
  - `validFor` (*rdflib.SDO* attribute), 585
  - `validFrom` (*rdflib.SDO* attribute), 585
  - `validIn` (*rdflib.SDO* attribute), 585
  - `validThrough` (*rdflib.SDO* attribute), 585
  - `validUntil` (*rdflib.SDO* attribute), 585
  - `value` (*rdflib.BRICK* attribute), 400
  - `value` (*rdflib.Literal* property), 451
  - `value` (*rdflib.plugins.sparql.aggregates.Maximum* attribute), 144
  - `value` (*rdflib.plugins.sparql.aggregates.Minimum* attribute), 144
  - `value` (*rdflib.PROV* attribute), 474
  - `value` (*rdflib.RDF* attribute), 478
  - `value` (*rdflib.SDO* attribute), 585
  - `value` (*rdflib.SH* attribute), 595
  - `value` (*rdflib.term.Literal* property), 326
  - `value()` (*in module rdflib.plugins.sparql.parserutils*), 173
  - `value()` (*rdflib.extras.describer.Describer* *method*), 51
  - `value()` (*rdflib.Graph* *method*), 441
  - `value()` (*rdflib.graph.Graph* *method*), 268
  - `value()` (*rdflib.resource.Resource* *method*), 305
  - `value_key` (*rdflib.plugins.shared.jsonld.context.Context* property), 129
  - `valueAddedTaxIncluded` (*rdflib.SDO* attribute), 585



- `valueMaxLength` (*rdflib.SDO attribute*), 586
- `valueMinLength` (*rdflib.SDO attribute*), 586
- `valueName` (*rdflib.SDO attribute*), 586
- `valuePattern` (*rdflib.SDO attribute*), 586
- `valueReference` (*rdflib.SDO attribute*), 586
- `valueRequired` (*rdflib.SDO attribute*), 586
- `Values()` (in module *rdflib.plugins.sparql.algebra*), 148
- `valueUrl` (*rdflib.CSVW attribute*), 403
- `Valve` (*rdflib.BRICK attribute*), 394
- `Valve_Command` (*rdflib.BRICK attribute*), 394
- `Valve_Position_Sensor` (*rdflib.BRICK attribute*), 394
- `VANN` (class in *rdflib*), 606
- `Variable` (class in *rdflib*), 608
- `Variable` (class in *rdflib.term*), 329
- `variable()` (*rdflib.plugins.parsers.notation3.SinkParser method*), 94
- `Variable_Air_Volume_Box` (*rdflib.BRICK attribute*), 394
- `Variable_Air_Volume_Box_With_Reheat` (*rdflib.BRICK attribute*), 394
- `Variable_Frequency_Drive` (*rdflib.BRICK attribute*), 394
- `variableMeasured` (*rdflib.SDO attribute*), 586
- `variantCover` (*rdflib.SDO attribute*), 586
- `variesBy` (*rdflib.SDO attribute*), 586
- `vars` (*rdflib.plugins.sparql.processor.SPARQLResult attribute*), 174
- `vars` (*rdflib.plugins.sparql.results.jsonresults.JSONResult attribute*), 133
- `vars` (*rdflib.plugins.sparql.results.rdfresults.RDFResult attribute*), 135
- `vars` (*rdflib.plugins.sparql.results.xmlresults.XMLResult attribute*), 138
- `vars` (*rdflib.query.Result attribute*), 294
- `vatID` (*rdflib.SDO attribute*), 586
- `VAV` (*rdflib.BRICK attribute*), 394
- `VeganDiet` (*rdflib.SDO attribute*), 528
- `VegetarianDiet` (*rdflib.SDO attribute*), 528
- `Vehicle` (*rdflib.SDO attribute*), 528
- `vehicleConfiguration` (*rdflib.SDO attribute*), 586
- `vehicleEngine` (*rdflib.SDO attribute*), 586
- `vehicleIdentificationNumber` (*rdflib.SDO attribute*), 586
- `vehicleInteriorColor` (*rdflib.SDO attribute*), 586
- `vehicleInteriorType` (*rdflib.SDO attribute*), 586
- `vehicleModelDate` (*rdflib.SDO attribute*), 586
- `vehicleSeatingCapacity` (*rdflib.SDO attribute*), 586
- `vehicleSpecialUsage` (*rdflib.SDO attribute*), 586
- `vehicleTransmission` (*rdflib.SDO attribute*), 586
- `Vein` (*rdflib.SDO attribute*), 528
- `Velocity_Pressure_Sensor` (*rdflib.BRICK attribute*), 394
- `Velocity_Pressure_Setpoint` (*rdflib.BRICK attribute*), 394
- `vendor` (*rdflib.DOAP attribute*), 415
- `vendor` (*rdflib.SDO attribute*), 586
- `Vent_Operating_Mode_Status` (*rdflib.BRICK attribute*), 395
- `Ventilation_Air_Flow_Ratio_Limit` (*rdflib.BRICK attribute*), 395
- `Ventilation_Air_System` (*rdflib.BRICK attribute*), 395
- `VenueMap` (*rdflib.SDO attribute*), 528
- `verb()` (*rdflib.plugins.parsers.notation3.SinkParser method*), 94
- `verb()` (*rdflib.plugins.serializers.longturtle.LongTurtleSerializer method*), 116
- `verb()` (*rdflib.plugins.serializers.turtle.TurtleSerializer method*), 123
- `verificationFactCheckingPolicy` (*rdflib.SDO attribute*), 586
- `Version` (*rdflib.DOAP attribute*), 414
- `version` (*rdflib.ODRL2 attribute*), 460
- `version` (*rdflib.SDO attribute*), 586
- `versionInfo` (*rdflib.OWL attribute*), 466
- `versionIRI` (*rdflib.OWL attribute*), 466
- `Vertical_Space` (*rdflib.BRICK attribute*), 395
- `Vessel` (*rdflib.SDO attribute*), 528
- `VeterinaryCare` (*rdflib.SDO attribute*), 528
- `VFD` (*rdflib.BRICK attribute*), 394
- `VFD_Enable_Command` (*rdflib.BRICK attribute*), 394
- `vhash()` (*rdflib.tools.graphisomorphism.IsomorphicTestableGraph method*), 220
- `vhashtriple()` (*rdflib.tools.graphisomorphism.IsomorphicTestableGraph method*), 220
- `vhashtriples()` (*rdflib.tools.graphisomorphism.IsomorphicTestableGraph method*), 220
- `video` (*rdflib.SDO attribute*), 586
- `Video_Intercom` (*rdflib.BRICK attribute*), 395
- `Video_Surveillance_Equipment` (*rdflib.BRICK attribute*), 395
- `videoFormat` (*rdflib.SDO attribute*), 586
- `videoFrameSize` (*rdflib.SDO attribute*), 587
- `VideoGallery` (*rdflib.SDO attribute*), 528
- `VideoGame` (*rdflib.SDO attribute*), 528
- `VideoGameClip` (*rdflib.SDO attribute*), 528
- `VideoGameSeries` (*rdflib.SDO attribute*), 528
- `VideoObject` (*rdflib.SDO attribute*), 528
- `VideoObjectSnapshot` (*rdflib.SDO attribute*), 528
- `videoQuality` (*rdflib.SDO attribute*), 587
- `ViewAction` (*rdflib.SDO attribute*), 528
- `VinylFormat` (*rdflib.SDO attribute*), 528
- `Violation` (*rdflib.SH attribute*), 591
- `virtual` (*rdflib.CSVW attribute*), 403
- `virtualLocation` (*rdflib.ODRL2 attribute*), 460
- `VirtualLocation` (*rdflib.SDO attribute*), 528
- `Virus` (*rdflib.SDO attribute*), 528
- `Visitor_Lobby` (*rdflib.BRICK attribute*), 395

VisualArtsEvent (*rdflib.SDO* attribute), 528  
 VisualArtwork (*rdflib.SDO* attribute), 528  
 VitalSign (*rdflib.SDO* attribute), 528  
 vocabulary (*rdflib.VOID* attribute), 608  
 VOID (class in *rdflib*), 606  
 Volcano (*rdflib.SDO* attribute), 528  
 Voltage\_Imbalance\_Sensor (*rdflib.BRICK* attribute), 395  
 Voltage\_Sensor (*rdflib.BRICK* attribute), 395  
 volume (*rdflib.BRICK* attribute), 400  
 volumeNumber (*rdflib.SDO* attribute), 587  
 VoteAction (*rdflib.SDO* attribute), 528

## W

W3CDTF (*rdflib.DCTERMS* attribute), 412  
 W3CNTriplesParser (class in *rdflib.plugins.parsers.ntriples*), 99  
 WantAction (*rdflib.SDO* attribute), 528  
 Wardrobe (*rdflib.BRICK* attribute), 395  
 Warm\_Cool\_Adjust\_Sensor (*rdflib.BRICK* attribute), 395  
 Warmest\_Zone\_Air\_Temperature\_Sensor (*rdflib.BRICK* attribute), 395  
 warning (*rdflib.SDO* attribute), 587  
 Warning (*rdflib.SH* attribute), 591  
 warranty (*rdflib.SDO* attribute), 587  
 WarrantyPromise (*rdflib.SDO* attribute), 528  
 warrantyPromise (*rdflib.SDO* attribute), 587  
 WarrantyScope (*rdflib.SDO* attribute), 528  
 warrantyScope (*rdflib.SDO* attribute), 587  
 wasActivityOfInfluence (*rdflib.PROV* attribute), 474  
 wasAssociatedWith (*rdflib.PROV* attribute), 474  
 wasAssociateFor (*rdflib.PROV* attribute), 474  
 wasAttributedTo (*rdflib.PROV* attribute), 474  
 wasDerivedFrom (*rdflib.PROV* attribute), 474  
 wasEndedBy (*rdflib.PROV* attribute), 475  
 wasGeneratedBy (*rdflib.PROV* attribute), 475  
 wasInfluencedBy (*rdflib.PROV* attribute), 475  
 wasInformedBy (*rdflib.PROV* attribute), 475  
 wasInvalidatedBy (*rdflib.PROV* attribute), 475  
 wasMemberOf (*rdflib.PROV* attribute), 475  
 wasOriginatedBy (*rdflib.SSN* attribute), 600  
 wasPlanOf (*rdflib.PROV* attribute), 475  
 wasPrimarySourceOf (*rdflib.PROV* attribute), 475  
 wasQuotedFrom (*rdflib.PROV* attribute), 475  
 wasRevisionOf (*rdflib.PROV* attribute), 475  
 wasRoleIn (*rdflib.PROV* attribute), 475  
 wasStartedBy (*rdflib.PROV* attribute), 475  
 Waste\_Storage (*rdflib.BRICK* attribute), 395  
 wasUsedBy (*rdflib.PROV* attribute), 475  
 wasUsedInDerivation (*rdflib.PROV* attribute), 475  
 WatchAction (*rdflib.SDO* attribute), 528  
 Water (*rdflib.BRICK* attribute), 395  
 Water\_Alarm (*rdflib.BRICK* attribute), 395

Water\_Differential\_Pressure\_Setpoint (*rdflib.BRICK* attribute), 395  
 Water\_Differential\_Temperature\_Sensor (*rdflib.BRICK* attribute), 395  
 Water\_Differential\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 395  
 Water\_Distribution (*rdflib.BRICK* attribute), 395  
 Water\_Flow\_Sensor (*rdflib.BRICK* attribute), 396  
 Water\_Flow\_Setpoint (*rdflib.BRICK* attribute), 396  
 Water\_Heater (*rdflib.BRICK* attribute), 396  
 Water\_Level\_Alarm (*rdflib.BRICK* attribute), 396  
 Water\_Level\_Sensor (*rdflib.BRICK* attribute), 396  
 Water\_Loop (*rdflib.BRICK* attribute), 396  
 Water\_Loss\_Alarm (*rdflib.BRICK* attribute), 396  
 Water\_Meter (*rdflib.BRICK* attribute), 396  
 Water\_Pump (*rdflib.BRICK* attribute), 396  
 Water\_System (*rdflib.BRICK* attribute), 396  
 Water\_Tank (*rdflib.BRICK* attribute), 396  
 Water\_Temperature\_Alarm (*rdflib.BRICK* attribute), 396  
 Water\_Temperature\_Sensor (*rdflib.BRICK* attribute), 396  
 Water\_Temperature\_Setpoint (*rdflib.BRICK* attribute), 396  
 Water\_Usage\_Sensor (*rdflib.BRICK* attribute), 396  
 Water\_Valve (*rdflib.BRICK* attribute), 396  
 Waterfall (*rdflib.SDO* attribute), 529  
 watermark (*rdflib.ODRL2* attribute), 460  
 WearableMeasurementBack (*rdflib.SDO* attribute), 529  
 WearableMeasurementChestOrBust (*rdflib.SDO* attribute), 529  
 WearableMeasurementCollar (*rdflib.SDO* attribute), 529  
 WearableMeasurementCup (*rdflib.SDO* attribute), 529  
 WearableMeasurementHeight (*rdflib.SDO* attribute), 529  
 WearableMeasurementHips (*rdflib.SDO* attribute), 529  
 WearableMeasurementInseam (*rdflib.SDO* attribute), 529  
 WearableMeasurementLength (*rdflib.SDO* attribute), 529  
 WearableMeasurementOutsideLeg (*rdflib.SDO* attribute), 529  
 WearableMeasurementSleeve (*rdflib.SDO* attribute), 529  
 WearableMeasurementTypeEnumeration (*rdflib.SDO* attribute), 529  
 WearableMeasurementWaist (*rdflib.SDO* attribute), 529  
 WearableMeasurementWidth (*rdflib.SDO* attribute), 529  
 WearableSizeGroupBig (*rdflib.SDO* attribute), 529  
 WearableSizeGroupBoys (*rdflib.SDO* attribute), 529

- WearableSizeGroupEnumeration (*rdflib.SDO attribute*), 529
- WearableSizeGroupExtraShort (*rdflib.SDO attribute*), 529
- WearableSizeGroupExtraTall (*rdflib.SDO attribute*), 529
- WearableSizeGroupGirls (*rdflib.SDO attribute*), 530
- WearableSizeGroupHusky (*rdflib.SDO attribute*), 530
- WearableSizeGroupInfants (*rdflib.SDO attribute*), 530
- WearableSizeGroupJuniors (*rdflib.SDO attribute*), 530
- WearableSizeGroupMaternity (*rdflib.SDO attribute*), 530
- WearableSizeGroupMens (*rdflib.SDO attribute*), 530
- WearableSizeGroupMisses (*rdflib.SDO attribute*), 530
- WearableSizeGroupPetite (*rdflib.SDO attribute*), 530
- WearableSizeGroupPlus (*rdflib.SDO attribute*), 530
- WearableSizeGroupRegular (*rdflib.SDO attribute*), 530
- WearableSizeGroupShort (*rdflib.SDO attribute*), 530
- WearableSizeGroupTall (*rdflib.SDO attribute*), 530
- WearableSizeGroupWomens (*rdflib.SDO attribute*), 530
- WearableSizeSystemAU (*rdflib.SDO attribute*), 530
- WearableSizeSystemBR (*rdflib.SDO attribute*), 530
- WearableSizeSystemCN (*rdflib.SDO attribute*), 530
- WearableSizeSystemContinental (*rdflib.SDO attribute*), 530
- WearableSizeSystemDE (*rdflib.SDO attribute*), 530
- WearableSizeSystemEN13402 (*rdflib.SDO attribute*), 530
- WearableSizeSystemEnumeration (*rdflib.SDO attribute*), 531
- WearableSizeSystemEurope (*rdflib.SDO attribute*), 531
- WearableSizeSystemFR (*rdflib.SDO attribute*), 531
- WearableSizeSystemGS1 (*rdflib.SDO attribute*), 531
- WearableSizeSystemIT (*rdflib.SDO attribute*), 531
- WearableSizeSystemJP (*rdflib.SDO attribute*), 531
- WearableSizeSystemMX (*rdflib.SDO attribute*), 531
- WearableSizeSystemUK (*rdflib.SDO attribute*), 531
- WearableSizeSystemUS (*rdflib.SDO attribute*), 531
- WearAction (*rdflib.SDO attribute*), 529
- Weather\_Station (*rdflib.BRICK attribute*), 396
- WebAPI (*rdflib.SDO attribute*), 531
- WebApplication (*rdflib.SDO attribute*), 531
- webCheckinTime (*rdflib.SDO attribute*), 587
- WebContent (*rdflib.SDO attribute*), 531
- webFeed (*rdflib.SDO attribute*), 587
- weblog (*rdflib.FOAF attribute*), 423
- WebPage (*rdflib.SDO attribute*), 531
- WebPageElement (*rdflib.SDO attribute*), 531
- WebSite (*rdflib.SDO attribute*), 531
- Wednesday (*rdflib.SDO attribute*), 531
- Wednesday (*rdflib.TIME attribute*), 601
- week (*rdflib.TIME attribute*), 603
- weeks (*rdflib.TIME attribute*), 604
- weight (*rdflib.SDO attribute*), 587
- weightTotal (*rdflib.SDO attribute*), 587
- WesternConventional (*rdflib.SDO attribute*), 531
- wheelbase (*rdflib.SDO attribute*), 587
- where\_pattern (*rdflib.plugins.stores.sparqlstore.SPARQLUpdateStore attribute*), 218
- whiteSpace (*rdflib.XSD attribute*), 612
- Wholesale (*rdflib.SDO attribute*), 531
- WholesaleStore (*rdflib.SDO attribute*), 531
- width (*rdflib.SDO attribute*), 587
- wiki (*rdflib.DOAP attribute*), 415
- WinAction (*rdflib.SDO attribute*), 531
- Wind\_Direction\_Sensor (*rdflib.BRICK attribute*), 396
- Wind\_Speed\_Sensor (*rdflib.BRICK attribute*), 396
- Winery (*rdflib.SDO attribute*), 531
- Wing (*rdflib.BRICK attribute*), 397
- winner (*rdflib.SDO attribute*), 587
- Withdrawn (*rdflib.SDO attribute*), 531
- withRestrictions (*rdflib.OWL attribute*), 466
- wordCount (*rdflib.SDO attribute*), 587
- WorkBasedProgram (*rdflib.SDO attribute*), 531
- WorkersUnion (*rdflib.SDO attribute*), 532
- workExample (*rdflib.SDO attribute*), 587
- workFeatured (*rdflib.SDO attribute*), 587
- workHours (*rdflib.SDO attribute*), 587
- workInfoHomepage (*rdflib.FOAF attribute*), 423
- workload (*rdflib.SDO attribute*), 587
- workLocation (*rdflib.SDO attribute*), 587
- workPerformed (*rdflib.SDO attribute*), 587
- workplaceHomepage (*rdflib.FOAF attribute*), 423
- workPresented (*rdflib.SDO attribute*), 587
- worksFor (*rdflib.SDO attribute*), 587
- Workshop (*rdflib.BRICK attribute*), 397
- workTranslation (*rdflib.SDO attribute*), 587
- worstRating (*rdflib.SDO attribute*), 587
- WPAdBlock (*rdflib.SDO attribute*), 528
- WPFooter (*rdflib.SDO attribute*), 528
- WPHeader (*rdflib.SDO attribute*), 528
- WPSideBar (*rdflib.SDO attribute*), 528
- write (*rdflib.ODRL2 attribute*), 460
- write() (*rdflib.plugins.serializers.turtle.RecursiveSerializer method*), 122
- write() (*rdflib.query.EncodeOnlyUnicode method*), 291
- write\_ask() (*rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter method*), 137
- write\_binding() (*rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter method*), 137
- write\_end\_result() (*rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter method*), 137

[write\\_header\(\)](#) ([rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter](#) (class in [rdflib.plugins.serializers.xmlwriter](#)), 123)  
[write\\_results\\_header\(\)](#) ([rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter](#) (class in [rdflib.plugins.serializers.xmlwriter](#)), 123)  
[write\\_start\\_result\(\)](#) ([rdflib.plugins.sparql.results.xmlresults.SPARQLXMLWriter](#) (class in [rdflib.plugins.serializers.xmlwriter](#)), 123)  
[WriteAction](#) ([rdflib.SDO](#) attribute), 532  
[WritePermission](#) ([rdflib.SDO](#) attribute), 532  
[writeTo](#) ([rdflib.ODRL2](#) attribute), 461

## X

[XMLLiteral](#) ([rdflib.RDF](#) attribute), 478  
[XMLResult](#) (class in [rdflib.plugins.sparql.results.xmlresults](#)), 138  
[XMLResultParser](#) (class in [rdflib.plugins.sparql.results.xmlresults](#)), 138  
[XMLResultSerializer](#) (class in [rdflib.plugins.sparql.results.xmlresults](#)), 138  
[XMLSerializer](#) (class in [rdflib.plugins.serializers.rdfxml](#)), 119  
[XMLWriter](#) (class in [rdflib.plugins.serializers.xmlwriter](#)), 123  
[xone](#) ([rdflib.ODRL2](#) attribute), 461  
[xone](#) ([rdflib.SH](#) attribute), 595  
[XoneConstraintComponent](#) ([rdflib.SH](#) attribute), 591  
[xpath](#) ([rdflib.SDO](#) attribute), 587  
[XPathType](#) ([rdflib.SDO](#) attribute), 532  
[XRay](#) ([rdflib.SDO](#) attribute), 532  
[XSD](#) (class in [rdflib](#)), 609  
[xsdDateTime](#) ([rdflib.TIME](#) attribute), 604

## Y

[yahooChatID](#) ([rdflib.FOAF](#) attribute), 423  
[Year](#) ([rdflib.TIME](#) attribute), 601  
[year](#) ([rdflib.TIME](#) attribute), 604  
[year](#) ([rdflib.XSD](#) attribute), 612  
[yearBuilt](#) ([rdflib.BRICK](#) attribute), 400  
[yearBuilt](#) ([rdflib.SDO](#) attribute), 587  
[yearlyRevenue](#) ([rdflib.SDO](#) attribute), 587  
[yearMonthDuration](#) ([rdflib.XSD](#) attribute), 612  
[years](#) ([rdflib.TIME](#) attribute), 604  
[yearsInOperation](#) ([rdflib.SDO](#) attribute), 587

## Z

[zeroOrMorePath](#) ([rdflib.SH](#) attribute), 595  
[zeroOrOnePath](#) ([rdflib.SH](#) attribute), 595  
[Zone](#) ([rdflib.BRICK](#) attribute), 397  
[Zone\\_Air](#) ([rdflib.BRICK](#) attribute), 397  
[Zone\\_Air\\_Cooling\\_Temperature\\_Setpoint](#) ([rdflib.BRICK](#) attribute), 397  
[Zone\\_Air\\_Dewpoint\\_Sensor](#) ([rdflib.BRICK](#) attribute), 397