

---

# OWLAPY

*Release 0.1.2*

**Ontolearn Team**

**Jun 05, 2024**

## Contents:

<b>1</b>	<b>About owlapy</b>	<b>2</b>
1.1	What is owlapy? . . . . .	2
1.2	What does owlapy have to offer? . . . . .	2
1.3	How to install? . . . . .	3
<b>2</b>	<b>Basic Usage</b>	<b>3</b>
2.1	Atomic Classes . . . . .	3
2.2	Object Property . . . . .	4
2.3	Complex class expressions . . . . .	4
2.4	Convert to SPARQL, DL or Manchester syntax . . . . .	5
<b>3</b>	<b>Ontologies</b>	<b>5</b>
3.1	Loading an Ontology . . . . .	6
3.2	Modifying an Ontology . . . . .	6
3.3	Save an Ontology . . . . .	8
3.4	Worlds . . . . .	8
<b>4</b>	<b>Reasoners</b>	<b>9</b>
4.1	Usage of the Reasoner . . . . .	10
4.2	Class Reasoning . . . . .	10
4.3	Object Properties and Data Properties Reasoning . . . . .	11
4.4	Find Instances . . . . .	11
<b>5</b>	<b>Reasoning Details</b>	<b>12</b>
5.1	Sync Reasoner . . . . .	12
5.2	Isolated World . . . . .	12
5.3	Capabilities . . . . .	13
5.4	Concrete Example . . . . .	14
<b>6</b>	<b>owlapy</b>	<b>17</b>
6.1	Subpackages . . . . .	17
6.2	Submodules . . . . .	62
6.3	Attributes . . . . .	334
6.4	Functions . . . . .	334
6.5	Package Contents . . . . .	334
	<b>Python Module Index</b>	<b>335</b>

---

OWLAPY<sup>1</sup>: Representation of OWL objects in python.

## 1 About owlapy

**Version:** owlapy 1.1.0

**GitHub repository:** <https://github.com/dice-group/owlapy>

**Publisher and maintainer:** DICE<sup>2</sup> - data science research group of Paderborn University<sup>3</sup>.

**Contact:** [onto-learn@lists.uni-paderborn.de](mailto:onto-learn@lists.uni-paderborn.de)

**License:** MIT License

---

### 1.1 What is owlapy?

Owlapy is an open-source software library in python that is used to represent entities in OWL 2 Web Ontology Language.

We identified the gap of having a library that will serve as a base structure for representing OWL entities and for manipulating OWL Ontologies in python, and like that, owlapy was created. Owlapy is loosely based on its java-counterpart, *owlapi*. Owlapy is currently utilized by powerful libraries such as [Ontolearn](#)<sup>4</sup> and [OntoSample](#)<sup>5</sup>.

Owlapy is the perfect choice for machine learning projects that are built in python and focus on knowledge graphs and class expression learnings.

---

### 1.2 What does owlapy have to offer?

- Create, manipulate and save Ontologies.
- Retrieving information from the signature of the ontology.
- Reasoning over ontology.
- Represent every notation in [OWL 2 Structural Specification and Functional-Style Syntax](#)<sup>6</sup> including:
  - Entities, Literals, and Anonymous Individuals
  - Property Expressions
  - Data Ranges
  - Class Expressions
  - Axioms

---

<sup>1</sup> <https://github.com/dice-group/owlapy>

<sup>2</sup> <https://dice-research.org/>

<sup>3</sup> <https://www.uni-paderborn.de/en/university>

<sup>4</sup> <https://github.com/dice-group/Ontolearn>

<sup>5</sup> <https://github.com/alkidbaci/OntoSample>

<sup>6</sup> <https://www.w3.org/TR/owl2-syntax/>

- Annotations
- Construct complex class expressions.
- Provide interfaces for OWL Ontology, Ontology manager and Reasoner.
- Convert owl expression to SPARQL queries.
- Render owl expression to Description Logics or Manchester syntax.
- Parse Description Logics or Manchester expression to owl expression.

## 1.3 How to install?

Installation from source:

```
git clone https://github.com/dice-group/owlapy
conda create -n temp_owlapy python=3.10.13 --no-default-packages && conda activate_
temp_owlapy && pip3 install -e .
```

or using PyPI:

```
pip3 install owlapy
```

## 2 Basic Usage

The main usage for owlapy is to use it for class expression construction. Class expression learning algorithms require such basic structure to work upon. Let's walk through an example of constructing some class expressions.

In this example we will be using the *family* ontology, a simple ontology with namespace: `http://example.com/family#`. Here is a hierarchical diagram that shows the classes and their relationship:

```

    Thing
    |
  person
  /  |
male female

```

It contains only one object property which is `hasChild` and in total there are six persons (individuals), of which four are males and two are females.

### 2.1 Atomic Classes

To represent the classes `male`, `female`, and `person` we can simply use the class `OWLClass`<sup>7</sup>:

```
from owlapy.class_expression import OWLClass
from owlapy.iri import IRI

namespace = "http://example.com/family#"

male = OWLClass(IRI(namespace, "male"))
female = OWLClass(IRI(namespace, "female"))
```

(continues on next page)

<sup>7</sup> [https://dice-group.github.io/owlapy/autoapi/owlapy/class\\_expression/owl\\_class/index.html#owlapy.class\\_expression.owl\\_class.OWLClass](https://dice-group.github.io/owlapy/autoapi/owlapy/class_expression/owl_class/index.html#owlapy.class_expression.owl_class.OWLClass)

(continued from previous page)

```
person = OWLClass(URI(namespace, "person"))
```

Notice that we created an IRI object for every class. [IRI](#)<sup>8</sup> is used to represent an *IRI*. Every named entity requires an IRI, whereas Anonymous entities does not. However, in owlapy you can create an *OWLClass* by passing the *IRI* directly as a string, like so:

```
male = OWLClass("http://example.com/family#male")
```

## 2.2 Object Property

To represent the object property `hasChild` we can use the class `OWLObjectProperty`<sup>9</sup>:

```
from owlapy.owl_property import OWLObjectProperty

hasChild = OWLObjectProperty("http://example.com/family#hasChild")
```

**Tip:** In owlapy the naming of the classes is made in accordance with the notations from OWL 2 specification but with the word “OWL” in the beginning. Example: “*OWLObjectProperty*” represents the notation “*ObjectProperty*”.

## 2.3 Complex class expressions

Now that we have these atomic entities, we can construct more complex class expressions. Let’s say we want to represent all individuals which are `male` and have at least 1 child.

We already have the concept of `male`. We need to find the appropriate class for the second part: “*have at least 1 child*”. In OWL 2 specification that would be [ObjectMinCardinality](#)<sup>10</sup>. In owlapy, as we said, we simply add the word “OWL” upfront to find the correct class:

```
from owlapy.class_expression import OWLObjectMinCardinality

has_at_least_one_child = OWLObjectMinCardinality(
    cardinality = 1,
    property = hasChild,
    filler = person
)
```

As you can see, to create an object of class [OWLObjectMinCardinality](#)<sup>11</sup> is as easy as that. You specify the cardinality which in this case is 1, the object property where we apply this cardinality restriction and the filler class in case you want to restrict the domain of the class expression. In this case we used `person`.

Now let’s merge both class expressions together using [OWLObjectIntersectionOf](#)<sup>12</sup>:

```
from owlapy.class_expression import OWLObjectIntersectionOf

ce = OWLObjectIntersectionOf([male, has_at_least_one_child])
```

<sup>8</sup> <https://dice-group.github.io/owlapy/autoapi/owlapy/iri/index.html#owlapy.iri>

<sup>9</sup> [https://dice-group.github.io/owlapy/autoapi/owlapy/owl\\_property/index.html#owlapy.owl\\_property.OWLObjectProperty](https://dice-group.github.io/owlapy/autoapi/owlapy/owl_property/index.html#owlapy.owl_property.OWLObjectProperty)

<sup>10</sup> [https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality)

<sup>11</sup> [https://dice-group.github.io/owlapy/autoapi/owlapy/class\\_expression/restriction/index.html#owlapy.class\\_expression.restriction.OWLObjectMinCardinality](https://dice-group.github.io/owlapy/autoapi/owlapy/class_expression/restriction/index.html#owlapy.class_expression.restriction.OWLObjectMinCardinality)

<sup>12</sup> [https://dice-group.github.io/owlapy/autoapi/owlapy/class\\_expression/nary\\_boolean\\_expression/index.html#owlapy.class\\_expression.nary\\_boolean\\_expression.OWLObjectIntersectionOf](https://dice-group.github.io/owlapy/autoapi/owlapy/class_expression/nary_boolean_expression/index.html#owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf)

## 2.4 Convert to SPARQL, DL or Manchester syntax

Owlapy is not just a library to represent OWL entities, you can also use it to convert owl expressions into other formats:

```
from owlapy import owl_expression_to_sparql, owl_expression_to_dl, owl_expression_to_
    ↳manchester

print(owl_expression_to_dl(ce))
# Result: male ♂ (≥ 1 hasChild.person)

print(owl_expression_to_sparql(ce))
# Result: SELECT DISTINCT ?x WHERE { ?x a <http://example.com/family#male> . { SELECT_
    ↳?x WHERE { ?x <http://example.com/family#hasChild> ?s_1 . ?s_1 a <http://example.
    ↳com/family#person> . } GROUP BY ?x HAVING ( COUNT ( ?s_1 ) >= 1 ) } }

print(owl_expression_to_manchester(ce))
# Result: male and (hasChild min 1 person)
```

To parse a DL or Manchester expression to owl expression you can use the following convenient methods:

```
from owlapy import dl_to_owl_expression, manchester_to_owl_expression

print(dl_to_owl_expression("∃ hasChild.male", namespace))
# Result: OWLObjectSomeValuesFrom(property=OWLObjectProperty(IRI('http://example.com/
    ↳family#', 'hasChild')),filler=OWLObject(IRI('http://example.com/family#', 'male')))

print(manchester_to_owl_expression("female and (hasChild max 2 person)", namespace))
# Result: OWLObjectIntersectionOf((OWLObject(IRI('http://example.com/family#', 'female
    ↳')), OWLObjectMaxCardinality(property=OWLObjectProperty(IRI('http://example.com/
    ↳family#', 'hasChild')),2,filler=OWLObject(IRI('http://example.com/family#', 'person
    ↳')))))
```

In these examples we showed a fraction of **owlapy**. You can explore the *api documentation* to learn more about all classes in owlapy and check more examples in the `examples`<sup>13</sup> directory.

## 3 Ontologies

To get started with Structured Machine Learning, the first thing required is an *Ontology*<sup>14</sup> with *Named Individuals*<sup>15</sup>. In this guide we show the basics of working with ontologies in Owlapy. We will use the *father* ontology for the following examples.

<sup>13</sup> <https://github.com/dice-group/owlapy/tree/develop/examples>

<sup>14</sup> <https://www.w3.org/TR/owl2-overview/>

<sup>15</sup> [https://www.w3.org/TR/owl-syntax/#Named\\_Individuals](https://www.w3.org/TR/owl-syntax/#Named_Individuals)

### 3.1 Loading an Ontology

To load an ontology as well as to manage it, you will need an *OWL*OntologyManager. An ontology can be loaded using the following Python code:

```
from owlapy.iri import IRI
from owlapy.owl_ontology_manager import OntologyManager

manager = OntologyManager()
onto = manager.load_ontology(IRI.create("file://KGs/Family/father.owl"))
```

First, we import the *IRI* class and a suitable *OWL*OntologyManager. To load a file from our computer, we have to reference it with an *IRI*. Secondly, we need the Ontology Manager. Owlapy contains one such manager: The *OntologyManager*.

Now, we can already inspect the contents of the ontology. For example, to list all individuals:

```
for ind in onto.individuals_in_signature():
    print(ind)
```

You can get the object properties in the signature:

```
onto.object_properties_in_signature()
```

For more methods, see the abstract class *OWL*Ontology or the concrete implementation *Ontology*.

### 3.2 Modifying an Ontology

Axioms in ontology serve as the basis for defining the vocabulary of a domain and for making statements about the relationships between individuals and concepts in that domain. They provide a formal and precise way to represent knowledge and allow for automated reasoning and inference. Axioms can be **added**, **modified**, or **removed** from an ontology, allowing the ontology to evolve and adapt as new knowledge is gained.

In owlapy we also have different axioms represented by different classes. You can check all the axioms classes *here*. Some frequently used axioms are:

- *OWLDeclarationAxiom*
- *OWLObjectPropertyAssertionAxiom*
- *OWLDataPropertyAssertionAxiom*
- *OWLClassAssertionAxiom*
- *OWLSubClassOfAxiom*
- *OWLEquivalentClassesAxiom*

## Add a new Class

Let's suppose you want to add a new class in our example ontology `KGs/Family/father.owl`. It can be done as follows:

```
from owlapy.class_expression import OWLClass
from owlapy.owl_axiom import OWLDeclarationAxiom

iri = IRI('http://example.com/father#', 'child')
child_class = OWLClass(iri)
child_class_declaration_axiom = OWLDeclarationAxiom(child_class)

manager.add_axiom(onto, child_class_declaration_axiom)
```

In this example, we added the class 'child' to the *father.owl* ontology. Firstly we create an instance of *OWLClass* to represent the concept of 'child' by using an *IRI*. On the other side, an instance of *IRI* is created by passing two arguments which are the namespace of the ontology and the remainder 'child'. To declare this new class we need an axiom of type *OWLDeclarationAxiom*. We simply pass the *child\_class* to create an instance of this axiom. The final step is to add this axiom to the ontology using the *OWLOntologyManager*. We use the *add\_axiom* method of the manager to add into the ontology onto the axiom *child\_class\_declaration\_axiom*.

## Add a new Object Property / Data Property

The idea is the same as adding a new class. Instead of *OWLClass*, for object properties, you can use the class *OWLObjectProperty* and for data properties you can use the class *OWLDataProperty*.

```
from owlapy.owl_property import OWLObjectProperty, OWLDataProperty

# adding the object property 'hasParent'
hasParent_op = OWLObjectProperty(IRI('http://example.com/father#', 'hasParent'))
hasParent_op_declaration_axiom = OWLDeclarationAxiom(hasParent_op)
manager.add_axiom(onto, hasParent_op_declaration_axiom)

# adding the data property 'hasAge'
hasAge_dp = OWLDataProperty(IRI('http://example.com/father#', 'hasAge'))
hasAge_dp_declaration_axiom = OWLDeclarationAxiom(hasAge_dp)
manager.add_axiom(onto, hasAge_dp_declaration_axiom)
```

See the *owlapy* for more OWL entities that you can add as a declaration axiom.

## Add an Assertion Axiom

To assign a class to a specific individual use the following code:

```
from owlapy.owl_axiom import OWLClassAssertionAxiom

individuals = list(onto.individuals_in_signature())
heinz = individuals[1] # get the 2nd individual in the list which is 'heinz'

class_assertion_axiom = OWLClassAssertionAxiom(heinz, child_class)

manager.add_axiom(onto, class_assertion_axiom)
```

We have used the previous method *individuals\_in\_signature()* to get all the individuals and converted them to a list, so we can access them by using indexes. In this example, we want to assert a class axiom for the individual *heinz*.

We have used the class `OWLClassAssertionAxiom` where the first argument is the ‘individual’ `heinz` and the second argument is the ‘class\_expression’. As the class expression, we used the previously defined class `child_Class`. Finally, add the axiom by using `add_axiom` method of the *OWLOntologyManager*.

Let’s show one more example using a `OWLObjectPropertyAssertionAxiom` to assign the age of 17 to `heinz`.

```
from owlapy.owl_literal import OWLLiteral
from owlapy.owl_axiom import OWLObjectPropertyAssertionAxiom

literal_17 = OWLLiteral(17)
dp_assertion_axiom = OWLObjectPropertyAssertionAxiom(heinz, hasAge_dp, literal_17)

manager.add_axiom(onto, dp_assertion_axiom)
```

*OWLLiteral* is a class that represents the literal values in Owlapy. We have stored the integer literal value of ‘17’ in the variable `literal_17`. Then we construct the `OWLObjectPropertyAssertionAxiom` by passing as the first argument, the individual `heinz`, as the second argument the data property `hasAge_dp`, and the third argument the literal value `literal_17`. Finally, add it to the ontology by using `add_axiom` method.

Check the *owlapy* to see all the OWL assertion axioms that you can use.

## Remove an Axiom

To remove an axiom you can use the `remove_axiom` method of the ontology manager as follows:

```
manager.remove_axiom(onto, dp_assertion_axiom)
```

The first argument is the ontology you want to remove the axiom from and the second argument is the axiom you want to remove.

## 3.3 Save an Ontology

If you modified an ontology, you may want to save it as a new file. To do this you can use the `save_ontology` method of the *OWLOntologyManager*. It requires two arguments, the first is the ontology you want to save and The second is the IRI of the new ontology.

```
manager.save_ontology(onto, IRI.create('file://' + 'test' + '.owl'))
```

The above line of code will save the ontology `onto` in the file `test.owl` which will be created in the same directory as the file you are running this code.

## 3.4 Worlds

Owlready2 stores every triple in a ‘World’ object, and it can handle several Worlds in parallel. Owlready2 uses an optimized quadstore to store the world. Each world object is stored in a separate quadstore and by default the quadstore is stored in memory, but it can also be stored in an SQLite3 file. The method `save_world()` of the ontology manager does the latter. When an *OWLOntologyManager* object is created, a new world is also created as an attribute of the manager. By calling the method `load_ontology(iri)` the ontology is loaded to this world.

It possible to create several isolated “worlds”, sometimes called “universe of speech”. This makes it possible in particular to load the same ontology several times, independently, that is to say, without the modifications made on one copy affecting the other copy. Sometimes the need to *isolate an ontology* arise. What that means is that you can have multiple reference of the same ontology in different worlds.



It is important that an ontology is associated with a reasoner which is used to inferring knowledge from the ontology, i.e. to perform ontology reasoning. In the next guide we will see how to use a reasoner in Owlapy.

## 4 Reasoners

To validate facts about statements in the ontology, the help of a reasoner component is required.

For this guide we will also consider the ‘father’ ontology that we slightly described [here](#):

```
from owlapy.owl_ontology_manager import OntologyManager

manager = OntologyManager()
onto = manager.load_ontology(IRI.create("KGs/Family/father.owl"))
```

In our Owlapy library, we provide several **reasoners** to choose from. Currently, there are the following reasoners available:

- ***OntologyReasoner***

Or differently Structural Reasoner, is the base reasoner in Owlapy. The functionalities of this reasoner are limited. It does not provide full reasoning in *ALCH*. Furthermore, it has no support for instances of complex class expressions, which is covered by the other reasoners (SyncReasoner and FIC). We recommend to use the other reasoners for any heavy reasoning tasks.

**Initialization:**

```
from owlapy.owl_reasoner import OntologyReasoner

structural_reasoner = OntologyReasoner(onto)
```

The structural reasoner requires an ontology (*OWL*Ontology). The second argument is `isolate` argument which isolates the world (therefore the ontology) where the reasoner is performing the reasoning. More on that on [Reasoning Details](#).

- ***SyncReasoner***

Can perform full reasoning in *ALCH* due to the use of HermiT/Pellet and provides support for complex class expression instances (when using the method `instances`). SyncReasoner is more useful when your main goal is reasoning over the ontology.

**Initialization:**

```
from owlapy.owl_reasoner import SyncReasoner, BaseReasoner

sync_reasoner = SyncReasoner(onto, BaseReasoner.HERMIT, infer_property_values = True)
```

Sync Reasoner requires an ontology and a base reasoner of type *BaseReasoner* which is just an enumeration with two possible values: `BaseReasoner.HERMIT` and `BaseReasoner.PELLET`. You can set the `infer_property_values` argument to `True` if you want the reasoner to infer property values. `infer_data_property_values` is an additional argument when the base reasoner is set to `BaseReasoner.PELLET`. The argument `isolated` is inherited from the base class

- ***FastInstanceCheckerReasoner (FIC)***

FIC also provides support for complex class expression but the rest of the methods are the same as in the base reasoner. It has a cache storing system that allows for faster execution of some reasoning functionalities. Due to this feature, FIC is more appropriate to be used in concept learning.

**Initialization:**

```

from owlapy.owl_reasoner import FastInstanceCheckerReasoner

fic_reasoner = FastInstanceCheckerReasoner(onto, structural_reasoner, property_
    ↪ cache = True,
                                                    negation_default = True, sub_
    ↪ properties = False)

```

Besides the ontology, FIC requires a base reasoner to delegate any reasoning tasks not covered by it. This base reasoner can be any other reasoner in Owlapy. `property_cache` specifies whether to cache property values. This requires more memory, but it speeds up the reasoning processes. If `negation_default` argument is set to `True` the missing facts in the ontology means false. The argument `sub_properties` is another boolean argument to specify whether you want to take sub properties in consideration for `instances()` method.

## 4.1 Usage of the Reasoner

All the reasoners available in the Owlapy library inherit from the class: *OWLReasonerEx*. This class provides some extra convenient methods compared to its base class *OWLReasoner*, which is an abstract class. Further on, in this guide, we use *SyncReasoner*. to show the capabilities of a reasoner in Owlapy.

To give examples we consider the *father* dataset. If you are not already familiar with this small dataset, you can find an overview of it [here](#).

## 4.2 Class Reasoning

Using an *OWLOntology* you can list all the classes in the signature, but a reasoner can give you more than that. You can get the subclasses, superclasses or the equivalent classes of a class in the ontology:

```

from owlapy.class_expression import OWLClass
from owlapy.iri import IRI

namespace = "http://example.com/father#"
male = OWLClass(IRI(namespace, "male"))

male_super_classes = sync_reasoner.super_classes(male)
male_sub_classes = sync_reasoner.sub_classes(male)
male_equivalent_classes = sync_reasoner.equivalent_classes(male)

```

We define the *male* class by creating an *OWLClass* object. The methods `super_classes` and `sub_classes` have 2 more boolean arguments: `direct` and `only_named`. If `direct=True` then only the direct classes in the hierarchy will be returned, else it will return every class in the hierarchy depending on the method(`sub_classes` or `super_classes`). By default, its value is *False*. The next argument `only_named` specifies whether you want to show only named classes or complex classes as well. By default, its value is *True* which means that it will return only the named classes.

**NOTE:** The extra arguments `direct` and `only_named` are also used in other methods that reason upon the class, object property, or data property hierarchy.

You can get all the types of a certain individual using `types` method:

```

anna = list(onto.individuals_in_signature()).pop()

anna_types = sync_reasoner.types(anna)

```

We retrieve *anna* as the first individual on the list of individuals of the 'Father' ontology. The `type` method only returns named classes.

## 4.3 Object Properties and Data Properties Reasoning

Owlapy reasoners offers some convenient methods for working with object properties and data properties. Below we show some of them, but you can always check all the methods in the *SyncReasoner* class documentation.

You can get all the object properties that an individual has by using the following method:

```
anna = individuals[0]
object_properties = sync_reasoner.ind_object_properties(anna)
```

In this example, `object_properties` contains all the object properties that *anna* has, which in our case would only be *hasChild*. Now we can get the individuals of this object property for *anna*.

```
for op in object_properties:
    object_properties_values = sync_reasoner.object_property_values(anna, op)
    for individual in object_properties_values:
        print(individual)
```

In this example we iterated over the `object_properties`, assuming that there are more than 1, and we use the reasoner to get the values for each object property `op` of the individual *anna*. The values are individuals which we store in the variable `object_properties_values` and are printed in the end. The method `object_property_values` requires as the first argument, an *OWLNamedIndividual* that is the subject of the object property values and the second argument an *OWLObjectProperty* whose values are to be retrieved for the specified individual.

**NOTE:** You can as well get all the data properties of an individual in the same way by using `ind_data_properties` instead of `ind_object_properties` and `data_property_values` instead of `object_property_values`. Keep in mind that `data_property_values` returns literal values (type of *OWLLiteral*).

In the same way as with classes, you can also get the sub object properties or equivalent object properties.

```
from owlapy.owl_property import OWLObjectProperty

hasChild = OWLObjectProperty(IRI(namespace, "hasChild"))

equivalent_to_hasChild = sync_reasoner.equivalent_object_properties(hasChild)
hasChild_sub_properties = sync_reasoner.sub_object_properties(hasChild)
```

In case you want to get the domains and ranges of an object property use the following:

```
hasChild_domains = sync_reasoner.object_property_domains(hasChild)
hasChild_ranges = sync_reasoner.object_property_ranges(hasChild)
```

**NOTE:** Again, you can do the same for data properties but instead of the word ‘object’ in the method name you should use ‘data’.

## 4.4 Find Instances

The method `instances` is a very convenient method. It takes only 1 argument that is basically a class expression and returns all the individuals belonging to that class expression. In Owlapy we have implemented a Python class for each type of class expression. The argument is of type *OWLClassExpression*.

Let us now show a simple example by finding the instances of the class *male* and printing them:

```
male_individuals = sync_reasoner.instances(male)
for ind in male_individuals:
    print(ind)
```

---

In this guide we covered the main functionalities of the reasoners in Owlapy. More details are provided in the next guide.

## 5 Reasoning Details

In the previous guide we explained how to *use reasoners* in Owlapy. Here we cover a detailed explanation of the Owlapy reasoners, particularly *SyncReasoner*. Before we continue to talk about its *capabilities* we have to explain briefly the term *sync\_reasoner*.

### 5.1 Sync Reasoner

*sync\_reasoner* is a definition used in owlready2 to run *HermiT*<sup>16</sup> or *Pellet*<sup>17</sup> and automatically apply the facts deduced to the quadstore. In simple terms, by running HermiT or Pellet, one can infer more knowledge from the ontology (the specification are not mentioned here). We make use of this functionality in Owlapy, and it is represented by *SyncReasoner*. We explained the concept of “Worlds” in *Working with Ontologies*. Having that in mind you need to know that *sync\_reasoner* is applied to the World object. After this particular reasoner is instantiated, because the facts are applied to the quadstore, changes made in the ontology by using the ontology manager will not be reflected to the ontology. The reasoner will use the state of the ontology at the moment it is instantiated.

There are 2 boolean parameters for *sync\_reasoner* that you can specify when creating an instance of *SyncReasoner*. The first one *infer\_property\_values* tells HermiT or Pellet whether to infer (or not) property values. The same idea but for data properties is specified by the parameter *infer\_data\_property\_values* which is only relevant to Pellet.

Note: HermiT and Pellet are Java programs, so you will need to install a Java virtual machine to use them. If you don't have Java, you may install it from [www.java.com](http://www.java.com) (for Windows and macOS) or from the packages of your Linux distribution (the packages are often named “jre” or “jdk” for Java Runtime Environment and Java Development Kit).

### 5.2 Isolated World

In *Working with Ontologies* we mentioned that we can have multiple reference of in different worlds, which we can use to isolate an ontology to a specific World. For simplicity the terms “isolated world” and “isolated ontology” can be used interchangeably in this guide. The isolation comes in handy when we use multiple reasoners in the same script. If we create an instance of *SyncReasoner* it will apply *sync\_reasoner* in the world object of the ontology and this will affect also the other reasoner/s which is/are using the same world. To overcome this issue you can set the argument *isolate=True* when initializing a reasoner. *FastInstanceCheckerReasoner* (FIC) does not have this argument because it uses a base reasoner to delegate most of its methods. Therefore, if the base reasoner has *isolate=True* then FIC will also operate in the isolated world of it's base reasoner.

---

<sup>16</sup> <http://www.hermit-reasoner.com/>

<sup>17</sup> <https://github.com/stardog-union/pellet>

## Modifying an isolated ontology

When a reasoner is operating in an isolated ontology, every axiom added to the original ontology before or after the initialization, will not be reflected to the isolated ontology. To update the isolated ontology and add or remove any axiom, you can use `update_isolated_ontology(axioms_to_add, axioms_to_remove)`. This method accepts a list of axioms for every argument (i.e. the axioms that you want to add and the axioms that you want to remove).

## 5.3 Capabilities

*SyncReasoner* provides full reasoning in *ALCH*. We have adapted and build upon [owlready2](https://owlready2.readthedocs.io/en/latest/)<sup>18</sup> reasoner to provide our own implementation in python. Below we give more details about each functionality of our reasoner:

- **Sub and Super Classes**

You can retrieve sub (super) classes of a given class expression. Depending on your preferences you can retrieve the whole chain of sub (super) classes or only the direct sub (super) classes (`direct` argument). It is also possible to get anonymous classes in addition to named classes (`only_named` argument). Class equivalence entails subsumption of classes to each other.

- **Equivalent Classes**

You are able to get the equivalent classes of a given class expression. It can be decided whether only named classes should be returned or anonymous classes as well. If two classes are subclasses of each other they are considered equivalent.

- **Disjoint Classes**

Every class that is explicitly defined as disjoint with another class will be returned. In addition, every subclass and equivalent class of the disjoint classes will be returned. If a target class does not have explicitly-defined disjoint classes the search is transferred to the superclasses of that target class.

- **Equivalent Properties**

You are able to get equivalent properties of a given object or data property. If two properties are sub-properties of each other, they are considered equivalent.

- **Sub and Super Properties**

Our reasoner has support also for sub and super properties of a given property. You can set the `direct` argument like in sub (super) classes. Properties equivalence entails subsumption of properties to each other.

---

<sup>18</sup> <https://owlready2.readthedocs.io/en/latest/>

- **Disjoint Properties**

Similarly to disjoint classes, you can get the disjoint properties of a property. Same rules apply.

- **Property values**

Given an individual(instance) and an object property you can get all the object values. Similarly, given an individual and a data property you can get all the literal values. You can set whether you want only the direct values or all of them.

- **Property domain and range**

Easily retrieval available for domain and range for object properties and domain for data properties.

- **Instances**

This functionality enables you to get instances for a given named(atomic) class or complex class expression. For the moment direct instances of complex class expressions is not possible.

- **Types**

This functionality enables you to get the types of a given instance. It returns only named(atomic) classes. You can set the `direct` attribute.

- **Same and Different Individuals**

Given an individual you can get the individuals that are explicitly defined as same or different to that individual.

## 5.4 Concrete Example

You can find the associated [code](#)<sup>19</sup> for the following examples inside `examples/example_reasoner` (note that the naming of the classes/relations/individuals may change from the table below). We constructed an ontology for testing purposes. On the table we show for each **method** of the reasoner *SyncReasoner* the results depending on a given **TBox** and **ABox**. The level of complexity of the TBox-es is low compared to real world scenarios, but it's just to show the capabilities of the reasoner.

**Note:** not every method of the reasoner is used in this example. You can check all the methods at the [API documentation](#).

Method	TBox	ABox	Returns(T = Thing)
<code>Equivalent_classes(A)</code>	$A \equiv B$	-	[B]
<code>Equivalent_classes(B)</code>	$A \equiv B$	-	[A]
<code>Instances(A)</code>	$A \equiv B$	A(a),B(b)	[a,b]
<code>Instances(B)</code>	$A \equiv B$	A(a),B(b)	[a,b]
<code>Types(a)</code>	$A \equiv B$	A(a),B(b)	[T, A,B]
<code>Types(b)</code>	$A \equiv B$	A(a),B(b)	[T, A,B]

continues on next page

<sup>19</sup> [https://github.com/dice-group/owlapy/blob/develop/examples/ontology\\_reasoning.py](https://github.com/dice-group/owlapy/blob/develop/examples/ontology_reasoning.py)

Table 1 – continued from previous page

Method	TBox	ABox	Returns(T = Thing)
Sub_classes(A)	$A \equiv B$	-	[B]
Sub_classes(B)	$A \equiv B$	-	[A]
Super_classes(A)	$A \equiv B$	-	[B,T]
Super_classes(B)	$A \equiv B$	-	[A,T]
Equivalent_object_properties(r1)	$r1 \equiv r2$	-	[r2]
Equivalent_object_properties(r2)	$r1 \equiv r2$	-	[r1]
sub_object_properties(r1)	$r1 \equiv r2$	-	[r2]
sub_object_properties(r2)	$r1 \equiv r2$	-	[r1]
object_property_values(a, r1, direct=False)	$r1 \equiv r2$	$r1(a,b) \ r2(a,c)$	[c]
object_property_values(a, r2, direct=False)	$r1 \equiv r2$	$r1(a,b) \ r2(a,c)$	[c]
Sub_classes(B)	$A \sqsubseteq B$	-	[A]
Super_classes(A)	$A \sqsubseteq B$	-	[T, B]
Types(a)	$A \sqsubseteq B$	$A(a), B(b)$	[A,B,T]
Types(b)	$A \sqsubseteq B$	$A(a), B(b)$	[B,T]
Instances(A)	$A \sqsubseteq B$	$A(a), B(b)$	[a]
Instances(B)	$A \sqsubseteq B$	$A(a), B(b)$	[a,b]
sub_object_properties(r1)	$r2 \sqsubseteq r1$	-	[r2]
object_property_values(a, r2)	$r2 \sqsubseteq r1$	$r2(a,b)$	[b]
object_property_values(a, r1, direct=False)	$r2 \sqsubseteq r1$	$r2(a,b)$	[b]
Sub_classes(r1.T)	$r2 \sqsubseteq r1$	-	[r2.T]
Super_classes(D, only_named=False)	$D \sqsubseteq \exists r.E$	-	[T, $\exists r.E$ ]
Sub_classes( $\exists r.E$ )	$D \sqsubseteq \exists r.E$	-	[D]
Instances(D)	$D \sqsubseteq \exists r.E$	$D(d) \ r(i,e) \ E(e)$	[d]
Instances( $\exists r.E$ )	$D \sqsubseteq \exists r.E$	$D(d) \ r(i,e) \ E(e)$	[i, d]
types(d)	$D \sqsubseteq \exists r.E$	$D(d) \ r(i,e) \ E(e)$	[D,T]
types(i)	$D \sqsubseteq \exists r.E$	$D(d) \ r(i,e) \ E(e)$	[T]
object_property_values(i, r)	$D \sqsubseteq \exists r.E$	$r(i,e) \ E(e)$	[e]
Sub_classes(D, only_named=False)	$\exists r.E \sqsubseteq D$	-	[ $\exists r.E$ ]
Super_classes( $\exists r.E$ )	$\exists r.E \sqsubseteq D$	-	[D, T]
Instances(D)	$\exists r.E \sqsubseteq D$	$D(d) \ r(i,e) \ E(e)$	[i, d]
Instances( $\exists r.E$ )	$\exists r.E \sqsubseteq D$	$D(d) \ r(i,e) \ E(e)$	[i]
types(d)	$\exists r.E \sqsubseteq D$	$D(d) \ r(i,e) \ E(e)$	[D, T]
types(i)	$\exists r.E \sqsubseteq D$	$D(d) \ r(i,e) \ E(e)$	[D, T]
object_property_values(i, r)	$\exists r.E \sqsubseteq D$	$r(i,e) \ E(e)$	[e]
Sub_classes(A)	$A \sqsubseteq B, B \sqsubseteq A$	-	[A,B]
Sub_classes(B)	$A \sqsubseteq B, B \sqsubseteq A$	-	[A,B]
Super_classes(A)	$A \sqsubseteq B, B \sqsubseteq A$	-	[T, B]
Super_classes(B)	$A \sqsubseteq B, B \sqsubseteq A$	-	[T, A]
Types(a)	$A \sqsubseteq B, B \sqsubseteq A$	$A(a), B(b)$	[A,B,T]
Types(b)	$A \sqsubseteq B, B \sqsubseteq A$	$A(a), B(b)$	[A,B,T]
Instances(A)	$A \sqsubseteq B, B \sqsubseteq A$	$A(a), B(b)$	[a,b]
Instances(B)	$A \sqsubseteq B, B \sqsubseteq A$	$A(a), B(b)$	[a,b]
Equivalent_classes(A, only_named=False)	$A \sqsubseteq B, B \sqsubseteq A$	-	[B]
Equivalent_classes(B, only_named=False)	$A \sqsubseteq B, B \sqsubseteq A$	-	[A]
sub_object_properties(r1)	$r2 \sqsubseteq r1, r1 \sqsubseteq r2$	-	[r2,r1]
sub_object_properties(r2)	$r2 \sqsubseteq r1, r1 \sqsubseteq r2$	-	[r1,r2]
Equivalent_object_properties(r1)	$r2 \sqsubseteq r1, r1 \sqsubseteq r2$	-	[r2]
Equivalent_object_properties(r2)	$r2 \sqsubseteq r1, r1 \sqsubseteq r2$	-	[r1]
object_property_values(a, r1, direct=False)	$r2 \sqsubseteq r1, r1 \sqsubseteq r2$	$r1(a,b) \ r2(a,c)$	[b,c]
object_property_values(a, r2, direct=False)	$r2 \sqsubseteq r1, r1 \sqsubseteq r2$	$r1(a,b) \ r2(a,c)$	[b,c]

continues on next page

Table 1 – continued from previous page

Method	TBox	ABox	Returns(T = Thing)
Sub_classes(J $\sqcap$ K)	$I \sqsubseteq J \sqcap K$	-	[I]
Super_classes(I, only_named=False)	$I \sqsubseteq J \sqcap K$	-	[J $\sqcap$ K, J, K, T]
Instances(J $\sqcap$ K)	$I \sqsubseteq J \sqcap K$	I(c)	[c]
types(c)	$I \sqsubseteq J \sqcap K$	I(c)	[J, K, I, T]
Super_classes(J $\sqcap$ K)	$J \sqcap K \sqsubseteq I$	-	[I, T]
Sub_classes(I, only_named=False)	$J \sqcap K \sqsubseteq I$	-	[J $\sqcap$ K]
Instances(I)	$J \sqcap K \sqsubseteq I$	J(s),K(s)	[s]
Instances(J $\sqcap$ K)	$J \sqcap K \sqsubseteq I$	J(s),K(s)	[s]
types(s)	$J \sqcap K \sqsubseteq I$	J(s),K(s)	[J, K, I, T]
Sub_classes( $\exists r.E \sqcap B$ )	$D \sqsubseteq \exists r.E \sqcap B$	-	[D]
Super_classes(D, only_named=False)	$D \sqsubseteq \exists r.E \sqcap B$	-	[T, $\exists r.E \sqcap B$ , B]
Instances( $\exists r.E \sqcap B$ )	$D \sqsubseteq \exists r.E \sqcap B$	D(d) r(b,f) E(f) B(b)	[d,b]
Sub_classes(H, only_named=False)	$F \equiv \exists r.G, F \sqsubseteq H$	-	[F, $\exists r.G$ ]
Super_classes(F)	$F \equiv \exists r.G, F \sqsubseteq H$	-	[H, $\exists r.G$ , T]
Super_classes( $\exists r.G$ )	$F \equiv \exists r.G, F \sqsubseteq H$	-	[F, H, T]
Equivalent_classes(F, only_named=False)	$F \equiv \exists r.G, F \sqsubseteq H$	-	[ $\exists r.G$ ]
Equivalent_classes( $\exists r.G$ )	$F \equiv \exists r.G, F \sqsubseteq H$	-	[F]
Instances( $\exists r.G$ )	$F \equiv \exists r.G, F \sqsubseteq H$	r(i,g) G(g)	[i]
Instances(F)	$F \equiv \exists r.G, F \sqsubseteq H$	r(i,g) G(g)	[i]
Instances(H)	$F \equiv \exists r.G, F \sqsubseteq H$	r(i,g) G(g)	[i]
types(i)	$F \equiv \exists r.G, F \sqsubseteq H$	r(i,g) G(g)	[H, F, T]
Sub_classes(C, only_named=False)	$A \sqcap B \equiv R, R \sqsubseteq C$	-	[R, A $\sqcap$ B]
Super_classes(A $\sqcap$ B)	$A \sqcap B \equiv R, R \sqsubseteq C$	-	[R, C, A, B, T]
Equivalent_classes(R, only_named=False)	$A \sqcap B \equiv R, R \sqsubseteq C$	-	[A $\sqcap$ B]
Equivalent_classes(A $\sqcap$ B)	$A \sqcap B \equiv R, R \sqsubseteq C$	-	[R]
Instances(A $\sqcap$ B)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[e,a]
Instances(R)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[a, e]
Instances(C)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[a, e]
Types(a)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[A, B, R, C, T]
Types(e)	$A \sqcap B \equiv R, R \sqsubseteq C$	R(e) A(a) B(a)	[A, B, R, C, T]
Sub_classes(D, only_named=False)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	-	[E, $\exists r.P \sqcap C$ ]
Super_classes( $\exists r.P \sqcap C$ )	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	-	[E, D, T]
Equivalent_classes( $\exists r.P \sqcap C$ )	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	-	[E]
Equivalent_classes(E, only_named=False)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	-	[ $\exists r.P \sqcap C$ ]
Instances( $\exists r.P \sqcap C$ )	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	r(x,y) C(x) P(y)	[x]
Instances(E)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	r(x,y) C(x) P(y)	[x]
Instances(D)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	r(x,y) C(x) P(y)	[x]
Types(x)	$\exists r.P \sqcap C \equiv E, E \sqsubseteq D$	r(x,y) C(x) P(y)	[C]
disjoint_classes(A)	$A \sqcup B$	-	[B]
disjoint_classes(B)	$A \sqcup B$	-	[A]
disjoint_classes(A)	$A \sqcup B, B \equiv C$	-	[B, C]
disjoint_classes(B)	$A \sqcup B, B \equiv C$	-	[A]
disjoint_classes(C)	$A \sqcup B, B \equiv C$	-	[A]
object_property_domains(r)	Domain(r) = A	-	[A, T]
object_property_domains(r)	Domain(r) = AA $\equiv$ B	-	[A, T]
object_property_domains(r2)	Domain(r1) = Ar2 $\sqsubseteq$ r1	-	[A, T]



## 6 owlapy

### 6.1 Subpackages

#### owlapy.class\_expression

OWL Class Expressions [https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions) ClassExpression :=

```
owl_class.py:      Class      nary_boolean_expression.py:      ObjectIntersectionOf,      ObjectUnionOf
class_expression.py: ObjectComplementOf

restriction.py:      ObjectOneOf,      ObjectSomeValuesFrom,      ObjectAllValuesFrom,      ObjectHas-
Value, ObjectHasSelf,      ObjectMinCardinality,      ObjectMaxCardinality,      ObjectExactCardinality,      Data-
SomeValuesFrom,      DataAllValuesFrom,      DataHasValue,      DataMinCardinality,      DataMaxCardinality,
DataExactCardinality
```

#### Submodules

#### owlapy.class\_expression.class\_expression

OWL Base Classes Expressions

#### Classes

<i>OWLPropertyRange</i>	OWL Objects that can be the ranges of properties.
<i>HasOperands</i>	An interface to objects that have a collection of operands.
<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLANonymousClassExpression</i>	A Class Expression which is not a named Class.
<i>OWLBooleanClassExpression</i>	Represent an anonymous boolean class expression.
<i>OWLObjectComplementOf</i>	Represents an ObjectComplementOf class expression in the OWL 2 Specification.

#### Module Contents

**class** owlapy.class\_expression.class\_expression.OWLPropertyRange

Bases: *owlapy.owl\_object.OWLObject*

OWL Objects that can be the ranges of properties.

**class** owlapy.class\_expression.class\_expression.HasOperands

Bases: Generic[\_T]

An interface to objects that have a collection of operands.

##### Parameters

**\_T** – Operand type.

**\_\_slots\_\_** = ()

**abstract operands** () → Iterable[\_T]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**class** owlapy.class\_expression.class\_expression.**OWLClassExpression**

Bases: *owlapy.owl\_data\_ranges.OWLPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

**\_\_slots\_\_** = ()

**abstract is\_owl\_thing** () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**abstract is\_owl\_nothing** () → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of** () → *OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**abstract get\_nnf** () → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.class\_expression.class\_expression.**OWLAnonymousClassExpression**

Bases: *OWLClassExpression*

A Class Expression which is not a named Class.

**is\_owl\_nothing** () → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**is\_owl\_thing** () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**get\_object\_complement\_of** () → *OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf** () → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.class\_expression.class\_expression.**OWLBooleanClassExpression**

Bases: *OWLAnonymousClassExpression*

Represent an anonymous boolean class expression.

**\_\_slots\_\_** = ()

**class** owlapy.class\_expression.class\_expression.**OWLObjectComplementOf** (  
    *op: OWLClassExpression*)

Bases: *OWLBooleanClassExpression*, *owlapy.meta\_classes.HasOperands[OWLClassExpression]*

Represents an ObjectComplementOf class expression in the OWL 2 Specification.

**\_\_slots\_\_** = '\_operand'

**type\_index**: **Final** = 3003

**get\_operand** () → *OWLClassExpression*

**Returns**

The wrapped expression.

**operands** () → *Iterable[OWLClassExpression]*

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

## owlapy.class\_expression.nary\_boolean\_expression

OWL nary boolean expressions

### Classes

<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLBooleanClassExpression</i>	Represent an anonymous boolean class expression.
<i>HasOperands</i>	An interface to objects that have a collection of operands.
<i>OWLNaryBooleanClassExpression</i>	OWLNaryBooleanClassExpression.
<i>OWLObjectUnionOf</i>	A union class expression <code>ObjectUnionOf( CE1 ... CEn )</code> contains all individuals that are instances
<i>OWLObjectIntersectionOf</i>	An intersection class expression <code>ObjectIntersectionOf( CE1 ... CEn )</code> contains all individuals that are instances

### Module Contents

**class** owlapy.class\_expression.nary\_boolean\_expression.**OWLClassExpression**

Bases: *owlapy.owl\_data\_ranges.OWLPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by `ClassExpression`. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

**\_\_slots\_\_** = ()

**abstract is\_owl\_thing** () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**abstract is\_owl\_nothing** () → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of** () → *OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**abstract get\_nnf** () → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

```

class
owlapy.class_expression.nary_boolean_expression.OWLBooleanClassExpression
    Bases: OWLAnonymousClassExpression
    Represent an anonymous boolean class expression.
    __slots__ = ()

class owlapy.class_expression.nary_boolean_expression.HasOperands
    Bases: Generic[_T]
    An interface to objects that have a collection of operands.
        Parameters
            _T – Operand type.
    __slots__ = ()
    abstract operands () → Iterable[_T]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.
        Returns
            The operands.

class owlapy.class_expression.nary_boolean_expression.
    OWLNaryBooleanClassExpression (
        operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
    Bases: owlapy.class_expression.class_expression.OWLBooleanClassExpression,
        owlapy.meta_classes.HasOperands[owlapy.class_expression.class_expression.
        OWLClassExpression]
    OWLNaryBooleanClassExpression.
    __slots__ = ()
    operands () → Iterable[owlapy.class_expression.class_expression.OWLClassExpression]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.
        Returns
            The operands.
    __repr__ ()
        Return repr(self).
    __eq__ (other)
        Return self==value.
    __hash__ ()
        Return hash(self).

class owlapy.class_expression.nary_boolean_expression.OWLObjectUnionOf (
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
    Bases: OWLNaryBooleanClassExpression
    A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances of at least one
    class expression CEi for 1 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Class\_Expressions)
    __slots__ = '_operands'
    type_index: Final = 3002

```

```
class owlapy.class_expression.nary_boolean_expression.
```

```
    OWLObjectIntersectionOf (  
        operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
```

Bases: *OWLNaryBooleanClassExpression*

An intersection class expression `ObjectIntersectionOf( CE1 ... CEn )` contains all individuals that are instances of all class expressions `CEi` for  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Intersection\\_of\\_Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions))

```
    __slots__ = '_operands'
```

```
    type_index: Final = 3001
```

## owlapy.class\_expression.owl\_class

OWL Class

### Classes

<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLObjectComplementOf</i>	Represents an <code>ObjectComplementOf</code> class expression in the OWL 2 Specification.
<i>OWLEntity</i>	Represents Entities in the OWL 2 Specification.
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.

### Module Contents

```
class owlapy.class_expression.owl_class.OWLClassExpression
```

Bases: *owlapy.owl\_data\_ranges.OWLPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by `ClassExpression`. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

```
    __slots__ = ()
```

```
    abstract is_owl_thing () → bool
```

Determines if this expression is the built in class `owl:Thing`. This method does not determine if the class is equivalent to `owl:Thing`.

#### Returns

`Thing`.

#### Return type

True if this expression is owl

```

abstract is_owl_nothing () → bool
    Determines if this expression is the built in class owl:Nothing. This method does not determine if the class
    is equivalent to owl:Nothing.

abstract get_object_complement_of () → OWLObjectComplementOf
    Gets the object complement of this class expression.

    Returns
        A class expression that is the complement of this class expression.

abstract get_nnf () → OWLClassExpression
    Gets the negation normal form of the complement of this expression.

    Returns
        A expression that represents the NNF of the complement of this expression.

class owlapy.class_expression.owl_class.OWLObjectComplementOf (
    op: OWLClassExpression)
    Bases: OWLBooleanClassExpression, owlapy.meta_classes.
    HasOperands[OWLClassExpression]
    Represents an ObjectComplementOf class expression in the OWL 2 Specification.

    __slots__ = '_operand'

    type_index: Final = 3003

    get_operand () → OWLClassExpression

    Returns
        The wrapped expression.

    operands () → Iterable[OWLClassExpression]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

    Returns
        The operands.

    __repr__ ()
        Return repr(self).

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

class owlapy.class_expression.owl_class.OWLEntity
    Bases: OWLNamedObject
    Represents Entities in the OWL 2 Specification.

    __slots__ = ()

    to_string_id () → str

    is_anonymous () → bool

```

```

class owlapy.class_expression.owl_class.IRI (
    namespace: str | owlapy.namespaces.Namespaces, remainder: str)
    Bases: owlapy.owl_annotation.OWLAnnotationSubject, owlapy.owl_annotation.
    OWLAnnotationValue
    An IRI, consisting of a namespace and a remainder.
    __slots__ = ('_namespace', '_remainder', '__weakref__')
    type_index: Final = 0

    static create (namespace: owlapy.namespaces.Namespaces, remainder: str) → IRI
    static create (namespace: str, remainder: str) → IRI
    static create (string: str) → IRI

    __repr__ ()
        Return repr(self).

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

    is_nothing ()
        Determines if this IRI is equal to the IRI that owl:Nothing is named with.

        Returns
            True if this IRI is equal to <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.

    is_thing ()
        Determines if this IRI is equal to the IRI that owl:Thing is named with.

        Returns
            True if this IRI is equal to <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

    is_reserved_vocabulary () → bool
        Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http:
        //www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

        Returns
            True if the IRI is in the reserved vocabulary, otherwise False.

    as_iri () → IRI

        Returns
            if the value is an IRI, return it. Return None otherwise.

    as_str () → str
        CD: Should be deprecated. :returns: The string that specifies the IRI.

    property str: str
        Returns: The string that specifies the IRI.

    property remainder: str
        Returns: The string corresponding to the remainder of the IRI.

```



**get\_short\_form()** → str

Gets the short form.

**Returns**

A string that represents the short form.

**get\_namespace()** → str

**Returns**

The namespace as string.

**get\_remainder()** → str

**Returns**

The remainder (coincident with NCName usually) for this IRI.

**class** owlapy.class\_expression.owl\_class.**OWLClass** (iri: *owlapy.iri.IRI* | str)

Bases: *owlapy.class\_expression.class\_expression.OWLClassExpression*, *owlapy.owl\_object.OWLEntity*

An OWL 2 named Class. Classes can be understood as sets of individuals. (<https://www.w3.org/TR/owl2-syntax/#Classes>)

**\_\_slots\_\_** = ('\_iri', '\_is\_nothing', '\_is\_thing')

**type\_index:** Final = 1001

**property iri:** *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str**

**Gets the string representation of this object**

**Returns**

The IRI as string

**property reminder:** str

The reminder of the IRI

**is\_owl\_thing()** → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**is\_owl\_nothing()** → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**get\_object\_complement\_of()**

→ *owlapy.class\_expression.class\_expression.OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf** () → *OWLClass*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**owlapy.class\_expression.restriction**

OWL Restrictions

**Attributes**

*Literals*

**Classes**

<i>HasFiller</i>	An interface to objects that have a filler.
<i>HasCardinality</i>	An interface to objects that have a cardinality.
<i>HasOperands</i>	An interface to objects that have a collection of operands.
<i>OWLObjectIntersectionOf</i>	An intersection class expression <i>ObjectIntersectionOf</i> ( CE1 ... CEn ) contains all individuals that are instances
<i>OWLObjectUnionOf</i>	A union class expression <i>ObjectUnionOf</i> ( CE1 ... CEn ) contains all individuals that are instances
<i>OWLAnonymousClassExpression</i>	A Class Expression which is not a named Class.
<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLPropertyExpression</i>	Represents a property or possibly the inverse of a property.
<i>OWLObjectPropertyExpression</i>	A high level interface to describe different types of object properties.
<i>OWLDataPropertyExpression</i>	A high level interface to describe different types of data properties.
<i>OWLPropertyRange</i>	OWL Objects that can be the ranges of properties.
<i>OWLDataRange</i>	Represents a <i>DataRange</i> in the OWL 2 Specification.
<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
<i>OWLIndividual</i>	Represents a named or anonymous individual.
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
<i>OWLObject</i>	Base interface for OWL objects
<i>OWLFacet</i>	Enumerations for OWL facets.
<i>OWLRestriction</i>	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.

continues on next page

Table 2 – continued from previous page

<i>OWLHasValueRestriction</i>	Represent a HasValue restriction in the OWL 2
<i>OWLObjectRestriction</i>	Represents an Object Property Restriction in the OWL 2 specification.
<i>OWLQuantifiedRestriction</i>	Represents a quantified restriction.
<i>OWLCardinalityRestriction</i>	Base interface for owl min and max cardinality restriction.
<i>OWLQuantifiedObjectRestriction</i>	Represents a quantified object restriction.
<i>OWLObjectCardinalityRestriction</i>	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
<i>OWLObjectMinCardinality</i>	A minimum cardinality expression <i>ObjectMinCardinality</i> ( <i>n</i> OPE CE ) consists of a nonnegative integer <i>n</i> , an object
<i>OWLObjectMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</i> ( <i>n</i> OPE CE ) consists of a nonnegative integer <i>n</i> , an object
<i>OWLObjectExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> ( <i>n</i> OPE CE ) consists of a nonnegative integer <i>n</i> , an object
<i>OWLObjectSomeValuesFrom</i>	An existential class expression <i>ObjectSomeValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and
<i>OWLObjectAllValuesFrom</i>	A universal class expression <i>ObjectAllValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and a
<i>OWLObjectHasSelf</i>	A self-restriction <i>ObjectHasSelf</i> ( OPE ) consists of an object property expression OPE,
<i>OWLObjectHasValue</i>	A has-value class expression <i>ObjectHasValue</i> ( OPE <i>a</i> ) consists of an object property expression OPE and an
<i>OWLObjectOneOf</i>	An enumeration of individuals <i>ObjectOneOf</i> ( <i>a</i> <sub>1</sub> ... <i>a</i> <sub><i>n</i></sub> ) contains exactly the individuals <i>a</i> <sub><i>i</i></sub> with $1 \leq i \leq n$ .
<i>OWLDataRestriction</i>	Represents a Data Property Restriction.
<i>OWLQuantifiedDataRestriction</i>	Represents a quantified data restriction.
<i>OWLDataCardinalityRestriction</i>	Represents Data Property Cardinality Restrictions.
<i>OWLDataMinCardinality</i>	A minimum cardinality expression <i>DataMinCardinality</i> ( <i>n</i> DPE DR ) consists of a nonnegative integer <i>n</i> , a data
<i>OWLDataMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</i> ( <i>n</i> OPE CE ) consists of a nonnegative integer <i>n</i> , an object
<i>OWLDataExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> ( <i>n</i> OPE CE ) consists of a nonnegative integer <i>n</i> , an
<i>OWLDataSomeValuesFrom</i>	An existential class expression <i>DataSomeValuesFrom</i> ( DPE <sub>1</sub> ... DPE <sub><i>n</i></sub> DR ) consists of <i>n</i> data property expressions
<i>OWLDataAllValuesFrom</i>	A universal class expression <i>DataAllValuesFrom</i> ( DPE <sub>1</sub> ... DPE <sub><i>n</i></sub> DR ) consists of <i>n</i> data property expressions DPE <sub><i>i</i></sub> ,
<i>OWLDataHasValue</i>	A has-value class expression <i>DataHasValue</i> ( DPE <i>lt</i> ) consists of a data property expression DPE and a literal <i>lt</i> ,
<i>OWLDataOneOf</i>	An enumeration of literals <i>DataOneOf</i> ( <i>lt</i> <sub>1</sub> ... <i>lt</i> <sub><i>n</i></sub> ) contains exactly the explicitly specified literals <i>lt</i> <sub><i>i</i></sub> with
<i>OWLDatatypeRestriction</i>	A datatype restriction <i>DatatypeRestriction</i> ( DT F <sub>1</sub> <i>lt</i> <sub>1</sub> ... F <sub><i>n</i></sub> <i>lt</i> <sub><i>n</i></sub> ) consists of a unary datatype DT and <i>n</i> pairs
<i>OWLFacetRestriction</i>	A facet restriction is used to restrict a particular datatype.

## Module Contents

**class** owlapy.class\_expression.restriction.HasFiller

Bases: Generic[\_T]

An interface to objects that have a filler.

**Parameters**

**\_T** – Filler type.

**\_\_slots\_\_** = ()

**abstract** **get\_filler**() → \_T

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

**Returns**

the value

**class** owlapy.class\_expression.restriction.HasCardinality

An interface to objects that have a cardinality.

**\_\_slots\_\_** = ()

**abstract** **get\_cardinality**() → int

Gets the cardinality of a restriction.

**Returns**

The cardinality. A non-negative integer.

**class** owlapy.class\_expression.restriction.HasOperands

Bases: Generic[\_T]

An interface to objects that have a collection of operands.

**Parameters**

**\_T** – Operand type.

**\_\_slots\_\_** = ()

**abstract** **operands**() → Iterable[\_T]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**class** owlapy.class\_expression.restriction.OLObjectIntersectionOf(

*operands: Iterable[owlapy.class\_expression.class\_expression.OWLClassExpression]*)

Bases: OWLNaryBooleanClassExpression

An intersection class expression `ObjectIntersectionOf( CE1 ... CEn )` contains all individuals that are instances of all class expressions `CEi` for  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Intersection\\_of\\_Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions))

**\_\_slots\_\_** = '\_operands'

**type\_index: Final** = 3001

```

class owlapy.class_expression.restriction.OWLObjectUnionOf (
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
Bases: OWLNaryBooleanClassExpression

A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances of at least one
class expression CEi for  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Class\_Expressions)

__slots__ = '_operands'

type_index: Final = 3002

```

```

class owlapy.class_expression.restriction.OWLAnonymousClassExpression
Bases: OWLClassExpression

A Class Expression which is not a named Class.

is_owl_nothing () → bool
    Determines if this expression is the built in class owl:Nothing. This method does not determine if the class
    is equivalent to owl:Nothing.

is_owl_thing () → bool
    Determines if this expression is the built in class owl:Thing. This method does not determine if the class is
    equivalent to owl:Thing.

    Returns
        Thing.

    Return type
        True if this expression is owl

get_object_complement_of () → OWLObjectComplementOf
    Gets the object complement of this class expression.

    Returns
        A class expression that is the complement of this class expression.

get_nnf () → OWLClassExpression
    Gets the negation normal form of the complement of this expression.

    Returns
        A expression that represents the NNF of the complement of this expression.

```

```

class owlapy.class_expression.restriction.OWLClassExpression
Bases: owlapy.owl_data_ranges.OWLPropertyRange

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' prop-
erties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the
structural specification of OWL 2, class expressions are represented by ClassExpression. (https://www.w3.org/
TR/owl2-syntax/#Class\_Expressions)

__slots__ = ()

abstract is_owl_thing () → bool
    Determines if this expression is the built in class owl:Thing. This method does not determine if the class is
    equivalent to owl:Thing.

    Returns
        Thing.

    Return type
        True if this expression is owl

```

**abstract is\_owl\_nothing ()** → bool  
 Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of ()** → *OWLObjectComplementOf*  
 Gets the object complement of this class expression.

**Returns**  
 A class expression that is the complement of this class expression.

**abstract get\_nnf ()** → *OWLClassExpression*  
 Gets the negation normal form of the complement of this expression.

**Returns**  
 An expression that represents the NNF of the complement of this expression.

**class owlapy.class\_expression.restriction.OWLPropertyExpression**  
 Bases: *owlapy.owl\_object.OWLObject*  
 Represents a property or possibly the inverse of a property.

**\_\_slots\_\_ = ()**

**is\_data\_property\_expression ()** → bool

**Returns**  
 True if this is a data property.

**is\_object\_property\_expression ()** → bool

**Returns**  
 True if this is an object property.

**is\_owl\_top\_object\_property ()** → bool  
 Determines if this is the owl:topObjectProperty.

**Returns**  
 topObjectProperty.

**Return type**  
 True if this property is the owl

**is\_owl\_top\_data\_property ()** → bool  
 Determines if this is the owl:topDataProperty.

**Returns**  
 topDataProperty.

**Return type**  
 True if this property is the owl

**class owlapy.class\_expression.restriction.OWLObjectPropertyExpression**  
 Bases: *OWLPropertyExpression*  
 A high level interface to describe different types of object properties.

**\_\_slots\_\_ = ()**

**abstract get\_inverse\_property ()** → *OWLObjectPropertyExpression*  
 Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**abstract** `get_named_property()` → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either `inv(P)` or `P`.

**is\_object\_property\_expression()** → bool

**Returns**

True if this is an object property.

**class** `owlapy.class_expression.restriction.OWLDataPropertyExpression`

Bases: *OWLPropertyExpression*

A high level interface to describe different types of data properties.

**\_\_slots\_\_** = ()

**is\_data\_property\_expression()**

**Returns**

True if this is a data property.

**class** `owlapy.class_expression.restriction.OWLPropertyRange`

Bases: *owlapy.owl\_object.OWLObject*

OWL Objects that can be the ranges of properties.

**class** `owlapy.class_expression.restriction.OWLDatarange`

Bases: *OWLPropertyRange*

Represents a `Datarange` in the OWL 2 Specification.

**class** `owlapy.class_expression.restriction.OWLLiteral`

Bases: *owlapy.owl\_annotation.OWLAnnotationValue*

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(<https://www.w3.org/TR/owl2-syntax/#Literals>)

**\_\_slots\_\_** = ()

**type\_index: Final = 4008**

**get\_literal()** → str

Gets the lexical value of this literal. Note that the language tag is not included.

**Returns**

The lexical value of this literal.

**is\_boolean()** → bool

Whether this literal is typed as boolean.

**parse\_boolean()** → bool

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("<http://www.w3.org/2001/XMLSchema#boolean>").

**Returns**

A bool value that is represented by this literal.

**is\_double** () → bool

Whether this literal is typed as double.

**parse\_double** () → float

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

**Returns**

A double value that is represented by this literal.

**is\_integer** () → bool

Whether this literal is typed as integer.

**parse\_integer** () → int

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

**Returns**

An integer value that is represented by this literal.

**is\_string** () → bool

Whether this literal is typed as string.

**parse\_string** () → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("<http://www.w3.org/2001/XMLSchema#string>").

**Returns**

A string value that is represented by this literal.

**is\_date** () → bool

Whether this literal is typed as date.

**parse\_date** () → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("<http://www.w3.org/2001/XMLSchema#date>").

**Returns**

A date value that is represented by this literal.

**is\_datetime** () → bool

Whether this literal is typed as dateTime.

**parse\_datetime** () → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("<http://www.w3.org/2001/XMLSchema#dateTime>").

**Returns**

A datetime value that is represented by this literal.

**is\_duration** () → bool

Whether this literal is typed as duration.

**parse\_duration** () → pandas.Timedelta

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("<http://www.w3.org/2001/XMLSchema#duration>").



**Returns**

A Timedelta value that is represented by this literal.

**is\_literal()** → bool

**Returns**

true if the annotation value is a literal

**as\_literal()** → *OWLLiteral*

**Returns**

if the value is a literal, returns it. Return None otherwise

**to\_python()** → Literals

**abstract get\_datatype()** → *owlapy.owl\_datatype.OWLDatatype*

Gets the OWLDatatype which types this literal.

**Returns**

The OWLDatatype that types this literal.

**class** owlapy.class\_expression.restriction.**OWLIndividual**

Bases: *owlapy.owl\_object.OWLObject*

Represents a named or anonymous individual.

**\_\_slots\_\_** = ()

**class** owlapy.class\_expression.restriction.**OWLDatatype**(

*iri: owlapy.iri.IRI | owlapy.meta\_classes.HasIRI*)

Bases: *owlapy.owl\_object.OWLEntity, owlapy.owl\_data\_ranges.OWLDataRange*

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

**\_\_slots\_\_** = '\_iri'

**type\_index:** Final = 4001

**property iri:** *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str:** str

Gets the string representation of this object

**Returns**

The IRI as string

**class** owlapy.class\_expression.restriction.**OWLObject**

Base interface for OWL objects

**\_\_slots\_\_** = ()

```

abstract __eq__ (other)
    Return self==value.

abstract __hash__ ()
    Return hash(self).

abstract __repr__ ()
    Return repr(self).

is_anonymous () → bool

class owlapy.class_expression.restriction.OWLFacet (remainder: str, symbolic_form: str,
    operator: Callable[[_X, _X], bool])
    Bases: _Vocabulary, enum.Enum
    Enumerations for OWL facets.
    property symbolic_form
    property operator
    static from_str (name: str) → OWLFacet
    MIN_INCLUSIVE: Final
    MIN_EXCLUSIVE: Final
    MAX_INCLUSIVE: Final
    MAX_EXCLUSIVE: Final
    LENGTH: Final
    MIN_LENGTH: Final
    MAX_LENGTH: Final
    PATTERN: Final
    TOTAL_DIGITS: Final
    FRACTION_DIGITS: Final

owlapy.class_expression.restriction.Literals

class owlapy.class_expression.restriction.OWLRestriction
    Bases: owlapy.class_expression.class_expression.OWLAnonymousClassExpression
    Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
    __slots__ = ()
    abstract get_property () → owlapy.owl_property.OWLPropertyExpression

        Returns
            Property being restricted.

    is_data_restriction () → bool
        Determines if this is a data restriction.

        Returns
            True if this is a data restriction.

```

**is\_object\_restriction()** → bool

Determines if this is an object restriction.

**Returns**

True if this is an object restriction.

**class** owlapy.class\_expression.restriction.**OWLHasValueRestriction**(value: *\_T*)

Bases: *Generic[\_T]*, *OWLRestriction*, *owlapy.meta\_classes.HasFiller[\_T]*

Represent a HasValue restriction in the OWL 2

**Parameters**

*\_T* – The value type.

**\_\_slots\_\_** = ()

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

Return self==value.

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

Return hash(self).

**get\_filler()** → *\_T*

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

**Returns**

the value

**class** owlapy.class\_expression.restriction.**OWLObjectRestriction**

Bases: *OWLRestriction*

Represents an Object Property Restriction in the OWL 2 specification.

**\_\_slots\_\_** = ()

**is\_object\_restriction()** → bool

Determines if this is an object restriction.

**Returns**

True if this is an object restriction.

**abstract get\_property()** → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

**class** owlapy.class\_expression.restriction.**OWLQuantifiedRestriction**

Bases: *Generic[\_T]*, *OWLRestriction*, *owlapy.meta\_classes.HasFiller[\_T]*

Represents a quantified restriction.

**Parameters**

*\_T* – value type

**\_\_slots\_\_** = ()

```

class owlapy.class_expression.restriction.OWLCardinalityRestriction(
    cardinality: int, filler: _F)
    Bases: Generic[_F], OWLQuantifiedRestriction[_F], owlapy.meta_classes.
    HasCardinality
    Base interface for owl min and max cardinality restriction.

    Parameters
        _F – Type of filler.

    __slots__ = ()

    get_cardinality() → int
        Gets the cardinality of a restriction.

    Returns
        The cardinality. A non-negative integer.

    get_filler() → _F
        Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of
        a data restriction this will be a constant (data value). For quantified restriction this will be a class expression
        or a data range.

    Returns
        the value

class owlapy.class_expression.restriction.OWLQuantifiedObjectRestriction(
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
    Bases: OWLQuantifiedRestriction[owlapy.class_expression.class_expression.
    OWLClassExpression], OWLObjectRestriction
    Represents a quantified object restriction.

    __slots__ = ()

    get_filler() → owlapy.class_expression.class_expression.OWLClassExpression
        Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of
        a data restriction this will be a constant (data value). For quantified restriction this will be a class expression
        or a data range.

    Returns
        the value

class owlapy.class_expression.restriction.OWLObjectCardinalityRestriction(
    cardinality: int, property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
    Bases: OWLCardinalityRestriction[owlapy.class_expression.class_expression.
    OWLClassExpression], OWLQuantifiedObjectRestriction
    Represents Object Property Cardinality Restrictions in the OWL 2 specification.

    __slots__ = ()

    get_property() → owlapy.owl_property.OWLObjectPropertyExpression

    Returns
        Property being restricted.

    __repr__()
        Return repr(self).

```

```

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

```

**class** owlapy.class\_expression.restriction.OWLObjectMinCardinality(  
*cardinality: int, property: owlapy.owl\_property.OWLObjectPropertyExpression,*  
*filler: owlapy.class\_expression.class\_expression.OWLClassExpression)*  
Bases: *OWLObjectCardinalityRestriction*

A minimum cardinality expression ObjectMinCardinality( n OPE CE ) consists of a nonnegative integer n, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to at least n different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3008

```

**class** owlapy.class\_expression.restriction.OWLObjectMaxCardinality(  
*cardinality: int, property: owlapy.owl\_property.OWLObjectPropertyExpression,*  
*filler: owlapy.class\_expression.class\_expression.OWLClassExpression)*  
Bases: *OWLObjectCardinalityRestriction*

A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE

to at most n different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3010

```

**class** owlapy.class\_expression.restriction.OWLObjectExactCardinality(  
*cardinality: int, property: owlapy.owl\_property.OWLObjectPropertyExpression,*  
*filler: owlapy.class\_expression.class\_expression.OWLClassExpression)*  
Bases: *OWLObjectCardinalityRestriction*

**An exact cardinality expression ObjectExactCardinality( n OPE CE ) consists of a nonnegative integer n, an object**  
property expression OPE, and a class expression CE, and it contains all those individuals that are connected by to exactly n different individuals that are instances of CE.

([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3009

```

**as\_intersection\_of\_min\_max()**  
→ owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

**Returns**  
The semantically equivalent but structurally simpler form ( $= 1 \text{ R C} \Rightarrow 1 \text{ R C}$  and  $\leq 1 \text{ R C}$ ).

```
class owlapy.class_expression.restriction.OwlObjectSomeValuesFrom(
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: *OwlQuantifiedObjectRestriction*

An existential class expression ObjectSomeValuesFrom( OPE CE ) consists of an object property expression OPE and a class expression CE, and it contains all those individuals that are connected by OPE to an individual that is an instance of CE.

```
__slots__ = ('_property', '_filler')
```

```
type_index: Final = 3005
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
get_property() → owlapy.owl_property.OwlObjectPropertyExpression
```

**Returns**

Property being restricted.

```
class owlapy.class_expression.restriction.OwlObjectAllValuesFrom(
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: *OwlQuantifiedObjectRestriction*

A universal class expression ObjectAllValuesFrom( OPE CE ) consists of an object property expression OPE and a class expression CE, and it contains all those individuals that are connected by OPE only to individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification))

```
__slots__ = ('_property', '_filler')
```

```
type_index: Final = 3006
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
get_property() → owlapy.owl_property.OwlObjectPropertyExpression
```

**Returns**

Property being restricted.

```
class owlapy.class_expression.restriction.OwlObjectHasSelf(
    property: owlapy.owl_property.OwlObjectPropertyExpression)
```

Bases: *OwlObjectRestriction*

A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE, and it contains all those individuals that are connected by OPE to themselves. (<https://www.w3.org/TR/owl2-syntax/#Self-Restriction>)

```

__slots__ = '_property'

type_index: Final = 3011

get_property() → owlapy.owl_property.OWLObjectPropertyExpression

    Returns
        Property being restricted.

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.class_expression.restriction.OWLObjectHasValue(
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    individual: owlapy.owl_individual.OWLIndividual)

Bases: OWLHasValueRestriction[owlapy.owl_individual.OWLIndividual], OWLObjectRestriction

A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ). (https://www.w3.org/TR/owl2-syntax/#Individual\_Value\_Restriction)

__slots__ = ('_property', '_v')

type_index: Final = 3007

get_property() → owlapy.owl_property.OWLObjectPropertyExpression

    Returns
        Property being restricted.

as_some_values_from() → owlapy.class_expression.class_expression.OWLClassExpression
    A convenience method that obtains this restriction as an existential restriction with a nominal filler.

    Returns
        The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

__repr__()
    Return repr(self).

class owlapy.class_expression.restriction.OWLObjectOneOf(
    values: owlapy.owl_individual.OWLIndividual | Iterable[owlapy.owl_individual.OWLIndividual])

Bases: owlapy.class_expression.class_expression.OWLAnonymousClassExpression, owlapy.meta_classes.HasOperands[owlapy.owl_individual.OWLIndividual]

An enumeration of individuals ObjectOneOf( a1 ... an ) contains exactly the individuals ai with  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Enumeration\_of\_Individuals)

__slots__ = '_values'

type_index: Final = 3004

```

**individuals** () → Iterable[*owlapy.owl\_individual.OWLIndividual*]

Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this class expression.

**Returns**

The individuals that are the values of this {`@code ObjectOneOf`} class expression.

**operands** () → Iterable[*owlapy.owl\_individual.OWLIndividual*]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**as\_object\_union\_of** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

Simplifies this enumeration to a union of singleton nominals.

**Returns**

This enumeration in a more standard DL form.  $\text{simp}(\{a\}) = \{a\}$   $\text{simp}(\{a_0, \dots, \{a_n\}) = \text{unionOf}(\{a_0\}, \dots, \{a_n\})$

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

Return hash(self).

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

Return self==value.

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.class\_expression.restriction.OWLDataRestriction

Bases: *OWLRestriction*

Represents a Data Property Restriction.

**\_\_slots\_\_** = ()

**is\_data\_restriction** () → bool

Determines if this is a data restriction.

**Returns**

True if this is a data restriction.

**class** owlapy.class\_expression.restriction.OWLQuantifiedDataRestriction (  
    *filler: owlapy.owl\_data\_ranges.OWLDataRange*)

Bases: *OWLQuantifiedRestriction*[*owlapy.owl\_data\_ranges.OWLDataRange*], *OWLDataRestriction*

Represents a quantified data restriction.

**\_\_slots\_\_** = ()

**get\_filler** () → *owlapy.owl\_data\_ranges.OWLDataRange*

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

**Returns**

the value



```

class owlapy.class_expression.restriction.OWLDataCardinalityRestriction(
    cardinality: int, property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)

Bases: OWLCardinalityRestriction[owlapy.owl_data_ranges.OWLDataRange],
        OWLQuantifiedDataRestriction, OWLDataRestriction

Represents Data Property Cardinality Restrictions.

__slots__ = ()

get_property() → owlapy.owl_property.OWLDataPropertyExpression

    Returns
    Property being restricted.

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

class owlapy.class_expression.restriction.OWLDataMinCardinality(cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)

Bases: OWLDataCardinalityRestriction

A minimum cardinality expression DataMinCardinality( n DPE DR ) consists of a nonnegative integer n, a data
property expression DPE, and a unary data range DR, and it contains all those individuals that are connected by
DPE to at least n different literals in DR. (https://www.w3.org/TR/owl2-syntax/#Minimum\_Cardinality)

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3015

class owlapy.class_expression.restriction.OWLDataMaxCardinality(cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)

Bases: OWLDataCardinalityRestriction

A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an
object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by
OPE to at most n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Maximum\_Cardinality)

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3017

class owlapy.class_expression.restriction.OWLDataExactCardinality(
    cardinality: int, property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)

Bases: OWLDataCardinalityRestriction

An exact cardinality expression ObjectExactCardinality( n OPE CE ) consists of a nonnegative integer n, an object
property expression OPE, and a class expression CE, and it contains all those individuals that are connected

```

by OPE to exactly  $n$  different individuals that are instances of CE ([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3016
```

```
as_intersection_of_min_max()
```

→ *owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

#### Returns

The semantically equivalent but structurally simpler form  $(= 1 \text{ R D}) = \geq 1 \text{ R D}$  and  $\leq 1 \text{ R D}$ .

```
class owlapy.class_expression.restriction.OWLDataSomeValuesFrom(  
    property: owlapy.owl_property.OWLDataPropertyExpression,  
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: *OWLQuantifiedDataRestriction*

An existential class expression `DataSomeValuesFrom( DPE1 ... DPEn DR )` consists of  $n$  data property expressions `DPEi`,  $1 \leq i \leq n$ , and a data range `DR` whose arity must be  $n$ . Such a class expression contains all those individuals that are connected by `DPEi` to literals `lti`,  $1 \leq i \leq n$ , such that the tuple  $( lt_1 , \dots , lt_n )$  is in `DR`. A class expression of the form `DataSomeValuesFrom( DPE DR )` can be seen as a syntactic shortcut for the class expression `DataMinCardinality( 1 DPE DR )`. ([https://www.w3.org/TR/owl2-syntax/#Existential\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Existential_Quantification_2))

```
__slots__ = '_property'
```

```
type_index: Final = 3012
```

```
__repr__()
```

Return `repr(self)`.

```
__eq__(other)
```

Return `self==value`.

```
__hash__()
```

Return `hash(self)`.

```
get_property() → owlapy.owl_property.OWLDataPropertyExpression
```

#### Returns

Property being restricted.

```
class owlapy.class_expression.restriction.OWLDataAllValuesFrom(  
    property: owlapy.owl_property.OWLDataPropertyExpression,  
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: *OWLQuantifiedDataRestriction*

A universal class expression `DataAllValuesFrom( DPE1 ... DPEn DR )` consists of  $n$  data property expressions `DPEi`,  $1 \leq i \leq n$ , and a data range `DR` whose arity must be  $n$ . Such a class expression contains all those individuals that

**are connected by `DPEi` only to literals `lti`,  $1 \leq i \leq n$ , such that each tuple  $( lt_1 , \dots , lt_n )$  is in `DR`.**

#### A class

expression of the form `DataAllValuesFrom( DPE DR )` can be seen as a syntactic shortcut for the class expression `DataMaxCardinality( 0 DPE DataComplementOf( DR ) )`. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification_2))

```
__slots__ = '_property'
```

**type\_index: Final = 3013**

**\_\_repr\_\_()**

Return repr(self).

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

Return self==value.

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

Return hash(self).

**get\_property()** → *owlapy.owl\_property.OWLDataPropertyExpression*

#### Returns

Property being restricted.

```
class owlapy.class_expression.restriction.OWLDataHasValue(  
    property: owlapy.owl_property.OWLDataPropertyExpression,  
    value: owlapy.owl_literal.OWLLiteral)
```

Bases: *OWLHasValueRestriction[owlapy.owl\_literal.OWLLiteral], OWLDataRestriction*

A has-value class expression `DataHasValue( DPE lt )` consists of a data property expression DPE and a literal lt, and it contains all those individuals that are connected by DPE to lt. Each such class expression can be seen as a syntactic shortcut for the class expression `DataSomeValuesFrom( DPE DataOneOf( lt ) )`. ([https://www.w3.org/TR/owl2-syntax/#Literal\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Literal_Value_Restriction))

**\_\_slots\_\_ = '\_property'**

**type\_index: Final = 3014**

**\_\_repr\_\_()**

Return repr(self).

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

Return self==value.

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

Return hash(self).

**as\_some\_values\_from()** → *owlapy.class\_expression.class\_expression.OWLClassExpression*

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

#### Returns

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

**get\_property()** → *owlapy.owl\_property.OWLDataPropertyExpression*

#### Returns

Property being restricted.

```
class owlapy.class_expression.restriction.OWLDataOneOf(  
    values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])
```

Bases: *owlapy.owl\_data\_ranges.OWLDataRange, owlapy.meta\_classes.HasOperands[owlapy.owl\_literal.OWLLiteral]*

An enumeration of literals `DataOneOf( lt1 ... ltn )` contains exactly the explicitly specified literals lti with  $1 \leq i \leq n$ . The resulting data range has arity one. ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Literals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals))

```

type_index: Final = 4003

values () → Iterable[owlapy.owl_literal.OWLLiteral]
    Gets the values that are in the oneOf.

    Returns
        The values of this {@code DataOneOf} class expression.

operands () → Iterable[owlapy.owl_literal.OWLLiteral]
    Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

    Returns
        The operands.

__hash__ ()
    Return hash(self).

__eq__ (other)
    Return self==value.

__repr__ ()
    Return repr(self).

class owlapy.class_expression.restriction.OWLDatatypeRestriction (
    type_: owlapy.owl_datatype.OWLDatatype,
    facet_restrictions: OWLFacetRestriction | Iterable[OWLFacetRestriction])
Bases: owlapy.owl_data_ranges.OWLDataRange

A datatype restriction DatatypeRestriction( DT F1 lti ... Fn ltn ) consists of a unary datatype DT and n pairs (
Fi , lti ). The resulting data range is unary and is obtained by restricting the value space of DT according to the
semantics of all ( Fi , vi ) (multiple pairs are interpreted conjunctively), where vi are the data values of the literals
lti. (https://www.w3.org/TR/owl2-syntax/#Datatype\_Restrictions)

__slots__ = ('_type', '_facet_restrictions')

type_index: Final = 4006

get_datatype () → owlapy.owl_datatype.OWLDatatype

get_facet_restrictions () → Sequence[OWLFacetRestriction]

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

__repr__ ()
    Return repr(self).

class owlapy.class_expression.restriction.OWLFacetRestriction (
    facet: owlapy.vocab.OWLFacet, literal: Literals)
Bases: owlapy.owl_object.OWLObject

A facet restriction is used to restrict a particular datatype.

__slots__ = ('_facet', '_literal')

type_index: Final = 4007

```

```

get_facet () → owlapy.vocab.OWLFacet
get_facet_value () → owlapy.owl_literal.OWLLiteral

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

__repr__ ()
    Return repr(self).

```

## Attributes

<i>OWLThing</i>
<i>OWLNothing</i>

## Classes

<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLAnonymousClassExpression</i>	A Class Expression which is not a named Class.
<i>OWLBooleanClassExpression</i>	Represent an anonymous boolean class expression.
<i>OWLObjectComplementOf</i>	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>OWLNaryBooleanClassExpression</i>	OWLNaryBooleanClassExpression.
<i>OWLObjectUnionOf</i>	A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances
<i>OWLObjectIntersectionOf</i>	An intersection class expression ObjectIntersectionOf( CE1 ... CEn ) contains all individuals that are instances
<i>OWLRestriction</i>	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
<i>OWLQuantifiedRestriction</i>	Represents a quantified restriction.
<i>OWLQuantifiedObjectRestriction</i>	Represents a quantified object restriction.
<i>OWLObjectRestriction</i>	Represents an Object Property Restriction in the OWL 2 specification.
<i>OWLHasValueRestriction</i>	Represent a HasValue restriction in the OWL 2
<i>OWLDataRestriction</i>	Represents a Data Property Restriction.
<i>OWLCardinalityRestriction</i>	Base interface for owl min and max cardinality restriction.
<i>OWLObjectCardinalityRestriction</i>	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
<i>OWLObjectHasSelf</i>	A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE,

continues on next page

Table 3 – continued from previous page

<i>OWLDataOneOf</i>	An enumeration of literals $\text{DataOneOf}(l_1 \dots l_n)$ contains exactly the explicitly specified literals $l_i$ with
<i>OWLQuantifiedDataRestriction</i>	Represents a quantified data restriction.
<i>OWLDataCardinalityRestriction</i>	Represents Data Property Cardinality Restrictions.
<i>OWLObjectSomeValuesFrom</i>	An existential class expression $\text{ObjectSomeValuesFrom}(\text{OPE CE})$ consists of an object property expression $\text{OPE}$ and
<i>OWLObjectAllValuesFrom</i>	A universal class expression $\text{ObjectAllValuesFrom}(\text{OPE CE})$ consists of an object property expression $\text{OPE}$ and a
<i>OWLObjectHasValue</i>	A has-value class expression $\text{ObjectHasValue}(\text{OPE } a)$ consists of an object property expression $\text{OPE}$ and an
<i>OWLDatatypeRestriction</i>	A datatype restriction $\text{DatatypeRestriction}(\text{DT } F_1 l_1 \dots F_n l_n)$ consists of a unary datatype $\text{DT}$ and $n$ pairs
<i>OWLFacet</i>	Enumerations for OWL facets.
<i>OWLFacetRestriction</i>	A facet restriction is used to restrict a particular datatype.
<i>OWLObjectMinCardinality</i>	A minimum cardinality expression $\text{ObjectMinCardinality}(n \text{ OPE CE})$ consists of a nonnegative integer $n$ , an object
<i>OWLObjectMaxCardinality</i>	A maximum cardinality expression $\text{ObjectMaxCardinality}(n \text{ OPE CE})$ consists of a nonnegative integer $n$ , an object
<i>OWLObjectExactCardinality</i>	An exact cardinality expression $\text{ObjectExactCardinality}(n \text{ OPE CE})$ consists of a nonnegative integer $n$ , an object
<i>OWLDataSomeValuesFrom</i>	An existential class expression $\text{DataSomeValuesFrom}(\text{DPE}_1 \dots \text{DPE}_n \text{ DR})$ consists of $n$ data property expressions
<i>OWLDataAllValuesFrom</i>	A universal class expression $\text{DataAllValuesFrom}(\text{DPE}_1 \dots \text{DPE}_n \text{ DR})$ consists of $n$ data property expressions $\text{DPE}_i$ ,
<i>OWLDataHasValue</i>	A has-value class expression $\text{DataHasValue}(\text{DPE } l)$ consists of a data property expression $\text{DPE}$ and a literal $l$ ,
<i>OWLDataMinCardinality</i>	A minimum cardinality expression $\text{DataMinCardinality}(n \text{ DPE DR})$ consists of a nonnegative integer $n$ , a data
<i>OWLDataMaxCardinality</i>	A maximum cardinality expression $\text{ObjectMaxCardinality}(n \text{ OPE CE})$ consists of a nonnegative integer $n$ , an object
<i>OWLDataExactCardinality</i>	An exact cardinality expression $\text{ObjectExactCardinality}(n \text{ OPE CE})$ consists of a nonnegative integer $n$ , an
<i>OWLObjectOneOf</i>	An enumeration of individuals $\text{ObjectOneOf}(a_1 \dots a_n)$ contains exactly the individuals $a_i$ with $1 \leq i \leq n$ .
<i>OWLRDFVocabulary</i>	Enumerations for OWL/RDF vocabulary.

## Package Contents

**class** owlapy.class\_expression.OWLClassExpression

Bases: *owlapy.owl\_data\_ranges.OWLPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

**\_\_slots\_\_** = ()

**abstract is\_owl\_thing**() → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**abstract is\_owl\_nothing**() → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of**() → *OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**abstract get\_nnf**() → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.class\_expression.OWLAnonymousClassExpression

Bases: *OWLClassExpression*

A Class Expression which is not a named Class.

**is\_owl\_nothing**() → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**is\_owl\_thing**() → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**get\_object\_complement\_of**() → *OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf** () → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.class\_expression.OWLBooleanClassExpression

Bases: *OWLObjectClassExpression*

Represent an anonymous boolean class expression.

**\_\_slots\_\_** = ()

**class** owlapy.class\_expression.OWLObjectComplementOf (op: *OWLClassExpression*)

Bases: *OWLBooleanClassExpression*, *owlapy.meta\_classes.HasOperands[OWLClassExpression]*

Represents an ObjectComplementOf class expression in the OWL 2 Specification.

**\_\_slots\_\_** = '\_operand'

**type\_index**: Final = 3003

**get\_operand** () → *OWLClassExpression*

**Returns**

The wrapped expression.

**operands** () → Iterable[*OWLClassExpression*]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**class** owlapy.class\_expression.OWLClass (iri: *owlapy.iri.IRI* | str)

Bases: *owlapy.class\_expression.class\_expression.OWLClassExpression*, *owlapy.owl\_object.OWLEntity*

An OWL 2 named Class. Classes can be understood as sets of individuals. (<https://www.w3.org/TR/owl2-syntax/#Classes>)

**\_\_slots\_\_** = ('\_iri', '\_is\_nothing', '\_is\_thing')

**type\_index**: Final = 1001

**property iri**: *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.



**property str**

Gets the string representation of this object

**Returns**

The IRI as string

**property reminder: str**

The reminder of the IRI

**is\_owl\_thing()** → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**is\_owl\_nothing()** → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**get\_object\_complement\_of()**

→ *owlapy.class\_expression.class\_expression.OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf()** → *OWLClass*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.class\_expression.OWLNaryBooleanClassExpression (  
    *operands: Iterable[owlapy.class\_expression.class\_expression.OWLClassExpression]*)

Bases: *owlapy.class\_expression.class\_expression.OWLBooleanClassExpression*,  
*owlapy.meta\_classes.HasOperands[owlapy.class\_expression.class\_expression.OWLClassExpression]*

OWLNaryBooleanClassExpression.

**\_\_slots\_\_** = ()

**operands()** → *Iterable[owlapy.class\_expression.class\_expression.OWLClassExpression]*

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**\_\_repr\_\_()**

Return repr(self).

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

Return self==value.

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

Return hash(self).

```

class owlapy.class_expression.OWLObjectUnionOf (
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
    Bases: OWLNaryBooleanClassExpression

    A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances of at least one
    class expression CEi for  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Class\_Expressions)

    __slots__ = '_operands'

    type_index: Final = 3002

class owlapy.class_expression.OWLObjectIntersectionOf (
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
    Bases: OWLNaryBooleanClassExpression

    An intersection class expression ObjectIntersectionOf( CE1 ... CEn ) contains all individuals that are instances of
    all class expressions CEi for  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Intersection\_of\_Class\_Expressions)

    __slots__ = '_operands'

    type_index: Final = 3001

class owlapy.class_expression.OWLRestriction
    Bases: owlapy.class_expression.class_expression.OWLAnonymousClassExpression

    Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.

    __slots__ = ()

    abstract get_property () → owlapy.owl_property.OWLPropertyExpression

        Returns
            Property being restricted.

    is_data_restriction () → bool
        Determines if this is a data restriction.

        Returns
            True if this is a data restriction.

    is_object_restriction () → bool
        Determines if this is an object restriction.

        Returns
            True if this is an object restriction.

class owlapy.class_expression.OWLQuantifiedRestriction
    Bases: Generic[_T], OWLRestriction, owlapy.meta_classes.HasFiller[_T]

    Represents a quantified restriction.

    Parameters
        _T – value type

    __slots__ = ()

```

```

class owlapy.class_expression.OWLQuantifiedObjectRestriction (
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
    Bases:      OWLQuantifiedRestriction[owlapy.class_expression.class_expression.
        OWLClassExpression], OWLObjectRestriction
    Represents a quantified object restriction.

    __slots__ = ()

    get_filler() → owlapy.class_expression.class_expression.OWLClassExpression
        Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of
        a data restriction this will be a constant (data value). For quantified restriction this will be a class expression
        or a data range.

        Returns
            the value

class owlapy.class_expression.OWLObjectRestriction
    Bases: OWLRestriction
    Represents an Object Property Restriction in the OWL 2 specification.

    __slots__ = ()

    is_object_restriction() → bool
        Determines if this is an object restriction.

        Returns
            True if this is an object restriction.

    abstract get_property() → owlapy.owl_property.OWLObjectPropertyExpression

        Returns
            Property being restricted.

class owlapy.class_expression.OWLHasValueRestriction (value: _T)
    Bases: Generic[_T], OWLRestriction, owlapy.meta_classes.HasFiller[_T]
    Represent a HasValue restriction in the OWL 2

        Parameters
            _T – The value type.

    __slots__ = ()

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

    get_filler() → _T
        Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of
        a data restriction this will be a constant (data value). For quantified restriction this will be a class expression
        or a data range.

        Returns
            the value

```

```

class owlapy.class_expression.OWLDataRestriction
    Bases: OWLRestriction

    Represents a Data Property Restriction.

    __slots__ = ()

    is_data_restriction() → bool
        Determines if this is a data restriction.

        Returns
            True if this is a data restriction.

class owlapy.class_expression.OWLCardinalityRestriction(cardinality: int, filler: _F)
    Bases: Generic[_F], OWLQuantifiedRestriction[_F], owlapy.meta_classes.HasCardinality

    Base interface for owl min and max cardinality restriction.

    Parameters
        _F – Type of filler.

    __slots__ = ()

    get_cardinality() → int
        Gets the cardinality of a restriction.

    Returns
        The cardinality. A non-negative integer.

    get_filler() → _F
        Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of
        a data restriction this will be a constant (data value). For quantified restriction this will be a class expression
        or a data range.

    Returns
        the value

class owlapy.class_expression.OWLObjectCardinalityRestriction(cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
    Bases: OWLCardinalityRestriction[owlapy.class_expression.class_expression.OWLClassExpression], OWLQuantifiedObjectRestriction

    Represents Object Property Cardinality Restrictions in the OWL 2 specification.

    __slots__ = ()

    get_property() → owlapy.owl_property.OWLObjectPropertyExpression

    Returns
        Property being restricted.

    __repr__()
        Return repr(self).

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

```

```

class owlapy.class_expression.OWLObjectHasSelf (
    property: owlapy.owl_property.OWLObjectPropertyExpression)
    Bases: OWLObjectRestriction

    A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE, and it contains all those
    individuals that are connected by OPE to themselves. (https://www.w3.org/TR/owl2-syntax/#Self-Restriction)

    __slots__ = '_property'

    type_index: Final = 3011

    get_property () → owlapy.owl_property.OWLObjectPropertyExpression

        Returns
            Property being restricted.

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

    __repr__ ()
        Return repr(self).

class owlapy.class_expression.OWLDataOneOf (
    values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])
    Bases: owlapy.owl_data_ranges.OWLDataRange, owlapy.meta_classes.
    HasOperands[owlapy.owl_literal.OWLLiteral]

    An enumeration of literals DataOneOf( lt1 ... ltn ) contains exactly the explicitly specified literals lti with  $1 \leq i \leq n$ .
    The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration\_of\_Literals)

    type_index: Final = 4003

    values () → Iterable[owlapy.owl_literal.OWLLiteral]
        Gets the values that are in the oneOf.

        Returns
            The values of this {@code DataOneOf} class expression.

    operands () → Iterable[owlapy.owl_literal.OWLLiteral]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
            The operands.

    __hash__ ()
        Return hash(self).

    __eq__ (other)
        Return self==value.

    __repr__ ()
        Return repr(self).

class owlapy.class_expression.OWLQuantifiedDataRestriction (
    filler: owlapy.owl_data_ranges.OWLDataRange)

```

Bases: *OWLQuantifiedRestriction*[*owlapy.owl\_data\_ranges.OWLDataRange*], *OWLDataRestriction*

Represents a quantified data restriction.

**\_\_slots\_\_** = ()

**get\_filler**() → *owlapy.owl\_data\_ranges.OWLDataRange*

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

**Returns**

the value

**class** *owlapy.class\_expression.OWLDataCardinalityRestriction* (*cardinality: int*,  
*property: owlapy.owl\_property.OWLDataPropertyExpression*,  
*filler: owlapy.owl\_data\_ranges.OWLDataRange*)

Bases: *OWLCardinalityRestriction*[*owlapy.owl\_data\_ranges.OWLDataRange*],  
*OWLQuantifiedDataRestriction*, *OWLDataRestriction*

Represents Data Property Cardinality Restrictions.

**\_\_slots\_\_** = ()

**get\_property**() → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

**\_\_repr\_\_**()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**class** *owlapy.class\_expression.OWLObjectSomeValuesFrom* (  
*property: owlapy.owl\_property.OWLObjectPropertyExpression*,  
*filler: owlapy.class\_expression.class\_expression.OWLClassExpression*)

Bases: *OWLQuantifiedObjectRestriction*

An existential class expression *ObjectSomeValuesFrom*( OPE CE ) consists of an object property expression OPE and a class expression CE, and it contains all those individuals that are connected by OPE to an individual that is an instance of CE.

**\_\_slots\_\_** = ('\_property', '\_filler')

**type\_index**: **Final** = 3005

**\_\_repr\_\_**()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

**class** owlapy.class\_expression.OWLObjectAllValuesFrom (  
    *property: owlapy.owl\_property.OWLObjectPropertyExpression,*  
    *filler: owlapy.class\_expression.class\_expression.OWLClassExpression)*

Bases: *OWLQuantifiedObjectRestriction*

A universal class expression ObjectAllValuesFrom( OPE CE ) consists of an object property expression OPE and a class expression CE, and it contains all those individuals that are connected by OPE only to individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification))

**\_\_slots\_\_** = ('\_property', '\_filler')

**type\_index:** Final = 3006

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

**class** owlapy.class\_expression.OWLObjectHasValue (  
    *property: owlapy.owl\_property.OWLObjectPropertyExpression,*  
    *individual: owlapy.owl\_individual.OWLIndividual)*

Bases: *OWLHasValueRestriction[owlapy.owl\_individual.OWLIndividual], OWLObjectRestriction*

A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ). ([https://www.w3.org/TR/owl2-syntax/#Individual\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Individual_Value_Restriction))

**\_\_slots\_\_** = ('\_property', '\_v')

**type\_index:** Final = 3007

**get\_property** () → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

**as\_some\_values\_from** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

**Returns**

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

```

__repr__ ()
    Return repr(self).

class owlapy.class_expression.OWLDatatypeRestriction (
    type_: owlapy.owl_datatype.OWLDatatype,
    facet_restrictions: OWLFacetRestriction | Iterable[OWLFacetRestriction])
    Bases: owlapy.owl_data_ranges.OWLDataRange

    A datatype restriction DatatypeRestriction( DT F1 lt1 ... Fn ltn ) consists of a unary datatype DT and n pairs (
    Fi , lti ). The resulting data range is unary and is obtained by restricting the value space of DT according to the
    semantics of all ( Fi , vi ) (multiple pairs are interpreted conjunctively), where vi are the data values of the literals
    lti. (https://www.w3.org/TR/owl2-syntax/#Datatype\_Restrictions)

    __slots__ = ('_type', '_facet_restrictions')

    type_index: Final = 4006

    get_datatype () → owlapy.owl_datatype.OWLDatatype

    get_facet_restrictions () → Sequence[OWLFacetRestriction]

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

    __repr__ ()
        Return repr(self).

class owlapy.class_expression.OWLFacet (remainder: str, symbolic_form: str,
    operator: Callable[[_X, _X], bool])
    Bases: _Vocabulary, enum.Enum

    Enumerations for OWL facets.

    property symbolic_form

    property operator

    static from_str (name: str) → OWLFacet

    MIN_INCLUSIVE: Final

    MIN_EXCLUSIVE: Final

    MAX_INCLUSIVE: Final

    MAX_EXCLUSIVE: Final

    LENGTH: Final

    MIN_LENGTH: Final

    MAX_LENGTH: Final

    PATTERN: Final

    TOTAL_DIGITS: Final

```



```

FRACTION_DIGITS: Final

class owlapy.class_expression.OWLFacetRestriction (facet: owlapy.vocab.OWLFacet,
    literal: Literals)
    Bases: owlapy.owl_object.OWLObject
    A facet restriction is used to restrict a particular datatype.
    __slots__ = ('_facet', '_literal')
    type_index: Final = 4007
    get_facet () → owlapy.vocab.OWLFacet
    get_facet_value () → owlapy.owl_literal.OWLLiteral
    __eq__ (other)
        Return self==value.
    __hash__ ()
        Return hash(self).
    __repr__ ()
        Return repr(self).

class owlapy.class_expression.OWLObjectMinCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
    Bases: OWLObjectCardinalityRestriction
    A minimum cardinality expression ObjectMinCardinality( n OPE CE ) consists of a nonnegative integer n, an object
    property expression OPE, and a class expression CE, and it contains all those individuals that are connected by
    OPE to at least n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Minimum\_Cardinality)
    __slots__ = ('_cardinality', '_filler', '_property')
    type_index: Final = 3008

class owlapy.class_expression.OWLObjectMaxCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
    Bases: OWLObjectCardinalityRestriction
    A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an
    object property expression OPE, and a class expression CE, and it contains all those individuals that are connected
    by OPE
        to at most n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Maximum\_Cardinality)
    __slots__ = ('_cardinality', '_filler', '_property')
    type_index: Final = 3010

class owlapy.class_expression.OWLObjectExactCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
    Bases: OWLObjectCardinalityRestriction

```

An exact cardinality expression **ObjectExactCardinality( n OPE CE )** consists of a nonnegative integer *n*, an object

property expression OPE, and a class expression CE, and it contains all those individuals that are connected by to exactly *n* different individuals that are instances of CE.

([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3009
```

```
as_intersection_of_min_max()
```

→ *owlapy.class\_expression.nary\_boolean\_expression.OwlObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

#### Returns

The semantically equivalent but structurally simpler form ( $= 1 R C$ )  $\Rightarrow 1 R C$  and  $\leq 1 R C$ .

```
class owlapy.class_expression.OwlDataSomeValuesFrom(
```

```
    property: owlapy.owl_property.OwlDataPropertyExpression,
```

```
    filler: owlapy.owl_data_ranges.OwlDataRange)
```

Bases: *OwlQuantifiedDataRestriction*

An existential class expression **DataSomeValuesFrom( DPE<sub>1</sub> ... DPE<sub>n</sub> DR )** consists of *n* data property expressions DPE<sub>*i*</sub>,  $1 \leq i \leq n$ , and a data range DR whose arity must be *n*. Such a class expression contains all those individuals that are connected by DPE<sub>*i*</sub> to literals lti,  $1 \leq i \leq n$ , such that the tuple ( lt<sub>1</sub> , ..., lt<sub>n</sub> ) is in DR. A class expression of the form **DataSomeValuesFrom( DPE DR )** can be seen as a syntactic shortcut for the class expression **DataMinCardinality( 1 DPE DR )**. ([https://www.w3.org/TR/owl2-syntax/#Existential\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Existential_Quantification_2))

```
__slots__ = '_property'
```

```
type_index: Final = 3012
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
get_property() → owlapy.owl_property.OwlDataPropertyExpression
```

#### Returns

Property being restricted.

```
class owlapy.class_expression.OwlDataAllValuesFrom(
```

```
    property: owlapy.owl_property.OwlDataPropertyExpression,
```

```
    filler: owlapy.owl_data_ranges.OwlDataRange)
```

Bases: *OwlQuantifiedDataRestriction*

A universal class expression **DataAllValuesFrom( DPE<sub>1</sub> ... DPE<sub>n</sub> DR )** consists of *n* data property expressions DPE<sub>*i*</sub>,  $1 \leq i \leq n$ , and a data range DR whose arity must be *n*. Such a class expression contains all those individuals that

**are connected by DPE<sub>*i*</sub> only to literals lti,  $1 \leq i \leq n$ , such that each tuple ( lt<sub>1</sub> , ..., lt<sub>n</sub> ) is in DR.**

#### A class

expression of the form **DataAllValuesFrom( DPE DR )** can be seen as a syntactic shortcut for the class expression **DataMaxCardinality( 0 DPE DataComplementOf( DR ) )**. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification_2))

```

__slots__ = '_property'

type_index: Final = 3013

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

get_property () → owlapy.owl_property.OWLDataPropertyExpression

```

#### Returns

Property being restricted.

```

class owlapy.class_expression.OWLDataHasValue (
    property: owlapy.owl_property.OWLDataPropertyExpression,
    value: owlapy.owl_literal.OWLLiteral)

Bases: OWLHasValueRestriction[owlapy.owl_literal.OWLLiteral], OWLDataRestriction

```

A has-value class expression `DataHasValue( DPE It )` consists of a data property expression DPE and a literal It, and it contains all those individuals that are connected by DPE to It. Each such class expression can be seen as a syntactic shortcut for the class expression `DataSomeValuesFrom( DPE DataOneOf( It ) )`. ([https://www.w3.org/TR/owl2-syntax/#Literal\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Literal_Value_Restriction))

```

__slots__ = '_property'

type_index: Final = 3014

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

```

```

as_some_values_from () → owlapy.class_expression.class_expression.OWLClassExpression

```

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

#### Returns

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

```

get_property () → owlapy.owl_property.OWLDataPropertyExpression

```

#### Returns

Property being restricted.

```

class owlapy.class_expression.OWLDataMinCardinality (cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)

Bases: OWLDataCardinalityRestriction

```

A minimum cardinality expression `DataMinCardinality( n DPE DR )` consists of a nonnegative integer n, a data property expression DPE, and a unary data range DR, and it contains all those individuals that are connected by DPE to at least n different literals in DR. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3015

```

**class** owlapy.class\_expression.OWLDataMaxCardinality(*cardinality: int*,  
*property: owlapy.owl\_property.OWLDataPropertyExpression*,  
*filler: owlapy.owl\_data\_ranges.OWLDataRange*)

Bases: *OWLDataCardinalityRestriction*

A maximum cardinality expression ObjectMaxCardinality( *n* OPE CE ) consists of a nonnegative integer *n*, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to at most *n* different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3017

```

**class** owlapy.class\_expression.OWLDataExactCardinality(*cardinality: int*,  
*property: owlapy.owl\_property.OWLDataPropertyExpression*,  
*filler: owlapy.owl\_data\_range.OWLDataRange*)

Bases: *OWLDataCardinalityRestriction*

An exact cardinality expression ObjectExactCardinality( *n* OPE CE ) consists of a nonnegative integer *n*, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to exactly *n* different individuals that are instances of CE ([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3016

```

**as\_intersection\_of\_min\_max()**  
→ *owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

**Returns**  
The semantically equivalent but structurally simpler form ( $= 1 \text{ R D} \Rightarrow 1 \text{ R D}$  and  $\leq 1 \text{ R D}$ ).

**class** owlapy.class\_expression.OWLObjectOneOf(  
*values: owlapy.owl\_individual.OWLIndividual | Iterable[owlapy.owl\_individual.OWLIndividual]*)

Bases: *owlapy.class\_expression.class\_expression.OWLAnonymousClassExpression*,  
*owlapy.meta\_classes.HasOperands[owlapy.owl\_individual.OWLIndividual]*

An enumeration of individuals ObjectOneOf( *a1 ... an* ) contains exactly the individuals *ai* with  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals))

```

__slots__ = '_values'

type_index: Final = 3004

```

**individuals()** → *Iterable[owlapy.owl\_individual.OWLIndividual]*

Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this class expression.

**Returns**  
The individuals that are the values of this {`@code` ObjectOneOf} class expression.

**operands** () → Iterable[*owlapy.owl\_individual.OWLIndividual*]  
 Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**  
 The operands.

**as\_object\_union\_of** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*  
 Simplifies this enumeration to a union of singleton nominals.

**Returns**  
 This enumeration in a more standard DL form.  $\text{simp}(\{a\}) = \{a\}$   $\text{simp}(\{a_0, \dots, \{a_n\}) = \text{unionOf}(\{a_0\}, \dots, \{a_n\})$

**\_\_hash\_\_** ()  
 Return hash(self).

**\_\_eq\_\_** (other)  
 Return self==value.

**\_\_repr\_\_** ()  
 Return repr(self).

**class** owlapy.class\_expression.**OWLRDFVocabulary** (  
     *namespace: owlapy.namespaces.Namespaces, remainder: str*)  
 Bases: *\_Vocabulary, enum.Enum*  
 Enumerations for OWL/RDF vocabulary.

**OWL\_THING**

**OWL\_NOTHING**

**OWL\_CLASS**

**OWL\_NAMED\_INDIVIDUAL**

**OWL\_TOP\_OBJECT\_PROPERTY**

**OWL\_BOTTOM\_OBJECT\_PROPERTY**

**OWL\_TOP\_DATA\_PROPERTY**

**OWL\_BOTTOM\_DATA\_PROPERTY**

**RDFS\_LITERAL**

owlapy.class\_expression.**OWLThing**: **Final**

owlapy.class\_expression.**OWLNothing**: **Final**

## owlapy.entities

Entities are the fundamental building blocks of OWL 2 ontologies, and they define the vocabulary — the named terms — of an ontology. In logic, the set of entities is usually said to constitute the signature of an ontology.

Classes, datatypes, object properties, data properties, annotation properties, and named individuals are entities, and they are all uniquely identified by an IR.

## 6.2 Submodules

### owlapy.converter

Format converter.

### Attributes

*TopOWLDatatype*

*converter*

### Classes

<i>OWLObjectHasValue</i>	A has-value class expression <code>ObjectHasValue( OPE a )</code> consists of an object property expression OPE and an
<i>OWLObjectOneOf</i>	An enumeration of individuals <code>ObjectOneOf( a1 ... an )</code> contains exactly the individuals $a_i$ with $1 \leq i \leq n$ .
<i>OWLDatatypeRestriction</i>	A datatype restriction <code>DatatypeRestriction( DT F1 lt1 ... Fn ltn )</code> consists of a unary datatype DT and n pairs
<i>OWLDataMinCardinality</i>	A minimum cardinality expression <code>DataMinCardinality( n DPE DR )</code> consists of a nonnegative integer n, a data
<i>OWLDataMaxCardinality</i>	A maximum cardinality expression <code>ObjectMaxCardinality( n OPE CE )</code> consists of a nonnegative integer n, an object
<i>OWLDataExactCardinality</i>	An exact cardinality expression <code>ObjectExactCardinality( n OPE CE )</code> consists of a nonnegative integer n, an
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLObjectIntersectionOf</i>	An intersection class expression <code>ObjectIntersectionOf( CE1 ... CEn )</code> contains all individuals that are instances
<i>OWLObjectUnionOf</i>	A union class expression <code>ObjectUnionOf( CE1 ... CEn )</code> contains all individuals that are instances
<i>OWLObjectComplementOf</i>	Represents an <code>ObjectComplementOf</code> class expression in the OWL 2 Specification.

continues on next page

Table 4 – continued from previous page

<i>OWLObjectSomeValuesFrom</i>	An existential class expression <i>ObjectSomeValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and
<i>OWLObjectAllValuesFrom</i>	A universal class expression <i>ObjectAllValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and a
<i>OWLObjectCardinalityRestriction</i>	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
<i>OWLObjectMinCardinality</i>	A minimum cardinality expression <i>ObjectMinCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLObjectMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLObjectExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLDataCardinalityRestriction</i>	Represents Data Property Cardinality Restrictions.
<i>OWLObjectHasSelf</i>	A self-restriction <i>ObjectHasSelf</i> ( OPE ) consists of an object property expression OPE,
<i>OWLDataSomeValuesFrom</i>	An existential class expression <i>DataSomeValuesFrom</i> ( DPE1 ... DPE <sub>n</sub> DR ) consists of n data property expressions
<i>OWLDataAllValuesFrom</i>	A universal class expression <i>DataAllValuesFrom</i> ( DPE1 ... DPE <sub>n</sub> DR ) consists of n data property expressions DPE <sub>i</sub> ,
<i>OWLDataHasValue</i>	A has-value class expression <i>DataHasValue</i> ( DPE It ) consists of a data property expression DPE and a literal It,
<i>OWLDataOneOf</i>	An enumeration of literals <i>DataOneOf</i> ( It1 ... It <sub>n</sub> ) contains exactly the explicitly specified literals It <sub>i</sub> with
<i>OWLNamedIndividual</i>	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
<i>OWLEntity</i>	Represents Entities in the OWL 2 Specification.
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
<i>OWLFacet</i>	Enumerations for OWL facets.
<i>OWLRDFVocabulary</i>	Enumerations for OWL/RDF vocabulary.
<i>VariablesMapping</i>	Helper class for owl-to-sparql conversion.
<i>Owl2SparqlConverter</i>	Convert owl (owlapy model class expressions) to SPARQL.

## Functions

<code>peek(x)</code>	Peek the last element of an array.
<code>owl_expression_to_sparql(→ str)</code>	Convert an OWL Class Expression ( <a href="https://www.w3.org/TR/owl2-syntax/#Class_Expressions">https://www.w3.org/TR/owl2-syntax/#Class_Expressions</a> ) into a SPARQL query

## Module Contents

```
class owlapy.converter.OwlObjectHasValue(  
    property: owlapy.owl_property.OwlObjectPropertyExpression,  
    individual: owlapy.owl_individual.OwlIndividual)  
  
Bases: OwlHasValueRestriction[owlapy.owl_individual.OwlIndividual], OwlObjectRestriction  
  
A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ). (https://www.w3.org/TR/owl2-syntax/#Individual\_Value\_Restriction)  
  
__slots__ = ('_property', '_v')  
  
type_index: Final = 3007  
  
get_property() → owlapy.owl_property.OwlObjectPropertyExpression  
  
    Returns  
        Property being restricted.  
  
as_some_values_from() → owlapy.class_expression.class_expression.OwlClassExpression  
    A convenience method that obtains this restriction as an existential restriction with a nominal filler.  
  
    Returns  
        The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .  
  
__repr__()   
    Return repr(self).  
  
class owlapy.converter.OwlObjectOneOf(  
    values: owlapy.owl_individual.OwlIndividual | Iterable[owlapy.owl_individual.OwlIndividual])  
  
Bases: owlapy.class_expression.class_expression.OwlAnonymousClassExpression, owlapy.meta_classes.HasOperands[owlapy.owl_individual.OwlIndividual]  
  
An enumeration of individuals ObjectOneOf( a1 ... an ) contains exactly the individuals ai with  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Enumeration\_of\_Individuals)  
  
__slots__ = '_values'  
  
type_index: Final = 3004  
  
individuals() → Iterable[owlapy.owl_individual.OwlIndividual]  
    Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this class expression.  
  
    Returns  
        The individuals that are the values of this {@code ObjectOneOf} class expression.
```



**operands** () → Iterable[*owlapy.owl\_individual.OWLIndividual*]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**as\_object\_union\_of** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

Simplifies this enumeration to a union of singleton nominals.

**Returns**

This enumeration in a more standard DL form.  $\text{simp}(\{a\}) = \{a\}$   $\text{simp}(\{a_0, \dots, \{a_n\}) = \text{unionOf}(\{a_0\}, \dots, \{a_n\})$

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

Return hash(self).

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

Return self==value.

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.converter.**OWLDatatypeRestriction** (type\_: *owlapy.owl\_datatype.OWLDatatype*,  
facet\_restrictions: *OWLFacetRestriction* | Iterable[*OWLFacetRestriction*])

Bases: *owlapy.owl\_data\_ranges.OWLDataRange*

A datatype restriction DatatypeRestriction( DT F1 lt1 ... Fn ltn ) consists of a unary datatype DT and n pairs ( Fi , lti ). The resulting data range is unary and is obtained by restricting the value space of DT according to the semantics of all ( Fi , vi ) (multiple pairs are interpreted conjunctively), where vi are the data values of the literals lti. ([https://www.w3.org/TR/owl2-syntax/#Datatype\\_Restrictions](https://www.w3.org/TR/owl2-syntax/#Datatype_Restrictions))

**\_\_slots\_\_** = ('\_type', '\_facet\_restrictions')

**type\_index**: Final = 4006

**get\_datatype** () → *owlapy.owl\_datatype.OWLDatatype*

**get\_facet\_restrictions** () → Sequence[*OWLFacetRestriction*]

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

Return self==value.

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

Return hash(self).

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.converter.**OWLDataMinCardinality** (cardinality: int,  
property: *owlapy.owl\_property.OWLDataPropertyExpression*,  
filler: *owlapy.owl\_data\_ranges.OWLDataRange*)

Bases: *OWLDataCardinalityRestriction*

A minimum cardinality expression DataMinCardinality( n DPE DR ) consists of a nonnegative integer n, a data property expression DPE, and a unary data range DR, and it contains all those individuals that are connected by DPE to at least n different literals in DR. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

**\_\_slots\_\_** = ('\_cardinality', '\_filler', '\_property')

**type\_index: Final = 3015**

```
class owlapy.converter.OWLDataMaxCardinality (cardinality: int,  
        property: owlapy.owl_property.OWLDataPropertyExpression,  
        filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: *OWLDataCardinalityRestriction*

A maximum cardinality expression *ObjectMaxCardinality*( *n* *OPE* *CE* ) consists of a nonnegative integer *n*, an object property expression *OPE*, and a class expression *CE*, and it contains all those individuals that are connected by *OPE* to at most *n* different individuals that are instances of *CE*. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

**type\_index: Final = 3017**

```
class owlapy.converter.OWLDataExactCardinality (cardinality: int,  
        property: owlapy.owl_property.OWLDataPropertyExpression,  
        filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: *OWLDataCardinalityRestriction*

An exact cardinality expression *ObjectExactCardinality*( *n* *OPE* *CE* ) consists of a nonnegative integer *n*, an object property expression *OPE*, and a class expression *CE*, and it contains all those individuals that are connected

by *OPE* to exactly *n* different individuals that are instances of *CE* ([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

**type\_index: Final = 3016**

```
as_intersection_of_min_max()
```

→ *owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

#### Returns

The semantically equivalent but structurally simpler form  $(= 1 \text{ R D}) \Rightarrow 1 \text{ R D}$  and  $\leq 1 \text{ R D}$ .

```
class owlapy.converter.OWLClass (iri: owlapy.iri.IRI | str)
```

Bases: *owlapy.class\_expression.class\_expression.OWLClassExpression*, *owlapy.owl\_object.OWLEntity*

An OWL 2 named Class. Classes can be understood as sets of individuals. (<https://www.w3.org/TR/owl2-syntax/#Classes>)

```
__slots__ = ('_iri', '_is_nothing', '_is_thing')
```

**type\_index: Final = 1001**

```
property iri: owlapy.iri.IRI
```

Gets the IRI of this object.

#### Returns

The IRI of this object.

```
property str
```

**Gets the string representation of this object**

#### Returns

The IRI as string

**property reminder: str**  
The reminder of the IRI

**is\_owl\_thing() → bool**  
Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**  
Thing.

**Return type**  
True if this expression is owl

**is\_owl\_nothing() → bool**  
Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**get\_object\_complement\_of()**  
→ *owlapy.class\_expression.class\_expression.OWLObjectComplementOf*  
Gets the object complement of this class expression.

**Returns**  
A class expression that is the complement of this class expression.

**get\_nnf() → OWLClass**  
Gets the negation normal form of the complement of this expression.

**Returns**  
A expression that represents the NNF of the complement of this expression.

**class owlapy.converter.OWLClassExpression**  
Bases: *owlapy.owl\_data\_ranges.OWLPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

**\_\_slots\_\_ = ()**

**abstract is\_owl\_thing() → bool**  
Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**  
Thing.

**Return type**  
True if this expression is owl

**abstract is\_owl\_nothing() → bool**  
Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of() → OWLObjectComplementOf**  
Gets the object complement of this class expression.

**Returns**  
A class expression that is the complement of this class expression.

**abstract** `get_nnf()` → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** `owlapy.converter.OwlObjectIntersectionOf` (  
    *operands: Iterable[owlapy.class\_expression.class\_expression.OwlClassExpression]*)

Bases: `OWLNaryBooleanClassExpression`

An intersection class expression `ObjectIntersectionOf( CE1 ... CEn )` contains all individuals that are instances of all class expressions `CEi` for  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Intersection\\_of\\_Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions))

`__slots__ = '_operands'`

`type_index: Final = 3001`

**class** `owlapy.converter.OwlObjectUnionOf` (  
    *operands: Iterable[owlapy.class\_expression.class\_expression.OwlClassExpression]*)

Bases: `OWLNaryBooleanClassExpression`

A union class expression `ObjectUnionOf( CE1 ... CEn )` contains all individuals that are instances of at least one class expression `CEi` for  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Union\\_of\\_Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions))

`__slots__ = '_operands'`

`type_index: Final = 3002`

**class** `owlapy.converter.OwlObjectComplementOf` (*op: OWLClassExpression*)

Bases: `OWLBooleanClassExpression`, `owlapy.meta_classes.HasOperands[OWLClassExpression]`

Represents an `ObjectComplementOf` class expression in the OWL 2 Specification.

`__slots__ = '_operand'`

`type_index: Final = 3003`

`get_operand()` → *OWLClassExpression*

**Returns**

The wrapped expression.

`operands()` → *Iterable[OWLClassExpression]*

Gets the operands - e.g., the individuals in a `sameAs` axiom, or the classes in an `equivalentClasses` axiom.

**Returns**

The operands.

`__repr__()`

Return `repr(self)`.

`__eq__(other)`

Return `self==value`.

`__hash__()`

Return `hash(self)`.

```
class owlapy.converter.OwlObjectSomeValuesFrom (
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: OwlQuantifiedObjectRestriction

An existential class expression ObjectSomeValuesFrom( OPE CE ) consists of an object property expression OPE and a class expression CE, and it contains all those individuals that are connected by OPE to an individual that is an instance of CE.

```
__slots__ = ('_property', '_filler')
```

```
type_index: Final = 3005
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
get_property() → owlapy.owl_property.OwlObjectPropertyExpression
```

**Returns**

Property being restricted.

```
class owlapy.converter.OwlObjectAllValuesFrom (
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: OwlQuantifiedObjectRestriction

A universal class expression ObjectAllValuesFrom( OPE CE ) consists of an object property expression OPE and a class expression CE, and it contains all those individuals that are connected by OPE only to individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification))

```
__slots__ = ('_property', '_filler')
```

```
type_index: Final = 3006
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
get_property() → owlapy.owl_property.OwlObjectPropertyExpression
```

**Returns**

Property being restricted.

```
class owlapy.converter.OwlObjectCardinalityRestriction (cardinality: int,
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: OwlCardinalityRestriction[owlapy.class\_expression.class\_expression.OwlClassExpression], OwlQuantifiedObjectRestriction

Represents Object Property Cardinality Restrictions in the OWL 2 specification.

```

__slots__ = ()

get_property() → owlapy.owl_property.OwlObjectPropertyExpression

    Returns
        Property being restricted.

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

class owlapy.converter.OwlObjectMinCardinality (cardinality: int,
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
    Bases: OwlObjectCardinalityRestriction

    A minimum cardinality expression ObjectMinCardinality( n OPE CE ) consists of a nonnegative integer n, an object
    property expression OPE, and a class expression CE, and it contains all those individuals that are connected by
    OPE to at least n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Minimum\_Cardinality)

    __slots__ = ('_cardinality', '_filler', '_property')

    type_index: Final = 3008

class owlapy.converter.OwlObjectMaxCardinality (cardinality: int,
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
    Bases: OwlObjectCardinalityRestriction

    A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an
    object property expression OPE, and a class expression CE, and it contains all those individuals that are connected
    by OPE

        to at most n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Maximum\_Cardinality)

    __slots__ = ('_cardinality', '_filler', '_property')

    type_index: Final = 3010

class owlapy.converter.OwlObjectExactCardinality (cardinality: int,
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
    Bases: OwlObjectCardinalityRestriction

    An exact cardinality expression ObjectExactCardinality( n OPE CE ) consists of a nonnegative integer n,
    an object
        property expression OPE, and a class expression CE, and it contains all those individuals that are connected
        by to exactly n different individuals that are instances of CE.

    (https://www.w3.org/TR/owl2-syntax/#Exact\_Cardinality)

    __slots__ = ('_cardinality', '_filler', '_property')

```

```

type_index: Final = 3009

as_intersection_of_min_max()
    → owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf
    Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

Returns
    The semantically equivalent but structurally simpler form ( $= 1 \text{ R C}$ )  $= \geq 1 \text{ R C}$  and  $\leq 1 \text{ R C}$ .

class owlapy.converter.OWLDataCardinalityRestriction (cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)
    Bases: OWLCardinalityRestriction[owlapy.owl_data_ranges.OWLDataRange],
    OWLQuantifiedDataRestriction, OWLDataRestriction
    Represents Data Property Cardinality Restrictions.

    __slots__ = ()

    get_property() → owlapy.owl_property.OWLDataPropertyExpression

Returns
    Property being restricted.

    __repr__()
        Return repr(self).

    __eq__ (other)
        Return self==value.

    __hash__()
        Return hash(self).

class owlapy.converter.OWLObjectHasSelf (
    property: owlapy.owl_property.OWLObjectPropertyExpression)
    Bases: OWLObjectRestriction
    A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE, and it contains all those
    individuals that are connected by OPE to themselves. (https://www.w3.org/TR/owl2-syntax/#Self-Restriction)

    __slots__ = '_property'

    type_index: Final = 3011

    get_property() → owlapy.owl_property.OWLObjectPropertyExpression

Returns
    Property being restricted.

    __eq__ (other)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

```

```
class owlapy.converter.OWLDataSomeValuesFrom (
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: OWLQuantifiedDataRestriction

An existential class expression `DataSomeValuesFrom( DPE1 ... DPEn DR )` consists of  $n$  data property expressions `DPEi`,  $1 \leq i \leq n$ , and a data range `DR` whose arity must be  $n$ . Such a class expression contains all those individuals that are connected by `DPEi` to literals `lti`,  $1 \leq i \leq n$ , such that the tuple `( lt1 , ..., ltn )` is in `DR`. A class expression of the form `DataSomeValuesFrom( DPE DR )` can be seen as a syntactic shortcut for the class expression `DataMinCardinality( 1 DPE DR )`. ([https://www.w3.org/TR/owl2-syntax/#Existential\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Existential_Quantification_2))

```
__slots__ = '_property'
```

```
type_index: Final = 3012
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
get_property() → owlapy.owl_property.OWLDataPropertyExpression
```

**Returns**

Property being restricted.

```
class owlapy.converter.OWLDataAllValuesFrom (
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: OWLQuantifiedDataRestriction

A universal class expression `DataAllValuesFrom( DPE1 ... DPEn DR )` consists of  $n$  data property expressions `DPEi`,  $1 \leq i \leq n$ , and a data range `DR` whose arity must be  $n$ . Such a class expression contains all those individuals that

**are connected by DPE<sub>i</sub> only to literals lt<sub>i</sub>,  $1 \leq i \leq n$ , such that each tuple `( lt1 , ..., ltn )` is in `DR`.**

**A class**

expression of the form `DataAllValuesFrom( DPE DR )` can be seen as a syntactic shortcut for the class expression `DataMaxCardinality( 0 DPE DataComplementOf( DR ) )`. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification_2))

```
__slots__ = '_property'
```

```
type_index: Final = 3013
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).



**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.converter.OWLDataHasValue (
    property: owlapy.owl_property.OWLDataPropertyExpression,
    value: owlapy.owl_literal.OWLLiteral)
```

Bases: *OWLHasValueRestriction[owlapy.owl\_literal.OWLLiteral]*, *OWLDataRestriction*

A has-value class expression *DataHasValue( DPE lt )* consists of a data property expression DPE and a literal lt, and it contains all those individuals that are connected by DPE to lt. Each such class expression can be seen as a syntactic shortcut for the class expression *DataSomeValuesFrom( DPE DataOneOf( lt ) )*. ([https://www.w3.org/TR/owl2-syntax/#Literal\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Literal_Value_Restriction))

```
__slots__ = '_property'
```

```
type_index: Final = 3014
```

```
__repr__ ()
```

Return repr(self).

```
__eq__ (other)
```

Return self==value.

```
__hash__ ()
```

Return hash(self).

**as\_some\_values\_from** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

**Returns**

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.converter.OWLDataOneOf (
    values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])
```

Bases: *owlapy.owl\_data\_ranges.OWLDataRange*, *owlapy.meta\_classes.HasOperands[owlapy.owl\_literal.OWLLiteral]*

An enumeration of literals *DataOneOf( lt1 ... ltn )* contains exactly the explicitly specified literals lti with  $1 \leq i \leq n$ . The resulting data range has arity one. ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Literals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals))

```
type_index: Final = 4003
```

```
values () → Iterable[owlapy.owl_literal.OWLLiteral]
```

Gets the values that are in the oneOf.

**Returns**

The values of this {`@code DataOneOf`} class expression.

```
operands () → Iterable[owlapy.owl_literal.OWLLiteral]
```

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

```

__hash__()
    Return hash(self).

__eq__(other)
    Return self==value.

__repr__()
    Return repr(self).

```

**class** owlapy.converter.Owlnamedindividual (iri: owlapy.iri.IRI | str)

Bases: Owlindividual, owlapy.owl\_object.OwlEntity

Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities. IRIs from the reserved vocabulary must not be used to identify named individuals in an OWL 2 DL ontology.

([https://www.w3.org/TR/owl2-syntax/#Named\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Named_Individuals))

```

__slots__ = '_iri'

type_index: Final = 1005

property iri: owlapy.iri.IRI
    Gets the IRI of this object.

    Returns
        The IRI of this object.

property str
    Gets the string representation of this object

    Returns
        The IRI as string

```

**class** owlapy.converter.OwlLiteral

Bases: owlapy.owl\_annotation.OwlAnnotationValue

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(<https://www.w3.org/TR/owl2-syntax/#Literals>)

```

__slots__ = ()

type_index: Final = 4008

get_literal() → str
    Gets the lexical value of this literal. Note that the language tag is not included.

    Returns
        The lexical value of this literal.

is_boolean() → bool
    Whether this literal is typed as boolean.

parse_boolean() → bool
    Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("http://www.w3.org/2001/XMLSchema#boolean").

    Returns
        A bool value that is represented by this literal.

```

**is\_double** () → bool

Whether this literal is typed as double.

**parse\_double** () → float

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

**Returns**

A double value that is represented by this literal.

**is\_integer** () → bool

Whether this literal is typed as integer.

**parse\_integer** () → int

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

**Returns**

An integer value that is represented by this literal.

**is\_string** () → bool

Whether this literal is typed as string.

**parse\_string** () → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("<http://www.w3.org/2001/XMLSchema#string>").

**Returns**

A string value that is represented by this literal.

**is\_date** () → bool

Whether this literal is typed as date.

**parse\_date** () → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("<http://www.w3.org/2001/XMLSchema#date>").

**Returns**

A date value that is represented by this literal.

**is\_datetime** () → bool

Whether this literal is typed as dateTime.

**parse\_datetime** () → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("<http://www.w3.org/2001/XMLSchema#dateTime>").

**Returns**

A datetime value that is represented by this literal.

**is\_duration** () → bool

Whether this literal is typed as duration.

**parse\_duration** () → pandas.Timedelta

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("<http://www.w3.org/2001/XMLSchema#duration>").

**Returns**

A Timedelta value that is represented by this literal.

**is\_literal()** → bool

**Returns**  
true if the annotation value is a literal

**as\_literal()** → *OWLLiteral*

**Returns**  
if the value is a literal, returns it. Return None otherwise

**to\_python()** → Literals

**abstract get\_datatype()** → *owlapy.owl\_datatype.OWLDatatype*  
Gets the OWLDatatype which types this literal.

**Returns**  
The OWLDatatype that types this literal.

owlapy.converter.**TopOWLDatatype**: **Final**

**class** owlapy.converter.**OWLObjectProperty** (*iri: owlapy.iri.IRI | str*)  
Bases: OWLObjectPropertyExpression, OWLProperty  
Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.  
([https://www.w3.org/TR/owl2-syntax/#Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Object_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index**: **Final** = 1002

**get\_named\_property()** → *OWLObjectProperty*  
Get the named object property used in this property expression.

**Returns**  
P if this expression is either inv(P) or P.

**get\_inverse\_property()** → *OWLObjectInverseOf*  
Obtains the property that corresponds to the inverse of this property.

**Returns**  
The inverse of this property. Note that this property will not necessarily be in the simplest form.

**is\_owl\_top\_object\_property()** → bool  
Determines if this is the owl:topObjectProperty.

**Returns**  
topObjectProperty.

**Return type**  
True if this property is the owl

**class** owlapy.converter.**OWLDataProperty** (*iri: owlapy.iri.IRI | str*)  
Bases: OWLDataPropertyExpression, OWLProperty  
Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.  
([https://www.w3.org/TR/owl2-syntax/#Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Data_Properties))

**\_\_slots\_\_** = '\_iri'

```

type_index: Final = 1004

is_owl_top_data_property() → bool
    Determines if this is the owl:topDataProperty.

    Returns
        topDataProperty.

    Return type
        True if this property is the owl

```

**class** owlapy.converter.OWLEntity

Bases: OWLNamedObject

Represents Entities in the OWL 2 Specification.

```

__slots__ = ()

to_string_id() → str

is_anonymous() → bool

```

**class** owlapy.converter.OWLDatatype(*iri: owlapy.iri.IRI | owlapy.meta\_classes.HasIRI*)

Bases: owlapy.owl\_object.OWLEntity, owlapy.owl\_data\_ranges.OWLDataRange

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

```

__slots__ = '_iri'

type_index: Final = 4001

property iri: owlapy.iri.IRI
    Gets the IRI of this object.

    Returns
        The IRI of this object.

property str: str
    Gets the string representation of this object

    Returns
        The IRI as string

```

**class** owlapy.converter.OWLFacet(*remainder: str, symbolic\_form: str, operator: Callable[[\_X, \_X], bool]*)

Bases: \_Vocabulary, enum.Enum

Enumerations for OWL facets.

```

property symbolic_form

property operator

static from_str(name: str) → OWLFacet

```

```

MIN_INCLUSIVE: Final
MIN_EXCLUSIVE: Final
MAX_INCLUSIVE: Final
MAX_EXCLUSIVE: Final
LENGTH: Final
MIN_LENGTH: Final
MAX_LENGTH: Final
PATTERN: Final
TOTAL_DIGITS: Final
FRACTION_DIGITS: Final

class owlapy.converter.OWL RDFVocabulary (namespace: owlapy.namespaces.Namespaces,
    remainder: str)
    Bases: _Vocabulary, enum.Enum
    Enumerations for OWL/RDF vocabulary.
    OWL_THING
    OWL_NOTHING
    OWL_CLASS
    OWL_NAMED_INDIVIDUAL
    OWL_TOP_OBJECT_PROPERTY
    OWL_BOTTOM_OBJECT_PROPERTY
    OWL_TOP_DATA_PROPERTY
    OWL_BOTTOM_DATA_PROPERTY
    RDFS_LITERAL

owlapy.converter.peek(x)
    Peek the last element of an array.

    Returns
        The last element arr[-1].

class owlapy.converter.VariablesMapping
    Helper class for owl-to-sparql conversion.
    __slots__ = ('class_cnt', 'prop_cnt', 'ind_cnt', 'dict')
    get_variable(e: owlapy.owl_object.OWLEntity) → str
    new_individual_variable() → str
    new_property_variable() → str

```

```

__contains__ (item: owlapy.owl_object.OWLEntity) → bool
__getitem__ (item: owlapy.owl_object.OWLEntity) → str

class owlapy.converter.Owl2SparqlConverter
    Convert owl (owlapy model class expressions) to SPARQL.

    __slots__ = ('ce', 'sparql', 'variables', 'parent', 'parent_var',
        'properties', 'variable_entities', 'cnt', ...)

    ce: owlapy.class_expression.OWLClassExpression
    sparql: List[str]
    variables: List[str]
    parent: List[owlapy.class_expression.OWLClassExpression]
    parent_var: List[str]
    variable_entities: Set[owlapy.owl_object.OWLEntity]
    properties: Dict[int, List[owlapy.owl_object.OWLEntity]]
    mapping: VariablesMapping
    grouping_vars: Dict[owlapy.class_expression.OWLClassExpression, Set[str]]
    having_conditions: Dict[owlapy.class_expression.OWLClassExpression,
        Set[str]]

    cnt: int
    for_all_de_morgan: bool
    named_individuals: bool

    convert (root_variable: str, ce: owlapy.class_expression.OWLClassExpression,
        for_all_de_morgan: bool = True, named_individuals: bool = False)
        Used to convert owl class expression to SPARQL syntax.

        Parameters
            • root_variable (str) – Root variable name that will be used in SPARQL query.
            • ce (OWLClassExpression) – The owl class expression to convert.
            • named_individuals (bool) – If ‘True’ return only entities that are instances of
              owl:NamedIndividual.

        Returns
            The SPARQL query.

        Return type
            list[str]

    property modal_depth

    abstract render (e)

    stack_variable (var)

```

```

stack_parent (parent: owlapy.class_expression.OWLClassExpression)

property current_variable

abstract process (ce: owlapy.class_expression.OWLClassExpression)

forAll (ce: owlapy.class_expression.OWLObjectAllValuesFrom)

forAllDeMorgan (ce: owlapy.class_expression.OWLObjectAllValuesFrom)

new_count_var () → str

append_triple (subject, predicate, object_)

append (frag)

triple (subject, predicate, object_)

as_query (root_variable: str, ce: owlapy.class_expression.OWLClassExpression,
           for_all_de_morgan: bool = True, count: bool = False,
           values: Iterable[owlapy.owl_individual.OWLNamedIndividual] | None = None,
           named_individuals: bool = False) → str

```

owlapy.converter.**converter**

owlapy.converter.**owl\_expression\_to\_sparql** (
 expression: owlapy.class\_expression.OWLClassExpression = None, root\_variable: str = '?x',
 values: Iterable[owlapy.owl\_individual.OWLNamedIndividual] | None = None,
 for\_all\_de\_morgan: bool = True, named\_individuals: bool = False) → str

Convert an OWL Class Expression ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions)) into a SPARQL query root variable: the variable that will be projected expression: the class expression to be transformed to a SPARQL query

values: positive or negative examples from a class expression problem. Unclear for\_all\_de\_morgan: if set to True, the SPARQL mapping will use the mapping containing the nested FILTER NOT EXISTS patterns for the universal quantifier ( $\neg(\exists r. \neg C)$ ), instead of the counting query named\_individuals: if set to True, the generated SPARQL query will return only entities that are instances of owl:NamedIndividual

## owlapy.iri

### OWL IRI

### Classes

<i>OWLAnnotationSubject</i>	A marker interface for annotation subjects, which can either be IRIs or anonymous individuals
<i>OWLAnnotationValue</i>	A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.
<i>Namespaces</i>	Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.



## Module Contents

**class** owlapy.iri.OWLAnnotationSubject

Bases: OWLAnnotationObject

A marker interface for annotation subjects, which can either be IRIs or anonymous individuals

**\_\_slots\_\_** = ()

**class** owlapy.iri.OWLAnnotationValue

Bases: OWLAnnotationObject

A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.

**\_\_slots\_\_** = ()

**is\_literal**() → bool

**Returns**

true if the annotation value is a literal

**as\_literal**() → *OWLLiteral* | None

**Returns**

if the value is a literal, returns it. Return None otherwise

**class** owlapy.iri.Namespaces (prefix: str, ns: str)

Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references

**\_\_slots\_\_** = ('\_prefix', '\_ns')

**property ns:** str

**property prefix:** str

**\_\_repr\_\_**()

Return repr(self).

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

Return hash(self).

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

Return self==value.

**class** owlapy.iri.IRI (namespace: str | owlapy.namespaces.Namespaces, remainder: str)

Bases: owlapy.owl\_annotation.OWLAnnotationSubject, owlapy.owl\_annotation.OWLAnnotationValue

An IRI, consisting of a namespace and a remainder.

**\_\_slots\_\_** = ('\_namespace', '\_remainder', '\_\_weakref\_\_')

**type\_index:** Final = 0

**static create**(namespace: owlapy.namespaces.Namespaces, remainder: str) → IRI

**static create**(namespace: str, remainder: str) → IRI

**static create**(string: str) → IRI

**\_\_repr\_\_()**

Return repr(self).

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

Return self==value.

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

Return hash(self).

**is\_nothing()**

Determines if this IRI is equal to the IRI that owl:Nothing is named with.

**Returns**

True if this IRI is equal to <<http://www.w3.org/2002/07/owl#Nothing>> and otherwise False.

**is\_thing()**

Determines if this IRI is equal to the IRI that owl:Thing is named with.

**Returns**

True if this IRI is equal to <<http://www.w3.org/2002/07/owl#Thing>> and otherwise False.

**is\_reserved\_vocabulary()** → bool

Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>> or <<http://www.w3.org/2000/01/rdf-schema#>> or <<http://www.w3.org/2001/XMLSchema#>> or <<http://www.w3.org/2002/07/owl#>>.

**Returns**

True if the IRI is in the reserved vocabulary, otherwise False.

**as\_iri()** → *IRI*

**Returns**

if the value is an IRI, return it. Return None otherwise.

**as\_str()** → str

CD: Should be deprecated. :returns: The string that specifies the IRI.

**property str:** str

Returns: The string that specifies the IRI.

**property reminder:** str

Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form()** → str

Gets the short form.

**Returns**

A string that represents the short form.

**get\_namespace()** → str

**Returns**

The namespace as string.

**get\_remainder()** → str

**Returns**

The remainder (coincident with NCName usually) for this IRI.

## owlapy.meta\_classes

Meta classes for OWL objects.

### Classes

<i>HasIRI</i>	Simple class to access the IRI.
<i>HasOperands</i>	An interface to objects that have a collection of operands.
<i>HasFiller</i>	An interface to objects that have a filler.
<i>HasCardinality</i>	An interface to objects that have a cardinality.

### Module Contents

**class** owlapy.meta\_classes.**HasIRI**

Simple class to access the IRI.

**\_\_slots\_\_** = ()

**property** iri: *IRI*

**Abstractmethod**

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property** str: **str**

**Abstractmethod**

Gets the string representation of this object

**Returns**

The IRI as string

**class** owlapy.meta\_classes.**HasOperands**

Bases: `Generic[_T]`

An interface to objects that have a collection of operands.

**Parameters**

**\_T** – Operand type.

**\_\_slots\_\_** = ()

**abstract** operands () → `Iterable[_T]`

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**class** owlapy.meta\_classes.**HasFiller**

Bases: `Generic[_T]`

An interface to objects that have a filler.

### Parameters

**\_T** – Filler type.

**\_\_slots\_\_** = ()

**abstract get\_filler()** → **\_T**

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

### Returns

the value

**class** owlapy.meta\_classes.**HasCardinality**

An interface to objects that have a cardinality.

**\_\_slots\_\_** = ()

**abstract get\_cardinality()** → int

Gets the cardinality of a restriction.

### Returns

The cardinality. A non-negative integer.

## owlapy.namespaces

Namespaces.

### Attributes

---

*OWL*

*RDFS*

*RDF*

*XSD*

---

### Classes

---

*Namespaces*

Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup

---

## Module Contents

**class** owlapy.namespaces.Namespaces (*prefix: str, ns: str*)

Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references

**\_\_slots\_\_** = ('\_prefix', '\_ns')

**property ns:** str

**property prefix:** str

**\_\_repr\_\_** ()

Return repr(self).

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

Return hash(self).

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

Return self==value.

owlapy.namespaces.OWL: **Final**

owlapy.namespaces.RDFS: **Final**

owlapy.namespaces.RDF: **Final**

owlapy.namespaces.XSD: **Final**

## owlapy.owl\_annotation

OWL Annotations

## Classes

<i>OWLObject</i>	Base interface for OWL objects
<i>OWLAnnotationObject</i>	A marker interface for the values (objects) of annotations.
<i>OWLAnnotationSubject</i>	A marker interface for annotation subjects, which can either be IRIs or anonymous individuals
<i>OWLAnnotationValue</i>	A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.

## Module Contents

**class** owlapy.owl\_annotation.OWLObject

Base interface for OWL objects

**\_\_slots\_\_** = ()

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

Return self==value.

```

abstract __hash__()
    Return hash(self).

abstract __repr__()
    Return repr(self).

is_anonymous() → bool

class owlapy.owl_annotation.OWLAnnotationObject
    Bases: owlapy.owl_object.OWLObject
    A marker interface for the values (objects) of annotations.

    __slots__ = ()

    as_iri() → IRI | None

        Returns
            if the value is an IRI, return it. Return None otherwise.

    as_anonymous_individual()

        Returns
            if the value is an anonymous, return it. Return None otherwise.

class owlapy.owl_annotation.OWLAnnotationSubject
    Bases: OWLAnnotationObject
    A marker interface for annotation subjects, which can either be IRIs or anonymous individuals

    __slots__ = ()

class owlapy.owl_annotation.OWLAnnotationValue
    Bases: OWLAnnotationObject
    A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.

    __slots__ = ()

    is_literal() → bool

        Returns
            true if the annotation value is a literal

    as_literal() → OWLLiteral | None

        Returns
            if the value is a literal, returns it. Return None otherwise

```

## owlapy.owl\_axiom

OWL Axioms

## Attributes

<i>OWLNothing</i>
<i>OWLThing</i>

## Classes

<i>OWLObjectPropertyExpression</i>	A high level interface to describe different types of object properties.
<i>OWLDataPropertyExpression</i>	A high level interface to describe different types of data properties.
<i>OWLObject</i>	Base interface for OWL objects
<i>OWLEntity</i>	Represents Entities in the OWL 2 Specification.
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
<i>OWLDatatype</i>	Represents a <i>Domain</i> in the OWL 2 Specification.
<i>HasOperands</i>	An interface to objects that have a collection of operands.
<i>OWLPropertyExpression</i>	Represents a property or possibly the inverse of a property.
<i>OWLProperty</i>	A base class for properties that aren't expression i.e. named properties. By definition, properties
<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>OWLObjectUnionOf</i>	A union class expression <i>ObjectUnionOf( CE1 ... CEn )</i> contains all individuals that are instances
<i>OWLIndividual</i>	Represents a named or anonymous individual.
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>OWLAnnotationSubject</i>	A marker interface for annotation subjects, which can either be IRIs or anonymous individuals
<i>OWLAnnotationValue</i>	A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.
<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
<i>OWLAxiom</i>	Represents Axioms in the OWL 2 Specification.
<i>OWLLogicalAxiom</i>	A base interface of all axioms that affect the logical meaning of an ontology. This excludes declaration
<i>OWLPropertyAxiom</i>	The base interface for property axioms.
<i>OWLObjectPropertyAxiom</i>	The base interface for object property axioms.
<i>OWLDataPropertyAxiom</i>	The base interface for data property axioms.
<i>OWLIndividualAxiom</i>	The base interface for individual axioms.
<i>OWLClassAxiom</i>	The base interface for class axioms.
<i>OWLDeclarationAxiom</i>	Represents a Declaration axiom in the OWL 2 Specification. A declaration axiom declares an entity in an ontology.

continues on next page

Table 5 – continued from previous page

<i>OWLDatatypeDefinitionAxiom</i>	A datatype definition <code>DatatypeDefinition( DT DR )</code> defines a new datatype DT as being semantically
<i>OWLHasKeyAxiom</i>	A key axiom <code>HasKey( CE ( OPE1 ... OPEm ) ( DPE1 ... DPEn ) )</code> states that each
<i>OWLNaryAxiom</i>	Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
<i>OWLNaryClassAxiom</i>	Represents an axiom that contains two or more operands that could also be represented with
<i>OWLEquivalentClassesAxiom</i>	An equivalent classes axiom <code>EquivalentClasses( CE1 ... CEn )</code> states that all of the class expressions CE <sub>i</sub> ,
<i>OWLDisjointClassesAxiom</i>	A disjoint classes axiom <code>DisjointClasses( CE1 ... CEn )</code> states that all of the class expressions CE <sub>i</sub> , $1 \leq i \leq n$ ,
<i>OWLNaryIndividualAxiom</i>	Represents an axiom that contains two or more operands that could also be represented with
<i>OWLDifferentIndividualsAxiom</i>	An individual inequality axiom <code>DifferentIndividuals( a1 ... an )</code> states that all of the individuals a <sub>i</sub> ,
<i>OWLSameIndividualAxiom</i>	An individual equality axiom <code>SameIndividual( a1 ... an )</code> states that all of the individuals a <sub>i</sub> , $1 \leq i \leq n$ ,
<i>OWLNaryPropertyAxiom</i>	Represents an axiom that contains two or more operands that could also be represented with
<i>OWLEquivalentObjectPropertiesAxiom</i>	An equivalent object properties axiom <code>EquivalentObjectProperties( OPE1 ... OPEn )</code> states that all of the object
<i>OWLDisjointObjectPropertiesAxiom</i>	A disjoint object properties axiom <code>DisjointObjectProperties( OPE1 ... OPEn )</code> states that all of the object
<i>OWLInverseObjectPropertiesAxiom</i>	An inverse object properties axiom <code>InverseObjectProperties( OPE1 OPE2 )</code> states that the object property
<i>OWLEquivalentDataPropertiesAxiom</i>	An equivalent data properties axiom <code>EquivalentDataProperties( DPE1 ... DPEn )</code> states that all the data property
<i>OWLDisjointDataPropertiesAxiom</i>	A disjoint data properties axiom <code>DisjointDataProperties( DPE1 ... DPEn )</code> states that all of the data property
<i>OWLSubClassOfAxiom</i>	A subclass axiom <code>SubClassOf( CE1 CE2 )</code> states that the class expression CE <sub>1</sub> is a subclass of the class
<i>OWLDisjointUnionAxiom</i>	A disjoint union axiom <code>DisjointUnion( C CE1 ... CEn )</code> states that a class C is a disjoint union of the class
<i>OWLClassAssertionAxiom</i>	A class assertion <code>ClassAssertion( CE a )</code> states that the individual a is an instance of the class expression CE.
<i>OWLAnnotationProperty</i>	Represents an <code>AnnotationProperty</code> in the OWL 2 specification.
<i>OWLAnnotation</i>	Annotations are used in the various types of annotation axioms, which bind annotations to their subjects
<i>OWLAnnotationAxiom</i>	A super interface for annotation axioms.
<i>OWLAnnotationAssertionAxiom</i>	An annotation assertion <code>AnnotationAssertion( AP as av )</code> states that the annotation subject as — an IRI or an
<i>OWLSubAnnotationPropertyOfAxiom</i>	An annotation subproperty axiom <code>SubAnnotationPropertyOf( AP1 AP2 )</code> states that the annotation property AP <sub>1</sub> is
<i>OWLAnnotationPropertyDomainAxiom</i>	An annotation property domain axiom <code>AnnotationPropertyDomain( AP U )</code> states that the domain of the annotation
<i>OWLAnnotationPropertyRangeAxiom</i>	An annotation property range axiom <code>AnnotationPropertyRange( AP U )</code>
<i>OWLSubPropertyAxiom</i>	Base interface for object and data sub-property axioms.

continues on next page



Table 5 – continued from previous page

<i>OWLSubObjectPropertyOfAxiom</i>	Object subproperty axioms are analogous to subclass axioms, and they come in two forms.
<i>OWLSubDataPropertyOfAxiom</i>	A data subproperty axiom <i>SubDataPropertyOf( DPE1 DPE2 )</i> states that the data property expression DPE1 is a
<i>OWLPropertyAssertionAxiom</i>	Base class for Property Assertion axioms.
<i>OWLObjectPropertyAssertionAxiom</i>	A positive object property assertion <i>ObjectPropertyAssertion( OPE a1 a2 )</i> states that the individual a1 is
<i>OWLNegativeObjectPropertyAssertionAxiom</i>	A negative object property assertion <i>NegativeObjectPropertyAssertion( OPE a1 a2 )</i> states that the individual a1
<i>OWLDataPropertyAssertionAxiom</i>	A positive data property assertion <i>DataPropertyAssertion( DPE a lt )</i> states that the individual a is connected
<i>OWLNegativeDataPropertyAssertionAxiom</i>	A negative data property assertion <i>NegativeDataPropertyAssertion( DPE a lt )</i> states that the individual a is not
<i>OWLUnaryPropertyAxiom</i>	Base class for Unary property axiom.
<i>OWLObjectPropertyCharacteristicAxiom</i>	Base interface for functional object property axiom.
<i>OWLFunctionalObjectPropertyAxiom</i>	An object property functionality axiom <i>FunctionalObjectProperty( OPE )</i> states that
<i>OWLAsymmetricObjectPropertyAxiom</i>	An object property asymmetry axiom <i>AsymmetricObjectProperty( OPE )</i> states that
<i>OWLInverseFunctionalObjectPropertyAxiom</i>	An object property inverse functionality axiom <i>InverseFunctionalObjectProperty( OPE )</i>
<i>OWLIrreflexiveObjectPropertyAxiom</i>	An object property irreflexivity axiom <i>IrreflexiveObjectProperty( OPE )</i> states that the
<i>OWLReflexiveObjectPropertyAxiom</i>	An object property reflexivity axiom <i>ReflexiveObjectProperty( OPE )</i> states that the
<i>OWLSymmetricObjectPropertyAxiom</i>	An object property symmetry axiom <i>SymmetricObjectProperty( OPE )</i> states that
<i>OWLTransitiveObjectPropertyAxiom</i>	An object property transitivity axiom <i>TransitiveObjectProperty( OPE )</i> states that the
<i>OWLDataPropertyCharacteristicAxiom</i>	Base interface for Functional data property axiom.
<i>OWLFunctionalDataPropertyAxiom</i>	A data property functionality axiom <i>FunctionalDataProperty( DPE )</i> states that
<i>OWLPropertyDomainAxiom</i>	Base class for Property Domain axioms.
<i>OWLPropertyRangeAxiom</i>	Base class for Property Range axioms.
<i>OWLObjectPropertyDomainAxiom</i>	An object property domain axiom <i>ObjectPropertyDomain( OPE CE )</i> states that the domain of the
<i>OWLDataPropertyDomainAxiom</i>	A data property domain axiom <i>DataPropertyDomain( DPE CE )</i> states that the domain of the
<i>OWLObjectPropertyRangeAxiom</i>	An object property range axiom <i>ObjectPropertyRange( OPE CE )</i> states that the range of the object property
<i>OWLDataPropertyRangeAxiom</i>	A data property range axiom <i>DataPropertyRange( DPE DR )</i> states that the range of the data property

## Module Contents

**class** owlapy.owl\_axiom.OWLDataPropertyExpression

Bases: *OWLPropertyExpression*

A high level interface to describe different types of data properties.

**\_\_slots\_\_** = ()

**is\_data\_property\_expression**()

### Returns

True if this is a data property.

**class** owlapy.owl\_axiom.OWLObjectPropertyExpression

Bases: *OWLPropertyExpression*

A high level interface to describe different types of object properties.

**\_\_slots\_\_** = ()

**abstract get\_inverse\_property**() → *OWLObjectPropertyExpression*

Obtains the property that corresponds to the inverse of this property.

### Returns

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**abstract get\_named\_property**() → *OWLObjectProperty*

Get the named object property used in this property expression.

### Returns

P if this expression is either inv(P) or P.

**is\_object\_property\_expression**() → bool

### Returns

True if this is an object property.

**class** owlapy.owl\_axiom.OWLObject

Base interface for OWL objects

**\_\_slots\_\_** = ()

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

Return self==value.

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

Return hash(self).

**abstract \_\_repr\_\_**()

Return repr(self).

**is\_anonymous**() → bool

**class** owlapy.owl\_axiom.OWLEntity

Bases: *OWLNamedObject*

Represents Entities in the OWL 2 Specification.

**\_\_slots\_\_** = ()

```

to_string_id() → str

is_anonymous() → bool

```

**class** owlapy.owl\_axiom.**OWLDatatype** (*iri: owlapy.iri.IRI | owlapy.meta\_classes.HasIRI*)

Bases: *owlapy.owl\_object.OWLEntity, owlapy.owl\_data\_ranges.OWLDataRange*

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

```

__slots__ = '_iri'

type_index: Final = 4001

property iri: owlapy.iri.IRI
    Gets the IRI of this object.

    Returns
        The IRI of this object.

property str: str
    Gets the string representation of this object

    Returns
        The IRI as string

```

**class** owlapy.owl\_axiom.**OWLDataRange**

Bases: *OWLPropertyRange*

Represents a DataRange in the OWL 2 Specification.

**class** owlapy.owl\_axiom.**HasOperands**

Bases: *Generic[\_T]*

An interface to objects that have a collection of operands.

```

    Parameters
        _T – Operand type.

__slots__ = ()

abstract operands() → Iterable[_T]
    Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

    Returns
        The operands.

```

**class** owlapy.owl\_axiom.**OWLPropertyExpression**

Bases: *owlapy.owl\_object.OWLObject*

Represents a property or possibly the inverse of a property.

```

__slots__ = ()

```

**is\_data\_property\_expression** () → bool

**Returns**

True if this is a data property.

**is\_object\_property\_expression** () → bool

**Returns**

True if this is an object property.

**is\_owl\_top\_object\_property** () → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

**is\_owl\_top\_data\_property** () → bool

Determines if this is the owl:topDataProperty.

**Returns**

topDataProperty.

**Return type**

True if this property is the owl

**class** owlapy.owl\_axiom.OWLProperty (iri: owlapy.iri.IRI | str)

Bases: *OWLPropertyExpression*, *owlapy.owl\_object.OWLEntity*

A base class for properties that aren't expression i.e. named properties. By definition, properties are either data properties or object properties.

**\_\_slots\_\_** = '\_iri'

**property** str: str

Gets the string representation of this object

**Returns**

The IRI as string

**property** iri: owlapy.iri.IRI

Gets the IRI of this object.

**Returns**

The IRI of this object.

**class** owlapy.owl\_axiom.OWLClassExpression

Bases: *owlapy.owl\_data\_ranges.OWLPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

**\_\_slots\_\_** = ()

**abstract** is\_owl\_thing () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**abstract is\_owl\_nothing** () → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of** () → *OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**abstract get\_nnf** () → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

An expression that represents the NNF of the complement of this expression.

**class** owlapy.owl\_axiom.OWLClass (iri: *owlapy.iri.IRI* | str)

Bases: *owlapy.class\_expression.class\_expression.OWLClassExpression*, *owlapy.owl\_object.OWLEntity*

An OWL 2 named Class. Classes can be understood as sets of individuals. (<https://www.w3.org/TR/owl2-syntax/#Classes>)

**\_\_slots\_\_** = ('\_iri', '\_is\_nothing', '\_is\_thing')

**type\_index**: Final = 1001

**property iri**: *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str**

**Gets the string representation of this object**

**Returns**

The IRI as string

**property reminder**: str

The reminder of the IRI

**is\_owl\_thing** () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**is\_owl\_nothing()** → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**get\_object\_complement\_of()**

→ *owlapy.class\_expression.class\_expression.OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf()** → *OWLClass*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

`owlapy.owl_axiom.OWLNothing: Final`

`owlapy.owl_axiom.OWLThing: Final`

**class** `owlapy.owl_axiom.OWLObjectUnionOf` (

*operands: Iterable[owlapy.class\_expression.class\_expression.OWLClassExpression]*)

Bases: `OWLNaryBooleanClassExpression`

A union class expression `ObjectUnionOf( CE1 ... CEn )` contains all individuals that are instances of at least one class expression `CEi` for  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Union\\_of\\_Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions))

**\_\_slots\_\_** = `'_operands'`

**type\_index: Final** = 3002

**class** `owlapy.owl_axiom.OWLIndividual`

Bases: *owlapy.owl\_object.OWLObject*

Represents a named or anonymous individual.

**\_\_slots\_\_** = `()`

**class** `owlapy.owl_axiom.IRI` (*namespace: str | owlapy.namespaces.Namespaces, remainder: str*)

Bases: *owlapy.owl\_annotation.OWLAnnotationSubject*, *owlapy.owl\_annotation.OWLAnnotationValue*

An IRI, consisting of a namespace and a remainder.

**\_\_slots\_\_** = `('_namespace', '_remainder', '__weakref__')`

**type\_index: Final** = 0

**static create** (*namespace: owlapy.namespaces.Namespaces, remainder: str*) → *IRI*

**static create** (*namespace: str, remainder: str*) → *IRI*

**static create** (*string: str*) → *IRI*

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

**\_\_hash\_\_()**  
Return hash(self).

**is\_nothing()**  
Determines if this IRI is equal to the IRI that owl:Nothing is named with.

**Returns**  
True if this IRI is equal to <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.

**is\_thing()**  
Determines if this IRI is equal to the IRI that owl:Thing is named with.

**Returns**  
True if this IRI is equal to <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

**is\_reserved\_vocabulary()** → bool  
Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

**Returns**  
True if the IRI is in the reserved vocabulary, otherwise False.

**as\_iri()** → *IRI*  
**Returns**  
if the value is an IRI, return it. Return None otherwise.

**as\_str()** → str  
CD: Should be deprecated. :returns: The string that specifies the IRI.

**property str: str**  
Returns: The string that specifies the IRI.

**property reminder: str**  
Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form()** → str  
Gets the short form.

**Returns**  
A string that represents the short form.

**get\_namespace()** → str  
**Returns**  
The namespace as string.

**get\_remainder()** → str  
**Returns**  
The remainder (coincident with NCName usually) for this IRI.

**class owlapy.owl\_axiom.OWLAnnotationSubject**  
Bases: `OWLAnnotationObject`  
A marker interface for annotation subjects, which can either be IRIs or anonymous individuals

**\_\_slots\_\_ = ()**

**class** owlapy.owl\_axiom.OWLAnnotationValue

Bases: OWLAnnotationObject

A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.

**\_\_slots\_\_** = ()

**is\_literal**() → bool

**Returns**

true if the annotation value is a literal

**as\_literal**() → *OWLLiteral* | None

**Returns**

if the value is a literal, returns it. Return None otherwise

**class** owlapy.owl\_axiom.OWLLiteral

Bases: *owlapy.owl\_annotation.OWLAnnotationValue*

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(<https://www.w3.org/TR/owl2-syntax/#Literals>)

**\_\_slots\_\_** = ()

**type\_index**: Final = 4008

**get\_literal**() → str

Gets the lexical value of this literal. Note that the language tag is not included.

**Returns**

The lexical value of this literal.

**is\_boolean**() → bool

Whether this literal is typed as boolean.

**parse\_boolean**() → bool

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("<http://www.w3.org/2001/XMLSchema#boolean>").

**Returns**

A bool value that is represented by this literal.

**is\_double**() → bool

Whether this literal is typed as double.

**parse\_double**() → float

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

**Returns**

A double value that is represented by this literal.

**is\_integer**() → bool

Whether this literal is typed as integer.



**parse\_integer()** → int

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

**Returns**

An integer value that is represented by this literal.

**is\_string()** → bool

Whether this literal is typed as string.

**parse\_string()** → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("<http://www.w3.org/2001/XMLSchema#string>").

**Returns**

A string value that is represented by this literal.

**is\_date()** → bool

Whether this literal is typed as date.

**parse\_date()** → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("<http://www.w3.org/2001/XMLSchema#date>").

**Returns**

A date value that is represented by this literal.

**is\_datetime()** → bool

Whether this literal is typed as dateTime.

**parse\_datetime()** → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("<http://www.w3.org/2001/XMLSchema#dateTime>").

**Returns**

A datetime value that is represented by this literal.

**is\_duration()** → bool

Whether this literal is typed as duration.

**parse\_duration()** → pandas.Timedelta

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("<http://www.w3.org/2001/XMLSchema#duration>").

**Returns**

A Timedelta value that is represented by this literal.

**is\_literal()** → bool

**Returns**

true if the annotation value is a literal

**as\_literal()** → *OWLLiteral*

**Returns**

if the value is a literal, returns it. Return None otherwise

**to\_python()** → Literals

**abstract** `get_datatype()` → *owlapy.owl\_datatype.OWLDatatype*

Gets the OWLDatatype which types this literal.

**Returns**

The OWLDatatype that types this literal.

**class** `owlapy.owl_axiom.OWLXiom` (*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *owlapy.owl\_object.OWLObject*

Represents Axioms in the OWL 2 Specification.

An OWL ontology contains a set of axioms. These axioms can be annotation axioms, declaration axioms, imports axioms or logical axioms.

**\_\_slots\_\_** = **'\_annotations'**

**annotations** () → List[*OWLAnnotation*] | None

**is\_annotated** () → bool

**is\_logical\_axiom** () → bool

**is\_annotation\_axiom** () → bool

**class** `owlapy.owl_axiom.OWLLogicalXiom` (*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLXiom*

A base interface of all axioms that affect the logical meaning of an ontology. This excludes declaration axioms (including imports declarations) and annotation axioms.

**\_\_slots\_\_** = **()**

**is\_logical\_axiom** () → bool

**class** `owlapy.owl_axiom.OWLPropertyXiom` (*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLLogicalXiom*

The base interface for property axioms.

**\_\_slots\_\_** = **()**

**class** `owlapy.owl_axiom.OWLObjectPropertyXiom` (*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyXiom*

The base interface for object property axioms.

**\_\_slots\_\_** = **()**

**class** `owlapy.owl_axiom.OWLDataPropertyXiom` (*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyXiom*

The base interface for data property axioms.

**\_\_slots\_\_** = **()**

```

class owlapy.owl_axiom.OWLIndividualAxiom(
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLLogicalAxiom
    The base interface for individual axioms.
    __slots__ = ()

class owlapy.owl_axiom.OWLClassAxiom(annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLLogicalAxiom
    The base interface for class axioms.
    __slots__ = ()

class owlapy.owl_axiom.OWLDeclarationAxiom(entity: owlapy.owl_object.OWLEntity,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLAxiom
    Represents a Declaration axiom in the OWL 2 Specification. A declaration axiom declares an entity in an ontology.
    It doesn't affect the logical meaning of the ontology.
    __slots__ = '_entity'
    get_entity() → owlapy.owl_object.OWLEntity
    __eq__(other)
        Return self==value.
    __hash__()
        Return hash(self).
    __repr__()
        Return repr(self).

class owlapy.owl_axiom.OWLDatatypeDefinitionAxiom(
    datatype: owlapy.owl_datatype.OWLDatatype, datarange: owlapy.owl_datatype.OWLDataRange,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLLogicalAxiom
    A datatype definition DatatypeDefinition( DT DR ) defines a new datatype DT as being semantically equivalent to
    the data range DR; the latter must be a unary data range. This axiom allows one to use the defined datatype DT as
    a synonym for DR — that is, in any expression in the ontology containing such an axiom, DT can be replaced with
    DR without affecting the meaning of the ontology.
    (https://www.w3.org/TR/owl2-syntax/#Datatype\_Definitions)
    __slots__ = ('_datatype', '_datarange')
    get_datatype() → owlapy.owl_datatype.OWLDatatype
    get_datarange() → owlapy.owl_datatype.OWLDataRange
    __eq__(other)
        Return self==value.
    __hash__()
        Return hash(self).

```

**\_\_repr\_\_**()

Return repr(self).

```
class owlapy.owl_axiom.OWLHasKeyAxiom(  
    class_expression: owlapy.class_expression.OWLClassExpression,  
    property_expressions: List[owlapy.owl_property.OWLPropertyExpression],  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLLogicalAxiom*, *owlapy.meta\_classes.HasOperands[owlapy.owl\_property.OWLPropertyExpression]*

A key axiom `HasKey( CE ( OPE1 ... OPEm ) ( DPE1 ... DPEn ) )` states that each (named) instance of the class expression CE is uniquely identified by the object property expressions OPEi and/or the data property expressions DPEj — that is, no two distinct (named) instances of CE can coincide on the values of all object property expressions OPEi and all data property expressions DPEj. In each such axiom in an OWL ontology, m or n (or both) must be larger than zero. A key axiom of the form `HasKey( owl:Thing ( OPE ) ( ) )` is similar to the axiom `InverseFunctionalObjectProperty( OPE )`, the main differences being that the former axiom is applicable only to individuals that are explicitly named in an ontology, while the latter axiom is also applicable to anonymous individuals and individuals whose existence is implied by existential quantification.

(<https://www.w3.org/TR/owl2-syntax/#Keys>)

**\_\_slots\_\_** = ('\_class\_expression', '\_property\_expressions')

**get\_class\_expression**() → *owlapy.class\_expression.OWLClassExpression*

**get\_property\_expressions**() → *List[owlapy.owl\_property.OWLPropertyExpression]*

**operands**() → *Iterable[owlapy.owl\_property.OWLPropertyExpression]*

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

#### Returns

The operands.

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

Return self==value.

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

Return hash(self).

**\_\_repr\_\_**()

Return repr(self).

```
class owlapy.owl_axiom.OWLNaryAxiom(annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *Generic[\_C]*, *OWLAxiom*

Represents an axiom that contains two or more operands that could also be represented with multiple pairwise axioms.

#### Parameters

**\_C** – Class of contained objects.

**\_\_slots\_\_** = ()

**abstract as\_pairwise\_axioms**() → *Iterable[OWLNaryAxiom[\_C]]*

```
class owlapy.owl_axiom.OWLNaryClassAxiom(  
    class_expressions: List[owlapy.class_expression.OWLClassExpression],  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLClassAxiom*, *OWLNaryAxiom*[*owlapy.class\_expression.OWLClassExpression*]

Represents an axiom that contains two or more operands that could also be represented with multiple pairwise axioms.

**\_\_slots\_\_** = **'\_class\_expressions'**

**class\_expressions** () → Iterable[*owlapy.class\_expression.OWLClassExpression*]

Gets all of the top level class expressions that appear in this axiom.

**Returns**

Sorted stream of class expressions that appear in the axiom.

**as\_pairwise\_axioms** () → Iterable[*OWLNaryClassAxiom*]

Gets this axiom as a set of pairwise axioms; if the axiom contains only two operands, the axiom itself is returned unchanged, including its annotations.

**Returns**

This axiom as a set of pairwise axioms.

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

Return self==value.

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

Return hash(self).

**\_\_repr\_\_** ()

Return repr(self).

```
class owlapy.owl_axiom.OWLEquivalentClassesAxiom (
    class_expressions: List[owlapy.class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLNaryClassAxiom*

An equivalent classes axiom *EquivalentClasses*( *CE*<sub>1</sub> ... *CE*<sub>*n*</sub> ) states that all of the class expressions *CE*<sub>*i*</sub>, 1 ≤ *i* ≤ *n*, are semantically equivalent to each other. This axiom allows one to use each *CE*<sub>*i*</sub> as a synonym for each *CE*<sub>*j*</sub> — that is, in any expression in the ontology containing such an axiom, *CE*<sub>*i*</sub> can be replaced with *CE*<sub>*j*</sub> without affecting the meaning of the ontology.

([https://www.w3.org/TR/owl2-syntax/#Equivalent\\_Classes](https://www.w3.org/TR/owl2-syntax/#Equivalent_Classes))

**\_\_slots\_\_** = ()

**contains\_named\_equivalent\_class** () → bool

**contains\_owl\_nothing** () → bool

**contains\_owl\_thing** () → bool

**named\_classes** () → Iterable[*owlapy.class\_expression.OWLClass*]

```
class owlapy.owl_axiom.OWLDisjointClassesAxiom (
    class_expressions: List[owlapy.class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLNaryClassAxiom*

A disjoint classes axiom *DisjointClasses*( *CE*<sub>1</sub> ... *CE*<sub>*n*</sub> ) states that all of the class expressions *CE*<sub>*i*</sub>, 1 ≤ *i* ≤ *n*, are pairwise disjoint; that is, no individual can be at the same time an instance of both *CE*<sub>*i*</sub> and *CE*<sub>*j*</sub> for *i* ≠ *j*.

([https://www.w3.org/TR/owl2-syntax/#Disjoint\\_Classes](https://www.w3.org/TR/owl2-syntax/#Disjoint_Classes))

```

__slots__ = ()

class owlapy.owl_axiom.OWLNaryIndividualAxiom(
    individuals: List[owlapy.owl_individual.OWLIndividual],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLIndividualAxiom, OWLNaryAxiom[owlapy.owl_individual.OWLIndividual]

    Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
    individual axioms.

    __slots__ = '_individuals'

    individuals() → Iterable[owlapy.owl_individual.OWLIndividual]
        Get the individuals.

        Returns
            Generator containing the individuals.

    as_pairwise_axioms() → Iterable[OWLNaryIndividualAxiom]

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

class owlapy.owl_axiom.OWLDifferentIndividualsAxiom(
    individuals: List[owlapy.owl_individual.OWLIndividual],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryIndividualAxiom

    An individual inequality axiom DifferentIndividuals( a1 ... an ) states that all of the individuals ai, 1 ≤ i ≤ n, are
    different from each other; that is, no individuals ai and aj with i ≠ j can be derived to be equal. This axiom can
    be used to axiomatize the unique name assumption — the assumption that all different individual names denote
    different individuals. (https://www.w3.org/TR/owl2-syntax/#Individual\_Inequality)

    __slots__ = ()

class owlapy.owl_axiom.OWLSameIndividualAxiom(
    individuals: List[owlapy.owl_individual.OWLIndividual],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryIndividualAxiom

    An individual equality axiom SameIndividual( a1 ... an ) states that all of the individuals ai, 1 ≤ i ≤ n, are equal
    to each other. This axiom allows one to use each ai as a synonym for each aj — that is, in any expression in the
    ontology containing such an axiom, ai can be replaced with aj without affecting the meaning of the ontology.

    (https://www.w3.org/TR/owl2-syntax/#Individual\_Equality)

    __slots__ = ()

class owlapy.owl_axiom.OWLNaryPropertyAxiom(properties: List[_P],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: Generic[_P], OWLPropertyAxiom, OWLNaryAxiom[_P]

    Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
    property axioms.

```

```

__slots__ = '_properties'

properties () → Iterable[_P]
    Get all the properties that appear in the axiom.

    Returns
        Generator containing the properties.

as_pairwise_axioms () → Iterable[OWLNaryPropertyAxiom]

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

__repr__ ()
    Return repr(self).

```

**class** owlapy.owl\_axiom.**OWLEquivalentObjectPropertiesAxiom**(  
     *properties: List[owlapy.owl\_property.OWLObjectPropertyExpression],*  
     *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLNaryPropertyAxiom[owlapy.owl\_property.OWLObjectPropertyExpression], OWLObjectPropertyAxiom*

An equivalent object properties axiom *EquivalentObjectProperties*( OPE1 ... OPE<sub>n</sub> ) states that all of the object property expressions OPE<sub>i</sub>, 1 ≤ i ≤ n, are semantically equivalent to each other. This axiom allows one to use each OPE<sub>i</sub> as a synonym for each OPE<sub>j</sub> — that is, in any expression in the ontology containing such an axiom, OPE<sub>i</sub> can be replaced with OPE<sub>j</sub> without affecting the meaning of the ontology.

([https://www.w3.org/TR/owl2-syntax/#Equivalent\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Equivalent_Object_Properties))

```

__slots__ = ()

```

**class** owlapy.owl\_axiom.**OWLDisjointObjectPropertiesAxiom**(  
     *properties: List[owlapy.owl\_property.OWLObjectPropertyExpression],*  
     *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLNaryPropertyAxiom[owlapy.owl\_property.OWLObjectPropertyExpression], OWLObjectPropertyAxiom*

A disjoint object properties axiom *DisjointObjectProperties*( OPE1 ... OPE<sub>n</sub> ) states that all of the object property expressions OPE<sub>i</sub>, 1 ≤ i ≤ n, are pairwise disjoint; that is, no individual x can be connected to an individual y by both OPE<sub>i</sub> and OPE<sub>j</sub> for i ≠ j.

([https://www.w3.org/TR/owl2-syntax/#Disjoint\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Disjoint_Object_Properties))

```

__slots__ = ()

```

**class** owlapy.owl\_axiom.**OWLInverseObjectPropertiesAxiom**(  
     *first: owlapy.owl\_property.OWLObjectPropertyExpression,*  
     *second: owlapy.owl\_property.OWLObjectPropertyExpression,*  
     *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLNaryPropertyAxiom[owlapy.owl\_property.OWLObjectPropertyExpression], OWLObjectPropertyAxiom*

An inverse object properties axiom *InverseObjectProperties*( OPE1 OPE2 ) states that the object property expression OPE1 is an inverse of the object property expression OPE2. Thus, if an individual x is connected by OPE1 to an individual y, then y is also connected by OPE2 to x, and vice versa.

([https://www.w3.org/TR/owl2-syntax/#Inverse\\_Object\\_Properties\\_2](https://www.w3.org/TR/owl2-syntax/#Inverse_Object_Properties_2))

```

__slots__ = ('_first', '_second')

get_first_property() → owlapy.owl_property.OWLObjectPropertyExpression

get_second_property() → owlapy.owl_property.OWLObjectPropertyExpression

__repr__()
    Return repr(self).

```

**class** owlapy.owl\_axiom.**OWLEquivalentDataPropertiesAxiom**(  
*properties: List[owlapy.owl\_property.OWLDataPropertyExpression],*  
*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLNaryPropertyAxiom[owlapy.owl\_property.OWLDataPropertyExpression], OWLDataPropertyAxiom*

An equivalent data properties axiom `EquivalentDataProperties( DPE1 ... DPEn )` states that all the data property expressions `DPEi`,  $1 \leq i \leq n$ , are semantically equivalent to each other. This axiom allows one to use each `DPEi` as a synonym for each `DPEj` — that is, in any expression in the ontology containing such an axiom, `DPEi` can be replaced with `DPEj` without affecting the meaning of the ontology.

([https://www.w3.org/TR/owl2-syntax/#Equivalent\\_Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Equivalent_Data_Properties))

```

__slots__ = ()

```

**class** owlapy.owl\_axiom.**OWLDisjointDataPropertiesAxiom**(  
*properties: List[owlapy.owl\_property.OWLDataPropertyExpression],*  
*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLNaryPropertyAxiom[owlapy.owl\_property.OWLDataPropertyExpression], OWLDataPropertyAxiom*

A disjoint data properties axiom `DisjointDataProperties( DPE1 ... DPEn )` states that all of the data property expressions `DPEi`,  $1 \leq i \leq n$ , are pairwise disjoint; that is, no individual `x` can be connected to a literal `y` by both `DPEi` and `DPEj` for  $i \neq j$ .

([https://www.w3.org/TR/owl2-syntax/#Disjoint\\_Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Disjoint_Data_Properties))

```

__slots__ = ()

```

**class** owlapy.owl\_axiom.**OWLSubClassOfAxiom**(  
*sub\_class: owlapy.class\_expression.OWLClassExpression,*  
*super\_class: owlapy.class\_expression.OWLClassExpression,*  
*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLClassAxiom*

A subclass axiom `SubClassOf( CE1 CE2 )` states that the class expression `CE1` is a subclass of the class expression `CE2`. Roughly speaking, this states that `CE1` is more specific than `CE2`. Subclass axioms are a fundamental type of axioms in OWL 2 and can be used to construct a class hierarchy. Other kinds of class expression axiom can be seen as syntactic shortcuts for one or more subclass axioms.

([https://www.w3.org/TR/owl2-syntax/#Subclass\\_Axioms](https://www.w3.org/TR/owl2-syntax/#Subclass_Axioms))

```

__slots__ = ('_sub_class', '_super_class')

get_sub_class() → owlapy.class_expression.OWLClassExpression

get_super_class() → owlapy.class_expression.OWLClassExpression

__eq__(other)
    Return self==value.

```



```
__hash__ ()
    Return hash(self).
```

```
__repr__ ()
    Return repr(self).
```

```
class owlapy.owl_axiom.OWLDisjointUnionAxiom (cls_: owlapy.class_expression.OWLClass,
        class_expressions: List[owlapy.class_expression.OWLClassExpression],
        annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLClassAxiom*

A disjoint union axiom `DisjointUnion( C CE1 ... CEn )` states that a class `C` is a disjoint union of the class expressions `CEi`,  $1 \leq i \leq n$ , all of which are pairwise disjoint. Such axioms are sometimes referred to as covering axioms, as they state that the extensions of all `CEi` exactly cover the extension of `C`. Thus, each instance of `C` is an instance of exactly one `CEi`, and each instance of `CEi` is an instance of `C`.

([https://www.w3.org/TR/owl2-syntax/#Disjoint\\_Union\\_of\\_Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Disjoint_Union_of_Class_Expressions))

```
__slots__ = ('_cls', '_class_expressions')
```

```
get_owl_class () → owlapy.class_expression.OWLClass
```

```
get_class_expressions () → Iterable[owlapy.class_expression.OWLClassExpression]
```

```
get_owl_equivalent_classes_axiom () → OWLEquivalentClassesAxiom
```

```
get_owl_disjoint_classes_axiom () → OWLDisjointClassesAxiom
```

```
__eq__ (other)
    Return self==value.
```

```
__hash__ ()
    Return hash(self).
```

```
__repr__ ()
    Return repr(self).
```

```
class owlapy.owl_axiom.OWLClassAssertionAxiom (
    individual: owlapy.owl_individual.OWLIndividual,
    class_expression: owlapy.class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLIndividualAxiom*

A class assertion `ClassAssertion( CE a )` states that the individual `a` is an instance of the class expression `CE`.

([https://www.w3.org/TR/owl2-syntax/#Class\\_Assertions](https://www.w3.org/TR/owl2-syntax/#Class_Assertions))

```
__slots__ = ('_individual', '_class_expression')
```

```
get_individual () → owlapy.owl_individual.OWLIndividual
```

```
get_class_expression () → owlapy.class_expression.OWLClassExpression
```

```
__eq__ (other)
    Return self==value.
```

```
__hash__ ()
    Return hash(self).
```

```

__repr__()
    Return repr(self).

class owlapy.owl_axiom.OWLAnnotationProperty(iri: owlapy.iri.IRI | str)
    Bases: owlapy.owl_property.OWLProperty
    Represents an AnnotationProperty in the OWL 2 specification.

    __slots__ = '_iri'

    property iri: owlapy.iri.IRI
        Gets the IRI of this object.

        Returns
            The IRI of this object.

    property str: str
        Gets the string representation of this object

        Returns
            The IRI as string

class owlapy.owl_axiom.OWLAnnotation(property: OWLAnnotationProperty,
    value: owlapy.owl_annotation.OWLAnnotationValue)
    Bases: owlapy.owl_object.OWLObject
    Annotations are used in the various types of annotation axioms, which bind annotations to their subjects (i.e. axioms
    or declarations).

    __slots__ = ('_property', '_value')

    get_property() → OWLAnnotationProperty
        Gets the property that this annotation acts along.

        Returns
            The annotation property.

    get_value() → owlapy.owl_annotation.OWLAnnotationValue
        Gets the annotation value. The type of value will depend upon the type of the annotation e.g. whether the
        annotation is an OWLLiteral, an IRI or an OWLAnonymousIndividual.

        Returns
            The annotation value.

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

class owlapy.owl_axiom.OWLAnnotationAxiom(
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLAxiom
    A super interface for annotation axioms.

    __slots__ = ()

```

**is\_annotation\_axiom**() → bool

**class** owlapy.owl\_axiom.**OWLAnnotationAssertionAxiom**(  
    *subject: owlapy.owl\_annotation.OWLAnnotationSubject, annotation: OWLAnnotation*)

Bases: *OWLAnnotationAxiom*

An annotation assertion *AnnotationAssertion*( AP as av ) states that the annotation subject as — an IRI or an anonymous individual — is annotated with the annotation property AP and the annotation value av.

([https://www.w3.org/TR/owl2-syntax/#Annotation\\_Assertion](https://www.w3.org/TR/owl2-syntax/#Annotation_Assertion))

**\_\_slots\_\_** = ('\_subject', '\_annotation')

**get\_subject**() → *owlapy.owl\_annotation.OWLAnnotationSubject*

Gets the subject of this object.

**Returns**

The subject.

**get\_property**() → *OWLAnnotationProperty*

Gets the property.

**Returns**

The property.

**get\_value**() → *owlapy.owl\_annotation.OWLAnnotationValue*

Gets the annotation value. This is either an IRI, an *OWLAnonymousIndividual* or an *OWLLiteral*.

**Returns**

The annotation value.

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

Return self==value.

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

Return hash(self).

**\_\_repr\_\_**()

Return repr(self).

**class** owlapy.owl\_axiom.**OWLSubAnnotationPropertyOfAxiom**(  
    *sub\_property: OWLAnnotationProperty, super\_property: OWLAnnotationProperty,*  
    *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLAnnotationAxiom*

An annotation subproperty axiom *SubAnnotationPropertyOf*( AP1 AP2 ) states that the annotation property AP1 is a subproperty of the annotation property AP2.

([https://www.w3.org/TR/owl2-syntax/#Annotation\\_Subproperties](https://www.w3.org/TR/owl2-syntax/#Annotation_Subproperties))

**\_\_slots\_\_** = ('\_sub\_property', '\_super\_property')

**get\_sub\_property**() → *OWLAnnotationProperty*

**get\_super\_property**() → *OWLAnnotationProperty*

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

Return self==value.

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

Return hash(self).

```

__repr__()
    Return repr(self).

class owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom(
    property_: OWLAnnotationProperty, domain: owlapy.iri.IRI,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLAnnotationAxiom

    An annotation property domain axiom AnnotationPropertyDomain( AP U ) states that the domain of the annotation
    property AP is the IRI U.

    (https://www.w3.org/TR/owl2-syntax/#Annotation\_Property\_Domain)

    __slots__ = ('_property', '_domain')

    get_property() → OWLAnnotationProperty

    get_domain() → owlapy.iri.IRI

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

class owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom(
    property_: OWLAnnotationProperty, range_: owlapy.iri.IRI,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLAnnotationAxiom

    An annotation property range axiom AnnotationPropertyRange( AP U ) states that the range of the annotation
    property AP is the IRI U.

    (https://www.w3.org/TR/owl2-syntax/#Annotation\_Property\_Range)

    __slots__ = ('_property', '_range')

    get_property() → OWLAnnotationProperty

    get_range() → owlapy.iri.IRI

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

class owlapy.owl_axiom.OWLSubPropertyAxiom(sub_property: _P, super_property: _P,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: Generic[_P], OWLPropertyAxiom

    Base interface for object and data sub-property axioms.

    __slots__ = ('_sub_property', '_super_property')

```

```

get_sub_property() → _P

get_super_property() → _P

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.owl_axiom.OWLSubObjectPropertyOfAxiom(
    sub_property: owlapy.owl_property.OWLObjectPropertyExpression,
    super_property: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLSubPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
        OWLObjectPropertyAxiom

Object subproperty axioms are analogous to subclass axioms, and they come in two forms. The basic form is
SubObjectPropertyOf( OPE1 OPE2 ). This axiom states that the object property expression OPE1 is a subproperty
of the object property expression OPE2 — that is, if an individual x is connected by OPE1 to an individual y, then
x is also connected by OPE2 to y. The more complex form is SubObjectPropertyOf( ObjectPropertyChain( OPE1
... OPEn ) OPE ) but ObjectPropertyChain is not represented in owlapy yet.

(https://www.w3.org/TR/owl2-syntax/#Object\_Subproperties)

__slots__ = ()

class owlapy.owl_axiom.OWLSubDataPropertyOfAxiom(
    sub_property: owlapy.owl_property.OWLDataPropertyExpression,
    super_property: owlapy.owl_property.OWLDataPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLSubPropertyAxiom[owlapy.owl_property.OWLDataPropertyExpression],
        OWLDataPropertyAxiom

A data subproperty axiom SubDataPropertyOf( DPE1 DPE2 ) states that the data property expression DPE1 is a
subproperty of the data property expression DPE2 — that is, if an individual x is connected by DPE1 to a literal y,
then x is connected by DPE2 to y as well.

(https://www.w3.org/TR/owl2-syntax/#Data\_Subproperties)

__slots__ = ()

class owlapy.owl_axiom.OWLPropertyAssertionAxiom(
    subject: owlapy.owl_individual.OWLIndividual, property_: _P, object_: _C,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: Generic[_P, _C], OWLIndividualAxiom

Base class for Property Assertion axioms.

__slots__ = ('_subject', '_property', '_object')

get_subject() → owlapy.owl_individual.OWLIndividual

get_property() → _P

get_object() → _C

```

```

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

```

**class** owlapy.owl\_axiom.**OWLObjectPropertyAssertionAxiom**(  
*subject: owlapy.owl\_individual.OWLIndividual,*  
*property\_: owlapy.owl\_property.OWLObjectPropertyExpression,*  
*object\_: owlapy.owl\_individual.OWLIndividual,*  
*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyAssertionAxiom[owlapy.owl\_property.OWLObjectPropertyExpression, owlapy.owl\_individual.OWLIndividual]*

A positive object property assertion **ObjectPropertyAssertion**( OPE a1 a2 ) states that the individual a1 is connected by the object property expression OPE to the individual a2.

([https://www.w3.org/TR/owl2-syntax/#Positive\\_Object\\_Property\\_Assertions](https://www.w3.org/TR/owl2-syntax/#Positive_Object_Property_Assertions))

```

__slots__ = ()

```

**class** owlapy.owl\_axiom.**OWLNegativeObjectPropertyAssertionAxiom**(  
*subject: owlapy.owl\_individual.OWLIndividual,*  
*property\_: owlapy.owl\_property.OWLObjectPropertyExpression,*  
*object\_: owlapy.owl\_individual.OWLIndividual,*  
*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyAssertionAxiom[owlapy.owl\_property.OWLObjectPropertyExpression, owlapy.owl\_individual.OWLIndividual]*

A negative object property assertion **NegativeObjectPropertyAssertion**( OPE a1 a2 ) states that the individual a1 is not connected by the object property expression OPE to the individual a2.

([https://www.w3.org/TR/owl2-syntax/#Negative\\_Object\\_Property\\_Assertions](https://www.w3.org/TR/owl2-syntax/#Negative_Object_Property_Assertions))

```

__slots__ = ()

```

**class** owlapy.owl\_axiom.**OWLDataPropertyAssertionAxiom**(  
*subject: owlapy.owl\_individual.OWLIndividual,*  
*property\_: owlapy.owl\_property.OWLDataPropertyExpression,*  
*object\_: owlapy.owl\_literal.OWLLiteral, annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyAssertionAxiom[owlapy.owl\_property.OWLDataPropertyExpression, owlapy.owl\_literal.OWLLiteral]*

A positive data property assertion **DataPropertyAssertion**( DPE a lt ) states that the individual a is connected by the data property expression DPE to the literal lt.

([https://www.w3.org/TR/owl2-syntax/#Positive\\_Data\\_Property\\_Assertions](https://www.w3.org/TR/owl2-syntax/#Positive_Data_Property_Assertions))

```

__slots__ = ()

```

**class** owlapy.owl\_axiom.**OWLNegativeDataPropertyAssertionAxiom**(  
*subject: owlapy.owl\_individual.OWLIndividual,*  
*property\_: owlapy.owl\_property.OWLDataPropertyExpression,*  
*object\_: owlapy.owl\_literal.OWLLiteral, annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *OWLPropertyAssertionAxiom[owlapy.owl\_property.OWLDataPropertyExpression, owlapy.owl\_literal.OWLLiteral]*

A negative data property assertion `NegativeDataPropertyAssertion( DPE a lt )` states that the individual `a` is not connected by the data property expression `DPE` to the literal `lt`.

([https://www.w3.org/TR/owl2-syntax/#Negative\\_Data\\_Property\\_Assertions](https://www.w3.org/TR/owl2-syntax/#Negative_Data_Property_Assertions))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OWLUnaryPropertyAxiom (property_: _P,
        annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: `Generic[_P]`, `OWLPropertyAxiom`

Base class for Unary property axiom.

```
__slots__ = '_property'
```

```
get_property() → _P
```

```
class owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom (
        property_: owlapy.owl_property.OWLObjectPropertyExpression,
        annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: `OWLUnaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression]`, `OWLObjectPropertyAxiom`

Base interface for functional object property axiom.

```
__slots__ = ()
```

```
__eq__ (other)
```

Return `self==value`.

```
__hash__ ()
```

Return `hash(self)`.

```
__repr__ ()
```

Return `repr(self)`.

```
class owlapy.owl_axiom.OWLFunctionalObjectPropertyAxiom (
        property_: owlapy.owl_property.OWLObjectPropertyExpression,
        annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: `OWLObjectPropertyCharacteristicAxiom`

An object property functionality axiom `FunctionalObjectProperty( OPE )` states that the object property expression `OPE` is functional — that is, for each individual `x`, there can be at most one distinct individual `y` such that `x` is connected by `OPE` to `y`.

([https://www.w3.org/TR/owl2-syntax/#Functional\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Functional_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OWLAsymmetricObjectPropertyAxiom (
        property_: owlapy.owl_property.OWLObjectPropertyExpression,
        annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: `OWLObjectPropertyCharacteristicAxiom`

An object property asymmetry axiom `AsymmetricObjectProperty( OPE )` states that the object property expression `OPE` is asymmetric — that is, if an individual `x` is connected by `OPE` to an individual `y`, then `y` cannot be connected by `OPE` to `x`.

([https://www.w3.org/TR/owl2-syntax/#Symmetric\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Symmetric_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OwlInverseFunctionalObjectPropertyAxiom (
    property_: owlapy.owl_property.OwlObjectPropertyExpression,
    annotations: Iterable[OwlAnnotation] | None = None)
```

Bases: *OwlObjectPropertyCharacteristicAxiom*

An object property inverse functionality axiom *InverseFunctionalObjectProperty*( OPE ) states that the object property expression OPE is inverse-functional — that is, for each individual x, there can be at most one individual y such that y is connected by OPE with x.

([https://www.w3.org/TR/owl2-syntax/#Inverse-Functional\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Inverse-Functional_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OwlIrreflexiveObjectPropertyAxiom (
    property_: owlapy.owl_property.OwlObjectPropertyExpression,
    annotations: Iterable[OwlAnnotation] | None = None)
```

Bases: *OwlObjectPropertyCharacteristicAxiom*

An object property irreflexivity axiom *IrreflexiveObjectProperty*( OPE ) states that the object property expression OPE is irreflexive — that is, no individual is connected by OPE to itself.

([https://www.w3.org/TR/owl2-syntax/#Irreflexive\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Irreflexive_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OwlReflexiveObjectPropertyAxiom (
    property_: owlapy.owl_property.OwlObjectPropertyExpression,
    annotations: Iterable[OwlAnnotation] | None = None)
```

Bases: *OwlObjectPropertyCharacteristicAxiom*

An object property reflexivity axiom *ReflexiveObjectProperty*( OPE ) states that the object property expression OPE is reflexive — that is, each individual is connected by OPE to itself. Each such axiom can be seen as a syntactic shortcut for the following axiom: *SubClassOf*( owl:Thing *ObjectHasSelf*( OPE ) )

([https://www.w3.org/TR/owl2-syntax/#Reflexive\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Reflexive_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OwlSymmetricObjectPropertyAxiom (
    property_: owlapy.owl_property.OwlObjectPropertyExpression,
    annotations: Iterable[OwlAnnotation] | None = None)
```

Bases: *OwlObjectPropertyCharacteristicAxiom*

An object property symmetry axiom *SymmetricObjectProperty*( OPE ) states that the object property expression OPE is symmetric — that is, if an individual x is connected by OPE to an individual y, then y is also connected by OPE to x. Each such axiom can be seen as a syntactic shortcut for the following axiom:

*SubObjectPropertyOf*( OPE *ObjectInverseOf*( OPE ) )

([https://www.w3.org/TR/owl2-syntax/#Symmetric\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Symmetric_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OwlTransitiveObjectPropertyAxiom (
    property_: owlapy.owl_property.OwlObjectPropertyExpression,
    annotations: Iterable[OwlAnnotation] | None = None)
```

Bases: *OwlObjectPropertyCharacteristicAxiom*

An object property transitivity axiom *TransitiveObjectProperty*( OPE ) states that the object property expression OPE is transitive — that is, if an individual x is connected by OPE to an individual y that is connected by OPE to an individual z, then x is also connected by OPE to z. Each such axiom can be seen as a syntactic shortcut for the following axiom: *SubObjectPropertyOf*( *ObjectPropertyChain*( OPE OPE ) OPE )



([https://www.w3.org/TR/owl2-syntax/#Transitive\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Transitive_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom(  
    property_: owlapy.owl_property.OWLDataPropertyExpression,  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLUnaryPropertyAxiom*[*owlapy.owl\_property.OWLDataPropertyExpression*],  
*OWLDataPropertyAxiom*

Base interface for Functional data property axiom.

```
__slots__ = ()
```

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
__repr__()
```

Return repr(self).

```
class owlapy.owl_axiom.OWLFunctionalDataPropertyAxiom(  
    property_: owlapy.owl_property.OWLDataPropertyExpression,  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLDataPropertyCharacteristicAxiom*

A data property functionality axiom *FunctionalDataProperty*( DPE ) states that the data property expression DPE is functional — that is, for each individual x, there can be at most one distinct literal y such that x is connected by DPE with y. Each such axiom can be seen as a syntactic shortcut for the following axiom: *SubClassOf*( owl:Thing *DataMaxCardinality*( 1 DPE ) )

([https://www.w3.org/TR/owl2-syntax/#Transitive\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Transitive_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OWLPropertyDomainAxiom(property_: _P,  
    domain: owlapy.class_expression.OWLClassExpression,  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *Generic*[\_P], *OWLUnaryPropertyAxiom*[\_P]

Base class for Property Domain axioms.

```
__slots__ = '_domain'
```

```
get_domain() → owlapy.class_expression.OWLClassExpression
```

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
__repr__()
```

Return repr(self).

```
class owlapy.owl_axiom.OWLPropertyRangeAxiom (property_: _P, range_: _R,
        annotations: Iterable[OWLAnnotation] | None = None)
```

```
Bases: Generic[_P, _R], OWLUnaryPropertyAxiom[_P]
```

Base class for Property Range axioms.

```
__slots__ = '_range'
```

```
get_range() → _R
```

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
__repr__()
```

Return repr(self).

```
class owlapy.owl_axiom.OWLObjectPropertyDomainAxiom (
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    domain: owlapy.class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
```

```
Bases: OWLPropertyDomainAxiom[owlapy.owl_property.OWLObjectPropertyExpression]
```

An object property domain axiom ObjectPropertyDomain( OPE CE ) states that the domain of the object property expression OPE is the class expression CE — that is, if an individual x is connected by OPE with some other individual, then x is an instance of CE. Each such axiom can be seen as a syntactic shortcut for the following axiom: SubClassOf( ObjectSomeValuesFrom( OPE owl:Thing ) CE )

([https://www.w3.org/TR/owl2-syntax/#Object\\_Property\\_Domain](https://www.w3.org/TR/owl2-syntax/#Object_Property_Domain))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OWLDataPropertyDomainAxiom (
    property_: owlapy.owl_property.OWLDataPropertyExpression,
    domain: owlapy.class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
```

```
Bases: OWLPropertyDomainAxiom[owlapy.owl_property.OWLDataPropertyExpression]
```

A data property domain axiom DataPropertyDomain( DPE CE ) states that the domain of the data property expression DPE is the class expression CE — that is, if an individual x is connected by DPE with some literal, then x is an instance of CE. Each such axiom can be seen as a syntactic shortcut for the following axiom: SubClassOf( DataSomeValuesFrom( DPE rdfs:Literal ) CE )

([https://www.w3.org/TR/owl2-syntax/#Data\\_Property\\_Domain](https://www.w3.org/TR/owl2-syntax/#Data_Property_Domain))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OWLObjectPropertyRangeAxiom (
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    range_: owlapy.class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
```

```
Bases: OWLPropertyRangeAxiom[owlapy.owl_property.OWLObjectPropertyExpression,
    owlapy.class_expression.OWLClassExpression]
```

An object property range axiom ObjectPropertyRange( OPE CE ) states that the range of the object property expression OPE is the class expression CE — that is, if some individual is connected by OPE with an individual x, then x is an instance of CE. Each such axiom can be seen as a syntactic shortcut for the following axiom: SubClassOf( owl:Thing ObjectAllValuesFrom( OPE CE ) )

([https://www.w3.org/TR/owl2-syntax/#Object\\_Property\\_Range](https://www.w3.org/TR/owl2-syntax/#Object_Property_Range))

```
__slots__ = ()
```

```
class owlapy.owl_axiom.OWLDataPropertyRangeAxiom(  
    property_: owlapy.owl_property.OWLDataPropertyExpression,  
    range_: owlapy.owl_datatype.OWLDataRange,  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: `OWLPropertyRangeAxiom[owlapy.owl_property.OWLDataPropertyExpression, owlapy.owl_datatype.OWLDataRange]`

A data property range axiom `DataPropertyRange( DPE DR )` states that the range of the data property expression DPE is the data range DR — that is, if some individual is connected by DPE with a literal x, then x is in DR. The arity of DR must be one. Each such axiom can be seen as a syntactic shortcut for the following axiom: `SubClassOf( owl:Thing DataAllValuesFrom( DPE DR ) )`

([https://www.w3.org/TR/owl2-syntax/#Data\\_Property\\_Range](https://www.w3.org/TR/owl2-syntax/#Data_Property_Range))

```
__slots__ = ()
```

## owlapy.owl\_data\_ranges

OWL Data Ranges

[https://www.w3.org/TR/owl2-syntax/#Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Data_Ranges)

`DataRange := Datatype | DataIntersectionOf | DataUnionOf | DataComplementOf | DataOneOf | DatatypeRestriction`

## Classes

<i>OWLObject</i>	Base interface for OWL objects
<i>HasOperands</i>	An interface to objects that have a collection of operands.
<i>OWLPropertyRange</i>	OWL Objects that can be the ranges of properties.
<i>OWLDataRange</i>	Represents a <code>DataRange</code> in the OWL 2 Specification.
<i>OWLNaryDataRange</i>	<code>OWLNaryDataRange</code> .
<i>OWLDataIntersectionOf</i>	An intersection data range <code>DataIntersectionOf( DR1 ... DRn )</code> contains all tuples of literals that are contained
<i>OWLDataUnionOf</i>	A union data range <code>DataUnionOf( DR1 ... DRn )</code> contains all tuples of literals that are contained in the at least
<i>OWLDataComplementOf</i>	A complement data range <code>DataComplementOf( DR )</code> contains all tuples of literals that are not contained in the

## Module Contents

```
class owlapy.owl_data_ranges.OWLObject
```

Base interface for OWL objects

```
__slots__ = ()
```

```
abstract __eq__(other)
```

Return `self==value`.

```

abstract __hash__()
    Return hash(self).

abstract __repr__()
    Return repr(self).

is_anonymous() → bool

class owlapy.owl_data_ranges.HasOperands
    Bases: Generic[_T]
    An interface to objects that have a collection of operands.

        Parameters
            _T – Operand type.

    __slots__ = ()

    abstract operands() → Iterable[_T]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
            The operands.

class owlapy.owl_data_ranges.OWLPropertyRange
    Bases: owlapy.owl_object.OWLObject
    OWL Objects that can be the ranges of properties.

class owlapy.owl_data_ranges.OWLDataRange
    Bases: OWLPropertyRange
    Represents a DataRange in the OWL 2 Specification.

class owlapy.owl_data_ranges.OWLNaryDataRange(operands: Iterable[OWLDataRange])
    Bases: OWLDataRange, owlapy.meta_classes.HasOperands[OWLDataRange]
    OWLNaryDataRange.

    __slots__ = ()

    operands() → Iterable[OWLDataRange]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
            The operands.

    __repr__()
        Return repr(self).

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

class owlapy.owl_data_ranges.OWLDataIntersectionOf(
    operands: Iterable[OWLDataRange])
    Bases: OWLNaryDataRange

```

An intersection data range `DataIntersectionOf( DR1 ... DRn )` contains all tuples of literals that are contained in each data range `DRi` for  $1 \leq i \leq n$ . All data ranges `DRi` must be of the same arity, and the resulting data range is of that arity as well.

([https://www.w3.org/TR/owl2-syntax/#Intersection\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Intersection_of_Data_Ranges))

```
__slots__ = '_operands'
```

```
type_index: Final = 4004
```

```
class owlapy.owl_data_ranges.OWLDataUnionOf (operands: Iterable[OWLDataRange])
```

Bases: *OWLNaryDataRange*

A union data range `DataUnionOf( DR1 ... DRn )` contains all tuples of literals that are contained in the at least one data range `DRi` for  $1 \leq i \leq n$ . All data ranges `DRi` must be of the same arity, and the resulting data range is of that arity as well.

([https://www.w3.org/TR/owl2-syntax/#Union\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Union_of_Data_Ranges))

```
__slots__ = '_operands'
```

```
type_index: Final = 4005
```

```
class owlapy.owl_data_ranges.OWLDataComplementOf (data_range: OWLDataRange)
```

Bases: *OWLDataRange*

A complement data range `DataComplementOf( DR )` contains all tuples of literals that are not contained in the data range `DR`. The resulting data range has the arity equal to the arity of `DR`.

([https://www.w3.org/TR/owl2-syntax/#Complement\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Complement_of_Data_Ranges))

```
type_index: Final = 4002
```

```
get_data_range () → OWLDataRange
```

#### Returns

The wrapped data range.

```
__repr__ ()
```

Return `repr(self)`.

```
__eq__ (other)
```

Return `self==value`.

```
__hash__ ()
```

Return `hash(self)`.

## owlapy.owl\_datatype

OWL Datatype

## Classes

<i>OWLEntity</i>	Represents Entities in the OWL 2 Specification.
<i>OWLDataRange</i>	Represents a DataRange in the OWL 2 Specification.
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>HasIRI</i>	Simple class to access the IRI.
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,

## Module Contents

**class** owlapy.owl\_datatype.**OWLEntity**

Bases: OWLNamedObject

Represents Entities in the OWL 2 Specification.

**\_\_slots\_\_** = ()

**to\_string\_id**() → str

**is\_anonymous**() → bool

**class** owlapy.owl\_datatype.**OWLDataRange**

Bases: OWLPropertyRange

Represents a DataRange in the OWL 2 Specification.

**class** owlapy.owl\_datatype.**IRI** (*namespace: str* | *owlapy.namespaces.Namespaces*, *remainder: str*)

Bases: *owlapy.owl\_annotation.OWLAnnotationSubject*, *owlapy.owl\_annotation.OWLAnnotationValue*

An IRI, consisting of a namespace and a remainder.

**\_\_slots\_\_** = ('\_namespace', '\_remainder', '\_\_weakref\_\_')

**type\_index: Final** = 0

**static create** (*namespace: owlapy.namespaces.Namespaces*, *remainder: str*) → *IRI*

**static create** (*namespace: str*, *remainder: str*) → *IRI*

**static create** (*string: str*) → *IRI*

**\_\_repr\_\_**()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**is\_nothing**()

Determines if this IRI is equal to the IRI that owl:Nothing is named with.

### Returns

True if this IRI is equal to <<http://www.w3.org/2002/07/owl#Nothing>> and otherwise False.

**is\_thing()**

Determines if this IRI is equal to the IRI that owl:Thing is named with.

**Returns**

True if this IRI is equal to <<http://www.w3.org/2002/07/owl#Thing>> and otherwise False.

**is\_reserved\_vocabulary()** → bool

Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>> or <<http://www.w3.org/2000/01/rdf-schema#>> or <<http://www.w3.org/2001/XMLSchema#>> or <<http://www.w3.org/2002/07/owl#>>.

**Returns**

True if the IRI is in the reserved vocabulary, otherwise False.

**as\_iri()** → *IRI*

**Returns**

if the value is an IRI, return it. Return None otherwise.

**as\_str()** → str

CD: Should be deprecated. :returns: The string that specifies the IRI.

**property str: str**

Returns: The string that specifies the IRI.

**property reminder: str**

Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form()** → str

Gets the short form.

**Returns**

A string that represents the short form.

**get\_namespace()** → str

**Returns**

The namespace as string.

**get\_remainder()** → str

**Returns**

The remainder (coincident with NCName usually) for this IRI.

**class owlapy.owl\_datatype.HasIRI**

Simple class to access the IRI.

**\_\_slots\_\_ = ()**

**property iri: *IRI***

**Abstractmethod**

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str: str**

**Abstractmethod**

Gets the string representation of this object

**Returns**

The IRI as string

**class** owlapy.owl\_datatype.**OWLDatatype** (*iri: owlapy.iri.IRI | owlapy.meta\_classes.HasIRI*)  
Bases: *owlapy.owl\_object.OWLEntity, owlapy.owl\_data\_ranges.OWLDataRange*

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

**\_\_slots\_\_** = **'\_iri'**

**type\_index:** **Final** = **4001**

**property iri:** *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str:** **str**

Gets the string representation of this object

**Returns**

The IRI as string

## owlapy.owl\_hierarchy

Classes representing hierarchy in OWL.

### Attributes

*OWLThing*

*OWLNothing*

*OWLTopObjectProperty*

*OWLBottomObjectProperty*

*OWLTopDataProperty*

*OWLBottomDataProperty*



## Classes

<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>HasIRI</i>	Simple class to access the IRI.
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
<i>OWLObjectProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
<i>OWLReasoner</i>	An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of
<i>AbstractHierarchy</i>	Representation of an abstract hierarchy which can be used for classes or properties.
<i>ClassHierarchy</i>	Representation of a class hierarchy.
<i>ObjectPropertyHierarchy</i>	Representation of an object property hierarchy.
<i>DatatypePropertyHierarchy</i>	Representation of a data property hierarchy.

## Module Contents

**class** owlapy.owl\_hierarchy.OWLClass (*iri: owlapy.iri.IRI | str*)

Bases: *owlapy.class\_expression.class\_expression.OWLClassExpression, owlapy.owl\_object.OWLEntity*

An OWL 2 named Class. Classes can be understood as sets of individuals. (<https://www.w3.org/TR/owl2-syntax/#Classes>)

**\_\_slots\_\_** = ('\_iri', '\_is\_nothing', '\_is\_thing')

**type\_index:** Final = 1001

**property iri:** *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str**

**Gets the string representation of this object**

**Returns**

The IRI as string

**property reminder:** str

The reminder of the IRI

**is\_owl\_thing** () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

```

is_owl_nothing () → bool
    Determines if this expression is the built in class owl:Nothing. This method does not determine if the class
    is equivalent to owl:Nothing.

get_object_complement_of ()
    → owlapy.class_expression.class_expression.OWLObjectComplementOf
    Gets the object complement of this class expression.

    Returns
        A class expression that is the complement of this class expression.

get_nnf () → OWLClass
    Gets the negation normal form of the complement of this expression.

    Returns
        A expression that represents the NNF of the complement of this expression.

owlapy.owl_hierarchy.OWLThing: Final

owlapy.owl_hierarchy.OWLNothing: Final

class owlapy.owl_hierarchy.HasIRI
    Simple class to access the IRI.

    __slots__ = ()

    property iri: IRI

        Abstractmethod

        Gets the IRI of this object.

        Returns
            The IRI of this object.

    property str: str

        Abstractmethod

        Gets the string representation of this object

        Returns
            The IRI as string

owlapy.owl_hierarchy.OWLTopObjectProperty: Final

owlapy.owl_hierarchy.OWLBottomObjectProperty: Final

owlapy.owl_hierarchy.OWLTopDataProperty: Final

owlapy.owl_hierarchy.OWLBottomDataProperty: Final

class owlapy.owl_hierarchy.OWLObjectProperty (iri: owlapy.iri.IRI | str)
    Bases: OWLObjectPropertyExpression, OWLProperty
    Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
    (https://www.w3.org/TR/owl2-syntax/#Object\_Properties)

    __slots__ = '_iri'

    type_index: Final = 1002

```

**get\_named\_property** () → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**get\_inverse\_property** () → *OWLObjectInverseOf*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**is\_owl\_top\_object\_property** () → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

**class** owlapy.owl\_hierarchy.OWLDataProperty (*iri: owlapy.iri.IRI | str*)

Bases: OWLDataPropertyExpression, OWLProperty

Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.

([https://www.w3.org/TR/owl2-syntax/#Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Data_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index: Final** = 1004

**is\_owl\_top\_data\_property** () → bool

Determines if this is the owl:topDataProperty.

**Returns**

topDataProperty.

**Return type**

True if this property is the owl

**class** owlapy.owl\_hierarchy.OWLReasoner (*ontology: owlapy.owl\_ontology.OWLOntology*)

An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of a particular ontology - the “root” ontology.

**\_\_slots\_\_** = ()

**abstract data\_property\_domains** (*pe: owlapy.owl\_property.OWLDataProperty,*  
*direct: bool = False*) → Iterable[*owlapy.class\_expression.OWLClassExpression*]

**Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.**

**Parameters**

- **pe** – The property expression whose domains are to be retrieved.
- **direct** – Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

### Returns

Let  $N = \text{equivalent\_classes}(\text{DataSomeValuesFrom}(\text{pe } \text{rdfs:Literal}))$ . If `direct` is `True`: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of `super_classes(DataSomeValuesFrom(pe rdfs:Literal), true)`. If `direct` is `False`: then the result of `super_classes(DataSomeValuesFrom(pe rdfs:Literal), false)` together with  $N$  if  $N$  is non-empty. (Note, `rdfs:Literal` is the top datatype).

**abstract object\_property\_domains** (*pe*: *owlapy.owl\_property.OWLObjectProperty*,  
*direct*: *bool* = *False*)  $\rightarrow$  *Iterable*[*owlapy.class\_expression.OWLClassExpression*]

**Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.**

### Parameters

- **pe** – The property expression whose domains are to be retrieved.
- **direct** – Specifies if the direct domains should be retrieved (`True`), or if all domains should be retrieved (`False`).

### Returns

Let  $N = \text{equivalent\_classes}(\text{ObjectSomeValuesFrom}(\text{pe } \text{owl:Thing}))$ . If `direct` is `True`: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of `super_classes(ObjectSomeValuesFrom(pe owl:Thing), true)`. If `direct` is `False`: then the result of `super_classes(ObjectSomeValuesFrom(pe owl:Thing), false)` together with  $N$  if  $N$  is non-empty.

**abstract object\_property\_ranges** (*pe*: *owlapy.owl\_property.OWLObjectProperty*,  
*direct*: *bool* = *False*)  $\rightarrow$  *Iterable*[*owlapy.class\_expression.OWLClassExpression*]

**Gets the class expressions that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.**

### Parameters

- **pe** – The property expression whose ranges are to be retrieved.
- **direct** – Specifies if the direct ranges should be retrieved (`True`), or if all ranges should be retrieved (`False`).

### Returns

Let  $N = \text{equivalent\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe } \text{owl:Thing})))$ . If `direct` is `True`: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of `super_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe owl:Thing), true)`. If `direct` is `False`: then the result of `super_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe owl:Thing), false)` together with  $N$  if  $N$  is non-empty.

**abstract equivalent\_classes** (*ce*: *owlapy.class\_expression.OWLClassExpression*,  
*only\_named*: *bool* = *True*)  $\rightarrow$  *Iterable*[*owlapy.class\_expression.OWLClassExpression*]

**Gets the class expressions that are equivalent to the specified class expression with respect to the set of reasoner axioms.**

### Parameters

- **ce** – The class expression whose equivalent classes are to be retrieved.
- **only\_named** – Whether to only retrieve named equivalent classes or also complex class expressions.

### Returns

All class expressions C where the root ontology imports closure entails EquivalentClasses(ce C). If ce is not a class name (i.e. it is an anonymous class expression) and there are no such classes C then there will be no result. If ce is unsatisfiable with respect to the set of reasoner axioms then owl:Nothing, i.e. the bottom node, will be returned.

**abstract disjoint\_classes** (ce: *owlapy.class\_expression.OWLClassExpression*,  
only\_named: bool = True) → Iterable[*owlapy.class\_expression.OWLClassExpression*]

Gets the class expressions that are disjoint with specified class expression with respect to the set of reasoner axioms.

### Parameters

- **ce** – The class expression whose disjoint classes are to be retrieved.
- **only\_named** – Whether to only retrieve named disjoint classes or also complex class expressions.

### Returns

All class expressions D where the set of reasoner axioms entails EquivalentClasses(D Object-ComplementOf(ce)) or StrictSubClassOf(D ObjectComplementOf(ce)).

**abstract different\_individuals** (ind: *owlapy.owl\_individual.OWLNamedIndividual*)  
→ Iterable[*owlapy.owl\_individual.OWLNamedIndividual*]

Gets the individuals that are different from the specified individual with respect to the set of reasoner axioms.

### Parameters

**ind** – The individual whose different individuals are to be retrieved.

### Returns

All individuals x where the set of reasoner axioms entails DifferentIndividuals(ind x).

**abstract same\_individuals** (ind: *owlapy.owl\_individual.OWLNamedIndividual*)  
→ Iterable[*owlapy.owl\_individual.OWLNamedIndividual*]

Gets the individuals that are the same as the specified individual with respect to the set of reasoner axioms.

### Parameters

**ind** – The individual whose same individuals are to be retrieved.

### Returns

All individuals x where the root ontology imports closure entails SameIndividual(ind x).

**abstract equivalent\_object\_properties** (  
op: *owlapy.owl\_property.OWLObjectPropertyExpression*)  
→ Iterable[*owlapy.owl\_property.OWLObjectPropertyExpression*]

Gets the simplified object properties that are equivalent to the specified object property with respect to the set of reasoner axioms.

### Parameters

**op** – The object property whose equivalent object properties are to be retrieved.

### Returns

All simplified object properties e where the root ontology imports closure entails EquivalentObjectProperties(op e). If op is unsatisfiable with respect to the set of reasoner axioms then owl:bottomDataProperty will be returned.

**abstract equivalent\_data\_properties** (dp: *owlapy.owl\_property.OWLDataProperty*)  
→ Iterable[*owlapy.owl\_property.OWLDataProperty*]

Gets the data properties that are equivalent to the specified data property with respect to the set of reasoner axioms.

### Parameters

**dp** – The data property whose equivalent data properties are to be retrieved.

### Returns

All data properties *e* where the root ontology imports closure entails `EquivalentDataProperties(dp e)`. If *dp* is unsatisfiable with respect to the set of reasoner axioms then `owl:bottomDataProperty` will be returned.

**abstract data\_property\_values** (*ind*: *owlapy.owl\_individual.OWLNamedIndividual*,  
*pe*: *owlapy.owl\_property.OWLDataProperty*, *direct*: *bool = True*)  
→ *Iterable[owlapy.owl\_literal.OWLLiteral]*

Gets the data property values for the specified individual and data property expression.

### Parameters

- **ind** – The individual that is the subject of the data property values.
- **pe** – The data property expression whose values are to be retrieved for the specified individual.
- **direct** – Specifies if the direct values should be retrieved (*True*), or if all values should be retrieved (*False*), so that sub properties are taken into account.

### Returns

A set of *OWLLiterals* containing literals such that for each literal *l* in the set, the set of reasoner axioms entails `DataPropertyAssertion(pe ind l)`.

**abstract object\_property\_values** (*ind*: *owlapy.owl\_individual.OWLNamedIndividual*,  
*pe*: *owlapy.owl\_property.OWLObjectPropertyExpression*, *direct*: *bool = True*)  
→ *Iterable[owlapy.owl\_individual.OWLNamedIndividual]*

Gets the object property values for the specified individual and object property expression.

### Parameters

- **ind** – The individual that is the subject of the object property values.
- **pe** – The object property expression whose values are to be retrieved for the specified individual.
- **direct** – Specifies if the direct values should be retrieved (*True*), or if all values should be retrieved (*False*), so that sub properties are taken into account.

### Returns

The named individuals such that for each individual *j*, the set of reasoner axioms entails `ObjectPropertyAssertion(pe ind j)`.

**abstract flush** () → *None*

Flushes any changes stored in the buffer, which causes the reasoner to take into consideration the changes the current root ontology specified by the changes.

**abstract instances** (*ce*: *owlapy.class\_expression.OWLClassExpression*, *direct*: *bool = False*)  
→ *Iterable[owlapy.owl\_individual.OWLNamedIndividual]*

Gets the individuals which are instances of the specified class expression.

### Parameters

- **ce** – The class expression whose instances are to be retrieved.
- **direct** – Specifies if the direct instances should be retrieved (*True*), or if all instances should be retrieved (*False*).

### Returns

If `direct` is `True`, each named individual `j` where the set of reasoner axioms entails `DirectClassAssertion(ce, j)`. If `direct` is `False`, each named individual `j` where the set of reasoner axioms entails `ClassAssertion(ce, j)`. If `ce` is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

**abstract sub\_classes** (*ce: owlapy.class\_expression.OWLClassExpression, direct: bool = False, only\_named: bool = True*) → `Iterable[owlapy.class_expression.OWLClassExpression]`

Gets the set of named classes that are the strict (potentially direct) subclasses of the specified class expression with respect to the reasoner axioms.

### Parameters

- **ce** – The class expression whose strict (direct) subclasses are to be retrieved.
- **direct** – Specifies if the direct subclasses should be retrieved (`True`) or if the all subclasses (descendant) classes should be retrieved (`False`).
- **only\_named** – Whether to only retrieve named sub-classes or also complex class expressions.

### Returns

If `direct` is `True`, each class `C` where reasoner axioms entails `DirectSubClassOf(C, ce)`. If `direct` is `False`, each class `C` where reasoner axioms entails `StrictSubClassOf(C, ce)`. If `ce` is equivalent to `owl:Nothing` then nothing will be returned.

**abstract disjoint\_object\_properties** (*op: owlapy.owl\_property.OWLObjectPropertyExpression*) → `Iterable[owlapy.owl_property.OWLObjectPropertyExpression]`

Gets the simplified object properties that are disjoint with the specified object property with respect to the set of reasoner axioms.

### Parameters

- **op** – The object property whose disjoint object properties are to be retrieved.

### Returns

All simplified object properties `e` where the root ontology imports closure entails `EquivalentObjectProperties(e ObjectPropertyComplementOf(op))` or `StrictSubObjectPropertyOf(e ObjectPropertyComplementOf(op))`.

**abstract disjoint\_data\_properties** (*dp: owlapy.owl\_property.OWLDataProperty*) → `Iterable[owlapy.owl_property.OWLDataProperty]`

Gets the data properties that are disjoint with the specified data property with respect to the set of reasoner axioms.

### Parameters

- **dp** – The data property whose disjoint data properties are to be retrieved.

### Returns

All data properties `e` where the root ontology imports closure entails `EquivalentDataProperties(e DataPropertyComplementOf(dp))` or `StrictSubDataPropertyOf(e DataPropertyComplementOf(dp))`.

**abstract sub\_data\_properties** (*dp: owlapy.owl\_property.OWLDataProperty, direct: bool = False*) → `Iterable[owlapy.owl_property.OWLDataProperty]`

Gets the set of named data properties that are the strict (potentially direct) subproperties of the specified data property expression with respect to the imports closure of the root ontology.

### Parameters

- **dp** – The data property whose strict (direct) subproperties are to be retrieved.

- **direct** – Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

#### Returns

If **direct** is True, each property P where the set of reasoner axioms entails `DirectSubDataPropertyOf(P, pe)`. If **direct** is False, each property P where the set of reasoner axioms entails `StrictSubDataPropertyOf(P, pe)`. If **pe** is equivalent to `owl:bottomDataProperty` then nothing will be returned.

**abstract super\_data\_properties** (*dp: owlapy.owl\_property.OWLDataProperty, direct: bool = False*) → `Iterable[owlapy.owl_property.OWLDataProperty]`

Gets the stream of data properties that are the strict (potentially direct) super properties of the specified data property with respect to the imports closure of the root ontology.

#### Parameters

- **dp** (`OWLDataProperty`) – The data property whose super properties are to be retrieved.
- **direct** (`bool`) – Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

#### Returns

Iterable of super properties.

**abstract sub\_object\_properties** (*op: owlapy.owl\_property.OWLObjectPropertyExpression, direct: bool = False*) → `Iterable[owlapy.owl_property.OWLObjectPropertyExpression]`

Gets the stream of simplified object property expressions that are the strict (potentially direct) subproperties of the specified object property expression with respect to the imports closure of the root ontology.

#### Parameters

- **op** – The object property expression whose strict (direct) subproperties are to be retrieved.
- **direct** – Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

#### Returns

If **direct** is True, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails `DirectSubObjectPropertyOf(P, pe)`. If **direct** is False, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails `StrictSubObjectPropertyOf(P, pe)`. If **pe** is equivalent to `owl:bottomObjectProperty` then nothing will be returned.

**abstract super\_object\_properties** (*op: owlapy.owl\_property.OWLObjectPropertyExpression, direct: bool = False*) → `Iterable[owlapy.owl_property.OWLObjectPropertyExpression]`

Gets the stream of object properties that are the strict (potentially direct) super properties of the specified object property with respect to the imports closure of the root ontology.

#### Parameters

- **op** (`OWLObjectPropertyExpression`) – The object property expression whose super properties are to be retrieved.
- **direct** (`bool`) – Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

#### Returns

Iterable of super properties.

**abstract types** (*ind: owlapy.owl\_individual.OWLNamedIndividual, direct: bool = False*) → `Iterable[owlapy.class_expression.OWLClass]`

Gets the named classes which are (potentially direct) types of the specified named individual.



### Parameters

- **ind** – The individual whose types are to be retrieved.
- **direct** – Specifies if the direct types should be retrieved (True), or if all types should be retrieved (False).

### Returns

If **direct** is True, each named class C where the set of reasoner axioms entails `DirectClassAssertion(C, ind)`. If **direct** is False, each named class C where the set of reasoner axioms entails `ClassAssertion(C, ind)`.

**abstract** `get_root_ontology()` → *owlapy.owl\_ontology.OWL ontology*

Gets the “root” ontology that is loaded into this reasoner. The reasoner takes into account the axioms in this ontology and its import’s closure.

**abstract** `is_isolated()`

Return True if this reasoner is using an isolated ontology.

**abstract** `super_classes(ce: owlapy.class_expression.OWLClassExpression, direct: bool = False, only_named: bool = True)` → *Iterable[owlapy.class\_expression.OWLClassExpression]*

Gets the stream of named classes that are the strict (potentially direct) super classes of the specified class expression with respect to the imports closure of the root ontology.

### Parameters

- **ce** – The class expression whose strict (direct) super classes are to be retrieved.
- **direct** – Specifies if the direct super classes should be retrieved (True) or if the all super classes (ancestors) classes should be retrieved (False).
- **only\_named** – Whether to only retrieve named super classes or also complex class expressions.

### Returns

If **direct** is True, each class C where the set of reasoner axioms entails `DirectSubClassOf(ce, C)`. If **direct** is False, each class C where set of reasoner axioms entails `StrictSubClassOf(ce, C)`. If **ce** is equivalent to `owl:Thing` then nothing will be returned.

**class** `owlapy.owl_hierarchy.AbstractHierarchy` (*factory: Type[\_S], hierarchy\_down: Iterable[Tuple[\_S, Iterable[\_S]]]*)

**class** `owlapy.owl_hierarchy.AbstractHierarchy` (*factory: Type[\_S], reasoner: owlapy.owl\_reasoner.OWLReasoner*)

Bases: `Generic[_S]`

Representation of an abstract hierarchy which can be used for classes or properties.

### Parameters

- **hierarchy\_down** – A downwards hierarchy given as a mapping of Entities to sub-entities.
- **reasoner** – Alternatively, a reasoner whose `root_ontology` is queried for entities.

`__slots__ = ('_Type', '_ent_set', '_parents_map', '_parents_map_trans', '_children_map', ...)`

**classmethod** `get_top_entity()` → *\_S*

### Abstractmethod

The most general entity in this hierarchy, which contains all the entities.

**classmethod** **get\_bottom\_entity** () → *\_S*

**Abstractmethod**

The most specific entity in this hierarchy, which contains none of the entities.

**static restrict** (*hierarchy: \_U, \*, remove: Iterable[\_S] = None, allow: Iterable[\_S] = None*) → *\_U*

Restrict a given hierarchy to a set of allowed/removed entities.

**Parameters**

- **hierarchy** – An existing Entity hierarchy to restrict.
- **remove** – Set of entities which should be ignored.
- **allow** – Set of entities which should be used.

**Returns**

The restricted hierarchy.

**restrict\_and\_copy** (*\*, remove: Iterable[\_S] = None, allow: Iterable[\_S] = None*) → *\_U*

Restrict this hierarchy.

See restrict for more info.

**parents** (*entity: \_S, direct: bool = True*) → *Iterable[\_S]*

Parents of an entity.

**Parameters**

- **entity** – Entity for which to query parent entities.
- **direct** – False to return transitive parents.

**Returns**

Super-entities.

**is\_parent\_of** (*a: \_S, b: \_S*) → *bool*

if A is a parent of B.

---

**Note:** A is always a parent of A.

---

**is\_child\_of** (*a: \_S, b: \_S*) → *bool*

If A is a child of B.

---

**Note:** A is always a child of A.

---

**children** (*entity: \_S, direct: bool = True*) → *Iterable[\_S]*

Children of an entity.

**Parameters**

- **entity** – Entity for which to query child entities.
- **direct** – False to return transitive children.

**Returns**

Sub-entities.

**siblings** (*entity: \_S*) → *Iterable[\_S]*

```

items () → Iterable[_S]

roots (of: _S | None = None) → Iterable[_S]

leaves (of: _S | None = None) → Iterable[_S]

__contains__ (item: _S) → bool

__len__ ()

class owlapy.owl_hierarchy.ClassHierarchy (
    hierarchy_down: Iterable[Tuple[owlapy.class_expression.OWLClass, Iterable[owlapy.class_expression.OWLClass]]])

class owlapy.owl_hierarchy.ClassHierarchy (reasoner: owlapy.owl_reasoner.OWLReasoner)
    Bases: AbstractHierarchy[owlapy.class_expression.OWLClass]
    Representation of a class hierarchy.

    Parameters
    • hierarchy_down – A downwards hierarchy given as a mapping of Class to sub-classes.
    • reasoner – Alternatively, a reasoner whose root_ontology is queried for classes and sub-classes.

    classmethod get_top_entity () → owlapy.class_expression.OWLClass
        The most general entity in this hierarchy, which contains all the entities.

    classmethod get_bottom_entity () → owlapy.class_expression.OWLClass
        The most specific entity in this hierarchy, which contains none of the entities.

    sub_classes (entity: owlapy.class_expression.OWLClass, direct: bool = True)
        → Iterable[owlapy.class_expression.OWLClass]

    super_classes (entity: owlapy.class_expression.OWLClass, direct: bool = True)
        → Iterable[owlapy.class_expression.OWLClass]

    is_subclass_of (subclass: owlapy.class_expression.OWLClass,
        superclass: owlapy.class_expression.OWLClass) → bool

class owlapy.owl_hierarchy.ObjectPropertyHierarchy (
    hierarchy_down: Iterable[Tuple[owlapy.owl_property.OWLObjectProperty, Iterable[owlapy.owl_property.OWLObjectProperty]])

class owlapy.owl_hierarchy.ObjectPropertyHierarchy (
    reasoner: owlapy.owl_reasoner.OWLReasoner)
    Bases: AbstractHierarchy[owlapy.owl_property.OWLObjectProperty]
    Representation of an object property hierarchy.

    classmethod get_top_entity () → owlapy.owl_property.OWLObjectProperty
        The most general entity in this hierarchy, which contains all the entities.

    classmethod get_bottom_entity () → owlapy.owl_property.OWLObjectProperty
        The most specific entity in this hierarchy, which contains none of the entities.

    sub_object_properties (entity: owlapy.owl_property.OWLObjectProperty, direct: bool = True)
        → Iterable[owlapy.owl_property.OWLObjectProperty]

    super_object_properties (entity: owlapy.owl_property.OWLObjectProperty, direct: bool = True)
        → Iterable[owlapy.owl_property.OWLObjectProperty]

```

```

more_general_roles (role: owlapy.owl_property.OWLObjectProperty, direct: bool = True)
    → Iterable[owlapy.owl_property.OWLObjectProperty]

more_special_roles (role: owlapy.owl_property.OWLObjectProperty, direct: bool = True)
    → Iterable[owlapy.owl_property.OWLObjectProperty]

is_sub_property_of (sub_property: owlapy.owl_property.OWLObjectProperty,
    super_property: owlapy.owl_property.OWLObjectProperty) → bool

most_general_roles () → Iterable[owlapy.owl_property.OWLObjectProperty]

most_special_roles () → Iterable[owlapy.owl_property.OWLObjectProperty]

class owlapy.owl_hierarchy.DatatypePropertyHierarchy (
    hierarchy_down: Iterable[Tuple[owlapy.owl_property.OWLDataProperty, Iterable[owlapy.owl_property.OWLDataProperty]]]

class owlapy.owl_hierarchy.DatatypePropertyHierarchy (
    reasoner: owlapy.owl_reasoner.OWLReasoner)
    Bases: AbstractHierarchy[owlapy.owl_property.OWLDataProperty]
    Representation of a data property hierarchy.

    classmethod get_top_entity () → owlapy.owl_property.OWLDataProperty
        The most general entity in this hierarchy, which contains all the entities.

    classmethod get_bottom_entity () → owlapy.owl_property.OWLDataProperty
        The most specific entity in this hierarchy, which contains none of the entities.

    sub_data_properties (entity: owlapy.owl_property.OWLDataProperty, direct: bool = True)

    super_data_properties (entity: owlapy.owl_property.OWLDataProperty, direct: bool = True)

    more_general_roles (role: owlapy.owl_property.OWLDataProperty, direct: bool = True)
        → Iterable[owlapy.owl_property.OWLDataProperty]

    more_special_roles (role: owlapy.owl_property.OWLDataProperty, direct: bool = True)
        → Iterable[owlapy.owl_property.OWLDataProperty]

    is_sub_property_of (sub_property: owlapy.owl_property.OWLDataProperty,
        super_property: owlapy.owl_property.OWLDataProperty) → bool

    most_general_roles () → Iterable[owlapy.owl_property.OWLDataProperty]

    most_special_roles () → Iterable[owlapy.owl_property.OWLDataProperty]

```

## owlapy.owl\_individual

OWL Individuals

## Classes

<i>OWLObject</i>	Base interface for OWL objects
<i>OWLEntity</i>	Represents Entities in the OWL 2 Specification.
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>OWLIndividual</i>	Represents a named or anonymous individual.
<i>OWLNamedIndividual</i>	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.

## Module Contents

```
class owlapy.owl_individual.OWLObject
```

Base interface for OWL objects

```
__slots__ = ()
```

```
abstract __eq__(other)
```

Return self==value.

```
abstract __hash__()
```

Return hash(self).

```
abstract __repr__()
```

Return repr(self).

```
is_anonymous() → bool
```

```
class owlapy.owl_individual.OWLEntity
```

Bases: OWLNamedObject

Represents Entities in the OWL 2 Specification.

```
__slots__ = ()
```

```
to_string_id() → str
```

```
is_anonymous() → bool
```

```
class owlapy.owl_individual.IRI(namespace: str | owlapy.namespaces.Namespaces,  
    remainder: str)
```

Bases: *owlapy.owl\_annotation.OWLAnnotationSubject*, *owlapy.owl\_annotation.OWLAnnotationValue*

An IRI, consisting of a namespace and a remainder.

```
__slots__ = ('_namespace', '_remainder', '__weakref__')
```

```
type_index: Final = 0
```

```
static create(namespace: owlapy.namespaces.Namespaces, remainder: str) → IRI
```

```
static create(namespace: str, remainder: str) → IRI
```

```
static create(string: str) → IRI
```

```
__repr__()
```

Return repr(self).

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

Return self==value.

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

Return hash(self).

**is\_nothing** ()

Determines if this IRI is equal to the IRI that owl:Nothing is named with.

**Returns**

True if this IRI is equal to <<http://www.w3.org/2002/07/owl#Nothing>> and otherwise False.

**is\_thing** ()

Determines if this IRI is equal to the IRI that owl:Thing is named with.

**Returns**

True if this IRI is equal to <<http://www.w3.org/2002/07/owl#Thing>> and otherwise False.

**is\_reserved\_vocabulary** () → bool

Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>> or <<http://www.w3.org/2000/01/rdf-schema#>> or <<http://www.w3.org/2001/XMLSchema#>> or <<http://www.w3.org/2002/07/owl#>>.

**Returns**

True if the IRI is in the reserved vocabulary, otherwise False.

**as\_iri** () → *IRI*

**Returns**

if the value is an IRI, return it. Return None otherwise.

**as\_str** () → str

CD: Should be deprecated. :returns: The string that specifies the IRI.

**property str:** str

Returns: The string that specifies the IRI.

**property reminder:** str

Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form** () → str

Gets the short form.

**Returns**

A string that represents the short form.

**get\_namespace** () → str

**Returns**

The namespace as string.

**get\_remainder** () → str

**Returns**

The remainder (coincident with NCName usually) for this IRI.

**class** owlapy.owl\_individual.OWLIndividual

Bases: *owlapy.owl\_object.OWLObject*

Represents a named or anonymous individual.

```
__slots__ = ()
```

```
class owlapy.owl_individual.OWLNamedIndividual (iri: owlapy.iri.IRI | str)
```

```
Bases: OWLIndividual, owlapy.owl_object.OWLEntity
```

Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities. IRIs from the reserved vocabulary must not be used to identify named individuals in an OWL 2 DL ontology.

([https://www.w3.org/TR/owl2-syntax/#Named\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Named_Individuals))

```
__slots__ = '_iri'
```

```
type_index: Final = 1005
```

```
property iri: owlapy.iri.IRI
```

Gets the IRI of this object.

**Returns**

The IRI of this object.

```
property str
```

Gets the string representation of this object

**Returns**

The IRI as string

## **owlapy.owl\_literal**

OWL Literals

## Attributes

<i>Literals</i>
<i>OWLTopObjectProperty</i>
<i>OWLBottomObjectProperty</i>
<i>OWLTopDataProperty</i>
<i>OWLBottomDataProperty</i>
<i>DoubleOWLDatatype</i>
<i>IntegerOWLDatatype</i>
<i>BooleanOWLDatatype</i>
<i>StringOWLDatatype</i>
<i>DateOWLDatatype</i>
<i>DateTimeOWLDatatype</i>
<i>DurationOWLDatatype</i>
<i>TopOWLDatatype</i>
<i>NUMERIC_DATATYPES</i>
<i>TIME_DATATYPES</i>

## Classes

<i>OWLAnnotationValue</i>	A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
<i>OWLRDFVocabulary</i>	Enumerations for OWL/RDF vocabulary.
<i>XSDVocabulary</i>	Enumerations for XSD vocabulary.
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF



## Module Contents

**class** owlapy.owl\_literal.OWLAnnotationValue

Bases: OWLAnnotationObject

A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.

**\_\_slots\_\_** = ()

**is\_literal**() → bool

**Returns**

true if the annotation value is a literal

**as\_literal**() → *OWLLiteral* | None

**Returns**

if the value is a literal, returns it. Return None otherwise

**class** owlapy.owl\_literal.OWLDatatype(*iri: owlapy.iri.IRI | owlapy.meta\_classes.HasIRI*)

Bases: *owlapy.owl\_object.OWLEntity*, *owlapy.owl\_data\_ranges.OWLDataRange*

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

**\_\_slots\_\_** = '\_iri'

**type\_index**: Final = 4001

**property iri**: *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str**: str

Gets the string representation of this object

**Returns**

The IRI as string

**class** owlapy.owl\_literal.OWLRDFVocabulary(*namespace: owlapy.namespaces.Namespaces*,  
*remainder: str*)

Bases: *\_Vocabulary*, *enum.Enum*

Enumerations for OWL/RDF vocabulary.

**OWL\_THING**

**OWL\_NOTHING**

**OWL\_CLASS**

**OWL\_NAMED\_INDIVIDUAL**

OWL\_TOP\_OBJECT\_PROPERTY

OWL\_BOTTOM\_OBJECT\_PROPERTY

OWL\_TOP\_DATA\_PROPERTY

OWL\_BOTTOM\_DATA\_PROPERTY

RDFS\_LITERAL

```
class owlapy.owl_literal.XSDVocabulary(remainder: str)
```

Bases: `_Vocabulary`, `enum.Enum`

Enumerations for XSD vocabulary.

**DECIMAL:** `Final` = `'decimal'`

**INTEGER:** `Final` = `'integer'`

**LONG:** `Final` = `'long'`

**DOUBLE:** `Final` = `'double'`

**FLOAT:** `Final` = `'float'`

**BOOLEAN:** `Final` = `'boolean'`

**STRING:** `Final` = `'string'`

**DATE:** `Final` = `'date'`

**DATE\_TIME:** `Final` = `'dateTime'`

**DATE\_TIME\_STAMP:** `Final` = `'dateTimeStamp'`

**DURATION:** `Final` = `'duration'`

```
class owlapy.owl_literal.OWLObjectProperty(iri: owlapy.iri.IRI | str)
```

Bases: `OWLObjectPropertyExpression`, `OWLProperty`

Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.

([https://www.w3.org/TR/owl2-syntax/#Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Object_Properties))

**\_\_slots\_\_** = `'_iri'`

**type\_index:** `Final` = `1002`

**get\_named\_property()** → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either `inv(P)` or `P`.

**get\_inverse\_property()** → *OWLObjectInverseOf*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**is\_owl\_top\_object\_property()** → bool  
Determines if this is the owl:topObjectProperty.

**Returns**  
topObjectProperty.

**Return type**  
True if this property is the owl

**class** owlapy.owl\_literal.OWLObjectProperty(*iri: owlapy.iri.IRI | str*)

Bases: OWLObjectPropertyExpression, OWLObjectProperty

Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.

([https://www.w3.org/TR/owl2-syntax/#Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Data_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index:** Final = 1004

**is\_owl\_top\_data\_property()** → bool  
Determines if this is the owl:topDataProperty.

**Returns**  
topDataProperty.

**Return type**  
True if this property is the owl

owlapy.owl\_literal.Literals

**class** owlapy.owl\_literal.OWLLiteral

Bases: owlapy.owl\_annotation.OWLAnnotationValue

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(<https://www.w3.org/TR/owl2-syntax/#Literals>)

**\_\_slots\_\_** = ()

**type\_index:** Final = 4008

**get\_literal()** → str  
Gets the lexical value of this literal. Note that the language tag is not included.

**Returns**  
The lexical value of this literal.

**is\_boolean()** → bool  
Whether this literal is typed as boolean.

**parse\_boolean()** → bool

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("<http://www.w3.org/2001/XMLSchema#boolean>").

**Returns**  
A bool value that is represented by this literal.

**is\_double** () → bool

Whether this literal is typed as double.

**parse\_double** () → float

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

**Returns**

A double value that is represented by this literal.

**is\_integer** () → bool

Whether this literal is typed as integer.

**parse\_integer** () → int

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

**Returns**

An integer value that is represented by this literal.

**is\_string** () → bool

Whether this literal is typed as string.

**parse\_string** () → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("<http://www.w3.org/2001/XMLSchema#string>").

**Returns**

A string value that is represented by this literal.

**is\_date** () → bool

Whether this literal is typed as date.

**parse\_date** () → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("<http://www.w3.org/2001/XMLSchema#date>").

**Returns**

A date value that is represented by this literal.

**is\_datetime** () → bool

Whether this literal is typed as dateTime.

**parse\_datetime** () → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("<http://www.w3.org/2001/XMLSchema#dateTime>").

**Returns**

A datetime value that is represented by this literal.

**is\_duration** () → bool

Whether this literal is typed as duration.

**parse\_duration** () → pandas.Timedelta

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("<http://www.w3.org/2001/XMLSchema#duration>").

**Returns**

A Timedelta value that is represented by this literal.

**is\_literal()** → bool

**Returns**

true if the annotation value is a literal

**as\_literal()** → *OWLLiteral*

**Returns**

if the value is a literal, returns it. Return None otherwise

**to\_python()** → Literals

**abstract get\_datatype()** → *owlapy.owl\_datatype.OWLDatatype*

Gets the OWLDatatype which types this literal.

**Returns**

The OWLDatatype that types this literal.

`owlapy.owl_literal.OWLTopObjectProperty: Final`

`owlapy.owl_literal.OWLBottomObjectProperty: Final`

`owlapy.owl_literal.OWLTopDataProperty: Final`

`owlapy.owl_literal.OWLBottomDataProperty: Final`

`owlapy.owl_literal.DoubleOWLDatatype: Final`

`owlapy.owl_literal.IntegerOWLDatatype: Final`

`owlapy.owl_literal.BooleanOWLDatatype: Final`

`owlapy.owl_literal.StringOWLDatatype: Final`

`owlapy.owl_literal.DateOWLDatatype: Final`

`owlapy.owl_literal.DateTimeOWLDatatype: Final`

`owlapy.owl_literal.DurationOWLDatatype: Final`

`owlapy.owl_literal.TopOWLDatatype: Final`

`owlapy.owl_literal.NUMERIC_DATATYPES:`  
`Final[Set[owlapy.owl_datatype.OWLDatatype]]`

`owlapy.owl_literal.TIME_DATATYPES: Final[Set[owlapy.owl_datatype.OWLDatatype]]`

## **owlapy.owl\_object**

OWL Base classes

## Classes

<i>HasIRI</i>	Simple class to access the IRI.
<i>OWLObject</i>	Base interface for OWL objects
<i>OWLObjectRenderer</i>	Abstract class with a render method to render an OWL Object into a string.
<i>OWLObjectParser</i>	Abstract class with a parse method to parse a string to an OWL Object.
<i>OWLNamedObject</i>	Represents a named object for example, class, property, ontology etc. - i.e. anything that has an
<i>OWLEntity</i>	Represents Entities in the OWL 2 Specification.

## Module Contents

```
class owlapy.owl_object.HasIRI
```

Simple class to access the IRI.

```
__slots__ = ()
```

```
property iri: IRI
```

**Abstractmethod**

Gets the IRI of this object.

**Returns**

The IRI of this object.

```
property str: str
```

**Abstractmethod**

Gets the string representation of this object

**Returns**

The IRI as string

```
class owlapy.owl_object.OWLObject
```

Base interface for OWL objects

```
__slots__ = ()
```

```
abstract __eq__ (other)
```

Return self==value.

```
abstract __hash__ ()
```

Return hash(self).

```
abstract __repr__ ()
```

Return repr(self).

```
is_anonymous () → bool
```

```
class owlapy.owl_object.OWLObjectRenderer
```

Abstract class with a render method to render an OWL Object into a string.

```

abstract set_short_form_provider (short_form_provider) → None
    Configure a short form provider that shortens the OWL objects during rendering.

    Parameters
        short_form_provider – Short form provider.

abstract render (o: OWLObject) → str
    Render OWL Object to string.

    Parameters
        o – OWL Object.

    Returns
        String rendition of OWL object.

class owlapy.owl_object.OWLObjectParser
    Abstract class with a parse method to parse a string to an OWL Object.

    abstract parse_expression (expression_str: str) → OWLObject
        Parse a string to an OWL Object.

        Parameters
            expression_str (str) – Expression string.

        Returns
            The OWL Object which is represented by the string.

class owlapy.owl_object.OWLNamedObject
    Bases: OWLObject, owlapy.meta_classes.HasIRI
    Represents a named object for example, class, property, ontology etc. - i.e. anything that has an IRI as its name.

    __slots__ = ()

    __eq__ (other)
        Return self==value.

    __lt__ (other)
        Return self<value.

    __hash__ ()
        Return hash(self).

    __repr__ ()
        Return repr(self).

class owlapy.owl_object.OWLEntity
    Bases: OWLNamedObject
    Represents Entities in the OWL 2 Specification.

    __slots__ = ()

    to_string_id () → str

    is_anonymous () → bool

```

## owlapy.owl\_ontology

### OWL Ontology

#### Attributes

<i>IntegerOWLDatatype</i>
<i>DoubleOWLDatatype</i>
<i>BooleanOWLDatatype</i>
<i>StringOWLDatatype</i>
<i>DateOWLDatatype</i>
<i>DateTimeOWLDatatype</i>
<i>DurationOWLDatatype</i>
<i>OWLThing</i>
<i>logger</i>
<i>OWLREADY2_FACET_KEYS</i>

#### Classes

<i>OWLEquivalentClassesAxiom</i>	An equivalent classes axiom <code>EquivalentClasses( CE1 ... CEn )</code> states that all of the class expressions <code>CEi</code> ,
<i>OWLClassAxiom</i>	The base interface for class axioms.
<i>OWLDataPropertyDomainAxiom</i>	A data property domain axiom <code>DataPropertyDomain( DPE CE )</code> states that the domain of the
<i>OWLDataPropertyRangeAxiom</i>	A data property range axiom <code>DataPropertyRange( DPE DR )</code> states that the range of the data property
<i>OWLObjectPropertyDomainAxiom</i>	An object property domain axiom <code>ObjectPropertyDomain( OPE CE )</code> states that the domain of the
<i>OWLObjectPropertyRangeAxiom</i>	An object property range axiom <code>ObjectPropertyRange( OPE CE )</code> states that the range of the object property
<i>OWLSubClassOfAxiom</i>	A subclass axiom <code>SubClassOf( CE1 CE2 )</code> states that the class expression <code>CE1</code> is a subclass of the class
<i>OWLAnnotationProperty</i>	Represents an <code>AnnotationProperty</code> in the OWL 2 specification.
<i>OWLDatRange</i>	Represents a <code>DataRange</code> in the OWL 2 Specification.
<i>OWLDatComplementOf</i>	A complement data range <code>DataComplementOf( DR )</code> contains all tuples of literals that are not contained in the

continues on next page



Table 6 – continued from previous page

<i>OWLDataUnionOf</i>	A union data range <i>DataUnionOf</i> ( DR1 ... DRn ) contains all tuples of literals that are contained in the at least
<i>OWLDataIntersectionOf</i>	An intersection data range <i>DataIntersectionOf</i> ( DR1 ... DRn ) contains all tuples of literals that are contained
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
<i>OWLNamedIndividual</i>	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
<i>OWLIndividual</i>	Represents a named or anonymous individual.
<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
<i>OWLObject</i>	Base interface for OWL objects
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLObjectComplementOf</i>	Represents an <i>ObjectComplementOf</i> class expression in the OWL 2 Specification.
<i>OWLObjectUnionOf</i>	A union class expression <i>ObjectUnionOf</i> ( CE1 ... CEn ) contains all individuals that are instances
<i>OWLObjectIntersectionOf</i>	An intersection class expression <i>ObjectIntersectionOf</i> ( CE1 ... CEn ) contains all individuals that are instances
<i>OWLObjectSomeValuesFrom</i>	An existential class expression <i>ObjectSomeValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and
<i>OWLObjectAllValuesFrom</i>	A universal class expression <i>ObjectAllValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and a
<i>OWLObjectOneOf</i>	An enumeration of individuals <i>ObjectOneOf</i> ( a1 ... an ) contains exactly the individuals ai with $1 \leq i \leq n$ .
<i>OWLObjectExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLObjectMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLObjectMinCardinality</i>	A minimum cardinality expression <i>ObjectMinCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLObjectHasValue</i>	A has-value class expression <i>ObjectHasValue</i> ( OPE a ) consists of an object property expression OPE and an
<i>OWLDataSomeValuesFrom</i>	An existential class expression <i>DataSomeValuesFrom</i> ( DPE1 ... DPEn DR ) consists of n data property expressions
<i>OWLDataAllValuesFrom</i>	A universal class expression <i>DataAllValuesFrom</i> ( DPE1 ... DPEn DR ) consists of n data property expressions DPEi,
<i>OWLDataExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an
<i>OWLDataMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object

continues on next page

Table 6 – continued from previous page

<i>OWLDataMinCardinality</i>	A minimum cardinality expression <i>DataMinCardinality</i> ( <i>n</i> DPE DR ) consists of a nonnegative integer <i>n</i> , a data
<i>OWLDataHasValue</i>	A has-value class expression <i>DataHasValue</i> ( DPE <i>It</i> ) consists of a data property expression DPE and a literal <i>It</i> ,
<i>OWLDataOneOf</i>	An enumeration of literals <i>DataOneOf</i> ( <i>It</i> <sub>1</sub> ... <i>It</i> <sub><i>n</i></sub> ) contains exactly the explicitly specified literals <i>It</i> <sub><i>i</i></sub> with
<i>OWLDatatypeRestriction</i>	A datatype restriction <i>DatatypeRestriction</i> ( DT <i>F</i> <sub>1</sub> <i>It</i> <sub>1</sub> ... <i>F</i> <sub><i>n</i></sub> <i>It</i> <sub><i>n</i></sub> ) consists of a unary datatype DT and <i>n</i> pairs
<i>OWLRestriction</i>	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
<i>OWLObjectRestriction</i>	Represents an Object Property Restriction in the OWL 2 specification.
<i>OWLDataRestriction</i>	Represents a Data Property Restriction.
<i>OWLFacetRestriction</i>	A facet restriction is used to restrict a particular datatype.
<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
<i>OWLPropertyExpression</i>	Represents a property or possibly the inverse of a property.
<i>OWLObjectInverseOf</i>	Represents the inverse of a property expression ( <i>ObjectInverseOf</i> ). An inverse object property expression
<i>OWLObjectPropertyExpression</i>	A high level interface to describe different types of object properties.
<i>OWLDataPropertyExpression</i>	A high level interface to describe different types of data properties.
<i>OWLFacet</i>	Enumerations for OWL facets.
<i>OWLOntologyID</i>	An object that identifies an ontology. Since OWL 2, ontologies do not have to have an ontology IRI, or if they
<i>OWLOntology</i>	Represents an OWL 2 Ontology in the OWL 2 specification.
<i>Ontology</i>	Represents an OWL 2 Ontology in the OWL 2 specification.
<i>ToOwlready2</i>	
<i>FromOwlready2</i>	Map owlready2 classes to owlapy model classes.

## Module Contents

```
class owlapy.owl_ontology.OWLEquivalentClassesAxiom (
    class_expressions: List[owlapy.class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: OWLNaryClassAxiom

An equivalent classes axiom *EquivalentClasses*( *CE*<sub>1</sub> ... *CE*<sub>*n*</sub> ) states that all of the class expressions *CE*<sub>*i*</sub>,  $1 \leq i \leq n$ , are semantically equivalent to each other. This axiom allows one to use each *CE*<sub>*i*</sub> as a synonym for each *CE*<sub>*j*</sub> — that is, in any expression in the ontology containing such an axiom, *CE*<sub>*i*</sub> can be replaced with *CE*<sub>*j*</sub> without affecting the meaning of the ontology.

([https://www.w3.org/TR/owl2-syntax/#Equivalent\\_Classes](https://www.w3.org/TR/owl2-syntax/#Equivalent_Classes))

```
__slots__ = ()
```

**contains\_named\_equivalent\_class** () → bool

**contains\_owl\_nothing** () → bool

**contains\_owl\_thing** () → bool

**named\_classes** () → Iterable[owlapy.class\_expression.OWLClass]

```
class owlapy.owl_ontology.OWLClassAxiom(  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: OWLLogicalAxiom

The base interface for class axioms.

```
__slots__ = ()
```

```
class owlapy.owl_ontology.OWLDataPropertyDomainAxiom(  
    property_: owlapy.owl_property.OWLDataPropertyExpression,  
    domain: owlapy.class_expression.OWLClassExpression,  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: OWLPropertyDomainAxiom[owlapy.owl\_property.OWLDataPropertyExpression]

A data property domain axiom `DataPropertyDomain( DPE CE )` states that the domain of the data property expression DPE is the class expression CE — that is, if an individual *x* is connected by DPE with some literal, then *x* is an instance of CE. Each such axiom can be seen as a syntactic shortcut for the following axiom: `SubClassOf( DataSomeValuesFrom( DPE rdfs:Literal ) CE )`

([https://www.w3.org/TR/owl2-syntax/#Data\\_Property\\_Domain](https://www.w3.org/TR/owl2-syntax/#Data_Property_Domain))

```
__slots__ = ()
```

```
class owlapy.owl_ontology.OWLDataPropertyRangeAxiom(  
    property_: owlapy.owl_property.OWLDataPropertyExpression,  
    range_: owlapy.owl_datatype.OWLDataRange,  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: OWLPropertyRangeAxiom[owlapy.owl\_property.OWLDataPropertyExpression, owlapy.owl\_datatype.OWLDataRange]

A data property range axiom `DataPropertyRange( DPE DR )` states that the range of the data property expression DPE is the data range DR — that is, if some individual is connected by DPE with a literal *x*, then *x* is in DR. The arity of DR must be one. Each such axiom can be seen as a syntactic shortcut for the following axiom: `SubClassOf( owl:Thing DataAllValuesFrom( DPE DR ) )`

([https://www.w3.org/TR/owl2-syntax/#Data\\_Property\\_Range](https://www.w3.org/TR/owl2-syntax/#Data_Property_Range))

```
__slots__ = ()
```

```
class owlapy.owl_ontology.OWLObjectPropertyDomainAxiom(  
    property_: owlapy.owl_property.OWLObjectPropertyExpression,  
    domain: owlapy.class_expression.OWLClassExpression,  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: OWLPropertyDomainAxiom[owlapy.owl\_property.OWLObjectPropertyExpression]

An object property domain axiom `ObjectPropertyDomain( OPE CE )` states that the domain of the object property expression OPE is the class expression CE — that is, if an individual *x* is connected by OPE with some other individual, then *x* is an instance of CE. Each such axiom can be seen as a syntactic shortcut for the following axiom: `SubClassOf( ObjectSomeValuesFrom( OPE owl:Thing ) CE )`

([https://www.w3.org/TR/owl2-syntax/#Object\\_Property\\_Domain](https://www.w3.org/TR/owl2-syntax/#Object_Property_Domain))

```
__slots__ = ()
```

```
class owlapy.owl_ontology.OwlObjectPropertyRangeAxiom(  
    property_: owlapy.owl_property.OwlObjectPropertyExpression,  
    range_: owlapy.class_expression.OwlClassExpression,  
    annotations: Iterable[OwlAnnotation] | None = None)
```

```
Bases: OwlPropertyRangeAxiom[owlapy.owl_property.OwlObjectPropertyExpression,  
owlapy.class_expression.OwlClassExpression]
```

An object property range axiom `ObjectPropertyRange( OPE CE )` states that the range of the object property expression `OPE` is the class expression `CE` — that is, if some individual is connected by `OPE` with an individual `x`, then `x` is an instance of `CE`. Each such axiom can be seen as a syntactic shortcut for the following axiom: `SubClassOf( owl:Thing ObjectAllValuesFrom( OPE CE ) )`

([https://www.w3.org/TR/owl2-syntax/#Object\\_Property\\_Range](https://www.w3.org/TR/owl2-syntax/#Object_Property_Range))

```
__slots__ = ()
```

```
class owlapy.owl_ontology.OwlSubClassOfAxiom(  
    sub_class: owlapy.class_expression.OwlClassExpression,  
    super_class: owlapy.class_expression.OwlClassExpression,  
    annotations: Iterable[OwlAnnotation] | None = None)
```

```
Bases: OwlClassAxiom
```

A subclass axiom `SubClassOf( CE1 CE2 )` states that the class expression `CE1` is a subclass of the class expression `CE2`. Roughly speaking, this states that `CE1` is more specific than `CE2`. Subclass axioms are a fundamental type of axioms in OWL 2 and can be used to construct a class hierarchy. Other kinds of class expression axiom can be seen as syntactic shortcuts for one or more subclass axioms.

([https://www.w3.org/TR/owl2-syntax/#Subclass\\_Axioms](https://www.w3.org/TR/owl2-syntax/#Subclass_Axioms))

```
__slots__ = ('_sub_class', '_super_class')
```

```
get_sub_class() → owlapy.class_expression.OwlClassExpression
```

```
get_super_class() → owlapy.class_expression.OwlClassExpression
```

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
__repr__()
```

Return repr(self).

```
class owlapy.owl_ontology.OwlAnnotationProperty(iri: owlapy.iri.IRI | str)
```

```
Bases: owlapy.owl_property.OwlProperty
```

Represents an `AnnotationProperty` in the OWL 2 specification.

```
__slots__ = '_iri'
```

```
property iri: owlapy.iri.IRI
```

Gets the IRI of this object.

**Returns**

The IRI of this object.

```

property str: str
    Gets the string representation of this object

    Returns
        The IRI as string

class owlapy.owl_ontology.OWLDataRange
    Bases: OWLPropertyRange

    Represents a DataRange in the OWL 2 Specification.

class owlapy.owl_ontology.OWLDataComplementOf (data_range: OWLDataRange)
    Bases: OWLDataRange

    A complement data range DataComplementOf( DR ) contains all tuples of literals that are not contained in the
    data range DR. The resulting data range has the arity equal to the arity of DR.

    (https://www.w3.org/TR/owl2-syntax/#Complement\_of\_Data\_Ranges)

    type_index: Final = 4002

    get_data_range() → OWLDataRange

    Returns
        The wrapped data range.

    __repr__()
        Return repr(self).

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

class owlapy.owl_ontology.OWLDataUnionOf (operands: Iterable[OWLDataRange])
    Bases: OWLNaryDataRange

    A union data range DataUnionOf( DR1 ... DRn ) contains all tuples of literals that are contained in the at least one
    data range DRi for  $1 \leq i \leq n$ . All data ranges DRi must be of the same arity, and the resulting data range is of that
    arity as well.

    (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Data\_Ranges)

    __slots__ = '_operands'

    type_index: Final = 4005

class owlapy.owl_ontology.OWLDataIntersectionOf (operands: Iterable[OWLDataRange])
    Bases: OWLNaryDataRange

    An intersection data range DataIntersectionOf( DR1 ... DRn ) contains all tuples of literals that are contained in
    each data range DRi for  $1 \leq i \leq n$ . All data ranges DRi must be of the same arity, and the resulting data range is
    of that arity as well.

    (https://www.w3.org/TR/owl2-syntax/#Intersection\_of\_Data\_Ranges)

    __slots__ = '_operands'

    type_index: Final = 4004

```

```
class owlapy.owl_ontology.OWLDatatype(iri: owlapy.iri.IRI | owlapy.meta_classes.HasIRI)
    Bases: owlapy.owl_object.OWLEntity, owlapy.owl_data_ranges.OWLDataRange
```

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

```
__slots__ = '_iri'
```

```
type_index: Final = 4001
```

```
property iri: owlapy.iri.IRI
```

Gets the IRI of this object.

**Returns**

The IRI of this object.

```
property str: str
```

Gets the string representation of this object

**Returns**

The IRI as string

```
class owlapy.owl_ontology.OWLNamedIndividual(iri: owlapy.iri.IRI | str)
```

Bases: *OWLIndividual, owlapy.owl\_object.OWLEntity*

Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities. IRIs from the reserved vocabulary must not be used to identify named individuals in an OWL 2 DL ontology.

([https://www.w3.org/TR/owl2-syntax/#Named\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Named_Individuals))

```
__slots__ = '_iri'
```

```
type_index: Final = 1005
```

```
property iri: owlapy.iri.IRI
```

Gets the IRI of this object.

**Returns**

The IRI of this object.

```
property str
```

Gets the string representation of this object

**Returns**

The IRI as string

```
class owlapy.owl_ontology.OWLIndividual
```

Bases: *owlapy.owl\_object.OWLObject*

Represents a named or anonymous individual.

```
__slots__ = ()
```

```
owlapy.owl_ontology.IntegerOWLDatatype: Final
```

```
owlapy.owl_ontology.DoubleOWLDatatype: Final
```

```
owlapy.owl_ontology.BooleanOWLDatatype: Final
owlapy.owl_ontology.StringOWLDatatype: Final
owlapy.owl_ontology.DateOWLDatatype: Final
owlapy.owl_ontology.DateTimeOWLDatatype: Final
owlapy.owl_ontology.DurationOWLDatatype: Final
```

```
class owlapy.owl_ontology.OWLLiteral
```

Bases: *owlapy.owl\_annotation.OWLAnnotationValue*

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(<https://www.w3.org/TR/owl2-syntax/#Literals>)

```
__slots__ = ()
```

```
type_index: Final = 4008
```

```
get_literal() → str
```

Gets the lexical value of this literal. Note that the language tag is not included.

#### Returns

The lexical value of this literal.

```
is_boolean() → bool
```

Whether this literal is typed as boolean.

```
parse_boolean() → bool
```

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("<http://www.w3.org/2001/XMLSchema#boolean>").

#### Returns

A bool value that is represented by this literal.

```
is_double() → bool
```

Whether this literal is typed as double.

```
parse_double() → float
```

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

#### Returns

A double value that is represented by this literal.

```
is_integer() → bool
```

Whether this literal is typed as integer.

```
parse_integer() → int
```

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

#### Returns

An integer value that is represented by this literal.

**is\_string()** → bool

Whether this literal is typed as string.

**parse\_string()** → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("<http://www.w3.org/2001/XMLSchema#string>").

**Returns**

A string value that is represented by this literal.

**is\_date()** → bool

Whether this literal is typed as date.

**parse\_date()** → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("<http://www.w3.org/2001/XMLSchema#date>").

**Returns**

A date value that is represented by this literal.

**is\_datetime()** → bool

Whether this literal is typed as dateTime.

**parse\_datetime()** → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("<http://www.w3.org/2001/XMLSchema#dateTime>").

**Returns**

A datetime value that is represented by this literal.

**is\_duration()** → bool

Whether this literal is typed as duration.

**parse\_duration()** → pandas.Timedelta

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("<http://www.w3.org/2001/XMLSchema#duration>").

**Returns**

A Timedelta value that is represented by this literal.

**is\_literal()** → bool

**Returns**

true if the annotation value is a literal

**as\_literal()** → *OWLLiteral*

**Returns**

if the value is a literal, returns it. Return None otherwise

**to\_python()** → Literals

**abstract get\_datatype()** → *owlapy.owl\_datatype.OWLDatatype*

Gets the OWLDatatype which types this literal.

**Returns**

The OWLDatatype that types this literal.

**class owlapy.owl\_ontology.OWLObject**

Base interface for OWL objects



```

__slots__ = ()

abstract __eq__(other)
    Return self==value.

abstract __hash__()
    Return hash(self).

abstract __repr__()
    Return repr(self).

is_anonymous() → bool

class owlapy.owl_ontology.IRI(namespace: str | owlapy.namespaces.Namespaces, remainder: str)
    Bases: owlapy.owl_annotation.OWLAnnotationSubject, owlapy.owl_annotation.OWLAnnotationValue
    An IRI, consisting of a namespace and a remainder.

    __slots__ = ('_namespace', '_remainder', '__weakref__')

    type_index: Final = 0

    static create(namespace: owlapy.namespaces.Namespaces, remainder: str) → IRI
    static create(namespace: str, remainder: str) → IRI
    static create(string: str) → IRI

    __repr__()
        Return repr(self).

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    is_nothing()
        Determines if this IRI is equal to the IRI that owl:Nothing is named with.

        Returns
            True if this IRI is equal to <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.

    is_thing()
        Determines if this IRI is equal to the IRI that owl:Thing is named with.

        Returns
            True if this IRI is equal to <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

    is_reserved_vocabulary() → bool
        Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with
        <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

        Returns
            True if the IRI is in the reserved vocabulary, otherwise False.

    as_iri() → IRI

        Returns
            if the value is an IRI, return it. Return None otherwise.

```

**as\_str()** → str  
 CD: Should be deprecated. :returns: The string that specifies the IRI.

**property str:** str  
 Returns: The string that specifies the IRI.

**property reminder:** str  
 Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form()** → str  
 Gets the short form.

**Returns**  
 A string that represents the short form.

**get\_namespace()** → str  
**Returns**  
 The namespace as string.

**get\_remainder()** → str  
**Returns**  
 The remainder (coincident with NCName usually) for this IRI.

**class** owlapy.owl\_ontology.OWLClass(*iri: owlapy.iri.IRI | str*)  
 Bases: *owlapy.class\_expression.class\_expression.OWLClassExpression, owlapy.owl\_object.OWLEntity*  
 An OWL 2 named Class. Classes can be understood as sets of individuals. (<https://www.w3.org/TR/owl2-syntax/#Classes>)

**\_\_slots\_\_** = ('\_iri', '\_is\_nothing', '\_is\_thing')

**type\_index:** Final = 1001

**property iri:** *owlapy.iri.IRI*  
 Gets the IRI of this object.

**Returns**  
 The IRI of this object.

**property str**  
**Gets the string representation of this object**

**Returns**  
 The IRI as string

**property reminder:** str  
 The reminder of the IRI

**is\_owl\_thing()** → bool  
 Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**  
 Thing.

**Return type**  
 True if this expression is owl

**is\_owl\_nothing** () → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**get\_object\_complement\_of** ()

→ *owlapy.class\_expression.class\_expression.OwlObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf** () → *OWLClass*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

owlapy.owl\_ontology.OwlThing: **Final**

**class** owlapy.owl\_ontology.OwlClassExpression

Bases: *owlapy.owl\_data\_ranges.OwlPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

**\_\_slots\_\_** = ()

**abstract is\_owl\_thing** () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**abstract is\_owl\_nothing** () → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of** () → *OwlObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**abstract get\_nnf** () → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.owl\_ontology.OwlObjectComplementOf (*op*: *OWLClassExpression*)

Bases: OwlBooleanClassExpression, owlapy.meta\_classes.  
*HasOperands*[*OWLClassExpression*]

Represents an ObjectComplementOf class expression in the OWL 2 Specification.

```

__slots__ = '_operand'

type_index: Final = 3003

get_operand() → OWLClassExpression

    Returns
        The wrapped expression.

operands() → Iterable[OWLClassExpression]
    Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

    Returns
        The operands.

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

class owlapy.owl_ontology.OwlObjectUnionOf(
    operands: Iterable[owlapy.class_expression.class_expression.OwlClassExpression])
    Bases: OwlNaryBooleanClassExpression

    A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances of at least one
    class expression CEi for 1 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Class\_Expressions)

    __slots__ = '_operands'

    type_index: Final = 3002

class owlapy.owl_ontology.OwlObjectIntersectionOf(
    operands: Iterable[owlapy.class_expression.class_expression.OwlClassExpression])
    Bases: OwlNaryBooleanClassExpression

    An intersection class expression ObjectIntersectionOf( CE1 ... CEn ) contains all individuals that are instances of
    all class expressions CEi for 1 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Intersection\_of\_Class\_Expressions)

    __slots__ = '_operands'

    type_index: Final = 3001

class owlapy.owl_ontology.OwlObjectSomeValuesFrom(
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
    Bases: OwlQuantifiedObjectRestriction

    An existential class expression ObjectSomeValuesFrom( OPE CE ) consists of an object property expression OPE
    and a class expression CE, and it contains all those individuals that are connected by OPE to an individual that is
    an instance of CE.

    __slots__ = ('_property', '_filler')

    type_index: Final = 3005

```

```

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

get_property() → owlapy.owl_property.OwlObjectPropertyExpression

```

#### Returns

Property being restricted.

```

class owlapy.owl_ontology.OwlObjectAllValuesFrom(
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)

```

Bases: OwlQuantifiedObjectRestriction

A universal class expression ObjectAllValuesFrom( OPE CE ) consists of an object property expression OPE and a class expression CE, and it contains all those individuals that are connected by OPE only to individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification))

```

__slots__ = ('_property', '_filler')

type_index: Final = 3006

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

get_property() → owlapy.owl_property.OwlObjectPropertyExpression

```

#### Returns

Property being restricted.

```

class owlapy.owl_ontology.OwlObjectOneOf(
    values: owlapy.owl_individual.OwlIndividual | Iterable[owlapy.owl_individual.OwlIndividual])
Bases: owlapy.class_expression.class_expression.OwlAnonymousClassExpression,
owlapy.meta_classes.HasOperands[owlapy.owl_individual.OwlIndividual]

```

An enumeration of individuals ObjectOneOf( a1 ... an ) contains exactly the individuals ai with  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals))

```

__slots__ = '_values'

type_index: Final = 3004

individuals() → Iterable[owlapy.owl_individual.OwlIndividual]

```

Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this class expression.

#### Returns

The individuals that are the values of this {`@code` ObjectOneOf} class expression.

**operands** () → Iterable[*owlapy.owl\_individual.OWLIndividual*]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**as\_object\_union\_of** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

Simplifies this enumeration to a union of singleton nominals.

**Returns**

This enumeration in a more standard DL form.  $\text{simp}(\{a\}) = \{a\}$   $\text{simp}(\{a_0, \dots, \{a_n\}) = \text{unionOf}(\{a_0\}, \dots, \{a_n\})$

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

Return hash(self).

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

Return self==value.

**\_\_repr\_\_** ()

Return repr(self).

```
class owlapy.owl_ontology.OWLObjectExactCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: OWLObjectCardinalityRestriction

**An exact cardinality expression ObjectExactCardinality( n OPE CE ) consists of a nonnegative integer n, an object**

property expression OPE, and a class expression CE, and it contains all those individuals that are connected by to exactly n different individuals that are instances of CE.

([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

**\_\_slots\_\_** = ('\_cardinality', '\_filler', '\_property')

**type\_index:** Final = 3009

**as\_intersection\_of\_min\_max** ()

→ *owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

**Returns**

The semantically equivalent but structurally simpler form  $(= 1 R C) = \geq 1 R C$  and  $\leq 1 R C$ .

```
class owlapy.owl_ontology.OWLObjectMaxCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: OWLObjectCardinalityRestriction

**A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE**

to at most n different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

**\_\_slots\_\_** = ('\_cardinality', '\_filler', '\_property')

**type\_index:** Final = 3010

```
class owlapy.owl_ontology.OwlObjectMinCardinality (cardinality: int,
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: OwlObjectCardinalityRestriction

A minimum cardinality expression ObjectMinCardinality( n OPE CE ) consists of a nonnegative integer n, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to at least n different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3008
```

```
class owlapy.owl_ontology.OwlObjectHasValue (
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    individual: owlapy.owl_individual.OwlIndividual)
```

Bases: OwlHasValueRestriction[owlapy.owl\_individual.OwlIndividual], OwlObjectRestriction

A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ). ([https://www.w3.org/TR/owl2-syntax/#Individual\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Individual_Value_Restriction))

```
__slots__ = ('_property', '_v')
```

```
type_index: Final = 3007
```

```
get_property () → owlapy.owl_property.OwlObjectPropertyExpression
```

**Returns**

Property being restricted.

```
as_some_values_from () → owlapy.class_expression.class_expression.OwlClassExpression
```

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

**Returns**

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

```
__repr__ ()
```

Return repr(self).

```
class owlapy.owl_ontology.OwlDataSomeValuesFrom (
    property: owlapy.owl_property.OwlDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OwlDataRange)
```

Bases: OwlQuantifiedDataRestriction

An existential class expression DataSomeValuesFrom( DPE<sub>1</sub> ... DPE<sub>n</sub> DR ) consists of n data property expressions DPE<sub>i</sub>,  $1 \leq i \leq n$ , and a data range DR whose arity must be n. Such a class expression contains all those individuals that are connected by DPE<sub>i</sub> to literals lt<sub>i</sub>,  $1 \leq i \leq n$ , such that the tuple ( lt<sub>1</sub> , ..., lt<sub>n</sub> ) is in DR. A class expression of the form DataSomeValuesFrom( DPE DR ) can be seen as a syntactic shortcut for the class expression DataMinCardinality( 1 DPE DR ). ([https://www.w3.org/TR/owl2-syntax/#Existential\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Existential_Quantification_2))

```
__slots__ = '_property'
```

```
type_index: Final = 3012
```

```

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

get_property() → owlapy.owl_property.OWLDataPropertyExpression

```

#### Returns

Property being restricted.

```

class owlapy.owl_ontology.OWLDataAllValuesFrom (
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)

```

Bases: OWLQuantifiedDataRestriction

A universal class expression `DataAllValuesFrom( DPE1 ... DPEn DR )` consists of  $n$  data property expressions `DPEi`,  $1 \leq i \leq n$ , and a data range `DR` whose arity must be  $n$ . Such a class expression contains all those individuals that

**are connected by `DPEi` only to literals `lti`,  $1 \leq i \leq n$ , such that each tuple `( lt1 , ..., ltn )` is in `DR`.**

#### A class

expression of the form `DataAllValuesFrom( DPE DR )` can be seen as a syntactic shortcut for the class expression `DataMaxCardinality( 0 DPE DataComplementOf( DR ) )`. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification_2))

```

__slots__ = '_property'

type_index: Final = 3013

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

get_property() → owlapy.owl_property.OWLDataPropertyExpression

```

#### Returns

Property being restricted.

```

class owlapy.owl_ontology.OWLDataExactCardinality (cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)

```

Bases: OWLDataCardinalityRestriction

An exact cardinality expression `ObjectExactCardinality( n OPE CE )` consists of a nonnegative integer  $n$ , an object property expression `OPE`, and a class expression `CE`, and it contains all those individuals that are connected

by `OPE` to exactly  $n$  different individuals that are instances of `CE` ([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

```



**type\_index: Final = 3016**

**as\_intersection\_of\_min\_max()**

→ *owlapy.class\_expression.nary\_boolean\_expression.OwLObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

#### Returns

The semantically equivalent but structurally simpler form ( $= 1 \text{ R D}$ )  $= \geq 1 \text{ R D}$  and  $\leq 1 \text{ R D}$ .

```
class owlapy.owl_ontology.OwLDataMaxCardinality(cardinality: int,  
        property: owlapy.owl_property.OwLDataPropertyExpression,  
        filler: owlapy.owl_data_ranges.OwLDataRange)
```

Bases: *OwLDataCardinalityRestriction*

A maximum cardinality expression *ObjectMaxCardinality( n OPE CE )* consists of a nonnegative integer *n*, an object property expression *OPE*, and a class expression *CE*, and it contains all those individuals that are connected by *OPE* to at most *n* different individuals that are instances of *CE*. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

**\_\_slots\_\_ = ('\_cardinality', '\_filler', '\_property')**

**type\_index: Final = 3017**

```
class owlapy.owl_ontology.OwLDataMinCardinality(cardinality: int,  
        property: owlapy.owl_property.OwLDataPropertyExpression,  
        filler: owlapy.owl_data_ranges.OwLDataRange)
```

Bases: *OwLDataCardinalityRestriction*

A minimum cardinality expression *DataMinCardinality( n DPE DR )* consists of a nonnegative integer *n*, a data property expression *DPE*, and a unary data range *DR*, and it contains all those individuals that are connected by *DPE* to at least *n* different literals in *DR*. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

**\_\_slots\_\_ = ('\_cardinality', '\_filler', '\_property')**

**type\_index: Final = 3015**

```
class owlapy.owl_ontology.OwLDataHasValue(  
        property: owlapy.owl_property.OwLDataPropertyExpression,  
        value: owlapy.owl_literal.OwLLiteral)
```

Bases: *OwLHasValueRestriction[owlapy.owl\_literal.OwLLiteral], OwLDataRestriction*

A has-value class expression *DataHasValue( DPE lt )* consists of a data property expression *DPE* and a literal *lt*, and it contains all those individuals that are connected by *DPE* to *lt*. Each such class expression can be seen as a syntactic shortcut for the class expression *DataSomeValuesFrom( DPE DataOneOf( lt ) )*. ([https://www.w3.org/TR/owl2-syntax/#Literal\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Literal_Value_Restriction))

**\_\_slots\_\_ = '\_property'**

**type\_index: Final = 3014**

**\_\_repr\_\_()**

Return *repr(self)*.

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

Return *self==value*.

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

Return *hash(self)*.

**as\_some\_values\_from** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

**Returns**

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

**class** owlapy.owl\_ontology.OWLObjectOneOf (

*values:* *owlapy.owl\_literal.OWLLiteral* | *Iterable*[*owlapy.owl\_literal.OWLLiteral*])

Bases: *owlapy.owl\_data\_ranges.OWLDataRange*, *owlapy.meta\_classes.HasOperands*[*owlapy.owl\_literal.OWLLiteral*]

An enumeration of literals DataOneOf( *lt1* ... *ltn* ) contains exactly the explicitly specified literals *lti* with  $1 \leq i \leq n$ . The resulting data range has arity one. ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Literals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals))

**type\_index:** Final = 4003

**values** () → *Iterable*[*owlapy.owl\_literal.OWLLiteral*]

Gets the values that are in the oneOf.

**Returns**

The values of this {`DataOneOf`} class expression.

**operands** () → *Iterable*[*owlapy.owl\_literal.OWLLiteral*]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

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

Return hash(self).

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

Return self==value.

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.owl\_ontology.OWLDatatypeRestriction (

*type\_:* *owlapy.owl\_datatype.OWLDatatype*,

*facet\_restrictions:* *OWLFacetRestriction* | *Iterable*[*OWLFacetRestriction*])

Bases: *owlapy.owl\_data\_ranges.OWLDataRange*

A datatype restriction DatatypeRestriction( DT *F1* *lt1* ... *Fn* *ltn* ) consists of a unary datatype DT and *n* pairs ( *Fi* , *lti* ). The resulting data range is unary and is obtained by restricting the value space of DT according to the semantics of all ( *Fi* , *vi* ) (multiple pairs are interpreted conjunctively), where *vi* are the data values of the literals *lti*. ([https://www.w3.org/TR/owl2-syntax/#Datatype\\_Restrictions](https://www.w3.org/TR/owl2-syntax/#Datatype_Restrictions))

**\_\_slots\_\_** = ('\_type', '\_facet\_restrictions')

**type\_index:** Final = 4006

**get\_datatype** () → *owlapy.owl\_datatype.OWLDatatype*

**get\_facet\_restrictions** () → *Sequence*[*OWLFacetRestriction*]

```

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.owl_ontology.OWLRestriction
    Bases: owlapy.class_expression.class_expression.OWLAnonymousClassExpression
    Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.

    __slots__ = ()

    abstract get_property() → owlapy.owl_property.OWLPropertyExpression

        Returns
            Property being restricted.

    is_data_restriction() → bool
        Determines if this is a data restriction.

        Returns
            True if this is a data restriction.

    is_object_restriction() → bool
        Determines if this is an object restriction.

        Returns
            True if this is an object restriction.

class owlapy.owl_ontology.OWLObjectRestriction
    Bases: OWLRestriction
    Represents an Object Property Restriction in the OWL 2 specification.

    __slots__ = ()

    is_object_restriction() → bool
        Determines if this is an object restriction.

        Returns
            True if this is an object restriction.

    abstract get_property() → owlapy.owl_property.OWLObjectPropertyExpression

        Returns
            Property being restricted.

class owlapy.owl_ontology.OWLDataRestriction
    Bases: OWLRestriction
    Represents a Data Property Restriction.

    __slots__ = ()

```

**is\_data\_restriction**() → bool  
Determines if this is a data restriction.

**Returns**  
True if this is a data restriction.

**class** owlapy.owl\_ontology.**OWLFacetRestriction**(*facet: owlapy.vocab.OWLFacet*,  
*literal: Literals*)

Bases: *owlapy.owl\_object.OWLObject*

A facet restriction is used to restrict a particular datatype.

**\_\_slots\_\_** = ('\_facet', '\_literal')

**type\_index: Final** = 4007

**get\_facet**() → *owlapy.vocab.OWLFacet*

**get\_facet\_value**() → *owlapy.owl\_literal.OWLLiteral*

**\_\_eq\_\_**(*other*)  
Return self==value.

**\_\_hash\_\_**()  
Return hash(self).

**\_\_repr\_\_**()  
Return repr(self).

**class** owlapy.owl\_ontology.**OWLDataProperty**(*iri: owlapy.iri.IRI | str*)

Bases: *OWLDataPropertyExpression*, *OWLProperty*

Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.

([https://www.w3.org/TR/owl2-syntax/#Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Data_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index: Final** = 1004

**is\_owl\_top\_data\_property**() → bool  
Determines if this is the owl:topDataProperty.

**Returns**  
topDataProperty.

**Return type**  
True if this property is the owl

**class** owlapy.owl\_ontology.**OWLObjectProperty**(*iri: owlapy.iri.IRI | str*)

Bases: *OWLObjectPropertyExpression*, *OWLProperty*

Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.

([https://www.w3.org/TR/owl2-syntax/#Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Object_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index: Final** = 1002

**get\_named\_property** () → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**get\_inverse\_property** () → *OWLObjectInverseOf*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**is\_owl\_top\_object\_property** () → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

**class** owlapy.owl\_ontology.OWLPropertyExpression

Bases: *owlapy.owl\_object.OWLObject*

Represents a property or possibly the inverse of a property.

**\_\_slots\_\_** = ()

**is\_data\_property\_expression** () → bool

**Returns**

True if this is a data property.

**is\_object\_property\_expression** () → bool

**Returns**

True if this is an object property.

**is\_owl\_top\_object\_property** () → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

**is\_owl\_top\_data\_property** () → bool

Determines if this is the owl:topDataProperty.

**Returns**

topDataProperty.

**Return type**

True if this property is the owl

**class** owlapy.owl\_ontology.OWLObjectInverseOf (property: *OWLObjectProperty*)

Bases: *OWLObjectPropertyExpression*

Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression ObjectInverseOf( P ) connects an individual I1 with I2 if and only if the object property P connects I2 with I1. This can

be used to refer to the inverse of a property, without actually naming the property. For example, consider the property `hasPart`, the inverse property of `hasPart` (`isPartOf`) can be referred to using this interface `inverseOf(hasPart)`, which can be used in restrictions e.g. `inverseOf(hasPart) some Car` refers to the set of things that are part of at least one car.

([https://www.w3.org/TR/owl2-syntax/#Inverse\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Inverse_Object_Properties))

```
__slots__ = '_inverse_property'
```

```
type_index: Final = 1003
```

```
get_inverse () → OWLObjectProperty
```

Gets the property expression that this is the inverse of.

#### Returns

The object property expression such that this object property expression is an inverse of it.

```
get_inverse_property () → OWLObjectProperty
```

Obtains the property that corresponds to the inverse of this property.

#### Returns

The inverse of this property. Note that this property will not necessarily be in the simplest form.

```
get_named_property () → OWLObjectProperty
```

Get the named object property used in this property expression.

#### Returns

P if this expression is either `inv(P)` or `P`.

```
__repr__ ()
```

Return `repr(self)`.

```
__eq__ (other)
```

Return `self==value`.

```
__hash__ ()
```

Return `hash(self)`.

```
class owlapy.owl_ontology.OWLObjectPropertyExpression
```

Bases: *OWLPropertyExpression*

A high level interface to describe different types of object properties.

```
__slots__ = ()
```

```
abstract get_inverse_property () → OWLObjectPropertyExpression
```

Obtains the property that corresponds to the inverse of this property.

#### Returns

The inverse of this property. Note that this property will not necessarily be in the simplest form.

```
abstract get_named_property () → OWLObjectProperty
```

Get the named object property used in this property expression.

#### Returns

P if this expression is either `inv(P)` or `P`.

```
is_object_property_expression () → bool
```

#### Returns

True if this is an object property.

```
class owlapy.owl_ontology.OWLDataPropertyExpression
```

Bases: *OWLPropertyExpression*

A high level interface to describe different types of data properties.

```
__slots__ = ()
```

```
is_data_property_expression()
```

#### Returns

True if this is a data property.

```
class owlapy.owl_ontology.OWLFacet (remainder: str, symbolic_form: str,  
    operator: Callable[[X, X], bool])
```

Bases: *\_Vocabulary*, *enum.Enum*

Enumerations for OWL facets.

```
property symbolic_form
```

```
property operator
```

```
static from_str (name: str) → OWLFacet
```

```
MIN_INCLUSIVE: Final
```

```
MIN_EXCLUSIVE: Final
```

```
MAX_INCLUSIVE: Final
```

```
MAX_EXCLUSIVE: Final
```

```
LENGTH: Final
```

```
MIN_LENGTH: Final
```

```
MAX_LENGTH: Final
```

```
PATTERN: Final
```

```
TOTAL_DIGITS: Final
```

```
FRACTION_DIGITS: Final
```

```
owlapy.owl_ontology.logger
```

```
class owlapy.owl_ontology.OWLOntologyID (ontology_iri: owlapy.iri.IRI | None = None,  
    version_iri: owlapy.iri.IRI | None = None)
```

An object that identifies an ontology. Since OWL 2, ontologies do not have to have an ontology IRI, or if they have an ontology IRI then they can optionally also have a version IRI. Instances of this OWLOntologyID class bundle identifying information of an ontology together. If an ontology doesn't have an ontology IRI then we say that it is "anonymous".

```
__slots__ = ('_ontology_iri', '_version_iri')
```

```
get_ontology_iri() → owlapy.iri.IRI | None
```

Gets the ontology IRI.

#### Returns

Ontology IRI. If the ontology is anonymous, it will return None.

**get\_version\_iri** () → *owlapy.iri.IRI* | None

Gets the version IRI.

**Returns**

Version IRI or None.

**get\_default\_document\_iri** () → *owlapy.iri.IRI* | None

Gets the IRI which is used as a default for the document that contain a representation of an ontology with this ID. This will be the version IRI if there is an ontology IRI and version IRI, else it will be the ontology IRI if there is an ontology IRI but no version IRI, else it will be None if there is no ontology IRI. See Ontology Documents in the OWL 2 Structural Specification.

**Returns**

the IRI that can be used as a default for an ontology document, or None.

**is\_anonymous** () → bool

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

**class** owlapy.owl\_ontology.OWLontology

Bases: *owlapy.owl\_object.OWLObject*

Represents an OWL 2 Ontology in the OWL 2 specification.

An OWLontology consists of a possibly empty set of OWLAxioms and a possibly empty set of OWLAnnotations. An ontology can have an ontology IRI which can be used to identify the ontology. If it has an ontology IRI then it may also have an ontology version IRI. Since OWL 2, an ontology need not have an ontology IRI. (See the OWL 2 Structural Specification).

An ontology cannot be modified directly. Changes must be applied via its OWLontologyManager.

**\_\_slots\_\_** = ()

**type\_index**: Final = 1

**abstract classes\_in\_signature** () → Iterable[*owlapy.class\_expression.OWLClass*]

Gets the classes in the signature of this object.

**Returns**

Classes in the signature of this object.

**abstract data\_properties\_in\_signature** ()

→ Iterable[*owlapy.owl\_property.OWLDataProperty*]

Get the data properties that are in the signature of this object.

**Returns**

Data properties that are in the signature of this object.

**abstract object\_properties\_in\_signature** ()

→ Iterable[*owlapy.owl\_property.OWLObjectProperty*]

A convenience method that obtains the object properties that are in the signature of this object.

**Returns**

Object properties that are in the signature of this object.



```
abstract individuals_in_signature ()  
    → Iterable[owlapy.owl_individual.OWLNamedIndividual]
```

A convenience method that obtains the individuals that are in the signature of this object.

**Returns**

Individuals that are in the signature of this object.

```
abstract equivalent_classes_axioms (c: owlapy.class_expression.OWLClass)  
    → Iterable[owlapy.owl_axiom.OWLEquivalentClassesAxiom]
```

Gets all of the equivalent axioms in this ontology that contain the specified class as an operand.

**Parameters**

**c** – The class for which the EquivalentClasses axioms should be retrieved.

**Returns**

EquivalentClasses axioms contained in this ontology.

```
abstract general_class_axioms () → Iterable[owlapy.owl_axiom.OWLClassAxiom]
```

**Get the general class axioms of this ontology. This includes SubClass axioms with a complex class expression**

as the sub class and EquivalentClass axioms and DisjointClass axioms with only complex class expressions.

**Returns**

General class axioms contained in this ontology.

```
abstract data_property_domain_axioms (property: owlapy.owl_property.OWLDataProperty)  
    → Iterable[owlapy.owl_axiom.OWLDataPropertyDomainAxiom]
```

Gets the OWLDataPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract data_property_range_axioms (property: owlapy.owl_property.OWLDataProperty)  
    → Iterable[owlapy.owl_axiom.OWLDataPropertyRangeAxiom]
```

Gets the OWLDataPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract object_property_domain_axioms (  
    property: owlapy.owl_property.OWLObjectProperty)  
    → Iterable[owlapy.owl_axiom.OWLObjectPropertyDomainAxiom]
```

Gets the OWLObjectPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract object_property_range_axioms (
    property: owlapy.owl_property.OWLObjectProperty)
    → Iterable[owlapy.owl_axiom.OWLObjectPropertyRangeAxiom]
```

Gets the OWLObjectPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract get_owl_ontology_manager () → _M
```

Gets the manager that manages this ontology.

```
abstract get_ontology_id () → OWLOntologyID
```

Gets the OWLOntologyID belonging to this object.

**Returns**

The OWLOntologyID.

```
is_anonymous () → bool
```

Check whether this ontology does contain an IRI or not.

```
class owlapy.owl_ontology.Ontology (manager: OntologyManager, ontology_iri: owlapy.iri.IRI,
    load: bool)
```

Bases: *OWLOntology*

Represents an OWL 2 Ontology in the OWL 2 specification.

An OWLOntology consists of a possibly empty set of OWLAxioms and a possibly empty set of OWLAnnotations. An ontology can have an ontology IRI which can be used to identify the ontology. If it has an ontology IRI then it may also have an ontology version IRI. Since OWL 2, an ontology need not have an ontology IRI. (See the OWL 2 Structural Specification).

An ontology cannot be modified directly. Changes must be applied via its OWLOntologyManager.

```
__slots__ = ('_manager', '_iri', '_world', '_onto')
```

```
classes_in_signature () → Iterable[owlapy.class_expression.OWLClass]
```

Gets the classes in the signature of this object.

**Returns**

Classes in the signature of this object.

```
data_properties_in_signature () → Iterable[owlapy.owl_property.OWLDataProperty]
```

Get the data properties that are in the signature of this object.

**Returns**

Data properties that are in the signature of this object.

```
object_properties_in_signature () → Iterable[owlapy.owl_property.OWLObjectProperty]
```

A convenience method that obtains the object properties that are in the signature of this object.

**Returns**

Object properties that are in the signature of this object.

```
individuals_in_signature () → Iterable[owlapy.owl_individual.OWLNamedIndividual]
```

A convenience method that obtains the individuals that are in the signature of this object.

**Returns**

Individuals that are in the signature of this object.

**equivalent\_classes\_axioms** (*c*: *owlapy.class\_expression.OWLClass*)  
→ *Iterable[owlapy.owl\_axiom.OWLEquivalentClassesAxiom]*

Gets all of the equivalent axioms in this ontology that contain the specified class as an operand.

**Parameters**

**c** – The class for which the EquivalentClasses axioms should be retrieved.

**Returns**

EquivalentClasses axioms contained in this ontology.

**general\_class\_axioms** () → *Iterable[owlapy.owl\_axiom.OWLClassAxiom]*

**Get the general class axioms of this ontology. This includes SubClass axioms with a complex class expression**

as the sub class and EquivalentClass axioms and DisjointClass axioms with only complex class expressions.

**Returns**

General class axioms contained in this ontology.

**get\_owl\_ontology\_manager** () → *OntologyManager*

Gets the manager that manages this ontology.

**get\_ontology\_id** () → *OWLOntologyID*

Gets the OWLOntologyID belonging to this object.

**Returns**

The OWLOntologyID.

**data\_property\_domain\_axioms** (*pe*: *owlapy.owl\_property.OWLDataProperty*)  
→ *Iterable[owlapy.owl\_axiom.OWLDataPropertyDomainAxiom]*

Gets the OWLDataPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**data\_property\_range\_axioms** (*pe*: *owlapy.owl\_property.OWLDataProperty*)  
→ *Iterable[owlapy.owl\_axiom.OWLDataPropertyRangeAxiom]*

Gets the OWLDataPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**object\_property\_domain\_axioms** (*pe*: *owlapy.owl\_property.OWLObjectProperty*)  
→ *Iterable[owlapy.owl\_axiom.OWLObjectPropertyDomainAxiom]*

Gets the OWLObjectPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**object\_property\_range\_axioms** (*pe: owlapy.owl\_property.OWLObjectProperty*)  
→ Iterable[*owlapy.owl\_axiom.OWLObjectPropertyRangeAxiom*]

Gets the OWLObjectPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**get\_original\_iri** ()

Get the IRI argument that was used to create this ontology.

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

Return self==value.

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

Return hash(self).

**\_\_repr\_\_** ()

Return repr(self).

owlapy.owl\_ontology.OWLREADY2\_FACET\_KEYS

**class** owlapy.owl\_ontology.ToOwlready2 (*world: owlready2.World*)

**\_\_slots\_\_** = '\_world'

**abstract map\_object** (*o: owlapy.owl\_object.OWLObject*)

Map owlapy object classes.

**abstract map\_concept** (*o: owlapy.class\_expression.OWLClassExpression*)

→ owlready2.ClassConstruct | owlready2.ThingClass

Map owlapy concept classes.

**abstract map\_datarange** (*p: owlapy.owl\_data\_ranges.OWLDataRange*)

→ owlready2.ClassConstruct | type

Map owlapy data range classes.

**class** owlapy.owl\_ontology.FromOwlready2

Map owlready2 classes to owlapy model classes.

**\_\_slots\_\_** = ()

**abstract map\_concept** (*c: owlready2.ClassConstruct | owlready2.ThingClass*)

→ owlapy.class\_expression.OWLClassExpression

Map concept classes.

**abstract map\_datarange** (*p: owlready2.ClassConstruct*)

→ owlapy.owl\_data\_ranges.OWLDataRange

Map data range classes.

## Attributes

---

*OWLThing*


---

## Classes

<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>HasIRI</i>	Simple class to access the IRI.
<i>OWLObject</i>	Base interface for OWL objects
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>OWLQuantifiedDataRestriction</i>	Represents a quantified data restriction.
<i>OWLDataHasValue</i>	A has-value class expression <code>DataHasValue( DPE It )</code> consists of a data property expression DPE and a literal It,
<i>OWLNaryBooleanClassExpression</i>	<code>OWLNaryBooleanClassExpression</code> .
<i>OWLObjectOneOf</i>	An enumeration of individuals <code>ObjectOneOf( a1 ... an )</code> contains exactly the individuals $a_i$ with $1 \leq i \leq n$ .
<i>OWLObjectComplementOf</i>	Represents an <code>ObjectComplementOf</code> class expression in the OWL 2 Specification.
<i>OWLObjectHasValue</i>	A has-value class expression <code>ObjectHasValue( OPE a )</code> consists of an object property expression OPE and an
<i>OWLQuantifiedObjectRestriction</i>	Represents a quantified object restriction.
<i>OWLObjectPropertyRangeAxiom</i>	An object property range axiom <code>ObjectPropertyRange( OPE CE )</code> states that the range of the object property
<i>OWLAxiom</i>	Represents Axioms in the OWL 2 Specification.
<i>OWLSubClassOfAxiom</i>	A subclass axiom <code>SubClassOf( CE1 CE2 )</code> states that the class expression CE1 is a subclass of the class
<i>OWLEquivalentClassesAxiom</i>	An equivalent classes axiom <code>EquivalentClasses( CE1 ... CEn )</code> states that all of the class expressions $CE_i$ ,
<i>OWLDisjointUnionAxiom</i>	A disjoint union axiom <code>DisjointUnion( C CE1 ... CEn )</code> states that a class C is a disjoint union of the class
<i>OWLAnnotationAssertionAxiom</i>	An annotation assertion <code>AnnotationAssertion( AP as av )</code> states that the annotation subject as — an IRI or an
<i>OWLAnnotationProperty</i>	Represents an <code>AnnotationProperty</code> in the OWL 2 specification.
<i>OWLSubPropertyAxiom</i>	Base interface for object and data sub-property axioms.
<i>OWLPropertyRangeAxiom</i>	Base class for Property Range axioms.
<i>OWLClassAssertionAxiom</i>	A class assertion <code>ClassAssertion( CE a )</code> states that the individual a is an instance of the class expression CE.
<i>OWLDeclarationAxiom</i>	Represents a Declaration axiom in the OWL 2 Specification. A declaration axiom declares an entity in an ontology.
<i>OWLObjectPropertyAssertionAxiom</i>	A positive object property assertion <code>ObjectPropertyAssertion( OPE a1 a2 )</code> states that the individual a1 is

continues on next page

Table 7 – continued from previous page

<i>OWLSymmetricObjectPropertyAxiom</i>	An object property symmetry axiom <i>SymmetricObjectProperty</i> ( OPE ) states that
<i>OWLTransitiveObjectPropertyAxiom</i>	An object property transitivity axiom <i>TransitiveObjectProperty</i> ( OPE ) states that the
<i>OWLPropertyDomainAxiom</i>	Base class for Property Domain axioms.
<i>OWLAsymmetricObjectPropertyAxiom</i>	An object property asymmetry axiom <i>AsymmetricObjectProperty</i> ( OPE ) states that
<i>OWLDataPropertyCharacteristicAxiom</i>	Base interface for Functional data property axiom.
<i>OWLFunctionalDataPropertyAxiom</i>	A data property functionality axiom <i>FunctionalDataProperty</i> ( DPE ) states that
<i>OWLReflexiveObjectPropertyAxiom</i>	An object property reflexivity axiom <i>ReflexiveObjectProperty</i> ( OPE ) states that the
<i>OWLDataPropertyAssertionAxiom</i>	A positive data property assertion <i>DataPropertyAssertion</i> ( DPE a lt ) states that the individual a is connected
<i>OWLFunctionalObjectPropertyAxiom</i>	An object property functionality axiom <i>FunctionalObjectProperty</i> ( OPE ) states that
<i>OWLObjectPropertyCharacteristicAxiom</i>	Base interface for functional object property axiom.
<i>OWLIrreflexiveObjectPropertyAxiom</i>	An object property irreflexivity axiom <i>IrreflexiveObjectProperty</i> ( OPE ) states that the
<i>OWLInverseFunctionalObjectPropertyAxiom</i>	An object property inverse functionality axiom <i>InverseFunctionalObjectProperty</i> ( OPE )
<i>OWLDisjointDataPropertiesAxiom</i>	A disjoint data properties axiom <i>DisjointDataProperties</i> ( DPE1 ... DPE <sub>n</sub> ) states that all of the data property
<i>OWLDisjointObjectPropertiesAxiom</i>	A disjoint object properties axiom <i>DisjointObjectProperties</i> ( OPE1 ... OPE <sub>n</sub> ) states that all of the object
<i>OWLEquivalentDataPropertiesAxiom</i>	An equivalent data properties axiom <i>EquivalentDataProperties</i> ( DPE1 ... DPE <sub>n</sub> ) states that all the data property
<i>OWLEquivalentObjectPropertiesAxiom</i>	An equivalent object properties axiom <i>EquivalentObjectProperties</i> ( OPE1 ... OPE <sub>n</sub> ) states that all of the object
<i>OWLInverseObjectPropertiesAxiom</i>	An inverse object properties axiom <i>InverseObjectProperties</i> ( OPE1 OPE2 ) states that the object property
<i>OWLNaryPropertyAxiom</i>	Represents an axiom that contains two or more operands that could also be represented with
<i>OWLNaryIndividualAxiom</i>	Represents an axiom that contains two or more operands that could also be represented with
<i>OWLDifferentIndividualsAxiom</i>	An individual inequality axiom <i>DifferentIndividuals</i> ( a1 ... a <sub>n</sub> ) states that all of the individuals a <sub>i</sub> ,
<i>OWLDisjointClassesAxiom</i>	A disjoint classes axiom <i>DisjointClasses</i> ( CE1 ... CE <sub>n</sub> ) states that all of the class expressions CE <sub>i</sub> , 1 ≤ i ≤ n,
<i>OWLSameIndividualAxiom</i>	An individual equality axiom <i>SameIndividual</i> ( a1 ... a <sub>n</sub> ) states that all of the individuals a <sub>i</sub> , 1 ≤ i ≤ n,
<i>OWLNamedIndividual</i>	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
<i>OWLIndividual</i>	Represents a named or anonymous individual.
<i>OWLOntology</i>	Represents an OWL 2 Ontology in the OWL 2 specification.
<i>Ontology</i>	Represents an OWL 2 Ontology in the OWL 2 specification.
<i>ToOwlready2</i>	
<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.

continues on next page

Table 7 – continued from previous page

<i>OWLObjectInverseOf</i>	Represents the inverse of a property expression (Object-InverseOf). An inverse object property expression
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
<i>OWLProperty</i>	A base class for properties that aren't expression i.e. named properties. By definition, properties
<i>OWLOntologyChange</i>	Represents an ontology change.
<i>OWLOntologyManager</i>	An OWLOntologyManager manages a set of ontologies. It is the main point for creating, loading and accessing
<i>OWLImportsDeclaration</i>	Represents an import statement in an ontology.
<i>AddImport</i>	Represents an ontology change where an import statement is added to an ontology.
<i>OntologyManager</i>	An OWLOntologyManager manages a set of ontologies. It is the main point for creating, loading and accessing

## Module Contents

```
class owlapy.owl_ontology_manager.IRI (namespace: str | owlapy.namespaces.Namespaces,
    remainder: str)

Bases:    owlapy.owl_annotation.OWLAnnotationSubject,    owlapy.owl_annotation.
    OWLAnnotationValue

An IRI, consisting of a namespace and a remainder.

__slots__ = ('_namespace', '_remainder', '__weakref__')

type_index: Final = 0

static create (namespace: owlapy.namespaces.Namespaces, remainder: str) → IRI
static create (namespace: str, remainder: str) → IRI
static create (string: str) → IRI

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

is_nothing ()
    Determines if this IRI is equal to the IRI that owl:Nothing is named with.

Returns
    True if this IRI is equal to <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.

is_thing ()
    Determines if this IRI is equal to the IRI that owl:Thing is named with.

Returns
    True if this IRI is equal to <http://www.w3.org/2002/07/owl#Thing> and otherwise False.
```

**is\_reserved\_vocabulary** () → bool

Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

**Returns**

True if the IRI is in the reserved vocabulary, otherwise False.

**as\_iri** () → *IRI*

**Returns**

if the value is an IRI, return it. Return None otherwise.

**as\_str** () → str

CD: Should be deprecated. :returns: The string that specifies the IRI.

**property str: str**

Returns: The string that specifies the IRI.

**property reminder: str**

Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form** () → str

Gets the short form.

**Returns**

A string that represents the short form.

**get\_namespace** () → str

**Returns**

The namespace as string.

**get\_remainder** () → str

**Returns**

The remainder (coincident with NCName usually) for this IRI.

**class** owlapy.owl\_ontology\_manager.**HasIRI**

Simple class to access the IRI.

**\_\_slots\_\_** = ()

**property iri:** *IRI*

**Abstractmethod**

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str: str**

**Abstractmethod**

Gets the string representation of this object

**Returns**

The IRI as string



```

class owlapy.owl_ontology_manager.OWLObject
    Base interface for OWL objects

    __slots__ = ()

    abstract __eq__(other)
        Return self==value.

    abstract __hash__()
        Return hash(self).

    abstract __repr__()
        Return repr(self).

    is_anonymous() → bool

owlapy.owl_ontology_manager.OWLThing: Final

class owlapy.owl_ontology_manager.OWLClass(iri: owlapy.iri.IRI | str)
    Bases: owlapy.class_expression.class_expression.OWLClassExpression, owlapy.owl_object.OWLEntity

    An OWL 2 named Class. Classes can be understood as sets of individuals. (https://www.w3.org/TR/owl2-syntax/#Classes)

    __slots__ = ('_iri', '_is_nothing', '_is_thing')

    type_index: Final = 1001

    property iri: owlapy.iri.IRI
        Gets the IRI of this object.

        Returns
            The IRI of this object.

    property str
        Gets the string representation of this object

        Returns
            The IRI as string

    property reminder: str
        The reminder of the IRI

    is_owl_thing() → bool
        Determines if this expression is the built in class owl:Thing. This method does not determine if the class is
        equivalent to owl:Thing.

        Returns
            Thing.

        Return type
            True if this expression is owl

    is_owl_nothing() → bool
        Determines if this expression is the built in class owl:Nothing. This method does not determine if the class
        is equivalent to owl:Nothing.

```

**get\_object\_complement\_of()**

→ *owlapy.class\_expression.class\_expression.OwLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf()** → *OWLClass*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.owl\_ontology\_manager.OwLQuantifiedDataRestriction(  
    *filler: owlapy.owl\_data\_ranges.OwLDataRange*)

Bases: OwLQuantifiedRestriction[*owlapy.owl\_data\_ranges.OwLDataRange*], OwLDataRestriction

Represents a quantified data restriction.

**\_\_slots\_\_** = ()

**get\_filler()** → *owlapy.owl\_data\_ranges.OwLDataRange*

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

**Returns**

the value

**class** owlapy.owl\_ontology\_manager.OwLDataHasValue(  
    *property: owlapy.owl\_property.OwLDataPropertyExpression*,  
    *value: owlapy.owl\_literal.OwLLiteral*)

Bases: OwLHasValueRestriction[*owlapy.owl\_literal.OwLLiteral*], OwLDataRestriction

A has-value class expression DataHasValue( DPE It ) consists of a data property expression DPE and a literal It, and it contains all those individuals that are connected by DPE to It. Each such class expression can be seen as a syntactic shortcut for the class expression DataSomeValuesFrom( DPE DataOneOf( It ) ). ([https://www.w3.org/TR/owl2-syntax/#Literal\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Literal_Value_Restriction))

**\_\_slots\_\_** = '\_property'

**type\_index: Final** = 3014

**\_\_repr\_\_()**

Return repr(self).

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

Return self==value.

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

Return hash(self).

**as\_some\_values\_from()** → *owlapy.class\_expression.class\_expression.OwLClassExpression*

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

**Returns**

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

**class** owlapy.owl\_ontology\_manager.**OWLNaryBooleanClassExpression** (  
    *operands: Iterable[owlapy.class\_expression.class\_expression.OWLClassExpression]*)

Bases: *owlapy.class\_expression.class\_expression.OWLBooleanClassExpression*,  
*owlapy.meta\_classes.HasOperands[owlapy.class\_expression.class\_expression.OWLClassExpression]*

OWLNaryBooleanClassExpression.

**\_\_slots\_\_** = ()

**operands** () → *Iterable[owlapy.class\_expression.class\_expression.OWLClassExpression]*

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**class** owlapy.owl\_ontology\_manager.**OWLObjectOneOf** (  
    *values: owlapy.owl\_individual.OWLIndividual | Iterable[owlapy.owl\_individual.OWLIndividual]*)

Bases: *owlapy.class\_expression.class\_expression.OWLAnonymousClassExpression*,  
*owlapy.meta\_classes.HasOperands[owlapy.owl\_individual.OWLIndividual]*

An enumeration of individuals ObjectOneOf( *a*<sub>1</sub> ... *a*<sub>*n*</sub> ) contains exactly the individuals *a*<sub>*i*</sub> with 1 ≤ *i* ≤ *n*. ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals))

**\_\_slots\_\_** = '**\_values**'

**type\_index**: **Final** = 3004

**individuals** () → *Iterable[owlapy.owl\_individual.OWLIndividual]*

Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this class expression.

**Returns**

The individuals that are the values of this {`@code` ObjectOneOf} class expression.

**operands** () → *Iterable[owlapy.owl\_individual.OWLIndividual]*

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**as\_object\_union\_of** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

Simplifies this enumeration to a union of singleton nominals.

**Returns**

This enumeration in a more standard DL form.  $\text{simp}(\{a\}) = \{a\}$   $\text{simp}(\{a_0, \dots, \{a_n\}\}) = \text{unionOf}(\{a_0\}, \dots, \{a_n\})$

```

__hash__()
    Return hash(self).

__eq__(other)
    Return self==value.

__repr__()
    Return repr(self).

class owlapy.owl_ontology_manager.OwlObjectComplementOf (op: OwlClassExpression)
    Bases: OwlBooleanClassExpression, owlapy.meta_classes.
    HasOperands[OwlClassExpression]

    Represents an ObjectComplementOf class expression in the OWL 2 Specification.

    __slots__ = '_operand'

    type_index: Final = 3003

    get_operand() → OwlClassExpression

        Returns
        The wrapped expression.

    operands() → Iterable[OwlClassExpression]

        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
        The operands.

    __repr__()
        Return repr(self).

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

class owlapy.owl_ontology_manager.OwlObjectHasValue (
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    individual: owlapy.owl_individual.OwlIndividual)
    Bases: OwlHasValueRestriction[owlapy.owl_individual.OwlIndividual], OwlObjectRestriction

    A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ). (https://www.w3.org/TR/owl2-syntax/#Individual\_Value\_Restriction)

    __slots__ = ('_property', '_v')

    type_index: Final = 3007

    get_property() → owlapy.owl_property.OwlObjectPropertyExpression

        Returns
        Property being restricted.

```

**as\_some\_values\_from()** → *owlapy.class\_expression.class\_expression.OWLClassExpression*

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

#### Returns

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

**\_\_repr\_\_()**

Return repr(self).

**class** owlapy.owl\_ontology\_manager.OWLQuantifiedObjectRestriction(  
    *filler: owlapy.class\_expression.class\_expression.OWLClassExpression*)

Bases: OWLQuantifiedRestriction[*owlapy.class\_expression.class\_expression.OWLClassExpression*], OWLObjectRestriction

Represents a quantified object restriction.

**\_\_slots\_\_** = ()

**get\_filler()** → *owlapy.class\_expression.class\_expression.OWLClassExpression*

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

#### Returns

the value

**class** owlapy.owl\_ontology\_manager.OWLObjectPropertyRangeAxiom(  
    *property\_: owlapy.owl\_property.OWLObjectPropertyExpression,*  
    *range\_: owlapy.class\_expression.OWLClassExpression,*  
    *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: OWLPropertyRangeAxiom[*owlapy.owl\_property.OWLObjectPropertyExpression*, *owlapy.class\_expression.OWLClassExpression*]

An object property range axiom ObjectPropertyRange( OPE CE ) states that the range of the object property expression OPE is the class expression CE — that is, if some individual is connected by OPE with an individual x, then x is an instance of CE. Each such axiom can be seen as a syntactic shortcut for the following axiom: SubClassOf( owl:Thing ObjectAllValuesFrom( OPE CE ) )

([https://www.w3.org/TR/owl2-syntax/#Object\\_Property\\_Range](https://www.w3.org/TR/owl2-syntax/#Object_Property_Range))

**\_\_slots\_\_** = ()

**class** owlapy.owl\_ontology\_manager.OWLAxiom(  
    *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *owlapy.owl\_object.OWLObject*

Represents Axioms in the OWL 2 Specification.

An OWL ontology contains a set of axioms. These axioms can be annotation axioms, declaration axioms, imports axioms or logical axioms.

**\_\_slots\_\_** = '\_annotations'

**annotations()** → List[*OWLAnnotation*] | None

**is\_annotated()** → bool

**is\_logical\_axiom()** → bool

**is\_annotation\_axiom()** → bool

```

class owlapy.owl_ontology_manager.OWLSubClassOfAxiom (
    sub_class: owlapy.class_expression.OWLClassExpression,
    super_class: owlapy.class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)

```

Bases: OWLClassAxiom

A subclass axiom SubClassOf( CE1 CE2 ) states that the class expression CE1 is a subclass of the class expression CE2. Roughly speaking, this states that CE1 is more specific than CE2. Subclass axioms are a fundamental type of axioms in OWL 2 and can be used to construct a class hierarchy. Other kinds of class expression axiom can be seen as syntactic shortcuts for one or more subclass axioms.

([https://www.w3.org/TR/owl2-syntax/#Subclass\\_Axioms](https://www.w3.org/TR/owl2-syntax/#Subclass_Axioms))

```

__slots__ = ('_sub_class', '_super_class')

```

```

get_sub_class() → owlapy.class_expression.OWLClassExpression

```

```

get_super_class() → owlapy.class_expression.OWLClassExpression

```

```

__eq__(other)

```

Return self==value.

```

__hash__()

```

Return hash(self).

```

__repr__()

```

Return repr(self).

```

class owlapy.owl_ontology_manager.OWLEquivalentClassesAxiom (
    class_expressions: List[owlapy.class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

```

Bases: OWLNaryClassAxiom

An equivalent classes axiom EquivalentClasses( CE1 ... CEn ) states that all of the class expressions CEi,  $1 \leq i \leq n$ , are semantically equivalent to each other. This axiom allows one to use each CEi as a synonym for each CEj — that is, in any expression in the ontology containing such an axiom, CEi can be replaced with CEj without affecting the meaning of the ontology.

([https://www.w3.org/TR/owl2-syntax/#Equivalent\\_Classes](https://www.w3.org/TR/owl2-syntax/#Equivalent_Classes))

```

__slots__ = ()

```

```

contains_named_equivalent_class() → bool

```

```

contains_owl_nothing() → bool

```

```

contains_owl_thing() → bool

```

```

named_classes() → Iterable[owlapy.class_expression.OWLClass]

```

```

class owlapy.owl_ontology_manager.OWLDisjointUnionAxiom (
    cls_: owlapy.class_expression.OWLClass,
    class_expressions: List[owlapy.class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

```

Bases: OWLClassAxiom

A disjoint union axiom DisjointUnion( C CE1 ... CEn ) states that a class C is a disjoint union of the class expressions CEi,  $1 \leq i \leq n$ , all of which are pairwise disjoint. Such axioms are sometimes referred to as covering axioms, as they state that the extensions of all CEi exactly cover the extension of C. Thus, each instance of C is an instance of exactly one CEi, and each instance of CEi is an instance of C.

([https://www.w3.org/TR/owl2-syntax/#Disjoint\\_Union\\_of\\_Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Disjoint_Union_of_Class_Expressions))

```
__slots__ = ('_cls', '_class_expressions')
```

```
get_owl_class() → owlapy.class_expression.OWLClass
```

```
get_class_expressions() → Iterable[owlapy.class_expression.OWLClassExpression]
```

```
get_owl_equivalent_classes_axiom() → OWLEquivalentClassesAxiom
```

```
get_owl_disjoint_classes_axiom() → OWLDisjointClassesAxiom
```

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
__repr__()
```

Return repr(self).

```
class owlapy.owl_ontology_manager.OWLAnnotationAssertionAxiom(  
    subject: owlapy.owl_annotation.OWLAnnotationSubject, annotation: OWLAnnotation)
```

Bases: OWLAnnotationAxiom

An annotation assertion AnnotationAssertion( AP as av ) states that the annotation subject as — an IRI or an anonymous individual — is annotated with the annotation property AP and the annotation value av.

([https://www.w3.org/TR/owl2-syntax/#Annotation\\_Assertion](https://www.w3.org/TR/owl2-syntax/#Annotation_Assertion))

```
__slots__ = ('_subject', '_annotation')
```

```
get_subject() → owlapy.owl_annotation.OWLAnnotationSubject
```

Gets the subject of this object.

**Returns**

The subject.

```
get_property() → OWLAnnotationProperty
```

Gets the property.

**Returns**

The property.

```
get_value() → owlapy.owl_annotation.OWLAnnotationValue
```

Gets the annotation value. This is either an IRI, an OWLAnonymousIndividual or an OWLLiteral.

**Returns**

The annotation value.

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
__repr__()
```

Return repr(self).

```

class owlapy.owl_ontology_manager.OWLAnnotationProperty (iri: owlapy.iri.IRI | str)
    Bases: owlapy.owl_property.OWLProperty
    Represents an AnnotationProperty in the OWL 2 specification.
    __slots__ = '_iri'
    property iri: owlapy.iri.IRI
        Gets the IRI of this object.
        Returns
            The IRI of this object.
    property str: str
        Gets the string representation of this object
        Returns
            The IRI as string
class owlapy.owl_ontology_manager.OWLSubPropertyAxiom (sub_property: _P,
    super_property: _P, annotations: Iterable[OWLAnnotation] | None = None)
    Bases: Generic[_P], OWLPropertyAxiom
    Base interface for object and data sub-property axioms.
    __slots__ = ('_sub_property', '_super_property')
    get_sub_property () → _P
    get_super_property () → _P
    __eq__ (other)
        Return self==value.
    __hash__ ()
        Return hash(self).
    __repr__ ()
        Return repr(self).
class owlapy.owl_ontology_manager.OWLPropertyRangeAxiom (property_: _P, range_: _R,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: Generic[_P, _R], OWLUnaryPropertyAxiom[_P]
    Base class for Property Range axioms.
    __slots__ = '_range'
    get_range () → _R
    __eq__ (other)
        Return self==value.
    __hash__ ()
        Return hash(self).
    __repr__ ()
        Return repr(self).

```



```

class owlapy.owl_ontology_manager.OWLClassAssertionAxiom (
    individual: owlapy.owl_individual.OWLIndividual,
    class_expression: owlapy.class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)

```

Bases: OWLIndividualAxiom

A class assertion ClassAssertion( CE a ) states that the individual a is an instance of the class expression CE.

([https://www.w3.org/TR/owl2-syntax/#Class\\_Assertions](https://www.w3.org/TR/owl2-syntax/#Class_Assertions))

```

__slots__ = ('_individual', '_class_expression')

```

```

get_individual () → owlapy.owl_individual.OWLIndividual

```

```

get_class_expression () → owlapy.class_expression.OWLClassExpression

```

```

__eq__ (other)

```

Return self==value.

```

__hash__ ()

```

Return hash(self).

```

__repr__ ()

```

Return repr(self).

```

class owlapy.owl_ontology_manager.OWLDeclarationAxiom (
    entity: owlapy.owl_object.OWLEntity, annotations: Iterable[OWLAnnotation] | None = None)

```

Bases: OWLAxiom

Represents a Declaration axiom in the OWL 2 Specification. A declaration axiom declares an entity in an ontology. It doesn't affect the logical meaning of the ontology.

```

__slots__ = '_entity'

```

```

get_entity () → owlapy.owl_object.OWLEntity

```

```

__eq__ (other)

```

Return self==value.

```

__hash__ ()

```

Return hash(self).

```

__repr__ ()

```

Return repr(self).

```

class owlapy.owl_ontology_manager.OWLObjectPropertyAssertionAxiom (
    subject: owlapy.owl_individual.OWLIndividual,
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    object_: owlapy.owl_individual.OWLIndividual,
    annotations: Iterable[OWLAnnotation] | None = None)

```

Bases: OWLPropertyAssertionAxiom[owlapy.owl\_property.OWLObjectPropertyExpression, owlapy.owl\_individual.OWLIndividual]

A positive object property assertion ObjectPropertyAssertion( OPE a1 a2 ) states that the individual a1 is connected by the object property expression OPE to the individual a2.

([https://www.w3.org/TR/owl2-syntax/#Positive\\_Object\\_Property\\_Assertions](https://www.w3.org/TR/owl2-syntax/#Positive_Object_Property_Assertions))

```

__slots__ = ()

```

```
class owlapy.owl_ontology_manager.OWLSymmetricObjectPropertyAxiom (
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLObjectPropertyCharacteristicAxiom*

An object property symmetry axiom SymmetricObjectProperty( OPE ) states that the object property expression OPE is symmetric — that is, if an individual x is connected by OPE to an individual y, then y is also connected by OPE to x. Each such axiom can be seen as a syntactic shortcut for the following axiom:

SubObjectPropertyOf( OPE ObjectInverseOf( OPE ) )

([https://www.w3.org/TR/owl2-syntax/#Symmetric\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Symmetric_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_ontology_manager.OWLTransitiveObjectPropertyAxiom (
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLObjectPropertyCharacteristicAxiom*

An object property transitivity axiom TransitiveObjectProperty( OPE ) states that the object property expression OPE is transitive — that is, if an individual x is connected by OPE to an individual y that is connected by OPE to an individual z, then x is also connected by OPE to z. Each such axiom can be seen as a syntactic shortcut for the following axiom: SubObjectPropertyOf( ObjectPropertyChain( OPE OPE ) OPE )

([https://www.w3.org/TR/owl2-syntax/#Transitive\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Transitive_Object_Properties))

```
__slots__ = ()
```

```
class owlapy.owl_ontology_manager.OWLPropertyDomainAxiom (property_: _P,
    domain: owlapy.class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: Generic[\_P], OWLUnaryPropertyAxiom[\_P]

Base class for Property Domain axioms.

```
__slots__ = '_domain'
```

```
get_domain() → owlapy.class_expression.OWLClassExpression
```

```
__eq__ (other)
```

Return self==value.

```
__hash__ ()
```

Return hash(self).

```
__repr__ ()
```

Return repr(self).

```
class owlapy.owl_ontology_manager.OWLAsymmetricObjectPropertyAxiom (
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLObjectPropertyCharacteristicAxiom*

An object property asymmetry axiom AsymmetricObjectProperty( OPE ) states that the object property expression OPE is asymmetric — that is, if an individual x is connected by OPE to an individual y, then y cannot be connected by OPE to x.

([https://www.w3.org/TR/owl2-syntax/#Symmetric\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Symmetric_Object_Properties))

```

__slots__ = ()

class owlapy.owl_ontology_manager.OWLDataPropertyCharacteristicAxiom(
    property_: owlapy.owl_property.OWLDataPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLUnaryPropertyAxiom[owlapy.owl_property.OWLDataPropertyExpression],
        OWLDataPropertyAxiom

Base interface for Functional data property axiom.

__slots__ = ()

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.owl_ontology_manager.OWLFunctionalDataPropertyAxiom(
    property_: owlapy.owl_property.OWLDataPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLDataPropertyCharacteristicAxiom

A data property functionality axiom FunctionalDataProperty( DPE ) states that the data property expression DPE
is functional — that is, for each individual x, there can be at most one distinct literal y such that x is connected by
DPE with y. Each such axiom can be seen as a syntactic shortcut for the following axiom: SubClassOf( owl:Thing
DataMaxCardinality( 1 DPE ) )

(https://www.w3.org/TR/owl2-syntax/#Transitive\_Object\_Properties)

__slots__ = ()

class owlapy.owl_ontology_manager.OWLReflexiveObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLObjectPropertyCharacteristicAxiom

An object property reflexivity axiom ReflexiveObjectProperty( OPE ) states that the object property expression
OPE is reflexive — that is, each individual is connected by OPE to itself. Each such axiom can be seen as a
syntactic shortcut for the following axiom: SubClassOf( owl:Thing ObjectHasSelf( OPE ) )

(https://www.w3.org/TR/owl2-syntax/#Reflexive\_Object\_Properties)

__slots__ = ()

class owlapy.owl_ontology_manager.OWLDataPropertyAssertionAxiom(
    subject: owlapy.owl_individual.OWLIndividual,
    property_: owlapy.owl_property.OWLDataPropertyExpression,
    object_: owlapy.owl_literal.OWLLiteral, annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLPropertyAssertionAxiom[owlapy.owl_property.OWLDataPropertyExpression,
        owlapy.owl_literal.OWLLiteral]

A positive data property assertion DataPropertyAssertion( DPE a lt ) states that the individual a is connected by
the data property expression DPE to the literal lt.

(https://www.w3.org/TR/owl2-syntax/#Positive\_Data\_Property\_Assertions)

```

```

__slots__ = ()

class owlapy.owl_ontology_manager.OWLFunctionalObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLObjectPropertyCharacteristicAxiom

    An object property functionality axiom FunctionalObjectProperty( OPE ) states that the object property expression
    OPE is functional — that is, for each individual x, there can be at most one distinct individual y such that x is
    connected by OPE to y.

    (https://www.w3.org/TR/owl2-syntax/#Functional\_Object\_Properties)

__slots__ = ()

class owlapy.owl_ontology_manager.OWLObjectPropertyCharacteristicAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLUnaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
    OWLObjectPropertyAxiom

    Base interface for functional object property axiom.

__slots__ = ()

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.owl_ontology_manager.OWLIrreflexiveObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLObjectPropertyCharacteristicAxiom

    An object property irreflexivity axiom IrreflexiveObjectProperty( OPE ) states that the object property expression
    OPE is irreflexive — that is, no individual is connected by OPE to itself.

    (https://www.w3.org/TR/owl2-syntax/#Irreflexive\_Object\_Properties)

__slots__ = ()

class owlapy.owl_ontology_manager.OWLInverseFunctionalObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLObjectPropertyCharacteristicAxiom

    An object property inverse functionality axiom InverseFunctionalObjectProperty( OPE ) states that the object
    property expression OPE is inverse-functional — that is, for each individual x, there can be at most one individual
    y such that y is connected by OPE with x.

    (https://www.w3.org/TR/owl2-syntax/#Inverse-Functional\_Object\_Properties)

__slots__ = ()

```

```

class owlapy.owl_ontology_manager.OWLDisjointDataPropertiesAxiom(
    properties: List[owlapy.owl_property.OWLDataPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLDataPropertyExpression],
        OWLDataPropertyAxiom

A disjoint data properties axiom DisjointDataProperties( DPE1 ... DPEn ) states that all of the data property
expressions DPEi, 1 ≤ i ≤ n, are pairwise disjoint; that is, no individual x can be connected to a literal y by both
    DPEi and DPEj for i ≠ j.

(https://www.w3.org/TR/owl2-syntax/#Disjoint\_Data\_Properties)

__slots__ = ()

class owlapy.owl_ontology_manager.OWLDisjointObjectPropertiesAxiom(
    properties: List[owlapy.owl_property.OWLObjectPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
        OWLObjectPropertyAxiom

A disjoint object properties axiom DisjointObjectProperties( OPE1 ... OPEn ) states that all of the object property
expressions OPEi, 1 ≤ i ≤ n, are pairwise disjoint; that is, no individual x can be connected to an individual y by
both OPEi and OPEj for i ≠ j.

(https://www.w3.org/TR/owl2-syntax/#Disjoint\_Object\_Properties)

__slots__ = ()

class owlapy.owl_ontology_manager.OWLEquivalentDataPropertiesAxiom(
    properties: List[owlapy.owl_property.OWLDataPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLDataPropertyExpression],
        OWLDataPropertyAxiom

An equivalent data properties axiom EquivalentDataProperties( DPE1 ... DPEn ) states that all the data property
expressions DPEi, 1 ≤ i ≤ n, are semantically equivalent to each other. This axiom allows one to use each DPEi
as a synonym for each DPEj — that is, in any expression in the ontology containing such an axiom, DPEi can be
replaced with DPEj without affecting the meaning of the ontology.

(https://www.w3.org/TR/owl2-syntax/#Equivalent\_Data\_Properties)

__slots__ = ()

class owlapy.owl_ontology_manager.OWLEquivalentObjectPropertiesAxiom(
    properties: List[owlapy.owl_property.OWLObjectPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
        OWLObjectPropertyAxiom

An equivalent object properties axiom EquivalentObjectProperties( OPE1 ... OPEn ) states that all of the object
property expressions OPEi, 1 ≤ i ≤ n, are semantically equivalent to each other. This axiom allows one to use each
OPEi as a synonym for each OPEj — that is, in any expression in the ontology containing such an axiom, OPEi
can be replaced with OPEj without affecting the meaning of the ontology.

(https://www.w3.org/TR/owl2-syntax/#Equivalent\_Object\_Properties)

__slots__ = ()

```

```

class owlapy.owl_ontology_manager.OWLInverseObjectPropertiesAxiom(
    first: owlapy.owl_property.OWLObjectPropertyExpression,
    second: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
        OWLObjectPropertyAxiom

An inverse object properties axiom InverseObjectProperties( OPE1 OPE2 ) states that the object property expres-
sion OPE1 is an inverse of the object property expression OPE2. Thus, if an individual x is connected by OPE1
to an individual y, then y is also connected by OPE2 to x, and vice versa.

(https://www.w3.org/TR/owl2-syntax/#Inverse\_Object\_Properties\_2)

__slots__ = ('_first', '_second')

get_first_property() → owlapy.owl_property.OWLObjectPropertyExpression

get_second_property() → owlapy.owl_property.OWLObjectPropertyExpression

__repr__()
    Return repr(self).

class owlapy.owl_ontology_manager.OWLNaryPropertyAxiom(properties: List[_P],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: Generic[_P], OWLPropertyAxiom, OWLNaryAxiom[_P]

Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
property axioms.

__slots__ = '_properties'

properties() → Iterable[_P]
    Get all the properties that appear in the axiom.

    Returns
    Generator containing the properties.

as_pairwise_axioms() → Iterable[OWLNaryPropertyAxiom]

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.owl_ontology_manager.OWLNaryIndividualAxiom(
    individuals: List[owlapy.owl_individual.OWLIndividual],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLIndividualAxiom, OWLNaryAxiom[owlapy.owl_individual.OWLIndividual]

Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
individual axioms.

__slots__ = '_individuals'

```

**individuals** () → Iterable[*owlapy.owl\_individual.OWLIndividual*]

Get the individuals.

**Returns**

Generator containing the individuals.

**as\_pairwise\_axioms** () → Iterable[*OWLNaryIndividualAxiom*]

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

Return self==value.

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

Return hash(self).

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.owl\_ontology\_manager.**OWLDifferentIndividualsAxiom** (  
    *individuals*: List[*owlapy.owl\_individual.OWLIndividual*],  
    *annotations*: Iterable[*OWLAnnotation*] | None = None)

Bases: *OWLNaryIndividualAxiom*

An individual inequality axiom **DifferentIndividuals**( *a1* ... *an* ) states that all of the individuals *ai*,  $1 \leq i \leq n$ , are different from each other; that is, no individuals *ai* and *aj* with  $i \neq j$  can be derived to be equal. This axiom can be used to axiomatize the unique name assumption — the assumption that all different individual names denote different individuals. ([https://www.w3.org/TR/owl2-syntax/#Individual\\_Inequality](https://www.w3.org/TR/owl2-syntax/#Individual_Inequality))

**\_\_slots\_\_** = ()

**class** owlapy.owl\_ontology\_manager.**OWLDisjointClassesAxiom** (  
    *class\_expressions*: List[*owlapy.class\_expression.OWLClassExpression*],  
    *annotations*: Iterable[*OWLAnnotation*] | None = None)

Bases: *OWLNaryClassAxiom*

A disjoint classes axiom **DisjointClasses**( *CE1* ... *CEn* ) states that all of the class expressions *CEi*,  $1 \leq i \leq n$ , are pairwise disjoint; that is, no individual can be at the same time an instance of both *CEi* and *CEj* for  $i \neq j$ .

([https://www.w3.org/TR/owl2-syntax/#Disjoint\\_Classes](https://www.w3.org/TR/owl2-syntax/#Disjoint_Classes))

**\_\_slots\_\_** = ()

**class** owlapy.owl\_ontology\_manager.**OWLSameIndividualAxiom** (  
    *individuals*: List[*owlapy.owl\_individual.OWLIndividual*],  
    *annotations*: Iterable[*OWLAnnotation*] | None = None)

Bases: *OWLNaryIndividualAxiom*

An individual equality axiom **SameIndividual**( *a1* ... *an* ) states that all of the individuals *ai*,  $1 \leq i \leq n$ , are equal to each other. This axiom allows one to use each *ai* as a synonym for each *aj* — that is, in any expression in the ontology containing such an axiom, *ai* can be replaced with *aj* without affecting the meaning of the ontology.

([https://www.w3.org/TR/owl2-syntax/#Individual\\_Equality](https://www.w3.org/TR/owl2-syntax/#Individual_Equality))

**\_\_slots\_\_** = ()

**class** owlapy.owl\_ontology\_manager.**OWLNamedIndividual** (*iri*: *owlapy.iri.IRI* | str)

Bases: *OWLIndividual*, *owlapy.owl\_object.OWLEntity*

Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities. IRIs from the reserved vocabulary must not be used to identify named individuals in an OWL 2 DL ontology.

([https://www.w3.org/TR/owl2-syntax/#Named\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Named_Individuals))

```

__slots__ = '_iri'

type_index: Final = 1005

property iri: owlapy.iri.IRI
    Gets the IRI of this object.

    Returns
        The IRI of this object.

property str
    Gets the string representation of this object

    Returns
        The IRI as string

```

**class** *owlapy.owl\_ontology\_manager.OWLIndividual*  
 Bases: *owlapy.owl\_object.OWLObject*  
 Represents a named or anonymous individual.

```

__slots__ = ()

```

**class** *owlapy.owl\_ontology\_manager.OWLOntology*  
 Bases: *owlapy.owl\_object.OWLObject*  
 Represents an OWL 2 Ontology in the OWL 2 specification.

An OWLOntology consists of a possibly empty set of OWLAxioms and a possibly empty set of OWLAnnotations. An ontology can have an ontology IRI which can be used to identify the ontology. If it has an ontology IRI then it may also have an ontology version IRI. Since OWL 2, an ontology need not have an ontology IRI. (See the OWL 2 Structural Specification).

An ontology cannot be modified directly. Changes must be applied via its OWLOntologyManager.

```

__slots__ = ()

type_index: Final = 1

abstract classes_in_signature() → Iterable[owlapy.class_expression.OWLClass]
    Gets the classes in the signature of this object.

    Returns
        Classes in the signature of this object.

abstract data_properties_in_signature()
    → Iterable[owlapy.owl_property.OWLDataProperty]
    Get the data properties that are in the signature of this object.

    Returns
        Data properties that are in the signature of this object.

abstract object_properties_in_signature()
    → Iterable[owlapy.owl_property.OWLObjectProperty]
    A convenience method that obtains the object properties that are in the signature of this object.

    Returns
        Object properties that are in the signature of this object.

```



```
abstract individuals_in_signature ()  
    → Iterable[owlapy.owl_individual.OWLNamedIndividual]
```

A convenience method that obtains the individuals that are in the signature of this object.

**Returns**

Individuals that are in the signature of this object.

```
abstract equivalent_classes_axioms (c: owlapy.class_expression.OWLClass)  
    → Iterable[owlapy.owl_axiom.OWLEquivalentClassesAxiom]
```

Gets all of the equivalent axioms in this ontology that contain the specified class as an operand.

**Parameters**

**c** – The class for which the EquivalentClasses axioms should be retrieved.

**Returns**

EquivalentClasses axioms contained in this ontology.

```
abstract general_class_axioms () → Iterable[owlapy.owl_axiom.OWLClassAxiom]
```

**Get the general class axioms of this ontology. This includes SubClass axioms with a complex class expression**

as the sub class and EquivalentClass axioms and DisjointClass axioms with only complex class expressions.

**Returns**

General class axioms contained in this ontology.

```
abstract data_property_domain_axioms (property: owlapy.owl_property.OWLDataProperty)  
    → Iterable[owlapy.owl_axiom.OWLDataPropertyDomainAxiom]
```

Gets the OWLDataPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract data_property_range_axioms (property: owlapy.owl_property.OWLDataProperty)  
    → Iterable[owlapy.owl_axiom.OWLDataPropertyRangeAxiom]
```

Gets the OWLDataPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract object_property_domain_axioms (  
    property: owlapy.owl_property.OWLObjectProperty)  
    → Iterable[owlapy.owl_axiom.OWLObjectPropertyDomainAxiom]
```

Gets the OWLObjectPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract object_property_range_axioms (
    property: owlapy.owl_property.OWLObjectProperty)
    → Iterable[owlapy.owl_axiom.OWLObjectPropertyRangeAxiom]
```

Gets the OWLObjectPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract get_owl_ontology_manager () → _M
```

Gets the manager that manages this ontology.

```
abstract get_ontology_id () → OWLOntologyID
```

Gets the OWLOntologyID belonging to this object.

**Returns**

The OWLOntologyID.

```
is_anonymous () → bool
```

Check whether this ontology does contain an IRI or not.

```
class owlapy.owl_ontology_manager.Ontology (manager: OntologyManager,
    ontology_iri: owlapy.iri.IRI, load: bool)
```

Bases: *OWLOntology*

Represents an OWL 2 Ontology in the OWL 2 specification.

An OWLOntology consists of a possibly empty set of OWLAxioms and a possibly empty set of OWLAnnotations. An ontology can have an ontology IRI which can be used to identify the ontology. If it has an ontology IRI then it may also have an ontology version IRI. Since OWL 2, an ontology need not have an ontology IRI. (See the OWL 2 Structural Specification).

An ontology cannot be modified directly. Changes must be applied via its OWLOntologyManager.

```
__slots__ = ('_manager', '_iri', '_world', '_onto')
```

```
classes_in_signature () → Iterable[owlapy.class_expression.OWLClass]
```

Gets the classes in the signature of this object.

**Returns**

Classes in the signature of this object.

```
data_properties_in_signature () → Iterable[owlapy.owl_property.OWLDataProperty]
```

Get the data properties that are in the signature of this object.

**Returns**

Data properties that are in the signature of this object.

```
object_properties_in_signature () → Iterable[owlapy.owl_property.OWLObjectProperty]
```

A convenience method that obtains the object properties that are in the signature of this object.

**Returns**

Object properties that are in the signature of this object.

```
individuals_in_signature () → Iterable[owlapy.owl_individual.OWLNamedIndividual]
```

A convenience method that obtains the individuals that are in the signature of this object.

**Returns**

Individuals that are in the signature of this object.

**equivalent\_classes\_axioms** (*c*: *owlapy.class\_expression.OWLClass*)  
→ *Iterable[owlapy.owl\_axiom.OWLEquivalentClassesAxiom]*

Gets all of the equivalent axioms in this ontology that contain the specified class as an operand.

**Parameters**

**c** – The class for which the EquivalentClasses axioms should be retrieved.

**Returns**

EquivalentClasses axioms contained in this ontology.

**general\_class\_axioms** () → *Iterable[owlapy.owl\_axiom.OWLClassAxiom]*

**Get the general class axioms of this ontology. This includes SubClass axioms with a complex class expression**

as the sub class and EquivalentClass axioms and DisjointClass axioms with only complex class expressions.

**Returns**

General class axioms contained in this ontology.

**get\_owl\_ontology\_manager** () → *OntologyManager*

Gets the manager that manages this ontology.

**get\_ontology\_id** () → *OWLOntologyID*

Gets the OWLOntologyID belonging to this object.

**Returns**

The OWLOntologyID.

**data\_property\_domain\_axioms** (*pe*: *owlapy.owl\_property.OWLDataProperty*)  
→ *Iterable[owlapy.owl\_axiom.OWLDataPropertyDomainAxiom]*

Gets the OWLDataPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**data\_property\_range\_axioms** (*pe*: *owlapy.owl\_property.OWLDataProperty*)  
→ *Iterable[owlapy.owl\_axiom.OWLDataPropertyRangeAxiom]*

Gets the OWLDataPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**object\_property\_domain\_axioms** (*pe*: *owlapy.owl\_property.OWLObjectProperty*)  
→ *Iterable[owlapy.owl\_axiom.OWLObjectPropertyDomainAxiom]*

Gets the OWLObjectPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**object\_property\_range\_axioms** (*pe: owlapy.owl\_property.OWLObjectProperty*)  
 → Iterable[*owlapy.owl\_axiom.OWLObjectPropertyRangeAxiom*]

Gets the OWLObjectPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**get\_original\_iri** ()

Get the IRI argument that was used to create this ontology.

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

Return self==value.

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

Return hash(self).

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.owl\_ontology\_manager.**ToOwlready2** (*world: owlready2.World*)

**\_\_slots\_\_** = **'\_world'**

**abstract map\_object** (*o: owlapy.owl\_object.OWLObject*)

Map owlapy object classes.

**abstract map\_concept** (*o: owlapy.class\_expression.OWLClassExpression*)

→ owlready2.ClassConstruct | owlready2.ThingClass

Map owlapy concept classes.

**abstract map\_data\_range** (*p: owlapy.owl\_data\_ranges.OWLDataRange*)

→ owlready2.ClassConstruct | type

Map owlapy data range classes.

**class** owlapy.owl\_ontology\_manager.**OWLDataProperty** (*iri: owlapy.iri.IRI | str*)

Bases: OWLDataPropertyExpression, *OWLProperty*

Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.

([https://www.w3.org/TR/owl2-syntax/#Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Data_Properties))

**\_\_slots\_\_** = **'\_iri'**

**type\_index: Final** = **1004**

**is\_owl\_top\_data\_property** () → bool

Determines if this is the owl:topDataProperty.

**Returns**

topDataProperty.

**Return type**

True if this property is the owl

```
class owlapy.owl_ontology_manager.OWLObjectInverseOf (property: OWLObjectProperty)
```

Bases: OWLObjectPropertyExpression

Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression ObjectInverseOf( P ) connects an individual I1 with I2 if and only if the object property P connects I2 with I1. This can be used to refer to the inverse of a property, without actually naming the property. For example, consider the property hasPart, the inverse property of hasPart (isPartOf) can be referred to using this interface inverseOf(hasPart), which can be used in restrictions e.g. inverseOf(hasPart) some Car refers to the set of things that are part of at least one car.

([https://www.w3.org/TR/owl2-syntax/#Inverse\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Inverse_Object_Properties))

```
__slots__ = '_inverse_property'
```

```
type_index: Final = 1003
```

```
get_inverse () → OWLObjectProperty
```

Gets the property expression that this is the inverse of.

**Returns**

The object property expression such that this object property expression is an inverse of it.

```
get_inverse_property () → OWLObjectProperty
```

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

```
get_named_property () → OWLObjectProperty
```

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

```
__repr__ ()
```

Return repr(self).

```
__eq__ (other)
```

Return self==value.

```
__hash__ ()
```

Return hash(self).

```
class owlapy.owl_ontology_manager.OWLObjectProperty (iri: owlapy.iri.IRI | str)
```

Bases: OWLObjectPropertyExpression, *OWLProperty*

Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.

([https://www.w3.org/TR/owl2-syntax/#Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Object_Properties))

```
__slots__ = '_iri'
```

```
type_index: Final = 1002
```

```
get_named_property () → OWLObjectProperty
```

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**get\_inverse\_property** () → *OWLObjectInverseOf*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**is\_owl\_top\_object\_property** () → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

**class** owlapy.owl\_ontology\_manager.**OWLProperty** (*iri: owlapy.iri.IRI | str*)

Bases: OWLPropertyExpression, *owlapy.owl\_object.OWLEntity*

A base class for properties that aren't expression i.e. named properties. By definition, properties are either data properties or object properties.

**\_\_slots\_\_** = '\_iri'

**property str: str**

Gets the string representation of this object

**Returns**

The IRI as string

**property iri: owlapy.iri.IRI**

Gets the IRI of this object.

**Returns**

The IRI of this object.

**class** owlapy.owl\_ontology\_manager.**OWLOntologyChange** (

*ontology: owlapy.owl\_ontology.OWLOntology*)

Represents an ontology change.

**\_\_slots\_\_** = ()

**get\_ontology** () → *owlapy.owl\_ontology.OWLOntology*

Gets the ontology that the change is/was applied to.

**Returns**

The ontology that the change is applicable to.

**class** owlapy.owl\_ontology\_manager.**OWLOntologyManager**

An OWLOntologyManager manages a set of ontologies. It is the main point for creating, loading and accessing ontologies.

**abstract create\_ontology** (*iri: owlapy.iri.IRI*) → *owlapy.owl\_ontology.OWLOntology*

Creates a new (empty) ontology that has the specified ontology IRI (and no version IRI).

**Parameters**

**iri** – The IRI of the ontology to be created.

**Returns**

The newly created ontology, or if an ontology with the specified IRI already exists then this existing ontology will be returned.

**abstract load\_ontology** (*iri: owlapy.iri.IRI*) → *owlapy.owl\_ontology.OWLOntology*

Loads an ontology that is assumed to have the specified ontology IRI as its IRI or version IRI. The ontology IRI will be mapped to an ontology document IRI.

**Parameters**

**iri** – The IRI that identifies the ontology. It is expected that the ontology will also have this IRI (although the OWL API should tolerate situations where this is not the case).

**Returns**

The OWLOntology representation of the ontology that was loaded.

**abstract apply\_change** (*change: OWLOntologyChange*)

A convenience method that applies just one change to an ontology. When this method is used through an OWLOntologyManager implementation, the instance used should be the one that the ontology returns through the get\_owl\_ontology\_manager() call.

**Parameters**

**change** – The change to be applied.

**Raises**

**ChangeApplied.UNSUCCESSFULLY** – if the change was not applied successfully.

**abstract add\_axiom** (*ontology: owlapy.owl\_ontology.OWLOntology*,  
*axiom: owlapy.owl\_axiom.OWLXiom*)

A convenience method that adds a single axiom to an ontology.

**Parameters**

- **ontology** – The ontology to add the axiom to.
- **axiom** – The axiom to be added.

**abstract remove\_axiom** (*ontology: owlapy.owl\_ontology.OWLOntology*,  
*axiom: owlapy.owl\_axiom.OWLXiom*)

A convenience method that removes a single axiom from an ontology.

**Parameters**

- **ontology** – The ontology to remove the axiom from.
- **axiom** – The axiom to be removed.

**abstract save\_ontology** (*ontology: owlapy.owl\_ontology.OWLOntology*,  
*document\_iri: owlapy.iri.IRI*)

Saves the specified ontology, using the specified document IRI to determine where/how the ontology should be saved.

**Parameters**

- **ontology** – The ontology to be saved.
- **document\_iri** – The document IRI where the ontology should be saved to.

**class owlapy.owl\_ontology\_manager.OWLImportsDeclaration** (*import\_iri: owlapy.iri.IRI*)

Bases: *owlapy.meta\_classes.HasIRI*

Represents an import statement in an ontology.

**\_\_slots\_\_** = **'\_iri'**

**property** `iri`: *owlapy.iri.IRI*

Gets the import IRI.

**Returns**

The import IRI that points to the ontology to be imported. The imported ontology might have this IRI as its ontology IRI but this is not mandated. For example, an ontology with a non-resolvable ontology IRI can be deployed at a resolvable URL.

**property** `str`: `str`

Gets the string representation of this object

**Returns**

The IRI as string

**class** `owlapy.owl_ontology_manager.AddImport` (*ontology*: *owlapy.owl\_ontology.OWL ontology*,  
*import\_declaration*: *OWLImportsDeclaration*)

Bases: *OWL ontologyChange*

Represents an ontology change where an import statement is added to an ontology.

`__slots__` = ('\_ont', '\_declaration')

`get_import_declaration` () → *OWLImportsDeclaration*

Gets the import declaration that the change pertains to.

**Returns**

The import declaration.

**class** `owlapy.owl_ontology_manager.OntologyManager` (*world\_store*=None)

Bases: *OWL ontologyManager*

An *OWL ontologyManager* manages a set of ontologies. It is the main point for creating, loading and accessing ontologies.

`__slots__` = '\_world'

`create_ontology` (*iri*: *owlapy.iri.IRI*) → *owlapy.owl\_ontology.Ontology*

Creates a new (empty) ontology that has the specified ontology IRI (and no version IRI).

**Parameters**

**iri** – The IRI of the ontology to be created.

**Returns**

The newly created ontology, or if an ontology with the specified IRI already exists then this existing ontology will be returned.

`load_ontology` (*iri*: *owlapy.iri.IRI*) → *owlapy.owl\_ontology.Ontology*

Loads an ontology that is assumed to have the specified ontology IRI as its IRI or version IRI. The ontology IRI will be mapped to an ontology document IRI.

**Parameters**

**iri** – The IRI that identifies the ontology. It is expected that the ontology will also have this IRI (although the OWL API should tolerate situations where this is not the case).

**Returns**

The *OWL ontology* representation of the ontology that was loaded.

`apply_change` (*change*: *OWL ontologyChange*)

A convenience method that applies just one change to an ontology. When this method is used through an *OWL ontologyManager* implementation, the instance used should be the one that the ontology returns through the `get_owl_ontology_manager()` call.



#### Parameters

**change** – The change to be applied.

#### Raises

**ChangeApplied.UNSUCCESSFULLY** – if the change was not applied successfully.

**add\_axiom** (*ontology*: *owlapy.owl\_ontology.OWLontology*, *axiom*: *owlapy.owl\_axiom.OWLAxiom*)

A convenience method that adds a single axiom to an ontology.

#### Parameters

- **ontology** – The ontology to add the axiom to.
- **axiom** – The axiom to be added.

**remove\_axiom** (*ontology*: *owlapy.owl\_ontology.OWLontology*, *axiom*: *owlapy.owl\_axiom.OWLAxiom*)

A convenience method that removes a single axiom from an ontology.

#### Parameters

- **ontology** – The ontology to remove the axiom from.
- **axiom** – The axiom to be removed.

**save\_ontology** (*ontology*: *owlapy.owl\_ontology.OWLontology*, *document\_iri*: *owlapy.iri.IRI*)

Saves the specified ontology, using the specified document IRI to determine where/how the ontology should be saved.

#### Parameters

- **ontology** – The ontology to be saved.
- **document\_iri** – The document IRI where the ontology should be saved to.

**save\_world** ()

Saves the actual state of the quadstore in the SQLite3 file.

## owlapy.owl\_property

OWL Properties

## Classes

<i>OWLObject</i>	Base interface for OWL objects
<i>OWLEntity</i>	Represents Entities in the OWL 2 Specification.
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>OWLPropertyExpression</i>	Represents a property or possibly the inverse of a property.
<i>OWLObjectPropertyExpression</i>	A high level interface to describe different types of object properties.
<i>OWLDataPropertyExpression</i>	A high level interface to describe different types of data properties.
<i>OWLProperty</i>	A base class for properties that aren't expression i.e. named properties. By definition, properties
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
<i>OWLObjectInverseOf</i>	Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression
<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.

## Module Contents

**class** owlapy.owl\_property.OWLObject

Base interface for OWL objects

**\_\_slots\_\_** = ()

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

Return self==value.

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

Return hash(self).

**abstract** **\_\_repr\_\_** ()

Return repr(self).

**is\_anonymous** () → bool

**class** owlapy.owl\_property.OWLEntity

Bases: OWLNamedObject

Represents Entities in the OWL 2 Specification.

**\_\_slots\_\_** = ()

**to\_string\_id** () → str

**is\_anonymous** () → bool

**class** owlapy.owl\_property.IRI (*namespace: str | owlapy.namespaces.Namespaces, remainder: str*)

Bases: owlapy.owl\_annotation.OWLAnnotationSubject, owlapy.owl\_annotation.OWLAnnotationValue

An IRI, consisting of a namespace and a remainder.

```

__slots__ = ('_namespace', '_remainder', '__weakref__')

type_index: Final = 0

static create (namespace: owlpy.namespaces.Namespaces, remainder: str) → IRI
static create (namespace: str, remainder: str) → IRI
static create (string: str) → IRI

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

is_nothing ()
    Determines if this IRI is equal to the IRI that owl:Nothing is named with.

    Returns
        True if this IRI is equal to <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.

is_thing ()
    Determines if this IRI is equal to the IRI that owl:Thing is named with.

    Returns
        True if this IRI is equal to <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

is_reserved_vocabulary () → bool
    Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

    Returns
        True if the IRI is in the reserved vocabulary, otherwise False.

as_iri () → IRI

    Returns
        if the value is an IRI, return it. Return None otherwise.

as_str () → str
    CD: Should be deprecated. :returns: The string that specifies the IRI.

property str: str
    Returns: The string that specifies the IRI.

property remainder: str
    Returns: The string corresponding to the remainder of the IRI.

get_short_form () → str
    Gets the short form.

    Returns
        A string that represents the short form.

```

**get\_namespace()** → str

**Returns**

The namespace as string.

**get\_remainder()** → str

**Returns**

The remainder (coincident with NCName usually) for this IRI.

**class** owlapy.owl\_property.OWLPropertyExpression

Bases: *owlapy.owl\_object.OWLObject*

Represents a property or possibly the inverse of a property.

**\_\_slots\_\_** = ()

**is\_data\_property\_expression()** → bool

**Returns**

True if this is a data property.

**is\_object\_property\_expression()** → bool

**Returns**

True if this is an object property.

**is\_owl\_top\_object\_property()** → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

**is\_owl\_top\_data\_property()** → bool

Determines if this is the owl:topDataProperty.

**Returns**

topDataProperty.

**Return type**

True if this property is the owl

**class** owlapy.owl\_property.OWLObjectPropertyExpression

Bases: *OWLPropertyExpression*

A high level interface to describe different types of object properties.

**\_\_slots\_\_** = ()

**abstract get\_inverse\_property()** → *OWLObjectPropertyExpression*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**abstract get\_named\_property()** → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**is\_object\_property\_expression**() → bool

**Returns**

True if this is an object property.

**class** owlapy.owl\_property.**OWLDataPropertyExpression**

Bases: *OWLPropertyExpression*

A high level interface to describe different types of data properties.

**\_\_slots\_\_** = ()

**is\_data\_property\_expression**()

**Returns**

True if this is a data property.

**class** owlapy.owl\_property.**OWLProperty**(iri: *owlapy.iri.IRI* | str)

Bases: *OWLPropertyExpression*, *owlapy.owl\_object.OWLEntity*

A base class for properties that aren't expression i.e. named properties. By definition, properties are either data properties or object properties.

**\_\_slots\_\_** = '\_iri'

**property str: str**

Gets the string representation of this object

**Returns**

The IRI as string

**property iri: owlapy.iri.IRI**

Gets the IRI of this object.

**Returns**

The IRI of this object.

**class** owlapy.owl\_property.**OWLObjectProperty**(iri: *owlapy.iri.IRI* | str)

Bases: *OWLObjectPropertyExpression*, *OWLProperty*

Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.

([https://www.w3.org/TR/owl2-syntax/#Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Object_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index: Final = 1002**

**get\_named\_property**() → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**get\_inverse\_property**() → *OWLObjectInverseOf*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**is\_owl\_top\_object\_property** () → bool  
Determines if this is the owl:topObjectProperty.

**Returns**  
topObjectProperty.

**Return type**  
True if this property is the owl

**class** owlapy.owl\_property.**OWLObjectInverseOf** (property: *OWLObjectProperty*)

Bases: *OWLObjectPropertyExpression*

Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression ObjectInverseOf( P ) connects an individual I1 with I2 if and only if the object property P connects I2 with I1. This can be used to refer to the inverse of a property, without actually naming the property. For example, consider the property hasPart, the inverse property of hasPart (isPartOf) can be referred to using this interface inverseOf(hasPart), which can be used in restrictions e.g. inverseOf(hasPart) some Car refers to the set of things that are part of at least one car.

([https://www.w3.org/TR/owl2-syntax/#Inverse\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Inverse_Object_Properties))

**\_\_slots\_\_** = '\_inverse\_property'

**type\_index**: Final = 1003

**get\_inverse** () → *OWLObjectProperty*

Gets the property expression that this is the inverse of.

**Returns**  
The object property expression such that this object property expression is an inverse of it.

**get\_inverse\_property** () → *OWLObjectProperty*

Obtains the property that corresponds to the inverse of this property.

**Returns**  
The inverse of this property. Note that this property will not necessarily be in the simplest form.

**get\_named\_property** () → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**  
P if this expression is either inv(P) or P.

**\_\_repr\_\_** ()  
Return repr(self).

**\_\_eq\_\_** (other)  
Return self==value.

**\_\_hash\_\_** ()  
Return hash(self).

**class** owlapy.owl\_property.**OWLDataProperty** (iri: *owlapy.iri.IRI* | str)

Bases: *OWLDataPropertyExpression*, *OWLProperty*

Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.

([https://www.w3.org/TR/owl2-syntax/#Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Data_Properties))

```

__slots__ = '_iri'

type_index: Final = 1004

is_owl_top_data_property() → bool
    Determines if this is the owl:topDataProperty.

    Returns
        topDataProperty.

    Return type
        True if this property is the owl

```

### owlapy.owl\_reasoner

OWL Reasoner

### Attributes

<i>logger</i>
---------------

### Classes

<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLObjectSomeValuesFrom</i>	An existential class expression <code>ObjectSomeValuesFrom( OPE CE )</code> consists of an object property expression <code>OPE</code> and
<i>OWLObjectUnionOf</i>	A union class expression <code>ObjectUnionOf( CE1 ... CEn )</code> contains all individuals that are instances
<i>OWLObjectIntersectionOf</i>	An intersection class expression <code>ObjectIntersectionOf( CE1 ... CEn )</code> contains all individuals that are instances
<i>OWLObjectComplementOf</i>	Represents an <code>ObjectComplementOf</code> class expression in the OWL 2 Specification.
<i>OWLObjectAllValuesFrom</i>	A universal class expression <code>ObjectAllValuesFrom( OPE CE )</code> consists of an object property expression <code>OPE</code> and a
<i>OWLObjectOneOf</i>	An enumeration of individuals <code>ObjectOneOf( a1 ... an )</code> contains exactly the individuals <code>ai</code> with $1 \leq i \leq n$ .
<i>OWLObjectHasValue</i>	A has-value class expression <code>ObjectHasValue( OPE a )</code> consists of an object property expression <code>OPE</code> and an
<i>OWLObjectMinCardinality</i>	A minimum cardinality expression <code>ObjectMinCardinality( n OPE CE )</code> consists of a nonnegative integer <code>n</code> , an object
<i>OWLObjectMaxCardinality</i>	A maximum cardinality expression <code>ObjectMaxCardinality( n OPE CE )</code> consists of a nonnegative integer <code>n</code> , an object

continues on next page

Table 8 – continued from previous page

<i>OWLObjectExactCardinality</i>	An exact cardinality expression <code>ObjectExactCardinality( n OPE CE )</code> consists of a nonnegative integer <i>n</i> , an object
<i>OWLObjectCardinalityRestriction</i>	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
<i>OWLDataSomeValuesFrom</i>	An existential class expression <code>DataSomeValuesFrom( DPE1 ... DPE<sub>n</sub> DR )</code> consists of <i>n</i> data property expressions
<i>OWLDataOneOf</i>	An enumeration of literals <code>DataOneOf( lt1 ... lt<sub>n</sub> )</code> contains exactly the explicitly specified literals <i>lt<sub>i</sub></i> with
<i>OWLDatatypeRestriction</i>	A datatype restriction <code>DatatypeRestriction( DT F1 lt1 ... Fn lt<sub>n</sub> )</code> consists of a unary datatype <i>DT</i> and <i>n</i> pairs
<i>OWLFacetRestriction</i>	A facet restriction is used to restrict a particular datatype.
<i>OWLDataHasValue</i>	A has-value class expression <code>DataHasValue( DPE lt )</code> consists of a data property expression <i>DPE</i> and a literal <i>lt</i> ,
<i>OWLDataAllValuesFrom</i>	A universal class expression <code>DataAllValuesFrom( DPE1 ... DPE<sub>n</sub> DR )</code> consists of <i>n</i> data property expressions <i>DPE<sub>i</sub></i> ,
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>OWLAxiom</i>	Represents Axioms in the OWL 2 Specification.
<i>OWLSubClassOfAxiom</i>	A subclass axiom <code>SubClassOf( CE1 CE2 )</code> states that the class expression <i>CE1</i> is a subclass of the class
<i>OWLDataRange</i>	Represents a <code>DataRange</code> in the OWL 2 Specification.
<i>OWLDataComplementOf</i>	A complement data range <code>DataComplementOf( DR )</code> contains all tuples of literals that are not contained in the
<i>OWLDataUnionOf</i>	A union data range <code>DataUnionOf( DR1 ... DR<sub>n</sub> )</code> contains all tuples of literals that are contained in the at least
<i>OWLDataIntersectionOf</i>	An intersection data range <code>DataIntersectionOf( DR1 ... DR<sub>n</sub> )</code> contains all tuples of literals that are contained
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
<i>OWLOntology</i>	Represents an OWL 2 Ontology in the OWL 2 specification.
<i>Ontology</i>	Represents an OWL 2 Ontology in the OWL 2 specification.
<i>ToOwlready2</i>	
<i>OntologyManager</i>	An <code>OWLOntologyManager</code> manages a set of ontologies. It is the main point for creating, loading and accessing
<i>OWLObjectPropertyExpression</i>	A high level interface to describe different types of object properties.
<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
<i>OWLObjectInverseOf</i>	Represents the inverse of a property expression ( <code>ObjectInverseOf</code> ). An inverse object property expression
<i>OWLPropertyExpression</i>	Represents a property or possibly the inverse of a property.
<i>OWLDataPropertyExpression</i>	A high level interface to describe different types of data properties.

continues on next page



Table 8 – continued from previous page

<i>OWLNamedIndividual</i>	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
<i>LRUCache</i>	Constants shares by all lru cache instances.
<i>OWLReasoner</i>	An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of
<i>BaseReasoner</i>	Enumeration class for base reasoner when calling <code>sync_reasoner</code> .
<i>OWLReasonerEx</i>	Extra convenience methods for OWL Reasoners
<i>OntologyReasoner</i>	Extra convenience methods for OWL Reasoners
<i>FastInstanceCheckerReasoner</i>	Tries to check instances fast (but maybe incomplete).
<i>SyncReasoner</i>	Extra convenience methods for OWL Reasoners

## Module Contents

**class** owlapy.owl\_reasoner.OWLClassExpression

Bases: *owlapy.owl\_data\_ranges.OWLPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

**\_\_slots\_\_** = ()

**abstract is\_owl\_thing** () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**abstract is\_owl\_nothing** () → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of** () → *OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**abstract get\_nnf** () → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.owl\_reasoner.OWLObjectSomeValuesFrom (  
*property*: owlapy.owl\_property.OWLObjectPropertyExpression,  
*filler*: owlapy.class\_expression.class\_expression.OWLClassExpression)

Bases: `OWLQuantifiedObjectRestriction`

An existential class expression `ObjectSomeValuesFrom( OPE CE )` consists of an object property expression `OPE` and a class expression `CE`, and it contains all those individuals that are connected by `OPE` to an individual that is an instance of `CE`.

```
__slots__ = ('_property', '_filler')
```

```
type_index: Final = 3005
```

```
__repr__()
```

Return `repr(self)`.

```
__eq__(other)
```

Return `self==value`.

```
__hash__()
```

Return `hash(self)`.

```
get_property() → owlapy.owl_property.OWLObjectPropertyExpression
```

#### Returns

Property being restricted.

```
class owlapy.owl_reasoner.OWLObjectUnionOf(  
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
```

Bases: `OWLNaryBooleanClassExpression`

A union class expression `ObjectUnionOf( CE1 ... CEn )` contains all individuals that are instances of at least one class expression `CEi` for  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Union\\_of\\_Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Union_of_Class_Expressions))

```
__slots__ = '_operands'
```

```
type_index: Final = 3002
```

```
class owlapy.owl_reasoner.OWLObjectIntersectionOf(  
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
```

Bases: `OWLNaryBooleanClassExpression`

An intersection class expression `ObjectIntersectionOf( CE1 ... CEn )` contains all individuals that are instances of all class expressions `CEi` for  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Intersection\\_of\\_Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Intersection_of_Class_Expressions))

```
__slots__ = '_operands'
```

```
type_index: Final = 3001
```

```
class owlapy.owl_reasoner.OWLObjectComplementOf(op: OWLClassExpression)
```

Bases: `OWLBooleanClassExpression`, *owlapy.meta\_classes.HasOperands[OWLClassExpression]*

Represents an `ObjectComplementOf` class expression in the OWL 2 Specification.

```
__slots__ = '_operand'
```

```
type_index: Final = 3003
```

**get\_operand** () → *OWLClassExpression*

**Returns**

The wrapped expression.

**operands** () → *Iterable[OWLClassExpression]*

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

```
class owlapy.owl_reasoner.OWLObjectAllValuesFrom (
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: *OWLQuantifiedObjectRestriction*

A universal class expression *ObjectAllValuesFrom*( OPE CE ) consists of an object property expression OPE and a class expression CE, and it contains all those individuals that are connected by OPE only to individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification))

**\_\_slots\_\_** = ('\_property', '\_filler')

**type\_index**: **Final** = 3006

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.owl_reasoner.OWLObjectOneOf (
    values: owlapy.owl_individual.OWLIndividual | Iterable[owlapy.owl_individual.OWLIndividual])
```

Bases: *owlapy.class\_expression.class\_expression.OWLAnonymousClassExpression*,  
*owlapy.meta\_classes.HasOperands[owlapy.owl\_individual.OWLIndividual]*

An enumeration of individuals *ObjectOneOf*( a1 ... an ) contains exactly the individuals ai with 1 ≤ i ≤ n. ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals))

**\_\_slots\_\_** = '\_values'

**type\_index**: **Final** = 3004

**individuals** () → Iterable[*owlapy.owl\_individual.OWLIndividual*]

Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this class expression.

**Returns**

The individuals that are the values of this {`@code ObjectOneOf`} class expression.

**operands** () → Iterable[*owlapy.owl\_individual.OWLIndividual*]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**as\_object\_union\_of** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

Simplifies this enumeration to a union of singleton nominals.

**Returns**

This enumeration in a more standard DL form.  $\text{simp}(\{a\}) = \{a\}$   $\text{simp}(\{a_0, \dots, \{a_n\}) = \text{unionOf}(\{a_0\}, \dots, \{a_n\})$

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

Return hash(self).

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

Return self==value.

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.owl\_reasoner.OWLObjectHasValue (  
    *property: owlapy.owl\_property.OWLObjectPropertyExpression,*  
    *individual: owlapy.owl\_individual.OWLIndividual)*

Bases: OWLHasValueRestriction[*owlapy.owl\_individual.OWLIndividual*], OWLObjectRestriction

A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ). ([https://www.w3.org/TR/owl2-syntax/#Individual\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Individual_Value_Restriction))

**\_\_slots\_\_** = ('\_property', '\_v')

**type\_index:** Final = 3007

**get\_property** () → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

**as\_some\_values\_from** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

**Returns**

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

**\_\_repr\_\_** ()

Return repr(self).

```
class owlapy.owl_reasoner.OWLObjectMinCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: *OWLObjectCardinalityRestriction*

A minimum cardinality expression *ObjectMinCardinality*( *n* OPE CE ) consists of a nonnegative integer *n*, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to at least *n* different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3008
```

```
class owlapy.owl_reasoner.OWLObjectMaxCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: *OWLObjectCardinalityRestriction*

A maximum cardinality expression *ObjectMaxCardinality*( *n* OPE CE ) consists of a nonnegative integer *n*, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE

to at most *n* different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3010
```

```
class owlapy.owl_reasoner.OWLObjectExactCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: *OWLObjectCardinalityRestriction*

**An exact cardinality expression *ObjectExactCardinality*( *n* OPE CE ) consists of a nonnegative integer *n*, an object**

**property expression OPE, and a class expression CE, and it contains all those individuals that are connected by to exactly *n* different individuals that are instances of CE.**

([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3009
```

```
as_intersection_of_min_max()
```

→ *owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

#### Returns

The semantically equivalent but structurally simpler form  $(= 1 \text{ R C}) = \geq 1 \text{ R C}$  and  $\leq 1 \text{ R C}$ .

```
class owlapy.owl_reasoner.OWLObjectCardinalityRestriction (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: *OWLCardinalityRestriction*[*owlapy.class\_expression.class\_expression.OWLClassExpression*], *OWLQuantifiedObjectRestriction*

Represents Object Property Cardinality Restrictions in the OWL 2 specification.

**\_\_slots\_\_** = ()

**get\_property**() → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

**\_\_repr\_\_**()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**class** owlapy.owl\_reasoner.OWLDataSomeValuesFrom(  
    *property: owlapy.owl\_property.OWLDataPropertyExpression,*  
    *filler: owlapy.owl\_data\_ranges.OWLDataRange)*

Bases: OWLQuantifiedDataRestriction

An existential class expression DataSomeValuesFrom( DPE1 ... DPE<sub>n</sub> DR ) consists of n data property expressions DPE<sub>i</sub>, 1 ≤ i ≤ n, and a data range DR whose arity must be n. Such a class expression contains all those individuals that are connected by DPE<sub>i</sub> to literals lti, 1 ≤ i ≤ n, such that the tuple ( lt1 , ..., ltn ) is in DR. A class expression of the form DataSomeValuesFrom( DPE DR ) can be seen as a syntactic shortcut for the class expression DataMinCardinality( 1 DPE DR ). ([https://www.w3.org/TR/owl2-syntax/#Existential\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Existential_Quantification_2))

**\_\_slots\_\_** = '\_property'

**type\_index: Final** = 3012

**\_\_repr\_\_**()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**get\_property**() → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

**class** owlapy.owl\_reasoner.OWLDataOneOf(  
    *values: owlapy.owl\_literal.OWLLiteral | Iterable[owlapy.owl\_literal.OWLLiteral]*)

Bases: owlapy.owl\_data\_ranges.OWLDataRange, owlapy.meta\_classes.  
HasOperands[owlapy.owl\_literal.OWLLiteral]

An enumeration of literals DataOneOf( lt1 ... ltn ) contains exactly the explicitly specified literals lti with 1 ≤ i ≤ n. The resulting data range has arity one. ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Literals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals))

**type\_index: Final** = 4003

**values**() → Iterable[*owlapy.owl\_literal.OWLLiteral*]

Gets the values that are in the oneOf.

**Returns**

The values of this {`@code DataOneOf`} class expression.

```

operands () → Iterable[owlapy.owl_literal.OWLLiteral]
    Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

    Returns
        The operands.

__hash__ ()
    Return hash(self).

__eq__ (other)
    Return self==value.

__repr__ ()
    Return repr(self).

class owlapy.owl_reasoner.OWLDatatypeRestriction (
    type_: owlapy.owl_datatype.OWLDatatype,
    facet_restrictions: OWLFacetRestriction | Iterable[OWLFacetRestriction])
    Bases: owlapy.owl_data_ranges.OWLDataRange

    A datatype restriction DatatypeRestriction( DT F1 lt1 ... Fn ltn ) consists of a unary datatype DT and n pairs (
    Fi , lti ). The resulting data range is unary and is obtained by restricting the value space of DT according to the
    semantics of all ( Fi , vi ) (multiple pairs are interpreted conjunctively), where vi are the data values of the literals
    lti. (https://www.w3.org/TR/owl2-syntax/#Datatype\_Restrictions)

    __slots__ = ('_type', '_facet_restrictions')

    type_index: Final = 4006

    get_datatype () → owlapy.owl_datatype.OWLDatatype

    get_facet_restrictions () → Sequence[OWLFacetRestriction]

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

    __repr__ ()
        Return repr(self).

class owlapy.owl_reasoner.OWLFacetRestriction (facet: owlapy.vocab.OWLFacet,
    literal: Literals)
    Bases: owlapy.owl_object.OWLObject

    A facet restriction is used to restrict a particular datatype.

    __slots__ = ('_facet', '_literal')

    type_index: Final = 4007

    get_facet () → owlapy.vocab.OWLFacet

    get_facet_value () → owlapy.owl_literal.OWLLiteral

    __eq__ (other)
        Return self==value.

```

```

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.owl_reasoner.OWLDataHasValue(
    property: owlapy.owl_property.OWLDataPropertyExpression,
    value: owlapy.owl_literal.OWLLiteral)
    Bases: OWLHasValueRestriction[owlapy.owl_literal.OWLLiteral], OWLDataRestriction

    A has-value class expression DataHasValue( DPE lt ) consists of a data property expression DPE and a literal lt, and it contains all those individuals that are connected by DPE to lt. Each such class expression can be seen as a syntactic shortcut for the class expression DataSomeValuesFrom( DPE DataOneOf( lt ) ). (https://www.w3.org/TR/owl2-syntax/#Literal\_Value\_Restriction)

    __slots__ = '_property'

    type_index: Final = 3014

    __repr__()
        Return repr(self).

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    as_some_values_from() → owlapy.class_expression.class_expression.OWLClassExpression
        A convenience method that obtains this restriction as an existential restriction with a nominal filler.

        Returns
            The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

    get_property() → owlapy.owl_property.OWLDataPropertyExpression

        Returns
            Property being restricted.

class owlapy.owl_reasoner.OWLDataAllValuesFrom(
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)
    Bases: OWLQuantifiedDataRestriction

    A universal class expression DataAllValuesFrom( DPE1 ... DPEn DR ) consists of n data property expressions DPEi,  $1 \leq i \leq n$ , and a data range DR whose arity must be n. Such a class expression contains all those individuals that

        are connected by DPEi only to literals lti,  $1 \leq i \leq n$ , such that each tuple ( lt1 , ..., ltn ) is in DR.

    A class
        expression of the form DataAllValuesFrom( DPE DR ) can be seen as a syntactic shortcut for the class expression DataMaxCardinality( 0 DPE DataComplementOf( DR ) ). (https://www.w3.org/TR/owl2-syntax/#Universal\_Quantification\_2)

    __slots__ = '_property'

    type_index: Final = 3013

```



**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

**class** owlapy.owl\_reasoner.OWLClass (iri: *owlapy.iri.IRI* | str)

Bases: *owlapy.class\_expression.class\_expression.OWLClassExpression*, *owlapy.owl\_object.OWLEntity*

An OWL 2 named Class. Classes can be understood as sets of individuals. (<https://www.w3.org/TR/owl2-syntax/#Classes>)

**\_\_slots\_\_** = ('\_iri', '\_is\_nothing', '\_is\_thing')

**type\_index**: Final = 1001

**property iri**: *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str**

**Gets the string representation of this object**

**Returns**

The IRI as string

**property reminder**: str

The reminder of the IRI

**is\_owl\_thing** () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**is\_owl\_nothing** () → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**get\_object\_complement\_of** ()

→ *owlapy.class\_expression.class\_expression.OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf()** → *OWLClass*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

```
class owlapy.owl_reasoner.IRI (namespace: str | owlapy.namespaces.Namespaces, remainder: str)
    Bases: owlapy.owl_annotation.OWLAnnotationSubject, owlapy.owl_annotation.
            OWLAnnotationValue
```

An IRI, consisting of a namespace and a remainder.

```
__slots__ = ('_namespace', '_remainder', '__weakref__')
```

```
type_index: Final = 0
```

```
static create (namespace: owlapy.namespaces.Namespaces, remainder: str) → IRI
```

```
static create (namespace: str, remainder: str) → IRI
```

```
static create (string: str) → IRI
```

```
__repr__()
```

Return repr(self).

```
__eq__ (other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
is_nothing()
```

Determines if this IRI is equal to the IRI that owl:Nothing is named with.

**Returns**

True if this IRI is equal to <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.

```
is_thing()
```

Determines if this IRI is equal to the IRI that owl:Thing is named with.

**Returns**

True if this IRI is equal to <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

```
is_reserved_vocabulary() → bool
```

Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

**Returns**

True if the IRI is in the reserved vocabulary, otherwise False.

```
as_iri() → IRI
```

**Returns**

if the value is an IRI, return it. Return None otherwise.

```
as_str() → str
```

CD: Should be deprecated. :returns: The string that specifies the IRI.

```
property str: str
```

Returns: The string that specifies the IRI.

**property reminder: str**

Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form() → str**

Gets the short form.

**Returns**

A string that represents the short form.

**get\_namespace() → str**

**Returns**

The namespace as string.

**get\_remainder() → str**

**Returns**

The remainder (coincident with NCName usually) for this IRI.

**class** owlapy.owl\_reasoner.OWL`Axiom`(*annotations: Iterable[OWLAnnotation] | None = None*)

Bases: *owlapy.owl\_object.OWLObject*

Represents Axioms in the OWL 2 Specification.

An OWL ontology contains a set of axioms. These axioms can be annotation axioms, declaration axioms, imports axioms or logical axioms.

**\_\_slots\_\_ = '\_annotations'**

**annotations() → List[OWLAnnotation] | None**

**is\_annotated() → bool**

**is\_logical\_axiom() → bool**

**is\_annotation\_axiom() → bool**

**class** owlapy.owl\_reasoner.OWLSubClassOfAxiom(  
    *sub\_class: owlapy.class\_expression.OWLClassExpression,*  
    *super\_class: owlapy.class\_expression.OWLClassExpression,*  
    *annotations: Iterable[OWLAnnotation] | None = None*)

Bases: OWLClassAxiom

A subclass axiom SubClassOf( CE1 CE2 ) states that the class expression CE1 is a subclass of the class expression CE2. Roughly speaking, this states that CE1 is more specific than CE2. Subclass axioms are a fundamental type of axioms in OWL 2 and can be used to construct a class hierarchy. Other kinds of class expression axiom can be seen as syntactic shortcuts for one or more subclass axioms.

([https://www.w3.org/TR/owl2-syntax/#Subclass\\_Axioms](https://www.w3.org/TR/owl2-syntax/#Subclass_Axioms))

**\_\_slots\_\_ = ('\_sub\_class', '\_super\_class')**

**get\_sub\_class() → owlapy.class\_expression.OWLClassExpression**

**get\_super\_class() → owlapy.class\_expression.OWLClassExpression**

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

Return self==value.

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

Return hash(self).

```

    __repr__()
        Return repr(self).

class owlapy.owl_reasoner.OWLDataRange
    Bases: OWLPropertyRange

    Represents a DataRange in the OWL 2 Specification.

class owlapy.owl_reasoner.OWLDataComplementOf (data_range: OWLDataRange)
    Bases: OWLDataRange

    A complement data range DataComplementOf( DR ) contains all tuples of literals that are not contained in the
    data range DR. The resulting data range has the arity equal to the arity of DR.

    (https://www.w3.org/TR/owl2-syntax/#Complement\_of\_Data\_Ranges)

    type_index: Final = 4002

    get_data_range() → OWLDataRange

        Returns
            The wrapped data range.

    __repr__()
        Return repr(self).

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

class owlapy.owl_reasoner.OWLDataUnionOf (operands: Iterable[OWLDataRange])
    Bases: OWLNaryDataRange

    A union data range DataUnionOf( DR1 ... DRn ) contains all tuples of literals that are contained in the at least one
    data range DRi for  $1 \leq i \leq n$ . All data ranges DRi must be of the same arity, and the resulting data range is of that
    arity as well.

    (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Data\_Ranges)

    __slots__ = '_operands'

    type_index: Final = 4005

class owlapy.owl_reasoner.OWLDataIntersectionOf (operands: Iterable[OWLDataRange])
    Bases: OWLNaryDataRange

    An intersection data range DataIntersectionOf( DR1 ... DRn ) contains all tuples of literals that are contained in
    each data range DRi for  $1 \leq i \leq n$ . All data ranges DRi must be of the same arity, and the resulting data range is
    of that arity as well.

    (https://www.w3.org/TR/owl2-syntax/#Intersection\_of\_Data\_Ranges)

    __slots__ = '_operands'

    type_index: Final = 4004

```

```
class owlapy.owl_reasoner.OWLDatatype (iri: owlapy.iri.IRI | owlapy.meta_classes.HasIRI)
    Bases: owlapy.owl_object.OWLEntity, owlapy.owl_data_ranges.OWLDataRange
```

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

```
__slots__ = '_iri'
```

```
type_index: Final = 4001
```

```
property iri: owlapy.iri.IRI
```

Gets the IRI of this object.

**Returns**

The IRI of this object.

```
property str: str
```

Gets the string representation of this object

**Returns**

The IRI as string

```
class owlapy.owl_reasoner.OWLOntology
```

Bases: *owlapy.owl\_object.OWLObject*

Represents an OWL 2 Ontology in the OWL 2 specification.

An OWLOntology consists of a possibly empty set of OWLAxioms and a possibly empty set of OWLAnnotations. An ontology can have an ontology IRI which can be used to identify the ontology. If it has an ontology IRI then it may also have an ontology version IRI. Since OWL 2, an ontology need not have an ontology IRI. (See the OWL 2 Structural Specification).

An ontology cannot be modified directly. Changes must be applied via its OWLOntologyManager.

```
__slots__ = ()
```

```
type_index: Final = 1
```

```
abstract classes_in_signature () → Iterable[owlapy.class_expression.OWLClass]
```

Gets the classes in the signature of this object.

**Returns**

Classes in the signature of this object.

```
abstract data_properties_in_signature ()
    → Iterable[owlapy.owl_property.OWLDataProperty]
```

Get the data properties that are in the signature of this object.

**Returns**

Data properties that are in the signature of this object.

```
abstract object_properties_in_signature ()
    → Iterable[owlapy.owl_property.OWLObjectProperty]
```

A convenience method that obtains the object properties that are in the signature of this object.

**Returns**

Object properties that are in the signature of this object.

**abstract individuals\_in\_signature ()**  
→ Iterable[*owlapy.owl\_individual.OWLNamedIndividual*]

A convenience method that obtains the individuals that are in the signature of this object.

**Returns**

Individuals that are in the signature of this object.

**abstract equivalent\_classes\_axioms (c: *owlapy.class\_expression.OWLClass*)**  
→ Iterable[*owlapy.owl\_axiom.OWLEquivalentClassesAxiom*]

Gets all of the equivalent axioms in this ontology that contain the specified class as an operand.

**Parameters**

**c** – The class for which the EquivalentClasses axioms should be retrieved.

**Returns**

EquivalentClasses axioms contained in this ontology.

**abstract general\_class\_axioms ()** → Iterable[*owlapy.owl\_axiom.OWLClassAxiom*]

**Get the general class axioms of this ontology. This includes SubClass axioms with a complex class expression**

as the sub class and EquivalentClass axioms and DisjointClass axioms with only complex class expressions.

**Returns**

General class axioms contained in this ontology.

**abstract data\_property\_domain\_axioms (property: *owlapy.owl\_property.OWLDataProperty*)**  
→ Iterable[*owlapy.owl\_axiom.OWLDataPropertyDomainAxiom*]

Gets the OWLDataPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**abstract data\_property\_range\_axioms (property: *owlapy.owl\_property.OWLDataProperty*)**  
→ Iterable[*owlapy.owl\_axiom.OWLDataPropertyRangeAxiom*]

Gets the OWLDataPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**abstract object\_property\_domain\_axioms (**  
*property: owlapy.owl\_property.OWLObjectProperty*  
**)** → Iterable[*owlapy.owl\_axiom.OWLObjectPropertyDomainAxiom*]

Gets the OWLObjectPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract object_property_range_axioms (
    property: owlapy.owl_property.OWLObjectProperty)
    → Iterable[owlapy.owl_axiom.OWLObjectPropertyRangeAxiom]
```

Gets the OWLObjectPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

```
abstract get_owl_ontology_manager () → _M
```

Gets the manager that manages this ontology.

```
abstract get_ontology_id () → OWLOntologyID
```

Gets the OWLOntologyID belonging to this object.

**Returns**

The OWLOntologyID.

```
is_anonymous () → bool
```

Check whether this ontology does contain an IRI or not.

```
class owlapy.owl_reasoner.Ontology (manager: OntologyManager, ontology_iri: owlapy.iri.IRI,
    load: bool)
```

Bases: *OWLOntology*

Represents an OWL 2 Ontology in the OWL 2 specification.

An OWLOntology consists of a possibly empty set of OWLAxioms and a possibly empty set of OWLAnnotations. An ontology can have an ontology IRI which can be used to identify the ontology. If it has an ontology IRI then it may also have an ontology version IRI. Since OWL 2, an ontology need not have an ontology IRI. (See the OWL 2 Structural Specification).

An ontology cannot be modified directly. Changes must be applied via its OWLOntologyManager.

```
__slots__ = ('_manager', '_iri', '_world', '_onto')
```

```
classes_in_signature () → Iterable[owlapy.class_expression.OWLClass]
```

Gets the classes in the signature of this object.

**Returns**

Classes in the signature of this object.

```
data_properties_in_signature () → Iterable[owlapy.owl_property.OWLDataProperty]
```

Get the data properties that are in the signature of this object.

**Returns**

Data properties that are in the signature of this object.

```
object_properties_in_signature () → Iterable[owlapy.owl_property.OWLObjectProperty]
```

A convenience method that obtains the object properties that are in the signature of this object.

**Returns**

Object properties that are in the signature of this object.

```
individuals_in_signature () → Iterable[owlapy.owl_individual.OWLNamedIndividual]
```

A convenience method that obtains the individuals that are in the signature of this object.

**Returns**

Individuals that are in the signature of this object.

**equivalent\_classes\_axioms** (*c*: *owlapy.class\_expression.OWLClass*)  
→ *Iterable[owlapy.owl\_axiom.OWLEquivalentClassesAxiom]*

Gets all of the equivalent axioms in this ontology that contain the specified class as an operand.

**Parameters**

**c** – The class for which the EquivalentClasses axioms should be retrieved.

**Returns**

EquivalentClasses axioms contained in this ontology.

**general\_class\_axioms** () → *Iterable[owlapy.owl\_axiom.OWLClassAxiom]*

**Get the general class axioms of this ontology. This includes SubClass axioms with a complex class expression**

as the sub class and EquivalentClass axioms and DisjointClass axioms with only complex class expressions.

**Returns**

General class axioms contained in this ontology.

**get\_owl\_ontology\_manager** () → *OntologyManager*

Gets the manager that manages this ontology.

**get\_ontology\_id** () → *OWLOntologyID*

Gets the OWLOntologyID belonging to this object.

**Returns**

The OWLOntologyID.

**data\_property\_domain\_axioms** (*pe*: *owlapy.owl\_property.OWLDataProperty*)  
→ *Iterable[owlapy.owl\_axiom.OWLDataPropertyDomainAxiom]*

Gets the OWLDataPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**data\_property\_range\_axioms** (*pe*: *owlapy.owl\_property.OWLDataProperty*)  
→ *Iterable[owlapy.owl\_axiom.OWLDataPropertyRangeAxiom]*

Gets the OWLDataPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**object\_property\_domain\_axioms** (*pe*: *owlapy.owl\_property.OWLObjectProperty*)  
→ *Iterable[owlapy.owl\_axiom.OWLObjectPropertyDomainAxiom]*

Gets the OWLObjectPropertyDomainAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.



**object\_property\_range\_axioms** (*pe: owlapy.owl\_property.OWLObjectProperty*)  
 → Iterable[*owlapy.owl\_axiom.OWLObjectPropertyRangeAxiom*]

Gets the OWLObjectPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**get\_original\_iri** ()

Get the IRI argument that was used to create this ontology.

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

Return self==value.

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

Return hash(self).

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.owl\_reasoner.ToOwlready2 (*world: owlready2.World*)

**\_\_slots\_\_** = '\_world'

**abstract map\_object** (*o: owlapy.owl\_object.OWLObject*)

Map owlapy object classes.

**abstract map\_concept** (*o: owlapy.class\_expression.OWLClassExpression*)

→ owlready2.ClassConstruct | owlready2.ThingClass

Map owlapy concept classes.

**abstract map\_data\_range** (*p: owlapy.owl\_data\_ranges.OWLObjectPropertyDomain*)

→ owlready2.ClassConstruct | type

Map owlapy data range classes.

**class** owlapy.owl\_reasoner.OntologyManager (*world\_store=None*)

Bases: OWLOntologyManager

An OWLOntologyManager manages a set of ontologies. It is the main point for creating, loading and accessing ontologies.

**\_\_slots\_\_** = '\_world'

**create\_ontology** (*iri: owlapy.iri.IRI*) → owlapy.owl\_ontology.Ontology

Creates a new (empty) ontology that has the specified ontology IRI (and no version IRI).

**Parameters**

**iri** – The IRI of the ontology to be created.

**Returns**

The newly created ontology, or if an ontology with the specified IRI already exists then this existing ontology will be returned.

**load\_ontology** (*iri: owlapy.iri.IRI*) → owlapy.owl\_ontology.Ontology

Loads an ontology that is assumed to have the specified ontology IRI as its IRI or version IRI. The ontology IRI will be mapped to an ontology document IRI.

#### Parameters

**iri** – The IRI that identifies the ontology. It is expected that the ontology will also have this IRI (although the OWL API should tolerate situations where this is not the case).

#### Returns

The OWLOntology representation of the ontology that was loaded.

**apply\_change** (*change*: *OWLOntologyChange*)

A convenience method that applies just one change to an ontology. When this method is used through an OWLOntologyManager implementation, the instance used should be the one that the ontology returns through the `get_owl_ontology_manager()` call.

#### Parameters

**change** – The change to be applied.

#### Raises

**ChangeApplied.UNSUCCESSFULLY** – if the change was not applied successfully.

**add\_axiom** (*ontology*: *owlapy.owl\_ontology.OWLOntology*, *axiom*: *owlapy.owl\_axiom.OWLAxiom*)

A convenience method that adds a single axiom to an ontology.

#### Parameters

- **ontology** – The ontology to add the axiom to.
- **axiom** – The axiom to be added.

**remove\_axiom** (*ontology*: *owlapy.owl\_ontology.OWLOntology*, *axiom*: *owlapy.owl\_axiom.OWLAxiom*)

A convenience method that removes a single axiom from an ontology.

#### Parameters

- **ontology** – The ontology to remove the axiom from.
- **axiom** – The axiom to be removed.

**save\_ontology** (*ontology*: *owlapy.owl\_ontology.OWLOntology*, *document\_iri*: *owlapy.iri.IRI*)

Saves the specified ontology, using the specified document IRI to determine where/how the ontology should be saved.

#### Parameters

- **ontology** – The ontology to be saved.
- **document\_iri** – The document IRI where the ontology should be saved to.

**save\_world** ()

Saves the actual state of the quadstore in the SQLite3 file.

**class** `owlapy.owl_reasoner.OWLObjectPropertyExpression`

Bases: *OWLPropertyExpression*

A high level interface to describe different types of object properties.

**\_\_slots\_\_** = ()

**abstract** `get_inverse_property` () → *OWLObjectPropertyExpression*

Obtains the property that corresponds to the inverse of this property.

#### Returns

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**abstract get\_named\_property** () → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**is\_object\_property\_expression** () → bool

**Returns**

True if this is an object property.

**class** owlapy.owl\_reasoner.OWLDataProperty (*iri: owlapy.iri.IRI | str*)

Bases: *OWLDataPropertyExpression*, OWLProperty

Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.

([https://www.w3.org/TR/owl2-syntax/#Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Data_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index: Final** = 1004

**is\_owl\_top\_data\_property** () → bool

Determines if this is the owl:topDataProperty.

**Returns**

topDataProperty.

**Return type**

True if this property is the owl

**class** owlapy.owl\_reasoner.OWLObjectProperty (*iri: owlapy.iri.IRI | str*)

Bases: *OWLObjectPropertyExpression*, OWLProperty

Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.

([https://www.w3.org/TR/owl2-syntax/#Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Object_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index: Final** = 1002

**get\_named\_property** () → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**get\_inverse\_property** () → *OWLObjectInverseOf*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**is\_owl\_top\_object\_property** () → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

```
class owlapy.owl_reasoner.OWLObjectInverseOf (property: OWLObjectProperty)
```

Bases: *OWLObjectPropertyExpression*

Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression ObjectInverseOf( P ) connects an individual I1 with I2 if and only if the object property P connects I2 with I1. This can be used to refer to the inverse of a property, without actually naming the property. For example, consider the property hasPart, the inverse property of hasPart (isPartOf) can be referred to using this interface inverseOf(hasPart), which can be used in restrictions e.g. inverseOf(hasPart) some Car refers to the set of things that are part of at least one car.

([https://www.w3.org/TR/owl2-syntax/#Inverse\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Inverse_Object_Properties))

```
__slots__ = '_inverse_property'
```

```
type_index: Final = 1003
```

```
get_inverse() → OWLObjectProperty
```

Gets the property expression that this is the inverse of.

**Returns**

The object property expression such that this object property expression is an inverse of it.

```
get_inverse_property() → OWLObjectProperty
```

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

```
get_named_property() → OWLObjectProperty
```

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
class owlapy.owl_reasoner.OWLPropertyExpression
```

Bases: *owlapy.owl\_object.OWLObject*

Represents a property or possibly the inverse of a property.

```
__slots__ = ()
```

```
is_data_property_expression() → bool
```

**Returns**

True if this is a data property.

```
is_object_property_expression() → bool
```

**Returns**

True if this is an object property.

**is\_owl\_top\_object\_property()** → bool  
Determines if this is the owl:topObjectProperty.

**Returns**  
topObjectProperty.

**Return type**  
True if this property is the owl

**is\_owl\_top\_data\_property()** → bool  
Determines if this is the owl:topDataProperty.

**Returns**  
topDataProperty.

**Return type**  
True if this property is the owl

**class** owlapy.owl\_reasoner.OWLDataPropertyExpression

Bases: *OWLPropertyExpression*

A high level interface to describe different types of data properties.

**\_\_slots\_\_** = ()

**is\_data\_property\_expression()**

**Returns**  
True if this is a data property.

**class** owlapy.owl\_reasoner.OWLNamedIndividual (*iri: owlapy.iri.IRI | str*)

Bases: *OWLIndividual*, *owlapy.owl\_object.OWLEntity*

Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities. IRIs from the reserved vocabulary must not be used to identify named individuals in an OWL 2 DL ontology.

([https://www.w3.org/TR/owl2-syntax/#Named\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Named_Individuals))

**\_\_slots\_\_** = '\_iri'

**type\_index: Final** = 1005

**property iri:** *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**  
The IRI of this object.

**property str**

Gets the string representation of this object

**Returns**  
The IRI as string

**class** owlapy.owl\_reasoner.OWLLiteral

Bases: *owlapy.owl\_annotation.OWLAnnotationValue*

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(<https://www.w3.org/TR/owl2-syntax/#Literals>)

**\_\_slots\_\_** = ()

**type\_index**: Final = 4008

**get\_literal**() → str

Gets the lexical value of this literal. Note that the language tag is not included.

**Returns**

The lexical value of this literal.

**is\_boolean**() → bool

Whether this literal is typed as boolean.

**parse\_boolean**() → bool

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("<http://www.w3.org/2001/XMLSchema#boolean>").

**Returns**

A bool value that is represented by this literal.

**is\_double**() → bool

Whether this literal is typed as double.

**parse\_double**() → float

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

**Returns**

A double value that is represented by this literal.

**is\_integer**() → bool

Whether this literal is typed as integer.

**parse\_integer**() → int

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

**Returns**

An integer value that is represented by this literal.

**is\_string**() → bool

Whether this literal is typed as string.

**parse\_string**() → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("<http://www.w3.org/2001/XMLSchema#string>").

**Returns**

A string value that is represented by this literal.

**is\_date**() → bool

Whether this literal is typed as date.

**parse\_date**() → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("<http://www.w3.org/2001/XMLSchema#date>").

**Returns**

A date value that is represented by this literal.

**is\_datetime** () → bool

Whether this literal is typed as dateTime.

**parse\_datetime** () → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("<http://www.w3.org/2001/XMLSchema#dateTime>").

**Returns**

A datetime value that is represented by this literal.

**is\_duration** () → bool

Whether this literal is typed as duration.

**parse\_duration** () → pandas.Timedelta

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("<http://www.w3.org/2001/XMLSchema#duration>").

**Returns**

A Timedelta value that is represented by this literal.

**is\_literal** () → bool

**Returns**

true if the annotation value is a literal

**as\_literal** () → *OWLLiteral*

**Returns**

if the value is a literal, returns it. Return None otherwise

**to\_python** () → Literals

**abstract get\_datatype** () → *owlapy.owl\_datatype.OWLDatatype*

Gets the OWLDatatype which types this literal.

**Returns**

The OWLDatatype that types this literal.

**class** owlapy.owl\_reasoner.LRUCache (maxsize: int | None = None)

Bases: Generic[\_K, \_V]

Constants shares by all lru cache instances.

Adapted from functools.lru\_cache.

**sentinel**

Unique object used to signal cache misses.

**PREV**

Name for the link field 0.

**NEXT**

Name for the link field 1.

**KEY**

Name for the link field 2.

**RESULT**

Name for the link field 3.

**sentinel**

**\_\_contains\_\_** (*item*: *\_K*) → bool

**\_\_getitem\_\_** (*item*: *\_K*) → *\_V*

**\_\_setitem\_\_** (*key*: *\_K*, *value*: *\_V*)

**cache\_info** ()

Report cache statistics.

**cache\_clear** ()

Clear the cache and cache statistics.

`owlapy.owl_reasoner.logger`

**class** `owlapy.owl_reasoner.OWLReasoner` (*ontology*: *owlapy.owl\_ontology.OWLOntology*)

An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of a particular ontology - the “root” ontology.

**\_\_slots\_\_** = ()

**abstract data\_property\_domains** (*pe*: *owlapy.owl\_property.OWLDataProperty*,  
*direct*: bool = False) → Iterable[*owlapy.class\_expression.OWLClassExpression*]

**Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.**

#### Parameters

- **pe** – The property expression whose domains are to be retrieved.
- **direct** – Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

#### Returns

Let *N* = `equivalent_classes(DataSomeValuesFrom(pe rdfs:Literal))`. If *direct* is True: then if *N* is not empty then the return value is *N*, else the return value is the result of `super_classes(DataSomeValuesFrom(pe rdfs:Literal), true)`. If *direct* is False: then the result of `super_classes(DataSomeValuesFrom(pe rdfs:Literal), false)` together with *N* if *N* is non-empty. (Note, `rdfs:Literal` is the top datatype).

**abstract object\_property\_domains** (*pe*: *owlapy.owl\_property.OWLObjectProperty*,  
*direct*: bool = False) → Iterable[*owlapy.class\_expression.OWLClassExpression*]

**Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.**

#### Parameters

- **pe** – The property expression whose domains are to be retrieved.
- **direct** – Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

#### Returns

Let *N* = `equivalent_classes(ObjectSomeValuesFrom(pe owl:Thing))`. If *direct* is True: then if *N* is not empty then the return value is *N*, else the return value is the result of `super_classes(ObjectSomeValuesFrom(pe owl:Thing), true)`. If *direct* is False: then the result



of `super_classes(ObjectSomeValuesFrom(pe owl:Thing), false)` together with `N` if `N` is non-empty.

**abstract object\_property\_ranges** (*pe*: *owlapy.owl\_property.OWLObjectProperty*,  
*direct*: *bool = False*) → *Iterable[owlapy.class\_expression.OWLClassExpression]*

**Gets the class expressions that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.**

#### Parameters

- **pe** – The property expression whose ranges are to be retrieved.
- **direct** – Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

#### Returns

Let `N = equivalent_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing))`. If `direct` is True: then if `N` is not empty then the return value is `N`, else the return value is the result of `super_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing), true)`. If `direct` is False: then the result of `super_classes(ObjectSomeValuesFrom(ObjectInverseOf(pe) owl:Thing), false)` together with `N` if `N` is non-empty.

**abstract equivalent\_classes** (*ce*: *owlapy.class\_expression.OWLClassExpression*,  
*only\_named*: *bool = True*) → *Iterable[owlapy.class\_expression.OWLClassExpression]*

**Gets the class expressions that are equivalent to the specified class expression with respect to the set of reasoner axioms.**

#### Parameters

- **ce** – The class expression whose equivalent classes are to be retrieved.
- **only\_named** – Whether to only retrieve named equivalent classes or also complex class expressions.

#### Returns

All class expressions `C` where the root ontology imports closure entails `EquivalentClasses(ce C)`. If `ce` is not a class name (i.e. it is an anonymous class expression) and there are no such classes `C` then there will be no result. If `ce` is unsatisfiable with respect to the set of reasoner axioms then `owl:Nothing`, i.e. the bottom node, will be returned.

**abstract disjoint\_classes** (*ce*: *owlapy.class\_expression.OWLClassExpression*,  
*only\_named*: *bool = True*) → *Iterable[owlapy.class\_expression.OWLClassExpression]*

**Gets the class expressions that are disjoint with specified class expression with respect to the set of reasoner axioms.**

#### Parameters

- **ce** – The class expression whose disjoint classes are to be retrieved.
- **only\_named** – Whether to only retrieve named disjoint classes or also complex class expressions.

#### Returns

All class expressions `D` where the set of reasoner axioms entails `EquivalentClasses(D ObjectComplementOf(ce))` or `StrictSubClassOf(D ObjectComplementOf(ce))`.

**abstract different\_individuals** (*ind*: *owlapy.owl\_individual.OWLNamedIndividual*)  
→ *Iterable[owlapy.owl\_individual.OWLNamedIndividual]*

**Gets the individuals that are different from the specified individual with respect to the set of reasoner axioms.**

#### Parameters

**ind** – The individual whose different individuals are to be retrieved.

#### Returns

All individuals x where the set of reasoner axioms entails `DifferentIndividuals(ind x)`.

**abstract same\_individuals** (*ind: owlapy.owl\_individual.OWLNamedIndividual*)  
→ *Iterable[owlapy.owl\_individual.OWLNamedIndividual]*

Gets the individuals that are the same as the specified individual with respect to the set of reasoner axioms.

#### Parameters

**ind** – The individual whose same individuals are to be retrieved.

#### Returns

All individuals x where the root ontology imports closure entails `SameIndividual(ind x)`.

**abstract equivalent\_object\_properties** (*op: owlapy.owl\_property.OWLObjectPropertyExpression*)  
→ *Iterable[owlapy.owl\_property.OWLObjectPropertyExpression]*

Gets the simplified object properties that are equivalent to the specified object property with respect to the set of reasoner axioms.

#### Parameters

**op** – The object property whose equivalent object properties are to be retrieved.

#### Returns

All simplified object properties e where the root ontology imports closure entails `EquivalentObjectProperties(op e)`. If op is unsatisfiable with respect to the set of reasoner axioms then `owl:bottomDataProperty` will be returned.

**abstract equivalent\_data\_properties** (*dp: owlapy.owl\_property.OWLDataProperty*)  
→ *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets the data properties that are equivalent to the specified data property with respect to the set of reasoner axioms.

#### Parameters

**dp** – The data property whose equivalent data properties are to be retrieved.

#### Returns

All data properties e where the root ontology imports closure entails `EquivalentDataProperties(dp e)`. If dp is unsatisfiable with respect to the set of reasoner axioms then `owl:bottomDataProperty` will be returned.

**abstract data\_property\_values** (*ind: owlapy.owl\_individual.OWLNamedIndividual*,  
*pe: owlapy.owl\_property.OWLDataProperty*, *direct: bool = True*)  
→ *Iterable[owlapy.owl\_literal.OWLLiteral]*

Gets the data property values for the specified individual and data property expression.

#### Parameters

- **ind** – The individual that is the subject of the data property values.
- **pe** – The data property expression whose values are to be retrieved for the specified individual.
- **direct** – Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

#### Returns

A set of OWLLiterals containing literals such that for each literal l in the set, the set of reasoner axioms entails `DataPropertyAssertion(pe ind l)`.

**abstract object\_property\_values** (*ind: owlpy.owl\_individual.OWLNamedIndividual, pe: owlpy.owl\_property.OWLObjectPropertyExpression, direct: bool = True*)  
→ Iterable[*owlpy.owl\_individual.OWLNamedIndividual*]

Gets the object property values for the specified individual and object property expression.

#### Parameters

- **ind** – The individual that is the subject of the object property values.
- **pe** – The object property expression whose values are to be retrieved for the specified individual.
- **direct** – Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

#### Returns

The named individuals such that for each individual *j*, the set of reasoner axioms entails ObjectPropertyAssertion(*pe ind j*).

**abstract flush** () → None

Flushes any changes stored in the buffer, which causes the reasoner to take into consideration the changes the current root ontology specified by the changes.

**abstract instances** (*ce: owlpy.class\_expression.OWLClassExpression, direct: bool = False*)  
→ Iterable[*owlpy.owl\_individual.OWLNamedIndividual*]

Gets the individuals which are instances of the specified class expression.

#### Parameters

- **ce** – The class expression whose instances are to be retrieved.
- **direct** – Specifies if the direct instances should be retrieved (True), or if all instances should be retrieved (False).

#### Returns

If *direct* is True, each named individual *j* where the set of reasoner axioms entails DirectClassAssertion(*ce, j*). If *direct* is False, each named individual *j* where the set of reasoner axioms entails ClassAssertion(*ce, j*). If *ce* is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

**abstract sub\_classes** (*ce: owlpy.class\_expression.OWLClassExpression, direct: bool = False, only\_named: bool = True*) → Iterable[*owlpy.class\_expression.OWLClassExpression*]

Gets the set of named classes that are the strict (potentially direct) subclasses of the specified class expression with respect to the reasoner axioms.

#### Parameters

- **ce** – The class expression whose strict (direct) subclasses are to be retrieved.
- **direct** – Specifies if the direct subclasses should be retrieved (True) or if the all subclasses (descendant) classes should be retrieved (False).
- **only\_named** – Whether to only retrieve named sub-classes or also complex class expressions.

#### Returns

If *direct* is True, each class *C* where reasoner axioms entails DirectSubClassOf(*C, ce*). If *direct* is False, each class *C* where reasoner axioms entails StrictSubClassOf(*C, ce*). If *ce* is equivalent to owl:Nothing then nothing will be returned.

```
abstract disjoint_object_properties (  
    op: owlapy.owl_property.OWLObjectPropertyExpression  
    → Iterable[owlapy.owl_property.OWLObjectPropertyExpression]
```

Gets the simplified object properties that are disjoint with the specified object property with respect to the set of reasoner axioms.

#### Parameters

**op** – The object property whose disjoint object properties are to be retrieved.

#### Returns

All simplified object properties *e* where the root ontology imports closure entails `EquivalentObjectProperties(e ObjectPropertyComplementOf(op))` or `StrictSubObjectPropertyOf(e ObjectPropertyComplementOf(op))`.

```
abstract disjoint_data_properties (dp: owlapy.owl_property.OWLDataProperty)  
    → Iterable[owlapy.owl_property.OWLDataProperty]
```

Gets the data properties that are disjoint with the specified data property with respect to the set of reasoner axioms.

#### Parameters

**dp** – The data property whose disjoint data properties are to be retrieved.

#### Returns

All data properties *e* where the root ontology imports closure entails `EquivalentDataProperties(e DataPropertyComplementOf(dp))` or `StrictSubDataPropertyOf(e DataPropertyComplementOf(dp))`.

```
abstract sub_data_properties (dp: owlapy.owl_property.OWLDataProperty,  
    direct: bool = False) → Iterable[owlapy.owl_property.OWLDataProperty]
```

Gets the set of named data properties that are the strict (potentially direct) subproperties of the specified data property expression with respect to the imports closure of the root ontology.

#### Parameters

- **dp** – The data property whose strict (direct) subproperties are to be retrieved.
- **direct** – Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

#### Returns

If **direct** is True, each property *P* where the set of reasoner axioms entails `DirectSubDataPropertyOf(P, pe)`. If **direct** is False, each property *P* where the set of reasoner axioms entails `StrictSubDataPropertyOf(P, pe)`. If *pe* is equivalent to `owl:bottomDataProperty` then nothing will be returned.

```
abstract super_data_properties (dp: owlapy.owl_property.OWLDataProperty,  
    direct: bool = False) → Iterable[owlapy.owl_property.OWLDataProperty]
```

Gets the stream of data properties that are the strict (potentially direct) super properties of the specified data property with respect to the imports closure of the root ontology.

#### Parameters

- **dp** (*OWLDataProperty*) – The data property whose super properties are to be retrieved.
- **direct** (*bool*) – Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

#### Returns

Iterable of super properties.

**abstract sub\_object\_properties** (*op*: *owlapy.owl\_property.OWLObjectPropertyExpression*,  
*direct*: *bool* = *False*) → *Iterable*[*owlapy.owl\_property.OWLObjectPropertyExpression*]

Gets the stream of simplified object property expressions that are the strict (potentially direct) subproperties of the specified object property expression with respect to the imports closure of the root ontology.

#### Parameters

- **op** – The object property expression whose strict (direct) subproperties are to be retrieved.
- **direct** – Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

#### Returns

If *direct* is True, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails *DirectSubObjectPropertyOf*(P, pe). If *direct* is False, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails *StrictSubObjectPropertyOf*(P, pe). If pe is equivalent to *owl:bottomObjectProperty* then nothing will be returned.

**abstract super\_object\_properties** (*op*: *owlapy.owl\_property.OWLObjectPropertyExpression*,  
*direct*: *bool* = *False*) → *Iterable*[*owlapy.owl\_property.OWLObjectPropertyExpression*]

Gets the stream of object properties that are the strict (potentially direct) super properties of the specified object property with respect to the imports closure of the root ontology.

#### Parameters

- **op** (*OWLObjectPropertyExpression*) – The object property expression whose super properties are to be retrieved.
- **direct** (*bool*) – Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

#### Returns

Iterable of super properties.

**abstract types** (*ind*: *owlapy.owl\_individual.OWLNamedIndividual*, *direct*: *bool* = *False*)  
→ *Iterable*[*owlapy.class\_expression.OWLClass*]

Gets the named classes which are (potentially direct) types of the specified named individual.

#### Parameters

- **ind** – The individual whose types are to be retrieved.
- **direct** – Specifies if the direct types should be retrieved (True), or if all types should be retrieved (False).

#### Returns

If *direct* is True, each named class C where the set of reasoner axioms entails *DirectClassAssertion*(C, ind). If *direct* is False, each named class C where the set of reasoner axioms entails *ClassAssertion*(C, ind).

**abstract get\_root\_ontology** () → *owlapy.owl\_ontology.OWL ontology*

Gets the “root” ontology that is loaded into this reasoner. The reasoner takes into account the axioms in this ontology and its import’s closure.

**abstract is\_isolated** ()

Return True if this reasoner is using an isolated ontology.

**abstract super\_classes** (*ce*: *owlapy.class\_expression.OWLClassExpression*, *direct*: *bool* = *False*,  
*only\_named*: *bool* = *True*) → *Iterable*[*owlapy.class\_expression.OWLClassExpression*]

Gets the stream of named classes that are the strict (potentially direct) super classes of the specified class expression with respect to the imports closure of the root ontology.

#### Parameters

- **ce** – The class expression whose strict (direct) super classes are to be retrieved.
- **direct** – Specifies if the direct super classes should be retrieved (True) or if the all super classes (ancestors) classes should be retrieved (False).
- **only\_named** – Whether to only retrieve named super classes or also complex class expressions.

#### Returns

If **direct** is True, each class C where the set of reasoner axioms entails `DirectSubClassOf(ce, C)`. If **direct** is False, each class C where set of reasoner axioms entails `StrictSubClassOf(ce, C)`. If **ce** is equivalent to `owl:Thing` then nothing will be returned.

**class** owlapy.owl\_reasoner.BaseReasoner

Bases: `enum.Enum`

Enumeration class for base reasoner when calling `sync_reasoner`.

**PELLET**

Pellet base reasoner.

**HERMIT**

HermiT base reasoner.

**PELLET**

**HERMIT**

**class** owlapy.owl\_reasoner.OWLReasonerEx (ontology: owlapy.owl\_ontology.OWLontology)

Bases: *OWLReasoner*

Extra convenience methods for OWL Reasoners

**data\_property\_ranges** (pe: owlapy.owl\_property.OWLDataProperty, direct: bool = False)  
→ Iterable[owlapy.owl\_data\_ranges.OWLDataRange]

Gets the data ranges that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.

#### Parameters

- **pe** – The property expression whose ranges are to be retrieved.
- **direct** – Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

Returns:

**all\_data\_property\_values** (pe: owlapy.owl\_property.OWLDataProperty, direct: bool = True)  
→ Iterable[owlapy.owl\_literal.OWLLiteral]

Gets all values for the given data property expression that appear in the knowledge base.

#### Parameters

- **pe** – The data property expression whose values are to be retrieved
- **direct** – Specifies if only the direct values of the data property **pe** should be retrieved (True), or if the values of sub properties of **pe** should be taken into account (False).

### Returns

A set of OWLLiterals containing literals such that for each literal *l* in the set, the set of reasoner axioms entails `DataPropertyAssertion(pe ind l)` for any *ind*.

**ind\_data\_properties** (*ind*: *owlapy.owl\_individual.OWLNamedIndividual*, *direct*: *bool = True*)  
→ *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets all data properties for the given individual that appear in the knowledge base.

### Parameters

- **ind** – The named individual whose data properties are to be retrieved
- **direct** – Specifies if the direct data properties should be retrieved (True), or if all data properties should be retrieved (False), so that sub properties are taken into account.

### Returns

All data properties *pe* where the set of reasoner axioms entails `DataPropertyAssertion(pe ind l)` for atleast one *l*.

**ind\_object\_properties** (*ind*: *owlapy.owl\_individual.OWLNamedIndividual*, *direct*: *bool = True*)  
→ *Iterable[owlapy.owl\_property.OWLObjectProperty]*

Gets all object properties for the given individual that appear in the knowledge base.

### Parameters

- **ind** – The named individual whose object properties are to be retrieved
- **direct** – Specifies if the direct object properties should be retrieved (True), or if all object properties should be retrieved (False), so that sub properties are taken into account.

### Returns

All data properties *pe* where the set of reasoner axioms entails `ObjectPropertyAssertion(pe ind ind2)` for atleast one *ind2*.

**class** *owlapy.owl\_reasoner.OntologyReasoner* (*ontology*: *owlapy.owl\_ontology.Ontology*,  
*isolate*: *bool = False*)

Bases: *OWLReasonerEx*

Extra convenience methods for OWL Reasoners

**\_\_slots\_\_** = ('\_ontology', '\_world')

**update\_isolated\_ontology** (*axioms\_to\_add*: *List[owlapy.owl\_axiom.OWLAxiom]* = *None*,  
*axioms\_to\_remove*: *List[owlapy.owl\_axiom.OWLAxiom]* = *None*)

Add or remove axioms to the isolated ontology that the reasoner is using.

### Parameters

- **axioms\_to\_add** (*List [OWLAxiom]*) – Axioms to add to the isolated ontology.
- **axioms\_to\_remove** (*List [OWLAxiom]*) – Axioms to remove from the isolated ontology.

**data\_property\_domains** (*pe*: *owlapy.owl\_property.OWLDataProperty*, *direct*: *bool = False*)  
→ *Iterable[owlapy.class\_expression.OWLClassExpression]*

Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.

### Parameters

- **pe** – The property expression whose domains are to be retrieved.

- **direct** – Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

#### Returns

Let  $N = \text{equivalent\_classes}(\text{DataSomeValuesFrom}(\text{pe } \text{rdfs:Literal}))$ . If **direct** is True: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of  $\text{super\_classes}(\text{DataSomeValuesFrom}(\text{pe } \text{rdfs:Literal}), \text{true})$ . If **direct** is False: then the result of  $\text{super\_classes}(\text{DataSomeValuesFrom}(\text{pe } \text{rdfs:Literal}), \text{false})$  together with  $N$  if  $N$  is non-empty. (Note,  $\text{rdfs:Literal}$  is the top datatype).

**object\_property\_domains** (*pe*: *owlapy.owl\_property.OWLObjectProperty*, *direct*: *bool = False*)  
 → *Iterable[owlapy.class\_expression.OWLClassExpression]*

**Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.**

#### Parameters

- **pe** – The property expression whose domains are to be retrieved.
- **direct** – Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

#### Returns

Let  $N = \text{equivalent\_classes}(\text{ObjectSomeValuesFrom}(\text{pe } \text{owl:Thing}))$ . If **direct** is True: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{pe } \text{owl:Thing}), \text{true})$ . If **direct** is False: then the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{pe } \text{owl:Thing}), \text{false})$  together with  $N$  if  $N$  is non-empty.

**object\_property\_ranges** (*pe*: *owlapy.owl\_property.OWLObjectProperty*, *direct*: *bool = False*)  
 → *Iterable[owlapy.class\_expression.OWLClassExpression]*

**Gets the class expressions that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.**

#### Parameters

- **pe** – The property expression whose ranges are to be retrieved.
- **direct** – Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

#### Returns

Let  $N = \text{equivalent\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe } \text{owl:Thing})))$ . If **direct** is True: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe } \text{owl:Thing})), \text{true})$ . If **direct** is False: then the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe } \text{owl:Thing})), \text{false})$  together with  $N$  if  $N$  is non-empty.

**equivalent\_classes** (*ce*: *owlapy.class\_expression.OWLClassExpression*, *only\_named*: *bool = True*)  
 → *Iterable[owlapy.class\_expression.OWLClassExpression]*

**Gets the class expressions that are equivalent to the specified class expression with respect to the set of reasoner axioms.**

#### Parameters

- **ce** – The class expression whose equivalent classes are to be retrieved.



- **only\_named** – Whether to only retrieve named equivalent classes or also complex class expressions.

#### Returns

All class expressions C where the root ontology imports closure entails `EquivalentClasses(ce C)`. If ce is not a class name (i.e. it is an anonymous class expression) and there are no such classes C then there will be no result. If ce is unsatisfiable with respect to the set of reasoner axioms then `owl:Nothing`, i.e. the bottom node, will be returned.

**disjoint\_classes** (*ce: owlapy.class\_expression.OWLClassExpression, only\_named: bool = True*)  
 → `Iterable[owlapy.class_expression.OWLClassExpression]`

Gets the class expressions that are disjoint with specified class expression with respect to the set of reasoner axioms.

#### Parameters

- **ce** – The class expression whose disjoint classes are to be retrieved.
- **only\_named** – Whether to only retrieve named disjoint classes or also complex class expressions.

#### Returns

All class expressions D where the set of reasoner axioms entails `EquivalentClasses(D ObjectComplementOf(ce))` or `StrictSubClassOf(D ObjectComplementOf(ce))`.

**different\_individuals** (*ind: owlapy.owl\_individual.OWLNamedIndividual*)  
 → `Iterable[owlapy.owl_individual.OWLNamedIndividual]`

Gets the individuals that are different from the specified individual with respect to the set of reasoner axioms.

#### Parameters

**ind** – The individual whose different individuals are to be retrieved.

#### Returns

All individuals x where the set of reasoner axioms entails `DifferentIndividuals(ind x)`.

**same\_individuals** (*ind: owlapy.owl\_individual.OWLNamedIndividual*)  
 → `Iterable[owlapy.owl_individual.OWLNamedIndividual]`

Gets the individuals that are the same as the specified individual with respect to the set of reasoner axioms.

#### Parameters

**ind** – The individual whose same individuals are to be retrieved.

#### Returns

All individuals x where the root ontology imports closure entails `SameIndividual(ind x)`.

**data\_property\_values** (*ind: owlapy.owl\_individual.OWLNamedIndividual, pe: owlapy.owl\_property.OWLDataProperty, direct: bool = True*)  
 → `Iterable[owlapy.owl_literal.OWLLiteral]`

Gets the data property values for the specified individual and data property expression.

#### Parameters

- **ind** – The individual that is the subject of the data property values.
- **pe** – The data property expression whose values are to be retrieved for the specified individual.
- **direct** – Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

### Returns

A set of OWLLiterals containing literals such that for each literal *l* in the set, the set of reasoner axioms entails `DataPropertyAssertion(pe ind l)`.

**all\_data\_property\_values** (*pe*: *owlapy.owl\_property.OWLDataProperty*, *direct*: *bool = True*)  
→ `Iterable[owlapy.owl_literal.OWLLiteral]`

Gets all values for the given data property expression that appear in the knowledge base.

### Parameters

- **pe** – The data property expression whose values are to be retrieved
- **direct** – Specifies if only the direct values of the data property *pe* should be retrieved (True), or if the values of sub properties of *pe* should be taken into account (False).

### Returns

A set of OWLLiterals containing literals such that for each literal *l* in the set, the set of reasoner axioms entails `DataPropertyAssertion(pe ind l)` for any *ind*.

**object\_property\_values** (*ind*: *owlapy.owl\_individual.OWLNamedIndividual*,  
*pe*: *owlapy.owl\_property.OWLObjectPropertyExpression*, *direct*: *bool = False*)  
→ `Iterable[owlapy.owl_individual.OWLNamedIndividual]`

Gets the object property values for the specified individual and object property expression.

### Parameters

- **ind** – The individual that is the subject of the object property values.
- **pe** – The object property expression whose values are to be retrieved for the specified individual.
- **direct** – Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

### Returns

The named individuals such that for each individual *j*, the set of reasoner axioms entails `ObjectPropertyAssertion(pe ind j)`.

**flush** () → None

Flushes any changes stored in the buffer, which causes the reasoner to take into consideration the changes the current root ontology specified by the changes.

**instances** (*ce*: *owlapy.class\_expression.OWLClassExpression*, *direct*: *bool = False*)  
→ `Iterable[owlapy.owl_individual.OWLNamedIndividual]`

Gets the individuals which are instances of the specified class expression.

### Parameters

- **ce** – The class expression whose instances are to be retrieved.
- **direct** – Specifies if the direct instances should be retrieved (True), or if all instances should be retrieved (False).

### Returns

If *direct* is True, each named individual *j* where the set of reasoner axioms entails `DirectClassAssertion(ce, j)`. If *direct* is False, each named individual *j* where the set of reasoner axioms entails `ClassAssertion(ce, j)`. If *ce* is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

**sub\_classes** (*ce*: *owlapy.class\_expression.OWLClassExpression*, *direct*: *bool = False*,  
*only\_named*: *bool = True*) → `Iterable[owlapy.class_expression.OWLClassExpression]`

Gets the set of named classes that are the strict (potentially direct) subclasses of the specified class expression with respect to the reasoner axioms.

#### Parameters

- **ce** – The class expression whose strict (direct) subclasses are to be retrieved.
- **direct** – Specifies if the direct subclasses should be retrieved (True) or if the all subclasses (descendant) classes should be retrieved (False).
- **only\_named** – Whether to only retrieve named sub-classes or also complex class expressions.

#### Returns

If direct is True, each class C where reasoner axioms entails `DirectSubClassOf(C, ce)`. If direct is False, each class C where reasoner axioms entails `StrictSubClassOf(C, ce)`. If ce is equivalent to `owl:Nothing` then nothing will be returned.

**super\_classes** (*ce*: *owlapy.class\_expression.OWLClassExpression*, *direct*: *bool = False*, *only\_named*: *bool = True*) → *Iterable[owlapy.class\_expression.OWLClassExpression]*

Gets the stream of named classes that are the strict (potentially direct) super classes of the specified class expression with respect to the imports closure of the root ontology.

#### Parameters

- **ce** – The class expression whose strict (direct) super classes are to be retrieved.
- **direct** – Specifies if the direct super classes should be retrieved (True) or if the all super classes (ancestors) classes should be retrieved (False).
- **only\_named** – Whether to only retrieve named super classes or also complex class expressions.

#### Returns

If direct is True, each class C where the set of reasoner axioms entails `DirectSubClassOf(ce, C)`. If direct is False, each class C where set of reasoner axioms entails `StrictSubClassOf(ce, C)`. If ce is equivalent to `owl:Thing` then nothing will be returned.

**equivalent\_object\_properties** (*op*: *owlapy.owl\_property.OWLObjectPropertyExpression*) → *Iterable[owlapy.owl\_property.OWLObjectPropertyExpression]*

Gets the simplified object properties that are equivalent to the specified object property with respect to the set of reasoner axioms.

#### Parameters

**op** – The object property whose equivalent object properties are to be retrieved.

#### Returns

All simplified object properties e where the root ontology imports closure entails `EquivalentObjectProperties(op e)`. If op is unsatisfiable with respect to the set of reasoner axioms then `owl:bottomDataProperty` will be returned.

**equivalent\_data\_properties** (*dp*: *owlapy.owl\_property.OWLDataProperty*) → *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets the data properties that are equivalent to the specified data property with respect to the set of reasoner axioms.

#### Parameters

**dp** – The data property whose equivalent data properties are to be retrieved.

### Returns

All data properties *e* where the root ontology imports closure entails `EquivalentDataProperties(dp e)`. If *dp* is unsatisfiable with respect to the set of reasoner axioms then `owl:bottomDataProperty` will be returned.

**disjoint\_object\_properties** (*op*: *owlapy.owl\_property.OWLObjectPropertyExpression*)  
→ *Iterable[owlapy.owl\_property.OWLObjectPropertyExpression]*

Gets the simplified object properties that are disjoint with the specified object property with respect to the set of reasoner axioms.

### Parameters

**op** – The object property whose disjoint object properties are to be retrieved.

### Returns

All simplified object properties *e* where the root ontology imports closure entails `EquivalentObjectProperties(e ObjectPropertyComplementOf(op))` or `StrictSubObjectPropertyOf(e ObjectPropertyComplementOf(op))`.

**disjoint\_data\_properties** (*dp*: *owlapy.owl\_property.OWLDataProperty*)  
→ *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets the data properties that are disjoint with the specified data property with respect to the set of reasoner axioms.

### Parameters

**dp** – The data property whose disjoint data properties are to be retrieved.

### Returns

All data properties *e* where the root ontology imports closure entails `EquivalentDataProperties(e DataPropertyComplementOf(dp))` or `StrictSubDataPropertyOf(e DataPropertyComplementOf(dp))`.

**super\_data\_properties** (*dp*: *owlapy.owl\_property.OWLDataProperty*, *direct*: *bool = False*)  
→ *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets the stream of data properties that are the strict (potentially direct) super properties of the specified data property with respect to the imports closure of the root ontology.

### Parameters

- **dp** (*OWLDataProperty*) – The data property whose super properties are to be retrieved.
- **direct** (*bool*) – Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

### Returns

Iterable of super properties.

**sub\_data\_properties** (*dp*: *owlapy.owl\_property.OWLDataProperty*, *direct*: *bool = False*)  
→ *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets the set of named data properties that are the strict (potentially direct) subproperties of the specified data property expression with respect to the imports closure of the root ontology.

### Parameters

- **dp** – The data property whose strict (direct) subproperties are to be retrieved.
- **direct** – Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

### Returns

If *direct* is True, each property *P* where the set of reasoner axioms entails `DirectSubDataPropertyOf(P, pe)`. If *direct* is False, each property *P* where the set of reasoner axioms entails

StrictSubDataPropertyOf(P, pe). If pe is equivalent to owl:bottomDataProperty then nothing will be returned.

**super\_object\_properties** (*op*: owlapy.owl\_property.OWLObjectPropertyExpression,  
*direct*: bool = False) → Iterable[owlapy.owl\_property.OWLObjectPropertyExpression]

Gets the stream of object properties that are the strict (potentially direct) super properties of the specified object property with respect to the imports closure of the root ontology.

#### Parameters

- **op** (OWLObjectPropertyExpression) – The object property expression whose super properties are to be retrieved.
- **direct** (bool) – Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

#### Returns

Iterable of super properties.

**sub\_object\_properties** (*op*: owlapy.owl\_property.OWLObjectPropertyExpression,  
*direct*: bool = False) → Iterable[owlapy.owl\_property.OWLObjectPropertyExpression]

Gets the stream of simplified object property expressions that are the strict (potentially direct) subproperties of the specified object property expression with respect to the imports closure of the root ontology.

#### Parameters

- **op** – The object property expression whose strict (direct) subproperties are to be retrieved.
- **direct** – Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

#### Returns

If direct is True, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails DirectSubObjectPropertyOf(P, pe). If direct is False, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails StrictSubObjectPropertyOf(P, pe). If pe is equivalent to owl:bottomObjectProperty then nothing will be returned.

**types** (*ind*: owlapy.owl\_individual.OWLNamedIndividual, *direct*: bool = False)  
→ Iterable[owlapy.class\_expression.OWLClass]

Gets the named classes which are (potentially direct) types of the specified named individual.

#### Parameters

- **ind** – The individual whose types are to be retrieved.
- **direct** – Specifies if the direct types should be retrieved (True), or if all types should be retrieved (False).

#### Returns

If direct is True, each named class C where the set of reasoner axioms entails DirectClassAssertion(C, ind). If direct is False, each named class C where the set of reasoner axioms entails ClassAssertion(C, ind).

**get\_root\_ontology** () → owlapy.owl\_ontology.OWLOntology

Gets the “root” ontology that is loaded into this reasoner. The reasoner takes into account the axioms in this ontology and its import’s closure.

**is\_isolated** ()

Return True if this reasoner is using an isolated ontology.

```
class owlapy.owl_reasoner.FastInstanceCheckerReasoner (
    ontology: owlapy.owl_ontology.OWLOntology, base_reasoner: OWLReasoner, *,
    property_cache: bool = True, negation_default: bool = True, sub_properties: bool = False)
```

Bases: *OWLReasonerEx*

Tries to check instances fast (but maybe incomplete).

```
__slots__ = ('_ontology', '_base_reasoner', '_ind_set', '_cls_to_ind',
             '_has_prop', ...)
```

```
reset ()
```

The reset method shall reset any cached state.

```
is_isolated ()
```

Return True if this reasoner is using an isolated ontology.

```
is_using_triplestore ()
```

```
data_property_domains (pe: owlapy.owl_property.OWLDataProperty, direct: bool = False)
    → Iterable[owlapy.class_expression.OWLClassExpression]
```

**Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.**

#### Parameters

- **pe** – The property expression whose domains are to be retrieved.
- **direct** – Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

#### Returns

Let  $N = \text{equivalent\_classes}(\text{DataSomeValuesFrom}(\text{pe } \text{rdfs:Literal}))$ . If **direct** is True: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of  $\text{super\_classes}(\text{DataSomeValuesFrom}(\text{pe } \text{rdfs:Literal}), \text{true})$ . If **direct** is False: then the result of  $\text{super\_classes}(\text{DataSomeValuesFrom}(\text{pe } \text{rdfs:Literal}), \text{false})$  together with  $N$  if  $N$  is non-empty. (Note,  $\text{rdfs:Literal}$  is the top datatype).

```
data_property_ranges (pe: owlapy.owl_property.OWLDataProperty, direct: bool = False)
    → Iterable[owlapy.owl_data_ranges.OWLDataRange]
```

**Gets the data ranges that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.**

#### Parameters

- **pe** – The property expression whose ranges are to be retrieved.
- **direct** – Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

Returns:

```
object_property_domains (pe: owlapy.owl_property.OWLObjectProperty, direct: bool = False)
    → Iterable[owlapy.class_expression.OWLClassExpression]
```

**Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.**

#### Parameters

- **pe** – The property expression whose domains are to be retrieved.

- **direct** – Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

#### Returns

Let  $N = \text{equivalent\_classes}(\text{ObjectSomeValuesFrom}(\text{pe owl:Thing}))$ . If **direct** is True: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{pe owl:Thing}), \text{true})$ . If **direct** is False: then the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{pe owl:Thing}), \text{false})$  together with  $N$  if  $N$  is non-empty.

**object\_property\_ranges** (*pe: owlapy.owl\_property.OWLObjectProperty, direct: bool = False*)  
 → Iterable[*owlapy.class\_expression.OWLClassExpression*]

Gets the class expressions that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.

#### Parameters

- **pe** – The property expression whose ranges are to be retrieved.
- **direct** – Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

#### Returns

Let  $N = \text{equivalent\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe}) \text{ owl:Thing}))$ . If **direct** is True: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe}) \text{ owl:Thing}), \text{true})$ . If **direct** is False: then the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe}) \text{ owl:Thing}), \text{false})$  together with  $N$  if  $N$  is non-empty.

**equivalent\_classes** (*ce: owlapy.class\_expression.OWLClassExpression, only\_named: bool = True*)  
 → Iterable[*owlapy.class\_expression.OWLClassExpression*]

Gets the class expressions that are equivalent to the specified class expression with respect to the set of reasoner axioms.

#### Parameters

- **ce** – The class expression whose equivalent classes are to be retrieved.
- **only\_named** – Whether to only retrieve named equivalent classes or also complex class expressions.

#### Returns

All class expressions  $C$  where the root ontology imports closure entails  $\text{EquivalentClasses}(ce \ C)$ . If  $ce$  is not a class name (i.e. it is an anonymous class expression) and there are no such classes  $C$  then there will be no result. If  $ce$  is unsatisfiable with respect to the set of reasoner axioms then  $\text{owl:Nothing}$ , i.e. the bottom node, will be returned.

**disjoint\_classes** (*ce: owlapy.class\_expression.OWLClassExpression, only\_named: bool = True*)  
 → Iterable[*owlapy.class\_expression.OWLClassExpression*]

Gets the class expressions that are disjoint with specified class expression with respect to the set of reasoner axioms.

#### Parameters

- **ce** – The class expression whose disjoint classes are to be retrieved.
- **only\_named** – Whether to only retrieve named disjoint classes or also complex class expressions.

### Returns

All class expressions D where the set of reasoner axioms entails `EquivalentClasses(D ObjectComplementOf(ce))` or `StrictSubClassOf(D ObjectComplementOf(ce))`.

**different\_individuals** (*ce*: *owlapy.owl\_individual.OWLNamedIndividual*)  
→ *Iterable[owlapy.owl\_individual.OWLNamedIndividual]*

Gets the individuals that are different from the specified individual with respect to the set of reasoner axioms.

### Parameters

**ind** – The individual whose different individuals are to be retrieved.

### Returns

All individuals x where the set of reasoner axioms entails `DifferentIndividuals(ind x)`.

**same\_individuals** (*ce*: *owlapy.owl\_individual.OWLNamedIndividual*)  
→ *Iterable[owlapy.owl\_individual.OWLNamedIndividual]*

Gets the individuals that are the same as the specified individual with respect to the set of reasoner axioms.

### Parameters

**ind** – The individual whose same individuals are to be retrieved.

### Returns

All individuals x where the root ontology imports closure entails `SameIndividual(ind x)`.

**data\_property\_values** (*ind*: *owlapy.owl\_individual.OWLNamedIndividual*,  
*pe*: *owlapy.owl\_property.OWLDataProperty*, *direct*: *bool = True*)  
→ *Iterable[owlapy.owl\_literal.OWLLiteral]*

Gets the data property values for the specified individual and data property expression.

### Parameters

- **ind** – The individual that is the subject of the data property values.
- **pe** – The data property expression whose values are to be retrieved for the specified individual.
- **direct** – Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

### Returns

A set of OWLLiterals containing literals such that for each literal l in the set, the set of reasoner axioms entails `DataPropertyAssertion(pe ind l)`.

**all\_data\_property\_values** (*pe*: *owlapy.owl\_property.OWLDataProperty*, *direct*: *bool = True*)  
→ *Iterable[owlapy.owl\_literal.OWLLiteral]*

Gets all values for the given data property expression that appear in the knowledge base.

### Parameters

- **pe** – The data property expression whose values are to be retrieved
- **direct** – Specifies if only the direct values of the data property pe should be retrieved (True), or if the values of sub properties of pe should be taken into account (False).

### Returns

A set of OWLLiterals containing literals such that for each literal l in the set, the set of reasoner axioms entails `DataPropertyAssertion(pe ind l)` for any ind.

**object\_property\_values** (*ind*: *owlapy.owl\_individual.OWLNamedIndividual*,  
*pe*: *owlapy.owl\_property.OWLObjectPropertyExpression*, *direct*: *bool = True*)  
→ *Iterable[owlapy.owl\_individual.OWLNamedIndividual]*

Gets the object property values for the specified individual and object property expression.



### Parameters

- **ind** – The individual that is the subject of the object property values.
- **pe** – The object property expression whose values are to be retrieved for the specified individual.
- **direct** – Specifies if the direct values should be retrieved (True), or if all values should be retrieved (False), so that sub properties are taken into account.

### Returns

The named individuals such that for each individual *j*, the set of reasoner axioms entails `ObjectPropertyAssertion(pe ind j)`.

**flush()** → None

Flushes any changes stored in the buffer, which causes the reasoner to take into consideration the changes the current root ontology specified by the changes.

**instances** (*ce*: *owlapy.class\_expression.OWLClassExpression*, *direct*: *bool = False*)  
→ *Iterable[owlapy.owl\_individual.OWLNamedIndividual]*

Gets the individuals which are instances of the specified class expression.

### Parameters

- **ce** – The class expression whose instances are to be retrieved.
- **direct** – Specifies if the direct instances should be retrieved (True), or if all instances should be retrieved (False).

### Returns

If *direct* is True, each named individual *j* where the set of reasoner axioms entails `DirectClassAssertion(ce, j)`. If *direct* is False, each named individual *j* where the set of reasoner axioms entails `ClassAssertion(ce, j)`. If *ce* is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

**sub\_classes** (*ce*: *owlapy.class\_expression.OWLClassExpression*, *direct*: *bool = False*,  
*only\_named*: *bool = True*) → *Iterable[owlapy.class\_expression.OWLClassExpression]*

Gets the set of named classes that are the strict (potentially direct) subclasses of the specified class expression with respect to the reasoner axioms.

### Parameters

- **ce** – The class expression whose strict (direct) subclasses are to be retrieved.
- **direct** – Specifies if the direct subclasses should be retrieved (True) or if the all subclasses (descendant) classes should be retrieved (False).
- **only\_named** – Whether to only retrieve named sub-classes or also complex class expressions.

### Returns

If *direct* is True, each class *C* where reasoner axioms entails `DirectSubClassOf(C, ce)`. If *direct* is False, each class *C* where reasoner axioms entails `StrictSubClassOf(C, ce)`. If *ce* is equivalent to `owl:Nothing` then nothing will be returned.

**super\_classes** (*ce*: *owlapy.class\_expression.OWLClassExpression*, *direct*: *bool = False*,  
*only\_named*: *bool = True*) → *Iterable[owlapy.class\_expression.OWLClassExpression]*

Gets the stream of named classes that are the strict (potentially direct) super classes of the specified class expression with respect to the imports closure of the root ontology.

### Parameters

- **ce** – The class expression whose strict (direct) super classes are to be retrieved.

- **direct** – Specifies if the direct super classes should be retrieved (True) or if the all super classes (ancestors) classes should be retrieved (False).
- **only\_named** – Whether to only retrieve named super classes or also complex class expressions.

#### Returns

If direct is True, each class C where the set of reasoner axioms entails `DirectSubClassOf(ce, C)`. If direct is False, each class C where set of reasoner axioms entails `StrictSubClassOf(ce, C)`. If ce is equivalent to `owl:Thing` then nothing will be returned.

**types** (*ind*: `owlapy.owl_individual.OWLNamedIndividual`, *direct*: `bool = False`)  
 → `Iterable[owlapy.class_expression.OWLClass]`

Gets the named classes which are (potentially direct) types of the specified named individual.

#### Parameters

- **ind** – The individual whose types are to be retrieved.
- **direct** – Specifies if the direct types should be retrieved (True), or if all types should be retrieved (False).

#### Returns

If direct is True, each named class C where the set of reasoner axioms entails `DirectClassAssertion(C, ind)`. If direct is False, each named class C where the set of reasoner axioms entails `ClassAssertion(C, ind)`.

**equivalent\_object\_properties** (*dp*: `owlapy.owl_property.OWLObjectPropertyExpression`)  
 → `Iterable[owlapy.owl_property.OWLObjectPropertyExpression]`

Gets the simplified object properties that are equivalent to the specified object property with respect to the set of reasoner axioms.

#### Parameters

**op** – The object property whose equivalent object properties are to be retrieved.

#### Returns

All simplified object properties e where the root ontology imports closure entails `EquivalentObjectProperties(op e)`. If op is unsatisfiable with respect to the set of reasoner axioms then `owl:bottomDataProperty` will be returned.

**equivalent\_data\_properties** (*dp*: `owlapy.owl_property.OWLDataProperty`)  
 → `Iterable[owlapy.owl_property.OWLDataProperty]`

Gets the data properties that are equivalent to the specified data property with respect to the set of reasoner axioms.

#### Parameters

**dp** – The data property whose equivalent data properties are to be retrieved.

#### Returns

All data properties e where the root ontology imports closure entails `EquivalentDataProperties(dp e)`. If dp is unsatisfiable with respect to the set of reasoner axioms then `owl:bottomDataProperty` will be returned.

**disjoint\_object\_properties** (*dp*: `owlapy.owl_property.OWLObjectPropertyExpression`)  
 → `Iterable[owlapy.owl_property.OWLObjectPropertyExpression]`

Gets the simplified object properties that are disjoint with the specified object property with respect to the set of reasoner axioms.

#### Parameters

**op** – The object property whose disjoint object properties are to be retrieved.

### Returns

All simplified object properties *e* where the root ontology imports closure entails `EquivalentObjectProperties(e ObjectPropertyComplementOf(op))` or `StrictSubObjectPropertyOf(e ObjectPropertyComplementOf(op))`.

**disjoint\_data\_properties** (*dp*: *owlapy.owl\_property.OWLDataProperty*)  
→ *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets the data properties that are disjoint with the specified data property with respect to the set of reasoner axioms.

### Parameters

**dp** – The data property whose disjoint data properties are to be retrieved.

### Returns

All data properties *e* where the root ontology imports closure entails `EquivalentDataProperties(e DataPropertyComplementOf(dp))` or `StrictSubDataPropertyOf(e DataPropertyComplementOf(dp))`.

**sub\_data\_properties** (*dp*: *owlapy.owl\_property.OWLDataProperty*, *direct*: *bool = False*)  
→ *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets the set of named data properties that are the strict (potentially direct) subproperties of the specified data property expression with respect to the imports closure of the root ontology.

### Parameters

- **dp** – The data property whose strict (direct) subproperties are to be retrieved.
- **direct** – Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

### Returns

If *direct* is True, each property *P* where the set of reasoner axioms entails `DirectSubDataPropertyOf(P, pe)`. If *direct* is False, each property *P* where the set of reasoner axioms entails `StrictSubDataPropertyOf(P, pe)`. If *pe* is equivalent to `owl:bottomDataProperty` then nothing will be returned.

**super\_data\_properties** (*dp*: *owlapy.owl\_property.OWLDataProperty*, *direct*: *bool = False*)  
→ *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets the stream of data properties that are the strict (potentially direct) super properties of the specified data property with respect to the imports closure of the root ontology.

### Parameters

- **dp** (*OWLDataProperty*) – The data property whose super properties are to be retrieved.
- **direct** (*bool*) – Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

### Returns

Iterable of super properties.

**super\_object\_properties** (*op*: *owlapy.owl\_property.OWLObjectProperty*, *direct*: *bool = False*)  
→ *Iterable[owlapy.owl\_property.OWLDataProperty]*

Gets the stream of object properties that are the strict (potentially direct) super properties of the specified object property with respect to the imports closure of the root ontology.

### Parameters

- **op** (*OWLObjectPropertyExpression*) – The object property expression whose super properties are to be retrieved.

- **direct** (*bool*) – Specifies if the direct super properties should be retrieved (True) or if the all super properties (ancestors) should be retrieved (False).

#### Returns

Iterable of super properties.

**sub\_object\_properties** (*op: owlapy.owl\_property.OWLObjectPropertyExpression*,  
*direct: bool = False*) → *Iterable[owlapy.owl\_property.OWLObjectPropertyExpression]*

Gets the stream of simplified object property expressions that are the strict (potentially direct) subproperties of the specified object property expression with respect to the imports closure of the root ontology.

#### Parameters

- **op** – The object property expression whose strict (direct) subproperties are to be retrieved.
- **direct** – Specifies if the direct subproperties should be retrieved (True) or if the all subproperties (descendants) should be retrieved (False).

#### Returns

If **direct** is True, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails `DirectSubObjectPropertyOf(P, pe)`. If **direct** is False, simplified object property expressions, such that for each simplified object property expression, P, the set of reasoner axioms entails `StrictSubObjectPropertyOf(P, pe)`. If `pe` is equivalent to `owl:bottomObjectProperty` then nothing will be returned.

**get\_root\_ontology** () → *owlapy.owl\_ontology.OWLOntology*

Gets the “root” ontology that is loaded into this reasoner. The reasoner takes into account the axioms in this ontology and its import’s closure.

**class** `owlapy.owl_reasoner.SyncReasoner` (*ontology: owlapy.owl\_ontology.Ontology*,  
*base\_reasoner: BaseReasoner | None = None*, *infer\_property\_values: bool = True*,  
*infer\_data\_property\_values: bool = True*, *isolate: bool = False*)

Bases: *OntologyReasoner*

Extra convenience methods for OWL Reasoners

**\_\_slots\_\_** = ('\_cnt', '\_conv', '\_base\_reasoner')

**update\_isolated\_ontology** (*axioms\_to\_add: List[owlapy.owl\_axiom.OWLAxiom] = None*,  
*axioms\_to\_remove: List[owlapy.owl\_axiom.OWLAxiom] = None*)

Add or remove axioms to the isolated ontology that the reasoner is using.

#### Parameters

- **axioms\_to\_add** (*List [OWLAxiom]*) – Axioms to add to the isolated ontology.
- **axioms\_to\_remove** (*List [OWLAxiom]*) – Axioms to remove from the isolated ontology.

**instances** (*ce: owlapy.class\_expression.OWLClassExpression*, *direct: bool = False*)  
→ *Iterable[owlapy.owl\_individual.OWLNamedIndividual]*

Gets the individuals which are instances of the specified class expression.

#### Parameters

- **ce** – The class expression whose instances are to be retrieved.
- **direct** – Specifies if the direct instances should be retrieved (True), or if all instances should be retrieved (False).

### Returns

If `direct` is `True`, each named individual `j` where the set of reasoner axioms entails `DirectClassAssertion(ce, j)`. If `direct` is `False`, each named individual `j` where the set of reasoner axioms entails `ClassAssertion(ce, j)`. If `ce` is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

`__del__()`

## owlapy.parser

String to OWL parsers.

### Attributes

<i>IntegerOWLDatatype</i>
<i>BooleanOWLDatatype</i>
<i>DoubleOWLDatatype</i>
<i>StringOWLDatatype</i>
<i>DateOWLDatatype</i>
<i>DateTimeOWLDatatype</i>
<i>DurationOWLDatatype</i>
<i>MANCHESTER_GRAMMAR</i>
<i>DL_GRAMMAR</i>
<i>DLparser</i>
<i>ManchesterParser</i>

### Classes

<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>OWLNamedIndividual</i>	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
<i>OWLObjectPropertyExpression</i>	A high level interface to describe different types of object properties.
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.

continues on next page

Table 9 – continued from previous page

<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
<i>OWLObjectParser</i>	Abstract class with a parse method to parse a string to an OWL Object.
<i>Namespaces</i>	Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
<i>OWLFacet</i>	Enumerations for OWL facets.
<i>OWLRDFVocabulary</i>	Enumerations for OWL/RDF vocabulary.
<i>OWLObjectHasSelf</i>	A self-restriction <code>ObjectHasSelf( OPE )</code> consists of an object property expression OPE,
<i>OWLObjectIntersectionOf</i>	An intersection class expression <code>ObjectIntersectionOf( CE1 ... CEn )</code> contains all individuals that are instances
<i>OWLObjectMinCardinality</i>	A minimum cardinality expression <code>ObjectMinCardinality( n OPE CE )</code> consists of a nonnegative integer n, an object
<i>OWLObjectSomeValuesFrom</i>	An existential class expression <code>ObjectSomeValuesFrom( OPE CE )</code> consists of an object property expression OPE and
<i>OWLObjectUnionOf</i>	A union class expression <code>ObjectUnionOf( CE1 ... CEn )</code> contains all individuals that are instances
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>OWLObjectOneOf</i>	An enumeration of individuals <code>ObjectOneOf( a1 ... an )</code> contains exactly the individuals $a_i$ with $1 \leq i \leq n$ .
<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLObjectComplementOf</i>	Represents an <code>ObjectComplementOf</code> class expression in the OWL 2 Specification.
<i>OWLObjectExactCardinality</i>	An exact cardinality expression <code>ObjectExactCardinality( n OPE CE )</code> consists of a nonnegative integer n, an object
<i>OWLQuantifiedDataRestriction</i>	Represents a quantified data restriction.
<i>OWLQuantifiedObjectRestriction</i>	Represents a quantified object restriction.
<i>OWLFacetRestriction</i>	A facet restriction is used to restrict a particular datatype.
<i>OWLDDataSomeValuesFrom</i>	An existential class expression <code>DataSomeValuesFrom( DPE1 ... DPEn DR )</code> consists of n data property expressions
<i>OWLDDataExactCardinality</i>	An exact cardinality expression <code>ObjectExactCardinality( n OPE CE )</code> consists of a nonnegative integer n, an
<i>OWLObjectHasValue</i>	A has-value class expression <code>ObjectHasValue( OPE a )</code> consists of an object property expression OPE and an
<i>OWLDDataMaxCardinality</i>	A maximum cardinality expression <code>ObjectMaxCardinality( n OPE CE )</code> consists of a nonnegative integer n, an object
<i>OWLObjectMaxCardinality</i>	A maximum cardinality expression <code>ObjectMaxCardinality( n OPE CE )</code> consists of a nonnegative integer n, an object
<i>OWLDDataMinCardinality</i>	A minimum cardinality expression <code>DataMinCardinality( n DPE DR )</code> consists of a nonnegative integer n, a data

continues on next page

Table 9 – continued from previous page

<i>OWLDataHasValue</i>	A has-value class expression <i>DataHasValue</i> ( DPE lt ) consists of a data property expression DPE and a literal lt,
<i>OWLDataOneOf</i>	An enumeration of literals <i>DataOneOf</i> ( lt1 ... ltn ) contains exactly the explicitly specified literals lti with
<i>OWLObjectCardinalityRestriction</i>	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
<i>OWLDatatypeRestriction</i>	A datatype restriction <i>DatatypeRestriction</i> ( DT F1 lt1 ... Fn ltn ) consists of a unary datatype DT and n pairs
<i>OWLDataCardinalityRestriction</i>	Represents Data Property Cardinality Restrictions.
<i>OWLObjectAllValuesFrom</i>	A universal class expression <i>ObjectAllValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and a
<i>OWLDataAllValuesFrom</i>	A universal class expression <i>DataAllValuesFrom</i> ( DPE1 ... DPEn DR ) consists of n data property expressions DPEi,
<i>OWLDataIntersectionOf</i>	An intersection data range <i>DataIntersectionOf</i> ( DR1 ... DRn ) contains all tuples of literals that are contained
<i>OWLDataUnionOf</i>	A union data range <i>DataUnionOf</i> ( DR1 ... DRn ) contains all tuples of literals that are contained in the at least
<i>OWLDataComplementOf</i>	A complement data range <i>DataComplementOf</i> ( DR ) contains all tuples of literals that are not contained in the
<i>OWLDatRange</i>	Represents a <i>DataRange</i> in the OWL 2 Specification.
<i>ManchesterOWLSyntaxParser</i>	Manchester Syntax parser to parse strings to OWLClass-Expressions.
<i>DLSyntaxParser</i>	Description Logic Syntax parser to parse strings to OWL-ClassExpressions.

## Functions

```

dl_to_owl_expression(dl_expression, namespace)
manchester_to_owl_expression(manchester_ex
...)
```

## Module Contents

```

class owlapy.parser.IRI (namespace: str | owlapy.namespaces.Namespaces, remainder: str)
    Bases: owlapy.owl_annotation.OWLAnnotationSubject, owlapy.owl_annotation.
            OWLAnnotationValue
    An IRI, consisting of a namespace and a remainder.
    __slots__ = ('_namespace', '_remainder', '__weakref__')
    type_index: Final = 0
    static create (namespace: owlapy.namespaces.Namespaces, remainder: str) → IRI
    static create (namespace: str, remainder: str) → IRI
    static create (string: str) → IRI
```

**\_\_repr\_\_()**

Return repr(self).

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

Return self==value.

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

Return hash(self).

**is\_nothing()**

Determines if this IRI is equal to the IRI that owl:Nothing is named with.

**Returns**

True if this IRI is equal to <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.

**is\_thing()**

Determines if this IRI is equal to the IRI that owl:Thing is named with.

**Returns**

True if this IRI is equal to <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

**is\_reserved\_vocabulary()** → bool

Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

**Returns**

True if the IRI is in the reserved vocabulary, otherwise False.

**as\_iri()** → *IRI*

**Returns**

if the value is an IRI, return it. Return None otherwise.

**as\_str()** → str

CD: Should be deprecated. :returns: The string that specifies the IRI.

**property str:** str

Returns: The string that specifies the IRI.

**property reminder:** str

Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form()** → str

Gets the short form.

**Returns**

A string that represents the short form.

**get\_namespace()** → str

**Returns**

The namespace as string.

**get\_remainder()** → str

**Returns**

The remainder (coincident with NCName usually) for this IRI.



```

class owlapy.parser.OWLNamedIndividual (iri: owlapy.iri.IRI | str)
    Bases: OWLIndividual, owlapy.owl_object.OWLEntity

    Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities. IRIs
    from the reserved vocabulary must not be used to identify named individuals in an OWL 2 DL ontology.

    (https://www.w3.org/TR/owl2-syntax/#Named\_Individuals)

    __slots__ = '_iri'

    type_index: Final = 1005

    property iri: owlapy.iri.IRI
        Gets the IRI of this object.

        Returns
            The IRI of this object.

    property str
        Gets the string representation of this object

        Returns
            The IRI as string

owlapy.parser.IntegerOWLDatatype: Final
owlapy.parser.BooleanOWLDatatype: Final
owlapy.parser.DoubleOWLDatatype: Final
owlapy.parser.StringOWLDatatype: Final
owlapy.parser.DateOWLDatatype: Final
owlapy.parser.DateTimeOWLDatatype: Final
owlapy.parser.DurationOWLDatatype: Final

class owlapy.parser.OWLLiteral
    Bases: owlapy.owl_annotation.OWLAnnotationValue

    Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and
    can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string,
    and a datatype.

    (https://www.w3.org/TR/owl2-syntax/#Literals)

    __slots__ = ()

    type_index: Final = 4008

    get_literal () → str
        Gets the lexical value of this literal. Note that the language tag is not included.

        Returns
            The lexical value of this literal.

    is_boolean () → bool
        Whether this literal is typed as boolean.

```

**parse\_boolean()** → bool

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("<http://www.w3.org/2001/XMLSchema#boolean>").

**Returns**

A bool value that is represented by this literal.

**is\_double()** → bool

Whether this literal is typed as double.

**parse\_double()** → float

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

**Returns**

A double value that is represented by this literal.

**is\_integer()** → bool

Whether this literal is typed as integer.

**parse\_integer()** → int

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

**Returns**

An integer value that is represented by this literal.

**is\_string()** → bool

Whether this literal is typed as string.

**parse\_string()** → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("<http://www.w3.org/2001/XMLSchema#string>").

**Returns**

A string value that is represented by this literal.

**is\_date()** → bool

Whether this literal is typed as date.

**parse\_date()** → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("<http://www.w3.org/2001/XMLSchema#date>").

**Returns**

A date value that is represented by this literal.

**is\_datetime()** → bool

Whether this literal is typed as dateTime.

**parse\_datetime()** → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("<http://www.w3.org/2001/XMLSchema#dateTime>").

**Returns**

A datetime value that is represented by this literal.

**is\_duration()** → bool

Whether this literal is typed as duration.

**parse\_duration()** → pandas.Timedelta

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("<http://www.w3.org/2001/XMLSchema#duration>").

**Returns**

A Timedelta value that is represented by this literal.

**is\_literal()** → bool

**Returns**

true if the annotation value is a literal

**as\_literal()** → *OWLLiteral*

**Returns**

if the value is a literal, returns it. Return None otherwise

**to\_python()** → Literals

**abstract get\_datatype()** → *owlapy.owl\_datatype.OWLDatatype*

Gets the OWLDatatype which types this literal.

**Returns**

The OWLDatatype that types this literal.

**class owlapy.parser.OWLObjectPropertyExpression**

Bases: OWLPropertyExpression

A high level interface to describe different types of object properties.

**\_\_slots\_\_ = ()**

**abstract get\_inverse\_property()** → *OWLObjectPropertyExpression*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**abstract get\_named\_property()** → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**is\_object\_property\_expression()** → bool

**Returns**

True if this is an object property.

**class owlapy.parser.OWLObjectProperty** (*iri: owlapy.iri.IRI | str*)

Bases: *OWLObjectPropertyExpression*, OWLProperty

Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals. ([https://www.w3.org/TR/owl2-syntax/#Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Object_Properties))

**\_\_slots\_\_ = '\_iri'**

**type\_index: Final = 1002**

**get\_named\_property** () → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**get\_inverse\_property** () → *OWLObjectInverseOf*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**is\_owl\_top\_object\_property** () → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

**class** owlapy.parser.OWLDataProperty (*iri: owlapy.iri.IRI | str*)

Bases: OWLDataPropertyExpression, OWLProperty

Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.

([https://www.w3.org/TR/owl2-syntax/#Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Data_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index:** Final = 1004

**is\_owl\_top\_data\_property** () → bool

Determines if this is the owl:topDataProperty.

**Returns**

topDataProperty.

**Return type**

True if this property is the owl

**class** owlapy.parser.OWLObjectParser

Abstract class with a parse method to parse a string to an OWL Object.

**abstract parse\_expression** (*expression\_str: str*) → *OWLObject*

Parse a string to an OWL Object.

**Parameters**

**expression\_str** (*str*) – Expression string.

**Returns**

The OWL Object which is represented by the string.

**class** owlapy.parser.Namespaces (*prefix: str, ns: str*)

Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references

**\_\_slots\_\_** = ('\_prefix', '\_ns')

**property ns:** str

```

property prefix: str

__repr__()
    Return repr(self).

__hash__()
    Return hash(self).

__eq__(other)
    Return self==value.

```

**class** owlapy.parser.OWLDatatype (iri: owlapy.iri.IRI | owlapy.meta\_classes.HasIRI)

Bases: owlapy.owl\_object.OWLEntity, owlapy.owl\_data\_ranges.OWLDataRange

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

```

__slots__ = '_iri'

type_index: Final = 4001

property iri: owlapy.iri.IRI
    Gets the IRI of this object.

    Returns
        The IRI of this object.

property str: str
    Gets the string representation of this object

    Returns
        The IRI as string

```

**class** owlapy.parser.OWLFacet (remainder: str, symbolic\_form: str, operator: Callable[[\_X, \_X], bool])

Bases: \_Vocabulary, enum.Enum

Enumerations for OWL facets.

```

property symbolic_form

property operator

static from_str (name: str) → OWLFacet

MIN_INCLUSIVE: Final

MIN_EXCLUSIVE: Final

MAX_INCLUSIVE: Final

MAX_EXCLUSIVE: Final

LENGTH: Final

```

```

MIN_LENGTH: Final

MAX_LENGTH: Final

PATTERN: Final

TOTAL_DIGITS: Final

FRACTION_DIGITS: Final

class owlapy.parser.OWLRDFVocabulary (namespace: owlapy.namespaces.Namespaces,
    remainder: str)
    Bases: _Vocabulary, enum, Enum
    Enumerations for OWL/RDF vocabulary.

    OWL_THING

    OWL_NOTHING

    OWL_CLASS

    OWL_NAMED_INDIVIDUAL

    OWL_TOP_OBJECT_PROPERTY

    OWL_BOTTOM_OBJECT_PROPERTY

    OWL_TOP_DATA_PROPERTY

    OWL_BOTTOM_DATA_PROPERTY

    RDFS_LITERAL

class owlapy.parser.OWLObjectHasSelf (
    property: owlapy.owl_property.OWLObjectPropertyExpression)
    Bases: OWLObjectRestriction

    A self-restriction ObjectHasSelf( OPE ) consists of an object property expression OPE, and it contains all those
    individuals that are connected by OPE to themselves. (https://www.w3.org/TR/owl2-syntax/#Self-Restriction)

    __slots__ = '_property'

    type_index: Final = 3011

    get_property () → owlapy.owl_property.OWLObjectPropertyExpression

        Returns
            Property being restricted.

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

    __repr__ ()
        Return repr(self).

```

```

class owlapy.parser.OwlObjectIntersectionOf (
    operands: Iterable[owlapy.class_expression.class_expression.OwlClassExpression])
Bases: OwlNaryBooleanClassExpression

An intersection class expression ObjectIntersectionOf( CE1 ... CEn ) contains all individuals that are instances of
all class expressions CEi for  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Intersection\_of\_Class\_Expressions)

__slots__ = '_operands'

type_index: Final = 3001

class owlapy.parser.OwlObjectMinCardinality (cardinality: int,
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
Bases: OwlObjectCardinalityRestriction

A minimum cardinality expression ObjectMinCardinality( n OPE CE ) consists of a nonnegative integer n, an object
property expression OPE, and a class expression CE, and it contains all those individuals that are connected by
OPE to at least n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Minimum\_Cardinality)

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3008

class owlapy.parser.OwlObjectSomeValuesFrom (
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OwlClassExpression)
Bases: OwlQuantifiedObjectRestriction

An existential class expression ObjectSomeValuesFrom( OPE CE ) consists of an object property expression OPE
and a class expression CE, and it contains all those individuals that are connected by OPE to an individual that is
an instance of CE.

__slots__ = ('_property', '_filler')

type_index: Final = 3005

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

get_property () → owlapy.owl_property.OwlObjectPropertyExpression

Returns
    Property being restricted.

class owlapy.parser.OwlObjectUnionOf (
    operands: Iterable[owlapy.class_expression.class_expression.OwlClassExpression])
Bases: OwlNaryBooleanClassExpression

A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances of at least one
class expression CEi for  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Class\_Expressions)

```

```

__slots__ = '_operands'

type_index: Final = 3002

class owlapy.parser.OWLClass (iri: owlapy.iri.IRI | str)
    Bases: owlapy.class_expression.class_expression.OWLClassExpression, owlapy.
            owl_object.OWLEntity

    An OWL 2 named Class. Classes can be understood as sets of individuals. (https://www.w3.org/TR/owl2-syntax/#Classes)

    __slots__ = ('_iri', '_is_nothing', '_is_thing')

    type_index: Final = 1001

    property iri: owlapy.iri.IRI
        Gets the IRI of this object.

        Returns
            The IRI of this object.

    property str
        Gets the string representation of this object

        Returns
            The IRI as string

    property reminder: str
        The reminder of the IRI

    is_owl_thing () → bool
        Determines if this expression is the built in class owl:Thing. This method does not determine if the class is
        equivalent to owl:Thing.

        Returns
            Thing.

        Return type
            True if this expression is owl

    is_owl_nothing () → bool
        Determines if this expression is the built in class owl:Nothing. This method does not determine if the class
        is equivalent to owl:Nothing.

    get_object_complement_of ()
        → owlapy.class_expression.class_expression.OWLObjectComplementOf
        Gets the object complement of this class expression.

        Returns
            A class expression that is the complement of this class expression.

    get_nnf () → OWLClass
        Gets the negation normal form of the complement of this expression.

        Returns
            A expression that represents the NNF of the complement of this expression.

```



```

class owlapy.parser.OWLObjectOneOf (
    values: owlapy.owl_individual.OWLIndividual | Iterable[owlapy.owl_individual.OWLIndividual])
    Bases: owlapy.class_expression.class_expression.OWLAnonymousClassExpression,
    owlapy.meta_classes.HasOperands[owlapy.owl_individual.OWLIndividual]

    An enumeration of individuals ObjectOneOf( a1 ... an ) contains exactly the individuals ai with  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Enumeration\_of\_Individuals)

    __slots__ = '_values'

    type_index: Final = 3004

    individuals () → Iterable[owlapy.owl_individual.OWLIndividual]
        Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this
        class expression.

        Returns
            The individuals that are the values of this {@code ObjectOneOf} class expression.

    operands () → Iterable[owlapy.owl_individual.OWLIndividual]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
            The operands.

    as_object_union_of () → owlapy.class_expression.class_expression.OWLClassExpression
        Simplifies this enumeration to a union of singleton nominals.

        Returns
            This enumeration in a more standard DL form.  $\text{simp}(\{a\}) = \{a\}$   $\text{simp}(\{a_0, \dots, \{a_n\}\}) = \text{unionOf}(\{a_0\}, \dots, \{a_n\})$ 

    __hash__ ()
        Return hash(self).

    __eq__ (other)
        Return self==value.

    __repr__ ()
        Return repr(self).

class owlapy.parser.OWLClassExpression
    Bases: owlapy.owl_data_ranges.OWLPropertyRange

    OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' prop-
    erties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the
    structural specification of OWL 2, class expressions are represented by ClassExpression. (https://www.w3.org/TR/owl2-syntax/#Class\_Expressions)

    __slots__ = ()

    abstract is_owl_thing () → bool
        Determines if this expression is the built in class owl:Thing. This method does not determine if the class is
        equivalent to owl:Thing.

        Returns
            Thing.

        Return type
            True if this expression is owl

```

```

abstract is_owl_nothing () → bool
    Determines if this expression is the built in class owl:Nothing. This method does not determine if the class
    is equivalent to owl:Nothing.

abstract get_object_complement_of () → OWLObjectComplementOf
    Gets the object complement of this class expression.

    Returns
        A class expression that is the complement of this class expression.

abstract get_nnf () → OWLClassExpression
    Gets the negation normal form of the complement of this expression.

    Returns
        A expression that represents the NNF of the complement of this expression.

class owlapy.parser.OWLObjectComplementOf (op: OWLClassExpression)
    Bases: OWLBooleanClassExpression, owlapy.meta_classes.
    HasOperands[OWLClassExpression]
    Represents an ObjectComplementOf class expression in the OWL 2 Specification.

    __slots__ = '_operand'

    type_index: Final = 3003

    get_operand () → OWLClassExpression

    Returns
        The wrapped expression.

    operands () → Iterable[OWLClassExpression]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

    Returns
        The operands.

    __repr__ ()
        Return repr(self).

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

class owlapy.parser.OWLObjectExactCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
    Bases: OWLObjectCardinalityRestriction

    An exact cardinality expression ObjectExactCardinality( n OPE CE ) consists of a nonnegative integer n,
    an object
    property expression OPE, and a class expression CE, and it contains all those individuals that are connected
    by to exactly n different individuals that are instances of CE.

    (https://www.w3.org/TR/owl2-syntax/#Exact\_Cardinality)

    __slots__ = ('_cardinality', '_filler', '_property')

```

```

type_index: Final = 3009

as_intersection_of_min_max()
    → owlapy.class_expression.nary_boolean_expression.OwlObjectIntersectionOf

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

Returns
    The semantically equivalent but structurally simpler form ( $= 1 R C$ )  $= \geq 1 R C$  and  $\leq 1 R C$ .

class owlapy.parser.OwlQuantifiedDataRestriction(
    filler: owlapy.owl_data_ranges.OwlDataRange)

Bases: OwlQuantifiedRestriction[owlapy.owl_data_ranges.OwlDataRange], OwlDataRestriction

Represents a quantified data restriction.

__slots__ = ()

get_filler() → owlapy.owl_data_ranges.OwlDataRange

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

Returns
    the value

class owlapy.parser.OwlQuantifiedObjectRestriction(
    filler: owlapy.class_expression.class_expression.OwlClassExpression)

Bases: OwlQuantifiedRestriction[owlapy.class_expression.class_expression.OwlClassExpression], OwlObjectRestriction

Represents a quantified object restriction.

__slots__ = ()

get_filler() → owlapy.class_expression.class_expression.OwlClassExpression

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

Returns
    the value

class owlapy.parser.OwlFacetRestriction(facet: owlapy.vocab.OwlFacet, literal: Literals)

Bases: owlapy.owl_object.OwlObject

A facet restriction is used to restrict a particular datatype.

__slots__ = ('_facet', '_literal')

type_index: Final = 4007

get_facet() → owlapy.vocab.OwlFacet

get_facet_value() → owlapy.owl_literal.OwlLiteral

__eq__(other)

Return self==value.

```

**\_\_hash\_\_()**  
Return hash(self).

**\_\_repr\_\_()**  
Return repr(self).

```
class owlapy.parser.OWLDataSomeValuesFrom (
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: *OWLQuantifiedDataRestriction*

An existential class expression `DataSomeValuesFrom( DPE1 ... DPEn DR )` consists of  $n$  data property expressions  $DPE_i$ ,  $1 \leq i \leq n$ , and a data range  $DR$  whose arity must be  $n$ . Such a class expression contains all those individuals that are connected by  $DPE_i$  to literals  $l_{ti}$ ,  $1 \leq i \leq n$ , such that the tuple  $( l_{t1} , \dots , l_{tn} )$  is in  $DR$ . A class expression of the form `DataSomeValuesFrom( DPE DR )` can be seen as a syntactic shortcut for the class expression `DataMinCardinality( 1 DPE DR )`. ([https://www.w3.org/TR/owl2-syntax/#Existential\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Existential_Quantification_2))

**\_\_slots\_\_** = **'\_property'**

**type\_index:** **Final** = 3012

**\_\_repr\_\_()**  
Return repr(self).

**\_\_eq\_\_** (*other*)  
Return self==value.

**\_\_hash\_\_()**  
Return hash(self).

**get\_property()** → *owlapy.owl\_property.OWLDataPropertyExpression*

#### Returns

Property being restricted.

```
class owlapy.parser.OWLDataExactCardinality (cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: *OWLDataCardinalityRestriction*

An exact cardinality expression `ObjectExactCardinality( n OPE CE )` consists of a nonnegative integer  $n$ , an object property expression  $OPE$ , and a class expression  $CE$ , and it contains all those individuals that are connected

by  $OPE$  to exactly  $n$  different individuals that are instances of  $CE$  ([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

**\_\_slots\_\_** = (**'\_cardinality'**, **'\_filler'**, **'\_property'**)

**type\_index:** **Final** = 3016

**as\_intersection\_of\_min\_max()**  
→ *owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

#### Returns

The semantically equivalent but structurally simpler form  $(= 1 \text{ R } D) = \geq 1 \text{ R } D$  and  $\leq 1 \text{ R } D$ .

```

class owlapy.parser.OWLObjectHasValue (
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    individual: owlapy.owl_individual.OWLIndividual)

Bases: OWLHasValueRestriction[owlapy.owl_individual.OWLIndividual], OWLObjectRestriction

A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ). (https://www.w3.org/TR/owl2-syntax/#Individual\_Value\_Restriction)

__slots__ = ('_property', '_v')

type_index: Final = 3007

get_property() → owlapy.owl_property.OWLObjectPropertyExpression

    Returns
        Property being restricted.

as_some_values_from() → owlapy.class_expression.class_expression.OWLClassExpression
    A convenience method that obtains this restriction as an existential restriction with a nominal filler.

    Returns
        The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

__repr__()
    Return repr(self).

class owlapy.parser.OWLDataMaxCardinality (cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)

Bases: OWLDataCardinalityRestriction

A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to at most n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Maximum\_Cardinality)

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3017

class owlapy.parser.OWLObjectMaxCardinality (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)

Bases: OWLObjectCardinalityRestriction

A maximum cardinality expression ObjectMaxCardinality( n OPE CE ) consists of a nonnegative integer n, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE
    to at most n different individuals that are instances of CE. (https://www.w3.org/TR/owl2-syntax/#Maximum\_Cardinality)

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3010

```

```
class owlapy.parser.OWLDataMinCardinality (cardinality: int,
      property: owlapy.owl_property.OWLDataPropertyExpression,
      filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: *OWLDataCardinalityRestriction*

A minimum cardinality expression *DataMinCardinality*( *n* *DPE* *DR* ) consists of a nonnegative integer *n*, a data property expression *DPE*, and a unary data range *DR*, and it contains all those individuals that are connected by *DPE* to at least *n* different literals in *DR*. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3015
```

```
class owlapy.parser.OWLDataHasValue (
      property: owlapy.owl_property.OWLDataPropertyExpression,
      value: owlapy.owl_literal.OWLLiteral)
```

Bases: *OWLHasValueRestriction*[*owlapy.owl\_literal.OWLLiteral*], *OWLDataRestriction*

A has-value class expression *DataHasValue*( *DPE* *It* ) consists of a data property expression *DPE* and a literal *It*, and it contains all those individuals that are connected by *DPE* to *It*. Each such class expression can be seen as a syntactic shortcut for the class expression *DataSomeValuesFrom*( *DPE* *DataOneOf*( *It* ) ). ([https://www.w3.org/TR/owl2-syntax/#Literal\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Literal_Value_Restriction))

```
__slots__ = '_property'
```

```
type_index: Final = 3014
```

```
__repr__ ()
```

Return repr(self).

```
__eq__ (other)
```

Return self==value.

```
__hash__ ()
```

Return hash(self).

```
as_some_values_from () → owlapy.class_expression.class_expression.OWLClassExpression
```

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

**Returns**

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

```
get_property () → owlapy.owl_property.OWLDataPropertyExpression
```

**Returns**

Property being restricted.

```
class owlapy.parser.OWLDataOneOf (
      values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])
```

Bases: *owlapy.owl\_data\_ranges.OWLDataRange*, *owlapy.meta\_classes.HasOperands*[*owlapy.owl\_literal.OWLLiteral*]

An enumeration of literals *DataOneOf*( *It*<sub>1</sub> ... *It*<sub>*n*</sub> ) contains exactly the explicitly specified literals *It*<sub>*i*</sub> with  $1 \leq i \leq n$ . The resulting data range has arity one. ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Literals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Literals))

```
type_index: Final = 4003
```

**values** () → Iterable[*owlapy.owl\_literal.OWLLiteral*]  
 Gets the values that are in the oneOf.

**Returns**  
 The values of this {`@code DataOneOf`} class expression.

**operands** () → Iterable[*owlapy.owl\_literal.OWLLiteral*]  
 Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**  
 The operands.

**\_\_hash\_\_** ()  
 Return hash(self).

**\_\_eq\_\_** (other)  
 Return self==value.

**\_\_repr\_\_** ()  
 Return repr(self).

**class** owlapy.parser.**OWLObjectCardinalityRestriction** (cardinality: int,  
 property: *owlapy.owl\_property.OWLObjectPropertyExpression*,  
 filler: *owlapy.class\_expression.class\_expression.OWLClassExpression*)  
 Bases: *OWLCardinalityRestriction[owlapy.class\_expression.class\_expression.OWLClassExpression]*, *OWLQuantifiedObjectRestriction*  
 Represents Object Property Cardinality Restrictions in the OWL 2 specification.

**\_\_slots\_\_** = ()

**get\_property** () → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**  
 Property being restricted.

**\_\_repr\_\_** ()  
 Return repr(self).

**\_\_eq\_\_** (other)  
 Return self==value.

**\_\_hash\_\_** ()  
 Return hash(self).

**class** owlapy.parser.**OWLDatatypeRestriction** (type\_: *owlapy.owl\_datatype.OWLDatatype*,  
 facet\_restrictions: *OWLFacetRestriction* | Iterable[*OWLFacetRestriction*])  
 Bases: *owlapy.owl\_data\_ranges.OWLDataRange*  
 A datatype restriction DatatypeRestriction( DT F1 lt1 ... Fn ltn ) consists of a unary datatype DT and n pairs ( Fi , lti ). The resulting data range is unary and is obtained by restricting the value space of DT according to the semantics of all ( Fi , vi ) (multiple pairs are interpreted conjunctively), where vi are the data values of the literals lti. ([https://www.w3.org/TR/owl2-syntax/#Datatype\\_Restrictions](https://www.w3.org/TR/owl2-syntax/#Datatype_Restrictions))

**\_\_slots\_\_** = ('\_type', '\_facet\_restrictions')

**type\_index**: Final = 4006

**get\_datatype** () → *owlapy.owl\_datatype.OWLDatatype*

```

get_facet_restrictions () → Sequence[OWLFacetRestriction]

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

__repr__ ()
    Return repr(self).

class owlapy.parser.OWLDataCardinalityRestriction (cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)

    Bases: OWLCardinalityRestriction[owlapy.owl_data_ranges.OWLDataRange],
    OWLQuantifiedDataRestriction, OWLDataRestriction

    Represents Data Property Cardinality Restrictions.

    __slots__ = ()

    get_property () → owlapy.owl_property.OWLDataPropertyExpression

        Returns
        Property being restricted.

    __repr__ ()
        Return repr(self).

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

class owlapy.parser.OWLObjectAllValuesFrom (
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)

    Bases: OWLQuantifiedObjectRestriction

    A universal class expression ObjectAllValuesFrom( OPE CE ) consists of an object property expression OPE and
    a class expression CE, and it contains all those individuals that are connected by OPE only to individuals that are
    instances of CE. (https://www.w3.org/TR/owl2-syntax/#Universal\_Quantification)

    __slots__ = ('_property', '_filler')

    type_index: Final = 3006

    __repr__ ()
        Return repr(self).

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

```



**get\_property** () → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.parser.OWLDataAllValuesFrom(  
    property: owlapy.owl_property.OWLDataPropertyExpression,  
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: *OWLQuantifiedDataRestriction*

A universal class expression *DataAllValuesFrom*( DPE<sub>1</sub> ... DPE<sub>n</sub> DR ) consists of n data property expressions DPE<sub>i</sub>, 1 ≤ i ≤ n, and a data range DR whose arity must be n. Such a class expression contains all those individuals that

**are connected by DPE<sub>i</sub> only to literals lti, 1 ≤ i ≤ n, such that each tuple ( lt<sub>1</sub> , ..., lt<sub>n</sub> ) is in DR.**

**A class**

expression of the form *DataAllValuesFrom*( DPE DR ) can be seen as a syntactic shortcut for the class expression *DataMaxCardinality*( 0 DPE *DataComplementOf*( DR ) ). ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification_2))

```
__slots__ = '_property'
```

```
type_index: Final = 3013
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.parser.OWLDataIntersectionOf(operands: Iterable[OWLDataRange])
```

Bases: *OWLNaryDataRange*

An intersection data range *DataIntersectionOf*( DR<sub>1</sub> ... DR<sub>n</sub> ) contains all tuples of literals that are contained in each data range DR<sub>i</sub> for 1 ≤ i ≤ n. All data ranges DR<sub>i</sub> must be of the same arity, and the resulting data range is of that arity as well.

([https://www.w3.org/TR/owl2-syntax/#Intersection\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Intersection_of_Data_Ranges))

```
__slots__ = '_operands'
```

```
type_index: Final = 4004
```

```
class owlapy.parser.OWLDataUnionOf(operands: Iterable[OWLDataRange])
```

Bases: *OWLNaryDataRange*

A union data range *DataUnionOf*( DR<sub>1</sub> ... DR<sub>n</sub> ) contains all tuples of literals that are contained in the at least one data range DR<sub>i</sub> for 1 ≤ i ≤ n. All data ranges DR<sub>i</sub> must be of the same arity, and the resulting data range is of that arity as well.

([https://www.w3.org/TR/owl2-syntax/#Union\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Union_of_Data_Ranges))

```
__slots__ = '_operands'
```

```

type_index: Final = 4005

class owlapy.parser.OWLDataComplementOf (data_range: OWLDataRange)
    Bases: OWLDataRange

    A complement data range DataComplementOf( DR ) contains all tuples of literals that are not contained in the
    data range DR. The resulting data range has the arity equal to the arity of DR.

    (https://www.w3.org/TR/owl2-syntax/#Complement\_of\_Data\_Ranges)

    type_index: Final = 4002

    get_data_range () → OWLDataRange

        Returns
            The wrapped data range.

    __repr__ ()
        Return repr(self).

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

class owlapy.parser.OWLDataRange
    Bases: OWLPropertyRange

    Represents a DataRange in the OWL 2 Specification.

owlapy.parser.MANCHESTER_GRAMMAR

class owlapy.parser.ManchesterOWLSyntaxParser (
    namespace: str | owlapy.namespaces.Namespaces | None = None, grammar=None)
    Bases: parsimonious.nodes.NodeVisitor, owlapy.owl_object.OWLObjectParser

    Manchester Syntax parser to parse strings to OWLClassExpressions. Following: https://www.w3.org/TR/owl2-manchester-syntax.

    slots = ('ns', 'grammar')

    ns: str | owlapy.namespaces.Namespaces | None

    parse_expression (expression_str: str) → owlapy.class_expression.OWLClassExpression
        Parse a string to an OWL Object.

        Parameters
            expression_str (str) – Expression string.

        Returns
            The OWL Object which is represented by the string.

    visit_union (node, children) → owlapy.class_expression.OWLClassExpression

    visit_intersection (node, children) → owlapy.class_expression.OWLClassExpression

    visit_primary (node, children) → owlapy.class_expression.OWLClassExpression

    visit_some_only_res (node, children) → owlapy.class_expression.OWLQuantifiedObjectRestriction

```

**visit\_cardinality\_res** (*node*, *children*)  
     → *owlapy.class\_expression.OWLObjectCardinalityRestriction*

**visit\_value\_res** (*node*, *children*) → *owlapy.class\_expression.OWLObjectHasValue*

**visit\_has\_self** (*node*, *children*) → *owlapy.class\_expression.OWLObjectHasSelf*

**visit\_object\_property** (*node*, *children*) → *owlapy.owl\_property.OWLObjectPropertyExpression*

**visit\_class\_expression** (*node*, *children*) → *owlapy.class\_expression.OWLClassExpression*

**visit\_individual\_list** (*node*, *children*) → *owlapy.class\_expression.OWLObjectOneOf*

**visit\_data\_primary** (*node*, *children*) → *owlapy.owl\_data\_ranges.OWLDataRange*

**visit\_data\_some\_only\_res** (*node*, *children*)  
     → *owlapy.class\_expression.OWLQuantifiedDataRestriction*

**visit\_data\_cardinality\_res** (*node*, *children*)  
     → *owlapy.class\_expression.OWLDataCardinalityRestriction*

**visit\_data\_value\_res** (*node*, *children*) → *owlapy.class\_expression.OWLDataHasValue*

**visit\_data\_union** (*node*, *children*) → *owlapy.owl\_data\_ranges.OWLDataRange*

**visit\_data\_intersection** (*node*, *children*) → *owlapy.owl\_data\_ranges.OWLDataRange*

**visit\_literal\_list** (*node*, *children*) → *owlapy.class\_expression.OWLDataOneOf*

**visit\_data\_parentheses** (*node*, *children*) → *owlapy.owl\_data\_ranges.OWLDataRange*

**visit\_datatype\_restriction** (*node*, *children*)  
     → *owlapy.class\_expression.OWLDatatypeRestriction*

**visit\_facet\_restrictions** (*node*, *children*)  
     → *List[owlapy.class\_expression.OWLFacetRestriction]*

**visit\_facet\_restriction** (*node*, *children*) → *owlapy.class\_expression.OWLFacetRestriction*

**visit\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

**visit\_typed\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

**abstract\_visit\_string\_literal\_language** (*node*, *children*)

**visit\_string\_literal\_no\_language** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

**visit\_quoted\_string** (*node*, *children*) → *str*

**visit\_float\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

**visit\_decimal\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

**visit\_integer\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

**visit\_boolean\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

**visit\_datetime\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

**visit\_duration\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

```

visit_date_literal (node, children) → owlapy.owl_literal.OWLLiteral
visit_non_negative_integer (node, children) → int
visit_datatype_iri (node, children) → str
visit_datatype (node, children) → owlapy.owl_datatype.OWLDatatype
visit_facet (node, children) → owlapy.vocab.OWLFacet
visit_class_iri (node, children) → owlapy.class_expression.OWLClass
visit_individual_iri (node, children) → owlapy.owl_individual.OWLNamedIndividual
visit_object_property_iri (node, children) → owlapy.owl_property.OWLObjectProperty
visit_data_property_iri (node, children) → owlapy.owl_property.OWLDataProperty
visit_iri (node, children) → owlapy.iri.IRI
visit_full_iri (node, children) → owlapy.iri.IRI
abstract visit_abbreviated_iri (node, children)
visit_simple_iri (node, children) → owlapy.iri.IRI
visit_parentheses (node, children) → owlapy.class_expression.OWLClassExpression
generic_visit (node, children)

```

Default visitor method

#### Parameters

- **node** – The node we’re visiting
- **visited\_children** – The results of visiting the children of that node, in a list

I’m not sure there’s an implementation of this that makes sense across all (or even most) use cases, so we leave it to subclasses to implement for now.

```
owlapy.parser.DL_GRAMMAR
```

```

class owlapy.parser.DLSyntaxParser (
    namespace: str | owlapy.namespaces.Namespaces | None = None, grammar=None)
Bases: parsimonious.nodes.NodeVisitor, owlapy.owl_object.OWLObjectParser
Description Logic Syntax parser to parse strings to OWLClassExpressions.
slots = ('ns', 'grammar')
ns: str | owlapy.namespaces.Namespaces | None
parse_expression (expression_str: str) → owlapy.class_expression.OWLClassExpression
    Parse a string to an OWL Object.

```

#### Parameters

**expression\_str** (str) – Expression string.

#### Returns

The OWL Object which is represented by the string.

```

visit_union (node, children) → owlapy.class_expression.OWLClassExpression

```

**visit\_intersection** (*node*, *children*) → *owlapy.class\_expression.OWLClassExpression*  
**visit\_primary** (*node*, *children*) → *owlapy.class\_expression.OWLClassExpression*  
**visit\_some\_only\_res** (*node*, *children*) → *owlapy.class\_expression.OWLQuantifiedObjectRestriction*  
**visit\_cardinality\_res** (*node*, *children*)  
     → *owlapy.class\_expression.OWLObjectCardinalityRestriction*  
**visit\_value\_res** (*node*, *children*) → *owlapy.class\_expression.OWLObjectHasValue*  
**visit\_has\_self** (*node*, *children*) → *owlapy.class\_expression.OWLObjectHasSelf*  
**visit\_object\_property** (*node*, *children*) → *owlapy.owl\_property.OWLObjectPropertyExpression*  
**visit\_class\_expression** (*node*, *children*) → *owlapy.class\_expression.OWLClassExpression*  
**visit\_individual\_list** (*node*, *children*) → *owlapy.class\_expression.OWLObjectOneOf*  
**visit\_data\_primary** (*node*, *children*) → *owlapy.owl\_data\_ranges.OWLDataRange*  
**visit\_data\_some\_only\_res** (*node*, *children*)  
     → *owlapy.class\_expression.OWLQuantifiedDataRestriction*  
**visit\_data\_cardinality\_res** (*node*, *children*)  
     → *owlapy.class\_expression.OWLDataCardinalityRestriction*  
**visit\_data\_value\_res** (*node*, *children*) → *owlapy.class\_expression.OWLDataHasValue*  
**visit\_data\_union** (*node*, *children*) → *owlapy.owl\_data\_ranges.OWLDataRange*  
**visit\_data\_intersection** (*node*, *children*) → *owlapy.owl\_data\_ranges.OWLDataRange*  
**visit\_literal\_list** (*node*, *children*) → *owlapy.class\_expression.OWLDataOneOf*  
**visit\_data\_parentheses** (*node*, *children*) → *owlapy.owl\_data\_ranges.OWLDataRange*  
**visit\_datatype\_restriction** (*node*, *children*)  
     → *owlapy.class\_expression.OWLDatatypeRestriction*  
**visit\_facet\_restrictions** (*node*, *children*)  
     → *List[owlapy.class\_expression.OWLFacetRestriction]*  
**visit\_facet\_restriction** (*node*, *children*) → *owlapy.class\_expression.OWLFacetRestriction*  
**visit\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*  
**visit\_typed\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*  
**abstract visit\_string\_literal\_language** (*node*, *children*)  
**visit\_string\_literal\_no\_language** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*  
**visit\_quoted\_string** (*node*, *children*) → *str*  
**visit\_float\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*  
**visit\_decimal\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*  
**visit\_integer\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*

**visit\_boolean\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*  
**visit\_datetime\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*  
**visit\_duration\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*  
**visit\_date\_literal** (*node*, *children*) → *owlapy.owl\_literal.OWLLiteral*  
**visit\_non\_negative\_integer** (*node*, *children*) → int  
**visit\_datatype\_iri** (*node*, *children*) → str  
**visit\_datatype** (*node*, *children*) → *owlapy.owl\_datatype.OWLDatatype*  
**visit\_facet** (*node*, *children*) → *owlapy.vocab.OWLFacet*  
**visit\_class\_iri** (*node*, *children*) → *owlapy.class\_expression.OWLClass*  
**visit\_individual\_iri** (*node*, *children*) → *owlapy.owl\_individual.OWLNamedIndividual*  
**visit\_object\_property\_iri** (*node*, *children*) → *owlapy.owl\_property.OWLObjectProperty*  
**visit\_data\_property\_iri** (*node*, *children*) → *owlapy.owl\_property.OWLDataProperty*  
**visit\_iri** (*node*, *children*) → *owlapy.iri.IRI*  
**visit\_full\_iri** (*node*, *children*) → *owlapy.iri.IRI*  
**abstract visit\_abbreviated\_iri** (*node*, *children*)  
**visit\_simple\_iri** (*node*, *children*) → *owlapy.iri.IRI*  
**visit\_parentheses** (*node*, *children*) → *owlapy.class\_expression.OWLClassExpression*  
**generic\_visit** (*node*, *children*)

Default visitor method

#### Parameters

- **node** – The node we’re visiting
- **visited\_children** – The results of visiting the children of that node, in a list

I’m not sure there’s an implementation of this that makes sense across all (or even most) use cases, so we leave it to subclasses to implement for now.

`owlapy.parser.DLparser`

`owlapy.parser.ManchesterParser`

`owlapy.parser.dl_to_owl_expression` (*dl\_expression*: str, *namespace*: str)

`owlapy.parser.manchester_to_owl_expression` (*manchester\_expression*: str, *namespace*: str)

## owlapy.providers

OWL Datatype restriction constructors.

### Attributes

---

*Restriction\_Literals*

---

### Classes

<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
<i>OWLDatatypeRestriction</i>	A datatype restriction <code>DatatypeRestriction( DT F1 lt1 ... Fn ltn )</code> consists of a unary datatype <code>DT</code> and <code>n</code> pairs
<i>OWLFacet</i>	Enumerations for OWL facets.
<i>OWLFacetRestriction</i>	A facet restriction is used to restrict a particular datatype.

### Functions

<i>owl_datatype_max_exclusive_restriction</i>	Create a max exclusive restriction.
<i>owl_datatype_min_exclusive_restriction</i>	Create a min exclusive restriction.
<i>owl_datatype_max_inclusive_restriction</i>	Create a max inclusive restriction.
<i>owl_datatype_min_inclusive_restriction</i>	Create a min inclusive restriction.
<i>owl_datatype_min_max_exclusive_restric</i>	Create a min-max exclusive restriction.
<i>owl_datatype_min_max_inclusive_restric</i>	Create a min-max inclusive restriction.

### Module Contents

**class** owlapy.providers.OWLLiteral

Bases: *owlapy.owl\_annotation.OWLAnnotationValue*

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(<https://www.w3.org/TR/owl2-syntax/#Literals>)

**\_\_slots\_\_** = ()

**type\_index:** Final = 4008

**get\_literal** () → str

Gets the lexical value of this literal. Note that the language tag is not included.

#### Returns

The lexical value of this literal.

**is\_boolean** () → bool

Whether this literal is typed as boolean.

**parse\_boolean** () → bool

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("<http://www.w3.org/2001/XMLSchema#boolean>").

**Returns**

A bool value that is represented by this literal.

**is\_double** () → bool

Whether this literal is typed as double.

**parse\_double** () → float

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

**Returns**

A double value that is represented by this literal.

**is\_integer** () → bool

Whether this literal is typed as integer.

**parse\_integer** () → int

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

**Returns**

An integer value that is represented by this literal.

**is\_string** () → bool

Whether this literal is typed as string.

**parse\_string** () → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("<http://www.w3.org/2001/XMLSchema#string>").

**Returns**

A string value that is represented by this literal.

**is\_date** () → bool

Whether this literal is typed as date.

**parse\_date** () → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("<http://www.w3.org/2001/XMLSchema#date>").

**Returns**

A date value that is represented by this literal.

**is\_datetime** () → bool

Whether this literal is typed as dateTime.

**parse\_datetime** () → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("<http://www.w3.org/2001/XMLSchema#dateTime>").

**Returns**

A datetime value that is represented by this literal.



**is\_duration()** → bool  
Whether this literal is typed as duration.

**parse\_duration()** → pandas.Timedelta  
Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("<http://www.w3.org/2001/XMLSchema#duration>").

**Returns**  
A Timedelta value that is represented by this literal.

**is\_literal()** → bool

**Returns**  
true if the annotation value is a literal

**as\_literal()** → *OWLLiteral*

**Returns**  
if the value is a literal, returns it. Return None otherwise

**to\_python()** → Literals

**abstract get\_datatype()** → *owlapy.owl\_datatype.OWLDatatype*  
Gets the OWLDatatype which types this literal.

**Returns**  
The OWLDatatype that types this literal.

**class** owlapy.providers.**OWLDatatypeRestriction**(type\_: *owlapy.owl\_datatype.OWLDatatype*,  
facet\_restrictions: *OWLFacetRestriction* | *Iterable[OWLFacetRestriction]*)  
Bases: *owlapy.owl\_data\_ranges.OWLDataRange*

A datatype restriction DatatypeRestriction( DT F1 lt1 ... Fn ltn ) consists of a unary datatype DT and n pairs ( Fi , lti ). The resulting data range is unary and is obtained by restricting the value space of DT according to the semantics of all ( Fi , vi ) (multiple pairs are interpreted conjunctively), where vi are the data values of the literals lti. ([https://www.w3.org/TR/owl2-syntax/#Datatype\\_Restrictions](https://www.w3.org/TR/owl2-syntax/#Datatype_Restrictions))

**\_\_slots\_\_** = ('\_type', '\_facet\_restrictions')

**type\_index:** Final = 4006

**get\_datatype()** → *owlapy.owl\_datatype.OWLDatatype*

**get\_facet\_restrictions()** → Sequence[*OWLFacetRestriction*]

**\_\_eq\_\_**(other)  
Return self==value.

**\_\_hash\_\_**()  
Return hash(self).

**\_\_repr\_\_**()  
Return repr(self).

**class** owlapy.providers.**OWLFacet**(remainder: str, symbolic\_form: str,  
operator: Callable[[X, X], bool])  
Bases: *\_Vocabulary, enum.Enum*

Enumerations for OWL facets.

```

property symbolic_form
property operator

static from_str(name: str) → OWLFacet

MIN_INCLUSIVE: Final
MIN_EXCLUSIVE: Final
MAX_INCLUSIVE: Final
MAX_EXCLUSIVE: Final

LENGTH: Final
MIN_LENGTH: Final
MAX_LENGTH: Final
PATTERN: Final
TOTAL_DIGITS: Final
FRACTION_DIGITS: Final

class owlapy.providers.OWLFacetRestriction(facet: owlapy.vocab.OWLFacet,
    literal: Literals)
    Bases: owlapy.owl_object.OWLObject
    A facet restriction is used to restrict a particular datatype.
    __slots__ = ('_facet', '_literal')
    type_index: Final = 4007
    get_facet() → owlapy.vocab.OWLFacet
    get_facet_value() → owlapy.owl_literal.OWLLiteral
    __eq__(other)
        Return self==value.
    __hash__()
        Return hash(self).
    __repr__()
        Return repr(self).

owlapy.providers.Restriction_Literals

owlapy.providers.owl_datatype_max_exclusive_restriction(max_: Restriction_Literals)
    → owlapy.class_expression.OWLDatatypeRestriction
    Create a max exclusive restriction.

owlapy.providers.owl_datatype_min_exclusive_restriction(min_: Restriction_Literals)
    → owlapy.class_expression.OWLDatatypeRestriction
    Create a min exclusive restriction.

```

```
owlapy.providers.owl_datatype_max_inclusive_restriction(max_: Restriction_Literals)
    → owlapy.class_expression.OWLDatatypeRestriction
```

Create a max inclusive restriction.

```
owlapy.providers.owl_datatype_min_inclusive_restriction(min_: Restriction_Literals)
    → owlapy.class_expression.OWLDatatypeRestriction
```

Create a min inclusive restriction.

```
owlapy.providers.owl_datatype_min_max_exclusive_restriction(
    min_: Restriction_Literals, max_: Restriction_Literals)
    → owlapy.class_expression.OWLDatatypeRestriction
```

Create a min-max exclusive restriction.

```
owlapy.providers.owl_datatype_min_max_inclusive_restriction(
    min_: Restriction_Literals, max_: Restriction_Literals)
    → owlapy.class_expression.OWLDatatypeRestriction
```

Create a min-max inclusive restriction.

## owlapy.render

Renderers for different syntax.

## Attributes

*DLrenderer*

*ManchesterRenderer*

## Classes

<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>OWLNamedIndividual</i>	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
<i>OWLObjectRenderer</i>	Abstract class with a render method to render an OWL Object into a string.
<i>OWLEntity</i>	Represents Entities in the OWL 2 Specification.
<i>OWLObject</i>	Base interface for OWL objects
<i>OWLObjectInverseOf</i>	Represents the inverse of a property expression (Object-InverseOf). An inverse object property expression
<i>OWLPropertyExpression</i>	Represents a property or possibly the inverse of a property.
<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLBooleanClassExpression</i>	Represent an anonymous boolean class expression.

continues on next page

Table 10 – continued from previous page

<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>OWLObjectSomeValuesFrom</i>	An existential class expression <i>ObjectSomeValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and
<i>OWLObjectAllValuesFrom</i>	A universal class expression <i>ObjectAllValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and a
<i>OWLObjectUnionOf</i>	A union class expression <i>ObjectUnionOf</i> ( CE1 ... CEn ) contains all individuals that are instances
<i>OWLObjectIntersectionOf</i>	An intersection class expression <i>ObjectIntersectionOf</i> ( CE1 ... CEn ) contains all individuals that are instances
<i>OWLObjectComplementOf</i>	Represents an <i>ObjectComplementOf</i> class expression in the OWL 2 Specification.
<i>OWLObjectMinCardinality</i>	A minimum cardinality expression <i>ObjectMinCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLObjectExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLObjectMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLObjectHasSelf</i>	A self-restriction <i>ObjectHasSelf</i> ( OPE ) consists of an object property expression OPE,
<i>OWLDataSomeValuesFrom</i>	An existential class expression <i>DataSomeValuesFrom</i> ( DPE1 ... DPEn DR ) consists of n data property expressions
<i>OWLDataAllValuesFrom</i>	A universal class expression <i>DataAllValuesFrom</i> ( DPE1 ... DPEn DR ) consists of n data property expressions DPEi,
<i>OWLDataHasValue</i>	A has-value class expression <i>DataHasValue</i> ( DPE It ) consists of a data property expression DPE and a literal It,
<i>OWLDataMinCardinality</i>	A minimum cardinality expression <i>DataMinCardinality</i> ( n DPE DR ) consists of a nonnegative integer n, a data
<i>OWLDataExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an
<i>OWLDataMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</i> ( n OPE CE ) consists of a nonnegative integer n, an object
<i>OWLDataOneOf</i>	An enumeration of literals <i>DataOneOf</i> ( It1 ... Itn ) contains exactly the explicitly specified literals Iti with
<i>OWLNaryBooleanClassExpression</i>	<i>OWLNaryBooleanClassExpression</i> .
<i>OWLRestriction</i>	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
<i>OWLFacet</i>	Enumerations for OWL facets.
<i>OWLNaryDataRange</i>	<i>OWLNaryDataRange</i> .
<i>OWLDataComplementOf</i>	A complement data range <i>DataComplementOf</i> ( DR ) contains all tuples of literals that are not contained in the
<i>OWLDataUnionOf</i>	A union data range <i>DataUnionOf</i> ( DR1 ... DRn ) contains all tuples of literals that are contained in the at least
<i>OWLDataIntersectionOf</i>	An intersection data range <i>DataIntersectionOf</i> ( DR1 ... DRn ) contains all tuples of literals that are contained

continues on next page

Table 10 – continued from previous page

<i>OWLObjectHasValue</i>	A has-value class expression <code>ObjectHasValue( OPE a )</code> consists of an object property expression OPE and an
<i>OWLFacetRestriction</i>	A facet restriction is used to restrict a particular datatype.
<i>OWLDatatypeRestriction</i>	A datatype restriction <code>DatatypeRestriction( DT F1 lt1 ... Fn ltn )</code> consists of a unary datatype DT and n pairs
<i>OWLObjectOneOf</i>	An enumeration of individuals <code>ObjectOneOf( a1 ... an )</code> contains exactly the individuals $a_i$ with $1 \leq i \leq n$ .
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
<i>DLSyntaxObjectRenderer</i>	DL Syntax renderer for OWL Objects.
<i>ManchesterOWLSyntaxOWLObjectRenderer</i>	Manchester Syntax renderer for OWL Objects

## Functions

---

```
owl_expression_to_dl( $\rightarrow$  str)

owl_expression_to_manchester( $\rightarrow$  str)
```

---

## Module Contents

```
class owlapy.render.IRI (namespace: str | owlapy.namespaces.Namespaces, remainder: str)
    Bases: owlapy.owl_annotation.OWLAnnotationSubject, owlapy.owl_annotation.
            OWLAnnotationValue
    An IRI, consisting of a namespace and a remainder.
    __slots__ = ('_namespace', '_remainder', '__weakref__')
    type_index: Final = 0
    static create (namespace: owlapy.namespaces.Namespaces, remainder: str)  $\rightarrow$  IRI
    static create (namespace: str, remainder: str)  $\rightarrow$  IRI
    static create (string: str)  $\rightarrow$  IRI
    __repr__ ()
        Return repr(self).
    __eq__ (other)
        Return self==value.
    __hash__ ()
        Return hash(self).
    is_nothing ()
        Determines if this IRI is equal to the IRI that owl:Nothing is named with.
    Returns
        True if this IRI is equal to <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.
```

**is\_thing()**

Determines if this IRI is equal to the IRI that owl:Thing is named with.

**Returns**

True if this IRI is equal to <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

**is\_reserved\_vocabulary()** → bool

Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with [<http://www.w3.org/1999/02/22-rdf-syntax-ns#>](http://www.w3.org/1999/02/22-rdf-syntax-ns#) or [<http://www.w3.org/2000/01/rdf-schema#>](http://www.w3.org/2000/01/rdf-schema#) or [<http://www.w3.org/2001/XMLSchema#>](http://www.w3.org/2001/XMLSchema#) or [<http://www.w3.org/2002/07/owl#>](http://www.w3.org/2002/07/owl#).

**Returns**

True if the IRI is in the reserved vocabulary, otherwise False.

**as\_iri()** → *IRI*

**Returns**

if the value is an IRI, return it. Return None otherwise.

**as\_str()** → str

CD: Should be deprecated. :returns: The string that specifies the IRI.

**property str: str**

Returns: The string that specifies the IRI.

**property reminder: str**

Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form()** → str

Gets the short form.

**Returns**

A string that represents the short form.

**get\_namespace()** → str

**Returns**

The namespace as string.

**get\_remainder()** → str

**Returns**

The remainder (coincident with NCName usually) for this IRI.

**class** owlapy.render.OWLNamedIndividual (*iri: owlapy.iri.IRI | str*)

Bases: OWLIndividual, *owlapy.owl\_object.OWLEntity*

Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities. IRIs from the reserved vocabulary must not be used to identify named individuals in an OWL 2 DL ontology.

([https://www.w3.org/TR/owl2-syntax/#Named\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Named_Individuals))

**\_\_slots\_\_** = '\_iri'

**type\_index:** Final = 1005

**property iri:** *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str**

**Gets the string representation of this object**

**Returns**

The IRI as string

**class** owlapy.render.OWLLiteral

Bases: *owlapy.owl\_annotation.OWLAnnotationValue*

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(<https://www.w3.org/TR/owl2-syntax/#Literals>)

**\_\_slots\_\_** = ()

**type\_index: Final = 4008**

**get\_literal()** → str

Gets the lexical value of this literal. Note that the language tag is not included.

**Returns**

The lexical value of this literal.

**is\_boolean()** → bool

Whether this literal is typed as boolean.

**parse\_boolean()** → bool

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("<http://www.w3.org/2001/XMLSchema#boolean>").

**Returns**

A bool value that is represented by this literal.

**is\_double()** → bool

Whether this literal is typed as double.

**parse\_double()** → float

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

**Returns**

A double value that is represented by this literal.

**is\_integer()** → bool

Whether this literal is typed as integer.

**parse\_integer()** → int

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

**Returns**

An integer value that is represented by this literal.

**is\_string()** → bool

Whether this literal is typed as string.

**parse\_string()** → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("http://www.w3.org/2001/XMLSchema#string").

**Returns**

A string value that is represented by this literal.

**is\_date()** → bool

Whether this literal is typed as date.

**parse\_date()** → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("http://www.w3.org/2001/XMLSchema#date").

**Returns**

A date value that is represented by this literal.

**is\_datetime()** → bool

Whether this literal is typed as dateTime.

**parse\_datetime()** → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("http://www.w3.org/2001/XMLSchema#dateTime").

**Returns**

A datetime value that is represented by this literal.

**is\_duration()** → bool

Whether this literal is typed as duration.

**parse\_duration()** → pandas.Timedelta

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("http://www.w3.org/2001/XMLSchema#duration").

**Returns**

A Timedelta value that is represented by this literal.

**is\_literal()** → bool

**Returns**

true if the annotation value is a literal

**as\_literal()** → *OWLLiteral*

**Returns**

if the value is a literal, returns it. Return None otherwise

**to\_python()** → Literals

**abstract get\_datatype()** → *owlapy.owl\_datatype.OWLDatatype*

Gets the OWLDatatype which types this literal.

**Returns**

The OWLDatatype that types this literal.

**class owlapy.render.OWLObjectRenderer**

Abstract class with a render method to render an OWL Object into a string.



```

abstract set_short_form_provider (short_form_provider) → None
    Configure a short form provider that shortens the OWL objects during rendering.

    Parameters
        short_form_provider – Short form provider.

abstract render (o: OWLObject) → str
    Render OWL Object to string.

    Parameters
        o – OWL Object.

    Returns
        String rendition of OWL object.

class owlapy.render.OWLEntity
    Bases: OWLNamedObject
    Represents Entities in the OWL 2 Specification.

    __slots__ = ()

    to_string_id () → str

    is_anonymous () → bool

class owlapy.render.OWLObject
    Base interface for OWL objects

    __slots__ = ()

    abstract __eq__ (other)
        Return self==value.

    abstract __hash__ ()
        Return hash(self).

    abstract __repr__ ()
        Return repr(self).

    is_anonymous () → bool

class owlapy.render.OWLObjectInverseOf (property: OWLObjectProperty)
    Bases: OWLObjectPropertyExpression

    Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression ObjectInverseOf( P ) connects an individual I1 with I2 if and only if the object property P connects I2 with I1. This can be used to refer to the inverse of a property, without actually naming the property. For example, consider the property hasPart, the inverse property of hasPart (isPartOf) can be referred to using this interface inverseOf(hasPart), which can be used in restrictions e.g. inverseOf(hasPart) some Car refers to the set of things that are part of at least one car.

    (https://www.w3.org/TR/owl2-syntax/#Inverse\_Object\_Properties)

    __slots__ = '_inverse_property'

    type_index: Final = 1003

```

**get\_inverse** () → *OWLObjectProperty*

Gets the property expression that this is the inverse of.

**Returns**

The object property expression such that this object property expression is an inverse of it.

**get\_inverse\_property** () → *OWLObjectProperty*

Obtains the property that corresponds to the inverse of this property.

**Returns**

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**get\_named\_property** () → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

P if this expression is either inv(P) or P.

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**class** owlapy.render.OWLPropertyExpression

Bases: *owlapy.owl\_object.OWLObject*

Represents a property or possibly the inverse of a property.

**\_\_slots\_\_** = ()

**is\_data\_property\_expression** () → bool

**Returns**

True if this is a data property.

**is\_object\_property\_expression** () → bool

**Returns**

True if this is an object property.

**is\_owl\_top\_object\_property** () → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

**is\_owl\_top\_data\_property** () → bool

Determines if this is the owl:topDataProperty.

**Returns**

topDataProperty.

**Return type**

True if this property is the owl

**class** owlapy.render.OWLClassExpression

Bases: *owlapy.owl\_data\_ranges.OWLPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

**\_\_slots\_\_** = ()

**abstract is\_owl\_thing** () → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**abstract is\_owl\_nothing** () → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of** () → *OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**abstract get\_nnf** () → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.render.OWLBooleanClassExpression

Bases: *OWLAnonymousClassExpression*

Represent an anonymous boolean class expression.

**\_\_slots\_\_** = ()

**class** owlapy.render.OWLClass (*iri: owlapy.iri.IRI | str*)

Bases: *owlapy.class\_expression.class\_expression.OWLClassExpression*, *owlapy.owl\_object.OWLEntity*

An OWL 2 named Class. Classes can be understood as sets of individuals. (<https://www.w3.org/TR/owl2-syntax/#Classes>)

**\_\_slots\_\_** = ('\_iri', '\_is\_nothing', '\_is\_thing')

**type\_index**: Final = 1001

**property iri**: *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str**

**Gets the string representation of this object**

**Returns**

The IRI as string

**property reminder: str**

The reminder of the IRI

**is\_owl\_thing () → bool**

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**is\_owl\_nothing () → bool**

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**get\_object\_complement\_of ()**

→ *owlapy.class\_expression.class\_expression.OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf () → *OWLClass***

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

```
class owlapy.render.OWLObjectSomeValuesFrom (  
    property: owlapy.owl_property.OWLObjectPropertyExpression,  
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: *OWLQuantifiedObjectRestriction*

An existential class expression *ObjectSomeValuesFrom*( OPE CE ) consists of an object property expression OPE and a class expression CE, and it contains all those individuals that are connected by OPE to an individual that is an instance of CE.

**\_\_slots\_\_ = ('\_property', '\_filler')**

**type\_index: Final = 3005**

**\_\_repr\_\_ ()**

Return repr(self).

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

Return self==value.

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

Return hash(self).

**get\_property () → *owlapy.owl\_property.OWLObjectPropertyExpression***

**Returns**

Property being restricted.

```

class owlapy.render.OWLObjectAllValuesFrom (
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
Bases: OWLQuantifiedObjectRestriction

A universal class expression ObjectAllValuesFrom( OPE CE ) consists of an object property expression OPE and
a class expression CE, and it contains all those individuals that are connected by OPE only to individuals that are
instances of CE. (https://www.w3.org/TR/owl2-syntax/#Universal\_Quantification)

__slots__ = ('_property', '_filler')

type_index: Final = 3006

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

get_property () → owlapy.owl_property.OWLObjectPropertyExpression

    Returns
    Property being restricted.

class owlapy.render.OWLObjectUnionOf (
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
Bases: OWLNaryBooleanClassExpression

A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances of at least one
class expression CEi for  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Class\_Expressions)

__slots__ = '_operands'

type_index: Final = 3002

class owlapy.render.OWLObjectIntersectionOf (
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
Bases: OWLNaryBooleanClassExpression

An intersection class expression ObjectIntersectionOf( CE1 ... CEn ) contains all individuals that are instances of
all class expressions CEi for  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Intersection\_of\_Class\_Expressions)

__slots__ = '_operands'

type_index: Final = 3001

class owlapy.render.OWLObjectComplementOf (op: OWLClassExpression)
Bases: OWLBooleanClassExpression, owlapy.meta_classes.HasOperands[OWLClassExpression]

Represents an ObjectComplementOf class expression in the OWL 2 Specification.

__slots__ = '_operand'

type_index: Final = 3003

```

**get\_operand** () → *OWLClassExpression*

**Returns**

The wrapped expression.

**operands** () → *Iterable[OWLClassExpression]*

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

```
class owlapy.render.OwlObjectMinCardinality (cardinality: int,  
        property: owlapy.owl_property.OwlObjectPropertyExpression,  
        filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: OwlObjectCardinalityRestriction

A minimum cardinality expression **ObjectMinCardinality**( n OPE CE ) consists of a nonnegative integer n, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to at least n different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3008
```

```
class owlapy.render.OwlObjectExactCardinality (cardinality: int,  
        property: owlapy.owl_property.OwlObjectPropertyExpression,  
        filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: OwlObjectCardinalityRestriction

An exact cardinality expression **ObjectExactCardinality**( n OPE CE ) consists of a nonnegative integer n, an object

property expression OPE, and a class expression CE, and it contains all those individuals that are connected by to exactly n different individuals that are instances of CE.

([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3009
```

```
as_intersection_of_min_max ()
```

→ *owlapy.class\_expression.nary\_boolean\_expression.OwlObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

**Returns**

The semantically equivalent but structurally simpler form ( $= 1 \text{ R C} \Rightarrow 1 \text{ R C}$  and  $\leq 1 \text{ R C}$ ).

```
class owlapy.render.OwlObjectMaxCardinality (cardinality: int,  
        property: owlapy.owl_property.OwlObjectPropertyExpression,  
        filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: OWLObjectCardinalityRestriction

A maximum cardinality expression `ObjectMaxCardinality( n OPE CE )` consists of a nonnegative integer  $n$ , an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE

to at most  $n$  different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

```
type_index: Final = 3010
```

```
class owlapy.render.OWLObjectHasSelf(  
    property: owlapy.owl_property.OWLObjectPropertyExpression)
```

Bases: OWLObjectRestriction

A self-restriction `ObjectHasSelf( OPE )` consists of an object property expression OPE, and it contains all those individuals that are connected by OPE to themselves. (<https://www.w3.org/TR/owl2-syntax/#Self-Restriction>)

```
__slots__ = '_property'
```

```
type_index: Final = 3011
```

```
get_property() → owlapy.owl_property.OWLObjectPropertyExpression
```

**Returns**

Property being restricted.

```
__eq__(other)
```

Return `self==value`.

```
__hash__()
```

Return `hash(self)`.

```
__repr__()
```

Return `repr(self)`.

```
class owlapy.render.OWLDataSomeValuesFrom(  
    property: owlapy.owl_property.OWLDataPropertyExpression,  
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: OWLQuantifiedDataRestriction

An existential class expression `DataSomeValuesFrom( DPE1 ... DPEn DR )` consists of  $n$  data property expressions DPE <sub>$i$</sub> ,  $1 \leq i \leq n$ , and a data range DR whose arity must be  $n$ . Such a class expression contains all those individuals that are connected by DPE <sub>$i$</sub>  to literals  $lt_i$ ,  $1 \leq i \leq n$ , such that the tuple  $( lt_1, \dots, lt_n )$  is in DR. A class expression of the form `DataSomeValuesFrom( DPE DR )` can be seen as a syntactic shortcut for the class expression `DataMinCardinality( 1 DPE DR )`. ([https://www.w3.org/TR/owl2-syntax/#Existential\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Existential_Quantification_2))

```
__slots__ = '_property'
```

```
type_index: Final = 3012
```

```
__repr__()
```

Return `repr(self)`.

```
__eq__(other)
```

Return `self==value`.

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

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.render.OWLDataAllValuesFrom (
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: OWLQuantifiedDataRestriction

A universal class expression DataAllValuesFrom( DPE<sub>1</sub> ... DPE<sub>n</sub> DR ) consists of n data property expressions DPE<sub>i</sub>, 1 ≤ i ≤ n, and a data range DR whose arity must be n. Such a class expression contains all those individuals that

**are connected by DPE<sub>i</sub> only to literals lti, 1 ≤ i ≤ n, such that each tuple ( lt<sub>1</sub> , ..., lt<sub>n</sub> ) is in DR.**

**A class**

expression of the form DataAllValuesFrom( DPE DR ) can be seen as a syntactic shortcut for the class expression DataMaxCardinality( 0 DPE DataComplementOf( DR ) ). ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification_2))

**\_\_slots\_\_** = **'\_property'**

**type\_index:** **Final** = 3013

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.render.OWLDataHasValue (
    property: owlapy.owl_property.OWLDataPropertyExpression,
    value: owlapy.owl_literal.OWLLiteral)
```

Bases: OWLHasValueRestriction[*owlapy.owl\_literal.OWLLiteral*], OWLDataRestriction

A has-value class expression DataHasValue( DPE lt ) consists of a data property expression DPE and a literal lt, and it contains all those individuals that are connected by DPE to lt. Each such class expression can be seen as a syntactic shortcut for the class expression DataSomeValuesFrom( DPE DataOneOf( lt ) ). ([https://www.w3.org/TR/owl2-syntax/#Literal\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Literal_Value_Restriction))

**\_\_slots\_\_** = **'\_property'**

**type\_index:** **Final** = 3014

**\_\_repr\_\_** ()

Return repr(self).



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

Return self==value.

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

Return hash(self).

**as\_some\_values\_from** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

**Returns**

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.render.OWLDataMinCardinality (cardinality: int,  
      property: owlapy.owl_property.OWLDataPropertyExpression,  
      filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: OWLDataCardinalityRestriction

A minimum cardinality expression DataMinCardinality( *n* DPE DR ) consists of a nonnegative integer *n*, a data property expression DPE, and a unary data range DR, and it contains all those individuals that are connected by DPE to at least *n* different literals in DR. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

**\_\_slots\_\_** = ('\_cardinality', '\_filler', '\_property')

**type\_index:** Final = 3015

```
class owlapy.render.OWLDataExactCardinality (cardinality: int,  
      property: owlapy.owl_property.OWLDataPropertyExpression,  
      filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: OWLDataCardinalityRestriction

An exact cardinality expression ObjectExactCardinality( *n* OPE CE ) consists of a nonnegative integer *n*, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected

by OPE to exactly *n* different individuals that are instances of CE ([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

**\_\_slots\_\_** = ('\_cardinality', '\_filler', '\_property')

**type\_index:** Final = 3016

```
as_intersection_of_min_max ()  
      → owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf
```

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

**Returns**

The semantically equivalent but structurally simpler form  $(= 1\ R\ D) = \geq 1\ R\ D\ \text{and}\ \leq 1\ R\ D$ .

```
class owlapy.render.OWLDataMaxCardinality (cardinality: int,  
      property: owlapy.owl_property.OWLDataPropertyExpression,  
      filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: OWLDataCardinalityRestriction

A maximum cardinality expression ObjectMaxCardinality( *n* OPE CE ) consists of a nonnegative integer *n*, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to at most *n* different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3017

class owlapy.render.OWLDataOneOf (
    values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])
    Bases: owlapy.owl_data_ranges.OWLDataRange, owlapy.meta_classes.
    HasOperands[owlapy.owl_literal.OWLLiteral]

    An enumeration of literals DataOneOf( lt1 ... ltn ) contains exactly the explicitly specified literals lti with  $1 \leq i \leq n$ . The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration\_of\_Literals)

    type_index: Final = 4003

    values () → Iterable[owlapy.owl_literal.OWLLiteral]
        Gets the values that are in the oneOf.

        Returns
            The values of this { @code DataOneOf } class expression.

    operands () → Iterable[owlapy.owl_literal.OWLLiteral]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
            The operands.

    __hash__ ()
        Return hash(self).

    __eq__ (other)
        Return self==value.

    __repr__ ()
        Return repr(self).

class owlapy.render.OWLNaryBooleanClassExpression (
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
    Bases: owlapy.class_expression.class_expression.OWLBooleanClassExpression,
    owlapy.meta_classes.HasOperands[owlapy.class_expression.class_expression.
    OWLClassExpression]

    OWLNaryBooleanClassExpression.

    __slots__ = ()

    operands () → Iterable[owlapy.class_expression.class_expression.OWLClassExpression]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
            The operands.

    __repr__ ()
        Return repr(self).

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

```

```

class owlapy.render.OWLRestriction
    Bases: owlapy.class_expression.class_expression.OWLAnonymousClassExpression
    Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
    __slots__ = ()
    abstract get_property() → owlapy.owl_property.OWLPropertyExpression

        Returns
            Property being restricted.

    is_data_restriction() → bool
        Determines if this is a data restriction.

        Returns
            True if this is a data restriction.

    is_object_restriction() → bool
        Determines if this is an object restriction.

        Returns
            True if this is an object restriction.

class owlapy.render.OWLFacet (remainder: str, symbolic_form: str,
                                operator: Callable[[_X, _X], bool])
    Bases: _Vocabulary, enum.Enum
    Enumerations for OWL facets.

    property symbolic_form

    property operator

    static from_str(name: str) → OWLFacet

    MIN_INCLUSIVE: Final

    MIN_EXCLUSIVE: Final

    MAX_INCLUSIVE: Final

    MAX_EXCLUSIVE: Final

    LENGTH: Final

    MIN_LENGTH: Final

    MAX_LENGTH: Final

    PATTERN: Final

    TOTAL_DIGITS: Final

    FRACTION_DIGITS: Final

class owlapy.render.OWLNaryDataRange (operands: Iterable[OWLDataRange])
    Bases: OWLDataRange, owlapy.meta_classes.HasOperands[OWLDataRange]
    OWLNaryDataRange.

```

**\_\_slots\_\_** = ()

**operands** () → Iterable[*OWLDataRange*]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**class** owlapy.render.**OWLDataComplementOf** (data\_range: *OWLDataRange*)

Bases: *OWLDataRange*

A complement data range *DataComplementOf*( DR ) contains all tuples of literals that are not contained in the data range DR. The resulting data range has the arity equal to the arity of DR.

([https://www.w3.org/TR/owl2-syntax/#Complement\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Complement_of_Data_Ranges))

**type\_index**: **Final** = 4002

**get\_data\_range** () → *OWLDataRange*

**Returns**

The wrapped data range.

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**class** owlapy.render.**OWLDataUnionOf** (operands: Iterable[*OWLDataRange*])

Bases: *OWLNaryDataRange*

A union data range *DataUnionOf*( DR1 ... DRn ) contains all tuples of literals that are contained in the at least one data range DRi for  $1 \leq i \leq n$ . All data ranges DRi must be of the same arity, and the resulting data range is of that arity as well.

([https://www.w3.org/TR/owl2-syntax/#Union\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Union_of_Data_Ranges))

**\_\_slots\_\_** = '\_operands'

**type\_index**: **Final** = 4005

**class** owlapy.render.**OWLDataIntersectionOf** (operands: Iterable[*OWLDataRange*])

Bases: *OWLNaryDataRange*

An intersection data range *DataIntersectionOf*( DR1 ... DRn ) contains all tuples of literals that are contained in each data range DRi for  $1 \leq i \leq n$ . All data ranges DRi must be of the same arity, and the resulting data range is of that arity as well.

([https://www.w3.org/TR/owl2-syntax/#Intersection\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Intersection_of_Data_Ranges))

```

__slots__ = '_operands'

type_index: Final = 4004

class owlapy.render.OWLObjectHasValue(
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    individual: owlapy.owl_individual.OWLIndividual)
    Bases: OWLHasValueRestriction[owlapy.owl_individual.OWLIndividual], OWLObjectRestriction

    A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ). (https://www.w3.org/TR/owl2-syntax/#Individual\_Value\_Restriction)

    __slots__ = ('_property', '_v')

    type_index: Final = 3007

    get_property() → owlapy.owl_property.OWLObjectPropertyExpression

        Returns
            Property being restricted.

    as_some_values_from() → owlapy.class_expression.class_expression.OWLClassExpression
        A convenience method that obtains this restriction as an existential restriction with a nominal filler.

        Returns
            The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

    __repr__()
        Return repr(self).

class owlapy.render.OWLFacetRestriction(facet: owlapy.vocab.OWLFacet, literal: Literals)
    Bases: owlapy.owl_object.OWLObject

    A facet restriction is used to restrict a particular datatype.

    __slots__ = ('_facet', '_literal')

    type_index: Final = 4007

    get_facet() → owlapy.vocab.OWLFacet

    get_facet_value() → owlapy.owl_literal.OWLLiteral

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

class owlapy.render.OWLDatatypeRestriction(type_: owlapy.owl_datatype.OWLDatatype,
    facet_restrictions: OWLFacetRestriction | Iterable[OWLFacetRestriction])
    Bases: owlapy.owl_data_ranges.OWLDataRange

    A datatype restriction DatatypeRestriction( DT F1 lt1 ... Fn ltn ) consists of a unary datatype DT and n pairs ( Fi , lti ). The resulting data range is unary and is obtained by restricting the value space of DT according to the

```

semantics of all (  $F_i$  ,  $v_i$  ) (multiple pairs are interpreted conjunctively), where  $v_i$  are the data values of the literals  $l_{ti}$ . ([https://www.w3.org/TR/owl2-syntax/#Datatype\\_Restrictions](https://www.w3.org/TR/owl2-syntax/#Datatype_Restrictions))

```
__slots__ = ('_type', '_facet_restrictions')
```

```
type_index: Final = 4006
```

```
get_datatype() → owlapy.owl_datatype.OWLDatatype
```

```
get_facet_restrictions() → Sequence[OWLFacetRestriction]
```

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
__repr__()
```

Return repr(self).

```
class owlapy.render.OWLObjectOneOf(
```

```
    values: owlapy.owl_individual.OWLIndividual | Iterable[owlapy.owl_individual.OWLIndividual])
```

```
Bases: owlapy.class_expression.class_expression.OWLAnonymousClassExpression,  
owlapy.meta_classes.HasOperands[owlapy.owl_individual.OWLIndividual]
```

An enumeration of individuals `ObjectOneOf( a1 ... an )` contains exactly the individuals  $a_i$  with  $1 \leq i \leq n$ . ([https://www.w3.org/TR/owl2-syntax/#Enumeration\\_of\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Enumeration_of_Individuals))

```
__slots__ = '_values'
```

```
type_index: Final = 3004
```

```
individuals() → Iterable[owlapy.owl_individual.OWLIndividual]
```

Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this class expression.

#### Returns

The individuals that are the values of this {`@code` ObjectOneOf} class expression.

```
operands() → Iterable[owlapy.owl_individual.OWLIndividual]
```

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

#### Returns

The operands.

```
as_object_union_of() → owlapy.class_expression.class_expression.OWLClassExpression
```

Simplifies this enumeration to a union of singleton nominals.

#### Returns

This enumeration in a more standard DL form.  $\text{simp}(\{a\}) = \{a\}$   $\text{simp}(\{a_0, \dots, \{a_n\}) = \text{unionOf}(\{a_0\}, \dots, \{a_n\})$

```
__hash__()
```

Return hash(self).

```
__eq__(other)
```

Return self==value.

```
__repr__()
```

Return repr(self).

```
class owlapy.render.OWLDatatype (iri: owlapy.iri.IRI | owlapy.meta_classes.HasIRI)
```

Bases: *owlapy.owl\_object.OWLEntity*, *owlapy.owl\_data\_ranges.OWLDataRange*

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

```
__slots__ = '_iri'
```

```
type_index: Final = 4001
```

```
property iri: owlapy.iri.IRI
```

Gets the IRI of this object.

**Returns**

The IRI of this object.

```
property str: str
```

Gets the string representation of this object

**Returns**

The IRI as string

```
class owlapy.render.DLSyntaxObjectRenderer (
```

```
    short_form_provider: Callable[[owlapy.owl_object.OWLEntity], str] = _simple_short_form_provider)
```

Bases: *owlapy.owl\_object.OWLObjectRenderer*

DL Syntax renderer for OWL Objects.

```
__slots__ = '_sfp'
```

```
set_short_form_provider (short_form_provider: Callable[[owlapy.owl_object.OWLEntity], str])
```

→ None

Configure a short form provider that shortens the OWL objects during rendering.

**Parameters**

**short\_form\_provider** – Short form provider.

```
render (o: owlapy.owl_object.OWLObject) → str
```

Render OWL Object to string.

**Parameters**

**o** – OWL Object.

**Returns**

String rendition of OWL object.

```
class owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer (
```

```
    short_form_provider: Callable[[owlapy.owl_object.OWLEntity], str] = _simple_short_form_provider,
```

```
    no_render_thing=False)
```

Bases: *owlapy.owl\_object.OWLObjectRenderer*

Manchester Syntax renderer for OWL Objects

```
__slots__ = ('_sfp', '_no_render_thing')
```

**set\_short\_form\_provider** (*short\_form\_provider*: Callable[[owlapy.owl\_object.OWLEntity], str])  
 → None

Configure a short form provider that shortens the OWL objects during rendering.

**Parameters**

**short\_form\_provider** – Short form provider.

**render** (*o*: owlapy.owl\_object.OWLObject) → str

Render OWL Object to string.

**Parameters**

**o** – OWL Object.

**Returns**

String rendition of OWL object.

owlapy.render.DLrenderer

owlapy.render.ManchesterRenderer

owlapy.render.owl\_expression\_to\_dl (*o*: owlapy.owl\_object.OWLObject) → str

owlapy.render.owl\_expression\_to\_manchester (*o*: owlapy.owl\_object.OWLObject) → str

## owlapy.static\_funcs

Static functions for general purposes.

## Functions

<code>move(*args)</code>	"Move" an imported class to the current module by setting the classes <code>__module__</code> attribute.
<code>download_external_files(ftp_link)</code>	

## Module Contents

owlapy.static\_funcs.**move** (*\*args*)

“Move” an imported class to the current module by setting the classes `__module__` attribute.

This is useful for documentation purposes to hide internal packages in sphinx.

**Parameters**

**args** – List of classes to move.

owlapy.static\_funcs.**download\_external\_files** (*ftp\_link*: str)



## owlapy.utils

Owlapy utils.

### Attributes

<i>OWLNothing</i>
<i>OWLThing</i>
<i>measurer</i>

### Classes

<i>OWLNamedIndividual</i>	Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities.
<i>HasIRI</i>	Simple class to access the IRI.
<i>HasFiller</i>	An interface to objects that have a filler.
<i>HasCardinality</i>	An interface to objects that have a cardinality.
<i>HasOperands</i>	An interface to objects that have a collection of operands.
<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
<i>OWLObjectInverseOf</i>	Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.
<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.
<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>OWLObjectCardinalityRestriction</i>	Represents Object Property Cardinality Restrictions in the OWL 2 specification.
<i>OWLObjectComplementOf</i>	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
<i>OWLRestriction</i>	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
<i>OWLObjectSomeValuesFrom</i>	An existential class expression ObjectSomeValuesFrom( OPE CE ) consists of an object property expression OPE and
<i>OWLObjectMinCardinality</i>	A minimum cardinality expression ObjectMinCardinality( n OPE CE ) consists of a nonnegative integer n, an object

continues on next page

Table 11 – continued from previous page

<i>OWLObjectMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</i> ( <i>n</i> OPE CE ) consists of a nonnegative integer <i>n</i> , an object
<i>OWLObjectExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> ( <i>n</i> OPE CE ) consists of a nonnegative integer <i>n</i> , an object
<i>OWLObjectHasSelf</i>	A self-restriction <i>ObjectHasSelf</i> ( OPE ) consists of an object property expression OPE,
<i>OWLDataMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</i> ( <i>n</i> OPE CE ) consists of a nonnegative integer <i>n</i> , an object
<i>OWLDataMinCardinality</i>	A minimum cardinality expression <i>DataMinCardinality</i> ( <i>n</i> DPE DR ) consists of a nonnegative integer <i>n</i> , a data
<i>OWLDataExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> ( <i>n</i> OPE CE ) consists of a nonnegative integer <i>n</i> , an
<i>OWLDataHasValue</i>	A has-value class expression <i>DataHasValue</i> ( DPE lt ) consists of a data property expression DPE and a literal lt,
<i>OWLDataAllValuesFrom</i>	A universal class expression <i>DataAllValuesFrom</i> ( DPE1 ... DPE <sub>n</sub> DR ) consists of <i>n</i> data property expressions DPE <sub>i</sub> ,
<i>OWLDataSomeValuesFrom</i>	An existential class expression <i>DataSomeValuesFrom</i> ( DPE1 ... DPE <sub>n</sub> DR ) consists of <i>n</i> data property expressions
<i>OWLObjectAllValuesFrom</i>	A universal class expression <i>ObjectAllValuesFrom</i> ( OPE CE ) consists of an object property expression OPE and a
<i>OWLDataOneOf</i>	An enumeration of literals <i>DataOneOf</i> ( lt1 ... lt <sub>n</sub> ) contains exactly the explicitly specified literals lt <sub>i</sub> with
<i>OWLObjectIntersectionOf</i>	An intersection class expression <i>ObjectIntersectionOf</i> ( CE1 ... CE <sub>n</sub> ) contains all individuals that are instances
<i>OWLDataCardinalityRestriction</i>	Represents Data Property Cardinality Restrictions.
<i>OWLNaryBooleanClassExpression</i>	OWLNaryBooleanClassExpression.
<i>OWLObjectUnionOf</i>	A union class expression <i>ObjectUnionOf</i> ( CE1 ... CE <sub>n</sub> ) contains all individuals that are instances
<i>OWLObjectHasValue</i>	A has-value class expression <i>ObjectHasValue</i> ( OPE a ) consists of an object property expression OPE and an
<i>OWLDatatypeRestriction</i>	A datatype restriction <i>DatatypeRestriction</i> ( DT F1 lt1 ... F <sub>n</sub> lt <sub>n</sub> ) consists of a unary datatype DT and <i>n</i> pairs
<i>OWLFacetRestriction</i>	A facet restriction is used to restrict a particular datatype.
<i>OWLObjectOneOf</i>	An enumeration of individuals <i>ObjectOneOf</i> ( a1 ... a <sub>n</sub> ) contains exactly the individuals a <sub>i</sub> with 1 ≤ <i>i</i> ≤ <i>n</i> .
<i>OWLDataComplementOf</i>	A complement data range <i>DataComplementOf</i> ( DR ) contains all tuples of literals that are not contained in the
<i>OWLDataUnionOf</i>	A union data range <i>DataUnionOf</i> ( DR1 ... DR <sub>n</sub> ) contains all tuples of literals that are contained in the at least
<i>OWLDataIntersectionOf</i>	An intersection data range <i>DataIntersectionOf</i> ( DR1 ... DR <sub>n</sub> ) contains all tuples of literals that are contained
<i>OWLNaryDataRange</i>	OWLNaryDataRange.
<i>OWLDatatype</i>	Represents a <i>DataRange</i> in the OWL 2 Specification.
<i>OWLPropertyRange</i>	OWL Objects that can be the ranges of properties.
<i>OWLObject</i>	Base interface for OWL objects
<i>OWLDatatype</i>	Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,
<i>OWLClassExpressionLengthMetric</i>	Length calculation of OWLClassExpression

continues on next page

Table 11 – continued from previous page

<i>EvaluatedDescriptionSet</i>	Abstract base class for generic types.
<i>ConceptOperandSorter</i>	
<i>OperandSetTransform</i>	
<i>HasIndex</i>	Interface for types with an index; this is used to group objects by type when sorting.
<i>OrderedOWLObject</i>	Holder of OWL Objects that can be used for Python sorted.
<i>NNF</i>	This class contains functions to transform a Class Expression into Negation Normal Form.
<i>TopLevelCNF</i>	This class contains functions to transform a class expression into Top-Level Conjunctive Normal Form.
<i>TopLevelDNF</i>	This class contains functions to transform a class expression into Top-Level Disjunctive Normal Form.
<i>LRUCache</i>	Constants shares by all lru cache instances.

## Functions

<i>get_expression_length</i> (→ int)	
<i>combine_nary_expressions</i> (...)	Shortens an OWLClassExpression or OWLDataRange by combining all nested nary expressions of the same type.
<i>iter_count</i> (→ int)	Count the number of elements in an iterable.
<i>as_index</i> (→ HasIndex)	Cast OWL Object to HasIndex.

## Module Contents

**class** owlapy.utils.OWLNamedIndividual (*iri: owlapy.iri.IRI | str*)

Bases: OWLIndividual, owlapy.owl\_object.OWLEntity

Named individuals are identified using an IRI. Since they are given an IRI, named individuals are entities. IRIs from the reserved vocabulary must not be used to identify named individuals in an OWL 2 DL ontology.

([https://www.w3.org/TR/owl2-syntax/#Named\\_Individuals](https://www.w3.org/TR/owl2-syntax/#Named_Individuals))

**\_\_slots\_\_** = '\_iri'

**type\_index:** Final = 1005

**property iri:** owlapy.iri.IRI

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str**

**Gets the string representation of this object**

**Returns**

The IRI as string

```
class owlapy.utils.HasIRI
```

Simple class to access the IRI.

```
__slots__ = ()
```

```
property iri: IRI
```

**Abstractmethod**

Gets the IRI of this object.

**Returns**

The IRI of this object.

```
property str: str
```

**Abstractmethod**

Gets the string representation of this object

**Returns**

The IRI as string

```
class owlapy.utils.HasFiller
```

Bases: Generic[\_T]

An interface to objects that have a filler.

**Parameters**

**\_T** – Filler type.

```
__slots__ = ()
```

```
abstract get_filler() → _T
```

Gets the filler for this restriction. In the case of an object restriction this will be an individual, in the case of a data restriction this will be a constant (data value). For quantified restriction this will be a class expression or a data range.

**Returns**

the value

```
class owlapy.utils.HasCardinality
```

An interface to objects that have a cardinality.

```
__slots__ = ()
```

```
abstract get_cardinality() → int
```

Gets the cardinality of a restriction.

**Returns**

The cardinality. A non-negative integer.

```
class owlapy.utils.HasOperands
```

Bases: Generic[\_T]

An interface to objects that have a collection of operands.

**Parameters**

**\_T** – Operand type.

```
__slots__ = ()
```

**abstract operands** () → Iterable[\_T]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

**class** owlapy.utils.OWLLiteral

Bases: *owlapy.owl\_annotation.OWLAnnotationValue*

Literals represent data values such as particular strings or integers. They are analogous to typed RDF literals and can also be understood as individuals denoting data values. Each literal consists of a lexical form, which is a string, and a datatype.

(<https://www.w3.org/TR/owl2-syntax/#Literals>)

**\_\_slots\_\_** = ()

**type\_index**: Final = 4008

**get\_literal** () → str

Gets the lexical value of this literal. Note that the language tag is not included.

**Returns**

The lexical value of this literal.

**is\_boolean** () → bool

Whether this literal is typed as boolean.

**parse\_boolean** () → bool

Parses the lexical value of this literal into a bool. The lexical value of this literal should be in the lexical space of the boolean datatype ("<http://www.w3.org/2001/XMLSchema#boolean>").

**Returns**

A bool value that is represented by this literal.

**is\_double** () → bool

Whether this literal is typed as double.

**parse\_double** () → float

Parses the lexical value of this literal into a double. The lexical value of this literal should be in the lexical space of the double datatype ("<http://www.w3.org/2001/XMLSchema#double>").

**Returns**

A double value that is represented by this literal.

**is\_integer** () → bool

Whether this literal is typed as integer.

**parse\_integer** () → int

Parses the lexical value of this literal into an integer. The lexical value of this literal should be in the lexical space of the integer datatype ("<http://www.w3.org/2001/XMLSchema#integer>").

**Returns**

An integer value that is represented by this literal.

**is\_string** () → bool

Whether this literal is typed as string.

**parse\_string()** → str

Parses the lexical value of this literal into a string. The lexical value of this literal should be in the lexical space of the string datatype ("http://www.w3.org/2001/XMLSchema#string").

**Returns**

A string value that is represented by this literal.

**is\_date()** → bool

Whether this literal is typed as date.

**parse\_date()** → datetime.date

Parses the lexical value of this literal into a date. The lexical value of this literal should be in the lexical space of the date datatype ("http://www.w3.org/2001/XMLSchema#date").

**Returns**

A date value that is represented by this literal.

**is\_datetime()** → bool

Whether this literal is typed as dateTime.

**parse\_datetime()** → datetime.datetime

Parses the lexical value of this literal into a datetime. The lexical value of this literal should be in the lexical space of the dateTime datatype ("http://www.w3.org/2001/XMLSchema#dateTime").

**Returns**

A datetime value that is represented by this literal.

**is\_duration()** → bool

Whether this literal is typed as duration.

**parse\_duration()** → pandas.Timedelta

Parses the lexical value of this literal into a Timedelta. The lexical value of this literal should be in the lexical space of the duration datatype ("http://www.w3.org/2001/XMLSchema#duration").

**Returns**

A Timedelta value that is represented by this literal.

**is\_literal()** → bool

**Returns**

true if the annotation value is a literal

**as\_literal()** → *OWLLiteral*

**Returns**

if the value is a literal, returns it. Return None otherwise

**to\_python()** → Literals

**abstract get\_datatype()** → *owlapy.owl\_datatype.OWLDatatype*

Gets the OWLDatatype which types this literal.

**Returns**

The OWLDatatype that types this literal.

**class** owlapy.utils.**OWLObjectInverseOf** (*property: OWLObjectProperty*)

Bases: OWLObjectPropertyExpression

Represents the inverse of a property expression (ObjectInverseOf). An inverse object property expression ObjectInverseOf( P ) connects an individual I1 with I2 if and only if the object property P connects I2 with I1. This can

be used to refer to the inverse of a property, without actually naming the property. For example, consider the property `hasPart`, the inverse property of `hasPart` (`isPartOf`) can be referred to using this interface `inverseOf(hasPart)`, which can be used in restrictions e.g. `inverseOf(hasPart) some Car` refers to the set of things that are part of at least one car.

([https://www.w3.org/TR/owl2-syntax/#Inverse\\_Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Inverse_Object_Properties))

```
__slots__ = '_inverse_property'
```

```
type_index: Final = 1003
```

```
get_inverse () → OWLObjectProperty
```

Gets the property expression that this is the inverse of.

#### Returns

The object property expression such that this object property expression is an inverse of it.

```
get_inverse_property () → OWLObjectProperty
```

Obtains the property that corresponds to the inverse of this property.

#### Returns

The inverse of this property. Note that this property will not necessarily be in the simplest form.

```
get_named_property () → OWLObjectProperty
```

Get the named object property used in this property expression.

#### Returns

P if this expression is either `inv(P)` or `P`.

```
__repr__ ()
```

Return `repr(self)`.

```
__eq__ (other)
```

Return `self==value`.

```
__hash__ ()
```

Return `hash(self)`.

```
class owlapy.utils.OWLObjectProperty (iri: owlapy.iri.IRI | str)
```

Bases: `OWLObjectPropertyExpression`, `OWLProperty`

Represents an Object Property in the OWL 2 Specification. Object properties connect pairs of individuals.

([https://www.w3.org/TR/owl2-syntax/#Object\\_Properties](https://www.w3.org/TR/owl2-syntax/#Object_Properties))

```
__slots__ = '_iri'
```

```
type_index: Final = 1002
```

```
get_named_property () → OWLObjectProperty
```

Get the named object property used in this property expression.

#### Returns

P if this expression is either `inv(P)` or `P`.

```
get_inverse_property () → OWLObjectInverseOf
```

Obtains the property that corresponds to the inverse of this property.

#### Returns

The inverse of this property. Note that this property will not necessarily be in the simplest form.

**is\_owl\_top\_object\_property()** → bool  
Determines if this is the owl:topObjectProperty.

**Returns**  
topObjectProperty.

**Return type**  
True if this property is the owl

**class** owlapy.utils.**OWLDataProperty**(*iri: owlapy.iri.IRI | str*)  
Bases: OWLDataPropertyExpression, OWLProperty

Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals. In some knowledge representation systems, functional data properties are called attributes.

([https://www.w3.org/TR/owl2-syntax/#Data\\_Properties](https://www.w3.org/TR/owl2-syntax/#Data_Properties))

**\_\_slots\_\_** = '\_iri'

**type\_index:** Final = 1004

**is\_owl\_top\_data\_property()** → bool  
Determines if this is the owl:topDataProperty.

**Returns**  
topDataProperty.

**Return type**  
True if this property is the owl

**class** owlapy.utils.**OWLClassExpression**  
Bases: *owlapy.owl\_data\_ranges.OWLPropertyRange*

OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties; individuals satisfying these conditions are said to be instances of the respective class expressions. In the structural specification of OWL 2, class expressions are represented by ClassExpression. ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions))

**\_\_slots\_\_** = ()

**abstract is\_owl\_thing()** → bool  
Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**  
Thing.

**Return type**  
True if this expression is owl

**abstract is\_owl\_nothing()** → bool  
Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**abstract get\_object\_complement\_of()** → *OWLObjectComplementOf*  
Gets the object complement of this class expression.

**Returns**  
A class expression that is the complement of this class expression.



**abstract get\_nnf()** → *OWLClassExpression*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

**class** owlapy.utils.**OWLClass** (*iri: owlapy.iri.IRI | str*)

Bases: *owlapy.class\_expression.class\_expression.OWLClassExpression, owlapy.owl\_object.OWLEntity*

An OWL 2 named Class. Classes can be understood as sets of individuals. (<https://www.w3.org/TR/owl2-syntax/#Classes>)

**\_\_slots\_\_** = ('\_iri', '\_is\_nothing', '\_is\_thing')

**type\_index:** Final = 1001

**property iri:** *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str**

Gets the string representation of this object

**Returns**

The IRI as string

**property reminder:** str

The reminder of the IRI

**is\_owl\_thing()** → bool

Determines if this expression is the built in class owl:Thing. This method does not determine if the class is equivalent to owl:Thing.

**Returns**

Thing.

**Return type**

True if this expression is owl

**is\_owl\_nothing()** → bool

Determines if this expression is the built in class owl:Nothing. This method does not determine if the class is equivalent to owl:Nothing.

**get\_object\_complement\_of()**

→ *owlapy.class\_expression.class\_expression.OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

A class expression that is the complement of this class expression.

**get\_nnf()** → *OWLClass*

Gets the negation normal form of the complement of this expression.

**Returns**

A expression that represents the NNF of the complement of this expression.

```

class owlapy.utils.OWLObjectCardinalityRestriction (cardinality: int,
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    filler: owlapy.class_expression.class_expression.OWLClassExpression)

Bases: OWLCardinalityRestriction[owlapy.class_expression.class_expression.
    OWLClassExpression], OWLQuantifiedObjectRestriction

Represents Object Property Cardinality Restrictions in the OWL 2 specification.

__slots__ = ()

get_property () → owlapy.owl_property.OWLObjectPropertyExpression

    Returns
        Property being restricted.

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

class owlapy.utils.OWLObjectComplementOf (op: OWLClassExpression)

Bases: OWLBooleanClassExpression, owlapy.meta_classes.
    HasOperands[OWLClassExpression]

Represents an ObjectComplementOf class expression in the OWL 2 Specification.

__slots__ = '_operand'

type_index: Final = 3003

get_operand () → OWLClassExpression

    Returns
        The wrapped expression.

operands () → Iterable[OWLClassExpression]
    Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

    Returns
        The operands.

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

owlapy.utils.OWLNothing: Final

class owlapy.utils.OWLRestriction

Bases: owlapy.class_expression.class_expression.OWLAnonymousClassExpression

Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.

```

```
__slots__ = ()
```

```
abstract get_property() → owlapy.owl_property.OWLPropertyExpression
```

**Returns**

Property being restricted.

```
is_data_restriction() → bool
```

Determines if this is a data restriction.

**Returns**

True if this is a data restriction.

```
is_object_restriction() → bool
```

Determines if this is an object restriction.

**Returns**

True if this is an object restriction.

```
owlapy.utils.OWLThing: Final
```

```
class owlapy.utils.OWLObjectSomeValuesFrom(  
    property: owlapy.owl_property.OWLObjectPropertyExpression,  
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: *OWLQuantifiedObjectRestriction*

An existential class expression *ObjectSomeValuesFrom*( *OPE CE* ) consists of an object property expression *OPE* and a class expression *CE*, and it contains all those individuals that are connected by *OPE* to an individual that is an instance of *CE*.

```
__slots__ = ('_property', '_filler')
```

```
type_index: Final = 3005
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

```
get_property() → owlapy.owl_property.OWLObjectPropertyExpression
```

**Returns**

Property being restricted.

```
class owlapy.utils.OWLObjectMinCardinality(cardinality: int,  
    property: owlapy.owl_property.OWLObjectPropertyExpression,  
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: *OWLObjectCardinalityRestriction*

A minimum cardinality expression *ObjectMinCardinality*( *n OPE CE* ) consists of a nonnegative integer *n*, an object property expression *OPE*, and a class expression *CE*, and it contains all those individuals that are connected by *OPE* to at least *n* different individuals that are instances of *CE*. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

**type\_index: Final = 3008**

```
class owlapy.utils.OwlObjectMaxCardinality (cardinality: int,  
      property: owlapy.owl_property.OwlObjectPropertyExpression,  
      filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: *OwlObjectCardinalityRestriction*

A maximum cardinality expression *ObjectMaxCardinality*( *n* *OPE* *CE* ) consists of a nonnegative integer *n*, an object property expression *OPE*, and a class expression *CE*, and it contains all those individuals that are connected by *OPE*

to at most *n* different individuals that are instances of *CE*. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

**type\_index: Final = 3010**

```
class owlapy.utils.OwlObjectExactCardinality (cardinality: int,  
      property: owlapy.owl_property.OwlObjectPropertyExpression,  
      filler: owlapy.class_expression.class_expression.OwlClassExpression)
```

Bases: *OwlObjectCardinalityRestriction*

An exact cardinality expression *ObjectExactCardinality*( *n* *OPE* *CE* ) consists of a nonnegative integer *n*, an object

property expression *OPE*, and a class expression *CE*, and it contains all those individuals that are connected by to exactly *n* different individuals that are instances of *CE*.

([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```
__slots__ = ('_cardinality', '_filler', '_property')
```

**type\_index: Final = 3009**

```
as_intersection_of_min_max()  
    → owlapy.class_expression.nary_boolean_expression.OwlObjectIntersectionOf
```

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

#### Returns

The semantically equivalent but structurally simpler form  $(= 1 \text{ R } C) = \geq 1 \text{ R } C \text{ and } \leq 1 \text{ R } C$ .

```
class owlapy.utils.OwlObjectHasSelf (  
      property: owlapy.owl_property.OwlObjectPropertyExpression)
```

Bases: *OwlObjectRestriction*

A self-restriction *ObjectHasSelf*( *OPE* ) consists of an object property expression *OPE*, and it contains all those individuals that are connected by *OPE* to themselves. (<https://www.w3.org/TR/owl2-syntax/#Self-Restriction>)

```
__slots__ = '_property'
```

**type\_index: Final = 3011**

```
get_property() → owlapy.owl_property.OwlObjectPropertyExpression
```

#### Returns

Property being restricted.

```
__eq__ (other)
```

Return self==value.

```

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

```

**class** owlapy.utils.**OWLDataMaxCardinality** (*cardinality: int*,  
*property: owlapy.owl\_property.OWLDataPropertyExpression*,  
*filler: owlapy.owl\_data\_ranges.OWLDataRange*)

Bases: *OWLDataCardinalityRestriction*

A maximum cardinality expression *ObjectMaxCardinality*( *n* OPE CE ) consists of a nonnegative integer *n*, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to at most *n* different individuals that are instances of CE. ([https://www.w3.org/TR/owl2-syntax/#Maximum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Maximum_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3017

```

**class** owlapy.utils.**OWLDataMinCardinality** (*cardinality: int*,  
*property: owlapy.owl\_property.OWLDataPropertyExpression*,  
*filler: owlapy.owl\_data\_ranges.OWLDataRange*)

Bases: *OWLDataCardinalityRestriction*

A minimum cardinality expression *DataMinCardinality*( *n* DPE DR ) consists of a nonnegative integer *n*, a data property expression DPE, and a unary data range DR, and it contains all those individuals that are connected by DPE to at least *n* different literals in DR. ([https://www.w3.org/TR/owl2-syntax/#Minimum\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3015

```

**class** owlapy.utils.**OWLDataExactCardinality** (*cardinality: int*,  
*property: owlapy.owl\_property.OWLDataPropertyExpression*,  
*filler: owlapy.owl\_data\_ranges.OWLDataRange*)

Bases: *OWLDataCardinalityRestriction*

An exact cardinality expression *ObjectExactCardinality*( *n* OPE CE ) consists of a nonnegative integer *n*, an object property expression OPE, and a class expression CE, and it contains all those individuals that are connected by OPE to exactly *n* different individuals that are instances of CE ([https://www.w3.org/TR/owl2-syntax/#Exact\\_Cardinality](https://www.w3.org/TR/owl2-syntax/#Exact_Cardinality))

```

__slots__ = ('_cardinality', '_filler', '_property')

type_index: Final = 3016

```

**as\_intersection\_of\_min\_max**()

→ *owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf*

Obtains an equivalent form that is a conjunction of a min cardinality and max cardinality restriction.

**Returns**

The semantically equivalent but structurally simpler form ( $= 1 \text{ R D} = \geq 1 \text{ R D}$  and  $\leq 1 \text{ R D}$ ).

**class** owlapy.utils.**OWLDataHasValue** (  
*property: owlapy.owl\_property.OWLDataPropertyExpression*,  
*value: owlapy.owl\_literal.OWLLiteral*)

Bases: `OWLHasValueRestriction[owlapy.owl_literal.OWLLiteral]`, `OWLDataRestriction`

A has-value class expression `DataHasValue( DPE lt )` consists of a data property expression DPE and a literal lt, and it contains all those individuals that are connected by DPE to lt. Each such class expression can be seen as a syntactic shortcut for the class expression `DataSomeValuesFrom( DPE DataOneOf( lt ) )`. ([https://www.w3.org/TR/owl2-syntax/#Literal\\_Value\\_Restriction](https://www.w3.org/TR/owl2-syntax/#Literal_Value_Restriction))

`__slots__ = '_property'`

`type_index: Final = 3014`

`__repr__()`

Return repr(self).

`__eq__(other)`

Return self==value.

`__hash__()`

Return hash(self).

`as_some_values_from()` → `owlapy.class_expression.class_expression.OWLClassExpression`

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

#### Returns

The existential equivalent of this value restriction. `simp(HasValue(p a)) = some(p {a})`.

`get_property()` → `owlapy.owl_property.OWLDataPropertyExpression`

#### Returns

Property being restricted.

`class owlapy.utils.OWLDataAllValuesFrom(`  
     `property: owlapy.owl_property.OWLDataPropertyExpression,`  
     `filler: owlapy.owl_data_ranges.OWLDataRange)`

Bases: `OWLQuantifiedDataRestriction`

A universal class expression `DataAllValuesFrom( DPE1 ... DPEn DR )` consists of n data property expressions DPE<sub>i</sub>, 1 ≤ i ≤ n, and a data range DR whose arity must be n. Such a class expression contains all those individuals that

**are connected by DPE<sub>i</sub> only to literals lti, 1 ≤ i ≤ n, such that each tuple ( lt1 , ..., ltn ) is in DR.**

#### A class

expression of the form `DataAllValuesFrom( DPE DR )` can be seen as a syntactic shortcut for the class expression `DataMaxCardinality( 0 DPE DataComplementOf( DR ) )`. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification_2))

`__slots__ = '_property'`

`type_index: Final = 3013`

`__repr__()`

Return repr(self).

`__eq__(other)`

Return self==value.

`__hash__()`

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.utils.OWLDataSomeValuesFrom(  
    property: owlapy.owl_property.OWLDataPropertyExpression,  
    filler: owlapy.owl_data_ranges.OWLDataRange)
```

Bases: *OWLQuantifiedDataRestriction*

An existential class expression *DataSomeValuesFrom*( *DPE*<sub>1</sub> ... *DPE*<sub>*n*</sub> *DR* ) consists of *n* data property expressions *DPE*<sub>*i*</sub>,  $1 \leq i \leq n$ , and a data range *DR* whose arity must be *n*. Such a class expression contains all those individuals that are connected by *DPE*<sub>*i*</sub> to literals *lti*,  $1 \leq i \leq n$ , such that the tuple ( *lt*<sub>1</sub> , ..., *lt*<sub>*n*</sub> ) is in *DR*. A class expression of the form *DataSomeValuesFrom*( *DPE* *DR* ) can be seen as a syntactic shortcut for the class expression *DataMinCardinality*( 1 *DPE* *DR* ). ([https://www.w3.org/TR/owl2-syntax/#Existential\\_Quantification\\_2](https://www.w3.org/TR/owl2-syntax/#Existential_Quantification_2))

```
__slots__ = '_property'
```

```
type_index: Final = 3012
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLDataPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.utils.OWLObjectAllValuesFrom(  
    property: owlapy.owl_property.OWLObjectPropertyExpression,  
    filler: owlapy.class_expression.class_expression.OWLClassExpression)
```

Bases: *OWLQuantifiedObjectRestriction*

A universal class expression *ObjectAllValuesFrom*( *OPE* *CE* ) consists of an object property expression *OPE* and a class expression *CE*, and it contains all those individuals that are connected by *OPE* only to individuals that are instances of *CE*. ([https://www.w3.org/TR/owl2-syntax/#Universal\\_Quantification](https://www.w3.org/TR/owl2-syntax/#Universal_Quantification))

```
__slots__ = ('_property', '_filler')
```

```
type_index: Final = 3006
```

```
__repr__()
```

Return repr(self).

```
__eq__(other)
```

Return self==value.

```
__hash__()
```

Return hash(self).

**get\_property** () → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

```

class owlapy.utils.OWLDataOneOf (
    values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])
    Bases: owlapy.owl_data_ranges.OWLDataRange, owlapy.meta_classes.
    HasOperands[owlapy.owl_literal.OWLLiteral]

    An enumeration of literals DataOneOf( lt1 ... ltn ) contains exactly the explicitly specified literals lti with  $1 \leq i \leq n$ . The resulting data range has arity one. (https://www.w3.org/TR/owl2-syntax/#Enumeration\_of\_Literals)

    type_index: Final = 4003

    values () → Iterable[owlapy.owl_literal.OWLLiteral]
        Gets the values that are in the oneOf.

        Returns
            The values of this { @code DataOneOf } class expression.

    operands () → Iterable[owlapy.owl_literal.OWLLiteral]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
            The operands.

    __hash__ ()
        Return hash(self).

    __eq__ (other)
        Return self==value.

    __repr__ ()
        Return repr(self).

class owlapy.utils.OWLObjectIntersectionOf (
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
    Bases: OWLNaryBooleanClassExpression

    An intersection class expression ObjectIntersectionOf( CE1 ... CEn ) contains all individuals that are instances of all class expressions CEi for  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Intersection\_of\_Class\_Expressions)

    __slots__ = '_operands'

    type_index: Final = 3001

class owlapy.utils.OWLDataCardinalityRestriction (cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_data_ranges.OWLDataRange)
    Bases: OWLCardinalityRestriction[owlapy.owl_data_ranges.OWLDataRange],
    OWLQuantifiedDataRestriction, OWLDataRestriction

    Represents Data Property Cardinality Restrictions.

    __slots__ = ()

    get_property () → owlapy.owl_property.OWLDataPropertyExpression

        Returns
            Property being restricted.

    __repr__ ()
        Return repr(self).

```



```

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

class owlapy.utils.OWLNaryBooleanClassExpression(
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
    Bases: owlapy.class_expression.class_expression.OWLBooleanClassExpression,
            owlapy.meta_classes.HasOperands[owlapy.class_expression.class_expression.
            OWLClassExpression]
    OWLNaryBooleanClassExpression.

    __slots__ = ()

    operands() → Iterable[owlapy.class_expression.class_expression.OWLClassExpression]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
            The operands.

    __repr__()
        Return repr(self).

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

class owlapy.utils.OWLObjectUnionOf(
    operands: Iterable[owlapy.class_expression.class_expression.OWLClassExpression])
    Bases: OWLNaryBooleanClassExpression
    A union class expression ObjectUnionOf( CE1 ... CEn ) contains all individuals that are instances of at least one
    class expression CEi for 1 ≤ i ≤ n. (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Class\_Expressions)

    __slots__ = '_operands'

    type_index: Final = 3002

class owlapy.utils.OWLObjectHasValue(
    property: owlapy.owl_property.OWLObjectPropertyExpression,
    individual: owlapy.owl_individual.OWLIndividual)
    Bases: OWLHasValueRestriction[owlapy.owl_individual.OWLIndividual], OWLObjectRestriction
    A has-value class expression ObjectHasValue( OPE a ) consists of an object property expression OPE and an
    individual a, and it contains all those individuals that are connected by OPE to a. Each such class expression
    can be seen as a syntactic shortcut for the class expression ObjectSomeValuesFrom( OPE ObjectOneOf( a ) ).
    (https://www.w3.org/TR/owl2-syntax/#Individual\_Value\_Restriction)

    __slots__ = ('_property', '_v')

    type_index: Final = 3007

```

**get\_property** () → *owlapy.owl\_property.OWLObjectPropertyExpression*

**Returns**

Property being restricted.

**as\_some\_values\_from** () → *owlapy.class\_expression.class\_expression.OWLClassExpression*

A convenience method that obtains this restriction as an existential restriction with a nominal filler.

**Returns**

The existential equivalent of this value restriction.  $\text{simp}(\text{HasValue}(p\ a)) = \text{some}(p\ \{a\})$ .

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.utils.**OWLDatatypeRestriction** (type\_: *owlapy.owl\_datatype.OWLDatatype*,  
facet\_restrictions: *OWLFacetRestriction* | *Iterable[OWLFacetRestriction]*)

Bases: *owlapy.owl\_data\_ranges.OWLDataRange*

A datatype restriction `DatatypeRestriction( DT F1 lt1 ... Fn ltn )` consists of a unary datatype `DT` and `n` pairs `( Fi , lti )`. The resulting data range is unary and is obtained by restricting the value space of `DT` according to the semantics of all `( Fi , vi )` (multiple pairs are interpreted conjunctively), where `vi` are the data values of the literals `lti`. ([https://www.w3.org/TR/owl2-syntax/#Datatype\\_Restrictions](https://www.w3.org/TR/owl2-syntax/#Datatype_Restrictions))

**\_\_slots\_\_** = ('\_type', '\_facet\_restrictions')

**type\_index:** Final = 4006

**get\_datatype** () → *owlapy.owl\_datatype.OWLDatatype*

**get\_facet\_restrictions** () → *Sequence[OWLFacetRestriction]*

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

Return self==value.

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

Return hash(self).

**\_\_repr\_\_** ()

Return repr(self).

**class** owlapy.utils.**OWLFacetRestriction** (facet: *owlapy.vocab.OWLFacet*, literal: *Literals*)

Bases: *owlapy.owl\_object.OWLObject*

A facet restriction is used to restrict a particular datatype.

**\_\_slots\_\_** = ('\_facet', '\_literal')

**type\_index:** Final = 4007

**get\_facet** () → *owlapy.vocab.OWLFacet*

**get\_facet\_value** () → *owlapy.owl\_literal.OWLLiteral*

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

Return self==value.

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

Return hash(self).

```

__repr__()
    Return repr(self).

class owlapy.utils.OwlObjectOneOf (
    values: owlapy.owl_individual.OwlIndividual | Iterable[owlapy.owl_individual.OwlIndividual])
    Bases: owlapy.class_expression.class_expression.OwlAnonymousClassExpression,
    owlapy.meta_classes.HasOperands[owlapy.owl_individual.OwlIndividual]

    An enumeration of individuals ObjectOneOf( a1 ... an ) contains exactly the individuals ai with  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Enumeration\_of\_Individuals)

    __slots__ = '_values'

    type_index: Final = 3004

    individuals () → Iterable[owlapy.owl_individual.OwlIndividual]
        Gets the individuals that are in the oneOf. These individuals represent the exact instances (extension) of this
        class expression.

        Returns
            The individuals that are the values of this {@code ObjectOneOf} class expression.

    operands () → Iterable[owlapy.owl_individual.OwlIndividual]
        Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

        Returns
            The operands.

    as_object_union_of () → owlapy.class_expression.class_expression.OwlClassExpression
        Simplifies this enumeration to a union of singleton nominals.

        Returns
            This enumeration in a more standard DL form.  $\text{simp}(\{a\}) = \{a\}$   $\text{simp}(\{a_0, \dots, \{a_n\}\}) = \text{unionOf}(\{a_0\}, \dots, \{a_n\})$ 

    __hash__()
        Return hash(self).

    __eq__ (other)
        Return self==value.

    __repr__()
        Return repr(self).

class owlapy.utils.OwlDataComplementOf (data_range: OwlDataRange)
    Bases: OwlDataRange

    A complement data range DataComplementOf( DR ) contains all tuples of literals that are not contained in the
    data range DR. The resulting data range has the arity equal to the arity of DR.

    (https://www.w3.org/TR/owl2-syntax/#Complement\_of\_Data\_Ranges)

    type_index: Final = 4002

    get_data_range () → OwlDataRange

    Returns
        The wrapped data range.

```

```

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

```

**class** owlapy.utils.OWLDataUnionOf (operands: Iterable[OWLDataRange])

Bases: *OWLNaryDataRange*

A union data range DataUnionOf( DR1 ... DRn ) contains all tuples of literals that are contained in the at least one data range DRi for  $1 \leq i \leq n$ . All data ranges DRi must be of the same arity, and the resulting data range is of that arity as well.

([https://www.w3.org/TR/owl2-syntax/#Union\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Union_of_Data_Ranges))

```

__slots__ = '_operands'

type_index: Final = 4005

```

**class** owlapy.utils.OWLDataIntersectionOf (operands: Iterable[OWLDataRange])

Bases: *OWLNaryDataRange*

An intersection data range DataIntersectionOf( DR1 ... DRn ) contains all tuples of literals that are contained in each data range DRi for  $1 \leq i \leq n$ . All data ranges DRi must be of the same arity, and the resulting data range is of that arity as well.

([https://www.w3.org/TR/owl2-syntax/#Intersection\\_of\\_Data\\_Ranges](https://www.w3.org/TR/owl2-syntax/#Intersection_of_Data_Ranges))

```

__slots__ = '_operands'

type_index: Final = 4004

```

**class** owlapy.utils.OWLNaryDataRange (operands: Iterable[OWLDataRange])

Bases: *OWLDataRange*, *owlapy.meta\_classes.HasOperands*[OWLDataRange]

OWLNaryDataRange.

```

__slots__ = ()

```

**operands** () → Iterable[OWLDataRange]

Gets the operands - e.g., the individuals in a sameAs axiom, or the classes in an equivalent classes axiom.

**Returns**

The operands.

```

__repr__()
    Return repr(self).

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

```

**class** owlapy.utils.OWLDataRange

Bases: *OWLPropertyRange*

Represents a DataRange in the OWL 2 Specification.

```
class owlapy.utils.OWLPropertyRange
    Bases: owlapy.owl_object.OWLObject
    OWL Objects that can be the ranges of properties.
```

```
class owlapy.utils.OWLObject
```

Base interface for OWL objects

```
__slots__ = ()
```

```
abstract __eq__ (other)
```

Return self==value.

```
abstract __hash__ ()
```

Return hash(self).

```
abstract __repr__ ()
```

Return repr(self).

```
is_anonymous () → bool
```

```
class owlapy.utils.OWLDatatype (iri: owlapy.iri.IRI | owlapy.meta_classes.HasIRI)
```

Bases: *owlapy.owl\_object.OWLEntity*, *owlapy.owl\_data\_ranges.OWLDataRange*

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes, the main difference being that the former contain data values such as strings and numbers, rather than individuals. Datatypes are a kind of data range, which allows them to be used in restrictions. Each data range is associated with an arity; for datatypes, the arity is always one. The built-in datatype `rdfs:Literal` denotes any set of data values that contains the union of the value spaces of all datatypes.

(<https://www.w3.org/TR/owl2-syntax/#Datatypes>)

```
__slots__ = '_iri'
```

```
type_index: Final = 4001
```

```
property iri: owlapy.iri.IRI
```

Gets the IRI of this object.

**Returns**

The IRI of this object.

```
property str: str
```

Gets the string representation of this object

**Returns**

The IRI as string

```
class owlapy.utils.OWLClassExpressionLengthMetric (*, class_length: int,
    object_intersection_length: int, object_union_length: int, object_complement_length: int,
    object_some_values_length: int, object_all_values_length: int, object_has_value_length: int,
    object_cardinality_length: int, object_has_self_length: int, object_one_of_length: int,
    data_some_values_length: int, data_all_values_length: int, data_has_value_length: int,
    data_cardinality_length: int, object_property_length: int, object_inverse_length: int,
    data_property_length: int, datatype_length: int, data_one_of_length: int,
    data_complement_length: int, data_intersection_length: int, data_union_length: int)
```

Length calculation of OWLClassExpression

**Parameters**

- `class_length` – Class: “C”
- `object_intersection_length` – Intersection:  $A \sqcap B$
- `object_union_length` – Union:  $A \sqcup B$
- `object_complement_length` – Complement:  $\neg C$
- `object_some_values_length` – Obj. Some Values:  $\exists r.C$
- `object_all_values_length` – Obj. All Values:  $\forall r.C$
- `object_has_value_length` – Obj. Has Value:  $\exists r.\{I\}$
- `object_cardinality_length` – Obj. Cardinality restriction:  $\leq n\ r.C$
- `object_has_self_length` – Obj. Self restriction:  $\exists r.\text{Self}$
- `object_one_of_length` – Obj. One of:  $\exists r.\{X,Y,Z\}$
- `data_some_values_length` – Data Some Values:  $\exists p.t$
- `data_all_values_length` – Data All Values:  $\forall p.t$
- `data_has_value_length` – Data Has Value:  $\exists p.\{V\}$
- `data_cardinality_length` – Data Cardinality restriction:  $\leq n\ r.t$
- `object_property_length` – Obj. Property:  $\exists r.C$
- `object_inverse_length` – Inverse property:  $\exists r^{-}.C$
- `data_property_length` – Data Property:  $\exists p.t$
- `datatype_length` – Datatype:  $\wedge \text{datatype}$
- `data_one_of_length` – Data One of:  $\exists p.\{U,V,W\}$
- `data_complement_length` – Data Complement:  $\neg \text{datatype}$
- `data_intersection_length` – Data Intersection:  $\text{datatype} \sqcap \text{datatype}$
- `data_union_length` – Data Union:  $\text{datatype} \sqcup \text{datatype}$

```
__slots__ = ('class_length', 'object_intersection_length',
'object_union_length', ...
```

```
class_length: int
object_intersection_length: int
object_union_length: int
object_complement_length: int
object_some_values_length: int
object_all_values_length: int
object_has_value_length: int
object_cardinality_length: int
object_has_self_length: int
object_one_of_length: int
```

```

data_some_values_length: int
data_all_values_length: int
data_has_value_length: int
data_cardinality_length: int
object_property_length: int
object_inverse_length: int
data_property_length: int
datatype_length: int
data_one_of_length: int
data_complement_length: int
data_intersection_length: int
data_union_length: int
static get_default() → OWLClassExpressionLengthMetric
abstract length(o: owlapy.owl_object.OWLObject) → int

```

owlapy.utils.measurer

owlapy.utils.get\_expression\_length(ce: *owlapy.class\_expression.OWLClassExpression*) → int

**class** owlapy.utils.EvaluatedDescriptionSet (ordering: Callable[[\_N], \_O],  
max\_size: int = 10)

Bases: Generic[\_N, \_O]

Abstract base class for generic types.

A generic type is typically declared by inheriting from this class parameterized with one or more type variables. For example, a generic mapping type might be defined as:

```

class Mapping(Generic[KT, VT]):
    def __getitem__(self, key: KT) -> VT:
        ...
    # Etc.

```

This class can then be used as follows:

```

def lookup_name(mapping: Mapping[KT, VT], key: KT, default: VT) -> VT:
    try:
        return mapping[key]
    except KeyError:
        return default

```

```
__slots__ = ('items', '_max_size', '_Ordering')
```

```
items: SortedSet[_N]
```

```
maybe_add(node: _N)
```

```

clean()

worst()

best()

best_quality_value() → float

__iter__() → Iterable[_N]

class owlapy.utils.ConceptOperandSorter

    abstract sort(o: _O) → _O

class owlapy.utils.OperandSetTransform

    simplify(o: owlapy.class_expression.OWLClassExpression)
        → owlapy.class_expression.OWLClassExpression

class owlapy.utils.HasIndex
    Bases: Protocol

    Interface for types with an index; this is used to group objects by type when sorting.

    type_index: ClassVar[int]

    __eq__(other)
        Return self==value.

class owlapy.utils.OrderedOWLObject(o: _HasIndex)
    Holder of OWL Objects that can be used for Python sorted.

    The Ordering is dependent on the type_index of the impl. classes recursively followed by all components of the
    OWL Object.

    o
        OWL object.

    __slots__ = ('o', '_chain')

    o: _HasIndex

    __lt__(other)
        Return self<value.

    __eq__(other)
        Return self==value.

class owlapy.utils.NNF
    This class contains functions to transform a Class Expression into Negation Normal Form.

    abstract get_class_nnf(ce: owlapy.class_expression.OWLClassExpression,
        negated: bool = False) → owlapy.class_expression.OWLClassExpression
        Convert a Class Expression to Negation Normal Form. Operands will be sorted.

    Parameters
        • ce – Class Expression.
        • negated – Whether the result should be negated.

```



### Returns

Class Expression in Negation Normal Form.

**class** owlapy.utils.**TopLevelCNF**

This class contains functions to transform a class expression into Top-Level Conjunctive Normal Form.

**get\_top\_level\_cnf** (*ce: owlapy.class\_expression.OWLClassExpression*)  
→ *owlapy.class\_expression.OWLClassExpression*

Convert a class expression into Top-Level Conjunctive Normal Form. Operands will be sorted.

### Parameters

**ce** – Class Expression.

### Returns

Class Expression in Top-Level Conjunctive Normal Form.

**class** owlapy.utils.**TopLevelDNF**

This class contains functions to transform a class expression into Top-Level Disjunctive Normal Form.

**get\_top\_level\_dnf** (*ce: owlapy.class\_expression.OWLClassExpression*)  
→ *owlapy.class\_expression.OWLClassExpression*

Convert a class expression into Top-Level Disjunctive Normal Form. Operands will be sorted.

### Parameters

**ce** – Class Expression.

### Returns

Class Expression in Top-Level Disjunctive Normal Form.

owlapy.utils.**combine\_nary\_expressions** (*ce: owlapy.class\_expression.OWLClassExpression*)  
→ *owlapy.class\_expression.OWLClassExpression*

owlapy.utils.**combine\_nary\_expressions** (*ce: owlapy.owl\_data\_ranges.OWLDataRange*)  
→ *owlapy.owl\_data\_ranges.OWLDataRange*

Shortens an OWLClassExpression or OWLDataRange by combining all nested nary expressions of the same type. Operands will be sorted.

E.g. OWLObjectUnionOf(A, OWLObjectUnionOf(C, B)) -> OWLObjectUnionOf(A, B, C).

owlapy.utils.**iter\_count** (*i: Iterable*) → int

Count the number of elements in an iterable.

owlapy.utils.**as\_index** (*o: owlapy.owl\_object.OWLObject*) → *HasIndex*

Cast OWL Object to HasIndex.

**class** owlapy.utils.**LRUCache** (*maxsize: int | None = None*)

Bases: *Generic[\_K, \_V]*

Constants shares by all lru cache instances.

Adapted from functools.lru\_cache.

### **sentinel**

Unique object used to signal cache misses.

### **PREV**

Name for the link field 0.

### **NEXT**

Name for the link field 1.

**KEY**

Name for the link field 2.

**RESULT**

Name for the link field 3.

**sentinel**

**\_\_contains\_\_** (*item*: *\_K*) → bool

**\_\_getitem\_\_** (*item*: *\_K*) → *\_V*

**\_\_setitem\_\_** (*key*: *\_K*, *value*: *\_V*)

**cache\_info** ()

Report cache statistics.

**cache\_clear** ()

Clear the cache and cache statistics.

**owlapy.vocab**

Enumerations.

**Classes**

<i>HasIRI</i>	Simple class to access the IRI.
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>Namespaces</i>	Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup
<i>OWLRDFVocabulary</i>	Enumerations for OWL/RDF vocabulary.
<i>XSDVocabulary</i>	Enumerations for XSD vocabulary.
<i>OWLFacet</i>	Enumerations for OWL facets.

**Module Contents**

**class** owlapy.vocab.**HasIRI**

Simple class to access the IRI.

**\_\_slots\_\_** = ()

**property** iri: *IRI*

**Abstractmethod**

Gets the IRI of this object.

**Returns**

The IRI of this object.

**property str: str**

**Abstractmethod**

Gets the string representation of this object

**Returns**

The IRI as string

**class** owlapy.vocab.**IRI** (namespace: str | owlapy.namespaces.Namespaces, remainder: str)

Bases: owlapy.owl\_annotation.OWLAnnotationSubject, owlapy.owl\_annotation.OWLAnnotationValue

An IRI, consisting of a namespace and a remainder.

**\_\_slots\_\_** = ('\_namespace', '\_remainder', '\_\_weakref\_\_')

**type\_index: Final** = 0

**static create** (namespace: owlapy.namespaces.Namespaces, remainder: str) → IRI

**static create** (namespace: str, remainder: str) → IRI

**static create** (string: str) → IRI

**\_\_repr\_\_** ()

Return repr(self).

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

Return self==value.

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

Return hash(self).

**is\_nothing** ()

Determines if this IRI is equal to the IRI that owl:Nothing is named with.

**Returns**

True if this IRI is equal to <http://www.w3.org/2002/07/owl#Nothing> and otherwise False.

**is\_thing** ()

Determines if this IRI is equal to the IRI that owl:Thing is named with.

**Returns**

True if this IRI is equal to <http://www.w3.org/2002/07/owl#Thing> and otherwise False.

**is\_reserved\_vocabulary** () → bool

Determines if this IRI is in the reserved vocabulary. An IRI is in the reserved vocabulary if it starts with <http://www.w3.org/1999/02/22-rdf-syntax-ns#> or <http://www.w3.org/2000/01/rdf-schema#> or <http://www.w3.org/2001/XMLSchema#> or <http://www.w3.org/2002/07/owl#>.

**Returns**

True if the IRI is in the reserved vocabulary, otherwise False.

**as\_iri** () → IRI

**Returns**

if the value is an IRI, return it. Return None otherwise.

**as\_str** () → str

CD: Should be deprecated. :returns: The string that specifies the IRI.

**property str: str**

Returns: The string that specifies the IRI.

**property reminder: str**

Returns: The string corresponding to the reminder of the IRI.

**get\_short\_form()** → str

Gets the short form.

**Returns**

A string that represents the short form.

**get\_namespace()** → str

**Returns**

The namespace as string.

**get\_remainder()** → str

**Returns**

The remainder (coincident with NCName usually) for this IRI.

**class owlapy.vocab.Namespaces** (*prefix: str, ns: str*)

Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references

**\_\_slots\_\_** = ('\_prefix', '\_ns')

**property ns: str**

**property prefix: str**

**\_\_repr\_\_()**

Return repr(self).

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

Return hash(self).

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

Return self==value.

**class owlapy.vocab.OWLRFDFVocabulary** (*namespace: owlapy.namespaces.Namespaces, remainder: str*)

Bases: `_Vocabulary`, `enum.Enum`

Enumerations for OWL/RDF vocabulary.

**OWL\_THING**

**OWL\_NOTHING**

**OWL\_CLASS**

**OWL\_NAMED\_INDIVIDUAL**

**OWL\_TOP\_OBJECT\_PROPERTY**

**OWL\_BOTTOM\_OBJECT\_PROPERTY**

**OWL\_TOP\_DATA\_PROPERTY**

```

    OWL_BOTTOM_DATA_PROPERTY

    RDFS_LITERAL

class owlapy.vocab.XSDVocabulary(remainder: str)
    Bases: _Vocabulary, enum.Enum
    Enumerations for XSD vocabulary.
    DECIMAL: Final = 'decimal'
    INTEGER: Final = 'integer'
    LONG: Final = 'long'
    DOUBLE: Final = 'double'
    FLOAT: Final = 'float'
    BOOLEAN: Final = 'boolean'
    STRING: Final = 'string'
    DATE: Final = 'date'
    DATE_TIME: Final = 'dateTime'
    DATE_TIME_STAMP: Final = 'dateTimeStamp'
    DURATION: Final = 'duration'

class owlapy.vocab.OWLFacet(remainder: str, symbolic_form: str,
    operator: Callable[[_X, _X], bool])
    Bases: _Vocabulary, enum.Enum
    Enumerations for OWL facets.
    property symbolic_form
    property operator
    static from_str(name: str) → OWLFacet
    MIN_INCLUSIVE: Final
    MIN_EXCLUSIVE: Final
    MAX_INCLUSIVE: Final
    MAX_EXCLUSIVE: Final
    LENGTH: Final
    MIN_LENGTH: Final
    MAX_LENGTH: Final
    PATTERN: Final
    TOTAL_DIGITS: Final
    FRACTION_DIGITS: Final

```

## 6.3 Attributes

```
__version__
```

## 6.4 Functions

```
owl_expression_to_dl(→ str)
```

```
owl_expression_to_manchester(→ str)
```

```
dl_to_owl_expression(dl_expression, namespace)
```

```
manchester_to_owl_expression(manchester_ex  
...)
```

```
owl_expression_to_sparql(→ str)
```

Convert an OWL Class Expression ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions)) into a SPARQL query

## 6.5 Package Contents

```
owlapy.owl_expression_to_dl (o: owlapy.owl_object.OWLObject) → str
```

```
owlapy.owl_expression_to_manchester (o: owlapy.owl_object.OWLObject) → str
```

```
owlapy.dl_to_owl_expression (dl_expression: str, namespace: str)
```

```
owlapy.manchester_to_owl_expression (manchester_expression: str, namespace: str)
```

```
owlapy.owl_expression_to_sparql (  
    expression: owlapy.class_expression.OWLClassExpression = None, root_variable: str = '?x',  
    values: Iterable[owlapy.owl_individual.OWLNamedIndividual] | None = None,  
    for_all_de_morgan: bool = True, named_individuals: bool = False) → str
```

Convert an OWL Class Expression ([https://www.w3.org/TR/owl2-syntax/#Class\\_Expressions](https://www.w3.org/TR/owl2-syntax/#Class_Expressions)) into a SPARQL query  
root variable: the variable that will be projected expression: the class expression to be transformed to a SPARQL query

values: positive or negative examples from a class expression problem. Unclear for\_all\_de\_morgan: if set to True, the SPARQL mapping will use the mapping containing the nested FILTER NOT EXISTS patterns for the universal quantifier ( $\neg(\exists r. \neg C)$ ), instead of the counting query named\_individuals: if set to True, the generated SPARQL query will return only entities that are instances of owl:NamedIndividual

```
owlapy.__version__ = '1.1.0'
```

## Python Module Index

### O

- [owlapy](#), 17
- [owlapy.class\\_expression](#), 17
- [owlapy.class\\_expression.class\\_expression](#), 17
- [owlapy.class\\_expression.nary\\_boolean\\_expression](#), 20
- [owlapy.class\\_expression.owl\\_class](#), 22
- [owlapy.class\\_expression.restriction](#), 26
- [owlapy.converter](#), 62
- [owlapy.entities](#), 62
- [owlapy.iri](#), 80
- [owlapy.meta\\_classes](#), 83
- [owlapy.namespaces](#), 84
- [owlapy.owl\\_annotation](#), 85
- [owlapy.owl\\_axiom](#), 86
- [owlapy.owl\\_data\\_ranges](#), 115
- [owlapy.owl\\_datatype](#), 117
- [owlapy.owl\\_hierarchy](#), 120
- [owlapy.owl\\_individual](#), 132
- [owlapy.owl\\_literal](#), 135
- [owlapy.owl\\_object](#), 141
- [owlapy.owl\\_ontology](#), 144
- [owlapy.owl\\_ontology\\_manager](#), 173
- [owlapy.owl\\_property](#), 201
- [owlapy.owl\\_reasoner](#), 207
- [owlapy.parser](#), 253
- [owlapy.providers](#), 279
- [owlapy.render](#), 283
- [owlapy.static\\_funcs](#), 304
- [owlapy.utils](#), 305
- [owlapy.vocab](#), 330

## Index

### Non-alphabetical

`__contains__` () (owlapy.converter.VariablesMapping method), 78  
`__contains__` () (owlapy.owl\_hierarchy.AbstractHierarchy method), 131  
`__contains__` () (owlapy.owl\_reasoner.LRUCache method), 232  
`__contains__` () (owlapy.utils.LRUCache method), 330  
`__del__` () (owlapy.owl\_reasoner.SyncReasoner method), 253  
`__eq__` () (owlapy.class\_expression.class\_expression.OwlObjectComplementOf method), 19  
`__eq__` () (owlapy.class\_expression.nary\_boolean\_expression.OwlNaryBooleanClassExpression method), 21  
`__eq__` () (owlapy.class\_expression.owl\_class.IRI method), 24  
`__eq__` () (owlapy.class\_expression.owl\_class.OwlObjectComplementOf method), 23  
`__eq__` () (owlapy.class\_expression.OwlDataAllValuesFrom method), 59  
`__eq__` () (owlapy.class\_expression.OwlDataCardinalityRestriction method), 54  
`__eq__` () (owlapy.class\_expression.OwlDataHasValue method), 59  
`__eq__` () (owlapy.class\_expression.OwlDataOneOf method), 53  
`__eq__` () (owlapy.class\_expression.OwlDataSomeValuesFrom method), 58  
`__eq__` () (owlapy.class\_expression.OwlDatatypeRestriction method), 56  
`__eq__` () (owlapy.class\_expression.OwlFacetRestriction method), 57  
`__eq__` () (owlapy.class\_expression.OwlHasValueRestriction method), 51  
`__eq__` () (owlapy.class\_expression.OwlNaryBooleanClassExpression method), 49  
`__eq__` () (owlapy.class\_expression.OwlObjectAllValuesFrom method), 55  
`__eq__` () (owlapy.class\_expression.OwlObjectCardinalityRestriction method), 52  
`__eq__` () (owlapy.class\_expression.OwlObjectComplementOf method), 48  
`__eq__` () (owlapy.class\_expression.OwlObjectHasSelf method), 53  
`__eq__` () (owlapy.class\_expression.OwlObjectOneOf method), 61  
`__eq__` () (owlapy.class\_expression.OwlObjectSomeValuesFrom method), 54  
`__eq__` () (owlapy.class\_expression.restriction.OwlDataAllValuesFrom method), 43  
`__eq__` () (owlapy.class\_expression.restriction.OwlDataCardinalityRestriction method), 41  
`__eq__` () (owlapy.class\_expression.restriction.OwlDataHasValue method), 43  
`__eq__` () (owlapy.class\_expression.restriction.OwlDataOneOf method), 44  
`__eq__` () (owlapy.class\_expression.restriction.OwlDataSomeValuesFrom method), 42  
`__eq__` () (owlapy.class\_expression.restriction.OwlDatatypeRestriction method), 44  
`__eq__` () (owlapy.class\_expression.restriction.OwlFacetRestriction method), 45  
`__eq__` () (owlapy.class\_expression.restriction.OwlHasValueRestriction method), 35  
`__eq__` () (owlapy.class\_expression.restriction.OwlObject method), 33  
`__eq__` () (owlapy.class\_expression.restriction.OwlObjectAllValuesFrom method), 38  
`__eq__` () (owlapy.class\_expression.restriction.OwlObjectCardinalityRestriction method), 36  
`__eq__` () (owlapy.class\_expression.restriction.OwlObjectHasSelf method), 39  
`__eq__` () (owlapy.class\_expression.restriction.OwlObjectOneOf method), 40  
`__eq__` () (owlapy.class\_expression.restriction.OwlObjectSomeValuesFrom method), 38  
`__eq__` () (owlapy.converter.OwlDataAllValuesFrom method), 72  
`__eq__` () (owlapy.converter.OwlDataCardinalityRestriction method), 71  
`__eq__` () (owlapy.converter.OwlDataHasValue method), 73  
`__eq__` () (owlapy.converter.OwlDataOneOf method), 74  
`__eq__` () (owlapy.converter.OwlDataSomeValuesFrom method), 72  
`__eq__` () (owlapy.converter.OwlDatatypeRestriction method), 65  
`__eq__` () (owlapy.converter.OwlObjectAllValuesFrom method), 69  
`__eq__` () (owlapy.converter.OwlObjectCardinalityRestriction method), 70  
`__eq__` () (owlapy.converter.OwlObjectComplementOf method), 68  
`__eq__` () (owlapy.converter.OwlObjectHasSelf method), 71  
`__eq__` () (owlapy.converter.OwlObjectOneOf method), 65  
`__eq__` () (owlapy.converter.OwlObjectSomeValuesFrom method), 69  
`__eq__` () (owlapy.iri.IRI method), 82  
`__eq__` () (owlapy.iri.Namespaces method), 81  
`__eq__` () (owlapy.namespaces.Namespaces method), 85  
`__eq__` () (owlapy.owl\_annotation.OwlObject method), 85  
`__eq__` () (owlapy.owl\_axiom.IRI method), 94  
`__eq__` () (owlapy.owl\_axiom.OwlAnnotation method), 106  
`__eq__` () (owlapy.owl\_axiom.OwlAnnotationAssertionAxiom method), 107  
`__eq__` () (owlapy.owl\_axiom.OwlAnnotationPropertyDomainAxiom method), 108  
`__eq__` () (owlapy.owl\_axiom.OwlAnnotationPropertyRangeAxiom method), 108  
`__eq__` () (owlapy.owl\_axiom.OwlClassAssertionAxiom method), 105  
`__eq__` () (owlapy.owl\_axiom.OwlDataPropertyCharacteristicAxiom method), 113  
`__eq__` () (owlapy.owl\_axiom.OwlDatatypeDefinitionAxiom method), 99  
`__eq__` () (owlapy.owl\_axiom.OwlDeclarationAxiom method), 99



\_\_eq\_\_ () (owlapy.owl\_axiom.OWLDisjointUnionAxiom method), 105  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLHasKeyAxiom method), 100  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLNaryClassAxiom method), 101  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLNaryIndividualAxiom method), 102  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLNaryPropertyAxiom method), 103  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLObject method), 90  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLObjectPropertyCharacteristicAxiom method), 111  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLPropertyAssertionAxiom method), 109  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLPropertyDomainAxiom method), 113  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLPropertyRangeAxiom method), 114  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLSubAnnotationPropertyOfAxiom method), 107  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLSubClassOfAxiom method), 104  
 \_\_eq\_\_ () (owlapy.owl\_axiom.OWLSubPropertyAxiom method), 109  
 \_\_eq\_\_ () (owlapy.owl\_data\_ranges.OWLDataComplementOf method), 117  
 \_\_eq\_\_ () (owlapy.owl\_data\_ranges.OWLNaryDataRange method), 116  
 \_\_eq\_\_ () (owlapy.owl\_data\_ranges.OWLObject method), 115  
 \_\_eq\_\_ () (owlapy.owl\_datatype.IRI method), 118  
 \_\_eq\_\_ () (owlapy.owl\_individual.IRI method), 133  
 \_\_eq\_\_ () (owlapy.owl\_individual.OWLObject method), 133  
 \_\_eq\_\_ () (owlapy.owl\_object.OWLNamedObject method), 143  
 \_\_eq\_\_ () (owlapy.owl\_object.OWLObject method), 142  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.IRI method), 175  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.Ontology method), 196  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLAnnotationAssertionAxiom method), 183  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLClassAssertionAxiom method), 185  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLDataHasValue method), 178  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLDataPropertyCharacteristicAxiom method), 187  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLDeclarationAxiom method), 185  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLDisjointUnionAxiom method), 183  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLNaryBooleanClassExpression method), 179  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLNaryIndividualAxiom method), 191  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLNaryPropertyAxiom method), 190  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLObject method), 177  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectComplementOf method), 180  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectInverseOf method), 197  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectOneOf method), 180  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectPropertyCharacteristicAxiom method), 188  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLPropertyDomainAxiom method), 186  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLPropertyRangeAxiom method), 184  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLSubClassOfAxiom method), 182  
 \_\_eq\_\_ () (owlapy.owl\_ontology\_manager.OWLSubPropertyAxiom method), 184  
 \_\_eq\_\_ () (owlapy.owl\_ontology.IRI method), 153  
 \_\_eq\_\_ () (owlapy.owl\_ontology.Ontology method), 172  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLDataAllValuesFrom method), 160  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLDataComplementOf method), 149  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLDataHasValue method), 161  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLDataOneOf method), 162  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLDataSomeValuesFrom method), 160  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLDatatypeRestriction method), 162  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLFacetRestriction method), 164  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLObject method), 153  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLObjectAllValuesFrom method), 157  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLObjectComplementOf method), 156  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLObjectInverseOf method), 166  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLObjectOneOf method), 158  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLObjectSomeValuesFrom method), 157  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLOntologyID method), 168  
 \_\_eq\_\_ () (owlapy.owl\_ontology.OWLSubClassOfAxiom method), 148  
 \_\_eq\_\_ () (owlapy.owl\_property.IRI method), 203  
 \_\_eq\_\_ () (owlapy.owl\_property.OWLObject method), 202  
 \_\_eq\_\_ () (owlapy.owl\_property.OWLObjectInverseOf method), 206  
 \_\_eq\_\_ () (owlapy.owl\_reasoner.IRI method), 218  
 \_\_eq\_\_ () (owlapy.owl\_reasoner.Ontology method), 225  
 \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLDataAllValuesFrom method), 217  
 \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLDataComplementOf method), 220  
 \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLDataHasValue method), 216  
 \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLDataOneOf method), 215  
 \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLDataSomeValuesFrom method), 214

- \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLDatatypeRestriction method), 215
- \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLFacetRestriction method), 215
- \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLObjectAllValuesFrom method), 211
- \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLObjectCardinalityRestriction method), 214
- \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLObjectComplementOf method), 211
- \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLObjectInverseOf method), 228
- \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLObjectOneOf method), 212
- \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLObjectSomeValuesFrom method), 210
- \_\_eq\_\_ () (owlapy.owl\_reasoner.OWLSubClassOfAxiom method), 219
- \_\_eq\_\_ () (owlapy.parser.IRI method), 256
- \_\_eq\_\_ () (owlapy.parser.Namespaces method), 261
- \_\_eq\_\_ () (owlapy.parser.OWLDataAllValuesFrom method), 273
- \_\_eq\_\_ () (owlapy.parser.OWLDataCardinalityRestriction method), 272
- \_\_eq\_\_ () (owlapy.parser.OWLDataComplementOf method), 274
- \_\_eq\_\_ () (owlapy.parser.OWLDataHasValue method), 270
- \_\_eq\_\_ () (owlapy.parser.OWLDataOneOf method), 271
- \_\_eq\_\_ () (owlapy.parser.OWLDataSomeValuesFrom method), 268
- \_\_eq\_\_ () (owlapy.parser.OWLDatatypeRestriction method), 272
- \_\_eq\_\_ () (owlapy.parser.OWLFacetRestriction method), 267
- \_\_eq\_\_ () (owlapy.parser.OWLObjectAllValuesFrom method), 272
- \_\_eq\_\_ () (owlapy.parser.OWLObjectCardinalityRestriction method), 271
- \_\_eq\_\_ () (owlapy.parser.OWLObjectComplementOf method), 266
- \_\_eq\_\_ () (owlapy.parser.OWLObjectHasSelf method), 262
- \_\_eq\_\_ () (owlapy.parser.OWLObjectOneOf method), 265
- \_\_eq\_\_ () (owlapy.parser.OWLObjectSomeValuesFrom method), 263
- \_\_eq\_\_ () (owlapy.providers.OWLDatatypeRestriction method), 281
- \_\_eq\_\_ () (owlapy.providers.OWLFacetRestriction method), 282
- \_\_eq\_\_ () (owlapy.render.IRI method), 285
- \_\_eq\_\_ () (owlapy.render.OWLDataAllValuesFrom method), 296
- \_\_eq\_\_ () (owlapy.render.OWLDataComplementOf method), 300
- \_\_eq\_\_ () (owlapy.render.OWLDataHasValue method), 296
- \_\_eq\_\_ () (owlapy.render.OWLDataOneOf method), 298
- \_\_eq\_\_ () (owlapy.render.OWLDataSomeValuesFrom method), 295
- \_\_eq\_\_ () (owlapy.render.OWLDatatypeRestriction method), 302
- \_\_eq\_\_ () (owlapy.render.OWLFacetRestriction method), 301
- \_\_eq\_\_ () (owlapy.render.OWLNaryBooleanClassExpression method), 298
- \_\_eq\_\_ () (owlapy.render.OWLNaryDataRange method), 300
- \_\_eq\_\_ () (owlapy.render.OWLObject method), 289
- \_\_eq\_\_ () (owlapy.render.OWLObjectAllValuesFrom method), 293
- \_\_eq\_\_ () (owlapy.render.OWLObjectComplementOf method), 294
- \_\_eq\_\_ () (owlapy.render.OWLObjectHasSelf method), 295
- \_\_eq\_\_ () (owlapy.render.OWLObjectInverseOf method), 290
- \_\_eq\_\_ () (owlapy.render.OWLObjectOneOf method), 302
- \_\_eq\_\_ () (owlapy.render.OWLObjectSomeValuesFrom method), 292
- \_\_eq\_\_ () (owlapy.utils.HasIndex method), 328
- \_\_eq\_\_ () (owlapy.utils.OrderedOWLObject method), 328
- \_\_eq\_\_ () (owlapy.utils.OWLDataAllValuesFrom method), 318
- \_\_eq\_\_ () (owlapy.utils.OWLDataCardinalityRestriction method), 320
- \_\_eq\_\_ () (owlapy.utils.OWLDataComplementOf method), 324
- \_\_eq\_\_ () (owlapy.utils.OWLDataHasValue method), 318
- \_\_eq\_\_ () (owlapy.utils.OWLDataOneOf method), 320
- \_\_eq\_\_ () (owlapy.utils.OWLDataSomeValuesFrom method), 319
- \_\_eq\_\_ () (owlapy.utils.OWLDatatypeRestriction method), 322
- \_\_eq\_\_ () (owlapy.utils.OWLFacetRestriction method), 322
- \_\_eq\_\_ () (owlapy.utils.OWLNaryBooleanClassExpression method), 321
- \_\_eq\_\_ () (owlapy.utils.OWLNaryDataRange method), 324
- \_\_eq\_\_ () (owlapy.utils.OWLObject method), 325
- \_\_eq\_\_ () (owlapy.utils.OWLObjectAllValuesFrom method), 319
- \_\_eq\_\_ () (owlapy.utils.OWLObjectCardinalityRestriction method), 314
- \_\_eq\_\_ () (owlapy.utils.OWLObjectComplementOf method), 314
- \_\_eq\_\_ () (owlapy.utils.OWLObjectHasSelf method), 316
- \_\_eq\_\_ () (owlapy.utils.OWLObjectInverseOf method), 311
- \_\_eq\_\_ () (owlapy.utils.OWLObjectOneOf method), 323
- \_\_eq\_\_ () (owlapy.utils.OWLObjectSomeValuesFrom method), 315
- \_\_eq\_\_ () (owlapy.vocab.IRI method), 331
- \_\_eq\_\_ () (owlapy.vocab.Namespaces method), 332
- \_\_getitem\_\_ () (owlapy.converter.VariablesMapping method), 79
- \_\_getitem\_\_ () (owlapy.owl\_reasoner.LRUCache method), 232

\_\_getitem\_\_() (owlapy.utils.LRUCache method), 330  
 \_\_hash\_\_() (owlapy.class\_expression.class\_expression.OwlObjectComplementOf method), 19  
 \_\_hash\_\_() (owlapy.class\_expression.nary\_boolean\_expression.OwlNaryBooleanClassExpression method), 21  
 \_\_hash\_\_() (owlapy.class\_expression.owl\_class.IRI method), 24  
 \_\_hash\_\_() (owlapy.class\_expression.owl\_class.OwlObjectComplementOf method), 23  
 \_\_hash\_\_() (owlapy.class\_expression.OwlDataAllValuesFrom method), 59  
 \_\_hash\_\_() (owlapy.class\_expression.OwlDataCardinalityRestriction method), 54  
 \_\_hash\_\_() (owlapy.class\_expression.OwlDataHasValue method), 59  
 \_\_hash\_\_() (owlapy.class\_expression.OwlDataOneOf method), 53  
 \_\_hash\_\_() (owlapy.class\_expression.OwlDataSomeValuesFrom method), 58  
 \_\_hash\_\_() (owlapy.class\_expression.OwlDatatypeRestriction method), 56  
 \_\_hash\_\_() (owlapy.class\_expression.OwlFacetRestriction method), 57  
 \_\_hash\_\_() (owlapy.class\_expression.OwlHasValueRestriction method), 51  
 \_\_hash\_\_() (owlapy.class\_expression.OwlNaryBooleanClassExpression method), 49  
 \_\_hash\_\_() (owlapy.class\_expression.OwlObjectAllValuesFrom method), 55  
 \_\_hash\_\_() (owlapy.class\_expression.OwlObjectCardinalityRestriction method), 52  
 \_\_hash\_\_() (owlapy.class\_expression.OwlObjectComplementOf method), 48  
 \_\_hash\_\_() (owlapy.class\_expression.OwlObjectHasSelf method), 53  
 \_\_hash\_\_() (owlapy.class\_expression.OwlObjectOneOf method), 61  
 \_\_hash\_\_() (owlapy.class\_expression.OwlObjectSomeValuesFrom method), 54  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlDataAllValuesFrom method), 43  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlDataCardinalityRestriction method), 41  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlDataHasValue method), 43  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlDataOneOf method), 44  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlDataSomeValuesFrom method), 42  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlDatatypeRestriction method), 44  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlFacetRestriction method), 45  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlHasValueRestriction method), 35  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlObject method), 34  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlObjectAllValuesFrom method), 38  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlObjectCardinalityRestriction method), 37  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlObjectHasSelf method), 39  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlObjectOneOf method), 40  
 \_\_hash\_\_() (owlapy.class\_expression.restriction.OwlObjectSomeValuesFrom method), 38  
 \_\_hash\_\_() (owlapy.converter.OwlDataAllValuesFrom method), 72  
 \_\_hash\_\_() (owlapy.converter.OwlDataCardinalityRestriction method), 71  
 \_\_hash\_\_() (owlapy.converter.OwlDataHasValue method), 73  
 \_\_hash\_\_() (owlapy.converter.OwlDataOneOf method), 74  
 \_\_hash\_\_() (owlapy.converter.OwlDataSomeValuesFrom method), 72  
 \_\_hash\_\_() (owlapy.converter.OwlDatatypeRestriction method), 65  
 \_\_hash\_\_() (owlapy.converter.OwlObjectAllValuesFrom method), 69  
 \_\_hash\_\_() (owlapy.converter.OwlObjectCardinalityRestriction method), 70  
 \_\_hash\_\_() (owlapy.converter.OwlObjectComplementOf method), 68  
 \_\_hash\_\_() (owlapy.converter.OwlObjectHasSelf method), 71  
 \_\_hash\_\_() (owlapy.converter.OwlObjectOneOf method), 65  
 \_\_hash\_\_() (owlapy.converter.OwlObjectSomeValuesFrom method), 69  
 \_\_hash\_\_() (owlapy.iri.IRI method), 82  
 \_\_hash\_\_() (owlapy.iri.Namespaces method), 81  
 \_\_hash\_\_() (owlapy.namespaces.Namespaces method), 85  
 \_\_hash\_\_() (owlapy.owl\_annotation.OwlObject method), 85  
 \_\_hash\_\_() (owlapy.owl\_axiom.IRI method), 94  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlAnnotation method), 106  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlAnnotationAssertionAxiom method), 107  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlAnnotationPropertyDomainAxiom method), 108  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlAnnotationPropertyRangeAxiom method), 108  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlClassAssertionAxiom method), 105  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlDataPropertyCharacteristicAxiom method), 113  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlDatatypeDefinitionAxiom method), 99  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlDeclarationAxiom method), 99  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlDisjointUnionAxiom method), 105  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlHasKeyAxiom method), 100  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlNaryClassAxiom method), 101  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlNaryIndividualAxiom method), 102  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlNaryPropertyAxiom method), 103  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlObject method), 90  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlObjectPropertyCharacteristicAxiom method), 111  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlPropertyAssertionAxiom method), 110  
 \_\_hash\_\_() (owlapy.owl\_axiom.OwlPropertyDomainAxiom method), 113

\_\_hash\_\_ () (owlapy.owl\_axiom.OWLPropertyRangeAxiom method), 114  
 \_\_hash\_\_ () (owlapy.owl\_axiom.OWLSubAnnotationPropertyOfAxiom method), 107  
 \_\_hash\_\_ () (owlapy.owl\_axiom.OWLSubClassOfAxiom method), 104  
 \_\_hash\_\_ () (owlapy.owl\_axiom.OWLSubPropertyAxiom method), 109  
 \_\_hash\_\_ () (owlapy.owl\_data\_ranges.OWLDataComplementOf method), 117  
 \_\_hash\_\_ () (owlapy.owl\_data\_ranges.OWLNaryDataRange method), 116  
 \_\_hash\_\_ () (owlapy.owl\_data\_ranges.OWLObject method), 115  
 \_\_hash\_\_ () (owlapy.owl\_datatype.IRI method), 118  
 \_\_hash\_\_ () (owlapy.owl\_individual.IRI method), 134  
 \_\_hash\_\_ () (owlapy.owl\_individual.OWLObject method), 133  
 \_\_hash\_\_ () (owlapy.owl\_object.OWLNamedObject method), 143  
 \_\_hash\_\_ () (owlapy.owl\_object.OWLObject method), 142  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.IRI method), 175  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.Ontology method), 196  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLAnnotationAssertionAxiom method), 183  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLClassAssertionAxiom method), 185  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLDataHasValue method), 178  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLDataPropertyCharacteristicAxiom method), 187  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLDeclarationAxiom method), 185  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLDisjointUnionAxiom method), 183  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLNaryBooleanClassExpression method), 179  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLNaryIndividualAxiom method), 191  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLNaryPropertyAxiom method), 190  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLObject method), 177  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectComplementOf method), 180  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectInverseOf method), 197  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectOneOf method), 180  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectPropertyCharacteristicAxiom method), 188  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLPropertyDomainAxiom method), 186  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLPropertyRangeAxiom method), 184  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLSubClassOfAxiom method), 182  
 \_\_hash\_\_ () (owlapy.owl\_ontology\_manager.OWLSubPropertyAxiom method), 184  
 \_\_hash\_\_ () (owlapy.owl\_ontology.IRI method), 153  
 \_\_hash\_\_ () (owlapy.owl\_ontology.Ontology method), 172  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLDataAllValuesFrom method), 160  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLDataComplementOf method), 149  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLDataHasValue method), 161  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLDataOneOf method), 162  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLDataSomeValuesFrom method), 160  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLDatatypeRestriction method), 163  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLFacetRestriction method), 164  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLObject method), 153  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLObjectAllValuesFrom method), 157  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLObjectComplementOf method), 156  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLObjectInverseOf method), 166  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLObjectOneOf method), 158  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLObjectSomeValuesFrom method), 157  
 \_\_hash\_\_ () (owlapy.owl\_ontology.OWLSubClassOfAxiom method), 148  
 \_\_hash\_\_ () (owlapy.owl\_property.IRI method), 203  
 \_\_hash\_\_ () (owlapy.owl\_property.OWLObject method), 202  
 \_\_hash\_\_ () (owlapy.owl\_property.OWLObjectInverseOf method), 206  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.IRI method), 218  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.Ontology method), 225  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLDataAllValuesFrom method), 217  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLDataComplementOf method), 220  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLDataHasValue method), 216  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLDataOneOf method), 215  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLDataSomeValuesFrom method), 214  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLDatatypeRestriction method), 215  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLFacetRestriction method), 215  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLObjectAllValuesFrom method), 211  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLObjectCardinalityRestriction method), 214  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLObjectComplementOf method), 211  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLObjectInverseOf method), 228  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLObjectOneOf method), 212  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLObjectSomeValuesFrom method), 210  
 \_\_hash\_\_ () (owlapy.owl\_reasoner.OWLSubClassOfAxiom method), 219  
 \_\_hash\_\_ () (owlapy.parser.IRI method), 256



\_\_hash\_\_ () (owlapy.parser.Namespaces method), 261  
 \_\_hash\_\_ () (owlapy.parser.OWLDataAllValuesFrom method), 273  
 \_\_hash\_\_ () (owlapy.parser.OWLDataCardinalityRestriction method), 272  
 \_\_hash\_\_ () (owlapy.parser.OWLDataComplementOf method), 274  
 \_\_hash\_\_ () (owlapy.parser.OWLDataHasValue method), 270  
 \_\_hash\_\_ () (owlapy.parser.OWLDataOneOf method), 271  
 \_\_hash\_\_ () (owlapy.parser.OWLDataSomeValuesFrom method), 268  
 \_\_hash\_\_ () (owlapy.parser.OWLDatatypeRestriction method), 272  
 \_\_hash\_\_ () (owlapy.parser.OWLFacetRestriction method), 267  
 \_\_hash\_\_ () (owlapy.parser.OWLObjectAllValuesFrom method), 272  
 \_\_hash\_\_ () (owlapy.parser.OWLObjectCardinalityRestriction method), 271  
 \_\_hash\_\_ () (owlapy.parser.OWLObjectComplementOf method), 266  
 \_\_hash\_\_ () (owlapy.parser.OWLObjectHasSelf method), 262  
 \_\_hash\_\_ () (owlapy.parser.OWLObjectOneOf method), 265  
 \_\_hash\_\_ () (owlapy.parser.OWLObjectSomeValuesFrom method), 263  
 \_\_hash\_\_ () (owlapy.providers.OWLDatatypeRestriction method), 281  
 \_\_hash\_\_ () (owlapy.providers.OWLFacetRestriction method), 282  
 \_\_hash\_\_ () (owlapy.render.IRI method), 285  
 \_\_hash\_\_ () (owlapy.render.OWLDataAllValuesFrom method), 296  
 \_\_hash\_\_ () (owlapy.render.OWLDataComplementOf method), 300  
 \_\_hash\_\_ () (owlapy.render.OWLDataHasValue method), 297  
 \_\_hash\_\_ () (owlapy.render.OWLDataOneOf method), 298  
 \_\_hash\_\_ () (owlapy.render.OWLDataSomeValuesFrom method), 295  
 \_\_hash\_\_ () (owlapy.render.OWLDatatypeRestriction method), 302  
 \_\_hash\_\_ () (owlapy.render.OWLFacetRestriction method), 301  
 \_\_hash\_\_ () (owlapy.render.OWLNaryBooleanClassExpression method), 298  
 \_\_hash\_\_ () (owlapy.render.OWLNaryDataRange method), 300  
 \_\_hash\_\_ () (owlapy.render.OWLObject method), 289  
 \_\_hash\_\_ () (owlapy.render.OWLObjectAllValuesFrom method), 293  
 \_\_hash\_\_ () (owlapy.render.OWLObjectComplementOf method), 294  
 \_\_hash\_\_ () (owlapy.render.OWLObjectHasSelf method), 295  
 \_\_hash\_\_ () (owlapy.render.OWLObjectInverseOf method), 290  
 \_\_hash\_\_ () (owlapy.render.OWLObjectOneOf method), 302  
 \_\_hash\_\_ () (owlapy.render.OWLObjectSomeValuesFrom method), 292  
 \_\_hash\_\_ () (owlapy.utils.OWLDataAllValuesFrom method), 318  
 \_\_hash\_\_ () (owlapy.utils.OWLDataCardinalityRestriction method), 321  
 \_\_hash\_\_ () (owlapy.utils.OWLDataComplementOf method), 324  
 \_\_hash\_\_ () (owlapy.utils.OWLDataHasValue method), 318  
 \_\_hash\_\_ () (owlapy.utils.OWLDataOneOf method), 320  
 \_\_hash\_\_ () (owlapy.utils.OWLDataSomeValuesFrom method), 319  
 \_\_hash\_\_ () (owlapy.utils.OWLDatatypeRestriction method), 322  
 \_\_hash\_\_ () (owlapy.utils.OWLFacetRestriction method), 322  
 \_\_hash\_\_ () (owlapy.utils.OWLNaryBooleanClassExpression method), 321  
 \_\_hash\_\_ () (owlapy.utils.OWLNaryDataRange method), 324  
 \_\_hash\_\_ () (owlapy.utils.OWLObject method), 325  
 \_\_hash\_\_ () (owlapy.utils.OWLObjectAllValuesFrom method), 319  
 \_\_hash\_\_ () (owlapy.utils.OWLObjectCardinalityRestriction method), 314  
 \_\_hash\_\_ () (owlapy.utils.OWLObjectComplementOf method), 314  
 \_\_hash\_\_ () (owlapy.utils.OWLObjectHasSelf method), 316  
 \_\_hash\_\_ () (owlapy.utils.OWLObjectInverseOf method), 311  
 \_\_hash\_\_ () (owlapy.utils.OWLObjectOneOf method), 323  
 \_\_hash\_\_ () (owlapy.utils.OWLObjectSomeValuesFrom method), 315  
 \_\_hash\_\_ () (owlapy.vocab.IRI method), 331  
 \_\_hash\_\_ () (owlapy.vocab.Namespaces method), 332  
 iter\_\_ () (owlapy.utils.EvaluatedDescriptionSet method), 328  
 len\_\_ () (owlapy.owl\_hierarchy.AbstractHierarchy method), 131  
 lt\_\_ () (owlapy.owl\_object.OWLNamedObject method), 143  
 lt\_\_ () (owlapy.utils.OrderedOWLObject method), 328  
 repr\_\_ () (owlapy.class\_expression.class\_expression.OWLObjectComplementOf method), 19  
 repr\_\_ () (owlapy.class\_expression.nary\_boolean\_expression.OWLNaryBooleanClassExpression method), 21  
 repr\_\_ () (owlapy.class\_expression.owl\_class.IRI method), 24  
 repr\_\_ () (owlapy.class\_expression.owl\_class.OWLObjectComplementOf method), 23  
 repr\_\_ () (owlapy.class\_expression.OWLDataAllValuesFrom method), 59  
 repr\_\_ () (owlapy.class\_expression.OWLDataCardinalityRestriction method), 54  
 repr\_\_ () (owlapy.class\_expression.OWLDataHasValue method), 59  
 repr\_\_ () (owlapy.class\_expression.OWLDataOneOf method), 53  
 repr\_\_ () (owlapy.class\_expression.OWLDataSomeValuesFrom method), 58  
 repr\_\_ () (owlapy.class\_expression.OWLDatatypeRestriction method), 56

\_\_repr\_\_ () (owlapy.class\_expression.OwlFacetRestriction method), 57  
\_\_repr\_\_ () (owlapy.class\_expression.OwlNaryBooleanClassExpression method), 49  
\_\_repr\_\_ () (owlapy.class\_expression.OwlObjectAllValuesFrom method), 55  
\_\_repr\_\_ () (owlapy.class\_expression.OwlObjectCardinalityRestriction method), 52  
\_\_repr\_\_ () (owlapy.class\_expression.OwlObjectComplementOf method), 48  
\_\_repr\_\_ () (owlapy.class\_expression.OwlObjectHasSelf method), 53  
\_\_repr\_\_ () (owlapy.class\_expression.OwlObjectHasValue method), 55  
\_\_repr\_\_ () (owlapy.class\_expression.OwlObjectOneOf method), 61  
\_\_repr\_\_ () (owlapy.class\_expression.OwlObjectSomeValuesFrom method), 54  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlDataAllValuesFrom method), 43  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlDataCardinalityRestriction method), 41  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlDataHasValue method), 43  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlDataOneOf method), 44  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlDataSomeValuesFrom method), 42  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlDatatypeRestriction method), 44  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlFacetRestriction method), 45  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlObject method), 34  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlObjectAllValuesFrom method), 38  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlObjectCardinalityRestriction method), 36  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlObjectHasSelf method), 39  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlObjectHasValue method), 39  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlObjectOneOf method), 40  
\_\_repr\_\_ () (owlapy.class\_expression.restriction.OwlObjectSomeValuesFrom method), 38  
\_\_repr\_\_ () (owlapy.converter.OwlDataAllValuesFrom method), 72  
\_\_repr\_\_ () (owlapy.converter.OwlDataCardinalityRestriction method), 71  
\_\_repr\_\_ () (owlapy.converter.OwlDataHasValue method), 73  
\_\_repr\_\_ () (owlapy.converter.OwlDataOneOf method), 74  
\_\_repr\_\_ () (owlapy.converter.OwlDataSomeValuesFrom method), 72  
\_\_repr\_\_ () (owlapy.converter.OwlDatatypeRestriction method), 65  
\_\_repr\_\_ () (owlapy.converter.OwlObjectAllValuesFrom method), 69  
\_\_repr\_\_ () (owlapy.converter.OwlObjectCardinalityRestriction method), 70  
\_\_repr\_\_ () (owlapy.converter.OwlObjectComplementOf method), 68  
\_\_repr\_\_ () (owlapy.converter.OwlObjectHasSelf method), 71  
\_\_repr\_\_ () (owlapy.converter.OwlObjectHasValue method), 64  
\_\_repr\_\_ () (owlapy.converter.OwlObjectOneOf method), 65  
\_\_repr\_\_ () (owlapy.converter.OwlObjectSomeValuesFrom method), 69  
\_\_repr\_\_ () (owlapy.iri.IRI method), 81  
\_\_repr\_\_ () (owlapy.iri.Namespaces method), 81  
\_\_repr\_\_ () (owlapy.namespaces.Namespaces method), 85  
\_\_repr\_\_ () (owlapy.owl\_annotation.OwlObject method), 86  
\_\_repr\_\_ () (owlapy.owl\_axiom.IRI method), 94  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlAnnotation method), 106  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlAnnotationAssertionAxiom method), 107  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlAnnotationPropertyDomainAxiom method), 108  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlAnnotationPropertyRangeAxiom method), 108  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlClassAssertionAxiom method), 105  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlDataPropertyCharacteristicAxiom method), 113  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlDatatypeDefinitionAxiom method), 99  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlDeclarationAxiom method), 99  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlDisjointUnionAxiom method), 105  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlHasKeyAxiom method), 100  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlInverseObjectPropertiesAxiom method), 104  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlNaryClassAxiom method), 101  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlNaryIndividualAxiom method), 102  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlNaryPropertyAxiom method), 103  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlObject method), 90  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlObjectPropertyCharacteristicAxiom method), 111  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlPropertyAssertionAxiom method), 110  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlPropertyDomainAxiom method), 113  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlPropertyRangeAxiom method), 114  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlSubAnnotationPropertyOfAxiom method), 107  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlSubClassOfAxiom method), 105  
\_\_repr\_\_ () (owlapy.owl\_axiom.OwlSubPropertyAxiom method), 109  
\_\_repr\_\_ () (owlapy.owl\_data\_ranges.OwlDataComplementOf method), 117  
\_\_repr\_\_ () (owlapy.owl\_data\_ranges.OwlNaryDataRange method), 116  
\_\_repr\_\_ () (owlapy.owl\_data\_ranges.OwlObject method), 116  
\_\_repr\_\_ () (owlapy.owl\_datatype.IRI method), 118  
\_\_repr\_\_ () (owlapy.owl\_individual.IRI method), 133

\_\_repr\_\_ () (owlapy.owl\_individual.OWLObject method), 133  
\_\_repr\_\_ () (owlapy.owl\_object.OWLNamedObject method), 143  
\_\_repr\_\_ () (owlapy.owl\_object.OWLObject method), 142  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.iri method), 175  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.Ontology method), 196  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLAnnotationAssertionAxiom method), 183  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLClassAssertionAxiom method), 185  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLDataHasValue method), 178  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLDataPropertyCharacteristicAxiom method), 187  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLDeclarationAxiom method), 185  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLDisjointUnionAxiom method), 183  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLInverseObjectPropertiesAxiom method), 190  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLNaryBooleanClassExpression method), 179  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLNaryIndividualAxiom method), 191  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLNaryPropertyAxiom method), 190  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLObject method), 177  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectComplementOf method), 180  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectHasValue method), 181  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectInverseOf method), 197  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectOneOf method), 180  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLObjectPropertyCharacteristicAxiom method), 188  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLPropertyDomainAxiom method), 186  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLPropertyRangeAxiom method), 184  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLSubClassOfAxiom method), 182  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.OWLSubPropertyAxiom method), 184  
\_\_repr\_\_ () (owlapy.owl\_ontology\_manager.iri method), 153  
\_\_repr\_\_ () (owlapy.owl\_ontology.Ontology method), 172  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLDataAllValuesFrom method), 160  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLDataComplementOf method), 149  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLDataHasValue method), 161  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLDataOneOf method), 162  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLDataSomeValuesFrom method), 159  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLDatatypeRestriction method), 163  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLFacetRestriction method), 164  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLObject method), 153  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLObjectAllValuesFrom method), 157  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLObjectComplementOf method), 156  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLObjectHasValue method), 159  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLObjectInverseOf method), 166  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLObjectOneOf method), 158  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLObjectSomeValuesFrom method), 156  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLOntologyID method), 168  
\_\_repr\_\_ () (owlapy.owl\_ontology.OWLSubClassOfAxiom method), 148  
\_\_repr\_\_ () (owlapy.owl\_property.iri method), 203  
\_\_repr\_\_ () (owlapy.owl\_property.OWLObject method), 202  
\_\_repr\_\_ () (owlapy.owl\_property.OWLObjectInverseOf method), 206  
\_\_repr\_\_ () (owlapy.owl\_reasoner.iri method), 218  
\_\_repr\_\_ () (owlapy.owl\_reasoner.Ontology method), 225  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLDataAllValuesFrom method), 216  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLDataComplementOf method), 220  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLDataHasValue method), 216  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLDataOneOf method), 215  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLDataSomeValuesFrom method), 214  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLDatatypeRestriction method), 215  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLFacetRestriction method), 216  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLObjectAllValuesFrom method), 211  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLObjectCardinalityRestriction method), 214  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLObjectComplementOf method), 211  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLObjectHasValue method), 212  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLObjectInverseOf method), 228  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLObjectOneOf method), 212  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLObjectSomeValuesFrom method), 210  
\_\_repr\_\_ () (owlapy.owl\_reasoner.OWLSubClassOfAxiom method), 219  
\_\_repr\_\_ () (owlapy.parser.iri method), 255  
\_\_repr\_\_ () (owlapy.parser.Namespaces method), 261  
\_\_repr\_\_ () (owlapy.parser.OWLDataAllValuesFrom method), 273  
\_\_repr\_\_ () (owlapy.parser.OWLDataCardinalityRestriction method), 272  
\_\_repr\_\_ () (owlapy.parser.OWLDataComplementOf method), 274

- \_\_repr\_\_ () (owlapy.parser.OWLDataHasValue method), 270
- \_\_repr\_\_ () (owlapy.parser.OWLDataOneOf method), 271
- \_\_repr\_\_ () (owlapy.parser.OWLDataSomeValuesFrom method), 268
- \_\_repr\_\_ () (owlapy.parser.OWLDatatypeRestriction method), 272
- \_\_repr\_\_ () (owlapy.parser.OWLFacetRestriction method), 268
- \_\_repr\_\_ () (owlapy.parser.OWLObjectAllValuesFrom method), 272
- \_\_repr\_\_ () (owlapy.parser.OWLObjectCardinalityRestriction method), 271
- \_\_repr\_\_ () (owlapy.parser.OWLObjectComplementOf method), 266
- \_\_repr\_\_ () (owlapy.parser.OWLObjectHasSelf method), 262
- \_\_repr\_\_ () (owlapy.parser.OWLObjectHasValue method), 269
- \_\_repr\_\_ () (owlapy.parser.OWLObjectOneOf method), 265
- \_\_repr\_\_ () (owlapy.parser.OWLObjectSomeValuesFrom method), 263
- \_\_repr\_\_ () (owlapy.providers.OWLDatatypeRestriction method), 281
- \_\_repr\_\_ () (owlapy.providers.OWLFacetRestriction method), 282
- \_\_repr\_\_ () (owlapy.render.IRI method), 285
- \_\_repr\_\_ () (owlapy.render.OWLDataAllValuesFrom method), 296
- \_\_repr\_\_ () (owlapy.render.OWLDataComplementOf method), 300
- \_\_repr\_\_ () (owlapy.render.OWLDataHasValue method), 296
- \_\_repr\_\_ () (owlapy.render.OWLDataOneOf method), 298
- \_\_repr\_\_ () (owlapy.render.OWLDataSomeValuesFrom method), 295
- \_\_repr\_\_ () (owlapy.render.OWLDatatypeRestriction method), 302
- \_\_repr\_\_ () (owlapy.render.OWLFacetRestriction method), 301
- \_\_repr\_\_ () (owlapy.render.OWLNaryBooleanClassExpression method), 298
- \_\_repr\_\_ () (owlapy.render.OWLNaryDataRange method), 300
- \_\_repr\_\_ () (owlapy.render.OWLObject method), 289
- \_\_repr\_\_ () (owlapy.render.OWLObjectAllValuesFrom method), 293
- \_\_repr\_\_ () (owlapy.render.OWLObjectComplementOf method), 294
- \_\_repr\_\_ () (owlapy.render.OWLObjectHasSelf method), 295
- \_\_repr\_\_ () (owlapy.render.OWLObjectHasValue method), 301
- \_\_repr\_\_ () (owlapy.render.OWLObjectInverseOf method), 290
- \_\_repr\_\_ () (owlapy.render.OWLObjectOneOf method), 302
- \_\_repr\_\_ () (owlapy.render.OWLObjectSomeValuesFrom method), 292
- \_\_repr\_\_ () (owlapy.utils.OWLDataAllValuesFrom method), 318
- \_\_repr\_\_ () (owlapy.utils.OWLDataCardinalityRestriction method), 320
- \_\_repr\_\_ () (owlapy.utils.OWLDataComplementOf method), 323
- \_\_repr\_\_ () (owlapy.utils.OWLDataHasValue method), 318
- \_\_repr\_\_ () (owlapy.utils.OWLDataOneOf method), 320
- \_\_repr\_\_ () (owlapy.utils.OWLDataSomeValuesFrom method), 319
- \_\_repr\_\_ () (owlapy.utils.OWLDatatypeRestriction method), 322
- \_\_repr\_\_ () (owlapy.utils.OWLFacetRestriction method), 322
- \_\_repr\_\_ () (owlapy.utils.OWLNaryBooleanClassExpression method), 321
- \_\_repr\_\_ () (owlapy.utils.OWLNaryDataRange method), 324
- \_\_repr\_\_ () (owlapy.utils.OWLObject method), 325
- \_\_repr\_\_ () (owlapy.utils.OWLObjectAllValuesFrom method), 319
- \_\_repr\_\_ () (owlapy.utils.OWLObjectCardinalityRestriction method), 314
- \_\_repr\_\_ () (owlapy.utils.OWLObjectComplementOf method), 314
- \_\_repr\_\_ () (owlapy.utils.OWLObjectHasSelf method), 317
- \_\_repr\_\_ () (owlapy.utils.OWLObjectHasValue method), 322
- \_\_repr\_\_ () (owlapy.utils.OWLObjectInverseOf method), 311
- \_\_repr\_\_ () (owlapy.utils.OWLObjectOneOf method), 323
- \_\_repr\_\_ () (owlapy.utils.OWLObjectSomeValuesFrom method), 315
- \_\_repr\_\_ () (owlapy.vocab.IRI method), 331
- \_\_repr\_\_ () (owlapy.vocab.Namespaces method), 332
- \_\_setitem\_\_ () (owlapy.owl\_reasoner.LRUCache method), 232
- \_\_setitem\_\_ () (owlapy.utils.LRUCache method), 330
- \_\_slots\_\_ (owlapy.class\_expression.class\_expression.HasOperands attribute), 17
- \_\_slots\_\_ (owlapy.class\_expression.class\_expression.OWLBooleanClassExpression attribute), 19
- \_\_slots\_\_ (owlapy.class\_expression.class\_expression.OWLClassExpression attribute), 18
- \_\_slots\_\_ (owlapy.class\_expression.class\_expression.OWLObjectComplementOf attribute), 19
- \_\_slots\_\_ (owlapy.class\_expression.nary\_boolean\_expression.HasOperands attribute), 21
- \_\_slots\_\_ (owlapy.class\_expression.nary\_boolean\_expression.OWLBooleanClassExpression attribute), 21
- \_\_slots\_\_ (owlapy.class\_expression.nary\_boolean\_expression.OWLClassExpression attribute), 20
- \_\_slots\_\_ (owlapy.class\_expression.nary\_boolean\_expression.OWLNaryBooleanClassExpression attribute), 21
- \_\_slots\_\_ (owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf attribute), 22
- \_\_slots\_\_ (owlapy.class\_expression.nary\_boolean\_expression.OWLObjectUnionOf attribute), 21
- \_\_slots\_\_ (owlapy.class\_expression.owl\_class.IRI attribute), 24
- \_\_slots\_\_ (owlapy.class\_expression.owl\_class.OWLClass attribute), 25
- \_\_slots\_\_ (owlapy.class\_expression.owl\_class.OWLClassExpression attribute), 22



\_\_slots\_\_ (owlapy.class\_expression.owl\_class.OWLEntity attribute), 23  
 \_\_slots\_\_ (owlapy.class\_expression.owl\_class.OWLObjectComplementOf attribute), 23  
 \_\_slots\_\_ (owlapy.class\_expression.OWLBooleanClassExpression attribute), 48  
 \_\_slots\_\_ (owlapy.class\_expression.OWLCardinalityRestriction attribute), 52  
 \_\_slots\_\_ (owlapy.class\_expression.OWLClass attribute), 48  
 \_\_slots\_\_ (owlapy.class\_expression.OWLClassExpression attribute), 47  
 \_\_slots\_\_ (owlapy.class\_expression.OWLDataAllValuesFrom attribute), 59  
 \_\_slots\_\_ (owlapy.class\_expression.OWLDataCardinalityRestriction attribute), 54  
 \_\_slots\_\_ (owlapy.class\_expression.OWLDataExactCardinality attribute), 60  
 \_\_slots\_\_ (owlapy.class\_expression.OWLDataHasValue attribute), 59  
 \_\_slots\_\_ (owlapy.class\_expression.OWLDataMaxCardinality attribute), 60  
 \_\_slots\_\_ (owlapy.class\_expression.OWLDataMinCardinality attribute), 59  
 \_\_slots\_\_ (owlapy.class\_expression.OWLDataRestriction attribute), 52  
 \_\_slots\_\_ (owlapy.class\_expression.OWLDataSomeValuesFrom attribute), 58  
 \_\_slots\_\_ (owlapy.class\_expression.OWLDatatypeRestriction attribute), 56  
 \_\_slots\_\_ (owlapy.class\_expression.OWLFacetRestriction attribute), 57  
 \_\_slots\_\_ (owlapy.class\_expression.OWLHasValueRestriction attribute), 51  
 \_\_slots\_\_ (owlapy.class\_expression.OWLNaryBooleanClassExpression attribute), 49  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectAllValuesFrom attribute), 55  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectCardinalityRestriction attribute), 52  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectComplementOf attribute), 48  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectExactCardinality attribute), 58  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectHasSelf attribute), 53  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectHasValue attribute), 55  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectIntersectionOf attribute), 50  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectMaxCardinality attribute), 57  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectMinCardinality attribute), 57  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectOneOf attribute), 60  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectRestriction attribute), 51  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectSomeValuesFrom attribute), 54  
 \_\_slots\_\_ (owlapy.class\_expression.OWLObjectUnionOf attribute), 50  
 \_\_slots\_\_ (owlapy.class\_expression.OWLQuantifiedDataRestriction attribute), 54  
 \_\_slots\_\_ (owlapy.class\_expression.OWLQuantifiedObjectRestriction attribute), 51  
 \_\_slots\_\_ (owlapy.class\_expression.OWLQuantifiedRestriction attribute), 50  
 \_\_slots\_\_ (owlapy.class\_expression.OWLRestriction attribute), 50  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.HasCardinality attribute), 28  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.HasFiller attribute), 28  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.HasOperands attribute), 28  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLCardinalityRestriction attribute), 36  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLClassExpression attribute), 29  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDataAllValuesFrom attribute), 42  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDataCardinalityRestriction attribute), 41  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDataExactCardinality attribute), 42  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDataHasValue attribute), 43  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDataMaxCardinality attribute), 41  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDataMinCardinality attribute), 41  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDataPropertyExpression attribute), 31  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDataRestriction attribute), 40  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDataSomeValuesFrom attribute), 42  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDatatype attribute), 33  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLDatatypeRestriction attribute), 44  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLFacetRestriction attribute), 44  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLHasValueRestriction attribute), 35  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLIndividual attribute), 33  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLLiteral attribute), 31  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObject attribute), 33  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectAllValuesFrom attribute), 38  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectCardinalityRestriction attribute), 36  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectExactCardinality attribute), 37  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectHasSelf attribute), 38  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectHasValue attribute), 39  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectIntersectionOf attribute), 28  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectMaxCardinality attribute), 37  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectMinCardinality attribute), 37  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectOneOf attribute), 39  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectPropertyExpression attribute), 30  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectRestriction attribute), 35  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OWLObjectSomeValuesFrom attribute), 38

\_\_slots\_\_ (owlapy.class\_expression.restriction.OwlObjectUnionOf attribute), 29  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OwlPropertyExpression attribute), 30  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OwlQuantifiedDataRestriction attribute), 40  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OwlQuantifiedObjectRestriction attribute), 36  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OwlQuantifiedRestriction attribute), 35  
 \_\_slots\_\_ (owlapy.class\_expression.restriction.OwlRestriction attribute), 34  
 \_\_slots\_\_ (owlapy.converter.Owl2SparqlConverter attribute), 79  
 \_\_slots\_\_ (owlapy.converter.OwlClass attribute), 66  
 \_\_slots\_\_ (owlapy.converter.OwlClassExpression attribute), 67  
 \_\_slots\_\_ (owlapy.converter.OwlDataAllValuesFrom attribute), 72  
 \_\_slots\_\_ (owlapy.converter.OwlDataCardinalityRestriction attribute), 71  
 \_\_slots\_\_ (owlapy.converter.OwlDataExactCardinality attribute), 66  
 \_\_slots\_\_ (owlapy.converter.OwlDataHasValue attribute), 73  
 \_\_slots\_\_ (owlapy.converter.OwlDataMaxCardinality attribute), 66  
 \_\_slots\_\_ (owlapy.converter.OwlDataMinCardinality attribute), 65  
 \_\_slots\_\_ (owlapy.converter.OwlDataProperty attribute), 76  
 \_\_slots\_\_ (owlapy.converter.OwlDataSomeValuesFrom attribute), 72  
 \_\_slots\_\_ (owlapy.converter.OwlDatatype attribute), 77  
 \_\_slots\_\_ (owlapy.converter.OwlDatatypeRestriction attribute), 65  
 \_\_slots\_\_ (owlapy.converter.OwlEntity attribute), 77  
 \_\_slots\_\_ (owlapy.converter.OwlLiteral attribute), 74  
 \_\_slots\_\_ (owlapy.converter.OwlNamedIndividual attribute), 74  
 \_\_slots\_\_ (owlapy.converter.OwlObjectAllValuesFrom attribute), 69  
 \_\_slots\_\_ (owlapy.converter.OwlObjectCardinalityRestriction attribute), 69  
 \_\_slots\_\_ (owlapy.converter.OwlObjectComplementOf attribute), 68  
 \_\_slots\_\_ (owlapy.converter.OwlObjectExactCardinality attribute), 70  
 \_\_slots\_\_ (owlapy.converter.OwlObjectHasSelf attribute), 71  
 \_\_slots\_\_ (owlapy.converter.OwlObjectHasValue attribute), 64  
 \_\_slots\_\_ (owlapy.converter.OwlObjectIntersectionOf attribute), 68  
 \_\_slots\_\_ (owlapy.converter.OwlObjectMaxCardinality attribute), 70  
 \_\_slots\_\_ (owlapy.converter.OwlObjectMinCardinality attribute), 70  
 \_\_slots\_\_ (owlapy.converter.OwlObjectOneOf attribute), 64  
 \_\_slots\_\_ (owlapy.converter.OwlObjectProperty attribute), 76  
 \_\_slots\_\_ (owlapy.converter.OwlObjectSomeValuesFrom attribute), 69  
 \_\_slots\_\_ (owlapy.converter.OwlObjectUnionOf attribute), 68  
 \_\_slots\_\_ (owlapy.converter.VariablesMapping attribute), 78  
 \_\_slots\_\_ (owlapy.iri.IRI attribute), 81  
 \_\_slots\_\_ (owlapy.iri.Namespaces attribute), 81  
 \_\_slots\_\_ (owlapy.iri.OwlAnnotationSubject attribute), 81  
 \_\_slots\_\_ (owlapy.iri.OwlAnnotationValue attribute), 81  
 \_\_slots\_\_ (owlapy.meta\_classes.HasCardinality attribute), 84  
 \_\_slots\_\_ (owlapy.meta\_classes.HasFiller attribute), 84  
 \_\_slots\_\_ (owlapy.meta\_classes.HasIRI attribute), 83  
 \_\_slots\_\_ (owlapy.meta\_classes.HasOperands attribute), 83  
 \_\_slots\_\_ (owlapy.namespaces.Namespaces attribute), 85  
 \_\_slots\_\_ (owlapy.owl\_annotation.OwlAnnotationObject attribute), 86  
 \_\_slots\_\_ (owlapy.owl\_annotation.OwlAnnotationSubject attribute), 86  
 \_\_slots\_\_ (owlapy.owl\_annotation.OwlAnnotationValue attribute), 86  
 \_\_slots\_\_ (owlapy.owl\_annotation.OwlObject attribute), 85  
 \_\_slots\_\_ (owlapy.owl\_axiom.HasOperands attribute), 91  
 \_\_slots\_\_ (owlapy.owl\_axiom.IRI attribute), 94  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAnnotation attribute), 106  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAnnotationAssertionAxiom attribute), 107  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAnnotationAxiom attribute), 106  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAnnotationProperty attribute), 106  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAnnotationPropertyDomainAxiom attribute), 108  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAnnotationPropertyRangeAxiom attribute), 108  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAnnotationSubject attribute), 95  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAnnotationValue attribute), 96  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAsymmetricObjectPropertyAxiom attribute), 111  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlAxiom attribute), 98  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlClass attribute), 93  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlClassAssertionAxiom attribute), 105  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlClassAxiom attribute), 99  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlClassExpression attribute), 92  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlDataPropertyAssertionAxiom attribute), 110  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlDataPropertyAxiom attribute), 98  
 \_\_slots\_\_ (owlapy.owl\_axiom.OwlDataPropertyCharacteristicAxiom attribute), 113

```

_slots_ (owlapy.owl_axiom.OWLDataPropertyDomainAxiom attribute), 114
_slots_ (owlapy.owl_axiom.OWLDataPropertyExpression attribute), 90
_slots_ (owlapy.owl_axiom.OWLDataPropertyRangeAxiom attribute), 115
_slots_ (owlapy.owl_axiom.OWLDatatype attribute), 91
_slots_ (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom attribute), 99
_slots_ (owlapy.owl_axiom.OWLDeclarationAxiom attribute), 99
_slots_ (owlapy.owl_axiom.OWLDifferentIndividualsAxiom attribute), 102
_slots_ (owlapy.owl_axiom.OWLDisjointClassesAxiom attribute), 101
_slots_ (owlapy.owl_axiom.OWLDisjointDataPropertiesAxiom attribute), 104
_slots_ (owlapy.owl_axiom.OWLDisjointObjectPropertiesAxiom attribute), 103
_slots_ (owlapy.owl_axiom.OWLDisjointUnionAxiom attribute), 105
_slots_ (owlapy.owl_axiom.OWLEntity attribute), 90
_slots_ (owlapy.owl_axiom.OWLEquivalentClassesAxiom attribute), 101
_slots_ (owlapy.owl_axiom.OWLEquivalentDataPropertiesAxiom attribute), 104
_slots_ (owlapy.owl_axiom.OWLEquivalentObjectPropertiesAxiom attribute), 103
_slots_ (owlapy.owl_axiom.OWLFunctionalDataPropertyAxiom attribute), 113
_slots_ (owlapy.owl_axiom.OWLFunctionalObjectPropertyAxiom attribute), 111
_slots_ (owlapy.owl_axiom.OWLHasKeyAxiom attribute), 100
_slots_ (owlapy.owl_axiom.OWLIndividual attribute), 94
_slots_ (owlapy.owl_axiom.OWLIndividualAxiom attribute), 99
_slots_ (owlapy.owl_axiom.OWLInverseFunctionalObjectPropertyAxiom attribute), 112
_slots_ (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom attribute), 103
_slots_ (owlapy.owl_axiom.OWLIrreflexiveObjectPropertyAxiom attribute), 112
_slots_ (owlapy.owl_axiom.OWLLiteral attribute), 96
_slots_ (owlapy.owl_axiom.OWLLogicalAxiom attribute), 98
_slots_ (owlapy.owl_axiom.OWLNaryAxiom attribute), 100
_slots_ (owlapy.owl_axiom.OWLNaryClassAxiom attribute), 101
_slots_ (owlapy.owl_axiom.OWLNaryIndividualAxiom attribute), 102
_slots_ (owlapy.owl_axiom.OWLNaryPropertyAxiom attribute), 102
_slots_ (owlapy.owl_axiom.OWLNegativeDataPropertyAssertionAxiom attribute), 111
_slots_ (owlapy.owl_axiom.OWLNegativeObjectPropertyAssertionAxiom attribute), 110
_slots_ (owlapy.owl_axiom.OWLObject attribute), 90
_slots_ (owlapy.owl_axiom.OWLObjectPropertyAssertionAxiom attribute), 110
_slots_ (owlapy.owl_axiom.OWLObjectPropertyAxiom attribute), 98
_slots_ (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom attribute), 111
_slots_ (owlapy.owl_axiom.OWLObjectPropertyDomainAxiom attribute), 114
_slots_ (owlapy.owl_axiom.OWLObjectPropertyExpression attribute), 90
_slots_ (owlapy.owl_axiom.OWLObjectPropertyRangeAxiom attribute), 115
_slots_ (owlapy.owl_axiom.OWLObjectUnionOf attribute), 94
_slots_ (owlapy.owl_axiom.OWLProperty attribute), 92
_slots_ (owlapy.owl_axiom.OWLPropertyAssertionAxiom attribute), 109
_slots_ (owlapy.owl_axiom.OWLPropertyAxiom attribute), 98
_slots_ (owlapy.owl_axiom.OWLPropertyDomainAxiom attribute), 113
_slots_ (owlapy.owl_axiom.OWLPropertyExpression attribute), 91
_slots_ (owlapy.owl_axiom.OWLPropertyRangeAxiom attribute), 114
_slots_ (owlapy.owl_axiom.OWLReflexiveObjectPropertyAxiom attribute), 112
_slots_ (owlapy.owl_axiom.OWLSameIndividualAxiom attribute), 102
_slots_ (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom attribute), 107
_slots_ (owlapy.owl_axiom.OWLSubClassOfAxiom attribute), 104
_slots_ (owlapy.owl_axiom.OWLSubDataPropertyOfAxiom attribute), 109
_slots_ (owlapy.owl_axiom.OWLSubObjectPropertyOfAxiom attribute), 109
_slots_ (owlapy.owl_axiom.OWLSubPropertyAxiom attribute), 108
_slots_ (owlapy.owl_axiom.OWLSymmetricObjectPropertyAxiom attribute), 112
_slots_ (owlapy.owl_axiom.OWLTransitiveObjectPropertyAxiom attribute), 113
_slots_ (owlapy.owl_axiom.OWLUnaryPropertyAxiom attribute), 111
_slots_ (owlapy.owl_data_ranges.HasOperands attribute), 116
_slots_ (owlapy.owl_data_ranges.OWLDataIntersectionOf attribute), 117
_slots_ (owlapy.owl_data_ranges.OWLDataUnionOf attribute), 117
_slots_ (owlapy.owl_data_ranges.OWLNaryDataRange attribute), 116
_slots_ (owlapy.owl_data_ranges.OWLObject attribute), 115
_slots_ (owlapy.owl_datatype.HasIRI attribute), 119
_slots_ (owlapy.owl_datatype.IRI attribute), 118
_slots_ (owlapy.owl_datatype.OWLDatatype attribute), 120
_slots_ (owlapy.owl_datatype.OWLEntity attribute), 118
_slots_ (owlapy.owl_hierarchy.AbstractHierarchy attribute), 129
_slots_ (owlapy.owl_hierarchy.HasIRI attribute), 122
_slots_ (owlapy.owl_hierarchy.OWLClass attribute), 121
_slots_ (owlapy.owl_hierarchy.OWLDataProperty attribute), 123

```

- \_\_slots\_\_ (owlapy.owl\_hierarchy.OWLObjectProperty attribute), 122
- \_\_slots\_\_ (owlapy.owl\_hierarchy.OWLReasoner attribute), 123
- \_\_slots\_\_ (owlapy.owl\_individual.IRI attribute), 133
- \_\_slots\_\_ (owlapy.owl\_individual.OWLEntity attribute), 133
- \_\_slots\_\_ (owlapy.owl\_individual.OWLIndividual attribute), 134
- \_\_slots\_\_ (owlapy.owl\_individual.OWLNamedIndividual attribute), 135
- \_\_slots\_\_ (owlapy.owl\_individual.OWLObject attribute), 133
- \_\_slots\_\_ (owlapy.owl\_literal.OWLAnnotationValue attribute), 137
- \_\_slots\_\_ (owlapy.owl\_literal.OWLObjectProperty attribute), 139
- \_\_slots\_\_ (owlapy.owl\_literal.OWLDatatype attribute), 137
- \_\_slots\_\_ (owlapy.owl\_literal.OWLLiteral attribute), 139
- \_\_slots\_\_ (owlapy.owl\_literal.OWLObjectProperty attribute), 138
- \_\_slots\_\_ (owlapy.owl\_object.HasIRI attribute), 142
- \_\_slots\_\_ (owlapy.owl\_object.OWLEntity attribute), 143
- \_\_slots\_\_ (owlapy.owl\_object.OWLNamedObject attribute), 143
- \_\_slots\_\_ (owlapy.owl\_object.OWLObject attribute), 142
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.AddImport attribute), 200
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.HasIRI attribute), 176
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.IRI attribute), 175
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.Ontology attribute), 194
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OntologyManager attribute), 200
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLAnnotationAssertionAxiom attribute), 183
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLAnnotationProperty attribute), 184
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLAsymmetricObjectPropertyAxiom attribute), 186
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLAxiom attribute), 181
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLClass attribute), 177
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLClassAssertionAxiom attribute), 185
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectHasValue attribute), 178
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectProperty attribute), 196
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectPropertyAssertionAxiom attribute), 187
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectPropertyCharacteristicAxiom attribute), 187
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLDeclarationAxiom attribute), 185
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLDifferentIndividualsAxiom attribute), 191
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLDisjointClassesAxiom attribute), 191
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLDisjointDataPropertiesAxiom attribute), 189
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLDisjointObjectPropertiesAxiom attribute), 189
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLDisjointUnionAxiom attribute), 183
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLEquivalentClassesAxiom attribute), 182
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLEquivalentDataPropertiesAxiom attribute), 189
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLEquivalentObjectPropertiesAxiom attribute), 189
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLFunctionalDataPropertyAxiom attribute), 187
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLFunctionalObjectPropertyAxiom attribute), 188
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLImportsDeclaration attribute), 199
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLIndividual attribute), 192
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLInverseFunctionalObjectPropertyAxiom attribute), 188
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLInverseObjectPropertiesAxiom attribute), 190
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLIrreflexiveObjectPropertyAxiom attribute), 188
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLNamedIndividual attribute), 191
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLNaryBooleanClassExpression attribute), 179
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLNaryIndividualAxiom attribute), 190
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLNaryPropertyAxiom attribute), 190
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObject attribute), 177
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectComplementOf attribute), 180
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectHasValue attribute), 180
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectInverseOf attribute), 197
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectOneOf attribute), 179
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectProperty attribute), 197
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectPropertyAssertionAxiom attribute), 185
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectPropertyCharacteristicAxiom attribute), 188
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLObjectPropertyRangeAxiom attribute), 181
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLOntology attribute), 192
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLOntologyChange attribute), 198
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLProperty attribute), 198
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLPropertyDomainAxiom attribute), 186
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLPropertyRangeAxiom attribute), 184
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLQuantifiedDataRestriction attribute), 178
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLQuantifiedObjectRestriction attribute), 181
- \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLReflexiveObjectPropertyAxiom attribute), 187



\_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLSameIndividualAxiom attribute), 191  
 \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLSubClassOfAxiom attribute), 182  
 \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLSubPropertyAxiom attribute), 184  
 \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLSymmetricObjectPropertyAxiom attribute), 186  
 \_\_slots\_\_ (owlapy.owl\_ontology\_manager.OWLTransitiveObjectPropertyAxiom attribute), 186  
 \_\_slots\_\_ (owlapy.owl\_ontology\_manager.ToOwlready2 attribute), 196  
 \_\_slots\_\_ (owlapy.owl\_ontology.FromOwlready2 attribute), 172  
 \_\_slots\_\_ (owlapy.owl\_ontology.IRI attribute), 153  
 \_\_slots\_\_ (owlapy.owl\_ontology.Ontology attribute), 170  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLAnnotationProperty attribute), 148  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLClass attribute), 154  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLClassAxiom attribute), 147  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLClassExpression attribute), 155  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataAllValuesFrom attribute), 160  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataExactCardinality attribute), 160  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataHasValue attribute), 161  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataIntersectionOf attribute), 149  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataMaxCardinality attribute), 161  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataMinCardinality attribute), 161  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataProperty attribute), 164  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataPropertyDomainAxiom attribute), 147  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataPropertyExpression attribute), 167  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataPropertyRangeAxiom attribute), 147  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataRestriction attribute), 163  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataSomeValuesFrom attribute), 159  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDatatype attribute), 150  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDatatypeRestriction attribute), 162  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLDataUnionOf attribute), 149  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLEquivalentClassesAxiom attribute), 146  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLFacetRestriction attribute), 164  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLIndividual attribute), 150  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLLiteral attribute), 151  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLNamedIndividual attribute), 150  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObject attribute), 152  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectAllValuesFrom attribute), 157  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectComplementOf attribute), 155  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectExactCardinality attribute), 158  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectHasValue attribute), 159  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectIntersectionOf attribute), 156  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectInverseOf attribute), 166  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectMaxCardinality attribute), 158  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectMinCardinality attribute), 159  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectOneOf attribute), 157  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectProperty attribute), 164  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectPropertyDomainAxiom attribute), 147  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectPropertyExpression attribute), 166  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectPropertyRangeAxiom attribute), 148  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectRestriction attribute), 163  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectSomeValuesFrom attribute), 156  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLObjectUnionOf attribute), 156  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLOntology attribute), 168  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLOntologyID attribute), 167  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLPropertyExpression attribute), 165  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLRestriction attribute), 163  
 \_\_slots\_\_ (owlapy.owl\_ontology.OWLSubClassOfAxiom attribute), 148  
 \_\_slots\_\_ (owlapy.owl\_ontology.ToOwlready2 attribute), 172  
 \_\_slots\_\_ (owlapy.owl\_property.IRI attribute), 202  
 \_\_slots\_\_ (owlapy.owl\_property.OWLDataProperty attribute), 206  
 \_\_slots\_\_ (owlapy.owl\_property.OWLDataPropertyExpression attribute), 205  
 \_\_slots\_\_ (owlapy.owl\_property.OWLEntity attribute), 202  
 \_\_slots\_\_ (owlapy.owl\_property.OWLObject attribute), 202  
 \_\_slots\_\_ (owlapy.owl\_property.OWLObjectInverseOf attribute), 206  
 \_\_slots\_\_ (owlapy.owl\_property.OWLObjectProperty attribute), 205  
 \_\_slots\_\_ (owlapy.owl\_property.OWLObjectPropertyExpression attribute), 204  
 \_\_slots\_\_ (owlapy.owl\_property.OWLProperty attribute), 205  
 \_\_slots\_\_ (owlapy.owl\_property.OWLPropertyExpression attribute), 204  
 \_\_slots\_\_ (owlapy.owl\_reasoner.FastInstanceCheckerReasoner attribute), 246  
 \_\_slots\_\_ (owlapy.owl\_reasoner.IRI attribute), 218

- \_\_slots\_\_ (owlapy.owl\_reasoner.Ontology attribute), 223
- \_\_slots\_\_ (owlapy.owl\_reasoner.OntologyManager attribute), 225
- \_\_slots\_\_ (owlapy.owl\_reasoner.OntologyReasoner attribute), 239
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLAxiom attribute), 219
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLClass attribute), 217
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLClassExpression attribute), 209
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLDataAllValuesFrom attribute), 216
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLDataHasValue attribute), 216
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLDataIntersectionOf attribute), 220
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLDataProperty attribute), 227
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLDataPropertyExpression attribute), 229
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLDataSomeValuesFrom attribute), 214
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLDatatype attribute), 221
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLDatatypeRestriction attribute), 215
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLDataUnionOf attribute), 220
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLFacetRestriction attribute), 215
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLLiteral attribute), 229
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLNamedIndividual attribute), 229
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectAllValuesFrom attribute), 211
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectCardinalityRestriction attribute), 213
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectComplementOf attribute), 210
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectExactCardinality attribute), 213
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectHasValue attribute), 212
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectIntersectionOf attribute), 210
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectInverseOf attribute), 228
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectMaxCardinality attribute), 213
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectMinCardinality attribute), 213
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectOneOf attribute), 211
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectProperty attribute), 227
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectPropertyExpression attribute), 226
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectSomeValuesFrom attribute), 210
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLObjectUnionOf attribute), 210
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLOntology attribute), 221
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLPropertyExpression attribute), 228
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLReasoner attribute), 232
- \_\_slots\_\_ (owlapy.owl\_reasoner.OWLSubClassOfAxiom attribute), 219
- \_\_slots\_\_ (owlapy.owl\_reasoner.SyncReasoner attribute), 252
- \_\_slots\_\_ (owlapy.owl\_reasoner.ToOwlready2 attribute), 225
- \_\_slots\_\_ (owlapy.parser.IRI attribute), 255
- \_\_slots\_\_ (owlapy.parser.Namespaces attribute), 260
- \_\_slots\_\_ (owlapy.parser.OWLClass attribute), 264
- \_\_slots\_\_ (owlapy.parser.OWLClassExpression attribute), 265
- \_\_slots\_\_ (owlapy.parser.OWLDataAllValuesFrom attribute), 273
- \_\_slots\_\_ (owlapy.parser.OWLDataCardinalityRestriction attribute), 272
- \_\_slots\_\_ (owlapy.parser.OWLDataExactCardinality attribute), 268
- \_\_slots\_\_ (owlapy.parser.OWLDataHasValue attribute), 270
- \_\_slots\_\_ (owlapy.parser.OWLDataIntersectionOf attribute), 273
- \_\_slots\_\_ (owlapy.parser.OWLDataMaxCardinality attribute), 269
- \_\_slots\_\_ (owlapy.parser.OWLDataMinCardinality attribute), 270
- \_\_slots\_\_ (owlapy.parser.OWLDataProperty attribute), 260
- \_\_slots\_\_ (owlapy.parser.OWLDataSomeValuesFrom attribute), 268
- \_\_slots\_\_ (owlapy.parser.OWLDatatype attribute), 261
- \_\_slots\_\_ (owlapy.parser.OWLDatatypeRestriction attribute), 271
- \_\_slots\_\_ (owlapy.parser.OWLDataUnionOf attribute), 273
- \_\_slots\_\_ (owlapy.parser.OWLFacetRestriction attribute), 267
- \_\_slots\_\_ (owlapy.parser.OWLLiteral attribute), 257
- \_\_slots\_\_ (owlapy.parser.OWLNamedIndividual attribute), 257
- \_\_slots\_\_ (owlapy.parser.OWLObjectAllValuesFrom attribute), 272
- \_\_slots\_\_ (owlapy.parser.OWLObjectCardinalityRestriction attribute), 271
- \_\_slots\_\_ (owlapy.parser.OWLObjectComplementOf attribute), 266
- \_\_slots\_\_ (owlapy.parser.OWLObjectExactCardinality attribute), 266
- \_\_slots\_\_ (owlapy.parser.OWLObjectHasSelf attribute), 262
- \_\_slots\_\_ (owlapy.parser.OWLObjectHasValue attribute), 269
- \_\_slots\_\_ (owlapy.parser.OWLObjectIntersectionOf attribute), 263
- \_\_slots\_\_ (owlapy.parser.OWLObjectMaxCardinality attribute), 269
- \_\_slots\_\_ (owlapy.parser.OWLObjectMinCardinality attribute), 263
- \_\_slots\_\_ (owlapy.parser.OWLObjectOneOf attribute), 265
- \_\_slots\_\_ (owlapy.parser.OWLObjectProperty attribute), 259

\_\_slots\_\_ (owlapy.parser.OwlObjectPropertyExpression attribute), 259  
 \_\_slots\_\_ (owlapy.parser.OwlObjectSomeValuesFrom attribute), 263  
 \_\_slots\_\_ (owlapy.parser.OwlObjectUnionOf attribute), 263  
 \_\_slots\_\_ (owlapy.parser.OwlQuantifiedDataRestriction attribute), 267  
 \_\_slots\_\_ (owlapy.parser.OwlQuantifiedObjectRestriction attribute), 267  
 \_\_slots\_\_ (owlapy.providers.OwlDatatypeRestriction attribute), 281  
 \_\_slots\_\_ (owlapy.providers.OwlFacetRestriction attribute), 282  
 \_\_slots\_\_ (owlapy.providers.OwlLiteral attribute), 279  
 \_\_slots\_\_ (owlapy.render.DLSyntaxObjectRenderer attribute), 303  
 \_\_slots\_\_ (owlapy.render.IRI attribute), 285  
 \_\_slots\_\_ (owlapy.render.ManchesterOwlSyntaxOwlObjectRenderer attribute), 303  
 \_\_slots\_\_ (owlapy.render.OwlBooleanClassExpression attribute), 291  
 \_\_slots\_\_ (owlapy.render.OwlClass attribute), 291  
 \_\_slots\_\_ (owlapy.render.OwlClassExpression attribute), 291  
 \_\_slots\_\_ (owlapy.render.OwlDataAllValuesFrom attribute), 296  
 \_\_slots\_\_ (owlapy.render.OwlDataExactCardinality attribute), 297  
 \_\_slots\_\_ (owlapy.render.OwlDataHasValue attribute), 296  
 \_\_slots\_\_ (owlapy.render.OwlDataIntersectionOf attribute), 300  
 \_\_slots\_\_ (owlapy.render.OwlDataMaxCardinality attribute), 297  
 \_\_slots\_\_ (owlapy.render.OwlDataMinCardinality attribute), 297  
 \_\_slots\_\_ (owlapy.render.OwlDataSomeValuesFrom attribute), 295  
 \_\_slots\_\_ (owlapy.render.OwlDatatype attribute), 303  
 \_\_slots\_\_ (owlapy.render.OwlDatatypeRestriction attribute), 302  
 \_\_slots\_\_ (owlapy.render.OwlDataUnionOf attribute), 300  
 \_\_slots\_\_ (owlapy.render.OwlEntity attribute), 289  
 \_\_slots\_\_ (owlapy.render.OwlFacetRestriction attribute), 301  
 \_\_slots\_\_ (owlapy.render.OwlLiteral attribute), 287  
 \_\_slots\_\_ (owlapy.render.OwlNamedIndividual attribute), 286  
 \_\_slots\_\_ (owlapy.render.OwlNaryBooleanClassExpression attribute), 298  
 \_\_slots\_\_ (owlapy.render.OwlNaryDataRange attribute), 299  
 \_\_slots\_\_ (owlapy.render.OwlObject attribute), 289  
 \_\_slots\_\_ (owlapy.render.OwlObjectAllValuesFrom attribute), 293  
 \_\_slots\_\_ (owlapy.render.OwlObjectComplementOf attribute), 293  
 \_\_slots\_\_ (owlapy.render.OwlObjectExactCardinality attribute), 294  
 \_\_slots\_\_ (owlapy.render.OwlObjectHasSelf attribute), 295  
 \_\_slots\_\_ (owlapy.render.OwlObjectHasValue attribute), 301  
 \_\_slots\_\_ (owlapy.render.OwlObjectIntersectionOf attribute), 293  
 \_\_slots\_\_ (owlapy.render.OwlObjectInverseOf attribute), 289  
 \_\_slots\_\_ (owlapy.render.OwlObjectMaxCardinality attribute), 295  
 \_\_slots\_\_ (owlapy.render.OwlObjectMinCardinality attribute), 294  
 \_\_slots\_\_ (owlapy.render.OwlObjectOneOf attribute), 302  
 \_\_slots\_\_ (owlapy.render.OwlObjectSomeValuesFrom attribute), 292  
 \_\_slots\_\_ (owlapy.render.OwlObjectUnionOf attribute), 293  
 \_\_slots\_\_ (owlapy.render.OwlPropertyExpression attribute), 290  
 \_\_slots\_\_ (owlapy.render.OwlRestriction attribute), 299  
 \_\_slots\_\_ (owlapy.utils.EvaluatedDescriptionSet attribute), 327  
 \_\_slots\_\_ (owlapy.utils.HasCardinality attribute), 308  
 \_\_slots\_\_ (owlapy.utils.HasFiller attribute), 308  
 \_\_slots\_\_ (owlapy.utils.HasIRI attribute), 308  
 \_\_slots\_\_ (owlapy.utils.HasOperands attribute), 308  
 \_\_slots\_\_ (owlapy.utils.OrderedOwlObject attribute), 328  
 \_\_slots\_\_ (owlapy.utils.OwlClass attribute), 313  
 \_\_slots\_\_ (owlapy.utils.OwlClassExpression attribute), 312  
 \_\_slots\_\_ (owlapy.utils.OwlClassExpressionLengthMetric attribute), 326  
 \_\_slots\_\_ (owlapy.utils.OwlDataAllValuesFrom attribute), 318  
 \_\_slots\_\_ (owlapy.utils.OwlDataCardinalityRestriction attribute), 320  
 \_\_slots\_\_ (owlapy.utils.OwlDataExactCardinality attribute), 317  
 \_\_slots\_\_ (owlapy.utils.OwlDataHasValue attribute), 318  
 \_\_slots\_\_ (owlapy.utils.OwlDataIntersectionOf attribute), 324  
 \_\_slots\_\_ (owlapy.utils.OwlDataMaxCardinality attribute), 317  
 \_\_slots\_\_ (owlapy.utils.OwlDataMinCardinality attribute), 317  
 \_\_slots\_\_ (owlapy.utils.OwlDataProperty attribute), 312  
 \_\_slots\_\_ (owlapy.utils.OwlDataSomeValuesFrom attribute), 319  
 \_\_slots\_\_ (owlapy.utils.OwlDatatype attribute), 325  
 \_\_slots\_\_ (owlapy.utils.OwlDatatypeRestriction attribute), 322  
 \_\_slots\_\_ (owlapy.utils.OwlDataUnionOf attribute), 324  
 \_\_slots\_\_ (owlapy.utils.OwlFacetRestriction attribute), 322  
 \_\_slots\_\_ (owlapy.utils.OwlLiteral attribute), 309

- `__slots__` (*owlapy.utils.OWLNamedIndividual attribute*), 307
- `__slots__` (*owlapy.utils.OWLNaryBooleanClassExpression attribute*), 321
- `__slots__` (*owlapy.utils.OWLNaryDataRange attribute*), 324
- `__slots__` (*owlapy.utils.OWLObject attribute*), 325
- `__slots__` (*owlapy.utils.OWLObjectAllValuesFrom attribute*), 319
- `__slots__` (*owlapy.utils.OWLObjectCardinalityRestriction attribute*), 314
- `__slots__` (*owlapy.utils.OWLObjectComplementOf attribute*), 314
- `__slots__` (*owlapy.utils.OWLObjectExactCardinality attribute*), 316
- `__slots__` (*owlapy.utils.OWLObjectHasSelf attribute*), 316
- `__slots__` (*owlapy.utils.OWLObjectHasValue attribute*), 321
- `__slots__` (*owlapy.utils.OWLObjectIntersectionOf attribute*), 320
- `__slots__` (*owlapy.utils.OWLObjectInverseOf attribute*), 311
- `__slots__` (*owlapy.utils.OWLObjectMaxCardinality attribute*), 316
- `__slots__` (*owlapy.utils.OWLObjectMinCardinality attribute*), 315
- `__slots__` (*owlapy.utils.OWLObjectOneOf attribute*), 323
- `__slots__` (*owlapy.utils.OWLObjectProperty attribute*), 311
- `__slots__` (*owlapy.utils.OWLObjectSomeValuesFrom attribute*), 315
- `__slots__` (*owlapy.utils.OWLObjectUnionOf attribute*), 321
- `__slots__` (*owlapy.utils.OWLRestriction attribute*), 314
- `__slots__` (*owlapy.vocab.HasIRI attribute*), 330
- `__slots__` (*owlapy.vocab.IRI attribute*), 331
- `__slots__` (*owlapy.vocab.Namespaces attribute*), 332
- `__version__` (*in module owlapy*), 334

## A

- `AbstractHierarchy` (*class in owlapy.owl\_hierarchy*), 129
- `add_axiom()` (*owlapy.owl\_ontology\_manager.OntologyManager method*), 201
- `add_axiom()` (*owlapy.owl\_ontology\_manager.OWLOntologyManager method*), 199
- `add_axiom()` (*owlapy.owl\_reasoner.OntologyManager method*), 226
- `AddImport` (*class in owlapy.owl\_ontology\_manager*), 200
- `all_data_property_values()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 248
- `all_data_property_values()` (*owlapy.owl\_reasoner.OntologyReasoner method*), 242
- `all_data_property_values()` (*owlapy.owl\_reasoner.OWLReasonerEx method*), 238
- `annotations()` (*owlapy.owl\_axiom.OWLAxiom method*), 98
- `annotations()` (*owlapy.owl\_ontology\_manager.OWLAxiom method*), 181
- `annotations()` (*owlapy.owl\_reasoner.OWLAxiom method*), 219
- `append()` (*owlapy.converter.Owl2SparqlConverter method*), 80
- `append_triple()` (*owlapy.converter.Owl2SparqlConverter method*), 80
- `apply_change()` (*owlapy.owl\_ontology\_manager.OntologyManager method*), 200
- `apply_change()` (*owlapy.owl\_ontology\_manager.OWLOntologyManager method*), 199
- `apply_change()` (*owlapy.owl\_reasoner.OntologyManager method*), 226
- `as_anonymous_individual()` (*owlapy.owl\_annotation.OWLAnnotationObject method*), 86
- `as_index()` (*in module owlapy.utils*), 329
- `as_intersection_of_min_max()` (*owlapy.class\_expression.OWLDataExactCardinality method*), 60
- `as_intersection_of_min_max()` (*owlapy.class\_expression.OWLObjectExactCardinality method*), 58
- `as_intersection_of_min_max()` (*owlapy.class\_expression.restriction.OWLDataExactCardinality method*), 42
- `as_intersection_of_min_max()` (*owlapy.class\_expression.restriction.OWLObjectExactCardinality method*), 37
- `as_intersection_of_min_max()` (*owlapy.converter.OWLDataExactCardinality method*), 66
- `as_intersection_of_min_max()` (*owlapy.converter.OWLObjectExactCardinality method*), 71
- `as_intersection_of_min_max()` (*owlapy.owl\_ontology.OWLDataExactCardinality method*), 161
- `as_intersection_of_min_max()` (*owlapy.owl\_ontology.OWLObjectExactCardinality method*), 158
- `as_intersection_of_min_max()` (*owlapy.owl\_reasoner.OWLObjectExactCardinality method*), 213
- `as_intersection_of_min_max()` (*owlapy.parser.OWLDataExactCardinality method*), 268
- `as_intersection_of_min_max()` (*owlapy.parser.OWLObjectExactCardinality method*), 267
- `as_intersection_of_min_max()` (*owlapy.render.OWLDataExactCardinality method*), 297
- `as_intersection_of_min_max()` (*owlapy.render.OWLObjectExactCardinality method*), 294
- `as_intersection_of_min_max()` (*owlapy.utils.OWLDataExactCardinality method*), 317
- `as_intersection_of_min_max()` (*owlapy.utils.OWLObjectExactCardinality method*), 316
- `as_iri()` (*owlapy.class\_expression.owl\_class.IRI method*), 24
- `as_iri()` (*owlapy.iri.IRI method*), 82
- `as_iri()` (*owlapy.owl\_annotation.OWLAnnotationObject method*), 86
- `as_iri()` (*owlapy.owl\_axiom.IRI method*), 95
- `as_iri()` (*owlapy.owl\_datatype.IRI method*), 119
- `as_iri()` (*owlapy.owl\_individual.IRI method*), 134
- `as_iri()` (*owlapy.owl\_ontology\_manager.IRI method*), 176
- `as_iri()` (*owlapy.owl\_ontology.IRI method*), 153
- `as_iri()` (*owlapy.owl\_property.IRI method*), 203



`as_iri()` (*owlapy.owl\_reasoner.IRI method*), 218  
`as_iri()` (*owlapy.parser.IRI method*), 256  
`as_iri()` (*owlapy.render.IRI method*), 286  
`as_iri()` (*owlapy.vocab.IRI method*), 331  
`as_literal()` (*owlapy.class\_expression.restriction.OWLLiteral method*), 33  
`as_literal()` (*owlapy.converter.OWLLiteral method*), 76  
`as_literal()` (*owlapy.iri.OWLAnnotationValue method*), 81  
`as_literal()` (*owlapy.owl\_annotation.OWLAnnotationValue method*), 86  
`as_literal()` (*owlapy.owl\_axiom.OWLAnnotationValue method*), 96  
`as_literal()` (*owlapy.owl\_axiom.OWLLiteral method*), 97  
`as_literal()` (*owlapy.owl\_literal.OWLAnnotationValue method*), 137  
`as_literal()` (*owlapy.owl\_literal.OWLLiteral method*), 141  
`as_literal()` (*owlapy.owl\_ontology.OWLLiteral method*), 152  
`as_literal()` (*owlapy.owl\_reasoner.OWLLiteral method*), 231  
`as_literal()` (*owlapy.parser.OWLLiteral method*), 259  
`as_literal()` (*owlapy.providers.OWLLiteral method*), 281  
`as_literal()` (*owlapy.render.OWLLiteral method*), 288  
`as_literal()` (*owlapy.utils.OWLLiteral method*), 310  
`as_object_union_of()` (*owlapy.class\_expression.OWLObjectOneOf method*), 61  
`as_object_union_of()` (*owlapy.class\_expression.restriction.OWLObjectOneOf method*), 40  
`as_object_union_of()` (*owlapy.converter.OWLObjectOneOf method*), 65  
`as_object_union_of()` (*owlapy.owl\_ontology\_manager.OWLObjectOneOf method*), 179  
`as_object_union_of()` (*owlapy.owl\_ontology.OWLObjectOneOf method*), 158  
`as_object_union_of()` (*owlapy.owl\_reasoner.OWLObjectOneOf method*), 212  
`as_object_union_of()` (*owlapy.parser.OWLObjectOneOf method*), 265  
`as_object_union_of()` (*owlapy.render.OWLObjectOneOf method*), 302  
`as_object_union_of()` (*owlapy.utils.OWLObjectOneOf method*), 323  
`as_pairwise_axioms()` (*owlapy.owl\_axiom.OWLNaryAxiom method*), 100  
`as_pairwise_axioms()` (*owlapy.owl\_axiom.OWLNaryClassAxiom method*), 101  
`as_pairwise_axioms()` (*owlapy.owl\_axiom.OWLNaryIndividualAxiom method*), 102  
`as_pairwise_axioms()` (*owlapy.owl\_axiom.OWLNaryPropertyAxiom method*), 103  
`as_pairwise_axioms()` (*owlapy.owl\_ontology\_manager.OWLNaryIndividualAxiom method*), 191  
`as_pairwise_axioms()` (*owlapy.owl\_ontology\_manager.OWLNaryPropertyAxiom method*), 190  
`as_query()` (*owlapy.converter.Owl2SparqlConverter method*), 80  
`as_some_values_from()` (*owlapy.class\_expression.OWLDataHasValue method*), 59  
`as_some_values_from()` (*owlapy.class\_expression.OWLObjectHasValue method*), 55  
`as_some_values_from()` (*owlapy.class\_expression.restriction.OWLDataHasValue method*), 43  
`as_some_values_from()` (*owlapy.class\_expression.restriction.OWLObjectHasValue method*), 39  
`as_some_values_from()` (*owlapy.converter.OWLDataHasValue method*), 73  
`as_some_values_from()` (*owlapy.converter.OWLObjectHasValue method*), 64  
`as_some_values_from()` (*owlapy.owl\_ontology\_manager.OWLDataHasValue method*), 178  
`as_some_values_from()` (*owlapy.owl\_ontology\_manager.OWLObjectHasValue method*), 180  
`as_some_values_from()` (*owlapy.owl\_ontology.OWLDataHasValue method*), 161  
`as_some_values_from()` (*owlapy.owl\_ontology.OWLObjectHasValue method*), 159  
`as_some_values_from()` (*owlapy.owl\_reasoner.OWLDataHasValue method*), 216  
`as_some_values_from()` (*owlapy.owl\_reasoner.OWLObjectHasValue method*), 212  
`as_some_values_from()` (*owlapy.parser.OWLDataHasValue method*), 270  
`as_some_values_from()` (*owlapy.parser.OWLObjectHasValue method*), 269  
`as_some_values_from()` (*owlapy.render.OWLDataHasValue method*), 297  
`as_some_values_from()` (*owlapy.render.OWLObjectHasValue method*), 301  
`as_some_values_from()` (*owlapy.utils.OWLDataHasValue method*), 318  
`as_some_values_from()` (*owlapy.utils.OWLObjectHasValue method*), 322  
`as_str()` (*owlapy.class\_expression.owl\_class.IRI method*), 24  
`as_str()` (*owlapy.iri.IRI method*), 82  
`as_str()` (*owlapy.owl\_axiom.IRI method*), 95  
`as_str()` (*owlapy.owl\_datatype.IRI method*), 119  
`as_str()` (*owlapy.owl\_individual.IRI method*), 134  
`as_str()` (*owlapy.owl\_ontology\_manager.IRI method*), 176  
`as_str()` (*owlapy.owl\_ontology.IRI method*), 153  
`as_str()` (*owlapy.owl\_property.IRI method*), 203  
`as_str()` (*owlapy.owl\_reasoner.IRI method*), 218  
`as_str()` (*owlapy.parser.IRI method*), 256  
`as_str()` (*owlapy.render.IRI method*), 286  
`as_str()` (*owlapy.vocab.IRI method*), 331

## B

`BaseReasoner` (class in *owlapy.owl\_reasoner*), 238

best () (*owlapy.utils.EvaluatedDescriptionSet method*), 328  
 best\_quality\_value () (*owlapy.utils.EvaluatedDescriptionSet method*), 328  
 BOOLEAN (*owlapy.owl\_literal.XSDVocabulary attribute*), 138  
 BOOLEAN (*owlapy.vocab.XSDVocabulary attribute*), 333  
 BooleanOWLDatatype (*in module owlapy.owl\_literal*), 141  
 BooleanOWLDatatype (*in module owlapy.owl\_ontology*), 150  
 BooleanOWLDatatype (*in module owlapy.parser*), 257

## C

cache\_clear () (*owlapy.owl\_reasoner.LRUCache method*), 232  
 cache\_clear () (*owlapy.utils.LRUCache method*), 330  
 cache\_info () (*owlapy.owl\_reasoner.LRUCache method*), 232  
 cache\_info () (*owlapy.utils.LRUCache method*), 330  
 ce (*owlapy.converter.Owl2SparqlConverter attribute*), 79  
 children () (*owlapy.owl\_hierarchy.AbstractHierarchy method*), 130  
 class\_expressions () (*owlapy.owl\_axiom.OwLNaryClassAxiom method*), 101  
 class\_length (*owlapy.utils.OwLClassExpressionLengthMetric attribute*), 326  
 classes\_in\_signature () (*owlapy.owl\_ontology\_manager.Ontology method*), 194  
 classes\_in\_signature () (*owlapy.owl\_ontology\_manager.OwLOntology method*), 192  
 classes\_in\_signature () (*owlapy.owl\_ontology.Ontology method*), 170  
 classes\_in\_signature () (*owlapy.owl\_ontology.OwLOntology method*), 168  
 classes\_in\_signature () (*owlapy.owl\_reasoner.Ontology method*), 223  
 classes\_in\_signature () (*owlapy.owl\_reasoner.OwLOntology method*), 221  
 ClassHierarchy (*class in owlapy.owl\_hierarchy*), 131  
 clean () (*owlapy.utils.EvaluatedDescriptionSet method*), 327  
 cnt (*owlapy.converter.Owl2SparqlConverter attribute*), 79  
 combine\_nary\_expressions () (*in module owlapy.utils*), 329  
 ConceptOperandSorter (*class in owlapy.utils*), 328  
 contains\_named\_equivalent\_class () (*owlapy.owl\_axiom.OwLEquivalentClassesAxiom method*), 101  
 contains\_named\_equivalent\_class () (*owlapy.owl\_ontology\_manager.OwLEquivalentClassesAxiom method*), 182  
 contains\_named\_equivalent\_class () (*owlapy.owl\_ontology.OwLEquivalentClassesAxiom method*), 146  
 contains\_owl\_nothing () (*owlapy.owl\_axiom.OwLEquivalentClassesAxiom method*), 101  
 contains\_owl\_nothing () (*owlapy.owl\_ontology\_manager.OwLEquivalentClassesAxiom method*), 182  
 contains\_owl\_nothing () (*owlapy.owl\_ontology.OwLEquivalentClassesAxiom method*), 147  
 contains\_owl\_thing () (*owlapy.owl\_axiom.OwLEquivalentClassesAxiom method*), 101  
 contains\_owl\_thing () (*owlapy.owl\_ontology\_manager.OwLEquivalentClassesAxiom method*), 182  
 contains\_owl\_thing () (*owlapy.owl\_ontology.OwLEquivalentClassesAxiom method*), 147  
 convert () (*owlapy.converter.Owl2SparqlConverter method*), 79  
 converter (*in module owlapy.converter*), 80  
 create () (*owlapy.class\_expression.owl\_class.IRI static method*), 24  
 create () (*owlapy.iri.IRI static method*), 81  
 create () (*owlapy.owl\_axiom.IRI static method*), 94  
 create () (*owlapy.owl\_datatype.IRI static method*), 118  
 create () (*owlapy.owl\_individual.IRI static method*), 133  
 create () (*owlapy.owl\_ontology\_manager.IRI static method*), 175  
 create () (*owlapy.owl\_ontology.IRI static method*), 153  
 create () (*owlapy.owl\_property.IRI static method*), 203  
 create () (*owlapy.owl\_reasoner.IRI static method*), 218  
 create () (*owlapy.parser.IRI static method*), 255  
 create () (*owlapy.render.IRI static method*), 285  
 create () (*owlapy.vocab.IRI static method*), 331  
 create\_ontology () (*owlapy.owl\_ontology\_manager.OntologyManager method*), 200  
 create\_ontology () (*owlapy.owl\_ontology\_manager.OwLOntologyManager method*), 198  
 create\_ontology () (*owlapy.owl\_reasoner.OntologyManager method*), 225  
 current\_variable (*owlapy.converter.Owl2SparqlConverter property*), 80

## D

data\_all\_values\_length (*owlapy.utils.OwLClassExpressionLengthMetric attribute*), 327  
 data\_cardinality\_length (*owlapy.utils.OwLClassExpressionLengthMetric attribute*), 327  
 data\_complement\_length (*owlapy.utils.OwLClassExpressionLengthMetric attribute*), 327  
 data\_has\_value\_length (*owlapy.utils.OwLClassExpressionLengthMetric attribute*), 327  
 data\_intersection\_length (*owlapy.utils.OwLClassExpressionLengthMetric attribute*), 327  
 data\_one\_of\_length (*owlapy.utils.OwLClassExpressionLengthMetric attribute*), 327  
 data\_properties\_in\_signature () (*owlapy.owl\_ontology\_manager.Ontology method*), 194  
 data\_properties\_in\_signature () (*owlapy.owl\_ontology\_manager.OwLOntology method*), 192  
 data\_properties\_in\_signature () (*owlapy.owl\_ontology.Ontology method*), 170  
 data\_properties\_in\_signature () (*owlapy.owl\_ontology.OwLOntology method*), 168

`data_properties_in_signature()` (*owlapy.owl\_reasoner.Ontology method*), 223  
`data_properties_in_signature()` (*owlapy.owl\_reasoner.OWLOntology method*), 221  
`data_property_domain_axioms()` (*owlapy.owl\_ontology\_manager.Ontology method*), 195  
`data_property_domain_axioms()` (*owlapy.owl\_ontology\_manager.OWLOntology method*), 193  
`data_property_domain_axioms()` (*owlapy.owl\_ontology.Ontology method*), 171  
`data_property_domain_axioms()` (*owlapy.owl\_ontology.OWLOntology method*), 169  
`data_property_domain_axioms()` (*owlapy.owl\_reasoner.Ontology method*), 224  
`data_property_domain_axioms()` (*owlapy.owl\_reasoner.OWLOntology method*), 222  
`data_property_domains()` (*owlapy.owl\_hierarchy.OWLReasoner method*), 123  
`data_property_domains()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 246  
`data_property_domains()` (*owlapy.owl\_reasoner.OntologyReasoner method*), 239  
`data_property_domains()` (*owlapy.owl\_reasoner.OWLReasoner method*), 232  
`data_property_length` (*owlapy.utils.OWLClassExpressionLengthMetric attribute*), 327  
`data_property_range_axioms()` (*owlapy.owl\_ontology\_manager.Ontology method*), 195  
`data_property_range_axioms()` (*owlapy.owl\_ontology\_manager.OWLOntology method*), 193  
`data_property_range_axioms()` (*owlapy.owl\_ontology.Ontology method*), 171  
`data_property_range_axioms()` (*owlapy.owl\_ontology.OWLOntology method*), 169  
`data_property_range_axioms()` (*owlapy.owl\_reasoner.Ontology method*), 224  
`data_property_range_axioms()` (*owlapy.owl\_reasoner.OWLOntology method*), 222  
`data_property_ranges()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 246  
`data_property_ranges()` (*owlapy.owl\_reasoner.OWLReasonerEx method*), 238  
`data_property_values()` (*owlapy.owl\_hierarchy.OWLReasoner method*), 126  
`data_property_values()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 248  
`data_property_values()` (*owlapy.owl\_reasoner.OntologyReasoner method*), 241  
`data_property_values()` (*owlapy.owl\_reasoner.OWLReasoner method*), 234  
`data_some_values_length` (*owlapy.utils.OWLClassExpressionLengthMetric attribute*), 326  
`data_union_length` (*owlapy.utils.OWLClassExpressionLengthMetric attribute*), 327  
`datatype_length` (*owlapy.utils.OWLClassExpressionLengthMetric attribute*), 327  
`DatatypePropertyHierarchy` (class in *owlapy.owl\_hierarchy*), 132  
`DATE` (*owlapy.owl\_literal.XSDVocabulary attribute*), 138  
`DATE` (*owlapy.vocab.XSDVocabulary attribute*), 333  
`DATE_TIME` (*owlapy.owl\_literal.XSDVocabulary attribute*), 138  
`DATE_TIME` (*owlapy.vocab.XSDVocabulary attribute*), 333  
`DATE_TIME_STAMP` (*owlapy.owl\_literal.XSDVocabulary attribute*), 138  
`DATE_TIME_STAMP` (*owlapy.vocab.XSDVocabulary attribute*), 333  
`DateOWLDatatype` (in module *owlapy.owl\_literal*), 141  
`DateOWLDatatype` (in module *owlapy.owl\_ontology*), 151  
`DateOWLDatatype` (in module *owlapy.parser*), 257  
`DateTimeOWLDatatype` (in module *owlapy.owl\_literal*), 141  
`DateTimeOWLDatatype` (in module *owlapy.owl\_ontology*), 151  
`DateTimeOWLDatatype` (in module *owlapy.parser*), 257  
`DECIMAL` (*owlapy.owl\_literal.XSDVocabulary attribute*), 138  
`DECIMAL` (*owlapy.vocab.XSDVocabulary attribute*), 333  
`different_individuals()` (*owlapy.owl\_hierarchy.OWLReasoner method*), 125  
`different_individuals()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 248  
`different_individuals()` (*owlapy.owl\_reasoner.OntologyReasoner method*), 241  
`different_individuals()` (*owlapy.owl\_reasoner.OWLReasoner method*), 233  
`disjoint_classes()` (*owlapy.owl\_hierarchy.OWLReasoner method*), 125  
`disjoint_classes()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 247  
`disjoint_classes()` (*owlapy.owl\_reasoner.OntologyReasoner method*), 241  
`disjoint_classes()` (*owlapy.owl\_reasoner.OWLReasoner method*), 233  
`disjoint_data_properties()` (*owlapy.owl\_hierarchy.OWLReasoner method*), 127  
`disjoint_data_properties()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 251  
`disjoint_data_properties()` (*owlapy.owl\_reasoner.OntologyReasoner method*), 244  
`disjoint_data_properties()` (*owlapy.owl\_reasoner.OWLReasoner method*), 236  
`disjoint_object_properties()` (*owlapy.owl\_hierarchy.OWLReasoner method*), 127  
`disjoint_object_properties()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 250  
`disjoint_object_properties()` (*owlapy.owl\_reasoner.OntologyReasoner method*), 244  
`disjoint_object_properties()` (*owlapy.owl\_reasoner.OWLReasoner method*), 235  
`DL_GRAMMAR` (in module *owlapy.parser*), 276  
`dl_to_owl_expression()` (in module *owlapy*), 334  
`dl_to_owl_expression()` (in module *owlapy.parser*), 278  
`DLparser` (in module *owlapy.parser*), 278  
`DLrenderer` (in module *owlapy.render*), 304  
`DLSyntaxObjectRenderer` (class in *owlapy.render*), 303  
`DLSyntaxParser` (class in *owlapy.parser*), 276  
`DOUBLE` (*owlapy.owl\_literal.XSDVocabulary attribute*), 138  
`DOUBLE` (*owlapy.vocab.XSDVocabulary attribute*), 333

DoubleOWLDatatype (in module owlapy.owl\_literal), 141  
 DoubleOWLDatatype (in module owlapy.owl\_ontology), 150  
 DoubleOWLDatatype (in module owlapy.parser), 257  
 download\_external\_files() (in module owlapy.static\_funcs), 304  
 DURATION (owlapy.owl\_literal.XSDVocabulary attribute), 138  
 DURATION (owlapy.vocab.XSDVocabulary attribute), 333  
 DurationOWLDatatype (in module owlapy.owl\_literal), 141  
 DurationOWLDatatype (in module owlapy.owl\_ontology), 151  
 DurationOWLDatatype (in module owlapy.parser), 257

## E

equivalent\_classes() (owlapy.owl\_hierarchy.OWLReasoner method), 124  
 equivalent\_classes() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 247  
 equivalent\_classes() (owlapy.owl\_reasoner.OntologyReasoner method), 240  
 equivalent\_classes() (owlapy.owl\_reasoner.OWLReasoner method), 233  
 equivalent\_classes\_axioms() (owlapy.owl\_ontology\_manager.Ontology method), 194  
 equivalent\_classes\_axioms() (owlapy.owl\_ontology\_manager.OWLOntology method), 193  
 equivalent\_classes\_axioms() (owlapy.owl\_ontology.Ontology method), 170  
 equivalent\_classes\_axioms() (owlapy.owl\_ontology.OWLOntology method), 169  
 equivalent\_classes\_axioms() (owlapy.owl\_reasoner.Ontology method), 223  
 equivalent\_classes\_axioms() (owlapy.owl\_reasoner.OWLOntology method), 222  
 equivalent\_data\_properties() (owlapy.owl\_hierarchy.OWLReasoner method), 125  
 equivalent\_data\_properties() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 250  
 equivalent\_data\_properties() (owlapy.owl\_reasoner.OntologyReasoner method), 243  
 equivalent\_data\_properties() (owlapy.owl\_reasoner.OWLReasoner method), 234  
 equivalent\_object\_properties() (owlapy.owl\_hierarchy.OWLReasoner method), 125  
 equivalent\_object\_properties() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 250  
 equivalent\_object\_properties() (owlapy.owl\_reasoner.OntologyReasoner method), 243  
 equivalent\_object\_properties() (owlapy.owl\_reasoner.OWLReasoner method), 234  
 EvaluatedDescriptionSet (class in owlapy.utils), 327

## F

FastInstanceCheckerReasoner (class in owlapy.owl\_reasoner), 245  
 FLOAT (owlapy.owl\_literal.XSDVocabulary attribute), 138  
 FLOAT (owlapy.vocab.XSDVocabulary attribute), 333  
 flush() (owlapy.owl\_hierarchy.OWLReasoner method), 126  
 flush() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 249  
 flush() (owlapy.owl\_reasoner.OntologyReasoner method), 242  
 flush() (owlapy.owl\_reasoner.OWLReasoner method), 235  
 for\_all\_de\_morgan (owlapy.converter.Owl2SparqlConverter attribute), 79  
 forAll() (owlapy.converter.Owl2SparqlConverter method), 80  
 forAllDeMorgan() (owlapy.converter.Owl2SparqlConverter method), 80  
 FRACTION\_DIGITS (owlapy.class\_expression.OWLFacet attribute), 56  
 FRACTION\_DIGITS (owlapy.class\_expression.restriction.OWLFacet attribute), 34  
 FRACTION\_DIGITS (owlapy.converter.OWLFacet attribute), 78  
 FRACTION\_DIGITS (owlapy.owl\_ontology.OWLFacet attribute), 167  
 FRACTION\_DIGITS (owlapy.parser.OWLFacet attribute), 262  
 FRACTION\_DIGITS (owlapy.providers.OWLFacet attribute), 282  
 FRACTION\_DIGITS (owlapy.render.OWLFacet attribute), 299  
 FRACTION\_DIGITS (owlapy.vocab.OWLFacet attribute), 333  
 from\_str() (owlapy.class\_expression.OWLFacet static method), 56  
 from\_str() (owlapy.class\_expression.restriction.OWLFacet static method), 34  
 from\_str() (owlapy.converter.OWLFacet static method), 77  
 from\_str() (owlapy.owl\_ontology.OWLFacet static method), 167  
 from\_str() (owlapy.parser.OWLFacet static method), 261  
 from\_str() (owlapy.providers.OWLFacet static method), 282  
 from\_str() (owlapy.render.OWLFacet static method), 299  
 from\_str() (owlapy.vocab.OWLFacet static method), 333  
 FromOwlready2 (class in owlapy.owl\_ontology), 172

## G

general\_class\_axioms() (owlapy.owl\_ontology\_manager.Ontology method), 195  
 general\_class\_axioms() (owlapy.owl\_ontology\_manager.OWLOntology method), 193  
 general\_class\_axioms() (owlapy.owl\_ontology.Ontology method), 171  
 general\_class\_axioms() (owlapy.owl\_ontology.OWLOntology method), 169  
 general\_class\_axioms() (owlapy.owl\_reasoner.Ontology method), 224



`general_class_axioms()` (*owlapy.owl\_reasoner.OWL Ontology method*), 222  
`generic_visit()` (*owlapy.parser.DLSyntaxParser method*), 278  
`generic_visit()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`get_bottom_entity()` (*owlapy.owl\_hierarchy.AbstractHierarchy class method*), 129  
`get_bottom_entity()` (*owlapy.owl\_hierarchy.ClassHierarchy class method*), 131  
`get_bottom_entity()` (*owlapy.owl\_hierarchy.DatatypePropertyHierarchy class method*), 132  
`get_bottom_entity()` (*owlapy.owl\_hierarchy.ObjectPropertyHierarchy class method*), 131  
`get_cardinality()` (*owlapy.class\_expression.OWLCardinalityRestriction method*), 52  
`get_cardinality()` (*owlapy.class\_expression.restriction.HasCardinality method*), 28  
`get_cardinality()` (*owlapy.class\_expression.restriction.OWLCardinalityRestriction method*), 36  
`get_cardinality()` (*owlapy.meta\_classes.HasCardinality method*), 84  
`get_cardinality()` (*owlapy.utils.HasCardinality method*), 308  
`get_class_expression()` (*owlapy.owl\_axiom.OWLClassAssertionAxiom method*), 105  
`get_class_expression()` (*owlapy.owl\_axiom.OWLHasKeyAxiom method*), 100  
`get_class_expression()` (*owlapy.owl\_ontology\_manager.OWLClassAssertionAxiom method*), 185  
`get_class_expressions()` (*owlapy.owl\_axiom.OWLDisjointUnionAxiom method*), 105  
`get_class_expressions()` (*owlapy.owl\_ontology\_manager.OWLDisjointUnionAxiom method*), 183  
`get_class_nnf()` (*owlapy.utils.NNF method*), 328  
`get_data_range()` (*owlapy.owl\_data\_ranges.OWLDataComplementOf method*), 117  
`get_data_range()` (*owlapy.owl\_ontology.OWLDataComplementOf method*), 149  
`get_data_range()` (*owlapy.owl\_reasoner.OWLDataComplementOf method*), 220  
`get_data_range()` (*owlapy.parser.OWLDataComplementOf method*), 274  
`get_data_range()` (*owlapy.render.OWLDataComplementOf method*), 300  
`get_data_range()` (*owlapy.utils.OWLDataComplementOf method*), 323  
`get_datatype()` (*owlapy.owl\_axiom.OWLDatatypeDefinitionAxiom method*), 99  
`get_datatype()` (*owlapy.class\_expression.OWLDatatypeRestriction method*), 56  
`get_datatype()` (*owlapy.class\_expression.restriction.OWLDatatypeRestriction method*), 44  
`get_datatype()` (*owlapy.class\_expression.restriction.OWLLiteral method*), 33  
`get_datatype()` (*owlapy.converter.OWLDatatypeRestriction method*), 65  
`get_datatype()` (*owlapy.converter.OWLLiteral method*), 76  
`get_datatype()` (*owlapy.owl\_axiom.OWLDatatypeDefinitionAxiom method*), 99  
`get_datatype()` (*owlapy.owl\_axiom.OWLLiteral method*), 97  
`get_datatype()` (*owlapy.owl\_literal.OWLLiteral method*), 141  
`get_datatype()` (*owlapy.owl\_ontology.OWLDatatypeRestriction method*), 162  
`get_datatype()` (*owlapy.owl\_ontology.OWLLiteral method*), 152  
`get_datatype()` (*owlapy.owl\_reasoner.OWLDatatypeRestriction method*), 215  
`get_datatype()` (*owlapy.owl\_reasoner.OWLLiteral method*), 231  
`get_datatype()` (*owlapy.parser.OWLDatatypeRestriction method*), 271  
`get_datatype()` (*owlapy.parser.OWLLiteral method*), 259  
`get_datatype()` (*owlapy.providers.OWLDatatypeRestriction method*), 281  
`get_datatype()` (*owlapy.providers.OWLLiteral method*), 281  
`get_datatype()` (*owlapy.render.OWLDatatypeRestriction method*), 302  
`get_datatype()` (*owlapy.render.OWLLiteral method*), 288  
`get_datatype()` (*owlapy.utils.OWLDatatypeRestriction method*), 322  
`get_datatype()` (*owlapy.utils.OWLLiteral method*), 310  
`get_default()` (*owlapy.utils.OWLClassExpressionLengthMetric static method*), 327  
`get_default_document_iri()` (*owlapy.owl\_ontology.OWL OntologyID method*), 168  
`get_domain()` (*owlapy.owl\_axiom.OWLAnnotationPropertyDomainAxiom method*), 108  
`get_domain()` (*owlapy.owl\_axiom.OWLPropertyDomainAxiom method*), 113  
`get_domain()` (*owlapy.owl\_ontology\_manager.OWLPropertyDomainAxiom method*), 186  
`get_entity()` (*owlapy.owl\_axiom.OWLDeclarationAxiom method*), 99  
`get_entity()` (*owlapy.owl\_ontology\_manager.OWLDeclarationAxiom method*), 185  
`get_expression_length()` (*in module owlapy.utils*), 327  
`get_facet()` (*owlapy.class\_expression.OWLFacetRestriction method*), 57  
`get_facet()` (*owlapy.class\_expression.restriction.OWLFacetRestriction method*), 44  
`get_facet()` (*owlapy.owl\_ontology.OWLFacetRestriction method*), 164  
`get_facet()` (*owlapy.owl\_reasoner.OWLFacetRestriction method*), 215  
`get_facet()` (*owlapy.parser.OWLFacetRestriction method*), 267  
`get_facet()` (*owlapy.providers.OWLFacetRestriction method*), 282  
`get_facet()` (*owlapy.render.OWLFacetRestriction method*), 301  
`get_facet()` (*owlapy.utils.OWLFacetRestriction method*), 322  
`get_facet_restrictions()` (*owlapy.class\_expression.OWLDatatypeRestriction method*), 56  
`get_facet_restrictions()` (*owlapy.class\_expression.restriction.OWLDatatypeRestriction method*), 44  
`get_facet_restrictions()` (*owlapy.converter.OWLDatatypeRestriction method*), 65  
`get_facet_restrictions()` (*owlapy.owl\_ontology.OWLDatatypeRestriction method*), 162  
`get_facet_restrictions()` (*owlapy.owl\_reasoner.OWLDatatypeRestriction method*), 215  
`get_facet_restrictions()` (*owlapy.parser.OWLDatatypeRestriction method*), 271  
`get_facet_restrictions()` (*owlapy.providers.OWLDatatypeRestriction method*), 281

`get_facet_restrictions()` (*owlapy.render.OWLDatatypeRestriction method*), 302  
`get_facet_restrictions()` (*owlapy.utils.OWLDatatypeRestriction method*), 322  
`get_facet_value()` (*owlapy.class\_expression.OWLFacetRestriction method*), 57  
`get_facet_value()` (*owlapy.class\_expression.restriction.OWLFacetRestriction method*), 45  
`get_facet_value()` (*owlapy.owl\_ontology.OWLFacetRestriction method*), 164  
`get_facet_value()` (*owlapy.owl\_reasoner.OWLFacetRestriction method*), 215  
`get_facet_value()` (*owlapy.parser.OWLFacetRestriction method*), 267  
`get_facet_value()` (*owlapy.providers.OWLFacetRestriction method*), 282  
`get_facet_value()` (*owlapy.render.OWLFacetRestriction method*), 301  
`get_facet_value()` (*owlapy.utils.OWLFacetRestriction method*), 322  
`get_filler()` (*owlapy.class\_expression.OWLCardinalityRestriction method*), 52  
`get_filler()` (*owlapy.class\_expression.OWLHasValueRestriction method*), 51  
`get_filler()` (*owlapy.class\_expression.OWLQuantifiedDataRestriction method*), 54  
`get_filler()` (*owlapy.class\_expression.OWLQuantifiedObjectRestriction method*), 51  
`get_filler()` (*owlapy.class\_expression.restriction.HasFiller method*), 28  
`get_filler()` (*owlapy.class\_expression.restriction.OWLCardinalityRestriction method*), 36  
`get_filler()` (*owlapy.class\_expression.restriction.OWLHasValueRestriction method*), 35  
`get_filler()` (*owlapy.class\_expression.restriction.OWLQuantifiedDataRestriction method*), 40  
`get_filler()` (*owlapy.class\_expression.restriction.OWLQuantifiedObjectRestriction method*), 36  
`get_filler()` (*owlapy.meta\_classes.HasFiller method*), 84  
`get_filler()` (*owlapy.owl\_ontology\_manager.OWLQuantifiedDataRestriction method*), 178  
`get_filler()` (*owlapy.owl\_ontology\_manager.OWLQuantifiedObjectRestriction method*), 181  
`get_filler()` (*owlapy.parser.OWLQuantifiedDataRestriction method*), 267  
`get_filler()` (*owlapy.parser.OWLQuantifiedObjectRestriction method*), 267  
`get_filler()` (*owlapy.utils.HasFiller method*), 308  
`get_first_property()` (*owlapy.owl\_axiom.OWLInverseObjectPropertiesAxiom method*), 104  
`get_first_property()` (*owlapy.owl\_ontology\_manager.OWLInverseObjectPropertiesAxiom method*), 190  
`get_import_declaration()` (*owlapy.owl\_ontology\_manager.AddImport method*), 200  
`get_individual()` (*owlapy.owl\_axiom.OWLClassAssertionAxiom method*), 105  
`get_individual()` (*owlapy.owl\_ontology\_manager.OWLClassAssertionAxiom method*), 185  
`get_inverse()` (*owlapy.owl\_ontology\_manager.OWLObjectInverseOf method*), 197  
`get_inverse()` (*owlapy.owl\_ontology.OWLObjectInverseOf method*), 166  
`get_inverse()` (*owlapy.owl\_property.OWLObjectInverseOf method*), 206  
`get_inverse()` (*owlapy.owl\_reasoner.OWLObjectInverseOf method*), 228  
`get_inverse()` (*owlapy.render.OWLObjectInverseOf method*), 289  
`get_inverse()` (*owlapy.utils.OWLObjectInverseOf method*), 311  
`get_inverse_property()` (*owlapy.class\_expression.restriction.OWLObjectPropertyExpression method*), 30  
`get_inverse_property()` (*owlapy.converter.OWLObjectProperty method*), 76  
`get_inverse_property()` (*owlapy.owl\_axiom.OWLObjectPropertyExpression method*), 90  
`get_inverse_property()` (*owlapy.owl\_hierarchy.OWLObjectProperty method*), 123  
`get_inverse_property()` (*owlapy.owl\_literal.OWLObjectProperty method*), 138  
`get_inverse_property()` (*owlapy.owl\_ontology\_manager.OWLObjectInverseOf method*), 197  
`get_inverse_property()` (*owlapy.owl\_ontology\_manager.OWLObjectProperty method*), 197  
`get_inverse_property()` (*owlapy.owl\_ontology.OWLObjectInverseOf method*), 166  
`get_inverse_property()` (*owlapy.owl\_ontology.OWLObjectProperty method*), 165  
`get_inverse_property()` (*owlapy.owl\_ontology.OWLObjectPropertyExpression method*), 166  
`get_inverse_property()` (*owlapy.owl\_property.OWLObjectInverseOf method*), 206  
`get_inverse_property()` (*owlapy.owl\_property.OWLObjectProperty method*), 205  
`get_inverse_property()` (*owlapy.owl\_property.OWLObjectPropertyExpression method*), 204  
`get_inverse_property()` (*owlapy.owl\_reasoner.OWLObjectInverseOf method*), 228  
`get_inverse_property()` (*owlapy.owl\_reasoner.OWLObjectProperty method*), 227  
`get_inverse_property()` (*owlapy.owl\_reasoner.OWLObjectPropertyExpression method*), 226  
`get_inverse_property()` (*owlapy.parser.OWLObjectProperty method*), 260  
`get_inverse_property()` (*owlapy.parser.OWLObjectPropertyExpression method*), 259  
`get_inverse_property()` (*owlapy.render.OWLObjectInverseOf method*), 290  
`get_inverse_property()` (*owlapy.utils.OWLObjectInverseOf method*), 311  
`get_inverse_property()` (*owlapy.utils.OWLObjectProperty method*), 311  
`get_literal()` (*owlapy.class\_expression.restriction.OWLLiteral method*), 31  
`get_literal()` (*owlapy.converter.OWLLiteral method*), 74  
`get_literal()` (*owlapy.owl\_axiom.OWLLiteral method*), 96  
`get_literal()` (*owlapy.owl\_literal.OWLLiteral method*), 139  
`get_literal()` (*owlapy.owl\_ontology.OWLLiteral method*), 151  
`get_literal()` (*owlapy.owl\_reasoner.OWLLiteral method*), 230  
`get_literal()` (*owlapy.parser.OWLLiteral method*), 257  
`get_literal()` (*owlapy.providers.OWLLiteral method*), 279  
`get_literal()` (*owlapy.render.OWLLiteral method*), 287  
`get_literal()` (*owlapy.utils.OWLLiteral method*), 309  
`get_named_property()` (*owlapy.class\_expression.restriction.OWLObjectPropertyExpression method*), 31

`get_named_property()` (*owlapy.converter.OWLObjectProperty method*), 76  
`get_named_property()` (*owlapy.owl\_axiom.OWLObjectPropertyExpression method*), 90  
`get_named_property()` (*owlapy.owl\_hierarchy.OWLObjectProperty method*), 122  
`get_named_property()` (*owlapy.owl\_literal.OWLObjectProperty method*), 138  
`get_named_property()` (*owlapy.owl\_ontology\_manager.OWLObjectInverseOf method*), 197  
`get_named_property()` (*owlapy.owl\_ontology\_manager.OWLObjectProperty method*), 197  
`get_named_property()` (*owlapy.owl\_ontology.OWLObjectInverseOf method*), 166  
`get_named_property()` (*owlapy.owl\_ontology.OWLObjectProperty method*), 164  
`get_named_property()` (*owlapy.owl\_ontology.OWLObjectPropertyExpression method*), 166  
`get_named_property()` (*owlapy.owl\_property.OWLObjectInverseOf method*), 206  
`get_named_property()` (*owlapy.owl\_property.OWLObjectProperty method*), 205  
`get_named_property()` (*owlapy.owl\_property.OWLObjectPropertyExpression method*), 204  
`get_named_property()` (*owlapy.owl\_reasoner.OWLObjectInverseOf method*), 228  
`get_named_property()` (*owlapy.owl\_reasoner.OWLObjectProperty method*), 227  
`get_named_property()` (*owlapy.owl\_reasoner.OWLObjectPropertyExpression method*), 226  
`get_named_property()` (*owlapy.parser.OWLObjectProperty method*), 259  
`get_named_property()` (*owlapy.parser.OWLObjectPropertyExpression method*), 259  
`get_named_property()` (*owlapy.render.OWLObjectInverseOf method*), 290  
`get_named_property()` (*owlapy.utils.OWLObjectInverseOf method*), 311  
`get_named_property()` (*owlapy.utils.OWLObjectProperty method*), 311  
`get_namespace()` (*owlapy.class\_expression.owl\_class.IRI method*), 25  
`get_namespace()` (*owlapy.iri.IRI method*), 82  
`get_namespace()` (*owlapy.owl\_axiom.IRI method*), 95  
`get_namespace()` (*owlapy.owl\_datatype.IRI method*), 119  
`get_namespace()` (*owlapy.owl\_individual.IRI method*), 134  
`get_namespace()` (*owlapy.owl\_ontology\_manager.IRI method*), 176  
`get_namespace()` (*owlapy.owl\_ontology.IRI method*), 154  
`get_namespace()` (*owlapy.owl\_property.IRI method*), 203  
`get_namespace()` (*owlapy.owl\_reasoner.IRI method*), 219  
`get_namespace()` (*owlapy.parser.IRI method*), 256  
`get_namespace()` (*owlapy.render.IRI method*), 286  
`get_namespace()` (*owlapy.vocab.IRI method*), 332  
`get_nnf()` (*owlapy.class\_expression.class\_expression.OWLAnonymousClassExpression method*), 19  
`get_nnf()` (*owlapy.class\_expression.class\_expression.OWLClassExpression method*), 18  
`get_nnf()` (*owlapy.class\_expression.nary\_boolean\_expression.OWLClassExpression method*), 20  
`get_nnf()` (*owlapy.class\_expression.owl\_class.OWLClass method*), 26  
`get_nnf()` (*owlapy.class\_expression.owl\_class.OWLClassExpression method*), 23  
`get_nnf()` (*owlapy.class\_expression.OWLAnonymousClassExpression method*), 48  
`get_nnf()` (*owlapy.class\_expression.OWLClass method*), 49  
`get_nnf()` (*owlapy.class\_expression.OWLClassExpression method*), 47  
`get_nnf()` (*owlapy.class\_expression.restriction.OWLAnonymousClassExpression method*), 29  
`get_nnf()` (*owlapy.class\_expression.restriction.OWLClassExpression method*), 30  
`get_nnf()` (*owlapy.converter.OWLClass method*), 67  
`get_nnf()` (*owlapy.converter.OWLClassExpression method*), 67  
`get_nnf()` (*owlapy.owl\_axiom.OWLClass method*), 94  
`get_nnf()` (*owlapy.owl\_axiom.OWLClassExpression method*), 93  
`get_nnf()` (*owlapy.owl\_hierarchy.OWLClass method*), 122  
`get_nnf()` (*owlapy.owl\_ontology\_manager.OWLClass method*), 178  
`get_nnf()` (*owlapy.owl\_ontology.OWLClass method*), 155  
`get_nnf()` (*owlapy.owl\_ontology.OWLClassExpression method*), 155  
`get_nnf()` (*owlapy.owl\_reasoner.OWLClass method*), 217  
`get_nnf()` (*owlapy.owl\_reasoner.OWLClassExpression method*), 209  
`get_nnf()` (*owlapy.parser.OWLClass method*), 264  
`get_nnf()` (*owlapy.parser.OWLClassExpression method*), 266  
`get_nnf()` (*owlapy.render.OWLClass method*), 292  
`get_nnf()` (*owlapy.render.OWLClassExpression method*), 291  
`get_nnf()` (*owlapy.utils.OWLClass method*), 313  
`get_nnf()` (*owlapy.utils.OWLClassExpression method*), 312  
`get_object()` (*owlapy.owl\_axiom.OWLPropertyAssertionAxiom method*), 109  
`get_object_complement_of()` (*owlapy.class\_expression.class\_expression.OWLAnonymousClassExpression method*), 18  
`get_object_complement_of()` (*owlapy.class\_expression.class\_expression.OWLClassExpression method*), 18  
`get_object_complement_of()` (*owlapy.class\_expression.nary\_boolean\_expression.OWLClassExpression method*), 20  
`get_object_complement_of()` (*owlapy.class\_expression.owl\_class.OWLClass method*), 25  
`get_object_complement_of()` (*owlapy.class\_expression.owl\_class.OWLClassExpression method*), 23  
`get_object_complement_of()` (*owlapy.class\_expression.OWLAnonymousClassExpression method*), 47  
`get_object_complement_of()` (*owlapy.class\_expression.OWLClass method*), 49  
`get_object_complement_of()` (*owlapy.class\_expression.OWLClassExpression method*), 47  
`get_object_complement_of()` (*owlapy.class\_expression.restriction.OWLAnonymousClassExpression method*), 29

`get_object_complement_of()` (*owlapy.class\_expression.restriction.OWLClassExpression method*), 30  
`get_object_complement_of()` (*owlapy.converter.OWLClass method*), 67  
`get_object_complement_of()` (*owlapy.converter.OWLClassExpression method*), 67  
`get_object_complement_of()` (*owlapy.owl\_axiom.OWLClass method*), 94  
`get_object_complement_of()` (*owlapy.owl\_axiom.OWLClassExpression method*), 93  
`get_object_complement_of()` (*owlapy.owl\_hierarchy.OWLClass method*), 122  
`get_object_complement_of()` (*owlapy.owl\_ontology\_manager.OWLClass method*), 177  
`get_object_complement_of()` (*owlapy.owl\_ontology.OWLClass method*), 155  
`get_object_complement_of()` (*owlapy.owl\_ontology.OWLClassExpression method*), 155  
`get_object_complement_of()` (*owlapy.owl\_reasoner.OWLClass method*), 217  
`get_object_complement_of()` (*owlapy.owl\_reasoner.OWLClassExpression method*), 209  
`get_object_complement_of()` (*owlapy.parser.OWLClass method*), 264  
`get_object_complement_of()` (*owlapy.parser.OWLClassExpression method*), 266  
`get_object_complement_of()` (*owlapy.render.OWLClass method*), 292  
`get_object_complement_of()` (*owlapy.render.OWLClassExpression method*), 291  
`get_object_complement_of()` (*owlapy.utils.OWLClass method*), 313  
`get_object_complement_of()` (*owlapy.utils.OWLClassExpression method*), 312  
`get_ontology()` (*owlapy.owl\_ontology\_manager.OWLOntologyChange method*), 198  
`get_ontology_id()` (*owlapy.owl\_ontology\_manager.Ontology method*), 195  
`get_ontology_id()` (*owlapy.owl\_ontology\_manager.OWLOntology method*), 194  
`get_ontology_id()` (*owlapy.owl\_ontology.Ontology method*), 171  
`get_ontology_id()` (*owlapy.owl\_ontology.OWLOntology method*), 170  
`get_ontology_id()` (*owlapy.owl\_reasoner.Ontology method*), 224  
`get_ontology_id()` (*owlapy.owl\_reasoner.OWLOntology method*), 223  
`get_ontology_iri()` (*owlapy.owl\_ontology.OWLOntologyID method*), 167  
`get_operand()` (*owlapy.class\_expression.class\_expression.OWLObjectComplementOf method*), 19  
`get_operand()` (*owlapy.class\_expression.owl\_class.OWLObjectComplementOf method*), 23  
`get_operand()` (*owlapy.class\_expression.OWLObjectComplementOf method*), 48  
`get_operand()` (*owlapy.converter.OWLObjectComplementOf method*), 68  
`get_operand()` (*owlapy.owl\_ontology\_manager.OWLObjectComplementOf method*), 180  
`get_operand()` (*owlapy.owl\_ontology.OWLObjectComplementOf method*), 156  
`get_operand()` (*owlapy.owl\_reasoner.OWLObjectComplementOf method*), 210  
`get_operand()` (*owlapy.parser.OWLObjectComplementOf method*), 266  
`get_operand()` (*owlapy.render.OWLObjectComplementOf method*), 293  
`get_operand()` (*owlapy.utils.OWLObjectComplementOf method*), 314  
`get_original_iri()` (*owlapy.owl\_ontology\_manager.Ontology method*), 196  
`get_original_iri()` (*owlapy.owl\_ontology.Ontology method*), 172  
`get_original_iri()` (*owlapy.owl\_reasoner.Ontology method*), 225  
`get_owl_class()` (*owlapy.owl\_axiom.OWLDisjointUnionAxiom method*), 105  
`get_owl_class()` (*owlapy.owl\_ontology\_manager.OWLDisjointUnionAxiom method*), 183  
`get_owl_disjoint_classes_axiom()` (*owlapy.owl\_axiom.OWLDisjointUnionAxiom method*), 105  
`get_owl_disjoint_classes_axiom()` (*owlapy.owl\_ontology\_manager.OWLDisjointUnionAxiom method*), 183  
`get_owl_equivalent_classes_axiom()` (*owlapy.owl\_axiom.OWLDisjointUnionAxiom method*), 105  
`get_owl_equivalent_classes_axiom()` (*owlapy.owl\_ontology\_manager.OWLDisjointUnionAxiom method*), 183  
`get_owl_ontology_manager()` (*owlapy.owl\_ontology\_manager.Ontology method*), 195  
`get_owl_ontology_manager()` (*owlapy.owl\_ontology\_manager.OWLOntology method*), 194  
`get_owl_ontology_manager()` (*owlapy.owl\_ontology.Ontology method*), 171  
`get_owl_ontology_manager()` (*owlapy.owl\_ontology.OWLOntology method*), 170  
`get_owl_ontology_manager()` (*owlapy.owl\_reasoner.Ontology method*), 224  
`get_owl_ontology_manager()` (*owlapy.owl\_reasoner.OWLOntology method*), 223  
`get_property()` (*owlapy.class\_expression.OWLDataAllValuesFrom method*), 59  
`get_property()` (*owlapy.class\_expression.OWLDataCardinalityRestriction method*), 54  
`get_property()` (*owlapy.class\_expression.OWLDataHasValue method*), 59  
`get_property()` (*owlapy.class\_expression.OWLDataSomeValuesFrom method*), 58  
`get_property()` (*owlapy.class\_expression.OWLObjectAllValuesFrom method*), 55  
`get_property()` (*owlapy.class\_expression.OWLObjectCardinalityRestriction method*), 52  
`get_property()` (*owlapy.class\_expression.OWLObjectHasSelf method*), 53  
`get_property()` (*owlapy.class\_expression.OWLObjectHasValue method*), 55  
`get_property()` (*owlapy.class\_expression.OWLObjectRestriction method*), 51  
`get_property()` (*owlapy.class\_expression.OWLObjectSomeValuesFrom method*), 55  
`get_property()` (*owlapy.class\_expression.OWLRestriction method*), 50  
`get_property()` (*owlapy.class\_expression.restriction.OWLDataAllValuesFrom method*), 43  
`get_property()` (*owlapy.class\_expression.restriction.OWLDataCardinalityRestriction method*), 41  
`get_property()` (*owlapy.class\_expression.restriction.OWLDataHasValue method*), 43  
`get_property()` (*owlapy.class\_expression.restriction.OWLDataSomeValuesFrom method*), 42  
`get_property()` (*owlapy.class\_expression.restriction.OWLObjectAllValuesFrom method*), 38  
`get_property()` (*owlapy.class\_expression.restriction.OWLObjectCardinalityRestriction method*), 36  
`get_property()` (*owlapy.class\_expression.restriction.OWLObjectHasSelf method*), 39



`get_property()` (*owlapy.class\_expression.restriction.OwlObjectHasValue method*), 39  
`get_property()` (*owlapy.class\_expression.restriction.OwlObjectRestriction method*), 35  
`get_property()` (*owlapy.class\_expression.restriction.OwlObjectSomeValuesFrom method*), 38  
`get_property()` (*owlapy.class\_expression.restriction.OwlRestriction method*), 34  
`get_property()` (*owlapy.converter.OwlDataAllValuesFrom method*), 72  
`get_property()` (*owlapy.converter.OwlDataCardinalityRestriction method*), 71  
`get_property()` (*owlapy.converter.OwlDataHasValue method*), 73  
`get_property()` (*owlapy.converter.OwlDataSomeValuesFrom method*), 72  
`get_property()` (*owlapy.converter.OwlObjectAllValuesFrom method*), 69  
`get_property()` (*owlapy.converter.OwlObjectCardinalityRestriction method*), 70  
`get_property()` (*owlapy.converter.OwlObjectHasSelf method*), 71  
`get_property()` (*owlapy.converter.OwlObjectHasValue method*), 64  
`get_property()` (*owlapy.converter.OwlObjectSomeValuesFrom method*), 69  
`get_property()` (*owlapy.owl\_axiom.OwlAnnotation method*), 106  
`get_property()` (*owlapy.owl\_axiom.OwlAnnotationAssertionAxiom method*), 107  
`get_property()` (*owlapy.owl\_axiom.OwlAnnotationPropertyDomainAxiom method*), 108  
`get_property()` (*owlapy.owl\_axiom.OwlAnnotationPropertyRangeAxiom method*), 108  
`get_property()` (*owlapy.owl\_axiom.OwlPropertyAssertionAxiom method*), 109  
`get_property()` (*owlapy.owl\_axiom.OwlUnaryPropertyAxiom method*), 111  
`get_property()` (*owlapy.owl\_ontology\_manager.OwlAnnotationAssertionAxiom method*), 183  
`get_property()` (*owlapy.owl\_ontology\_manager.OwlDataHasValue method*), 178  
`get_property()` (*owlapy.owl\_ontology\_manager.OwlObjectHasValue method*), 180  
`get_property()` (*owlapy.owl\_ontology.OwlDataAllValuesFrom method*), 160  
`get_property()` (*owlapy.owl\_ontology.OwlDataHasValue method*), 162  
`get_property()` (*owlapy.owl\_ontology.OwlDataSomeValuesFrom method*), 160  
`get_property()` (*owlapy.owl\_ontology.OwlObjectAllValuesFrom method*), 157  
`get_property()` (*owlapy.owl\_ontology.OwlObjectHasValue method*), 159  
`get_property()` (*owlapy.owl\_ontology.OwlObjectRestriction method*), 163  
`get_property()` (*owlapy.owl\_ontology.OwlObjectSomeValuesFrom method*), 157  
`get_property()` (*owlapy.owl\_ontology.OwlRestriction method*), 163  
`get_property()` (*owlapy.owl\_reasoner.OwlDataAllValuesFrom method*), 217  
`get_property()` (*owlapy.owl\_reasoner.OwlDataHasValue method*), 216  
`get_property()` (*owlapy.owl\_reasoner.OwlDataSomeValuesFrom method*), 214  
`get_property()` (*owlapy.owl\_reasoner.OwlObjectAllValuesFrom method*), 211  
`get_property()` (*owlapy.owl\_reasoner.OwlObjectCardinalityRestriction method*), 214  
`get_property()` (*owlapy.owl\_reasoner.OwlObjectHasValue method*), 212  
`get_property()` (*owlapy.owl\_reasoner.OwlObjectSomeValuesFrom method*), 210  
`get_property()` (*owlapy.parser.OwlDataAllValuesFrom method*), 273  
`get_property()` (*owlapy.parser.OwlDataCardinalityRestriction method*), 272  
`get_property()` (*owlapy.parser.OwlDataHasValue method*), 270  
`get_property()` (*owlapy.parser.OwlDataSomeValuesFrom method*), 268  
`get_property()` (*owlapy.parser.OwlObjectAllValuesFrom method*), 272  
`get_property()` (*owlapy.parser.OwlObjectCardinalityRestriction method*), 271  
`get_property()` (*owlapy.parser.OwlObjectHasSelf method*), 262  
`get_property()` (*owlapy.parser.OwlObjectHasValue method*), 269  
`get_property()` (*owlapy.parser.OwlObjectSomeValuesFrom method*), 263  
`get_property()` (*owlapy.render.OwlDataAllValuesFrom method*), 296  
`get_property()` (*owlapy.render.OwlDataHasValue method*), 297  
`get_property()` (*owlapy.render.OwlDataSomeValuesFrom method*), 296  
`get_property()` (*owlapy.render.OwlObjectAllValuesFrom method*), 293  
`get_property()` (*owlapy.render.OwlObjectHasSelf method*), 295  
`get_property()` (*owlapy.render.OwlObjectHasValue method*), 301  
`get_property()` (*owlapy.render.OwlObjectSomeValuesFrom method*), 292  
`get_property()` (*owlapy.render.OwlRestriction method*), 299  
`get_property()` (*owlapy.utils.OwlDataAllValuesFrom method*), 318  
`get_property()` (*owlapy.utils.OwlDataCardinalityRestriction method*), 320  
`get_property()` (*owlapy.utils.OwlDataHasValue method*), 318  
`get_property()` (*owlapy.utils.OwlDataSomeValuesFrom method*), 319  
`get_property()` (*owlapy.utils.OwlObjectAllValuesFrom method*), 319  
`get_property()` (*owlapy.utils.OwlObjectCardinalityRestriction method*), 314  
`get_property()` (*owlapy.utils.OwlObjectHasSelf method*), 316  
`get_property()` (*owlapy.utils.OwlObjectHasValue method*), 321  
`get_property()` (*owlapy.utils.OwlObjectSomeValuesFrom method*), 315  
`get_property()` (*owlapy.utils.OwlRestriction method*), 315  
`get_property_expressions()` (*owlapy.owl\_axiom.OwlHasKeyAxiom method*), 100  
`get_range()` (*owlapy.owl\_axiom.OwlAnnotationPropertyRangeAxiom method*), 108  
`get_range()` (*owlapy.owl\_axiom.OwlPropertyRangeAxiom method*), 114  
`get_range()` (*owlapy.owl\_ontology\_manager.OwlPropertyRangeAxiom method*), 184

`get_remainder()` (*owlapy.class\_expression.owl\_class.IRI method*), 25  
`get_remainder()` (*owlapy.iri.IRI method*), 82  
`get_remainder()` (*owlapy.owl\_axiom.IRI method*), 95  
`get_remainder()` (*owlapy.owl\_datatype.IRI method*), 119  
`get_remainder()` (*owlapy.owl\_individual.IRI method*), 134  
`get_remainder()` (*owlapy.owl\_ontology\_manager.IRI method*), 176  
`get_remainder()` (*owlapy.owl\_ontology.IRI method*), 154  
`get_remainder()` (*owlapy.owl\_property.IRI method*), 204  
`get_remainder()` (*owlapy.owl\_reasoner.IRI method*), 219  
`get_remainder()` (*owlapy.parser.IRI method*), 256  
`get_remainder()` (*owlapy.render.IRI method*), 286  
`get_remainder()` (*owlapy.vocab.IRI method*), 332  
`get_root_ontology()` (*owlapy.owl\_hierarchy.OWLReasoner method*), 129  
`get_root_ontology()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 252  
`get_root_ontology()` (*owlapy.owl\_reasoner.OntologyReasoner method*), 245  
`get_root_ontology()` (*owlapy.owl\_reasoner.OWLReasoner method*), 237  
`get_second_property()` (*owlapy.owl\_axiom.OWLInverseObjectPropertiesAxiom method*), 104  
`get_second_property()` (*owlapy.owl\_ontology\_manager.OWLInverseObjectPropertiesAxiom method*), 190  
`get_short_form()` (*owlapy.class\_expression.owl\_class.IRI method*), 24  
`get_short_form()` (*owlapy.iri.IRI method*), 82  
`get_short_form()` (*owlapy.owl\_axiom.IRI method*), 95  
`get_short_form()` (*owlapy.owl\_datatype.IRI method*), 119  
`get_short_form()` (*owlapy.owl\_individual.IRI method*), 134  
`get_short_form()` (*owlapy.owl\_ontology\_manager.IRI method*), 176  
`get_short_form()` (*owlapy.owl\_ontology.IRI method*), 154  
`get_short_form()` (*owlapy.owl\_property.IRI method*), 203  
`get_short_form()` (*owlapy.owl\_reasoner.IRI method*), 219  
`get_short_form()` (*owlapy.parser.IRI method*), 256  
`get_short_form()` (*owlapy.render.IRI method*), 286  
`get_short_form()` (*owlapy.vocab.IRI method*), 332  
`get_sub_class()` (*owlapy.owl\_axiom.OWLSubClassOfAxiom method*), 104  
`get_sub_class()` (*owlapy.owl\_ontology\_manager.OWLSubClassOfAxiom method*), 182  
`get_sub_class()` (*owlapy.owl\_ontology.OWLSubClassOfAxiom method*), 148  
`get_sub_class()` (*owlapy.owl\_reasoner.OWLSubClassOfAxiom method*), 219  
`get_sub_property()` (*owlapy.owl\_axiom.OWLSubAnnotationPropertyOfAxiom method*), 107  
`get_sub_property()` (*owlapy.owl\_axiom.OWLSubPropertyAxiom method*), 108  
`get_sub_property()` (*owlapy.owl\_ontology\_manager.OWLSubPropertyAxiom method*), 184  
`get_subject()` (*owlapy.owl\_axiom.OWLAnnotationAssertionAxiom method*), 107  
`get_subject()` (*owlapy.owl\_axiom.OWLPropertyAssertionAxiom method*), 109  
`get_subject()` (*owlapy.owl\_ontology\_manager.OWLAnnotationAssertionAxiom method*), 183  
`get_super_class()` (*owlapy.owl\_axiom.OWLSubClassOfAxiom method*), 104  
`get_super_class()` (*owlapy.owl\_ontology\_manager.OWLSubClassOfAxiom method*), 182  
`get_super_class()` (*owlapy.owl\_ontology.OWLSubClassOfAxiom method*), 148  
`get_super_class()` (*owlapy.owl\_reasoner.OWLSubClassOfAxiom method*), 219  
`get_super_property()` (*owlapy.owl\_axiom.OWLSubAnnotationPropertyOfAxiom method*), 107  
`get_super_property()` (*owlapy.owl\_axiom.OWLSubPropertyAxiom method*), 109  
`get_super_property()` (*owlapy.owl\_ontology\_manager.OWLSubPropertyAxiom method*), 184  
`get_top_entity()` (*owlapy.owl\_hierarchy.AbstractHierarchy class method*), 129  
`get_top_entity()` (*owlapy.owl\_hierarchy.ClassHierarchy class method*), 131  
`get_top_entity()` (*owlapy.owl\_hierarchy.DatatypePropertyHierarchy class method*), 132  
`get_top_entity()` (*owlapy.owl\_hierarchy.ObjectPropertyHierarchy class method*), 131  
`get_top_level_cnf()` (*owlapy.utils.TopLevelCNF method*), 329  
`get_top_level_dnf()` (*owlapy.utils.TopLevelDNF method*), 329  
`get_value()` (*owlapy.owl\_axiom.OWLAnnotation method*), 106  
`get_value()` (*owlapy.owl\_axiom.OWLAnnotationAssertionAxiom method*), 107  
`get_value()` (*owlapy.owl\_ontology\_manager.OWLAnnotationAssertionAxiom method*), 183  
`get_variable()` (*owlapy.converter.VariablesMapping method*), 78  
`get_version_iri()` (*owlapy.owl\_ontology.OWLOntologyID method*), 167  
`grouping_vars` (*owlapy.converter.Owl2SparqlConverter attribute*), 79

## H

`HasCardinality` (*class in owlapy.class\_expression.restriction*), 28  
`HasCardinality` (*class in owlapy.meta\_classes*), 84  
`HasCardinality` (*class in owlapy.utils*), 308  
`HasFiller` (*class in owlapy.class\_expression.restriction*), 28  
`HasFiller` (*class in owlapy.meta\_classes*), 83  
`HasFiller` (*class in owlapy.utils*), 308

HasIndex (class in owlapy.utils), 328  
 HasIRI (class in owlapy.meta\_classes), 83  
 HasIRI (class in owlapy.owl\_datatype), 119  
 HasIRI (class in owlapy.owl\_hierarchy), 122  
 HasIRI (class in owlapy.owl\_object), 142  
 HasIRI (class in owlapy.owl\_ontology\_manager), 176  
 HasIRI (class in owlapy.utils), 307  
 HasIRI (class in owlapy.vocab), 330  
 HasOperands (class in owlapy.class\_expression.class\_expression), 17  
 HasOperands (class in owlapy.class\_expression.nary\_boolean\_expression), 21  
 HasOperands (class in owlapy.class\_expression.restriction), 28  
 HasOperands (class in owlapy.meta\_classes), 83  
 HasOperands (class in owlapy.owl\_axiom), 91  
 HasOperands (class in owlapy.owl\_data\_ranges), 116  
 HasOperands (class in owlapy.utils), 308  
 having\_conditions (owlapy.converter.Owl2SparqlConverter attribute), 79  
 HERMIT (owlapy.owl\_reasoner.BaseReasoner attribute), 238

## I

ind\_data\_properties() (owlapy.owl\_reasoner.OwlReasonerEx method), 239  
 ind\_object\_properties() (owlapy.owl\_reasoner.OwlReasonerEx method), 239  
 individuals() (owlapy.class\_expression.OwlObjectOneOf method), 60  
 individuals() (owlapy.class\_expression.restriction.OwlObjectOneOf method), 39  
 individuals() (owlapy.converter.OwlObjectOneOf method), 64  
 individuals() (owlapy.owl\_axiom.OwlNaryIndividualAxiom method), 102  
 individuals() (owlapy.owl\_ontology\_manager.OwlNaryIndividualAxiom method), 190  
 individuals() (owlapy.owl\_ontology\_manager.OwlObjectOneOf method), 179  
 individuals() (owlapy.owl\_ontology.OwlObjectOneOf method), 157  
 individuals() (owlapy.owl\_reasoner.OwlObjectOneOf method), 211  
 individuals() (owlapy.parser.OwlObjectOneOf method), 265  
 individuals() (owlapy.render.OwlObjectOneOf method), 302  
 individuals() (owlapy.utils.OwlObjectOneOf method), 323  
 individuals\_in\_signature() (owlapy.owl\_ontology\_manager.Ontology method), 194  
 individuals\_in\_signature() (owlapy.owl\_ontology\_manager.OwlOntology method), 192  
 individuals\_in\_signature() (owlapy.owl\_ontology.Ontology method), 170  
 individuals\_in\_signature() (owlapy.owl\_ontology.OwlOntology method), 168  
 individuals\_in\_signature() (owlapy.owl\_reasoner.Ontology method), 223  
 individuals\_in\_signature() (owlapy.owl\_reasoner.OwlOntology method), 222  
 instances() (owlapy.owl\_hierarchy.OwlReasoner method), 126  
 instances() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 249  
 instances() (owlapy.owl\_reasoner.OntologyReasoner method), 242  
 instances() (owlapy.owl\_reasoner.OwlReasoner method), 235  
 instances() (owlapy.owl\_reasoner.SyncReasoner method), 252  
 INTEGER (owlapy.owl\_literal.XSDVocabulary attribute), 138  
 INTEGER (owlapy.vocab.XSDVocabulary attribute), 333  
 IntegerOWLDatatype (in module owlapy.owl\_literal), 141  
 IntegerOWLDatatype (in module owlapy.owl\_ontology), 150  
 IntegerOWLDatatype (in module owlapy.parser), 257  
 IRI (class in owlapy.class\_expression.owl\_class), 23  
 IRI (class in owlapy.iri), 81  
 IRI (class in owlapy.owl\_axiom), 94  
 IRI (class in owlapy.owl\_datatype), 118  
 IRI (class in owlapy.owl\_individual), 133  
 IRI (class in owlapy.owl\_ontology), 153  
 IRI (class in owlapy.owl\_ontology\_manager), 175  
 IRI (class in owlapy.owl\_property), 202  
 IRI (class in owlapy.owl\_reasoner), 218  
 IRI (class in owlapy.parser), 255  
 IRI (class in owlapy.render), 285  
 IRI (class in owlapy.vocab), 331  
 iri (owlapy.class\_expression.owl\_class.OwlClass property), 25  
 iri (owlapy.class\_expression.OwlClass property), 48  
 iri (owlapy.class\_expression.restriction.OwlDatatype property), 33  
 iri (owlapy.converter.OwlClass property), 66  
 iri (owlapy.converter.OwlDatatype property), 77  
 iri (owlapy.converter.OwlNamedIndividual property), 74  
 iri (owlapy.meta\_classes.HasIRI property), 83

- `iri (owlapy.owl_axiom.OWLAnnotationProperty property), 106`
- `iri (owlapy.owl_axiom.OWLClass property), 93`
- `iri (owlapy.owl_axiom.OWLDatatype property), 91`
- `iri (owlapy.owl_axiom.OWLProperty property), 92`
- `iri (owlapy.owl_datatype.HasIRI property), 119`
- `iri (owlapy.owl_datatype.OWLDatatype property), 120`
- `iri (owlapy.owl_hierarchy.HasIRI property), 122`
- `iri (owlapy.owl_hierarchy.OWLClass property), 121`
- `iri (owlapy.owl_individual.OWLNamedIndividual property), 135`
- `iri (owlapy.owl_literal.OWLDatatype property), 137`
- `iri (owlapy.owl_object.HasIRI property), 142`
- `iri (owlapy.owl_ontology_manager.HasIRI property), 176`
- `iri (owlapy.owl_ontology_manager.OWLAnnotationProperty property), 184`
- `iri (owlapy.owl_ontology_manager.OWLClass property), 177`
- `iri (owlapy.owl_ontology_manager.OWLImportsDeclaration property), 199`
- `iri (owlapy.owl_ontology_manager.OWLNamedIndividual property), 192`
- `iri (owlapy.owl_ontology_manager.OWLProperty property), 198`
- `iri (owlapy.owl_ontology.OWLAnnotationProperty property), 148`
- `iri (owlapy.owl_ontology.OWLClass property), 154`
- `iri (owlapy.owl_ontology.OWLDatatype property), 150`
- `iri (owlapy.owl_ontology.OWLNamedIndividual property), 150`
- `iri (owlapy.owl_property.OWLProperty property), 205`
- `iri (owlapy.owl_reasoner.OWLClass property), 217`
- `iri (owlapy.owl_reasoner.OWLDatatype property), 221`
- `iri (owlapy.owl_reasoner.OWLNamedIndividual property), 229`
- `iri (owlapy.parser.OWLClass property), 264`
- `iri (owlapy.parser.OWLDatatype property), 261`
- `iri (owlapy.parser.OWLNamedIndividual property), 257`
- `iri (owlapy.render.OWLClass property), 291`
- `iri (owlapy.render.OWLDatatype property), 303`
- `iri (owlapy.render.OWLNamedIndividual property), 286`
- `iri (owlapy.utils.HasIRI property), 308`
- `iri (owlapy.utils.OWLClass property), 313`
- `iri (owlapy.utils.OWLDatatype property), 325`
- `iri (owlapy.utils.OWLNamedIndividual property), 307`
- `iri (owlapy.vocab.HasIRI property), 330`
- `is_annotated () (owlapy.owl_axiom.OWLAxiom method), 98`
- `is_annotated () (owlapy.owl_ontology_manager.OWLAxiom method), 181`
- `is_annotated () (owlapy.owl_reasoner.OWLAxiom method), 219`
- `is_annotation_axiom () (owlapy.owl_axiom.OWLAnnotationAxiom method), 106`
- `is_annotation_axiom () (owlapy.owl_axiom.OWLAxiom method), 98`
- `is_annotation_axiom () (owlapy.owl_ontology_manager.OWLAxiom method), 181`
- `is_annotation_axiom () (owlapy.owl_reasoner.OWLAxiom method), 219`
- `is_anonymous () (owlapy.class_expression.owl_class.OWLEntity method), 23`
- `is_anonymous () (owlapy.class_expression.restriction.OWLObject method), 34`
- `is_anonymous () (owlapy.converter.OWLEntity method), 77`
- `is_anonymous () (owlapy.owl_annotation.OWLObject method), 86`
- `is_anonymous () (owlapy.owl_axiom.OWLEntity method), 91`
- `is_anonymous () (owlapy.owl_axiom.OWLObject method), 90`
- `is_anonymous () (owlapy.owl_data_ranges.OWLObject method), 116`
- `is_anonymous () (owlapy.owl_datatype.OWLEntity method), 118`
- `is_anonymous () (owlapy.owl_individual.OWLEntity method), 133`
- `is_anonymous () (owlapy.owl_individual.OWLObject method), 133`
- `is_anonymous () (owlapy.owl_object.OWLEntity method), 143`
- `is_anonymous () (owlapy.owl_object.OWLObject method), 142`
- `is_anonymous () (owlapy.owl_ontology_manager.OWLObject method), 177`
- `is_anonymous () (owlapy.owl_ontology_manager.OWLOntology method), 194`
- `is_anonymous () (owlapy.owl_ontology.OWLObject method), 153`
- `is_anonymous () (owlapy.owl_ontology.OWLOntology method), 170`
- `is_anonymous () (owlapy.owl_ontology.OWLOntologyID method), 168`
- `is_anonymous () (owlapy.owl_property.OWLEntity method), 202`
- `is_anonymous () (owlapy.owl_property.OWLObject method), 202`
- `is_anonymous () (owlapy.owl_reasoner.OWLOntology method), 223`
- `is_anonymous () (owlapy.render.OWLEntity method), 289`
- `is_anonymous () (owlapy.render.OWLObject method), 289`
- `is_anonymous () (owlapy.utils.OWLObject method), 325`
- `is_boolean () (owlapy.class_expression.restriction.OWLLiteral method), 31`
- `is_boolean () (owlapy.converter.OWLLiteral method), 74`

is\_boolean() (owlapy.owl\_axiom.OWLLiteral method), 96  
 is\_boolean() (owlapy.owl\_literal.OWLLiteral method), 139  
 is\_boolean() (owlapy.owl\_ontology.OWLLiteral method), 151  
 is\_boolean() (owlapy.owl\_reasoner.OWLLiteral method), 230  
 is\_boolean() (owlapy.parser.OWLLiteral method), 257  
 is\_boolean() (owlapy.providers.OWLLiteral method), 279  
 is\_boolean() (owlapy.render.OWLLiteral method), 287  
 is\_boolean() (owlapy.utils.OWLLiteral method), 309  
 is\_child\_of() (owlapy.owl\_hierarchy.AbstractHierarchy method), 130  
 is\_data\_property\_expression() (owlapy.class\_expression.restriction.OWLDataPropertyExpression method), 31  
 is\_data\_property\_expression() (owlapy.class\_expression.restriction.OWLPropertyExpression method), 30  
 is\_data\_property\_expression() (owlapy.owl\_axiom.OWLDataPropertyExpression method), 90  
 is\_data\_property\_expression() (owlapy.owl\_axiom.OWLPropertyExpression method), 91  
 is\_data\_property\_expression() (owlapy.owl\_ontology.OWLDataPropertyExpression method), 167  
 is\_data\_property\_expression() (owlapy.owl\_ontology.OWLPropertyExpression method), 165  
 is\_data\_property\_expression() (owlapy.owl\_property.OWLDataPropertyExpression method), 205  
 is\_data\_property\_expression() (owlapy.owl\_property.OWLPropertyExpression method), 204  
 is\_data\_property\_expression() (owlapy.owl\_reasoner.OWLDataPropertyExpression method), 229  
 is\_data\_property\_expression() (owlapy.owl\_reasoner.OWLPropertyExpression method), 228  
 is\_data\_property\_expression() (owlapy.render.OWLPropertyExpression method), 290  
 is\_data\_restriction() (owlapy.class\_expression.OWLDataRestriction method), 52  
 is\_data\_restriction() (owlapy.class\_expression.OWLRestriction method), 50  
 is\_data\_restriction() (owlapy.class\_expression.restriction.OWLDataRestriction method), 40  
 is\_data\_restriction() (owlapy.class\_expression.restriction.OWLRestriction method), 34  
 is\_data\_restriction() (owlapy.owl\_ontology.OWLDataRestriction method), 163  
 is\_data\_restriction() (owlapy.owl\_ontology.OWLRestriction method), 163  
 is\_data\_restriction() (owlapy.render.OWLRestriction method), 299  
 is\_data\_restriction() (owlapy.utils.OWLRestriction method), 315  
 is\_date() (owlapy.class\_expression.restriction.OWLLiteral method), 32  
 is\_date() (owlapy.converter.OWLLiteral method), 75  
 is\_date() (owlapy.owl\_axiom.OWLLiteral method), 97  
 is\_date() (owlapy.owl\_literal.OWLLiteral method), 140  
 is\_date() (owlapy.owl\_ontology.OWLLiteral method), 152  
 is\_date() (owlapy.owl\_reasoner.OWLLiteral method), 230  
 is\_date() (owlapy.parser.OWLLiteral method), 258  
 is\_date() (owlapy.providers.OWLLiteral method), 280  
 is\_date() (owlapy.render.OWLLiteral method), 288  
 is\_date() (owlapy.utils.OWLLiteral method), 310  
 is\_datetime() (owlapy.class\_expression.restriction.OWLLiteral method), 32  
 is\_datetime() (owlapy.converter.OWLLiteral method), 75  
 is\_datetime() (owlapy.owl\_axiom.OWLLiteral method), 97  
 is\_datetime() (owlapy.owl\_literal.OWLLiteral method), 140  
 is\_datetime() (owlapy.owl\_ontology.OWLLiteral method), 152  
 is\_datetime() (owlapy.owl\_reasoner.OWLLiteral method), 230  
 is\_datetime() (owlapy.parser.OWLLiteral method), 258  
 is\_datetime() (owlapy.providers.OWLLiteral method), 280  
 is\_datetime() (owlapy.render.OWLLiteral method), 288  
 is\_datetime() (owlapy.utils.OWLLiteral method), 310  
 is\_double() (owlapy.class\_expression.restriction.OWLLiteral method), 32  
 is\_double() (owlapy.converter.OWLLiteral method), 74  
 is\_double() (owlapy.owl\_axiom.OWLLiteral method), 96  
 is\_double() (owlapy.owl\_literal.OWLLiteral method), 139  
 is\_double() (owlapy.owl\_ontology.OWLLiteral method), 151  
 is\_double() (owlapy.owl\_reasoner.OWLLiteral method), 230  
 is\_double() (owlapy.parser.OWLLiteral method), 258  
 is\_double() (owlapy.providers.OWLLiteral method), 280  
 is\_double() (owlapy.render.OWLLiteral method), 287  
 is\_double() (owlapy.utils.OWLLiteral method), 309  
 is\_duration() (owlapy.class\_expression.restriction.OWLLiteral method), 32  
 is\_duration() (owlapy.converter.OWLLiteral method), 75  
 is\_duration() (owlapy.owl\_axiom.OWLLiteral method), 97  
 is\_duration() (owlapy.owl\_literal.OWLLiteral method), 140  
 is\_duration() (owlapy.owl\_ontology.OWLLiteral method), 152  
 is\_duration() (owlapy.owl\_reasoner.OWLLiteral method), 231  
 is\_duration() (owlapy.parser.OWLLiteral method), 258  
 is\_duration() (owlapy.providers.OWLLiteral method), 280  
 is\_duration() (owlapy.render.OWLLiteral method), 288  
 is\_duration() (owlapy.utils.OWLLiteral method), 310



`is_integer()` (*owlapy.class\_expression.restriction.OWLLiteral method*), 32  
`is_integer()` (*owlapy.converter.OWLLiteral method*), 75  
`is_integer()` (*owlapy.owl\_axiom.OWLLiteral method*), 96  
`is_integer()` (*owlapy.owl\_literal.OWLLiteral method*), 140  
`is_integer()` (*owlapy.owl\_ontology.OWLLiteral method*), 151  
`is_integer()` (*owlapy.owl\_reasoner.OWLLiteral method*), 230  
`is_integer()` (*owlapy.parser.OWLLiteral method*), 258  
`is_integer()` (*owlapy.providers.OWLLiteral method*), 280  
`is_integer()` (*owlapy.render.OWLLiteral method*), 287  
`is_integer()` (*owlapy.utils.OWLLiteral method*), 309  
`is_isolated()` (*owlapy.owl\_hierarchy.OWLReasoner method*), 129  
`is_isolated()` (*owlapy.owl\_reasoner.FastInstanceCheckerReasoner method*), 246  
`is_isolated()` (*owlapy.owl\_reasoner.OntologyReasoner method*), 245  
`is_isolated()` (*owlapy.owl\_reasoner.OWLReasoner method*), 237  
`is_literal()` (*owlapy.class\_expression.restriction.OWLLiteral method*), 33  
`is_literal()` (*owlapy.converter.OWLLiteral method*), 75  
`is_literal()` (*owlapy.iri.OWLAnnotationValue method*), 81  
`is_literal()` (*owlapy.owl\_annotation.OWLAnnotationValue method*), 86  
`is_literal()` (*owlapy.owl\_axiom.OWLAnnotationValue method*), 96  
`is_literal()` (*owlapy.owl\_axiom.OWLLiteral method*), 97  
`is_literal()` (*owlapy.owl\_literal.OWLAnnotationValue method*), 137  
`is_literal()` (*owlapy.owl\_literal.OWLLiteral method*), 140  
`is_literal()` (*owlapy.owl\_ontology.OWLLiteral method*), 152  
`is_literal()` (*owlapy.owl\_reasoner.OWLLiteral method*), 231  
`is_literal()` (*owlapy.parser.OWLLiteral method*), 259  
`is_literal()` (*owlapy.providers.OWLLiteral method*), 281  
`is_literal()` (*owlapy.render.OWLLiteral method*), 288  
`is_literal()` (*owlapy.utils.OWLLiteral method*), 310  
`is_logical_axiom()` (*owlapy.owl\_axiom.OWLXiom method*), 98  
`is_logical_axiom()` (*owlapy.owl\_axiom.OWLLogicalXiom method*), 98  
`is_logical_axiom()` (*owlapy.owl\_ontology\_manager.OWLXiom method*), 181  
`is_logical_axiom()` (*owlapy.owl\_reasoner.OWLXiom method*), 219  
`is_nothing()` (*owlapy.class\_expression.owl\_class.IRI method*), 24  
`is_nothing()` (*owlapy.iri.IRI method*), 82  
`is_nothing()` (*owlapy.owl\_axiom.IRI method*), 95  
`is_nothing()` (*owlapy.owl\_datatype.IRI method*), 118  
`is_nothing()` (*owlapy.owl\_individual.IRI method*), 134  
`is_nothing()` (*owlapy.owl\_ontology\_manager.IRI method*), 175  
`is_nothing()` (*owlapy.owl\_ontology.IRI method*), 153  
`is_nothing()` (*owlapy.owl\_property.IRI method*), 203  
`is_nothing()` (*owlapy.owl\_reasoner.IRI method*), 218  
`is_nothing()` (*owlapy.parser.IRI method*), 256  
`is_nothing()` (*owlapy.render.IRI method*), 285  
`is_nothing()` (*owlapy.vocab.IRI method*), 331  
`is_object_property_expression()` (*owlapy.class\_expression.restriction.OWLObjectPropertyExpression method*), 31  
`is_object_property_expression()` (*owlapy.class\_expression.restriction.OWLPropertyExpression method*), 30  
`is_object_property_expression()` (*owlapy.owl\_axiom.OWLObjectPropertyExpression method*), 90  
`is_object_property_expression()` (*owlapy.owl\_axiom.OWLPropertyExpression method*), 92  
`is_object_property_expression()` (*owlapy.owl\_ontology.OWLObjectPropertyExpression method*), 166  
`is_object_property_expression()` (*owlapy.owl\_ontology.OWLPropertyExpression method*), 165  
`is_object_property_expression()` (*owlapy.owl\_property.OWLObjectPropertyExpression method*), 205  
`is_object_property_expression()` (*owlapy.owl\_property.OWLPropertyExpression method*), 204  
`is_object_property_expression()` (*owlapy.owl\_reasoner.OWLObjectPropertyExpression method*), 227  
`is_object_property_expression()` (*owlapy.owl\_reasoner.OWLPropertyExpression method*), 228  
`is_object_property_expression()` (*owlapy.parser.OWLObjectPropertyExpression method*), 259  
`is_object_property_expression()` (*owlapy.render.OWLPropertyExpression method*), 290  
`is_object_restriction()` (*owlapy.class\_expression.OWLObjectRestriction method*), 51  
`is_object_restriction()` (*owlapy.class\_expression.OWLRestriction method*), 50  
`is_object_restriction()` (*owlapy.class\_expression.restriction.OWLObjectRestriction method*), 35  
`is_object_restriction()` (*owlapy.class\_expression.restriction.OWLRestriction method*), 34  
`is_object_restriction()` (*owlapy.owl\_ontology.OWLObjectRestriction method*), 163  
`is_object_restriction()` (*owlapy.owl\_ontology.OWLRestriction method*), 163  
`is_object_restriction()` (*owlapy.render.OWLRestriction method*), 299  
`is_object_restriction()` (*owlapy.utils.OWLRestriction method*), 315  
`is_owl_nothing()` (*owlapy.class\_expression.class\_expression.OWLAnonymousClassExpression method*), 18  
`is_owl_nothing()` (*owlapy.class\_expression.class\_expression.OWLClassExpression method*), 18  
`is_owl_nothing()` (*owlapy.class\_expression.nary\_boolean\_expression.OWLClassExpression method*), 20  
`is_owl_nothing()` (*owlapy.class\_expression.owl\_class.OWLClass method*), 25

`is_owl_nothing()` (*owlapy.class\_expression.owl\_class.OWLClassExpression method*), 22  
`is_owl_nothing()` (*owlapy.class\_expression.OWLAnonymousClassExpression method*), 47  
`is_owl_nothing()` (*owlapy.class\_expression.OWLClass method*), 49  
`is_owl_nothing()` (*owlapy.class\_expression.OWLClassExpression method*), 47  
`is_owl_nothing()` (*owlapy.class\_expression.restriction.OWLAnonymousClassExpression method*), 29  
`is_owl_nothing()` (*owlapy.class\_expression.restriction.OWLClassExpression method*), 29  
`is_owl_nothing()` (*owlapy.converter.OWLClass method*), 67  
`is_owl_nothing()` (*owlapy.converter.OWLClassExpression method*), 67  
`is_owl_nothing()` (*owlapy.owl\_axiom.OWLClass method*), 93  
`is_owl_nothing()` (*owlapy.owl\_axiom.OWLClassExpression method*), 93  
`is_owl_nothing()` (*owlapy.owl\_hierarchy.OWLClass method*), 122  
`is_owl_nothing()` (*owlapy.owl\_ontology\_manager.OWLClass method*), 177  
`is_owl_nothing()` (*owlapy.owl\_ontology.OWLClass method*), 154  
`is_owl_nothing()` (*owlapy.owl\_ontology.OWLClassExpression method*), 155  
`is_owl_nothing()` (*owlapy.owl\_reasoner.OWLClass method*), 217  
`is_owl_nothing()` (*owlapy.owl\_reasoner.OWLClassExpression method*), 209  
`is_owl_nothing()` (*owlapy.parser.OWLClass method*), 264  
`is_owl_nothing()` (*owlapy.parser.OWLClassExpression method*), 265  
`is_owl_nothing()` (*owlapy.render.OWLClass method*), 292  
`is_owl_nothing()` (*owlapy.render.OWLClassExpression method*), 291  
`is_owl_nothing()` (*owlapy.utils.OWLClass method*), 313  
`is_owl_nothing()` (*owlapy.utils.OWLClassExpression method*), 312  
`is_owl_thing()` (*owlapy.class\_expression.class\_expression.OWLAnonymousClassExpression method*), 18  
`is_owl_thing()` (*owlapy.class\_expression.class\_expression.OWLClassExpression method*), 18  
`is_owl_thing()` (*owlapy.class\_expression.nary\_boolean\_expression.OWLClassExpression method*), 20  
`is_owl_thing()` (*owlapy.class\_expression.owl\_class.OWLClass method*), 25  
`is_owl_thing()` (*owlapy.class\_expression.owl\_class.OWLClassExpression method*), 22  
`is_owl_thing()` (*owlapy.class\_expression.OWLAnonymousClassExpression method*), 47  
`is_owl_thing()` (*owlapy.class\_expression.OWLClass method*), 49  
`is_owl_thing()` (*owlapy.class\_expression.OWLClassExpression method*), 47  
`is_owl_thing()` (*owlapy.class\_expression.restriction.OWLAnonymousClassExpression method*), 29  
`is_owl_thing()` (*owlapy.class\_expression.restriction.OWLClassExpression method*), 29  
`is_owl_thing()` (*owlapy.converter.OWLClass method*), 67  
`is_owl_thing()` (*owlapy.converter.OWLClassExpression method*), 67  
`is_owl_thing()` (*owlapy.owl\_axiom.OWLClass method*), 93  
`is_owl_thing()` (*owlapy.owl\_axiom.OWLClassExpression method*), 92  
`is_owl_thing()` (*owlapy.owl\_hierarchy.OWLClass method*), 121  
`is_owl_thing()` (*owlapy.owl\_ontology\_manager.OWLClass method*), 177  
`is_owl_thing()` (*owlapy.owl\_ontology.OWLClass method*), 154  
`is_owl_thing()` (*owlapy.owl\_ontology.OWLClassExpression method*), 155  
`is_owl_thing()` (*owlapy.owl\_reasoner.OWLClass method*), 217  
`is_owl_thing()` (*owlapy.owl\_reasoner.OWLClassExpression method*), 209  
`is_owl_thing()` (*owlapy.parser.OWLClass method*), 264  
`is_owl_thing()` (*owlapy.parser.OWLClassExpression method*), 265  
`is_owl_thing()` (*owlapy.render.OWLClass method*), 292  
`is_owl_thing()` (*owlapy.render.OWLClassExpression method*), 291  
`is_owl_thing()` (*owlapy.utils.OWLClass method*), 313  
`is_owl_thing()` (*owlapy.utils.OWLClassExpression method*), 312  
`is_owl_top_data_property()` (*owlapy.class\_expression.restriction.OWLPropertyExpression method*), 30  
`is_owl_top_data_property()` (*owlapy.converter.OWLDataProperty method*), 77  
`is_owl_top_data_property()` (*owlapy.owl\_axiom.OWLPropertyExpression method*), 92  
`is_owl_top_data_property()` (*owlapy.owl\_hierarchy.OWLDataProperty method*), 123  
`is_owl_top_data_property()` (*owlapy.owl\_literal.OWLDataProperty method*), 139  
`is_owl_top_data_property()` (*owlapy.owl\_ontology\_manager.OWLDataProperty method*), 196  
`is_owl_top_data_property()` (*owlapy.owl\_ontology.OWLDataProperty method*), 164  
`is_owl_top_data_property()` (*owlapy.owl\_ontology.OWLPropertyExpression method*), 165  
`is_owl_top_data_property()` (*owlapy.owl\_property.OWLDataProperty method*), 207  
`is_owl_top_data_property()` (*owlapy.owl\_property.OWLPropertyExpression method*), 204  
`is_owl_top_data_property()` (*owlapy.owl\_reasoner.OWLDataProperty method*), 227  
`is_owl_top_data_property()` (*owlapy.owl\_reasoner.OWLPropertyExpression method*), 229  
`is_owl_top_data_property()` (*owlapy.parser.OWLDataProperty method*), 260  
`is_owl_top_data_property()` (*owlapy.render.OWLPropertyExpression method*), 290  
`is_owl_top_data_property()` (*owlapy.utils.OWLDataProperty method*), 312  
`is_owl_top_object_property()` (*owlapy.class\_expression.restriction.OWLPropertyExpression method*), 30  
`is_owl_top_object_property()` (*owlapy.converter.OWLObjectProperty method*), 76  
`is_owl_top_object_property()` (*owlapy.owl\_axiom.OWLPropertyExpression method*), 92  
`is_owl_top_object_property()` (*owlapy.owl\_hierarchy.OWLObjectProperty method*), 123  
`is_owl_top_object_property()` (*owlapy.owl\_literal.OWLObjectProperty method*), 138

is\_owl\_top\_object\_property () (owlapy.owl\_ontology\_manager.OwlObjectProperty method), 198  
 is\_owl\_top\_object\_property () (owlapy.owl\_ontology.OwlObjectProperty method), 165  
 is\_owl\_top\_object\_property () (owlapy.owl\_ontology.OwlPropertyExpression method), 165  
 is\_owl\_top\_object\_property () (owlapy.owl\_property.OwlObjectProperty method), 205  
 is\_owl\_top\_object\_property () (owlapy.owl\_property.OwlPropertyExpression method), 204  
 is\_owl\_top\_object\_property () (owlapy.owl\_reasoner.OwlObjectProperty method), 227  
 is\_owl\_top\_object\_property () (owlapy.owl\_reasoner.OwlPropertyExpression method), 228  
 is\_owl\_top\_object\_property () (owlapy.parser.OwlObjectProperty method), 260  
 is\_owl\_top\_object\_property () (owlapy.render.OwlPropertyExpression method), 290  
 is\_owl\_top\_object\_property () (owlapy.utils.OwlObjectProperty method), 311  
 is\_parent\_of () (owlapy.owl\_hierarchy.AbstractHierarchy method), 130  
 is\_reserved\_vocabulary () (owlapy.class\_expression.owl\_class.IRI method), 24  
 is\_reserved\_vocabulary () (owlapy.iri.IRI method), 82  
 is\_reserved\_vocabulary () (owlapy.owl\_axiom.IRI method), 95  
 is\_reserved\_vocabulary () (owlapy.owl\_datatype.IRI method), 119  
 is\_reserved\_vocabulary () (owlapy.owl\_individual.IRI method), 134  
 is\_reserved\_vocabulary () (owlapy.owl\_ontology\_manager.IRI method), 175  
 is\_reserved\_vocabulary () (owlapy.owl\_ontology.IRI method), 153  
 is\_reserved\_vocabulary () (owlapy.owl\_property.IRI method), 203  
 is\_reserved\_vocabulary () (owlapy.owl\_reasoner.IRI method), 218  
 is\_reserved\_vocabulary () (owlapy.parser.IRI method), 256  
 is\_reserved\_vocabulary () (owlapy.render.IRI method), 286  
 is\_reserved\_vocabulary () (owlapy.vocab.IRI method), 331  
 is\_string () (owlapy.class\_expression.restriction.OwLLiteral method), 32  
 is\_string () (owlapy.converter.OwLLiteral method), 75  
 is\_string () (owlapy.owl\_axiom.OwLLiteral method), 97  
 is\_string () (owlapy.owl\_literal.OwLLiteral method), 140  
 is\_string () (owlapy.owl\_ontology.OwLLiteral method), 151  
 is\_string () (owlapy.owl\_reasoner.OwLLiteral method), 230  
 is\_string () (owlapy.parser.OwLLiteral method), 258  
 is\_string () (owlapy.providers.OwLLiteral method), 280  
 is\_string () (owlapy.render.OwLLiteral method), 287  
 is\_string () (owlapy.utils.OwLLiteral method), 309  
 is\_sub\_property\_of () (owlapy.owl\_hierarchy.DatatypePropertyHierarchy method), 132  
 is\_sub\_property\_of () (owlapy.owl\_hierarchy.ObjectPropertyHierarchy method), 132  
 is\_subclass\_of () (owlapy.owl\_hierarchy.ClassHierarchy method), 131  
 is\_thing () (owlapy.class\_expression.owl\_class.IRI method), 24  
 is\_thing () (owlapy.iri.IRI method), 82  
 is\_thing () (owlapy.owl\_axiom.IRI method), 95  
 is\_thing () (owlapy.owl\_datatype.IRI method), 118  
 is\_thing () (owlapy.owl\_individual.IRI method), 134  
 is\_thing () (owlapy.owl\_ontology\_manager.IRI method), 175  
 is\_thing () (owlapy.owl\_ontology.IRI method), 153  
 is\_thing () (owlapy.owl\_property.IRI method), 203  
 is\_thing () (owlapy.owl\_reasoner.IRI method), 218  
 is\_thing () (owlapy.parser.IRI method), 256  
 is\_thing () (owlapy.render.IRI method), 285  
 is\_thing () (owlapy.vocab.IRI method), 331  
 is\_using\_triplestore () (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 246  
 items (owlapy.utils.EvaluatedDescriptionSet attribute), 327  
 items () (owlapy.owl\_hierarchy.AbstractHierarchy method), 130  
 iter\_count () (in module owlapy.utils), 329

## K

KEY (owlapy.owl\_reasoner.LRUCache attribute), 231  
 KEY (owlapy.utils.LRUCache attribute), 329

## L

leaves () (owlapy.owl\_hierarchy.AbstractHierarchy method), 131  
 LENGTH (owlapy.class\_expression.OwLFacet attribute), 56  
 LENGTH (owlapy.class\_expression.restriction.OwLFacet attribute), 34  
 LENGTH (owlapy.converter.OwLFacet attribute), 78  
 LENGTH (owlapy.owl\_ontology.OwLFacet attribute), 167  
 LENGTH (owlapy.parser.OwLFacet attribute), 261  
 LENGTH (owlapy.providers.OwLFacet attribute), 282  
 LENGTH (owlapy.render.OwLFacet attribute), 299  
 LENGTH (owlapy.vocab.OwLFacet attribute), 333



- `length()` (*owlapy.utils.OWLClassExpressionLengthMetric method*), 327
- Literals (*in module owlapy.class\_expression.restriction*), 34
- Literals (*in module owlapy.owl\_literal*), 139
- `load_ontology()` (*owlapy.owl\_ontology\_manager.OntologyManager method*), 200
- `load_ontology()` (*owlapy.owl\_ontology\_manager.OWLOntologyManager method*), 198
- `load_ontology()` (*owlapy.owl\_reasoner.OntologyManager method*), 225
- `logger` (*in module owlapy.owl\_ontology*), 167
- `logger` (*in module owlapy.owl\_reasoner*), 232
- LONG (*owlapy.owl\_literal.XSDVocabulary attribute*), 138
- LONG (*owlapy.vocab.XSDVocabulary attribute*), 333
- LRUCache (*class in owlapy.owl\_reasoner*), 231
- LRUCache (*class in owlapy.utils*), 329

## M

- MANCHESTER\_GRAMMAR (*in module owlapy.parser*), 274
- `manchester_to_owl_expression()` (*in module owlapy*), 334
- `manchester_to_owl_expression()` (*in module owlapy.parser*), 278
- ManchesterOWLSyntaxOWLObjectRenderer (*class in owlapy.render*), 303
- ManchesterOWLSyntaxParser (*class in owlapy.parser*), 274
- ManchesterParser (*in module owlapy.parser*), 278
- ManchesterRenderer (*in module owlapy.render*), 304
- `map_concept()` (*owlapy.owl\_ontology\_manager.ToOwlready2 method*), 196
- `map_concept()` (*owlapy.owl\_ontology.FromOwlready2 method*), 172
- `map_concept()` (*owlapy.owl\_ontology.ToOwlready2 method*), 172
- `map_concept()` (*owlapy.owl\_reasoner.ToOwlready2 method*), 225
- `map_datarange()` (*owlapy.owl\_ontology\_manager.ToOwlready2 method*), 196
- `map_datarange()` (*owlapy.owl\_ontology.FromOwlready2 method*), 172
- `map_datarange()` (*owlapy.owl\_ontology.ToOwlready2 method*), 172
- `map_datarange()` (*owlapy.owl\_reasoner.ToOwlready2 method*), 225
- `map_object()` (*owlapy.owl\_ontology\_manager.ToOwlready2 method*), 196
- `map_object()` (*owlapy.owl\_ontology.ToOwlready2 method*), 172
- `map_object()` (*owlapy.owl\_reasoner.ToOwlready2 method*), 225
- mapping (*owlapy.converter.Owl2SparqlConverter attribute*), 79
- MAX\_EXCLUSIVE (*owlapy.class\_expression.OWLFacet attribute*), 56
- MAX\_EXCLUSIVE (*owlapy.class\_expression.restriction.OWLFacet attribute*), 34
- MAX\_EXCLUSIVE (*owlapy.converter.OWLFacet attribute*), 78
- MAX\_EXCLUSIVE (*owlapy.owl\_ontology.OWLFacet attribute*), 167
- MAX\_EXCLUSIVE (*owlapy.parser.OWLFacet attribute*), 261
- MAX\_EXCLUSIVE (*owlapy.providers.OWLFacet attribute*), 282
- MAX\_EXCLUSIVE (*owlapy.render.OWLFacet attribute*), 299
- MAX\_EXCLUSIVE (*owlapy.vocab.OWLFacet attribute*), 333
- MAX\_INCLUSIVE (*owlapy.class\_expression.OWLFacet attribute*), 56
- MAX\_INCLUSIVE (*owlapy.class\_expression.restriction.OWLFacet attribute*), 34
- MAX\_INCLUSIVE (*owlapy.converter.OWLFacet attribute*), 78
- MAX\_INCLUSIVE (*owlapy.owl\_ontology.OWLFacet attribute*), 167
- MAX\_INCLUSIVE (*owlapy.parser.OWLFacet attribute*), 261
- MAX\_INCLUSIVE (*owlapy.providers.OWLFacet attribute*), 282
- MAX\_INCLUSIVE (*owlapy.render.OWLFacet attribute*), 299
- MAX\_INCLUSIVE (*owlapy.vocab.OWLFacet attribute*), 333
- MAX\_LENGTH (*owlapy.class\_expression.OWLFacet attribute*), 56
- MAX\_LENGTH (*owlapy.class\_expression.restriction.OWLFacet attribute*), 34
- MAX\_LENGTH (*owlapy.converter.OWLFacet attribute*), 78
- MAX\_LENGTH (*owlapy.owl\_ontology.OWLFacet attribute*), 167
- MAX\_LENGTH (*owlapy.parser.OWLFacet attribute*), 262
- MAX\_LENGTH (*owlapy.providers.OWLFacet attribute*), 282
- MAX\_LENGTH (*owlapy.render.OWLFacet attribute*), 299
- MAX\_LENGTH (*owlapy.vocab.OWLFacet attribute*), 333
- `maybe_add()` (*owlapy.utils.EvaluatedDescriptionSet method*), 327
- measurer (*in module owlapy.utils*), 327
- MIN\_EXCLUSIVE (*owlapy.class\_expression.OWLFacet attribute*), 56
- MIN\_EXCLUSIVE (*owlapy.class\_expression.restriction.OWLFacet attribute*), 34
- MIN\_EXCLUSIVE (*owlapy.converter.OWLFacet attribute*), 78
- MIN\_EXCLUSIVE (*owlapy.owl\_ontology.OWLFacet attribute*), 167
- MIN\_EXCLUSIVE (*owlapy.parser.OWLFacet attribute*), 261
- MIN\_EXCLUSIVE (*owlapy.providers.OWLFacet attribute*), 282
- MIN\_EXCLUSIVE (*owlapy.render.OWLFacet attribute*), 299
- MIN\_EXCLUSIVE (*owlapy.vocab.OWLFacet attribute*), 333

MIN\_INCLUSIVE (*owlapy.class\_expression.OWLFacet attribute*), 56  
 MIN\_INCLUSIVE (*owlapy.class\_expression.restriction.OWLFacet attribute*), 34  
 MIN\_INCLUSIVE (*owlapy.converter.OWLFacet attribute*), 77  
 MIN\_INCLUSIVE (*owlapy.owl\_ontology.OWLFacet attribute*), 167  
 MIN\_INCLUSIVE (*owlapy.parser.OWLFacet attribute*), 261  
 MIN\_INCLUSIVE (*owlapy.providers.OWLFacet attribute*), 282  
 MIN\_INCLUSIVE (*owlapy.render.OWLFacet attribute*), 299  
 MIN\_INCLUSIVE (*owlapy.vocab.OWLFacet attribute*), 333  
 MIN\_LENGTH (*owlapy.class\_expression.OWLFacet attribute*), 56  
 MIN\_LENGTH (*owlapy.class\_expression.restriction.OWLFacet attribute*), 34  
 MIN\_LENGTH (*owlapy.converter.OWLFacet attribute*), 78  
 MIN\_LENGTH (*owlapy.owl\_ontology.OWLFacet attribute*), 167  
 MIN\_LENGTH (*owlapy.parser.OWLFacet attribute*), 261  
 MIN\_LENGTH (*owlapy.providers.OWLFacet attribute*), 282  
 MIN\_LENGTH (*owlapy.render.OWLFacet attribute*), 299  
 MIN\_LENGTH (*owlapy.vocab.OWLFacet attribute*), 333  
 modal\_depth (*owlapy.converter.Owl2SparqlConverter property*), 79  
 module  
     owlapy, 17  
     owlapy.class\_expression, 17  
     owlapy.class\_expression.class\_expression, 17  
     owlapy.class\_expression.nary\_boolean\_expression, 20  
     owlapy.class\_expression.owl\_class, 22  
     owlapy.class\_expression.restriction, 26  
     owlapy.converter, 62  
     owlapy.entities, 62  
     owlapy.iri, 80  
     owlapy.meta\_classes, 83  
     owlapy.namespaces, 84  
     owlapy.owl\_annotation, 85  
     owlapy.owl\_axiom, 86  
     owlapy.owl\_data\_ranges, 115  
     owlapy.owl\_datatype, 117  
     owlapy.owl\_hierarchy, 120  
     owlapy.owl\_individual, 132  
     owlapy.owl\_literal, 135  
     owlapy.owl\_object, 141  
     owlapy.owl\_ontology, 144  
     owlapy.owl\_ontology\_manager, 173  
     owlapy.owl\_property, 201  
     owlapy.owl\_reasoner, 207  
     owlapy.parser, 253  
     owlapy.providers, 279  
     owlapy.render, 283  
     owlapy.static\_funcs, 304  
     owlapy.utils, 305  
     owlapy.vocab, 330  
 more\_general\_roles() (*owlapy.owl\_hierarchy.DatatypePropertyHierarchy method*), 132  
 more\_general\_roles() (*owlapy.owl\_hierarchy.ObjectPropertyHierarchy method*), 131  
 more\_special\_roles() (*owlapy.owl\_hierarchy.DatatypePropertyHierarchy method*), 132  
 more\_special\_roles() (*owlapy.owl\_hierarchy.ObjectPropertyHierarchy method*), 132  
 most\_general\_roles() (*owlapy.owl\_hierarchy.DatatypePropertyHierarchy method*), 132  
 most\_general\_roles() (*owlapy.owl\_hierarchy.ObjectPropertyHierarchy method*), 132  
 most\_special\_roles() (*owlapy.owl\_hierarchy.DatatypePropertyHierarchy method*), 132  
 most\_special\_roles() (*owlapy.owl\_hierarchy.ObjectPropertyHierarchy method*), 132  
 move() (*in module owlapy.static\_funcs*), 304

## N

named\_classes() (*owlapy.owl\_axiom.OWLEquivalentClassesAxiom method*), 101  
 named\_classes() (*owlapy.owl\_ontology\_manager.OWLEquivalentClassesAxiom method*), 182  
 named\_classes() (*owlapy.owl\_ontology.OWLEquivalentClassesAxiom method*), 147  
 named\_individuals (*owlapy.converter.Owl2SparqlConverter attribute*), 79  
 Namespaces (*class in owlapy.iri*), 81  
 Namespaces (*class in owlapy.namespaces*), 85  
 Namespaces (*class in owlapy.parser*), 260  
 Namespaces (*class in owlapy.vocab*), 332  
 new\_count\_var() (*owlapy.converter.Owl2SparqlConverter method*), 80

new\_individual\_variable() (owlapy.converter.VariablesMapping method), 78  
 new\_property\_variable() (owlapy.converter.VariablesMapping method), 78  
 NEXT (owlapy.owl\_reasoner.LRUCache attribute), 231  
 NEXT (owlapy.utils.LRUCache attribute), 329  
 NNF (class in owlapy.utils), 328  
 ns (owlapy.iri.Namespaces property), 81  
 ns (owlapy.namespaces.Namespaces property), 85  
 ns (owlapy.parser.DLSyntaxParser attribute), 276  
 ns (owlapy.parser.ManchesterOWLSyntaxParser attribute), 274  
 ns (owlapy.parser.Namespaces property), 260  
 ns (owlapy.vocab.Namespaces property), 332  
 NUMERIC\_DATATYPES (in module owlapy.owl\_literal), 141

## O

o (owlapy.utils.OrderedOWLObject attribute), 328  
 object\_all\_values\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 326  
 object\_cardinality\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 326  
 object\_complement\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 326  
 object\_has\_self\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 326  
 object\_has\_value\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 326  
 object\_intersection\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 326  
 object\_inverse\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 327  
 object\_one\_of\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 326  
 object\_properties\_in\_signature() (owlapy.owl\_ontology\_manager.Ontology method), 194  
 object\_properties\_in\_signature() (owlapy.owl\_ontology\_manager.OWLOntology method), 192  
 object\_properties\_in\_signature() (owlapy.owl\_ontology.Ontology method), 170  
 object\_properties\_in\_signature() (owlapy.owl\_ontology.OWLOntology method), 168  
 object\_properties\_in\_signature() (owlapy.owl\_reasoner.Ontology method), 223  
 object\_properties\_in\_signature() (owlapy.owl\_reasoner.OWLOntology method), 221  
 object\_property\_domain\_axioms() (owlapy.owl\_ontology\_manager.Ontology method), 195  
 object\_property\_domain\_axioms() (owlapy.owl\_ontology\_manager.OWLOntology method), 193  
 object\_property\_domain\_axioms() (owlapy.owl\_ontology.Ontology method), 171  
 object\_property\_domain\_axioms() (owlapy.owl\_ontology.OWLOntology method), 169  
 object\_property\_domain\_axioms() (owlapy.owl\_reasoner.Ontology method), 224  
 object\_property\_domain\_axioms() (owlapy.owl\_reasoner.OWLOntology method), 222  
 object\_property\_domains() (owlapy.owl\_hierarchy.OWLReasoner method), 124  
 object\_property\_domains() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 246  
 object\_property\_domains() (owlapy.owl\_reasoner.OntologyReasoner method), 240  
 object\_property\_domains() (owlapy.owl\_reasoner.OWLReasoner method), 232  
 object\_property\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 327  
 object\_property\_range\_axioms() (owlapy.owl\_ontology\_manager.Ontology method), 195  
 object\_property\_range\_axioms() (owlapy.owl\_ontology\_manager.OWLOntology method), 193  
 object\_property\_range\_axioms() (owlapy.owl\_ontology.Ontology method), 171  
 object\_property\_range\_axioms() (owlapy.owl\_ontology.OWLOntology method), 169  
 object\_property\_range\_axioms() (owlapy.owl\_reasoner.Ontology method), 224  
 object\_property\_range\_axioms() (owlapy.owl\_reasoner.OWLOntology method), 222  
 object\_property\_ranges() (owlapy.owl\_hierarchy.OWLReasoner method), 124  
 object\_property\_ranges() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 247  
 object\_property\_ranges() (owlapy.owl\_reasoner.OntologyReasoner method), 240  
 object\_property\_ranges() (owlapy.owl\_reasoner.OWLReasoner method), 233  
 object\_property\_values() (owlapy.owl\_hierarchy.OWLReasoner method), 126  
 object\_property\_values() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 248  
 object\_property\_values() (owlapy.owl\_reasoner.OntologyReasoner method), 242  
 object\_property\_values() (owlapy.owl\_reasoner.OWLReasoner method), 234  
 object\_some\_values\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 326  
 object\_union\_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 326  
 ObjectPropertyHierarchy (class in owlapy.owl\_hierarchy), 131  
 Ontology (class in owlapy.owl\_ontology), 170  
 Ontology (class in owlapy.owl\_ontology\_manager), 194  
 Ontology (class in owlapy.owl\_reasoner), 223  
 OntologyManager (class in owlapy.owl\_ontology\_manager), 200  
 OntologyManager (class in owlapy.owl\_reasoner), 225  
 OntologyReasoner (class in owlapy.owl\_reasoner), 239  
 operands() (owlapy.class\_expression.class\_expression.HasOperands method), 17  
 operands() (owlapy.class\_expression.class\_expression.OWLObjectComplementOf method), 19  
 operands() (owlapy.class\_expression.nary\_boolean\_expression.HasOperands method), 21  
 operands() (owlapy.class\_expression.nary\_boolean\_expression.OWLNaryBooleanClassExpression method), 21

- `operands()` (`owlapy.class_expression.owl_class.OwLObjectComplementOf` method), 23
- `operands()` (`owlapy.class_expression.OwLDataOneOf` method), 53
- `operands()` (`owlapy.class_expression.OwLNaryBooleanClassExpression` method), 49
- `operands()` (`owlapy.class_expression.OwLObjectComplementOf` method), 48
- `operands()` (`owlapy.class_expression.OwLObjectOneOf` method), 60
- `operands()` (`owlapy.class_expression.restriction.HasOperands` method), 28
- `operands()` (`owlapy.class_expression.restriction.OwLDataOneOf` method), 44
- `operands()` (`owlapy.class_expression.restriction.OwLObjectOneOf` method), 40
- `operands()` (`owlapy.converter.OwLDataOneOf` method), 73
- `operands()` (`owlapy.converter.OwLObjectComplementOf` method), 68
- `operands()` (`owlapy.converter.OwLObjectOneOf` method), 64
- `operands()` (`owlapy.meta_classes.HasOperands` method), 83
- `operands()` (`owlapy.owl_axiom.HasOperands` method), 91
- `operands()` (`owlapy.owl_axiom.OwLHasKeyAxiom` method), 100
- `operands()` (`owlapy.owl_data_ranges.HasOperands` method), 116
- `operands()` (`owlapy.owl_data_ranges.OwLNaryDataRange` method), 116
- `operands()` (`owlapy.owl_ontology_manager.OwLNaryBooleanClassExpression` method), 179
- `operands()` (`owlapy.owl_ontology_manager.OwLObjectComplementOf` method), 180
- `operands()` (`owlapy.owl_ontology_manager.OwLObjectOneOf` method), 179
- `operands()` (`owlapy.owl_ontology.OwLDataOneOf` method), 162
- `operands()` (`owlapy.owl_ontology.OwLObjectComplementOf` method), 156
- `operands()` (`owlapy.owl_ontology.OwLObjectOneOf` method), 157
- `operands()` (`owlapy.owl_reasoner.OwLDataOneOf` method), 214
- `operands()` (`owlapy.owl_reasoner.OwLObjectComplementOf` method), 211
- `operands()` (`owlapy.owl_reasoner.OwLObjectOneOf` method), 212
- `operands()` (`owlapy.parser.OwLDataOneOf` method), 271
- `operands()` (`owlapy.parser.OwLObjectComplementOf` method), 266
- `operands()` (`owlapy.parser.OwLObjectOneOf` method), 265
- `operands()` (`owlapy.render.OwLDataOneOf` method), 298
- `operands()` (`owlapy.render.OwLNaryBooleanClassExpression` method), 298
- `operands()` (`owlapy.render.OwLNaryDataRange` method), 300
- `operands()` (`owlapy.render.OwLObjectComplementOf` method), 294
- `operands()` (`owlapy.render.OwLObjectOneOf` method), 302
- `operands()` (`owlapy.utils.HasOperands` method), 308
- `operands()` (`owlapy.utils.OwLDataOneOf` method), 320
- `operands()` (`owlapy.utils.OwLNaryBooleanClassExpression` method), 321
- `operands()` (`owlapy.utils.OwLNaryDataRange` method), 324
- `operands()` (`owlapy.utils.OwLObjectComplementOf` method), 314
- `operands()` (`owlapy.utils.OwLObjectOneOf` method), 323
- `OperandSetTransform` (class in `owlapy.utils`), 328
- `operator` (`owlapy.class_expression.OwLFacet` property), 56
- `operator` (`owlapy.class_expression.restriction.OwLFacet` property), 34
- `operator` (`owlapy.converter.OwLFacet` property), 77
- `operator` (`owlapy.owl_ontology.OwLFacet` property), 167
- `operator` (`owlapy.parser.OwLFacet` property), 261
- `operator` (`owlapy.providers.OwLFacet` property), 282
- `operator` (`owlapy.render.OwLFacet` property), 299
- `operator` (`owlapy.vocab.OwLFacet` property), 333
- `OrderedOwLObject` (class in `owlapy.utils`), 328
- `OwL` (in module `owlapy.namespaces`), 85
- `OwL2SparqlConverter` (class in `owlapy.converter`), 79
- `OwL_BOTTOM_DATA_PROPERTY` (`owlapy.class_expression.OwLRDFVocabulary` attribute), 61
- `OwL_BOTTOM_DATA_PROPERTY` (`owlapy.converter.OwLRDFVocabulary` attribute), 78
- `OwL_BOTTOM_DATA_PROPERTY` (`owlapy.owl_literal.OwLRDFVocabulary` attribute), 138
- `OwL_BOTTOM_DATA_PROPERTY` (`owlapy.parser.OwLRDFVocabulary` attribute), 262
- `OwL_BOTTOM_DATA_PROPERTY` (`owlapy.vocab.OwLRDFVocabulary` attribute), 332
- `OwL_BOTTOM_OBJECT_PROPERTY` (`owlapy.class_expression.OwLRDFVocabulary` attribute), 61
- `OwL_BOTTOM_OBJECT_PROPERTY` (`owlapy.converter.OwLRDFVocabulary` attribute), 78
- `OwL_BOTTOM_OBJECT_PROPERTY` (`owlapy.owl_literal.OwLRDFVocabulary` attribute), 138
- `OwL_BOTTOM_OBJECT_PROPERTY` (`owlapy.parser.OwLRDFVocabulary` attribute), 262
- `OwL_BOTTOM_OBJECT_PROPERTY` (`owlapy.vocab.OwLRDFVocabulary` attribute), 332
- `OwL_CLASS` (`owlapy.class_expression.OwLRDFVocabulary` attribute), 61
- `OwL_CLASS` (`owlapy.converter.OwLRDFVocabulary` attribute), 78
- `OwL_CLASS` (`owlapy.owl_literal.OwLRDFVocabulary` attribute), 137
- `OwL_CLASS` (`owlapy.parser.OwLRDFVocabulary` attribute), 262
- `OwL_CLASS` (`owlapy.vocab.OwLRDFVocabulary` attribute), 332
- `owl_datatype_max_exclusive_restriction()` (in module `owlapy.providers`), 282
- `owl_datatype_max_inclusive_restriction()` (in module `owlapy.providers`), 282

`owl_datatype_min_exclusive_restriction()` (in module `owlapy.providers`), 282  
`owl_datatype_min_inclusive_restriction()` (in module `owlapy.providers`), 283  
`owl_datatype_min_max_exclusive_restriction()` (in module `owlapy.providers`), 283  
`owl_datatype_min_max_inclusive_restriction()` (in module `owlapy.providers`), 283  
`owl_expression_to_dl()` (in module `owlapy`), 334  
`owl_expression_to_dl()` (in module `owlapy.render`), 304  
`owl_expression_to_manchester()` (in module `owlapy`), 334  
`owl_expression_to_manchester()` (in module `owlapy.render`), 304  
`owl_expression_to_sparql()` (in module `owlapy`), 334  
`owl_expression_to_sparql()` (in module `owlapy.converter`), 80  
`OWL_NAMED_INDIVIDUAL` (`owlapy.class_expression.OWLRDFVocabulary` attribute), 61  
`OWL_NAMED_INDIVIDUAL` (`owlapy.converter.OWLRDFVocabulary` attribute), 78  
`OWL_NAMED_INDIVIDUAL` (`owlapy.owl_literal.OWLRDFVocabulary` attribute), 137  
`OWL_NAMED_INDIVIDUAL` (`owlapy.parser.OWLRDFVocabulary` attribute), 262  
`OWL_NAMED_INDIVIDUAL` (`owlapy.vocab.OWLRDFVocabulary` attribute), 332  
`OWL_NOTHING` (`owlapy.class_expression.OWLRDFVocabulary` attribute), 61  
`OWL_NOTHING` (`owlapy.converter.OWLRDFVocabulary` attribute), 78  
`OWL_NOTHING` (`owlapy.owl_literal.OWLRDFVocabulary` attribute), 137  
`OWL_NOTHING` (`owlapy.parser.OWLRDFVocabulary` attribute), 262  
`OWL_NOTHING` (`owlapy.vocab.OWLRDFVocabulary` attribute), 332  
`OWL_THING` (`owlapy.class_expression.OWLRDFVocabulary` attribute), 61  
`OWL_THING` (`owlapy.converter.OWLRDFVocabulary` attribute), 78  
`OWL_THING` (`owlapy.owl_literal.OWLRDFVocabulary` attribute), 137  
`OWL_THING` (`owlapy.parser.OWLRDFVocabulary` attribute), 262  
`OWL_THING` (`owlapy.vocab.OWLRDFVocabulary` attribute), 332  
`OWL_TOP_DATA_PROPERTY` (`owlapy.class_expression.OWLRDFVocabulary` attribute), 61  
`OWL_TOP_DATA_PROPERTY` (`owlapy.converter.OWLRDFVocabulary` attribute), 78  
`OWL_TOP_DATA_PROPERTY` (`owlapy.owl_literal.OWLRDFVocabulary` attribute), 138  
`OWL_TOP_DATA_PROPERTY` (`owlapy.parser.OWLRDFVocabulary` attribute), 262  
`OWL_TOP_DATA_PROPERTY` (`owlapy.vocab.OWLRDFVocabulary` attribute), 332  
`OWL_TOP_OBJECT_PROPERTY` (`owlapy.class_expression.OWLRDFVocabulary` attribute), 61  
`OWL_TOP_OBJECT_PROPERTY` (`owlapy.converter.OWLRDFVocabulary` attribute), 78  
`OWL_TOP_OBJECT_PROPERTY` (`owlapy.owl_literal.OWLRDFVocabulary` attribute), 137  
`OWL_TOP_OBJECT_PROPERTY` (`owlapy.parser.OWLRDFVocabulary` attribute), 262  
`OWL_TOP_OBJECT_PROPERTY` (`owlapy.vocab.OWLRDFVocabulary` attribute), 332  
`OWLAnnotation` (class in `owlapy.owl_axiom`), 106  
`OWLAnnotationAssertionAxiom` (class in `owlapy.owl_axiom`), 107  
`OWLAnnotationAssertionAxiom` (class in `owlapy.owl_ontology_manager`), 183  
`OWLAnnotationAxiom` (class in `owlapy.owl_axiom`), 106  
`OWLAnnotationObject` (class in `owlapy.owl_annotation`), 86  
`OWLAnnotationProperty` (class in `owlapy.owl_axiom`), 106  
`OWLAnnotationProperty` (class in `owlapy.owl_ontology`), 148  
`OWLAnnotationProperty` (class in `owlapy.owl_ontology_manager`), 183  
`OWLAnnotationPropertyDomainAxiom` (class in `owlapy.owl_axiom`), 108  
`OWLAnnotationPropertyRangeAxiom` (class in `owlapy.owl_axiom`), 108  
`OWLAnnotationSubject` (class in `owlapy.iri`), 81  
`OWLAnnotationSubject` (class in `owlapy.owl_annotation`), 86  
`OWLAnnotationSubject` (class in `owlapy.owl_axiom`), 95  
`OWLAnnotationValue` (class in `owlapy.iri`), 81  
`OWLAnnotationValue` (class in `owlapy.owl_annotation`), 86  
`OWLAnnotationValue` (class in `owlapy.owl_axiom`), 95  
`OWLAnnotationValue` (class in `owlapy.owl_literal`), 137  
`OWLAnonymousClassExpression` (class in `owlapy.class_expression`), 47  
`OWLAnonymousClassExpression` (class in `owlapy.class_expression.class_expression`), 18  
`OWLAnonymousClassExpression` (class in `owlapy.class_expression.restriction`), 29  
`owlapy`  
    module, 17  
`owlapy.class_expression`  
    module, 17  
`owlapy.class_expression.class_expression`  
    module, 17  
`owlapy.class_expression.nary_boolean_expression`  
    module, 20  
`owlapy.class_expression.owl_class`  
    module, 22  
`owlapy.class_expression.restriction`  
    module, 26  
`owlapy.converter`



- module, 62
- owlapy.entities
  - module, 62
- owlapy.iri
  - module, 80
- owlapy.meta\_classes
  - module, 83
- owlapy.namespaces
  - module, 84
- owlapy.owl\_annotation
  - module, 85
- owlapy.owl\_axiom
  - module, 86
- owlapy.owl\_data\_ranges
  - module, 115
- owlapy.owl\_datatype
  - module, 117
- owlapy.owl\_hierarchy
  - module, 120
- owlapy.owl\_individual
  - module, 132
- owlapy.owl\_literal
  - module, 135
- owlapy.owl\_object
  - module, 141
- owlapy.owl\_ontology
  - module, 144
- owlapy.owl\_ontology\_manager
  - module, 173
- owlapy.owl\_property
  - module, 201
- owlapy.owl\_reasoner
  - module, 207
- owlapy.parser
  - module, 253
- owlapy.providers
  - module, 279
- owlapy.render
  - module, 283
- owlapy.static\_funcs
  - module, 304
- owlapy.utils
  - module, 305
- owlapy.vocab
  - module, 330
- OWLAsymmetricObjectPropertyAxiom (*class in owlapy.owl\_axiom*), 111
- OWLAsymmetricObjectPropertyAxiom (*class in owlapy.owl\_ontology\_manager*), 186
- OWLAxiom (*class in owlapy.owl\_axiom*), 98
- OWLAxiom (*class in owlapy.owl\_ontology\_manager*), 181
- OWLAxiom (*class in owlapy.owl\_reasoner*), 219
- OWLBooleanClassExpression (*class in owlapy.class\_expression*), 48
- OWLBooleanClassExpression (*class in owlapy.class\_expression.class\_expression*), 19
- OWLBooleanClassExpression (*class in owlapy.class\_expression.nary\_boolean\_expression*), 21
- OWLBooleanClassExpression (*class in owlapy.render*), 291
- OWLBottomDataProperty (*in module owlapy.owl\_hierarchy*), 122
- OWLBottomDataProperty (*in module owlapy.owl\_literal*), 141
- OWLBottomObjectProperty (*in module owlapy.owl\_hierarchy*), 122
- OWLBottomObjectProperty (*in module owlapy.owl\_literal*), 141
- OWLCardinalityRestriction (*class in owlapy.class\_expression*), 52
- OWLCardinalityRestriction (*class in owlapy.class\_expression.restriction*), 35
- OWLClass (*class in owlapy.class\_expression*), 48
- OWLClass (*class in owlapy.class\_expression.owl\_class*), 25
- OWLClass (*class in owlapy.converter*), 66
- OWLClass (*class in owlapy.owl\_axiom*), 93
- OWLClass (*class in owlapy.owl\_hierarchy*), 121
- OWLClass (*class in owlapy.owl\_ontology*), 154
- OWLClass (*class in owlapy.owl\_ontology\_manager*), 177
- OWLClass (*class in owlapy.owl\_reasoner*), 217

OWLClass (class in owlapy.parser), 264  
 OWLClass (class in owlapy.render), 291  
 OWLClass (class in owlapy.utils), 313  
 OWLClassAssertionAxiom (class in owlapy.owl\_axiom), 105  
 OWLClassAssertionAxiom (class in owlapy.owl\_ontology\_manager), 184  
 OWLClassAxiom (class in owlapy.owl\_axiom), 99  
 OWLClassAxiom (class in owlapy.owl\_ontology), 147  
 OWLClassExpression (class in owlapy.class\_expression), 47  
 OWLClassExpression (class in owlapy.class\_expression.class\_expression), 18  
 OWLClassExpression (class in owlapy.class\_expression.nary\_boolean\_expression), 20  
 OWLClassExpression (class in owlapy.class\_expression.owl\_class), 22  
 OWLClassExpression (class in owlapy.class\_expression.restriction), 29  
 OWLClassExpression (class in owlapy.converter), 67  
 OWLClassExpression (class in owlapy.owl\_axiom), 92  
 OWLClassExpression (class in owlapy.owl\_ontology), 155  
 OWLClassExpression (class in owlapy.owl\_reasoner), 209  
 OWLClassExpression (class in owlapy.parser), 265  
 OWLClassExpression (class in owlapy.render), 290  
 OWLClassExpression (class in owlapy.utils), 312  
 OWLClassExpressionLengthMetric (class in owlapy.utils), 325  
 OWLDataAllValuesFrom (class in owlapy.class\_expression), 58  
 OWLDataAllValuesFrom (class in owlapy.class\_expression.restriction), 42  
 OWLDataAllValuesFrom (class in owlapy.converter), 72  
 OWLDataAllValuesFrom (class in owlapy.owl\_ontology), 160  
 OWLDataAllValuesFrom (class in owlapy.owl\_reasoner), 216  
 OWLDataAllValuesFrom (class in owlapy.parser), 273  
 OWLDataAllValuesFrom (class in owlapy.render), 296  
 OWLDataAllValuesFrom (class in owlapy.utils), 318  
 OWLDataCardinalityRestriction (class in owlapy.class\_expression), 54  
 OWLDataCardinalityRestriction (class in owlapy.class\_expression.restriction), 40  
 OWLDataCardinalityRestriction (class in owlapy.converter), 71  
 OWLDataCardinalityRestriction (class in owlapy.parser), 272  
 OWLDataCardinalityRestriction (class in owlapy.utils), 320  
 OWLDataComplementOf (class in owlapy.owl\_data\_ranges), 117  
 OWLDataComplementOf (class in owlapy.owl\_ontology), 149  
 OWLDataComplementOf (class in owlapy.owl\_reasoner), 220  
 OWLDataComplementOf (class in owlapy.parser), 274  
 OWLDataComplementOf (class in owlapy.render), 300  
 OWLDataComplementOf (class in owlapy.utils), 323  
 OWLDataExactCardinality (class in owlapy.class\_expression), 60  
 OWLDataExactCardinality (class in owlapy.class\_expression.restriction), 41  
 OWLDataExactCardinality (class in owlapy.converter), 66  
 OWLDataExactCardinality (class in owlapy.owl\_ontology), 160  
 OWLDataExactCardinality (class in owlapy.parser), 268  
 OWLDataExactCardinality (class in owlapy.render), 297  
 OWLDataExactCardinality (class in owlapy.utils), 317  
 OWLDataHasValue (class in owlapy.class\_expression), 59  
 OWLDataHasValue (class in owlapy.class\_expression.restriction), 43  
 OWLDataHasValue (class in owlapy.converter), 73  
 OWLDataHasValue (class in owlapy.owl\_ontology), 161  
 OWLDataHasValue (class in owlapy.owl\_ontology\_manager), 178  
 OWLDataHasValue (class in owlapy.owl\_reasoner), 216  
 OWLDataHasValue (class in owlapy.parser), 270  
 OWLDataHasValue (class in owlapy.render), 296  
 OWLDataHasValue (class in owlapy.utils), 317  
 OWLDataIntersectionOf (class in owlapy.owl\_data\_ranges), 116  
 OWLDataIntersectionOf (class in owlapy.owl\_ontology), 149  
 OWLDataIntersectionOf (class in owlapy.owl\_reasoner), 220  
 OWLDataIntersectionOf (class in owlapy.parser), 273  
 OWLDataIntersectionOf (class in owlapy.render), 300  
 OWLDataIntersectionOf (class in owlapy.utils), 324  
 OWLDataMaxCardinality (class in owlapy.class\_expression), 60  
 OWLDataMaxCardinality (class in owlapy.class\_expression.restriction), 41  
 OWLDataMaxCardinality (class in owlapy.converter), 66  
 OWLDataMaxCardinality (class in owlapy.owl\_ontology), 161  
 OWLDataMaxCardinality (class in owlapy.parser), 269  
 OWLDataMaxCardinality (class in owlapy.render), 297  
 OWLDataMaxCardinality (class in owlapy.utils), 317

OWLDataMinCardinality (class in owlapy.class\_expression), 59  
 OWLDataMinCardinality (class in owlapy.class\_expression.restriction), 41  
 OWLDataMinCardinality (class in owlapy.converter), 65  
 OWLDataMinCardinality (class in owlapy.owl\_ontology), 161  
 OWLDataMinCardinality (class in owlapy.parser), 269  
 OWLDataMinCardinality (class in owlapy.render), 297  
 OWLDataMinCardinality (class in owlapy.utils), 317  
 OWLDataOneOf (class in owlapy.class\_expression), 53  
 OWLDataOneOf (class in owlapy.class\_expression.restriction), 43  
 OWLDataOneOf (class in owlapy.converter), 73  
 OWLDataOneOf (class in owlapy.owl\_ontology), 162  
 OWLDataOneOf (class in owlapy.owl\_reasoner), 214  
 OWLDataOneOf (class in owlapy.parser), 270  
 OWLDataOneOf (class in owlapy.render), 298  
 OWLDataOneOf (class in owlapy.utils), 319  
 OWLDataProperty (class in owlapy.converter), 76  
 OWLDataProperty (class in owlapy.owl\_hierarchy), 123  
 OWLDataProperty (class in owlapy.owl\_literal), 139  
 OWLDataProperty (class in owlapy.owl\_ontology), 164  
 OWLDataProperty (class in owlapy.owl\_ontology\_manager), 196  
 OWLDataProperty (class in owlapy.owl\_property), 206  
 OWLDataProperty (class in owlapy.owl\_reasoner), 227  
 OWLDataProperty (class in owlapy.parser), 260  
 OWLDataProperty (class in owlapy.utils), 312  
 OWLDataPropertyAssertionAxiom (class in owlapy.owl\_axiom), 110  
 OWLDataPropertyAssertionAxiom (class in owlapy.owl\_ontology\_manager), 187  
 OWLDataPropertyAxiom (class in owlapy.owl\_axiom), 98  
 OWLDataPropertyCharacteristicAxiom (class in owlapy.owl\_axiom), 113  
 OWLDataPropertyCharacteristicAxiom (class in owlapy.owl\_ontology\_manager), 187  
 OWLDataPropertyDomainAxiom (class in owlapy.owl\_axiom), 114  
 OWLDataPropertyDomainAxiom (class in owlapy.owl\_ontology), 147  
 OWLDataPropertyExpression (class in owlapy.class\_expression.restriction), 31  
 OWLDataPropertyExpression (class in owlapy.owl\_axiom), 90  
 OWLDataPropertyExpression (class in owlapy.owl\_ontology), 166  
 OWLDataPropertyExpression (class in owlapy.owl\_property), 205  
 OWLDataPropertyExpression (class in owlapy.owl\_reasoner), 229  
 OWLDataPropertyRangeAxiom (class in owlapy.owl\_axiom), 115  
 OWLDataPropertyRangeAxiom (class in owlapy.owl\_ontology), 147  
 OWLDataRange (class in owlapy.class\_expression.restriction), 31  
 OWLDataRange (class in owlapy.owl\_axiom), 91  
 OWLDataRange (class in owlapy.owl\_data\_ranges), 116  
 OWLDataRange (class in owlapy.owl\_datatype), 118  
 OWLDataRange (class in owlapy.owl\_ontology), 149  
 OWLDataRange (class in owlapy.owl\_reasoner), 220  
 OWLDataRange (class in owlapy.parser), 274  
 OWLDataRange (class in owlapy.utils), 324  
 OWLDataRestriction (class in owlapy.class\_expression), 51  
 OWLDataRestriction (class in owlapy.class\_expression.restriction), 40  
 OWLDataRestriction (class in owlapy.owl\_ontology), 163  
 OWLDataSomeValuesFrom (class in owlapy.class\_expression), 58  
 OWLDataSomeValuesFrom (class in owlapy.class\_expression.restriction), 42  
 OWLDataSomeValuesFrom (class in owlapy.converter), 71  
 OWLDataSomeValuesFrom (class in owlapy.owl\_ontology), 159  
 OWLDataSomeValuesFrom (class in owlapy.owl\_reasoner), 214  
 OWLDataSomeValuesFrom (class in owlapy.parser), 268  
 OWLDataSomeValuesFrom (class in owlapy.render), 295  
 OWLDataSomeValuesFrom (class in owlapy.utils), 319  
 OWLDatatype (class in owlapy.class\_expression.restriction), 33  
 OWLDatatype (class in owlapy.converter), 77  
 OWLDatatype (class in owlapy.owl\_axiom), 91  
 OWLDatatype (class in owlapy.owl\_datatype), 120  
 OWLDatatype (class in owlapy.owl\_literal), 137  
 OWLDatatype (class in owlapy.owl\_ontology), 149  
 OWLDatatype (class in owlapy.owl\_reasoner), 220  
 OWLDatatype (class in owlapy.parser), 261  
 OWLDatatype (class in owlapy.render), 302  
 OWLDatatype (class in owlapy.utils), 325  
 OWLDatatypeDefinitionAxiom (class in owlapy.owl\_axiom), 99



OWLDatatypeRestriction (class in owlapy.class\_expression), 56  
 OWLDatatypeRestriction (class in owlapy.class\_expression.restriction), 44  
 OWLDatatypeRestriction (class in owlapy.converter), 65  
 OWLDatatypeRestriction (class in owlapy.owl\_ontology), 162  
 OWLDatatypeRestriction (class in owlapy.owl\_reasoner), 215  
 OWLDatatypeRestriction (class in owlapy.parser), 271  
 OWLDatatypeRestriction (class in owlapy.providers), 281  
 OWLDatatypeRestriction (class in owlapy.render), 301  
 OWLDatatypeRestriction (class in owlapy.utils), 322  
 OWLDataUnionOf (class in owlapy.owl\_data\_ranges), 117  
 OWLDataUnionOf (class in owlapy.owl\_ontology), 149  
 OWLDataUnionOf (class in owlapy.owl\_reasoner), 220  
 OWLDataUnionOf (class in owlapy.parser), 273  
 OWLDataUnionOf (class in owlapy.render), 300  
 OWLDataUnionOf (class in owlapy.utils), 324  
 OWLDeclarationAxiom (class in owlapy.owl\_axiom), 99  
 OWLDeclarationAxiom (class in owlapy.owl\_ontology\_manager), 185  
 OWLDifferentIndividualsAxiom (class in owlapy.owl\_axiom), 102  
 OWLDifferentIndividualsAxiom (class in owlapy.owl\_ontology\_manager), 191  
 OWLDisjointClassesAxiom (class in owlapy.owl\_axiom), 101  
 OWLDisjointClassesAxiom (class in owlapy.owl\_ontology\_manager), 191  
 OWLDisjointDataPropertiesAxiom (class in owlapy.owl\_axiom), 104  
 OWLDisjointDataPropertiesAxiom (class in owlapy.owl\_ontology\_manager), 188  
 OWLDisjointObjectPropertiesAxiom (class in owlapy.owl\_axiom), 103  
 OWLDisjointObjectPropertiesAxiom (class in owlapy.owl\_ontology\_manager), 189  
 OWLDisjointUnionAxiom (class in owlapy.owl\_axiom), 105  
 OWLDisjointUnionAxiom (class in owlapy.owl\_ontology\_manager), 182  
 OWLEntity (class in owlapy.class\_expression.owl\_class), 23  
 OWLEntity (class in owlapy.converter), 77  
 OWLEntity (class in owlapy.owl\_axiom), 90  
 OWLEntity (class in owlapy.owl\_datatype), 118  
 OWLEntity (class in owlapy.owl\_individual), 133  
 OWLEntity (class in owlapy.owl\_object), 143  
 OWLEntity (class in owlapy.owl\_property), 202  
 OWLEntity (class in owlapy.render), 289  
 OWLEquivalentClassesAxiom (class in owlapy.owl\_axiom), 101  
 OWLEquivalentClassesAxiom (class in owlapy.owl\_ontology), 146  
 OWLEquivalentClassesAxiom (class in owlapy.owl\_ontology\_manager), 182  
 OWLEquivalentDataPropertiesAxiom (class in owlapy.owl\_axiom), 104  
 OWLEquivalentDataPropertiesAxiom (class in owlapy.owl\_ontology\_manager), 189  
 OWLEquivalentObjectPropertiesAxiom (class in owlapy.owl\_axiom), 103  
 OWLEquivalentObjectPropertiesAxiom (class in owlapy.owl\_ontology\_manager), 189  
 OWLFacet (class in owlapy.class\_expression), 56  
 OWLFacet (class in owlapy.class\_expression.restriction), 34  
 OWLFacet (class in owlapy.converter), 77  
 OWLFacet (class in owlapy.owl\_ontology), 167  
 OWLFacet (class in owlapy.parser), 261  
 OWLFacet (class in owlapy.providers), 281  
 OWLFacet (class in owlapy.render), 299  
 OWLFacet (class in owlapy.vocab), 333  
 OWLFacetRestriction (class in owlapy.class\_expression), 57  
 OWLFacetRestriction (class in owlapy.class\_expression.restriction), 44  
 OWLFacetRestriction (class in owlapy.owl\_ontology), 164  
 OWLFacetRestriction (class in owlapy.owl\_reasoner), 215  
 OWLFacetRestriction (class in owlapy.parser), 267  
 OWLFacetRestriction (class in owlapy.providers), 282  
 OWLFacetRestriction (class in owlapy.render), 301  
 OWLFacetRestriction (class in owlapy.utils), 322  
 OWLFunctionalDataPropertyAxiom (class in owlapy.owl\_axiom), 113  
 OWLFunctionalDataPropertyAxiom (class in owlapy.owl\_ontology\_manager), 187  
 OWLFunctionalObjectPropertyAxiom (class in owlapy.owl\_axiom), 111  
 OWLFunctionalObjectPropertyAxiom (class in owlapy.owl\_ontology\_manager), 188  
 OWLHasKeyAxiom (class in owlapy.owl\_axiom), 100  
 OWLHasValueRestriction (class in owlapy.class\_expression), 51  
 OWLHasValueRestriction (class in owlapy.class\_expression.restriction), 35  
 OWLImportsDeclaration (class in owlapy.owl\_ontology\_manager), 199  
 OWLIndividual (class in owlapy.class\_expression.restriction), 33  
 OWLIndividual (class in owlapy.owl\_axiom), 94

OWLIndividual (class in owlapy.owl\_individual), 134  
 OWLIndividual (class in owlapy.owl\_ontology), 150  
 OWLIndividual (class in owlapy.owl\_ontology\_manager), 192  
 OWLIndividualAxiom (class in owlapy.owl\_axiom), 98  
 OWLInverseFunctionalObjectPropertyAxiom (class in owlapy.owl\_axiom), 111  
 OWLInverseFunctionalObjectPropertyAxiom (class in owlapy.owl\_ontology\_manager), 188  
 OWLInverseObjectPropertiesAxiom (class in owlapy.owl\_axiom), 103  
 OWLInverseObjectPropertiesAxiom (class in owlapy.owl\_ontology\_manager), 189  
 OWLIrreflexiveObjectPropertyAxiom (class in owlapy.owl\_axiom), 112  
 OWLIrreflexiveObjectPropertyAxiom (class in owlapy.owl\_ontology\_manager), 188  
 OWLLiteral (class in owlapy.class\_expression.restriction), 31  
 OWLLiteral (class in owlapy.converter), 74  
 OWLLiteral (class in owlapy.owl\_axiom), 96  
 OWLLiteral (class in owlapy.owl\_literal), 139  
 OWLLiteral (class in owlapy.owl\_ontology), 151  
 OWLLiteral (class in owlapy.owl\_reasoner), 229  
 OWLLiteral (class in owlapy.parser), 257  
 OWLLiteral (class in owlapy.providers), 279  
 OWLLiteral (class in owlapy.render), 287  
 OWLLiteral (class in owlapy.utils), 309  
 OWLLogicalAxiom (class in owlapy.owl\_axiom), 98  
 OWLNamedIndividual (class in owlapy.converter), 74  
 OWLNamedIndividual (class in owlapy.owl\_individual), 135  
 OWLNamedIndividual (class in owlapy.owl\_ontology), 150  
 OWLNamedIndividual (class in owlapy.owl\_ontology\_manager), 191  
 OWLNamedIndividual (class in owlapy.owl\_reasoner), 229  
 OWLNamedIndividual (class in owlapy.parser), 256  
 OWLNamedIndividual (class in owlapy.render), 286  
 OWLNamedIndividual (class in owlapy.utils), 307  
 OWLNamedObject (class in owlapy.owl\_object), 143  
 OWLNaryAxiom (class in owlapy.owl\_axiom), 100  
 OWLNaryBooleanClassExpression (class in owlapy.class\_expression), 49  
 OWLNaryBooleanClassExpression (class in owlapy.class\_expression.nary\_boolean\_expression), 21  
 OWLNaryBooleanClassExpression (class in owlapy.owl\_ontology\_manager), 179  
 OWLNaryBooleanClassExpression (class in owlapy.render), 298  
 OWLNaryBooleanClassExpression (class in owlapy.utils), 321  
 OWLNaryClassAxiom (class in owlapy.owl\_axiom), 100  
 OWLNaryDataRange (class in owlapy.owl\_data\_ranges), 116  
 OWLNaryDataRange (class in owlapy.render), 299  
 OWLNaryDataRange (class in owlapy.utils), 324  
 OWLNaryIndividualAxiom (class in owlapy.owl\_axiom), 102  
 OWLNaryIndividualAxiom (class in owlapy.owl\_ontology\_manager), 190  
 OWLNaryPropertyAxiom (class in owlapy.owl\_axiom), 102  
 OWLNaryPropertyAxiom (class in owlapy.owl\_ontology\_manager), 190  
 OWLNegativeDataPropertyAssertionAxiom (class in owlapy.owl\_axiom), 110  
 OWLNegativeObjectPropertyAssertionAxiom (class in owlapy.owl\_axiom), 110  
 OWLNothing (in module owlapy.class\_expression), 61  
 OWLNothing (in module owlapy.owl\_axiom), 94  
 OWLNothing (in module owlapy.owl\_hierarchy), 122  
 OWLNothing (in module owlapy.utils), 314  
 OWLObject (class in owlapy.class\_expression.restriction), 33  
 OWLObject (class in owlapy.owl\_annotation), 85  
 OWLObject (class in owlapy.owl\_axiom), 90  
 OWLObject (class in owlapy.owl\_data\_ranges), 115  
 OWLObject (class in owlapy.owl\_individual), 133  
 OWLObject (class in owlapy.owl\_object), 142  
 OWLObject (class in owlapy.owl\_ontology), 152  
 OWLObject (class in owlapy.owl\_ontology\_manager), 176  
 OWLObject (class in owlapy.owl\_property), 202  
 OWLObject (class in owlapy.render), 289  
 OWLObject (class in owlapy.utils), 325  
 OWLObjectAllValuesFrom (class in owlapy.class\_expression), 55  
 OWLObjectAllValuesFrom (class in owlapy.class\_expression.restriction), 38  
 OWLObjectAllValuesFrom (class in owlapy.converter), 69  
 OWLObjectAllValuesFrom (class in owlapy.owl\_ontology), 157  
 OWLObjectAllValuesFrom (class in owlapy.owl\_reasoner), 211  
 OWLObjectAllValuesFrom (class in owlapy.parser), 272  
 OWLObjectAllValuesFrom (class in owlapy.render), 293

OWLObjectAllValuesFrom (class in owlapy.utils), 319  
 OWLObjectCardinalityRestriction (class in owlapy.class\_expression), 52  
 OWLObjectCardinalityRestriction (class in owlapy.class\_expression.restriction), 36  
 OWLObjectCardinalityRestriction (class in owlapy.converter), 69  
 OWLObjectCardinalityRestriction (class in owlapy.owl\_reasoner), 213  
 OWLObjectCardinalityRestriction (class in owlapy.parser), 271  
 OWLObjectCardinalityRestriction (class in owlapy.utils), 313  
 OWLObjectComplementOf (class in owlapy.class\_expression), 48  
 OWLObjectComplementOf (class in owlapy.class\_expression.class\_expression), 19  
 OWLObjectComplementOf (class in owlapy.class\_expression.owl\_class), 23  
 OWLObjectComplementOf (class in owlapy.converter), 68  
 OWLObjectComplementOf (class in owlapy.owl\_ontology), 155  
 OWLObjectComplementOf (class in owlapy.owl\_ontology\_manager), 180  
 OWLObjectComplementOf (class in owlapy.owl\_reasoner), 210  
 OWLObjectComplementOf (class in owlapy.parser), 266  
 OWLObjectComplementOf (class in owlapy.render), 293  
 OWLObjectComplementOf (class in owlapy.utils), 314  
 OWLObjectExactCardinality (class in owlapy.class\_expression), 57  
 OWLObjectExactCardinality (class in owlapy.class\_expression.restriction), 37  
 OWLObjectExactCardinality (class in owlapy.converter), 70  
 OWLObjectExactCardinality (class in owlapy.owl\_ontology), 158  
 OWLObjectExactCardinality (class in owlapy.owl\_reasoner), 213  
 OWLObjectExactCardinality (class in owlapy.parser), 266  
 OWLObjectExactCardinality (class in owlapy.render), 294  
 OWLObjectExactCardinality (class in owlapy.utils), 316  
 OWLObjectHasSelf (class in owlapy.class\_expression), 52  
 OWLObjectHasSelf (class in owlapy.class\_expression.restriction), 38  
 OWLObjectHasSelf (class in owlapy.converter), 71  
 OWLObjectHasSelf (class in owlapy.parser), 262  
 OWLObjectHasSelf (class in owlapy.render), 295  
 OWLObjectHasSelf (class in owlapy.utils), 316  
 OWLObjectHasValue (class in owlapy.class\_expression), 55  
 OWLObjectHasValue (class in owlapy.class\_expression.restriction), 39  
 OWLObjectHasValue (class in owlapy.converter), 64  
 OWLObjectHasValue (class in owlapy.owl\_ontology), 159  
 OWLObjectHasValue (class in owlapy.owl\_ontology\_manager), 180  
 OWLObjectHasValue (class in owlapy.owl\_reasoner), 212  
 OWLObjectHasValue (class in owlapy.parser), 268  
 OWLObjectHasValue (class in owlapy.render), 301  
 OWLObjectHasValue (class in owlapy.utils), 321  
 OWLObjectIntersectionOf (class in owlapy.class\_expression), 50  
 OWLObjectIntersectionOf (class in owlapy.class\_expression.nary\_boolean\_expression), 21  
 OWLObjectIntersectionOf (class in owlapy.class\_expression.restriction), 28  
 OWLObjectIntersectionOf (class in owlapy.converter), 68  
 OWLObjectIntersectionOf (class in owlapy.owl\_ontology), 156  
 OWLObjectIntersectionOf (class in owlapy.owl\_reasoner), 210  
 OWLObjectIntersectionOf (class in owlapy.parser), 262  
 OWLObjectIntersectionOf (class in owlapy.render), 293  
 OWLObjectIntersectionOf (class in owlapy.utils), 320  
 OWLObjectInverseOf (class in owlapy.owl\_ontology), 165  
 OWLObjectInverseOf (class in owlapy.owl\_ontology\_manager), 196  
 OWLObjectInverseOf (class in owlapy.owl\_property), 206  
 OWLObjectInverseOf (class in owlapy.owl\_reasoner), 228  
 OWLObjectInverseOf (class in owlapy.render), 289  
 OWLObjectInverseOf (class in owlapy.utils), 310  
 OWLObjectMaxCardinality (class in owlapy.class\_expression), 57  
 OWLObjectMaxCardinality (class in owlapy.class\_expression.restriction), 37  
 OWLObjectMaxCardinality (class in owlapy.converter), 70  
 OWLObjectMaxCardinality (class in owlapy.owl\_ontology), 158  
 OWLObjectMaxCardinality (class in owlapy.owl\_reasoner), 213  
 OWLObjectMaxCardinality (class in owlapy.parser), 269  
 OWLObjectMaxCardinality (class in owlapy.render), 294  
 OWLObjectMaxCardinality (class in owlapy.utils), 316  
 OWLObjectMinCardinality (class in owlapy.class\_expression), 57  
 OWLObjectMinCardinality (class in owlapy.class\_expression.restriction), 37  
 OWLObjectMinCardinality (class in owlapy.converter), 70  
 OWLObjectMinCardinality (class in owlapy.owl\_ontology), 158  
 OWLObjectMinCardinality (class in owlapy.owl\_reasoner), 212

OWLObjectMinCardinality (class in owlapy.parser), 263  
 OWLObjectMinCardinality (class in owlapy.render), 294  
 OWLObjectMinCardinality (class in owlapy.utils), 315  
 OWLObjectOneOf (class in owlapy.class\_expression), 60  
 OWLObjectOneOf (class in owlapy.class\_expression.restriction), 39  
 OWLObjectOneOf (class in owlapy.converter), 64  
 OWLObjectOneOf (class in owlapy.owl\_ontology), 157  
 OWLObjectOneOf (class in owlapy.owl\_ontology\_manager), 179  
 OWLObjectOneOf (class in owlapy.owl\_reasoner), 211  
 OWLObjectOneOf (class in owlapy.parser), 264  
 OWLObjectOneOf (class in owlapy.render), 302  
 OWLObjectOneOf (class in owlapy.utils), 323  
 OWLObjectParser (class in owlapy.owl\_object), 143  
 OWLObjectParser (class in owlapy.parser), 260  
 OWLObjectProperty (class in owlapy.converter), 76  
 OWLObjectProperty (class in owlapy.owl\_hierarchy), 122  
 OWLObjectProperty (class in owlapy.owl\_literal), 138  
 OWLObjectProperty (class in owlapy.owl\_ontology), 164  
 OWLObjectProperty (class in owlapy.owl\_ontology\_manager), 197  
 OWLObjectProperty (class in owlapy.owl\_property), 205  
 OWLObjectProperty (class in owlapy.owl\_reasoner), 227  
 OWLObjectProperty (class in owlapy.parser), 259  
 OWLObjectProperty (class in owlapy.utils), 311  
 OWLObjectPropertyAssertionAxiom (class in owlapy.owl\_axiom), 110  
 OWLObjectPropertyAssertionAxiom (class in owlapy.owl\_ontology\_manager), 185  
 OWLObjectPropertyAxiom (class in owlapy.owl\_axiom), 98  
 OWLObjectPropertyCharacteristicAxiom (class in owlapy.owl\_axiom), 111  
 OWLObjectPropertyCharacteristicAxiom (class in owlapy.owl\_ontology\_manager), 188  
 OWLObjectPropertyDomainAxiom (class in owlapy.owl\_axiom), 114  
 OWLObjectPropertyDomainAxiom (class in owlapy.owl\_ontology), 147  
 OWLObjectPropertyExpression (class in owlapy.class\_expression.restriction), 30  
 OWLObjectPropertyExpression (class in owlapy.owl\_axiom), 90  
 OWLObjectPropertyExpression (class in owlapy.owl\_ontology), 166  
 OWLObjectPropertyExpression (class in owlapy.owl\_property), 204  
 OWLObjectPropertyExpression (class in owlapy.owl\_reasoner), 226  
 OWLObjectPropertyExpression (class in owlapy.parser), 259  
 OWLObjectPropertyRangeAxiom (class in owlapy.owl\_axiom), 114  
 OWLObjectPropertyRangeAxiom (class in owlapy.owl\_ontology), 148  
 OWLObjectPropertyRangeAxiom (class in owlapy.owl\_ontology\_manager), 181  
 OWLObjectRenderer (class in owlapy.owl\_object), 142  
 OWLObjectRenderer (class in owlapy.render), 288  
 OWLObjectRestriction (class in owlapy.class\_expression), 51  
 OWLObjectRestriction (class in owlapy.class\_expression.restriction), 35  
 OWLObjectRestriction (class in owlapy.owl\_ontology), 163  
 OWLObjectSomeValuesFrom (class in owlapy.class\_expression), 54  
 OWLObjectSomeValuesFrom (class in owlapy.class\_expression.restriction), 37  
 OWLObjectSomeValuesFrom (class in owlapy.converter), 68  
 OWLObjectSomeValuesFrom (class in owlapy.owl\_ontology), 156  
 OWLObjectSomeValuesFrom (class in owlapy.owl\_reasoner), 209  
 OWLObjectSomeValuesFrom (class in owlapy.parser), 263  
 OWLObjectSomeValuesFrom (class in owlapy.render), 292  
 OWLObjectSomeValuesFrom (class in owlapy.utils), 315  
 OWLObjectUnionOf (class in owlapy.class\_expression), 49  
 OWLObjectUnionOf (class in owlapy.class\_expression.nary\_boolean\_expression), 21  
 OWLObjectUnionOf (class in owlapy.class\_expression.restriction), 28  
 OWLObjectUnionOf (class in owlapy.converter), 68  
 OWLObjectUnionOf (class in owlapy.owl\_axiom), 94  
 OWLObjectUnionOf (class in owlapy.owl\_ontology), 156  
 OWLObjectUnionOf (class in owlapy.owl\_reasoner), 210  
 OWLObjectUnionOf (class in owlapy.parser), 263  
 OWLObjectUnionOf (class in owlapy.render), 293  
 OWLObjectUnionOf (class in owlapy.utils), 321  
 OWLOntology (class in owlapy.owl\_ontology), 168  
 OWLOntology (class in owlapy.owl\_ontology\_manager), 192  
 OWLOntology (class in owlapy.owl\_reasoner), 221  
 OWLOntologyChange (class in owlapy.owl\_ontology\_manager), 198  
 OWLOntologyID (class in owlapy.owl\_ontology), 167  
 OWLOntologyManager (class in owlapy.owl\_ontology\_manager), 198

OWLProperty (class in owlapy.owl\_axiom), 92  
 OWLProperty (class in owlapy.owl\_ontology\_manager), 198  
 OWLProperty (class in owlapy.owl\_property), 205  
 OWLPropertyAssertionAxiom (class in owlapy.owl\_axiom), 109  
 OWLPropertyAxiom (class in owlapy.owl\_axiom), 98  
 OWLPropertyDomainAxiom (class in owlapy.owl\_axiom), 113  
 OWLPropertyDomainAxiom (class in owlapy.owl\_ontology\_manager), 186  
 OWLPropertyExpression (class in owlapy.class\_expression.restriction), 30  
 OWLPropertyExpression (class in owlapy.owl\_axiom), 91  
 OWLPropertyExpression (class in owlapy.owl\_ontology), 165  
 OWLPropertyExpression (class in owlapy.owl\_property), 204  
 OWLPropertyExpression (class in owlapy.owl\_reasoner), 228  
 OWLPropertyExpression (class in owlapy.render), 290  
 OWLPropertyRange (class in owlapy.class\_expression.class\_expression), 17  
 OWLPropertyRange (class in owlapy.class\_expression.restriction), 31  
 OWLPropertyRange (class in owlapy.owl\_data\_ranges), 116  
 OWLPropertyRange (class in owlapy.utils), 324  
 OWLPropertyRangeAxiom (class in owlapy.owl\_axiom), 113  
 OWLPropertyRangeAxiom (class in owlapy.owl\_ontology\_manager), 184  
 OWLQuantifiedDataRestriction (class in owlapy.class\_expression), 53  
 OWLQuantifiedDataRestriction (class in owlapy.class\_expression.restriction), 40  
 OWLQuantifiedDataRestriction (class in owlapy.owl\_ontology\_manager), 178  
 OWLQuantifiedDataRestriction (class in owlapy.parser), 267  
 OWLQuantifiedObjectRestriction (class in owlapy.class\_expression), 50  
 OWLQuantifiedObjectRestriction (class in owlapy.class\_expression.restriction), 36  
 OWLQuantifiedObjectRestriction (class in owlapy.owl\_ontology\_manager), 181  
 OWLQuantifiedObjectRestriction (class in owlapy.parser), 267  
 OWLQuantifiedRestriction (class in owlapy.class\_expression), 50  
 OWLQuantifiedRestriction (class in owlapy.class\_expression.restriction), 35  
 OWLRDFVocabulary (class in owlapy.class\_expression), 61  
 OWLRDFVocabulary (class in owlapy.converter), 78  
 OWLRDFVocabulary (class in owlapy.owl\_literal), 137  
 OWLRDFVocabulary (class in owlapy.parser), 262  
 OWLRDFVocabulary (class in owlapy.vocab), 332  
 OWLREADY2\_FACET\_KEYS (in module owlapy.owl\_ontology), 172  
 OWLReasoner (class in owlapy.owl\_hierarchy), 123  
 OWLReasoner (class in owlapy.owl\_reasoner), 232  
 OWLReasonerEx (class in owlapy.owl\_reasoner), 238  
 OWLReflexiveObjectPropertyAxiom (class in owlapy.owl\_axiom), 112  
 OWLReflexiveObjectPropertyAxiom (class in owlapy.owl\_ontology\_manager), 187  
 OWLRestriction (class in owlapy.class\_expression), 50  
 OWLRestriction (class in owlapy.class\_expression.restriction), 34  
 OWLRestriction (class in owlapy.owl\_ontology), 163  
 OWLRestriction (class in owlapy.render), 298  
 OWLRestriction (class in owlapy.utils), 314  
 OWLSameIndividualAxiom (class in owlapy.owl\_axiom), 102  
 OWLSameIndividualAxiom (class in owlapy.owl\_ontology\_manager), 191  
 OWLSubAnnotationPropertyOfAxiom (class in owlapy.owl\_axiom), 107  
 OWLSubClassOfAxiom (class in owlapy.owl\_axiom), 104  
 OWLSubClassOfAxiom (class in owlapy.owl\_ontology), 148  
 OWLSubClassOfAxiom (class in owlapy.owl\_ontology\_manager), 181  
 OWLSubClassOfAxiom (class in owlapy.owl\_reasoner), 219  
 OWLSubDataPropertyOfAxiom (class in owlapy.owl\_axiom), 109  
 OWLSubObjectPropertyOfAxiom (class in owlapy.owl\_axiom), 109  
 OWLSubPropertyAxiom (class in owlapy.owl\_axiom), 108  
 OWLSubPropertyAxiom (class in owlapy.owl\_ontology\_manager), 184  
 OWLSymmetricObjectPropertyAxiom (class in owlapy.owl\_axiom), 112  
 OWLSymmetricObjectPropertyAxiom (class in owlapy.owl\_ontology\_manager), 185  
 OWLThing (in module owlapy.class\_expression), 61  
 OWLThing (in module owlapy.owl\_axiom), 94  
 OWLThing (in module owlapy.owl\_hierarchy), 122  
 OWLThing (in module owlapy.owl\_ontology), 155  
 OWLThing (in module owlapy.owl\_ontology\_manager), 177  
 OWLThing (in module owlapy.utils), 315  
 OWLTopDataProperty (in module owlapy.owl\_hierarchy), 122  
 OWLTopDataProperty (in module owlapy.owl\_literal), 141  
 OWLTopObjectProperty (in module owlapy.owl\_hierarchy), 122  
 OWLTopObjectProperty (in module owlapy.owl\_literal), 141



OWLTransitiveObjectPropertyAxiom (class in owlapy.owl\_axiom), 112  
OWLTransitiveObjectPropertyAxiom (class in owlapy.owl\_ontology\_manager), 186  
OWLUnaryPropertyAxiom (class in owlapy.owl\_axiom), 111

## P

parent (owlapy.converter.Owl2SparqlConverter attribute), 79  
parent\_var (owlapy.converter.Owl2SparqlConverter attribute), 79  
parents () (owlapy.owl\_hierarchy.AbstractHierarchy method), 130  
parse\_boolean () (owlapy.class\_expression.restriction.OwLLiteral method), 31  
parse\_boolean () (owlapy.converter.OwLLiteral method), 74  
parse\_boolean () (owlapy.owl\_axiom.OwLLiteral method), 96  
parse\_boolean () (owlapy.owl\_literal.OwLLiteral method), 139  
parse\_boolean () (owlapy.owl\_ontology.OwLLiteral method), 151  
parse\_boolean () (owlapy.owl\_reasoner.OwLLiteral method), 230  
parse\_boolean () (owlapy.parser.OwLLiteral method), 257  
parse\_boolean () (owlapy.providers.OwLLiteral method), 280  
parse\_boolean () (owlapy.render.OwLLiteral method), 287  
parse\_boolean () (owlapy.utils.OwLLiteral method), 309  
parse\_date () (owlapy.class\_expression.restriction.OwLLiteral method), 32  
parse\_date () (owlapy.converter.OwLLiteral method), 75  
parse\_date () (owlapy.owl\_axiom.OwLLiteral method), 97  
parse\_date () (owlapy.owl\_literal.OwLLiteral method), 140  
parse\_date () (owlapy.owl\_ontology.OwLLiteral method), 152  
parse\_date () (owlapy.owl\_reasoner.OwLLiteral method), 230  
parse\_date () (owlapy.parser.OwLLiteral method), 258  
parse\_date () (owlapy.providers.OwLLiteral method), 280  
parse\_date () (owlapy.render.OwLLiteral method), 288  
parse\_date () (owlapy.utils.OwLLiteral method), 310  
parse\_datetime () (owlapy.class\_expression.restriction.OwLLiteral method), 32  
parse\_datetime () (owlapy.converter.OwLLiteral method), 75  
parse\_datetime () (owlapy.owl\_axiom.OwLLiteral method), 97  
parse\_datetime () (owlapy.owl\_literal.OwLLiteral method), 140  
parse\_datetime () (owlapy.owl\_ontology.OwLLiteral method), 152  
parse\_datetime () (owlapy.owl\_reasoner.OwLLiteral method), 231  
parse\_datetime () (owlapy.parser.OwLLiteral method), 258  
parse\_datetime () (owlapy.providers.OwLLiteral method), 280  
parse\_datetime () (owlapy.render.OwLLiteral method), 288  
parse\_datetime () (owlapy.utils.OwLLiteral method), 310  
parse\_double () (owlapy.class\_expression.restriction.OwLLiteral method), 32  
parse\_double () (owlapy.converter.OwLLiteral method), 75  
parse\_double () (owlapy.owl\_axiom.OwLLiteral method), 96  
parse\_double () (owlapy.owl\_literal.OwLLiteral method), 140  
parse\_double () (owlapy.owl\_ontology.OwLLiteral method), 151  
parse\_double () (owlapy.owl\_reasoner.OwLLiteral method), 230  
parse\_double () (owlapy.parser.OwLLiteral method), 258  
parse\_double () (owlapy.providers.OwLLiteral method), 280  
parse\_double () (owlapy.render.OwLLiteral method), 287  
parse\_double () (owlapy.utils.OwLLiteral method), 309  
parse\_duration () (owlapy.class\_expression.restriction.OwLLiteral method), 32  
parse\_duration () (owlapy.converter.OwLLiteral method), 75  
parse\_duration () (owlapy.owl\_axiom.OwLLiteral method), 97  
parse\_duration () (owlapy.owl\_literal.OwLLiteral method), 140  
parse\_duration () (owlapy.owl\_ontology.OwLLiteral method), 152  
parse\_duration () (owlapy.owl\_reasoner.OwLLiteral method), 231  
parse\_duration () (owlapy.parser.OwLLiteral method), 258  
parse\_duration () (owlapy.providers.OwLLiteral method), 281  
parse\_duration () (owlapy.render.OwLLiteral method), 288  
parse\_duration () (owlapy.utils.OwLLiteral method), 310  
parse\_expression () (owlapy.owl\_object.OwLObjectParser method), 143  
parse\_expression () (owlapy.parser.DLSyntaxParser method), 276  
parse\_expression () (owlapy.parser.ManchesterOWLSyntaxParser method), 274  
parse\_expression () (owlapy.parser.OwLObjectParser method), 260  
parse\_integer () (owlapy.class\_expression.restriction.OwLLiteral method), 32  
parse\_integer () (owlapy.converter.OwLLiteral method), 75  
parse\_integer () (owlapy.owl\_axiom.OwLLiteral method), 96  
parse\_integer () (owlapy.owl\_literal.OwLLiteral method), 140  
parse\_integer () (owlapy.owl\_ontology.OwLLiteral method), 151

`parse_integer()` (*owlapy.owl\_reasoner.OWLLiteral method*), 230  
`parse_integer()` (*owlapy.parser.OWLLiteral method*), 258  
`parse_integer()` (*owlapy.providers.OWLLiteral method*), 280  
`parse_integer()` (*owlapy.render.OWLLiteral method*), 287  
`parse_integer()` (*owlapy.utils.OWLLiteral method*), 309  
`parse_string()` (*owlapy.class\_expression.restriction.OWLLiteral method*), 32  
`parse_string()` (*owlapy.converter.OWLLiteral method*), 75  
`parse_string()` (*owlapy.owl\_axiom.OWLLiteral method*), 97  
`parse_string()` (*owlapy.owl\_literal.OWLLiteral method*), 140  
`parse_string()` (*owlapy.owl\_ontology.OWLLiteral method*), 152  
`parse_string()` (*owlapy.owl\_reasoner.OWLLiteral method*), 230  
`parse_string()` (*owlapy.parser.OWLLiteral method*), 258  
`parse_string()` (*owlapy.providers.OWLLiteral method*), 280  
`parse_string()` (*owlapy.render.OWLLiteral method*), 287  
`parse_string()` (*owlapy.utils.OWLLiteral method*), 309  
`PATTERN` (*owlapy.class\_expression.OWLFacet attribute*), 56  
`PATTERN` (*owlapy.class\_expression.restriction.OWLFacet attribute*), 34  
`PATTERN` (*owlapy.converter.OWLFacet attribute*), 78  
`PATTERN` (*owlapy.owl\_ontology.OWLFacet attribute*), 167  
`PATTERN` (*owlapy.parser.OWLFacet attribute*), 262  
`PATTERN` (*owlapy.providers.OWLFacet attribute*), 282  
`PATTERN` (*owlapy.render.OWLFacet attribute*), 299  
`PATTERN` (*owlapy.vocab.OWLFacet attribute*), 333  
`peek()` (*in module owlapy.converter*), 78  
`PELLET` (*owlapy.owl\_reasoner.BaseReasoner attribute*), 238  
`prefix` (*owlapy.iri.Namespaces property*), 81  
`prefix` (*owlapy.namespaces.Namespaces property*), 85  
`prefix` (*owlapy.parser.Namespaces property*), 260  
`prefix` (*owlapy.vocab.Namespaces property*), 332  
`PREV` (*owlapy.owl\_reasoner.LRUCache attribute*), 231  
`PREV` (*owlapy.utils.LRUCache attribute*), 329  
`process()` (*owlapy.converter.Owl2SparqlConverter method*), 80  
`properties` (*owlapy.converter.Owl2SparqlConverter attribute*), 79  
`properties()` (*owlapy.owl\_axiom.OwLNaryPropertyAxiom method*), 103  
`properties()` (*owlapy.owl\_ontology\_manager.OwLNaryPropertyAxiom method*), 190

## R

`RDF` (*in module owlapy.namespaces*), 85  
`RDFS` (*in module owlapy.namespaces*), 85  
`RDFS_LITERAL` (*owlapy.class\_expression.OwLRDFVocabulary attribute*), 61  
`RDFS_LITERAL` (*owlapy.converter.OwLRDFVocabulary attribute*), 78  
`RDFS_LITERAL` (*owlapy.owl\_literal.OwLRDFVocabulary attribute*), 138  
`RDFS_LITERAL` (*owlapy.parser.OwLRDFVocabulary attribute*), 262  
`RDFS_LITERAL` (*owlapy.vocab.OwLRDFVocabulary attribute*), 333  
`reminder` (*owlapy.class\_expression.owl\_class.IRI property*), 24  
`reminder` (*owlapy.class\_expression.owl\_class.OwLClass property*), 25  
`reminder` (*owlapy.class\_expression.OwLClass property*), 49  
`reminder` (*owlapy.converter.OwLClass property*), 66  
`reminder` (*owlapy.iri.IRI property*), 82  
`reminder` (*owlapy.owl\_axiom.IRI property*), 95  
`reminder` (*owlapy.owl\_axiom.OwLClass property*), 93  
`reminder` (*owlapy.owl\_datatype.IRI property*), 119  
`reminder` (*owlapy.owl\_hierarchy.OwLClass property*), 121  
`reminder` (*owlapy.owl\_individual.IRI property*), 134  
`reminder` (*owlapy.owl\_ontology\_manager.IRI property*), 176  
`reminder` (*owlapy.owl\_ontology\_manager.OwLClass property*), 177  
`reminder` (*owlapy.owl\_ontology.IRI property*), 154  
`reminder` (*owlapy.owl\_ontology.OwLClass property*), 154  
`reminder` (*owlapy.owl\_property.IRI property*), 203  
`reminder` (*owlapy.owl\_reasoner.IRI property*), 218  
`reminder` (*owlapy.owl\_reasoner.OwLClass property*), 217  
`reminder` (*owlapy.parser.IRI property*), 256  
`reminder` (*owlapy.parser.OwLClass property*), 264  
`reminder` (*owlapy.render.IRI property*), 286  
`reminder` (*owlapy.render.OwLClass property*), 292  
`reminder` (*owlapy.utils.OwLClass property*), 313  
`reminder` (*owlapy.vocab.IRI property*), 332

remove\_axiom() (owlapy.owl\_ontology\_manager.OntologyManager method), 201  
 remove\_axiom() (owlapy.owl\_ontology\_manager.OwLOntologyManager method), 199  
 remove\_axiom() (owlapy.owl\_reasoner.OntologyManager method), 226  
 render() (owlapy.converter.Owl2SparqlConverter method), 79  
 render() (owlapy.owl\_object.OwLObjectRenderer method), 143  
 render() (owlapy.render.DLSyntaxObjectRenderer method), 303  
 render() (owlapy.render.ManchesterOWLSyntaxOwLObjectRenderer method), 304  
 render() (owlapy.render.OwLObjectRenderer method), 289  
 reset() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 246  
 restrict() (owlapy.owl\_hierarchy.AbstractHierarchy static method), 130  
 restrict\_and\_copy() (owlapy.owl\_hierarchy.AbstractHierarchy method), 130  
 Restriction\_Literals (in module owlapy.providers), 282  
 RESULT (owlapy.owl\_reasoner.LRUCache attribute), 231  
 RESULT (owlapy.utils.LRUCache attribute), 330  
 roots() (owlapy.owl\_hierarchy.AbstractHierarchy method), 131

## S

same\_individuals() (owlapy.owl\_hierarchy.OwLReasoner method), 125  
 same\_individuals() (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 248  
 same\_individuals() (owlapy.owl\_reasoner.OntologyReasoner method), 241  
 same\_individuals() (owlapy.owl\_reasoner.OwLReasoner method), 234  
 save\_ontology() (owlapy.owl\_ontology\_manager.OntologyManager method), 201  
 save\_ontology() (owlapy.owl\_ontology\_manager.OwLOntologyManager method), 199  
 save\_ontology() (owlapy.owl\_reasoner.OntologyManager method), 226  
 save\_world() (owlapy.owl\_ontology\_manager.OntologyManager method), 201  
 save\_world() (owlapy.owl\_reasoner.OntologyManager method), 226  
 sentinel (owlapy.owl\_reasoner.LRUCache attribute), 231  
 sentinel (owlapy.utils.LRUCache attribute), 329, 330  
 set\_short\_form\_provider() (owlapy.owl\_object.OwLObjectRenderer method), 142  
 set\_short\_form\_provider() (owlapy.render.DLSyntaxObjectRenderer method), 303  
 set\_short\_form\_provider() (owlapy.render.ManchesterOWLSyntaxOwLObjectRenderer method), 303  
 set\_short\_form\_provider() (owlapy.render.OwLObjectRenderer method), 288  
 siblings() (owlapy.owl\_hierarchy.AbstractHierarchy method), 130  
 simplify() (owlapy.utils.OperandSetTransform method), 328  
 slots (owlapy.parser.DLSyntaxParser attribute), 276  
 slots (owlapy.parser.ManchesterOWLSyntaxParser attribute), 274  
 sort() (owlapy.utils.ConceptOperandSorter method), 328  
 sparql (owlapy.converter.Owl2SparqlConverter attribute), 79  
 stack\_parent() (owlapy.converter.Owl2SparqlConverter method), 79  
 stack\_variable() (owlapy.converter.Owl2SparqlConverter method), 79  
 str (owlapy.class\_expression.owl\_class.IRI property), 24  
 str (owlapy.class\_expression.owl\_class.OwLClass property), 25  
 str (owlapy.class\_expression.OwLClass property), 49  
 str (owlapy.class\_expression.restriction.OwLDatatype property), 33  
 str (owlapy.converter.OwLClass property), 66  
 str (owlapy.converter.OwLDatatype property), 77  
 str (owlapy.converter.OwLNamedIndividual property), 74  
 str (owlapy.iri.IRI property), 82  
 str (owlapy.meta\_classes.HasIRI property), 83  
 str (owlapy.owl\_axiom.IRI property), 95  
 str (owlapy.owl\_axiom.OwLAnnotationProperty property), 106  
 str (owlapy.owl\_axiom.OwLClass property), 93  
 str (owlapy.owl\_axiom.OwLDatatype property), 91  
 str (owlapy.owl\_axiom.OwLProperty property), 92  
 str (owlapy.owl\_datatype.HasIRI property), 119  
 str (owlapy.owl\_datatype.IRI property), 119  
 str (owlapy.owl\_datatype.OwLDatatype property), 120  
 str (owlapy.owl\_hierarchy.HasIRI property), 122  
 str (owlapy.owl\_hierarchy.OwLClass property), 121  
 str (owlapy.owl\_individual.IRI property), 134  
 str (owlapy.owl\_individual.OwLNamedIndividual property), 135  
 str (owlapy.owl\_literal.OwLDatatype property), 137  
 str (owlapy.owl\_object.HasIRI property), 142  
 str (owlapy.owl\_ontology\_manager.HasIRI property), 176  
 str (owlapy.owl\_ontology\_manager.IRI property), 176  
 str (owlapy.owl\_ontology\_manager.OwLAnnotationProperty property), 184  
 str (owlapy.owl\_ontology\_manager.OwLClass property), 177



str (owlapy.owl\_ontology\_manager.OWLImportsDeclaration property), 200  
 str (owlapy.owl\_ontology\_manager.OWLNamedIndividual property), 192  
 str (owlapy.owl\_ontology\_manager.OWLProperty property), 198  
 str (owlapy.owl\_ontology.IRI property), 154  
 str (owlapy.owl\_ontology.OWLAnnotationProperty property), 148  
 str (owlapy.owl\_ontology.OWLClass property), 154  
 str (owlapy.owl\_ontology.OWLDatatype property), 150  
 str (owlapy.owl\_ontology.OWLNamedIndividual property), 150  
 str (owlapy.owl\_property.IRI property), 203  
 str (owlapy.owl\_property.OWLProperty property), 205  
 str (owlapy.owl\_reasoner.IRI property), 218  
 str (owlapy.owl\_reasoner.OWLClass property), 217  
 str (owlapy.owl\_reasoner.OWLDatatype property), 221  
 str (owlapy.owl\_reasoner.OWLNamedIndividual property), 229  
 str (owlapy.parser.IRI property), 256  
 str (owlapy.parser.OWLClass property), 264  
 str (owlapy.parser.OWLDatatype property), 261  
 str (owlapy.parser.OWLNamedIndividual property), 257  
 str (owlapy.render.IRI property), 286  
 str (owlapy.render.OWLClass property), 291  
 str (owlapy.render.OWLDatatype property), 303  
 str (owlapy.render.OWLNamedIndividual property), 286  
 str (owlapy.utils.HasIRI property), 308  
 str (owlapy.utils.OWLClass property), 313  
 str (owlapy.utils.OWLDatatype property), 325  
 str (owlapy.utils.OWLNamedIndividual property), 307  
 str (owlapy.vocab.HasIRI property), 330  
 str (owlapy.vocab.IRI property), 331  
 STRING (owlapy.owl\_literal.XSDVocabulary attribute), 138  
 STRING (owlapy.vocab.XSDVocabulary attribute), 333  
 StringOWLDatatype (in module owlapy.owl\_literal), 141  
 StringOWLDatatype (in module owlapy.owl\_ontology), 151  
 StringOWLDatatype (in module owlapy.parser), 257  
 sub\_classes () (owlapy.owl\_hierarchy.ClassHierarchy method), 131  
 sub\_classes () (owlapy.owl\_hierarchy.OWLReasoner method), 127  
 sub\_classes () (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 249  
 sub\_classes () (owlapy.owl\_reasoner.OntologyReasoner method), 242  
 sub\_classes () (owlapy.owl\_reasoner.OWLReasoner method), 235  
 sub\_data\_properties () (owlapy.owl\_hierarchy.DatatypePropertyHierarchy method), 132  
 sub\_data\_properties () (owlapy.owl\_hierarchy.OWLReasoner method), 127  
 sub\_data\_properties () (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 251  
 sub\_data\_properties () (owlapy.owl\_reasoner.OntologyReasoner method), 244  
 sub\_data\_properties () (owlapy.owl\_reasoner.OWLReasoner method), 236  
 sub\_object\_properties () (owlapy.owl\_hierarchy.ObjectPropertyHierarchy method), 131  
 sub\_object\_properties () (owlapy.owl\_hierarchy.OWLReasoner method), 128  
 sub\_object\_properties () (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 252  
 sub\_object\_properties () (owlapy.owl\_reasoner.OntologyReasoner method), 245  
 sub\_object\_properties () (owlapy.owl\_reasoner.OWLReasoner method), 236  
 super\_classes () (owlapy.owl\_hierarchy.ClassHierarchy method), 131  
 super\_classes () (owlapy.owl\_hierarchy.OWLReasoner method), 129  
 super\_classes () (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 249  
 super\_classes () (owlapy.owl\_reasoner.OntologyReasoner method), 243  
 super\_classes () (owlapy.owl\_reasoner.OWLReasoner method), 237  
 super\_data\_properties () (owlapy.owl\_hierarchy.DatatypePropertyHierarchy method), 132  
 super\_data\_properties () (owlapy.owl\_hierarchy.OWLReasoner method), 128  
 super\_data\_properties () (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 251  
 super\_data\_properties () (owlapy.owl\_reasoner.OntologyReasoner method), 244  
 super\_data\_properties () (owlapy.owl\_reasoner.OWLReasoner method), 236  
 super\_object\_properties () (owlapy.owl\_hierarchy.ObjectPropertyHierarchy method), 131  
 super\_object\_properties () (owlapy.owl\_hierarchy.OWLReasoner method), 128  
 super\_object\_properties () (owlapy.owl\_reasoner.FastInstanceCheckerReasoner method), 251  
 super\_object\_properties () (owlapy.owl\_reasoner.OntologyReasoner method), 245  
 super\_object\_properties () (owlapy.owl\_reasoner.OWLReasoner method), 237  
 symbolic\_form (owlapy.class\_expression.OWLFacet property), 56  
 symbolic\_form (owlapy.class\_expression.restriction.OWLFacet property), 34  
 symbolic\_form (owlapy.converter.OWLFacet property), 77  
 symbolic\_form (owlapy.owl\_ontology.OWLFacet property), 167  
 symbolic\_form (owlapy.parser.OWLFacet property), 261

`symbolic_form` (*owlapy.providers.OWLFacet* property), 281  
`symbolic_form` (*owlapy.render.OWLFacet* property), 299  
`symbolic_form` (*owlapy.vocab.OWLFacet* property), 333  
`SyncReasoner` (class in *owlapy.owl\_reasoner*), 252

## T

`TIME_DATATYPES` (in module *owlapy.owl\_literal*), 141  
`to_python()` (*owlapy.class\_expression.restriction.OWLLiteral* method), 33  
`to_python()` (*owlapy.converter.OWLLiteral* method), 76  
`to_python()` (*owlapy.owl\_axiom.OWLLiteral* method), 97  
`to_python()` (*owlapy.owl\_literal.OWLLiteral* method), 141  
`to_python()` (*owlapy.owl\_ontology.OWLLiteral* method), 152  
`to_python()` (*owlapy.owl\_reasoner.OWLLiteral* method), 231  
`to_python()` (*owlapy.parser.OWLLiteral* method), 259  
`to_python()` (*owlapy.providers.OWLLiteral* method), 281  
`to_python()` (*owlapy.render.OWLLiteral* method), 288  
`to_python()` (*owlapy.utils.OWLLiteral* method), 310  
`to_string_id()` (*owlapy.class\_expression.owl\_class.OWLEntity* method), 23  
`to_string_id()` (*owlapy.converter.OWLEntity* method), 77  
`to_string_id()` (*owlapy.owl\_axiom.OWLEntity* method), 90  
`to_string_id()` (*owlapy.owl\_datatype.OWLEntity* method), 118  
`to_string_id()` (*owlapy.owl\_individual.OWLEntity* method), 133  
`to_string_id()` (*owlapy.owl\_object.OWLEntity* method), 143  
`to_string_id()` (*owlapy.owl\_property.OWLEntity* method), 202  
`to_string_id()` (*owlapy.render.OWLEntity* method), 289  
`ToOwlready2` (class in *owlapy.owl\_ontology*), 172  
`ToOwlready2` (class in *owlapy.owl\_ontology\_manager*), 196  
`ToOwlready2` (class in *owlapy.owl\_reasoner*), 225  
`TopLevelCNF` (class in *owlapy.utils*), 329  
`TopLevelDNF` (class in *owlapy.utils*), 329  
`TopOWLDatatype` (in module *owlapy.converter*), 76  
`TopOWLDatatype` (in module *owlapy.owl\_literal*), 141  
`TOTAL_DIGITS` (*owlapy.class\_expression.OWLFacet* attribute), 56  
`TOTAL_DIGITS` (*owlapy.class\_expression.restriction.OWLFacet* attribute), 34  
`TOTAL_DIGITS` (*owlapy.converter.OWLFacet* attribute), 78  
`TOTAL_DIGITS` (*owlapy.owl\_ontology.OWLFacet* attribute), 167  
`TOTAL_DIGITS` (*owlapy.parser.OWLFacet* attribute), 262  
`TOTAL_DIGITS` (*owlapy.providers.OWLFacet* attribute), 282  
`TOTAL_DIGITS` (*owlapy.render.OWLFacet* attribute), 299  
`TOTAL_DIGITS` (*owlapy.vocab.OWLFacet* attribute), 333  
`triple()` (*owlapy.converter.Owl2SparqlConverter* method), 80  
`type_index` (*owlapy.class\_expression.class\_expression.OWLObjectComplementOf* attribute), 19  
`type_index` (*owlapy.class\_expression.nary\_boolean\_expression.OWLObjectIntersectionOf* attribute), 22  
`type_index` (*owlapy.class\_expression.nary\_boolean\_expression.OWLObjectUnionOf* attribute), 21  
`type_index` (*owlapy.class\_expression.owl\_class.IRI* attribute), 24  
`type_index` (*owlapy.class\_expression.owl\_class.OWLClass* attribute), 25  
`type_index` (*owlapy.class\_expression.owl\_class.OWLObjectComplementOf* attribute), 23  
`type_index` (*owlapy.class\_expression.OWLClass* attribute), 48  
`type_index` (*owlapy.class\_expression.OWLDataAllValuesFrom* attribute), 59  
`type_index` (*owlapy.class\_expression.OWLDataExactCardinality* attribute), 60  
`type_index` (*owlapy.class\_expression.OWLDataHasValue* attribute), 59  
`type_index` (*owlapy.class\_expression.OWLDataMaxCardinality* attribute), 60  
`type_index` (*owlapy.class\_expression.OWLDataMinCardinality* attribute), 60  
`type_index` (*owlapy.class\_expression.OWLDataOneOf* attribute), 53  
`type_index` (*owlapy.class\_expression.OWLDataSomeValuesFrom* attribute), 58  
`type_index` (*owlapy.class\_expression.OWLDatatypeRestriction* attribute), 56  
`type_index` (*owlapy.class\_expression.OWLFacetRestriction* attribute), 57  
`type_index` (*owlapy.class\_expression.OWLObjectAllValuesFrom* attribute), 55  
`type_index` (*owlapy.class\_expression.OWLObjectComplementOf* attribute), 48  
`type_index` (*owlapy.class\_expression.OWLObjectExactCardinality* attribute), 58  
`type_index` (*owlapy.class\_expression.OWLObjectHasSelf* attribute), 53  
`type_index` (*owlapy.class\_expression.OWLObjectHasValue* attribute), 55  
`type_index` (*owlapy.class\_expression.OWLObjectIntersectionOf* attribute), 50  
`type_index` (*owlapy.class\_expression.OWLObjectMaxCardinality* attribute), 57  
`type_index` (*owlapy.class\_expression.OWLObjectMinCardinality* attribute), 57  
`type_index` (*owlapy.class\_expression.OWLObjectOneOf* attribute), 60  
`type_index` (*owlapy.class\_expression.OWLObjectSomeValuesFrom* attribute), 54

type\_index (owlapy.class\_expression.OwLObjectUnionOf attribute), 50  
 type\_index (owlapy.class\_expression.restriction.OwLDataAllValuesFrom attribute), 42  
 type\_index (owlapy.class\_expression.restriction.OwLDataExactCardinality attribute), 42  
 type\_index (owlapy.class\_expression.restriction.OwLDataHasValue attribute), 43  
 type\_index (owlapy.class\_expression.restriction.OwLDataMaxCardinality attribute), 41  
 type\_index (owlapy.class\_expression.restriction.OwLDataMinCardinality attribute), 41  
 type\_index (owlapy.class\_expression.restriction.OwLDataOneOf attribute), 43  
 type\_index (owlapy.class\_expression.restriction.OwLDataSomeValuesFrom attribute), 42  
 type\_index (owlapy.class\_expression.restriction.OwLDatatype attribute), 33  
 type\_index (owlapy.class\_expression.restriction.OwLDatatypeRestriction attribute), 44  
 type\_index (owlapy.class\_expression.restriction.OwLFacetRestriction attribute), 44  
 type\_index (owlapy.class\_expression.restriction.OwLLiteral attribute), 31  
 type\_index (owlapy.class\_expression.restriction.OwLObjectAllValuesFrom attribute), 38  
 type\_index (owlapy.class\_expression.restriction.OwLObjectExactCardinality attribute), 37  
 type\_index (owlapy.class\_expression.restriction.OwLObjectHasSelf attribute), 39  
 type\_index (owlapy.class\_expression.restriction.OwLObjectHasValue attribute), 39  
 type\_index (owlapy.class\_expression.restriction.OwLObjectIntersectionOf attribute), 28  
 type\_index (owlapy.class\_expression.restriction.OwLObjectMaxCardinality attribute), 37  
 type\_index (owlapy.class\_expression.restriction.OwLObjectMinCardinality attribute), 37  
 type\_index (owlapy.class\_expression.restriction.OwLObjectOneOf attribute), 39  
 type\_index (owlapy.class\_expression.restriction.OwLObjectSomeValuesFrom attribute), 38  
 type\_index (owlapy.class\_expression.restriction.OwLObjectUnionOf attribute), 29  
 type\_index (owlapy.converter.OwLClass attribute), 66  
 type\_index (owlapy.converter.OwLDataAllValuesFrom attribute), 72  
 type\_index (owlapy.converter.OwLDataExactCardinality attribute), 66  
 type\_index (owlapy.converter.OwLDataHasValue attribute), 73  
 type\_index (owlapy.converter.OwLDataMaxCardinality attribute), 66  
 type\_index (owlapy.converter.OwLDataMinCardinality attribute), 65  
 type\_index (owlapy.converter.OwLDataOneOf attribute), 73  
 type\_index (owlapy.converter.OwLDataProperty attribute), 76  
 type\_index (owlapy.converter.OwLDataSomeValuesFrom attribute), 72  
 type\_index (owlapy.converter.OwLDatatype attribute), 77  
 type\_index (owlapy.converter.OwLDatatypeRestriction attribute), 65  
 type\_index (owlapy.converter.OwLLiteral attribute), 74  
 type\_index (owlapy.converter.OwLNamedIndividual attribute), 74  
 type\_index (owlapy.converter.OwLObjectAllValuesFrom attribute), 69  
 type\_index (owlapy.converter.OwLObjectComplementOf attribute), 68  
 type\_index (owlapy.converter.OwLObjectExactCardinality attribute), 70  
 type\_index (owlapy.converter.OwLObjectHasSelf attribute), 71  
 type\_index (owlapy.converter.OwLObjectHasValue attribute), 64  
 type\_index (owlapy.converter.OwLObjectIntersectionOf attribute), 68  
 type\_index (owlapy.converter.OwLObjectMaxCardinality attribute), 70  
 type\_index (owlapy.converter.OwLObjectMinCardinality attribute), 70  
 type\_index (owlapy.converter.OwLObjectOneOf attribute), 64  
 type\_index (owlapy.converter.OwLObjectProperty attribute), 76  
 type\_index (owlapy.converter.OwLObjectSomeValuesFrom attribute), 69  
 type\_index (owlapy.converter.OwLObjectUnionOf attribute), 68  
 type\_index (owlapy.iri.IRI attribute), 81  
 type\_index (owlapy.owl\_axiom.IRI attribute), 94  
 type\_index (owlapy.owl\_axiom.OwLClass attribute), 93  
 type\_index (owlapy.owl\_axiom.OwLDatatype attribute), 91  
 type\_index (owlapy.owl\_axiom.OwLLiteral attribute), 96  
 type\_index (owlapy.owl\_axiom.OwLObjectUnionOf attribute), 94  
 type\_index (owlapy.owl\_data\_ranges.OwLDataComplementOf attribute), 117  
 type\_index (owlapy.owl\_data\_ranges.OwLDataIntersectionOf attribute), 117  
 type\_index (owlapy.owl\_data\_ranges.OwLDataUnionOf attribute), 117  
 type\_index (owlapy.owl\_datatype.IRI attribute), 118  
 type\_index (owlapy.owl\_datatype.OwLDatatype attribute), 120  
 type\_index (owlapy.owl\_hierarchy.OwLClass attribute), 121  
 type\_index (owlapy.owl\_hierarchy.OwLDataProperty attribute), 123  
 type\_index (owlapy.owl\_hierarchy.OwLObjectProperty attribute), 122  
 type\_index (owlapy.owl\_individual.IRI attribute), 133  
 type\_index (owlapy.owl\_individual.OwLNamedIndividual attribute), 135  
 type\_index (owlapy.owl\_literal.OwLDataProperty attribute), 139  
 type\_index (owlapy.owl\_literal.OwLDatatype attribute), 137  
 type\_index (owlapy.owl\_literal.OwLLiteral attribute), 139  
 type\_index (owlapy.owl\_literal.OwLObjectProperty attribute), 138  
 type\_index (owlapy.owl\_ontology\_manager.IRI attribute), 175

type\_index (owlapy.owl\_ontology\_manager.OWLClass attribute), 177  
 type\_index (owlapy.owl\_ontology\_manager.OWLDataHasValue attribute), 178  
 type\_index (owlapy.owl\_ontology\_manager.OWLDataProperty attribute), 196  
 type\_index (owlapy.owl\_ontology\_manager.OWLNamedIndividual attribute), 192  
 type\_index (owlapy.owl\_ontology\_manager.OWLObjectComplementOf attribute), 180  
 type\_index (owlapy.owl\_ontology\_manager.OWLObjectHasValue attribute), 180  
 type\_index (owlapy.owl\_ontology\_manager.OWLObjectInverseOf attribute), 197  
 type\_index (owlapy.owl\_ontology\_manager.OWLObjectOneOf attribute), 179  
 type\_index (owlapy.owl\_ontology\_manager.OWLObjectProperty attribute), 197  
 type\_index (owlapy.owl\_ontology\_manager.OWLOntology attribute), 192  
 type\_index (owlapy.owl\_ontology.IRI attribute), 153  
 type\_index (owlapy.owl\_ontology.OWLClass attribute), 154  
 type\_index (owlapy.owl\_ontology.OWLDataAllValuesFrom attribute), 160  
 type\_index (owlapy.owl\_ontology.OWLDataComplementOf attribute), 149  
 type\_index (owlapy.owl\_ontology.OWLDataExactCardinality attribute), 160  
 type\_index (owlapy.owl\_ontology.OWLDataHasValue attribute), 161  
 type\_index (owlapy.owl\_ontology.OWLDataIntersectionOf attribute), 149  
 type\_index (owlapy.owl\_ontology.OWLDataMaxCardinality attribute), 161  
 type\_index (owlapy.owl\_ontology.OWLDataMinCardinality attribute), 161  
 type\_index (owlapy.owl\_ontology.OWLDataOneOf attribute), 162  
 type\_index (owlapy.owl\_ontology.OWLDataProperty attribute), 164  
 type\_index (owlapy.owl\_ontology.OWLDataSomeValuesFrom attribute), 159  
 type\_index (owlapy.owl\_ontology.OWLDatatype attribute), 150  
 type\_index (owlapy.owl\_ontology.OWLDatatypeRestriction attribute), 162  
 type\_index (owlapy.owl\_ontology.OWLDataUnionOf attribute), 149  
 type\_index (owlapy.owl\_ontology.OWLFacetRestriction attribute), 164  
 type\_index (owlapy.owl\_ontology.OWLLiteral attribute), 151  
 type\_index (owlapy.owl\_ontology.OWLNamedIndividual attribute), 150  
 type\_index (owlapy.owl\_ontology.OWLObjectAllValuesFrom attribute), 157  
 type\_index (owlapy.owl\_ontology.OWLObjectComplementOf attribute), 156  
 type\_index (owlapy.owl\_ontology.OWLObjectExactCardinality attribute), 158  
 type\_index (owlapy.owl\_ontology.OWLObjectHasValue attribute), 159  
 type\_index (owlapy.owl\_ontology.OWLObjectIntersectionOf attribute), 156  
 type\_index (owlapy.owl\_ontology.OWLObjectInverseOf attribute), 166  
 type\_index (owlapy.owl\_ontology.OWLObjectMaxCardinality attribute), 158  
 type\_index (owlapy.owl\_ontology.OWLObjectMinCardinality attribute), 159  
 type\_index (owlapy.owl\_ontology.OWLObjectOneOf attribute), 157  
 type\_index (owlapy.owl\_ontology.OWLObjectProperty attribute), 164  
 type\_index (owlapy.owl\_ontology.OWLObjectSomeValuesFrom attribute), 156  
 type\_index (owlapy.owl\_ontology.OWLObjectUnionOf attribute), 156  
 type\_index (owlapy.owl\_ontology.OWLOntology attribute), 168  
 type\_index (owlapy.owl\_property.IRI attribute), 203  
 type\_index (owlapy.owl\_property.OWLDataProperty attribute), 207  
 type\_index (owlapy.owl\_property.OWLObjectInverseOf attribute), 206  
 type\_index (owlapy.owl\_property.OWLObjectProperty attribute), 205  
 type\_index (owlapy.owl\_reasoner.IRI attribute), 218  
 type\_index (owlapy.owl\_reasoner.OWLClass attribute), 217  
 type\_index (owlapy.owl\_reasoner.OWLDataAllValuesFrom attribute), 216  
 type\_index (owlapy.owl\_reasoner.OWLDataComplementOf attribute), 220  
 type\_index (owlapy.owl\_reasoner.OWLDataHasValue attribute), 216  
 type\_index (owlapy.owl\_reasoner.OWLDataIntersectionOf attribute), 220  
 type\_index (owlapy.owl\_reasoner.OWLDataOneOf attribute), 214  
 type\_index (owlapy.owl\_reasoner.OWLDataProperty attribute), 227  
 type\_index (owlapy.owl\_reasoner.OWLDataSomeValuesFrom attribute), 214  
 type\_index (owlapy.owl\_reasoner.OWLDatatype attribute), 221  
 type\_index (owlapy.owl\_reasoner.OWLDatatypeRestriction attribute), 215  
 type\_index (owlapy.owl\_reasoner.OWLDataUnionOf attribute), 220  
 type\_index (owlapy.owl\_reasoner.OWLFacetRestriction attribute), 215  
 type\_index (owlapy.owl\_reasoner.OWLLiteral attribute), 230  
 type\_index (owlapy.owl\_reasoner.OWLNamedIndividual attribute), 229  
 type\_index (owlapy.owl\_reasoner.OWLObjectAllValuesFrom attribute), 211  
 type\_index (owlapy.owl\_reasoner.OWLObjectComplementOf attribute), 210  
 type\_index (owlapy.owl\_reasoner.OWLObjectExactCardinality attribute), 213  
 type\_index (owlapy.owl\_reasoner.OWLObjectHasValue attribute), 212  
 type\_index (owlapy.owl\_reasoner.OWLObjectIntersectionOf attribute), 210  
 type\_index (owlapy.owl\_reasoner.OWLObjectInverseOf attribute), 228  
 type\_index (owlapy.owl\_reasoner.OWLObjectMaxCardinality attribute), 213  
 type\_index (owlapy.owl\_reasoner.OWLObjectMinCardinality attribute), 213



type\_index (owlapy.owl\_reasoner.OWLObjectOneOf attribute), 211  
 type\_index (owlapy.owl\_reasoner.OWLObjectProperty attribute), 227  
 type\_index (owlapy.owl\_reasoner.OWLObjectSomeValuesFrom attribute), 210  
 type\_index (owlapy.owl\_reasoner.OWLObjectUnionOf attribute), 210  
 type\_index (owlapy.owl\_reasoner.OWLOntology attribute), 221  
 type\_index (owlapy.parser.IRI attribute), 255  
 type\_index (owlapy.parser.OWLClass attribute), 264  
 type\_index (owlapy.parser.OWLDataAllValuesFrom attribute), 273  
 type\_index (owlapy.parser.OWLDataComplementOf attribute), 274  
 type\_index (owlapy.parser.OWLDataExactCardinality attribute), 268  
 type\_index (owlapy.parser.OWLDataHasValue attribute), 270  
 type\_index (owlapy.parser.OWLDataIntersectionOf attribute), 273  
 type\_index (owlapy.parser.OWLDataMaxCardinality attribute), 269  
 type\_index (owlapy.parser.OWLDataMinCardinality attribute), 270  
 type\_index (owlapy.parser.OWLDataOneOf attribute), 270  
 type\_index (owlapy.parser.OWLDataProperty attribute), 260  
 type\_index (owlapy.parser.OWLDataSomeValuesFrom attribute), 268  
 type\_index (owlapy.parser.OWLDatatype attribute), 261  
 type\_index (owlapy.parser.OWLDatatypeRestriction attribute), 271  
 type\_index (owlapy.parser.OWLDataUnionOf attribute), 273  
 type\_index (owlapy.parser.OWLFacetRestriction attribute), 267  
 type\_index (owlapy.parser.OWLLiteral attribute), 257  
 type\_index (owlapy.parser.OWLNamedIndividual attribute), 257  
 type\_index (owlapy.parser.OWLObjectAllValuesFrom attribute), 272  
 type\_index (owlapy.parser.OWLObjectComplementOf attribute), 266  
 type\_index (owlapy.parser.OWLObjectExactCardinality attribute), 266  
 type\_index (owlapy.parser.OWLObjectHasSelf attribute), 262  
 type\_index (owlapy.parser.OWLObjectHasValue attribute), 269  
 type\_index (owlapy.parser.OWLObjectIntersectionOf attribute), 263  
 type\_index (owlapy.parser.OWLObjectMaxCardinality attribute), 269  
 type\_index (owlapy.parser.OWLObjectMinCardinality attribute), 263  
 type\_index (owlapy.parser.OWLObjectOneOf attribute), 265  
 type\_index (owlapy.parser.OWLObjectProperty attribute), 259  
 type\_index (owlapy.parser.OWLObjectSomeValuesFrom attribute), 263  
 type\_index (owlapy.parser.OWLObjectUnionOf attribute), 264  
 type\_index (owlapy.providers.OWLDatatypeRestriction attribute), 281  
 type\_index (owlapy.providers.OWLFacetRestriction attribute), 282  
 type\_index (owlapy.providers.OWLLiteral attribute), 279  
 type\_index (owlapy.render.IRI attribute), 285  
 type\_index (owlapy.render.OWLClass attribute), 291  
 type\_index (owlapy.render.OWLDataAllValuesFrom attribute), 296  
 type\_index (owlapy.render.OWLDataComplementOf attribute), 300  
 type\_index (owlapy.render.OWLDataExactCardinality attribute), 297  
 type\_index (owlapy.render.OWLDataHasValue attribute), 296  
 type\_index (owlapy.render.OWLDataIntersectionOf attribute), 301  
 type\_index (owlapy.render.OWLDataMaxCardinality attribute), 298  
 type\_index (owlapy.render.OWLDataMinCardinality attribute), 297  
 type\_index (owlapy.render.OWLDataOneOf attribute), 298  
 type\_index (owlapy.render.OWLDataSomeValuesFrom attribute), 295  
 type\_index (owlapy.render.OWLDatatype attribute), 303  
 type\_index (owlapy.render.OWLDatatypeRestriction attribute), 302  
 type\_index (owlapy.render.OWLDataUnionOf attribute), 300  
 type\_index (owlapy.render.OWLFacetRestriction attribute), 301  
 type\_index (owlapy.render.OWLLiteral attribute), 287  
 type\_index (owlapy.render.OWLNamedIndividual attribute), 286  
 type\_index (owlapy.render.OWLObjectAllValuesFrom attribute), 293  
 type\_index (owlapy.render.OWLObjectComplementOf attribute), 293  
 type\_index (owlapy.render.OWLObjectExactCardinality attribute), 294  
 type\_index (owlapy.render.OWLObjectHasSelf attribute), 295  
 type\_index (owlapy.render.OWLObjectHasValue attribute), 301  
 type\_index (owlapy.render.OWLObjectIntersectionOf attribute), 293  
 type\_index (owlapy.render.OWLObjectInverseOf attribute), 289  
 type\_index (owlapy.render.OWLObjectMaxCardinality attribute), 295  
 type\_index (owlapy.render.OWLObjectMinCardinality attribute), 294  
 type\_index (owlapy.render.OWLObjectOneOf attribute), 302  
 type\_index (owlapy.render.OWLObjectSomeValuesFrom attribute), 292  
 type\_index (owlapy.render.OWLObjectUnionOf attribute), 293  
 type\_index (owlapy.utils.HasIndex attribute), 328

- `type_index (owlapy.utils.OWLClass attribute)`, 313
- `type_index (owlapy.utils.OWLDataAllValuesFrom attribute)`, 318
- `type_index (owlapy.utils.OWLDataComplementOf attribute)`, 323
- `type_index (owlapy.utils.OWLDataExactCardinality attribute)`, 317
- `type_index (owlapy.utils.OWLDataHasValue attribute)`, 318
- `type_index (owlapy.utils.OWLDataIntersectionOf attribute)`, 324
- `type_index (owlapy.utils.OWLDataMaxCardinality attribute)`, 317
- `type_index (owlapy.utils.OWLDataMinCardinality attribute)`, 317
- `type_index (owlapy.utils.OWLDataOneOf attribute)`, 320
- `type_index (owlapy.utils.OWLDataProperty attribute)`, 312
- `type_index (owlapy.utils.OWLDataSomeValuesFrom attribute)`, 319
- `type_index (owlapy.utils.OWLDataType attribute)`, 325
- `type_index (owlapy.utils.OWLDataTypeRestriction attribute)`, 322
- `type_index (owlapy.utils.OWLDataUnionOf attribute)`, 324
- `type_index (owlapy.utils.OWLFacetRestriction attribute)`, 322
- `type_index (owlapy.utils.OWLLiteral attribute)`, 309
- `type_index (owlapy.utils.OWLNamedIndividual attribute)`, 307
- `type_index (owlapy.utils.OWLObjectAllValuesFrom attribute)`, 319
- `type_index (owlapy.utils.OWLObjectComplementOf attribute)`, 314
- `type_index (owlapy.utils.OWLObjectExactCardinality attribute)`, 316
- `type_index (owlapy.utils.OWLObjectHasSelf attribute)`, 316
- `type_index (owlapy.utils.OWLObjectHasValue attribute)`, 321
- `type_index (owlapy.utils.OWLObjectIntersectionOf attribute)`, 320
- `type_index (owlapy.utils.OWLObjectInverseOf attribute)`, 311
- `type_index (owlapy.utils.OWLObjectMaxCardinality attribute)`, 316
- `type_index (owlapy.utils.OWLObjectMinCardinality attribute)`, 315
- `type_index (owlapy.utils.OWLObjectOneOf attribute)`, 323
- `type_index (owlapy.utils.OWLObjectProperty attribute)`, 311
- `type_index (owlapy.utils.OWLObjectSomeValuesFrom attribute)`, 315
- `type_index (owlapy.utils.OWLObjectUnionOf attribute)`, 321
- `type_index (owlapy.vocab.IRI attribute)`, 331
- `types () (owlapy.owl_hierarchy.OWLReasoner method)`, 128
- `types () (owlapy.owl_reasoner.FastInstanceCheckerReasoner method)`, 250
- `types () (owlapy.owl_reasoner.OntologyReasoner method)`, 245
- `types () (owlapy.owl_reasoner.OWLReasoner method)`, 237

## U

- `update_isolated_ontology () (owlapy.owl_reasoner.OntologyReasoner method)`, 239
- `update_isolated_ontology () (owlapy.owl_reasoner.SyncReasoner method)`, 252

## V

- `values () (owlapy.class_expression.OWLDataOneOf method)`, 53
- `values () (owlapy.class_expression.restriction.OWLDataOneOf method)`, 44
- `values () (owlapy.converter.OWLDataOneOf method)`, 73
- `values () (owlapy.owl_ontology.OWLDataOneOf method)`, 162
- `values () (owlapy.owl_reasoner.OWLDataOneOf method)`, 214
- `values () (owlapy.parser.OWLDataOneOf method)`, 270
- `values () (owlapy.render.OWLDataOneOf method)`, 298
- `values () (owlapy.utils.OWLDataOneOf method)`, 320
- `variable_entities (owlapy.converter.Owl2SparqlConverter attribute)`, 79
- `variables (owlapy.converter.Owl2SparqlConverter attribute)`, 79
- `VariablesMapping (class in owlapy.converter)`, 78
- `visit_abbreviated_iri () (owlapy.parser.DLSyntaxParser method)`, 278
- `visit_abbreviated_iri () (owlapy.parser.ManchesterOWLSyntaxParser method)`, 276
- `visit_boolean_literal () (owlapy.parser.DLSyntaxParser method)`, 277
- `visit_boolean_literal () (owlapy.parser.ManchesterOWLSyntaxParser method)`, 275
- `visit_cardinality_res () (owlapy.parser.DLSyntaxParser method)`, 277
- `visit_cardinality_res () (owlapy.parser.ManchesterOWLSyntaxParser method)`, 274
- `visit_class_expression () (owlapy.parser.DLSyntaxParser method)`, 277
- `visit_class_expression () (owlapy.parser.ManchesterOWLSyntaxParser method)`, 275
- `visit_class_iri () (owlapy.parser.DLSyntaxParser method)`, 278
- `visit_class_iri () (owlapy.parser.ManchesterOWLSyntaxParser method)`, 276
- `visit_data_cardinality_res () (owlapy.parser.DLSyntaxParser method)`, 277
- `visit_data_cardinality_res () (owlapy.parser.ManchesterOWLSyntaxParser method)`, 275
- `visit_data_intersection () (owlapy.parser.DLSyntaxParser method)`, 277
- `visit_data_intersection () (owlapy.parser.ManchesterOWLSyntaxParser method)`, 275
- `visit_data_parentheses () (owlapy.parser.DLSyntaxParser method)`, 277

`visit_data_parentheses()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_data_primary()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_data_primary()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_data_property_iri()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_data_property_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_data_some_only_res()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_data_some_only_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_data_union()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_data_union()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_data_value_res()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_data_value_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_datatype()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_datatype()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_datatype_iri()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_datatype_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_datatype_restriction()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_datatype_restriction()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_date_literal()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_date_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_datetime_literal()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_datetime_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_decimal_literal()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_decimal_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_duration_literal()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_duration_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_facet()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_facet()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_facet_restriction()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_facet_restriction()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_facet_restrictions()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_facet_restrictions()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_float_literal()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_float_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_full_iri()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_full_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_has_self()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_has_self()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_individual_iri()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_individual_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_individual_list()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_individual_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_integer_literal()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_integer_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_intersection()` (*owlapy.parser.DLSyntaxParser method*), 276  
`visit_intersection()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 274  
`visit_iri()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_literal()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_literal_list()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_literal_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_non_negative_integer()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_non_negative_integer()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_object_property()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_object_property()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_object_property_iri()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_object_property_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_parentheses()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_parentheses()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_primary()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_primary()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 274  
`visit_quoted_string()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_quoted_string()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_simple_iri()` (*owlapy.parser.DLSyntaxParser method*), 278  
`visit_simple_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 276  
`visit_some_only_res()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_some_only_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 274  
`visit_string_literal_language()` (*owlapy.parser.DLSyntaxParser method*), 277

`visit_string_literal_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_string_literal_no_language()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_string_literal_no_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_typed_literal()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_typed_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275  
`visit_union()` (*owlapy.parser.DLSyntaxParser method*), 276  
`visit_union()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 274  
`visit_value_res()` (*owlapy.parser.DLSyntaxParser method*), 277  
`visit_value_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 275

## W

`worst()` (*owlapy.utils.EvaluatedDescriptionSet method*), 328

## X

XSD (*in module owlapy.namespaces*), 85  
XSDVocabulary (*class in owlapy.owl\_literal*), 138  
XSDVocabulary (*class in owlapy.vocab*), 333