
OWLAPY

Release 0.1.2

Ontolearn Team

Jun 20, 2024

Contents:

1	About owlapy	2
1.1	What is owlapy?	2
1.2	What does owlapy have to offer?	2
1.3	How to install?	3
2	Basic Usage	3
2.1	Atomic Classes	4
2.2	Object Property	4
2.3	Complex class expressions	4
2.4	Convert to SPARQL, DL or Manchester syntax	5
3	Ontologies	6
3.1	Loading an Ontology	6
3.2	Modifying an Ontology	6
3.3	Save an Ontology	8
3.4	Worlds	8
4	Reasoners	9
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
5	Reasoning Details	12
5.1	Sync Reasoner	12
5.2	Isolated World	12
5.3	Capabilities	13
5.4	Concrete Example	14
6	Owlapi Adaptor	17
6.1	Initialization	17
6.2	Notes	17
6.3	Examples	17
7	owlapy	18
7.1	Subpackages	18
7.2	Submodules	63

7.3	Attributes	339
7.4	Functions	339
7.5	Package Contents	339
Python Module Index		340
Index		341

OWLAPY¹: 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: DICE² - data science research group of Paderborn University³.

Contact: 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](#)⁴ and [OntoSample](#)⁵.

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](#)⁶ including:
 - Entities, Literals, and Anonymous Individuals

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

² <https://dice-research.org/>

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

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

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

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

- Property Expressions
- Data Ranges
- Class Expressions
- Axioms
- 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`⁷:

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

Notice that we created an `IRI` object for every class. `IRI`⁸ 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`⁹:

```
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`¹⁰. 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
)
```

⁷ https://dice-group.github.io/owlapy/autoapi/owlapy/class_expression/owl_class/index.html#owlapy.class_expression.owl_class.OWLClass

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

⁹ https://dice-group.github.io/owlapy/autoapi/owlapy/owl_property/index.html#owlapy.owl_property.OWLObjectProperty

¹⁰ https://www.w3.org/TR/owl2-syntax/#Minimum_Cardinality

As you can see, to create an object of class `OWLObjectMinCardinality`¹¹ 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`¹²:

```
from owlapy.class_expression import OWLObjectIntersectionOf

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

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=OWLObjectClass(IRI('http://example.com/family#', 'male')))

print(manchester_to_owl_expression("female and (hasChild max 2 person)", namespace))
# Result: OWLObjectIntersectionOf((OWLObjectClass(IRI('http://example.com/family#', 'female
    ↳')), OWLObjectMaxCardinality(property=OWLObjectProperty(IRI('http://example.com/
    ↳family#', 'hasChild')), 2, filler=OWLObjectClass(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`¹³ directory.

¹¹ https://dice-group.github.io/owlapy/autoapi/owlapy/class_expression/restriction/index.html#owlapy.class_expression.restriction.OWLObjectMinCardinality

¹² https://dice-group.github.io/owlapy/autoapi/owlapy/class_expression/nary_boolean_expression/index.html#owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf

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

3 Ontologies

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

3.1 Loading an Ontology

To load an ontology as well as to manage it, you will need an *OWL ontology manager*. 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 ontology manager*. 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*

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

¹⁵ https://www.w3.org/TR/owl-syntax/#Named_Individuals

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 *OWL ontology* 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*¹⁶ or *Pellet*¹⁷ 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 (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.

¹⁶ <http://www.hermit-reasoner.com/>

¹⁷ <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/)¹⁸ 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.

¹⁸ <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](#)¹⁹ 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

¹⁹ 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 Owlapi Adaptor

As mentioned earlier, owlapy is loosely based in [owlapi](#)²⁰, a library for ontology modification in java.

We have created *OWLAPIAdaptor*, an adaptor class that facilitates the conversion of owl class expressions from owlapy to owlapi and vice-versa. This adaptor is still considered experimental, and it's in the initial phase of development.

We are able to use owlapi via [Jpype](#)²¹, a python module that provides access to Java via python. To start executing Java code via jpype, one needs to start the java virtual machine (JVM). This is automatically done when initializing a *OWLAPIAdaptor* object.

6.1 Initialization

To use the adaptor you have to initialize using the `with` statement in python. This way you will know where the JVM session starts and when it closes:

```
from owlapy.owlapi_adaptor import OWLAPIAdaptor

with OWLAPIAdaptor("KGs/Family/father.owl") as adaptor:
    # Use the adaptor
    print(f"Is the ontology consistent? {adaptor.has_consistent_ontology()}")

# The JVM will shut down when the thread is no longer used.
```

In the above code snippet, we created an adaptor for the father ontology by passing the local path of that ontology. Then we print whether the ontology is consistent or not.

6.2 Notes

An important note is that when initialising the adaptor you are basically starting a JVM in the background, and therefore you are able to import and use java classes as you would do in python. That means that you can play around with owlapi code in python. Isn't that awesome!

OWLAPIAdaptor uses *HermiT* reasoner for methods that require reasoning, such as `instances`, which returns all individuals belonging to a class expression.

6.3 Examples

You can check a usage example in the [examples](#)²² folder.

[Test cases](#)²³ for the adaptor can also serve as an example, so you can check that out as well.

²⁰ <https://github.com/owlcs/owlapi>

²¹ <https://jpype.readthedocs.io/en/latest/>

²² <https://github.com/dice-group/owlapy/tree/develop/examples>

²³ <https://github.com/dice-group/owlapy/tree/develop/tests>

7 owlapy

7.1 Subpackages

owlapy.class_expression

OWL 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)

__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)

__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)

```
    __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)

```
    __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> ₁ ... <i>a</i> _{<i>n</i>}) contains exactly the individuals <i>a</i> _{<i>i</i>} 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 ₁ ... DPE _{<i>n</i>} DR) consists of <i>n</i> data property expressions
<i>OWLDataAllValuesFrom</i>	A universal class expression <i>DataAllValuesFrom</i> (DPE ₁ ... DPE _{<i>n</i>} DR) consists of <i>n</i> data property expressions DPE _{<i>i</i>} ,
<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> ₁ ... <i>lt</i> _{<i>n</i>}) contains exactly the explicitly specified literals <i>lt</i> _{<i>i</i>} with
<i>OWLDatatypeRestriction</i>	A datatype restriction <i>DatatypeRestriction</i> (DT F ₁ <i>lt</i> ₁ ... F _{<i>n</i>} <i>lt</i> _{<i>n</i>}) 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.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 CE_i for 1 ≤ i ≤ n. (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
 A 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 $\text{inv}(P)$ 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)

```

__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)

```

__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)

```

__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)

```
__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)

```
__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)

```
__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 DPE_i only to literals lt_i, $1 \leq i \leq n$, 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)

```
__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)

__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)

```

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 $\text{ObjectSomeValuesFrom}(OPE\ CE)$ consists of an object property expression OPE and
<i>OWLObjectAllValuesFrom</i>	A universal class expression $\text{ObjectAllValuesFrom}(OPE\ CE)$ consists of an object property expression OPE and a
<i>OWLObjectHasValue</i>	A has-value class expression $\text{ObjectHasValue}(OPE\ a)$ consists of an object property expression OPE and an
<i>OWLDatatypeRestriction</i>	A datatype restriction $\text{DatatypeRestriction}(DT\ F_1\ l_1 \dots F_n\ l_n)$ consists of a unary datatype 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 $\text{ObjectMinCardinality}(n\ OPE\ CE)$ consists of a nonnegative integer n , an object
<i>OWLObjectMaxCardinality</i>	A maximum cardinality expression $\text{ObjectMaxCardinality}(n\ OPE\ CE)$ consists of a nonnegative integer n , an object
<i>OWLObjectExactCardinality</i>	An exact cardinality expression $\text{ObjectExactCardinality}(n\ OPE\ CE)$ consists of a nonnegative integer n , an object
<i>OWLDataSomeValuesFrom</i>	An existential class expression $\text{DataSomeValuesFrom}(DPE_1 \dots DPE_n\ DR)$ consists of n data property expressions
<i>OWLDataAllValuesFrom</i>	A universal class expression $\text{DataAllValuesFrom}(DPE_1 \dots DPE_n\ DR)$ consists of n data property expressions DPE_i ,
<i>OWLDataHasValue</i>	A has-value class expression $\text{DataHasValue}(DPE\ l)$ consists of a data property expression DPE and a literal l ,
<i>OWLDataMinCardinality</i>	A minimum cardinality expression $\text{DataMinCardinality}(n\ DPE\ DR)$ consists of a nonnegative integer n , a data
<i>OWLDataMaxCardinality</i>	A maximum cardinality expression $\text{ObjectMaxCardinality}(n\ OPE\ CE)$ consists of a nonnegative integer n , an object
<i>OWLDataExactCardinality</i>	An exact cardinality expression $\text{ObjectExactCardinality}(n\ OPE\ CE)$ 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)

__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)

__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)

__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)

```
__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 ($= 1 R C$) $=> 1 R C$ and $<= 1 R C$.

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

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

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

Bases: *OwlQuantifiedDataRestriction*

An existential class expression **DataSomeValuesFrom(DPE1 ... DPE_n 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 **lt_i**, $1 \leq i \leq n$, such that the tuple (**lt₁** , ..., **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)

```
__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(DPE1 ... DPE_n 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 only to literals lt_i, $1 \leq i \leq n$, such that each tuple (lt₁ , ..., 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)

```

__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)

```

__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)

```

__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)

__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)

__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)

__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.

7.2 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 _n DR) consists of n data property expressions
<i>OWLDataAllValuesFrom</i>	A universal class expression <i>DataAllValuesFrom</i> (DPE1 ... DPE _n DR) consists of n data property expressions DPE _i ,
<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 _n) contains exactly the explicitly specified literals It _i 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 (https://www.w3.org/TR/owl2-syntax/#Class_Expressions) 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 l1 ... Fn l_n) consists of a unary datatype DT and n pairs (Fi , l_i). 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 l_i. (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)

__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)

```
__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)

```
__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)

__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)

`__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)

`__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)

```
__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)

```
__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 `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.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* *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)

```
__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)

```
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)

```

__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)

__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)

__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) 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

<i>OWL</i>
<i>RDFS</i>
<i>RDF</i>
<i>XSD</i>

Classes

<i>Namespaces</i>	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>OWLDataPropertyExpression</i>	A high level interface to describe different types of data properties.
<i>OWLObjectPropertyExpression</i>	A high level interface to describe different types of object 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>OWLDatRange</i>	Represents a DataRange 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 <code>ObjectUnionOf(CE1 ... CEn)</code> 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 ... CE_n)</code> states that all of the class expressions CE _i ,
<i>OWLDisjointClassesAxiom</i>	A disjoint classes axiom <code>DisjointClasses(CE1 ... CE_n)</code> states that all of the class expressions CE _i , $1 \leq i \leq n$,
<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 ... a_n)</code> states that all of the individuals a _i ,
<i>OWLSameIndividualAxiom</i>	An individual equality axiom <code>SameIndividual(a1 ... a_n)</code> states that all of the individuals a _i , $1 \leq i \leq n$,
<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 ... OPE_n)</code> states that all of the object
<i>OWLDisjointObjectPropertiesAxiom</i>	A disjoint object properties axiom <code>DisjointObjectProperties(OPE1 ... OPE_n)</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 ... DPE_n)</code> states that all the data property
<i>OWLDisjointDataPropertiesAxiom</i>	A disjoint data properties axiom <code>DisjointDataProperties(DPE1 ... DPE_n)</code> states that all of the data property
<i>OWLSubClassOfAxiom</i>	A subclass axiom <code>SubClassOf(CE1 CE2)</code> states that the class expression CE1 is a subclass of the class
<i>OWLDisjointUnionAxiom</i>	A disjoint union axiom <code>DisjointUnion(C CE1 ... CE_n)</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 AP1 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 `equivalentClasses` 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)

__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)

__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#>](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.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 `equivalentClasses` 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*₁ ... *CE*_{*n*}) states that all of the class expressions *CE*_{*i*}, 1 ≤ *i* ≤ *n*, are semantically equivalent to each other. This axiom allows one to use each *CE*_{*i*} as a synonym for each *CE*_{*j*} — that is, in any expression in the ontology containing such an axiom, *CE*_{*i*} can be replaced with *CE*_{*j*} without affecting the meaning of the ontology.

(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*₁ ... *CE*_{*n*}) states that all of the class expressions *CE*_{*i*}, 1 ≤ *i* ≤ *n*, are pairwise disjoint; that is, no individual can be at the same time an instance of both *CE*_{*i*} and *CE*_{*j*} for *i* ≠ *j*.

(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 ... 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_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 ... 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_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 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_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 ≤ 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_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 ≤ 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_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)

__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)

```
__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)

```
__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)

__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)

__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)

__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)

__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)

__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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__slots__ = ()
```

owlapy.owl_data_ranges

OWL 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)

```
__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)

```
__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)

```
type_index: Final = 4002
```

```
get_data_range()  $\rightarrow$  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)

__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*) → *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*) → *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*) → *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)

```
__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)

__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)

__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> ₁ ... <i>It</i> _{<i>n</i>}) contains exactly the explicitly specified literals <i>It</i> _{<i>i</i>} with
<i>OWLDatatypeRestriction</i>	A datatype restriction <i>DatatypeRestriction</i> (DT <i>F</i> ₁ <i>It</i> ₁ ... <i>F</i> _{<i>n</i>} <i>It</i> _{<i>n</i>}) 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*₁ ... *CE*_{*n*}) states that all of the class expressions *CE*_{*i*}, 1 ≤ *i* ≤ *n*, are semantically equivalent to each other. This axiom allows one to use each *CE*_{*i*} as a synonym for each *CE*_{*j*} — that is, in any expression in the ontology containing such an axiom, *CE*_{*i*} can be replaced with *CE*_{*j*} without affecting the meaning of the ontology.

(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)

    __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)

    __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)

```

```
__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)

```
__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)

```
__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)

```
__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)

__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)

```

__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)

```

__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)

__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)

__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)

```
__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)

```
__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(DPE₁ ... DPE_n 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 lt_i, $1 \leq i \leq n$, such that the tuple (lt₁ , ..., 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)

```
__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)

```

__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)

```

__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)

__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)

__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)

__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($l_1 \dots l_n$) contains exactly the explicitly specified literals l_i 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.owl_ontology.OWLDatatypeRestriction (

type_: *owlapy.owl_datatype.OWLDatatype*,

facet_restrictions: *OWLFacetRestriction* | *Iterable*[*OWLFacetRestriction*])

Bases: *owlapy.owl_data_ranges.OWLDataRange*

A datatype restriction DatatypeRestriction(DT $F_1\ l_1 \dots F_n\ l_n$) consists of a unary datatype DT and n pairs ($F_i\ ,\ l_i$). 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_i . (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)

__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)

__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)

```
__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 lt)</code> consists of a data property expression DPE and a literal lt,
<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 _n) states that all of the data property
<i>OWLDisjointObjectPropertiesAxiom</i>	A disjoint object properties axiom <i>DisjointObjectProperties</i> (OPE1 ... OPE _n) states that all of the object
<i>OWLEquivalentDataPropertiesAxiom</i>	An equivalent data properties axiom <i>EquivalentDataProperties</i> (DPE1 ... DPE _n) states that all the data property
<i>OWLEquivalentObjectPropertiesAxiom</i>	An equivalent object properties axiom <i>EquivalentObjectProperties</i> (OPE1 ... OPE _n) 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 _n) states that all of the individuals a _i ,
<i>OWLDisjointClassesAxiom</i>	A disjoint classes axiom <i>DisjointClasses</i> (CE1 ... CE _n) states that all of the class expressions CE _i , 1 ≤ i ≤ n,
<i>OWLSameIndividualAxiom</i>	An individual equality axiom <i>SameIndividual</i> (a1 ... a _n) states that all of the individuals a _i , 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)

__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(*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.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)

__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)

```

__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)

```

__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)

```
__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)

```
__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)

```
__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)

```
__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)

__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)

__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)

```

__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**(*a*₁ ... *a*_{*n*}) states that all of the individuals *a*_{*i*}, 1 ≤ *i* ≤ *n*, are different from each other; that is, no individuals *a*_{*i*} and *a*_{*j*} 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_ontology_manager.**OWLDisjointClassesAxiom** (
 class_expressions: List[*owlapy.class_expression.OWLClassExpression*],
 annotations: Iterable[*OWLAnnotation*] | None = None)

Bases: *OWLNaryClassAxiom*

A disjoint classes axiom **DisjointClasses**(*CE*₁ ... *CE*_{*n*}) states that all of the class expressions *CE*_{*i*}, 1 ≤ *i* ≤ *n*, are pairwise disjoint; that is, no individual can be at the same time an instance of both *CE*_{*i*} and *CE*_{*j*} for *i* ≠ *j*.

(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**(*a*₁ ... *a*_{*n*}) states that all of the individuals *a*_{*i*}, 1 ≤ *i* ≤ *n*, are equal to each other. This axiom allows one to use each *a*_{*i*} as a synonym for each *a*_{*j*} — that is, in any expression in the ontology containing such an axiom, *a*_{*i*} can be replaced with *a*_{*j*} without affecting the meaning of the ontology.

(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)


```

__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)

__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)

```
__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)

```
__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.OWLOntology*,
import_declaration: *OWLImportsDeclaration*)

Bases: *OWLOntologyChange*

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: *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.

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)

__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)

__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)

```
__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_n DR)</code> consists of <i>n</i> data property expressions
<i>OWLDataOneOf</i>	An enumeration of literals <code>DataOneOf(lt1 ... lt_n)</code> contains exactly the explicitly specified literals <i>lt_i</i> with
<i>OWLDatatypeRestriction</i>	A datatype restriction <code>DatatypeRestriction(DT F1 lt1 ... Fn lt_n)</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_n DR)</code> consists of <i>n</i> data property expressions <i>DPE_i</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_n)</code> contains all tuples of literals that are contained in the at least
<i>OWLDataIntersectionOf</i>	An intersection data range <code>DataIntersectionOf(DR1 ... DR_n)</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)

__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)

```
__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)

```
__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)

__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)

__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)

__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)

```
__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)

```
__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)

```
__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_n DR) consists of n data property expressions DPE_i, 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_i 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)

__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)

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.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_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)

__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 OWL Ontology consists of a possibly empty set of OWL Axioms and a possibly empty set of OWL Annotations. 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 OWL Ontology Manager.

```
__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_datarange (*p: owlapy.owl_data_ranges.OWLDataRange*)

→ 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 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)

__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)

__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)

```
__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)

__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.OWL ontology*)

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: 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.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.

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, 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, 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 $N = \text{equivalent_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe owl:Thing}))$. If **direct** is True: then if N is not empty then the return value is N , else the return value is the result of $\text{super_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe owl:Thing}), \text{true})$. If **direct** is False: then the result of $\text{super_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe owl:Thing}), \text{false})$ 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 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.owlapi_adaptor

Classes

<i>OWLClassExpression</i>	OWL Class expressions represent sets of individuals by formally specifying conditions on the individuals' properties;
<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>OWLAPIAdaptor</i>	

Functions

<i>manchester_to_owl_expression</i> (manchester_ex ...)
<i>owl_expression_to_manchester</i> (\rightarrow str)

Module Contents

`owlapy.owlapi_adaptor.manchester_to_owl_expression` (*manchester_expression*: str,
namespace: str)

class `owlapy.owlapi_adaptor.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)

`__slots__` = ()

abstract `is_owl_thing`() \rightarrow 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.owlapi_adaptor.**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.owlapi_adaptor.**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.owlapi_adaptor.**owl_expression_to_manchester** (*o: owlapy.owl_object.OWLObject*)

→ str

class owlapy.owlapi_adaptor.**OWLAPIAdaptor** (*path: str*)

__enter__ ()

Initialization via the *with* statement

convert_to_owlapi (ce: *owlapy.class_expression.OWLClassExpression*)

Converts an owlapy ce to an owlapi ce.

Parameters

ce (*OWLClassExpression*) – class expression in owlapy format to be converted.

Returns

Class expression in owlapi format.

convert_from_owlapi (ce, namespace: str) → *owlapy.class_expression.OWLClassExpression*

Converts an owlapi ce to an owlapy ce.

Parameters

- **ce** – Class expression in owlapi format.
- **namespace** – Ontology's namespace where class expression belongs.

Returns

Class expression in owlapy format.

instances (ce: *owlapy.class_expression.OWLClassExpression*)

Get the instances for a given class expression using HermiT. :param ce: Class expression in owlapy format.

Returns

Individuals which are classified by the given class expression.

has_consistent_ontology () → bool

Check if the used ontology is consistent.

__exit__ (exc_type, exc_val, exc_tb)

Shuts down the java virtual machine hosted by jpye.

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.
<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 ObjectHasSelf(OPE) consists of an object property expression OPE,

continues on next page

Table 9 – continued from previous page

<i>OWLObjectIntersectionOf</i>	An intersection class expression <i>ObjectIntersectionOf</i> (CE1 ... CEn) contains all individuals that are instances
<i>OWLObjectMinCardinality</i>	A minimum cardinality expression <i>ObjectMinCardinality</i> (n OPE CE) consists of a nonnegative integer n, an object
<i>OWLObjectSomeValuesFrom</i>	An existential class expression <i>ObjectSomeValuesFrom</i> (OPE CE) consists of an object property expression OPE and
<i>OWLObjectUnionOf</i>	A union class expression <i>ObjectUnionOf</i> (CE1 ... CEn) 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 <i>ObjectOneOf</i> (a1 ... an) contains exactly the individuals ai 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 <i>ObjectComplementOf</i> class expression in the OWL 2 Specification.
<i>OWLObjectExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> (n OPE CE) 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>OWLDataSomeValuesFrom</i>	An existential class expression <i>DataSomeValuesFrom</i> (DPE1 ... DPEn DR) consists of n data property expressions
<i>OWLDataExactCardinality</i>	An exact cardinality expression <i>ObjectExactCardinality</i> (n OPE CE) consists of a nonnegative integer n, an
<i>OWLObjectHasValue</i>	A has-value class expression <i>ObjectHasValue</i> (OPE a) consists of an object property expression OPE and an
<i>OWLDataMaxCardinality</i>	A maximum cardinality expression <i>ObjectMaxCardinality</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>OWLDataMinCardinality</i>	A minimum cardinality expression <i>DataMinCardinality</i> (n DPE DR) consists of a nonnegative integer n, a data
<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

continues on next page

Table 9 – continued from previous page

<i>OWLDataAllValuesFrom</i>	A universal class expression <code>DataAllValuesFrom(DPE1 ... DPE_n DR)</code> consists of <i>n</i> data property expressions <code>DPE_i</code> ,
<i>OWLDataIntersectionOf</i>	An intersection data range <code>DataIntersectionOf(DR1 ... DR_n)</code> contains all tuples of literals that are contained
<i>OWLDataUnionOf</i>	A union data range <code>DataUnionOf(DR1 ... DR_n)</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
<i>OWLDataRange</i>	Represents a <code>DataRange</code> in the OWL 2 Specification.
<i>ManchesterOWLSyntaxParser</i>	Manchester Syntax parser to parse strings to <code>OWLClassExpressions</code> .
<i>DLSyntaxParser</i>	Description Logic Syntax parser to parse strings to <code>OWLClassExpressions</code> .

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: OWLObjectPropertyExpression

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*, OWLObjectProperty

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.parser.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)

__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_NOHING

    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 ≤ i ≤ 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 ≤ i ≤ 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' 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)

__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 ($= 1 \text{ R C} = \geq 1 \text{ R C}$ and $\leq 1 \text{ R C}$).

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 ... DPE_n DR) consists of n data property expressions DPE_i, 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_i to literals lti, 1 ≤ i ≤ n, such that the tuple (lt₁ , ..., 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)

__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)

__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) = >= 1 R D and <= 1 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)

```

__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 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.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( 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 {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 l1 ... 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.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( DPE1 ... DPEn DR ) consists of n data property expressions
DPEi, 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 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.parser.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.parser.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.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 DT and n 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)

    __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

<i>DLrenderer</i>
<i>ManchesterRenderer</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>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.
<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
<i>OWLObjectSomeValuesFrom</i>	An existential class expression ObjectSomeValuesFrom(OPE CE) consists of an object property expression OPE and
<i>OWLObjectAllValuesFrom</i>	A universal class expression ObjectAllValuesFrom(OPE CE) consists of an object property expression OPE and a

continues on next page

Table 10 – continued from previous page

<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 lt) consists of a data property expression DPE and a literal lt,
<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> (lt1 ... ltn) contains exactly the explicitly specified literals lti 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
<i>OWLObjectHasValue</i>	A has-value class expression <i>ObjectHasValue</i> (OPE a) 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 <i>DatatypeRestriction</i> (DT F1 lt1 ... Fn ltn) consists of a unary datatype DT and n pairs
<i>OWLObjectOneOf</i>	An enumeration of individuals <i>ObjectOneOf</i> (a1 ... an) contains exactly the individuals ai with $1 \leq i \leq n$.

continues on next page

Table 10 – continued from previous page

<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(→ str)

owl_expression_to_manchester(→ 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) → 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.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)

__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)

__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)

__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)

__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.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)

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

```
type_index: Final = 3010
```

```
class owlapy.render.OLObjectHasSelf(  
    property: owlapy.owl_property.OLObjectPropertyExpression)
```

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.OLObjectPropertyExpression
```

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 ... DPE_n 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 lti, $1 \leq i \leq n$, such that the tuple (lt₁ , ..., 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)

```
__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(DPE₁ ... DPE_n DR) consists of n data property expressions DPE_i, 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_i only to literals lti, 1 ≤ i ≤ n, such that each tuple (lt₁ , ..., 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)

```
__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)

```
__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)

```
__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)

```
__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)

```
__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)

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)

__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)

```

__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)

```
__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($a_1 \dots a_n$) contains exactly the individuals a_i 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.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

<i>move</i> (*args)	"Move" an imported class to the current module by setting the classes __module__ attribute.
<i>download_external_files</i> (ftp_link)	

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 _n DR) consists of <i>n</i> data property expressions DPE _i ,
<i>OWLDataSomeValuesFrom</i>	An existential class expression <i>DataSomeValuesFrom</i> (DPE1 ... DPE _n 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 _n) contains exactly the explicitly specified literals lt _i with
<i>OWLObjectIntersectionOf</i>	An intersection class expression <i>ObjectIntersectionOf</i> (CE1 ... CE _n) 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 _n) 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 _n lt _n) 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 _n) contains exactly the individuals a _i 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 _n) contains all tuples of literals that are contained in the at least
<i>OWLDataIntersectionOf</i>	An intersection data range <i>DataIntersectionOf</i> (DR1 ... DR _n) 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)

__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)

```
__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)

```
__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)

__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)

__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)

```
__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)

```
__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)

```
__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)

```

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

type_index: Final = 3017

```

class owlapy.utils.**OWLDataMinCardinality** (*cardinality: int*,
property: owlapy.owl_property.OWLDataPropertyExpression,
filler: owlapy.owl_data_range.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.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)

```

__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)

`__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.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_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 only to literals lti, $1 \leq i \leq n$, such that each tuple (lt₁ , ..., 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)

`__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*₁ ... *DPE*_{*n*} *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 *lti*, $1 \leq i \leq n$, such that the tuple (*lt*₁ , ..., *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)

```
__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)

```
__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* *F*₁ *l*₁ ... *F*_{*n*} *l*_{*n*}) consists of a unary datatype *DT* and *n* pairs (*F*_{*i*} , *l*_{*i*}). The resulting data range is unary and is obtained by restricting the value space of *DT* according to the semantics of all (*F*_{*i*} , *vi*) (multiple pairs are interpreted conjunctively), where *vi* are the data values of the literals *l*_{*i*}. (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)

```

__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)

```

__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.OWLRFDVocabulary (*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

7.3 Attributes

```
__version__
```

7.4 Functions

<code>owl_expression_to_dl(→ str)</code>	
<code>owl_expression_to_manchester(→ str)</code>	
<code>dl_to_owl_expression(dl_expression, namespace)</code>	
<code>manchester_to_owl_expression(manchester_ex ...)</code>	
<code>owl_expression_to_sparql(→ str)</code>	Convert an OWL Class Expression (https://www.w3.org/TR/owl2-syntax/#Class_Expressions) into a SPARQL query

7.5 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) 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`, 18
- `owlapy.class_expression`, 18
- `owlapy.class_expression.class_expression`, 18
- `owlapy.class_expression.nary_boolean_expression`, 21
- `owlapy.class_expression.owl_class`, 23
- `owlapy.class_expression.restriction`, 27
- `owlapy.converter`, 63
- `owlapy.entities`, 63
- `owlapy.iri`, 81
- `owlapy.meta_classes`, 84
- `owlapy.namespaces`, 85
- `owlapy.owl_annotation`, 86
- `owlapy.owl_axiom`, 87
- `owlapy.owl_data_ranges`, 116
- `owlapy.owl_datatype`, 118
- `owlapy.owl_hierarchy`, 121
- `owlapy.owl_individual`, 133
- `owlapy.owl_literal`, 136
- `owlapy.owl_object`, 142
- `owlapy.owl_ontology`, 145
- `owlapy.owl_ontology_manager`, 174
- `owlapy.owl_property`, 202
- `owlapy.owl_reasoner`, 208
- `owlapy.owlapi_adaptor`, 254
- `owlapy.parser`, 257
- `owlapy.providers`, 283
- `owlapy.render`, 288
- `owlapy.static_funcs`, 309
- `owlapy.utils`, 310
- `owlapy.vocab`, 335

Index

Non-alphabetical

`__contains__` () (owlapy.converter.VariablesMapping method), 79
`__contains__` () (owlapy.owl_hierarchy.AbstractHierarchy method), 132
`__contains__` () (owlapy.owl_reasoner.LRUCache method), 233
`__contains__` () (owlapy.utils.LRUCache method), 335
`__del__` () (owlapy.owl_reasoner.SyncReasoner method), 254
`__enter__` () (owlapy.owlapi_adaptor.OWLAPIAdaptor method), 256
`__eq__` () (owlapy.class_expression.class_expression.OWLObjectComplementOf method), 20
`__eq__` () (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression method), 22
`__eq__` () (owlapy.class_expression.owl_class.IRI method), 25
`__eq__` () (owlapy.class_expression.owl_class.OWLObjectComplementOf method), 24
`__eq__` () (owlapy.class_expression.OWLDataAllValuesFrom method), 60
`__eq__` () (owlapy.class_expression.OWLDataCardinalityRestriction method), 55
`__eq__` () (owlapy.class_expression.OWLDataHasValue method), 60
`__eq__` () (owlapy.class_expression.OWLDataOneOf method), 54
`__eq__` () (owlapy.class_expression.OWLDataSomeValuesFrom method), 59
`__eq__` () (owlapy.class_expression.OWLDatatypeRestriction method), 57
`__eq__` () (owlapy.class_expression.OWLFacetRestriction method), 58
`__eq__` () (owlapy.class_expression.OWLHasValueRestriction method), 52
`__eq__` () (owlapy.class_expression.OWLNaryBooleanClassExpression method), 50
`__eq__` () (owlapy.class_expression.OWLObjectAllValuesFrom method), 56
`__eq__` () (owlapy.class_expression.OWLObjectCardinalityRestriction method), 53
`__eq__` () (owlapy.class_expression.OWLObjectComplementOf method), 49
`__eq__` () (owlapy.class_expression.OWLObjectHasSelf method), 54
`__eq__` () (owlapy.class_expression.OWLObjectOneOf method), 62
`__eq__` () (owlapy.class_expression.OWLObjectSomeValuesFrom method), 55
`__eq__` () (owlapy.class_expression.restriction.OWLDataAllValuesFrom method), 44
`__eq__` () (owlapy.class_expression.restriction.OWLDataCardinalityRestriction method), 42
`__eq__` () (owlapy.class_expression.restriction.OWLDataHasValue method), 44
`__eq__` () (owlapy.class_expression.restriction.OWLDataOneOf method), 45
`__eq__` () (owlapy.class_expression.restriction.OWLDataSomeValuesFrom method), 43
`__eq__` () (owlapy.class_expression.restriction.OWLDatatypeRestriction method), 45
`__eq__` () (owlapy.class_expression.restriction.OWLFacetRestriction method), 46
`__eq__` () (owlapy.class_expression.restriction.OWLHasValueRestriction method), 36
`__eq__` () (owlapy.class_expression.restriction.OWLObject method), 34
`__eq__` () (owlapy.class_expression.restriction.OWLObjectAllValuesFrom method), 39
`__eq__` () (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction method), 37
`__eq__` () (owlapy.class_expression.restriction.OWLObjectHasSelf method), 40
`__eq__` () (owlapy.class_expression.restriction.OWLObjectOneOf method), 41
`__eq__` () (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom method), 39
`__eq__` () (owlapy.converter.OWLDataAllValuesFrom method), 73
`__eq__` () (owlapy.converter.OWLDataCardinalityRestriction method), 72
`__eq__` () (owlapy.converter.OWLDataHasValue method), 74
`__eq__` () (owlapy.converter.OWLDataOneOf method), 75
`__eq__` () (owlapy.converter.OWLDataSomeValuesFrom method), 73
`__eq__` () (owlapy.converter.OWLDatatypeRestriction method), 66
`__eq__` () (owlapy.converter.OWLObjectAllValuesFrom method), 70
`__eq__` () (owlapy.converter.OWLObjectCardinalityRestriction method), 71
`__eq__` () (owlapy.converter.OWLObjectComplementOf method), 69
`__eq__` () (owlapy.converter.OWLObjectHasSelf method), 72
`__eq__` () (owlapy.converter.OWLObjectOneOf method), 66
`__eq__` () (owlapy.converter.OWLObjectSomeValuesFrom method), 70
`__eq__` () (owlapy.iri.IRI method), 83
`__eq__` () (owlapy.iri.Namespaces method), 82
`__eq__` () (owlapy.namespaces.Namespaces method), 86
`__eq__` () (owlapy.owl_annotation.OWLObject method), 86
`__eq__` () (owlapy.owl_axiom.IRI method), 95
`__eq__` () (owlapy.owl_axiom.OWLAnnotation method), 107
`__eq__` () (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 108
`__eq__` () (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 109
`__eq__` () (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 109
`__eq__` () (owlapy.owl_axiom.OWLClassAssertionAxiom method), 106
`__eq__` () (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom method), 114
`__eq__` () (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method), 100

__eq__ () (owlapy.owl_axiom.OWLDeclarationAxiom method), 100
 __eq__ () (owlapy.owl_axiom.OWLDisjointUnionAxiom method), 106
 __eq__ () (owlapy.owl_axiom.OWLHasKeyAxiom method), 101
 __eq__ () (owlapy.owl_axiom.OWLNaryClassAxiom method), 102
 __eq__ () (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 103
 __eq__ () (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 104
 __eq__ () (owlapy.owl_axiom.OWLObject method), 91
 __eq__ () (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom method), 112
 __eq__ () (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 110
 __eq__ () (owlapy.owl_axiom.OWLPropertyDomainAxiom method), 114
 __eq__ () (owlapy.owl_axiom.OWLPropertyRangeAxiom method), 115
 __eq__ () (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 108
 __eq__ () (owlapy.owl_axiom.OWLSubClassOfAxiom method), 105
 __eq__ () (owlapy.owl_axiom.OWLSubPropertyAxiom method), 110
 __eq__ () (owlapy.owl_data_ranges.OWLDataComplementOf method), 118
 __eq__ () (owlapy.owl_data_ranges.OWLNaryDataRange method), 117
 __eq__ () (owlapy.owl_data_ranges.OWLObject method), 116
 __eq__ () (owlapy.owl_datatype.IRI method), 119
 __eq__ () (owlapy.owl_individual.IRI method), 134
 __eq__ () (owlapy.owl_individual.OWLObject method), 134
 __eq__ () (owlapy.owl_object.OWLNamedObject method), 144
 __eq__ () (owlapy.owl_object.OWLObject method), 143
 __eq__ () (owlapy.owl_ontology_manager.IRI method), 176
 __eq__ () (owlapy.owl_ontology_manager.Ontology method), 197
 __eq__ () (owlapy.owl_ontology_manager.OWLAnnotationAssertionAxiom method), 184
 __eq__ () (owlapy.owl_ontology_manager.OWLClassAssertionAxiom method), 186
 __eq__ () (owlapy.owl_ontology_manager.OWLDataHasValue method), 179
 __eq__ () (owlapy.owl_ontology_manager.OWLDataPropertyCharacteristicAxiom method), 188
 __eq__ () (owlapy.owl_ontology_manager.OWLDeclarationAxiom method), 186
 __eq__ () (owlapy.owl_ontology_manager.OWLDisjointUnionAxiom method), 184
 __eq__ () (owlapy.owl_ontology_manager.OWLNaryBooleanClassExpression method), 180
 __eq__ () (owlapy.owl_ontology_manager.OWLNaryIndividualAxiom method), 192
 __eq__ () (owlapy.owl_ontology_manager.OWLNaryPropertyAxiom method), 191
 __eq__ () (owlapy.owl_ontology_manager.OWLObject method), 178
 __eq__ () (owlapy.owl_ontology_manager.OWLObjectComplementOf method), 181
 __eq__ () (owlapy.owl_ontology_manager.OWLObjectInverseOf method), 198
 __eq__ () (owlapy.owl_ontology_manager.OWLObjectOneOf method), 181
 __eq__ () (owlapy.owl_ontology_manager.OWLObjectPropertyCharacteristicAxiom method), 189
 __eq__ () (owlapy.owl_ontology_manager.OWLPropertyDomainAxiom method), 187
 __eq__ () (owlapy.owl_ontology_manager.OWLPropertyRangeAxiom method), 185
 __eq__ () (owlapy.owl_ontology_manager.OWLSubClassOfAxiom method), 183
 __eq__ () (owlapy.owl_ontology_manager.OWLSubPropertyAxiom method), 185
 __eq__ () (owlapy.owl_ontology.IRI method), 154
 __eq__ () (owlapy.owl_ontology.Ontology method), 173
 __eq__ () (owlapy.owl_ontology.OWLDataAllValuesFrom method), 161
 __eq__ () (owlapy.owl_ontology.OWLDataComplementOf method), 150
 __eq__ () (owlapy.owl_ontology.OWLDataHasValue method), 162
 __eq__ () (owlapy.owl_ontology.OWLDataOneOf method), 163
 __eq__ () (owlapy.owl_ontology.OWLDataSomeValuesFrom method), 161
 __eq__ () (owlapy.owl_ontology.OWLDatatypeRestriction method), 163
 __eq__ () (owlapy.owl_ontology.OWLFacetRestriction method), 165
 __eq__ () (owlapy.owl_ontology.OWLObject method), 154
 __eq__ () (owlapy.owl_ontology.OWLObjectAllValuesFrom method), 158
 __eq__ () (owlapy.owl_ontology.OWLObjectComplementOf method), 157
 __eq__ () (owlapy.owl_ontology.OWLObjectInverseOf method), 167
 __eq__ () (owlapy.owl_ontology.OWLObjectOneOf method), 159
 __eq__ () (owlapy.owl_ontology.OWLObjectSomeValuesFrom method), 158
 __eq__ () (owlapy.owl_ontology.OWLOntologyID method), 169
 __eq__ () (owlapy.owl_ontology.OWLSubClassOfAxiom method), 149
 __eq__ () (owlapy.owl_property.IRI method), 204
 __eq__ () (owlapy.owl_property.OWLObject method), 203
 __eq__ () (owlapy.owl_property.OWLObjectInverseOf method), 207
 __eq__ () (owlapy.owl_reasoner.IRI method), 219
 __eq__ () (owlapy.owl_reasoner.Ontology method), 226
 __eq__ () (owlapy.owl_reasoner.OWLDataAllValuesFrom method), 218
 __eq__ () (owlapy.owl_reasoner.OWLDataComplementOf method), 221
 __eq__ () (owlapy.owl_reasoner.OWLDataHasValue method), 217
 __eq__ () (owlapy.owl_reasoner.OWLDataOneOf method), 216

- __eq__ (owlapy.owl_reasoner.OWLDataSomeValuesFrom method), 215
- __eq__ (owlapy.owl_reasoner.OWLDatatypeRestriction method), 216
- __eq__ (owlapy.owl_reasoner.OWLFacetRestriction method), 216
- __eq__ (owlapy.owl_reasoner.OWLObjectAllValuesFrom method), 212
- __eq__ (owlapy.owl_reasoner.OWLObjectCardinalityRestriction method), 215
- __eq__ (owlapy.owl_reasoner.OWLObjectComplementOf method), 212
- __eq__ (owlapy.owl_reasoner.OWLObjectInverseOf method), 229
- __eq__ (owlapy.owl_reasoner.OWLObjectOneOf method), 213
- __eq__ (owlapy.owl_reasoner.OWLObjectSomeValuesFrom method), 211
- __eq__ (owlapy.owl_reasoner.OWLSubClassOfAxiom method), 220
- __eq__ (owlapy.owlapi_adaptor.IRI method), 255
- __eq__ (owlapy.parser.IRI method), 260
- __eq__ (owlapy.parser.Namespaces method), 266
- __eq__ (owlapy.parser.OWLDataAllValuesFrom method), 278
- __eq__ (owlapy.parser.OWLDataCardinalityRestriction method), 277
- __eq__ (owlapy.parser.OWLDataComplementOf method), 279
- __eq__ (owlapy.parser.OWLDataHasValue method), 275
- __eq__ (owlapy.parser.OWLDataOneOf method), 276
- __eq__ (owlapy.parser.OWLDataSomeValuesFrom method), 273
- __eq__ (owlapy.parser.OWLDatatypeRestriction method), 276
- __eq__ (owlapy.parser.OWLFacetRestriction method), 272
- __eq__ (owlapy.parser.OWLObjectAllValuesFrom method), 277
- __eq__ (owlapy.parser.OWLObjectCardinalityRestriction method), 276
- __eq__ (owlapy.parser.OWLObjectComplementOf method), 271
- __eq__ (owlapy.parser.OWLObjectHasSelf method), 267
- __eq__ (owlapy.parser.OWLObjectOneOf method), 270
- __eq__ (owlapy.parser.OWLObjectSomeValuesFrom method), 268
- __eq__ (owlapy.providers.OWLDatatypeRestriction method), 286
- __eq__ (owlapy.providers.OWLFacetRestriction method), 287
- __eq__ (owlapy.render.IRI method), 290
- __eq__ (owlapy.render.OWLDataAllValuesFrom method), 301
- __eq__ (owlapy.render.OWLDataComplementOf method), 305
- __eq__ (owlapy.render.OWLDataHasValue method), 301
- __eq__ (owlapy.render.OWLDataOneOf method), 303
- __eq__ (owlapy.render.OWLDataSomeValuesFrom method), 300
- __eq__ (owlapy.render.OWLDatatypeRestriction method), 307
- __eq__ (owlapy.render.OWLFacetRestriction method), 306
- __eq__ (owlapy.render.OWLNaryBooleanClassExpression method), 303
- __eq__ (owlapy.render.OWLNaryDataRange method), 305
- __eq__ (owlapy.render.OWLObject method), 294
- __eq__ (owlapy.render.OWLObjectAllValuesFrom method), 298
- __eq__ (owlapy.render.OWLObjectComplementOf method), 299
- __eq__ (owlapy.render.OWLObjectHasSelf method), 300
- __eq__ (owlapy.render.OWLObjectInverseOf method), 295
- __eq__ (owlapy.render.OWLObjectOneOf method), 307
- __eq__ (owlapy.render.OWLObjectSomeValuesFrom method), 297
- __eq__ (owlapy.utils.HasIndex method), 333
- __eq__ (owlapy.utils.OrderedOWLObject method), 333
- __eq__ (owlapy.utils.OWLDataAllValuesFrom method), 323
- __eq__ (owlapy.utils.OWLDataCardinalityRestriction method), 325
- __eq__ (owlapy.utils.OWLDataComplementOf method), 329
- __eq__ (owlapy.utils.OWLDataHasValue method), 323
- __eq__ (owlapy.utils.OWLDataOneOf method), 325
- __eq__ (owlapy.utils.OWLDataSomeValuesFrom method), 324
- __eq__ (owlapy.utils.OWLDatatypeRestriction method), 327
- __eq__ (owlapy.utils.OWLFacetRestriction method), 327
- __eq__ (owlapy.utils.OWLNaryBooleanClassExpression method), 326
- __eq__ (owlapy.utils.OWLNaryDataRange method), 329
- __eq__ (owlapy.utils.OWLObject method), 330
- __eq__ (owlapy.utils.OWLObjectAllValuesFrom method), 324
- __eq__ (owlapy.utils.OWLObjectCardinalityRestriction method), 319
- __eq__ (owlapy.utils.OWLObjectComplementOf method), 319
- __eq__ (owlapy.utils.OWLObjectHasSelf method), 321
- __eq__ (owlapy.utils.OWLObjectInverseOf method), 316
- __eq__ (owlapy.utils.OWLObjectOneOf method), 328
- __eq__ (owlapy.utils.OWLObjectSomeValuesFrom method), 320
- __eq__ (owlapy.vocab.IRI method), 336
- __eq__ (owlapy.vocab.Namespaces method), 337

__exit__ () (owlapy.owlapi_adaptor.OWLAPIAdaptor method), 257
__getitem__ () (owlapy.converter.VariablesMapping method), 80
__getitem__ () (owlapy.owl_reasoner.LRUCache method), 233
__getitem__ () (owlapy.utils.LRUCache method), 335
__hash__ () (owlapy.class_expression.class_expression.OWLObjectComplementOf method), 20
__hash__ () (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression method), 22
__hash__ () (owlapy.class_expression.owl_class.IRI method), 25
__hash__ () (owlapy.class_expression.owl_class.OWLObjectComplementOf method), 24
__hash__ () (owlapy.class_expression.OWLDataAllValuesFrom method), 60
__hash__ () (owlapy.class_expression.OWLDataCardinalityRestriction method), 55
__hash__ () (owlapy.class_expression.OWLDataHasValue method), 60
__hash__ () (owlapy.class_expression.OWLDataOneOf method), 54
__hash__ () (owlapy.class_expression.OWLDataSomeValuesFrom method), 59
__hash__ () (owlapy.class_expression.OWLDatatypeRestriction method), 57
__hash__ () (owlapy.class_expression.OWLFacetRestriction method), 58
__hash__ () (owlapy.class_expression.OWLHasValueRestriction method), 52
__hash__ () (owlapy.class_expression.OWLNaryBooleanClassExpression method), 50
__hash__ () (owlapy.class_expression.OWLObjectAllValuesFrom method), 56
__hash__ () (owlapy.class_expression.OWLObjectCardinalityRestriction method), 53
__hash__ () (owlapy.class_expression.OWLObjectComplementOf method), 49
__hash__ () (owlapy.class_expression.OWLObjectHasSelf method), 54
__hash__ () (owlapy.class_expression.OWLObjectOneOf method), 62
__hash__ () (owlapy.class_expression.OWLObjectSomeValuesFrom method), 55
__hash__ () (owlapy.class_expression.restriction.OWLDataAllValuesFrom method), 44
__hash__ () (owlapy.class_expression.restriction.OWLDataCardinalityRestriction method), 42
__hash__ () (owlapy.class_expression.restriction.OWLDataHasValue method), 44
__hash__ () (owlapy.class_expression.restriction.OWLDataOneOf method), 45
__hash__ () (owlapy.class_expression.restriction.OWLDataSomeValuesFrom method), 43
__hash__ () (owlapy.class_expression.restriction.OWLDatatypeRestriction method), 45
__hash__ () (owlapy.class_expression.restriction.OWLFacetRestriction method), 46
__hash__ () (owlapy.class_expression.restriction.OWLHasValueRestriction method), 36
__hash__ () (owlapy.class_expression.restriction.OWLObject method), 35
__hash__ () (owlapy.class_expression.restriction.OWLObjectAllValuesFrom method), 39
__hash__ () (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction method), 38
__hash__ () (owlapy.class_expression.restriction.OWLObjectHasSelf method), 40
__hash__ () (owlapy.class_expression.restriction.OWLObjectOneOf method), 41
__hash__ () (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom method), 39
__hash__ () (owlapy.converter.OWLDataAllValuesFrom method), 73
__hash__ () (owlapy.converter.OWLDataCardinalityRestriction method), 72
__hash__ () (owlapy.converter.OWLDataHasValue method), 74
__hash__ () (owlapy.converter.OWLDataOneOf method), 75
__hash__ () (owlapy.converter.OWLDataSomeValuesFrom method), 73
__hash__ () (owlapy.converter.OWLDatatypeRestriction method), 66
__hash__ () (owlapy.converter.OWLObjectAllValuesFrom method), 70
__hash__ () (owlapy.converter.OWLObjectCardinalityRestriction method), 71
__hash__ () (owlapy.converter.OWLObjectComplementOf method), 69
__hash__ () (owlapy.converter.OWLObjectHasSelf method), 72
__hash__ () (owlapy.converter.OWLObjectOneOf method), 66
__hash__ () (owlapy.converter.OWLObjectSomeValuesFrom method), 70
__hash__ () (owlapy.iri.IRI method), 83
__hash__ () (owlapy.iri.Namespaces method), 82
__hash__ () (owlapy.namespaces.Namespaces method), 86
__hash__ () (owlapy.owl_annotation.OWLObject method), 86
__hash__ () (owlapy.owl_axiom.IRI method), 95
__hash__ () (owlapy.owl_axiom.OWLAnnotation method), 107
__hash__ () (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 108
__hash__ () (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 109
__hash__ () (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 109
__hash__ () (owlapy.owl_axiom.OWLClassAssertionAxiom method), 106
__hash__ () (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom method), 114
__hash__ () (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method), 100
__hash__ () (owlapy.owl_axiom.OWLDeclarationAxiom method), 100
__hash__ () (owlapy.owl_axiom.OWLDisjointUnionAxiom method), 106
__hash__ () (owlapy.owl_axiom.OWLHasKeyAxiom method), 101
__hash__ () (owlapy.owl_axiom.OWLNaryClassAxiom method), 102
__hash__ () (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 103
__hash__ () (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 104
__hash__ () (owlapy.owl_axiom.OWLObject method), 91

__hash__ () (owlapy.owl_axiom.OwlObjectPropertyCharacteristicAxiom method), 112
 __hash__ () (owlapy.owl_axiom.OwlPropertyAssertionAxiom method), 111
 __hash__ () (owlapy.owl_axiom.OwlPropertyDomainAxiom method), 114
 __hash__ () (owlapy.owl_axiom.OwlPropertyRangeAxiom method), 115
 __hash__ () (owlapy.owl_axiom.OwlSubAnnotationPropertyOfAxiom method), 108
 __hash__ () (owlapy.owl_axiom.OwlSubClassOfAxiom method), 105
 __hash__ () (owlapy.owl_axiom.OwlSubPropertyAxiom method), 110
 __hash__ () (owlapy.owl_data_ranges.OwlDataComplementOf method), 118
 __hash__ () (owlapy.owl_data_ranges.OwlNaryDataRange method), 117
 __hash__ () (owlapy.owl_data_ranges.OwlObject method), 116
 __hash__ () (owlapy.owl_datatype.IRI method), 119
 __hash__ () (owlapy.owl_individual.IRI method), 135
 __hash__ () (owlapy.owl_individual.OwlObject method), 134
 __hash__ () (owlapy.owl_object.OwlNamedObject method), 144
 __hash__ () (owlapy.owl_object.OwlObject method), 143
 __hash__ () (owlapy.owl_ontology_manager.IRI method), 176
 __hash__ () (owlapy.owl_ontology_manager.Ontology method), 197
 __hash__ () (owlapy.owl_ontology_manager.OwlAnnotationAssertionAxiom method), 184
 __hash__ () (owlapy.owl_ontology_manager.OwlClassAssertionAxiom method), 186
 __hash__ () (owlapy.owl_ontology_manager.OwlDataHasValue method), 179
 __hash__ () (owlapy.owl_ontology_manager.OwlDataPropertyCharacteristicAxiom method), 188
 __hash__ () (owlapy.owl_ontology_manager.OwlDeclarationAxiom method), 186
 __hash__ () (owlapy.owl_ontology_manager.OwlDisjointUnionAxiom method), 184
 __hash__ () (owlapy.owl_ontology_manager.OwlNaryBooleanClassExpression method), 180
 __hash__ () (owlapy.owl_ontology_manager.OwlNaryIndividualAxiom method), 192
 __hash__ () (owlapy.owl_ontology_manager.OwlNaryPropertyAxiom method), 191
 __hash__ () (owlapy.owl_ontology_manager.OwlObject method), 178
 __hash__ () (owlapy.owl_ontology_manager.OwlObjectComplementOf method), 181
 __hash__ () (owlapy.owl_ontology_manager.OwlObjectInverseOf method), 198
 __hash__ () (owlapy.owl_ontology_manager.OwlObjectOneOf method), 181
 __hash__ () (owlapy.owl_ontology_manager.OwlObjectPropertyCharacteristicAxiom method), 189
 __hash__ () (owlapy.owl_ontology_manager.OwlPropertyDomainAxiom method), 187
 __hash__ () (owlapy.owl_ontology_manager.OwlPropertyRangeAxiom method), 185
 __hash__ () (owlapy.owl_ontology_manager.OwlSubClassOfAxiom method), 183
 __hash__ () (owlapy.owl_ontology_manager.OwlSubPropertyAxiom method), 185
 __hash__ () (owlapy.owl_ontology.IRI method), 154
 __hash__ () (owlapy.owl_ontology.Ontology method), 173
 __hash__ () (owlapy.owl_ontology.OwlDataAllValuesFrom method), 161
 __hash__ () (owlapy.owl_ontology.OwlDataComplementOf method), 150
 __hash__ () (owlapy.owl_ontology.OwlDataHasValue method), 162
 __hash__ () (owlapy.owl_ontology.OwlDataOneOf method), 163
 __hash__ () (owlapy.owl_ontology.OwlDataSomeValuesFrom method), 161
 __hash__ () (owlapy.owl_ontology.OwlDatatypeRestriction method), 164
 __hash__ () (owlapy.owl_ontology.OwlFacetRestriction method), 165
 __hash__ () (owlapy.owl_ontology.OwlObject method), 154
 __hash__ () (owlapy.owl_ontology.OwlObjectAllValuesFrom method), 158
 __hash__ () (owlapy.owl_ontology.OwlObjectComplementOf method), 157
 __hash__ () (owlapy.owl_ontology.OwlObjectInverseOf method), 167
 __hash__ () (owlapy.owl_ontology.OwlObjectOneOf method), 159
 __hash__ () (owlapy.owl_ontology.OwlObjectSomeValuesFrom method), 158
 __hash__ () (owlapy.owl_ontology.OwlSubClassOfAxiom method), 149
 __hash__ () (owlapy.owl_property.IRI method), 204
 __hash__ () (owlapy.owl_property.OwlObject method), 203
 __hash__ () (owlapy.owl_property.OwlObjectInverseOf method), 207
 __hash__ () (owlapy.owl_reasoner.IRI method), 219
 __hash__ () (owlapy.owl_reasoner.Ontology method), 226
 __hash__ () (owlapy.owl_reasoner.OwlDataAllValuesFrom method), 218
 __hash__ () (owlapy.owl_reasoner.OwlDataComplementOf method), 221
 __hash__ () (owlapy.owl_reasoner.OwlDataHasValue method), 217
 __hash__ () (owlapy.owl_reasoner.OwlDataOneOf method), 216
 __hash__ () (owlapy.owl_reasoner.OwlDataSomeValuesFrom method), 215
 __hash__ () (owlapy.owl_reasoner.OwlDatatypeRestriction method), 216
 __hash__ () (owlapy.owl_reasoner.OwlFacetRestriction method), 216
 __hash__ () (owlapy.owl_reasoner.OwlObjectAllValuesFrom method), 212
 __hash__ () (owlapy.owl_reasoner.OwlObjectCardinalityRestriction method), 215
 __hash__ () (owlapy.owl_reasoner.OwlObjectComplementOf method), 212
 __hash__ () (owlapy.owl_reasoner.OwlObjectInverseOf method), 229
 __hash__ () (owlapy.owl_reasoner.OwlObjectOneOf method), 213

__hash__ () (owlapy.owl_reasoner.OWLObjectSomeValuesFrom method), 211
 __hash__ () (owlapy.owl_reasoner.OWLSubClassOfAxiom method), 220
 __hash__ () (owlapy.owlapi_adaptor.IRI method), 255
 __hash__ () (owlapy.parser.IRI method), 260
 __hash__ () (owlapy.parser.Namespaces method), 265
 __hash__ () (owlapy.parser.OWLDataAllValuesFrom method), 278
 __hash__ () (owlapy.parser.OWLDataCardinalityRestriction method), 277
 __hash__ () (owlapy.parser.OWLDataComplementOf method), 279
 __hash__ () (owlapy.parser.OWLDataHasValue method), 275
 __hash__ () (owlapy.parser.OWLDataOneOf method), 275
 __hash__ () (owlapy.parser.OWLDataSomeValuesFrom method), 273
 __hash__ () (owlapy.parser.OWLDatatypeRestriction method), 276
 __hash__ () (owlapy.parser.OWLFacetRestriction method), 272
 __hash__ () (owlapy.parser.OWLObjectAllValuesFrom method), 277
 __hash__ () (owlapy.parser.OWLObjectCardinalityRestriction method), 276
 __hash__ () (owlapy.parser.OWLObjectComplementOf method), 271
 __hash__ () (owlapy.parser.OWLObjectHasSelf method), 267
 __hash__ () (owlapy.parser.OWLObjectOneOf method), 270
 __hash__ () (owlapy.parser.OWLObjectSomeValuesFrom method), 268
 __hash__ () (owlapy.providers.OWLDatatypeRestriction method), 286
 __hash__ () (owlapy.providers.OWLFacetRestriction method), 287
 __hash__ () (owlapy.render.IRI method), 290
 __hash__ () (owlapy.render.OWLDataAllValuesFrom method), 301
 __hash__ () (owlapy.render.OWLDataComplementOf method), 305
 __hash__ () (owlapy.render.OWLDataHasValue method), 301
 __hash__ () (owlapy.render.OWLDataOneOf method), 303
 __hash__ () (owlapy.render.OWLDataSomeValuesFrom method), 300
 __hash__ () (owlapy.render.OWLDatatypeRestriction method), 307
 __hash__ () (owlapy.render.OWLFacetRestriction method), 306
 __hash__ () (owlapy.render.OWLNaryBooleanClassExpression method), 303
 __hash__ () (owlapy.render.OWLNaryDataRange method), 305
 __hash__ () (owlapy.render.OWLObject method), 294
 __hash__ () (owlapy.render.OWLObjectAllValuesFrom method), 298
 __hash__ () (owlapy.render.OWLObjectComplementOf method), 299
 __hash__ () (owlapy.render.OWLObjectHasSelf method), 300
 __hash__ () (owlapy.render.OWLObjectInverseOf method), 295
 __hash__ () (owlapy.render.OWLObjectOneOf method), 307
 __hash__ () (owlapy.render.OWLObjectSomeValuesFrom method), 297
 __hash__ () (owlapy.utils.OWLDataAllValuesFrom method), 323
 __hash__ () (owlapy.utils.OWLDataCardinalityRestriction method), 326
 __hash__ () (owlapy.utils.OWLDataComplementOf method), 329
 __hash__ () (owlapy.utils.OWLDataHasValue method), 323
 __hash__ () (owlapy.utils.OWLDataOneOf method), 325
 __hash__ () (owlapy.utils.OWLDataSomeValuesFrom method), 324
 __hash__ () (owlapy.utils.OWLDatatypeRestriction method), 327
 __hash__ () (owlapy.utils.OWLFacetRestriction method), 327
 __hash__ () (owlapy.utils.OWLNaryBooleanClassExpression method), 326
 __hash__ () (owlapy.utils.OWLNaryDataRange method), 329
 __hash__ () (owlapy.utils.OWLObject method), 330
 __hash__ () (owlapy.utils.OWLObjectAllValuesFrom method), 324
 __hash__ () (owlapy.utils.OWLObjectCardinalityRestriction method), 319
 __hash__ () (owlapy.utils.OWLObjectComplementOf method), 319
 __hash__ () (owlapy.utils.OWLObjectHasSelf method), 321
 __hash__ () (owlapy.utils.OWLObjectInverseOf method), 316
 __hash__ () (owlapy.utils.OWLObjectOneOf method), 328
 __hash__ () (owlapy.utils.OWLObjectSomeValuesFrom method), 320
 __hash__ () (owlapy.vocab.IRI method), 336
 __hash__ () (owlapy.vocab.Namespaces method), 337
 __iter__ () (owlapy.utils.EvaluatedDescriptionSet method), 333
 __len__ () (owlapy.owl_hierarchy.AbstractHierarchy method), 132
 __lt__ () (owlapy.owl_object.OWLNamedObject method), 144
 __lt__ () (owlapy.utils.OrderedOWLObject method), 333
 __repr__ () (owlapy.class_expression.class_expression.OWLObjectComplementOf method), 20
 __repr__ () (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression method), 22
 __repr__ () (owlapy.class_expression.owl_class.IRI method), 25
 __repr__ () (owlapy.class_expression.owl_class.OWLObjectComplementOf method), 24
 __repr__ () (owlapy.class_expression.OWLDataAllValuesFrom method), 60
 __repr__ () (owlapy.class_expression.OWLDataCardinalityRestriction method), 55

__repr__ () (owlapy.class_expression.OWLDataHasValue method), 60
__repr__ () (owlapy.class_expression.OWLDataOneOf method), 54
__repr__ () (owlapy.class_expression.OWLDataSomeValuesFrom method), 59
__repr__ () (owlapy.class_expression.OWLDatatypeRestriction method), 57
__repr__ () (owlapy.class_expression.OWLFacetRestriction method), 58
__repr__ () (owlapy.class_expression.OWLNaryBooleanClassExpression method), 50
__repr__ () (owlapy.class_expression.OWLObjectAllValuesFrom method), 56
__repr__ () (owlapy.class_expression.OWLObjectCardinalityRestriction method), 53
__repr__ () (owlapy.class_expression.OWLObjectComplementOf method), 49
__repr__ () (owlapy.class_expression.OWLObjectHasSelf method), 54
__repr__ () (owlapy.class_expression.OWLObjectHasValue method), 56
__repr__ () (owlapy.class_expression.OWLObjectOneOf method), 62
__repr__ () (owlapy.class_expression.OWLObjectSomeValuesFrom method), 55
__repr__ () (owlapy.class_expression.restriction.OWLDataAllValuesFrom method), 44
__repr__ () (owlapy.class_expression.restriction.OWLDataCardinalityRestriction method), 42
__repr__ () (owlapy.class_expression.restriction.OWLDataHasValue method), 44
__repr__ () (owlapy.class_expression.restriction.OWLDataOneOf method), 45
__repr__ () (owlapy.class_expression.restriction.OWLDataSomeValuesFrom method), 43
__repr__ () (owlapy.class_expression.restriction.OWLDatatypeRestriction method), 45
__repr__ () (owlapy.class_expression.restriction.OWLFacetRestriction method), 46
__repr__ () (owlapy.class_expression.restriction.OWLObject method), 35
__repr__ () (owlapy.class_expression.restriction.OWLObjectAllValuesFrom method), 39
__repr__ () (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction method), 37
__repr__ () (owlapy.class_expression.restriction.OWLObjectHasSelf method), 40
__repr__ () (owlapy.class_expression.restriction.OWLObjectHasValue method), 40
__repr__ () (owlapy.class_expression.restriction.OWLObjectOneOf method), 41
__repr__ () (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom method), 39
__repr__ () (owlapy.converter.OWLDataAllValuesFrom method), 73
__repr__ () (owlapy.converter.OWLDataCardinalityRestriction method), 72
__repr__ () (owlapy.converter.OWLDataHasValue method), 74
__repr__ () (owlapy.converter.OWLDataOneOf method), 75
__repr__ () (owlapy.converter.OWLDataSomeValuesFrom method), 73
__repr__ () (owlapy.converter.OWLDatatypeRestriction method), 66
__repr__ () (owlapy.converter.OWLObjectAllValuesFrom method), 70
__repr__ () (owlapy.converter.OWLObjectCardinalityRestriction method), 71
__repr__ () (owlapy.converter.OWLObjectComplementOf method), 69
__repr__ () (owlapy.converter.OWLObjectHasSelf method), 72
__repr__ () (owlapy.converter.OWLObjectHasValue method), 65
__repr__ () (owlapy.converter.OWLObjectOneOf method), 66
__repr__ () (owlapy.converter.OWLObjectSomeValuesFrom method), 70
__repr__ () (owlapy.iri.IRI method), 82
__repr__ () (owlapy.iri.Namespaces method), 82
__repr__ () (owlapy.namespaces.Namespaces method), 86
__repr__ () (owlapy.owl_annotation.OWLObject method), 87
__repr__ () (owlapy.owl_axiom.IRI method), 95
__repr__ () (owlapy.owl_axiom.OWLAnnotation method), 107
__repr__ () (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 108
__repr__ () (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 109
__repr__ () (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 109
__repr__ () (owlapy.owl_axiom.OWLClassAssertionAxiom method), 106
__repr__ () (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom method), 114
__repr__ () (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method), 100
__repr__ () (owlapy.owl_axiom.OWLDeclarationAxiom method), 100
__repr__ () (owlapy.owl_axiom.OWLDisjointUnionAxiom method), 106
__repr__ () (owlapy.owl_axiom.OWLHasKeyAxiom method), 101
__repr__ () (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom method), 105
__repr__ () (owlapy.owl_axiom.OWLNaryClassAxiom method), 102
__repr__ () (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 103
__repr__ () (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 104
__repr__ () (owlapy.owl_axiom.OWLObject method), 91
__repr__ () (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom method), 112
__repr__ () (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 111
__repr__ () (owlapy.owl_axiom.OWLPropertyDomainAxiom method), 114
__repr__ () (owlapy.owl_axiom.OWLPropertyRangeAxiom method), 115
__repr__ () (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 108
__repr__ () (owlapy.owl_axiom.OWLSubClassOfAxiom method), 106
__repr__ () (owlapy.owl_axiom.OWLSubPropertyAxiom method), 110
__repr__ () (owlapy.owl_data_ranges.OWLDataComplementOf method), 118

__repr__() (owlapy.owl_data_ranges.OWLNaryDataRange method), 117
__repr__() (owlapy.owl_data_ranges.OWLObject method), 117
__repr__() (owlapy.owl_datatype.IRI method), 119
__repr__() (owlapy.owl_individual.IRI method), 134
__repr__() (owlapy.owl_individual.OWLObject method), 134
__repr__() (owlapy.owl_object.OWLNamedObject method), 144
__repr__() (owlapy.owl_object.OWLObject method), 143
__repr__() (owlapy.owl_ontology_manager.IRI method), 176
__repr__() (owlapy.owl_ontology_manager.Ontology method), 197
__repr__() (owlapy.owl_ontology_manager.OWLAnnotationAssertionAxiom method), 184
__repr__() (owlapy.owl_ontology_manager.OWLClassAssertionAxiom method), 186
__repr__() (owlapy.owl_ontology_manager.OWLDataHasValue method), 179
__repr__() (owlapy.owl_ontology_manager.OWLDataPropertyCharacteristicAxiom method), 188
__repr__() (owlapy.owl_ontology_manager.OWLDeclarationAxiom method), 186
__repr__() (owlapy.owl_ontology_manager.OWLDisjointUnionAxiom method), 184
__repr__() (owlapy.owl_ontology_manager.OWLInverseObjectPropertiesAxiom method), 191
__repr__() (owlapy.owl_ontology_manager.OWLNaryBooleanClassExpression method), 180
__repr__() (owlapy.owl_ontology_manager.OWLNaryIndividualAxiom method), 192
__repr__() (owlapy.owl_ontology_manager.OWLNaryPropertyAxiom method), 191
__repr__() (owlapy.owl_ontology_manager.OWLObject method), 178
__repr__() (owlapy.owl_ontology_manager.OWLObjectComplementOf method), 181
__repr__() (owlapy.owl_ontology_manager.OWLObjectHasValue method), 182
__repr__() (owlapy.owl_ontology_manager.OWLObjectInverseOf method), 198
__repr__() (owlapy.owl_ontology_manager.OWLObjectOneOf method), 181
__repr__() (owlapy.owl_ontology_manager.OWLObjectPropertyCharacteristicAxiom method), 189
__repr__() (owlapy.owl_ontology_manager.OWLPropertyDomainAxiom method), 187
__repr__() (owlapy.owl_ontology_manager.OWLPropertyRangeAxiom method), 185
__repr__() (owlapy.owl_ontology_manager.OWLSubClassOfAxiom method), 183
__repr__() (owlapy.owl_ontology_manager.OWLSubPropertyAxiom method), 185
__repr__() (owlapy.owl_ontology.IRI method), 154
__repr__() (owlapy.owl_ontology.Ontology method), 173
__repr__() (owlapy.owl_ontology.OWLDataAllValuesFrom method), 161
__repr__() (owlapy.owl_ontology.OWLDataComplementOf method), 150
__repr__() (owlapy.owl_ontology.OWLDataHasValue method), 162
__repr__() (owlapy.owl_ontology.OWLDataOneOf method), 163
__repr__() (owlapy.owl_ontology.OWLDataSomeValuesFrom method), 160
__repr__() (owlapy.owl_ontology.OWLDatatypeRestriction method), 164
__repr__() (owlapy.owl_ontology.OWLFacetRestriction method), 165
__repr__() (owlapy.owl_ontology.OWLObject method), 154
__repr__() (owlapy.owl_ontology.OWLObjectAllValuesFrom method), 158
__repr__() (owlapy.owl_ontology.OWLObjectComplementOf method), 157
__repr__() (owlapy.owl_ontology.OWLObjectHasValue method), 160
__repr__() (owlapy.owl_ontology.OWLObjectInverseOf method), 167
__repr__() (owlapy.owl_ontology.OWLObjectOneOf method), 159
__repr__() (owlapy.owl_ontology.OWLObjectSomeValuesFrom method), 157
__repr__() (owlapy.owl_ontology.OWLOntologyID method), 169
__repr__() (owlapy.owl_ontology.OWLSubClassOfAxiom method), 149
__repr__() (owlapy.owl_property.IRI method), 204
__repr__() (owlapy.owl_property.OWLObject method), 203
__repr__() (owlapy.owl_property.OWLObjectInverseOf method), 207
__repr__() (owlapy.owl_reasoner.IRI method), 219
__repr__() (owlapy.owl_reasoner.Ontology method), 226
__repr__() (owlapy.owl_reasoner.OWLDataAllValuesFrom method), 217
__repr__() (owlapy.owl_reasoner.OWLDataComplementOf method), 221
__repr__() (owlapy.owl_reasoner.OWLDataHasValue method), 217
__repr__() (owlapy.owl_reasoner.OWLDataOneOf method), 216
__repr__() (owlapy.owl_reasoner.OWLDataSomeValuesFrom method), 215
__repr__() (owlapy.owl_reasoner.OWLDatatypeRestriction method), 216
__repr__() (owlapy.owl_reasoner.OWLFacetRestriction method), 217
__repr__() (owlapy.owl_reasoner.OWLObjectAllValuesFrom method), 212
__repr__() (owlapy.owl_reasoner.OWLObjectCardinalityRestriction method), 215
__repr__() (owlapy.owl_reasoner.OWLObjectComplementOf method), 212
__repr__() (owlapy.owl_reasoner.OWLObjectHasValue method), 213
__repr__() (owlapy.owl_reasoner.OWLObjectInverseOf method), 229
__repr__() (owlapy.owl_reasoner.OWLObjectOneOf method), 213
__repr__() (owlapy.owl_reasoner.OWLObjectSomeValuesFrom method), 211
__repr__() (owlapy.owl_reasoner.OWLSubClassOfAxiom method), 220
__repr__() (owlapy.owlapi_adaptor.IRI method), 255

__repr__() (owlapy.parser.IRI method), 260
__repr__() (owlapy.parser.Namespaces method), 265
__repr__() (owlapy.parser.OWLDataAllValuesFrom method), 278
__repr__() (owlapy.parser.OWLDataCardinalityRestriction method), 277
__repr__() (owlapy.parser.OWLDataComplementOf method), 279
__repr__() (owlapy.parser.OWLDataHasValue method), 275
__repr__() (owlapy.parser.OWLDataOneOf method), 276
__repr__() (owlapy.parser.OWLDataSomeValuesFrom method), 273
__repr__() (owlapy.parser.OWLDatatypeRestriction method), 276
__repr__() (owlapy.parser.OWLFacetRestriction method), 272
__repr__() (owlapy.parser.OWLObjectAllValuesFrom method), 277
__repr__() (owlapy.parser.OWLObjectCardinalityRestriction method), 276
__repr__() (owlapy.parser.OWLObjectComplementOf method), 271
__repr__() (owlapy.parser.OWLObjectHasSelf method), 267
__repr__() (owlapy.parser.OWLObjectHasValue method), 274
__repr__() (owlapy.parser.OWLObjectOneOf method), 270
__repr__() (owlapy.parser.OWLObjectSomeValuesFrom method), 268
__repr__() (owlapy.providers.OWLDatatypeRestriction method), 286
__repr__() (owlapy.providers.OWLFacetRestriction method), 287
__repr__() (owlapy.render.IRI method), 290
__repr__() (owlapy.render.OWLDataAllValuesFrom method), 301
__repr__() (owlapy.render.OWLDataComplementOf method), 305
__repr__() (owlapy.render.OWLDataHasValue method), 301
__repr__() (owlapy.render.OWLDataOneOf method), 303
__repr__() (owlapy.render.OWLDataSomeValuesFrom method), 300
__repr__() (owlapy.render.OWLDatatypeRestriction method), 307
__repr__() (owlapy.render.OWLFacetRestriction method), 306
__repr__() (owlapy.render.OWLNaryBooleanClassExpression method), 303
__repr__() (owlapy.render.OWLNaryDataRange method), 305
__repr__() (owlapy.render.OWLObject method), 294
__repr__() (owlapy.render.OWLObjectAllValuesFrom method), 298
__repr__() (owlapy.render.OWLObjectComplementOf method), 299
__repr__() (owlapy.render.OWLObjectHasSelf method), 300
__repr__() (owlapy.render.OWLObjectHasValue method), 306
__repr__() (owlapy.render.OWLObjectInverseOf method), 295
__repr__() (owlapy.render.OWLObjectOneOf method), 307
__repr__() (owlapy.render.OWLObjectSomeValuesFrom method), 297
__repr__() (owlapy.utils.OWLDataAllValuesFrom method), 323
__repr__() (owlapy.utils.OWLDataCardinalityRestriction method), 325
__repr__() (owlapy.utils.OWLDataComplementOf method), 328
__repr__() (owlapy.utils.OWLDataHasValue method), 323
__repr__() (owlapy.utils.OWLDataOneOf method), 325
__repr__() (owlapy.utils.OWLDataSomeValuesFrom method), 324
__repr__() (owlapy.utils.OWLDatatypeRestriction method), 327
__repr__() (owlapy.utils.OWLFacetRestriction method), 327
__repr__() (owlapy.utils.OWLNaryBooleanClassExpression method), 326
__repr__() (owlapy.utils.OWLNaryDataRange method), 329
__repr__() (owlapy.utils.OWLObject method), 330
__repr__() (owlapy.utils.OWLObjectAllValuesFrom method), 324
__repr__() (owlapy.utils.OWLObjectCardinalityRestriction method), 319
__repr__() (owlapy.utils.OWLObjectComplementOf method), 319
__repr__() (owlapy.utils.OWLObjectHasSelf method), 322
__repr__() (owlapy.utils.OWLObjectHasValue method), 327
__repr__() (owlapy.utils.OWLObjectInverseOf method), 316
__repr__() (owlapy.utils.OWLObjectOneOf method), 328
__repr__() (owlapy.utils.OWLObjectSomeValuesFrom method), 320
__repr__() (owlapy.vocab.IRI method), 336
__repr__() (owlapy.vocab.Namespaces method), 337
__setitem__() (owlapy.owl_reasoner.LRUCache method), 233
__setitem__() (owlapy.utils.LRUCache method), 335
__slots__ (owlapy.class_expression.class_expression.HasOperands attribute), 18
__slots__ (owlapy.class_expression.class_expression.OWLBooleanClassExpression attribute), 20
__slots__ (owlapy.class_expression.class_expression.OWLClassExpression attribute), 19
__slots__ (owlapy.class_expression.class_expression.OWLObjectComplementOf attribute), 20
__slots__ (owlapy.class_expression.nary_boolean_expression.HasOperands attribute), 22
__slots__ (owlapy.class_expression.nary_boolean_expression.OWLBooleanClassExpression attribute), 22
__slots__ (owlapy.class_expression.nary_boolean_expression.OWLClassExpression attribute), 21
__slots__ (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression attribute), 22

__slots__ (owlapy.class_expression.nary_boolean_expression.OwlObjectIntersectionOf attribute), 23
 __slots__ (owlapy.class_expression.nary_boolean_expression.OwlObjectUnionOf attribute), 22
 __slots__ (owlapy.class_expression.owl_class.IRI attribute), 25
 __slots__ (owlapy.class_expression.owl_class.OwlClass attribute), 26
 __slots__ (owlapy.class_expression.owl_class.OwlClassExpression attribute), 23
 __slots__ (owlapy.class_expression.owl_class.OwlEntity attribute), 24
 __slots__ (owlapy.class_expression.owl_class.OwlObjectComplementOf attribute), 24
 __slots__ (owlapy.class_expression.OwlBooleanClassExpression attribute), 49
 __slots__ (owlapy.class_expression.OwlCardinalityRestriction attribute), 53
 __slots__ (owlapy.class_expression.OwlClass attribute), 49
 __slots__ (owlapy.class_expression.OwlClassExpression attribute), 48
 __slots__ (owlapy.class_expression.OwlDataAllValuesFrom attribute), 60
 __slots__ (owlapy.class_expression.OwlDataCardinalityRestriction attribute), 55
 __slots__ (owlapy.class_expression.OwlDataExactCardinality attribute), 61
 __slots__ (owlapy.class_expression.OwlDataHasValue attribute), 60
 __slots__ (owlapy.class_expression.OwlDataMaxCardinality attribute), 61
 __slots__ (owlapy.class_expression.OwlDataMinCardinality attribute), 60
 __slots__ (owlapy.class_expression.OwlDataRestriction attribute), 53
 __slots__ (owlapy.class_expression.OwlDataSomeValuesFrom attribute), 59
 __slots__ (owlapy.class_expression.OwlDatatypeRestriction attribute), 57
 __slots__ (owlapy.class_expression.OwlFacetRestriction attribute), 58
 __slots__ (owlapy.class_expression.OwlHasValueRestriction attribute), 52
 __slots__ (owlapy.class_expression.OwlNaryBooleanClassExpression attribute), 50
 __slots__ (owlapy.class_expression.OwlObjectAllValuesFrom attribute), 56
 __slots__ (owlapy.class_expression.OwlObjectCardinalityRestriction attribute), 53
 __slots__ (owlapy.class_expression.OwlObjectComplementOf attribute), 49
 __slots__ (owlapy.class_expression.OwlObjectExactCardinality attribute), 59
 __slots__ (owlapy.class_expression.OwlObjectHasSelf attribute), 54
 __slots__ (owlapy.class_expression.OwlObjectHasValue attribute), 56
 __slots__ (owlapy.class_expression.OwlObjectIntersectionOf attribute), 51
 __slots__ (owlapy.class_expression.OwlObjectMaxCardinality attribute), 58
 __slots__ (owlapy.class_expression.OwlObjectMinCardinality attribute), 58
 __slots__ (owlapy.class_expression.OwlObjectOneOf attribute), 61
 __slots__ (owlapy.class_expression.OwlObjectRestriction attribute), 52
 __slots__ (owlapy.class_expression.OwlObjectSomeValuesFrom attribute), 55
 __slots__ (owlapy.class_expression.OwlObjectUnionOf attribute), 51
 __slots__ (owlapy.class_expression.OwlQuantifiedDataRestriction attribute), 55
 __slots__ (owlapy.class_expression.OwlQuantifiedObjectRestriction attribute), 52
 __slots__ (owlapy.class_expression.OwlQuantifiedRestriction attribute), 51
 __slots__ (owlapy.class_expression.OwlRestriction attribute), 51
 __slots__ (owlapy.class_expression.restriction.HasCardinality attribute), 29
 __slots__ (owlapy.class_expression.restriction.HasFiller attribute), 29
 __slots__ (owlapy.class_expression.restriction.HasOperands attribute), 29
 __slots__ (owlapy.class_expression.restriction.OwlCardinalityRestriction attribute), 37
 __slots__ (owlapy.class_expression.restriction.OwlClassExpression attribute), 30
 __slots__ (owlapy.class_expression.restriction.OwlDataAllValuesFrom attribute), 43
 __slots__ (owlapy.class_expression.restriction.OwlDataCardinalityRestriction attribute), 42
 __slots__ (owlapy.class_expression.restriction.OwlDataExactCardinality attribute), 43
 __slots__ (owlapy.class_expression.restriction.OwlDataHasValue attribute), 44
 __slots__ (owlapy.class_expression.restriction.OwlDataMaxCardinality attribute), 42
 __slots__ (owlapy.class_expression.restriction.OwlDataMinCardinality attribute), 42
 __slots__ (owlapy.class_expression.restriction.OwlDataPropertyExpression attribute), 32
 __slots__ (owlapy.class_expression.restriction.OwlDataRestriction attribute), 41
 __slots__ (owlapy.class_expression.restriction.OwlDataSomeValuesFrom attribute), 43
 __slots__ (owlapy.class_expression.restriction.OwlDatatype attribute), 34
 __slots__ (owlapy.class_expression.restriction.OwlDatatypeRestriction attribute), 45
 __slots__ (owlapy.class_expression.restriction.OwlFacetRestriction attribute), 45
 __slots__ (owlapy.class_expression.restriction.OwlHasValueRestriction attribute), 36
 __slots__ (owlapy.class_expression.restriction.OwlIndividual attribute), 34
 __slots__ (owlapy.class_expression.restriction.OwlLiteral attribute), 32
 __slots__ (owlapy.class_expression.restriction.OwlObject attribute), 34
 __slots__ (owlapy.class_expression.restriction.OwlObjectAllValuesFrom attribute), 39
 __slots__ (owlapy.class_expression.restriction.OwlObjectCardinalityRestriction attribute), 37
 __slots__ (owlapy.class_expression.restriction.OwlObjectExactCardinality attribute), 38
 __slots__ (owlapy.class_expression.restriction.OwlObjectHasSelf attribute), 39
 __slots__ (owlapy.class_expression.restriction.OwlObjectHasValue attribute), 40
 __slots__ (owlapy.class_expression.restriction.OwlObjectIntersectionOf attribute), 29
 __slots__ (owlapy.class_expression.restriction.OwlObjectMaxCardinality attribute), 38

- __slots__ (owlapy.class_expression.restriction.OWLObjectMinCardinality attribute), 38
- __slots__ (owlapy.class_expression.restriction.OWLObjectOneOf attribute), 40
- __slots__ (owlapy.class_expression.restriction.OWLObjectPropertyExpression attribute), 31
- __slots__ (owlapy.class_expression.restriction.OWLObjectRestriction attribute), 36
- __slots__ (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom attribute), 39
- __slots__ (owlapy.class_expression.restriction.OWLObjectUnionOf attribute), 30
- __slots__ (owlapy.class_expression.restriction.OWLPropertyExpression attribute), 31
- __slots__ (owlapy.class_expression.restriction.OWLQuantifiedDataRestriction attribute), 41
- __slots__ (owlapy.class_expression.restriction.OWLQuantifiedObjectRestriction attribute), 37
- __slots__ (owlapy.class_expression.restriction.OWLQuantifiedRestriction attribute), 36
- __slots__ (owlapy.class_expression.restriction.OWLRestriction attribute), 35
- __slots__ (owlapy.converter.Owl2SparqlConverter attribute), 80
- __slots__ (owlapy.converter.OWLClass attribute), 67
- __slots__ (owlapy.converter.OWLClassExpression attribute), 68
- __slots__ (owlapy.converter.OWLDataAllValuesFrom attribute), 73
- __slots__ (owlapy.converter.OWLDataCardinalityRestriction attribute), 72
- __slots__ (owlapy.converter.OWLDataExactCardinality attribute), 67
- __slots__ (owlapy.converter.OWLDataHasValue attribute), 74
- __slots__ (owlapy.converter.OWLDataMaxCardinality attribute), 67
- __slots__ (owlapy.converter.OWLDataMinCardinality attribute), 66
- __slots__ (owlapy.converter.OWLDataProperty attribute), 77
- __slots__ (owlapy.converter.OWLDataSomeValuesFrom attribute), 73
- __slots__ (owlapy.converter.OWLDatatype attribute), 78
- __slots__ (owlapy.converter.OWLDatatypeRestriction attribute), 66
- __slots__ (owlapy.converter.OWLEntity attribute), 78
- __slots__ (owlapy.converter.OWLLiteral attribute), 75
- __slots__ (owlapy.converter.OWLNamedIndividual attribute), 75
- __slots__ (owlapy.converter.OWLObjectAllValuesFrom attribute), 70
- __slots__ (owlapy.converter.OWLObjectCardinalityRestriction attribute), 70
- __slots__ (owlapy.converter.OWLObjectComplementOf attribute), 69
- __slots__ (owlapy.converter.OWLObjectExactCardinality attribute), 71
- __slots__ (owlapy.converter.OWLObjectHasSelf attribute), 72
- __slots__ (owlapy.converter.OWLObjectHasValue attribute), 65
- __slots__ (owlapy.converter.OWLObjectIntersectionOf attribute), 69
- __slots__ (owlapy.converter.OWLObjectMaxCardinality attribute), 71
- __slots__ (owlapy.converter.OWLObjectMinCardinality attribute), 71
- __slots__ (owlapy.converter.OWLObjectOneOf attribute), 65
- __slots__ (owlapy.converter.OWLObjectProperty attribute), 77
- __slots__ (owlapy.converter.OWLObjectSomeValuesFrom attribute), 70
- __slots__ (owlapy.converter.OWLObjectUnionOf attribute), 69
- __slots__ (owlapy.converter.VariablesMapping attribute), 79
- __slots__ (owlapy.iri.IRI attribute), 82
- __slots__ (owlapy.iri.Namespaces attribute), 82
- __slots__ (owlapy.iri.OWLAnnotationSubject attribute), 82
- __slots__ (owlapy.iri.OWLAnnotationValue attribute), 82
- __slots__ (owlapy.meta_classes.HasCardinality attribute), 85
- __slots__ (owlapy.meta_classes.HasFiller attribute), 85
- __slots__ (owlapy.meta_classes.HasIRI attribute), 84
- __slots__ (owlapy.meta_classes.HasOperands attribute), 84
- __slots__ (owlapy.namespaces.Namespaces attribute), 86
- __slots__ (owlapy.owl_annotation.OWLAnnotationObject attribute), 87
- __slots__ (owlapy.owl_annotation.OWLAnnotationSubject attribute), 87
- __slots__ (owlapy.owl_annotation.OWLAnnotationValue attribute), 87
- __slots__ (owlapy.owl_annotation.OWLObject attribute), 86
- __slots__ (owlapy.owl_axiom.HasOperands attribute), 92
- __slots__ (owlapy.owl_axiom.IRI attribute), 95
- __slots__ (owlapy.owl_axiom.OWLAnnotation attribute), 107
- __slots__ (owlapy.owl_axiom.OWLAnnotationAssertionAxiom attribute), 108
- __slots__ (owlapy.owl_axiom.OWLAnnotationAxiom attribute), 107
- __slots__ (owlapy.owl_axiom.OWLAnnotationProperty attribute), 107
- __slots__ (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom attribute), 109
- __slots__ (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom attribute), 109
- __slots__ (owlapy.owl_axiom.OWLAnnotationSubject attribute), 96
- __slots__ (owlapy.owl_axiom.OWLAnnotationValue attribute), 97
- __slots__ (owlapy.owl_axiom.OWLAsymmetricObjectPropertyAxiom attribute), 112
- __slots__ (owlapy.owl_axiom.OWLClassAssertionAxiom attribute), 106
- __slots__ (owlapy.owl_axiom.OWLClass attribute), 94
- __slots__ (owlapy.owl_axiom.OWLClassAssertionAxiom attribute), 106

__slots__ (owlapy.owl_axiom.OWLClassAxiom attribute), 100
 __slots__ (owlapy.owl_axiom.OWLClassExpression attribute), 93
 __slots__ (owlapy.owl_axiom.OWLDataPropertyAssertionAxiom attribute), 111
 __slots__ (owlapy.owl_axiom.OWLDataPropertyAxiom attribute), 99
 __slots__ (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom attribute), 114
 __slots__ (owlapy.owl_axiom.OWLDataPropertyDomainAxiom attribute), 115
 __slots__ (owlapy.owl_axiom.OWLDataPropertyExpression attribute), 91
 __slots__ (owlapy.owl_axiom.OWLDataPropertyRangeAxiom attribute), 116
 __slots__ (owlapy.owl_axiom.OWLDatatype attribute), 92
 __slots__ (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom attribute), 100
 __slots__ (owlapy.owl_axiom.OWLDeclarationAxiom attribute), 100
 __slots__ (owlapy.owl_axiom.OWLDifferentIndividualsAxiom attribute), 103
 __slots__ (owlapy.owl_axiom.OWLDisjointClassesAxiom attribute), 102
 __slots__ (owlapy.owl_axiom.OWLDisjointDataPropertiesAxiom attribute), 105
 __slots__ (owlapy.owl_axiom.OWLDisjointObjectPropertiesAxiom attribute), 104
 __slots__ (owlapy.owl_axiom.OWLDisjointUnionAxiom attribute), 106
 __slots__ (owlapy.owl_axiom.OWLEntity attribute), 91
 __slots__ (owlapy.owl_axiom.OWLEquivalentClassesAxiom attribute), 102
 __slots__ (owlapy.owl_axiom.OWLEquivalentDataPropertiesAxiom attribute), 105
 __slots__ (owlapy.owl_axiom.OWLEquivalentObjectPropertiesAxiom attribute), 104
 __slots__ (owlapy.owl_axiom.OWLFunctionalDataPropertyAxiom attribute), 114
 __slots__ (owlapy.owl_axiom.OWLFunctionalObjectPropertyAxiom attribute), 112
 __slots__ (owlapy.owl_axiom.OWLHasKeyAxiom attribute), 101
 __slots__ (owlapy.owl_axiom.OWLIndividual attribute), 95
 __slots__ (owlapy.owl_axiom.OWLIndividualAxiom attribute), 100
 __slots__ (owlapy.owl_axiom.OWLInverseFunctionalObjectPropertyAxiom attribute), 113
 __slots__ (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom attribute), 104
 __slots__ (owlapy.owl_axiom.OWLIrreflexiveObjectPropertyAxiom attribute), 113
 __slots__ (owlapy.owl_axiom.OWLLiteral attribute), 97
 __slots__ (owlapy.owl_axiom.OWLLogicalAxiom attribute), 99
 __slots__ (owlapy.owl_axiom.OWLNaryAxiom attribute), 101
 __slots__ (owlapy.owl_axiom.OWLNaryClassAxiom attribute), 102
 __slots__ (owlapy.owl_axiom.OWLNaryIndividualAxiom attribute), 103
 __slots__ (owlapy.owl_axiom.OWLNaryPropertyAxiom attribute), 103
 __slots__ (owlapy.owl_axiom.OWLNegativeDataPropertyAssertionAxiom attribute), 112
 __slots__ (owlapy.owl_axiom.OWLNegativeObjectPropertyAssertionAxiom attribute), 111
 __slots__ (owlapy.owl_axiom.OWLObject attribute), 91
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyAssertionAxiom attribute), 111
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyAxiom attribute), 99
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom attribute), 112
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyDomainAxiom attribute), 115
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyExpression attribute), 91
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyRangeAxiom attribute), 116
 __slots__ (owlapy.owl_axiom.OWLObjectUnionOf attribute), 95
 __slots__ (owlapy.owl_axiom.OWLProperty attribute), 93
 __slots__ (owlapy.owl_axiom.OWLPropertyAssertionAxiom attribute), 110
 __slots__ (owlapy.owl_axiom.OWLPropertyAxiom attribute), 99
 __slots__ (owlapy.owl_axiom.OWLPropertyDomainAxiom attribute), 114
 __slots__ (owlapy.owl_axiom.OWLPropertyExpression attribute), 92
 __slots__ (owlapy.owl_axiom.OWLPropertyRangeAxiom attribute), 115
 __slots__ (owlapy.owl_axiom.OWLReflexiveObjectPropertyAxiom attribute), 113
 __slots__ (owlapy.owl_axiom.OWLSameIndividualAxiom attribute), 103
 __slots__ (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom attribute), 108
 __slots__ (owlapy.owl_axiom.OWLSubClassOfAxiom attribute), 105
 __slots__ (owlapy.owl_axiom.OWLSubDataPropertyOfAxiom attribute), 110
 __slots__ (owlapy.owl_axiom.OWLSubObjectPropertyOfAxiom attribute), 110
 __slots__ (owlapy.owl_axiom.OWLSubPropertyAxiom attribute), 109
 __slots__ (owlapy.owl_axiom.OWLSymmetricObjectPropertyAxiom attribute), 113
 __slots__ (owlapy.owl_axiom.OWLTransitiveObjectPropertyAxiom attribute), 114
 __slots__ (owlapy.owl_axiom.OWLUnaryPropertyAxiom attribute), 112
 __slots__ (owlapy.owl_data_ranges.HasOperands attribute), 117
 __slots__ (owlapy.owl_data_ranges.OWLDataIntersectionOf attribute), 118
 __slots__ (owlapy.owl_data_ranges.OWLDataUnionOf attribute), 118
 __slots__ (owlapy.owl_data_ranges.OWLNaryDataRange attribute), 117
 __slots__ (owlapy.owl_data_ranges.OWLObject attribute), 116
 __slots__ (owlapy.owl_datatype.HasIRI attribute), 120
 __slots__ (owlapy.owl_datatype.IRI attribute), 119
 __slots__ (owlapy.owl_datatype.OWLDatatype attribute), 121

- __slots__ (owlapy.owl_datatype.OWLEntity attribute), 119
- __slots__ (owlapy.owl_hierarchy.AbstractHierarchy attribute), 130
- __slots__ (owlapy.owl_hierarchy.HasIRI attribute), 123
- __slots__ (owlapy.owl_hierarchy.OWLClass attribute), 122
- __slots__ (owlapy.owl_hierarchy.OWLDataProperty attribute), 124
- __slots__ (owlapy.owl_hierarchy.OWLObjectProperty attribute), 123
- __slots__ (owlapy.owl_hierarchy.OWLReasoner attribute), 124
- __slots__ (owlapy.owl_individual.IRI attribute), 134
- __slots__ (owlapy.owl_individual.OWLEntity attribute), 134
- __slots__ (owlapy.owl_individual.OWLIndividual attribute), 135
- __slots__ (owlapy.owl_individual.OWLNamedIndividual attribute), 136
- __slots__ (owlapy.owl_individual.OWLObject attribute), 134
- __slots__ (owlapy.owl_literal.OWLAnnotationValue attribute), 138
- __slots__ (owlapy.owl_literal.OWLDataProperty attribute), 140
- __slots__ (owlapy.owl_literal.OWLDatatype attribute), 138
- __slots__ (owlapy.owl_literal.OWLLiteral attribute), 140
- __slots__ (owlapy.owl_literal.OWLObjectProperty attribute), 139
- __slots__ (owlapy.owl_object.HasIRI attribute), 143
- __slots__ (owlapy.owl_object.OWLEntity attribute), 144
- __slots__ (owlapy.owl_object.OWLNamedObject attribute), 144
- __slots__ (owlapy.owl_object.OWLObject attribute), 143
- __slots__ (owlapy.owl_ontology_manager.AddImport attribute), 201
- __slots__ (owlapy.owl_ontology_manager.HasIRI attribute), 177
- __slots__ (owlapy.owl_ontology_manager.IRI attribute), 176
- __slots__ (owlapy.owl_ontology_manager.Ontology attribute), 195
- __slots__ (owlapy.owl_ontology_manager.OntologyManager attribute), 201
- __slots__ (owlapy.owl_ontology_manager.OWLAnnotationAssertionAxiom attribute), 184
- __slots__ (owlapy.owl_ontology_manager.OWLAnnotationProperty attribute), 185
- __slots__ (owlapy.owl_ontology_manager.OWLAsymmetricObjectPropertyAxiom attribute), 187
- __slots__ (owlapy.owl_ontology_manager.OWLAxiom attribute), 182
- __slots__ (owlapy.owl_ontology_manager.OWLClass attribute), 178
- __slots__ (owlapy.owl_ontology_manager.OWLClassAssertionAxiom attribute), 186
- __slots__ (owlapy.owl_ontology_manager.OWLDataHasValue attribute), 179
- __slots__ (owlapy.owl_ontology_manager.OWLDataProperty attribute), 197
- __slots__ (owlapy.owl_ontology_manager.OWLDataPropertyAssertionAxiom attribute), 188
- __slots__ (owlapy.owl_ontology_manager.OWLDataPropertyCharacteristicAxiom attribute), 188
- __slots__ (owlapy.owl_ontology_manager.OWLDeclarationAxiom attribute), 186
- __slots__ (owlapy.owl_ontology_manager.OWLDifferentIndividualsAxiom attribute), 192
- __slots__ (owlapy.owl_ontology_manager.OWLDisjointClassesAxiom attribute), 192
- __slots__ (owlapy.owl_ontology_manager.OWLDisjointDataPropertiesAxiom attribute), 190
- __slots__ (owlapy.owl_ontology_manager.OWLDisjointObjectPropertiesAxiom attribute), 190
- __slots__ (owlapy.owl_ontology_manager.OWLDisjointUnionAxiom attribute), 184
- __slots__ (owlapy.owl_ontology_manager.OWLEquivalentClassesAxiom attribute), 183
- __slots__ (owlapy.owl_ontology_manager.OWLEquivalentDataPropertiesAxiom attribute), 190
- __slots__ (owlapy.owl_ontology_manager.OWLEquivalentObjectPropertiesAxiom attribute), 190
- __slots__ (owlapy.owl_ontology_manager.OWLFunctionalDataPropertyAxiom attribute), 188
- __slots__ (owlapy.owl_ontology_manager.OWLFunctionalObjectPropertyAxiom attribute), 189
- __slots__ (owlapy.owl_ontology_manager.OWLImportsDeclaration attribute), 200
- __slots__ (owlapy.owl_ontology_manager.OWLIndividual attribute), 193
- __slots__ (owlapy.owl_ontology_manager.OWLInverseFunctionalObjectPropertyAxiom attribute), 189
- __slots__ (owlapy.owl_ontology_manager.OWLInverseObjectPropertiesAxiom attribute), 191
- __slots__ (owlapy.owl_ontology_manager.OWLIrreflexiveObjectPropertyAxiom attribute), 189
- __slots__ (owlapy.owl_ontology_manager.OWLNamedIndividual attribute), 192
- __slots__ (owlapy.owl_ontology_manager.OWLNaryBooleanClassExpression attribute), 180
- __slots__ (owlapy.owl_ontology_manager.OWLNaryIndividualAxiom attribute), 191
- __slots__ (owlapy.owl_ontology_manager.OWLNaryPropertyAxiom attribute), 191
- __slots__ (owlapy.owl_ontology_manager.OWLObject attribute), 178
- __slots__ (owlapy.owl_ontology_manager.OWLObjectComplementOf attribute), 181
- __slots__ (owlapy.owl_ontology_manager.OWLObjectHasValue attribute), 181
- __slots__ (owlapy.owl_ontology_manager.OWLObjectInverseOf attribute), 198
- __slots__ (owlapy.owl_ontology_manager.OWLObjectOneOf attribute), 180
- __slots__ (owlapy.owl_ontology_manager.OWLObjectProperty attribute), 198
- __slots__ (owlapy.owl_ontology_manager.OWLObjectPropertyAssertionAxiom attribute), 186
- __slots__ (owlapy.owl_ontology_manager.OWLObjectPropertyCharacteristicAxiom attribute), 189
- __slots__ (owlapy.owl_ontology_manager.OWLObjectPropertyRangeAxiom attribute), 182
- __slots__ (owlapy.owl_ontology_manager.OWLOntology attribute), 193
- __slots__ (owlapy.owl_ontology_manager.OWLOntologyChange attribute), 199
- __slots__ (owlapy.owl_ontology_manager.OWLProperty attribute), 199

__slots__ (owlapy.owl_ontology_manager.OWLPropertyDomainAxiom attribute), 187
 __slots__ (owlapy.owl_ontology_manager.OWLPropertyRangeAxiom attribute), 185
 __slots__ (owlapy.owl_ontology_manager.OWLQuantifiedDataRestriction attribute), 179
 __slots__ (owlapy.owl_ontology_manager.OWLQuantifiedObjectRestriction attribute), 182
 __slots__ (owlapy.owl_ontology_manager.OWLReflexiveObjectPropertyAxiom attribute), 188
 __slots__ (owlapy.owl_ontology_manager.OWLSameIndividualAxiom attribute), 192
 __slots__ (owlapy.owl_ontology_manager.OWLSubClassOfAxiom attribute), 183
 __slots__ (owlapy.owl_ontology_manager.OWLSubPropertyAxiom attribute), 185
 __slots__ (owlapy.owl_ontology_manager.OWLSymmetricObjectPropertyAxiom attribute), 187
 __slots__ (owlapy.owl_ontology_manager.OWLTransitiveObjectPropertyAxiom attribute), 187
 __slots__ (owlapy.owl_ontology_manager.ToOwlready2 attribute), 197
 __slots__ (owlapy.owl_ontology.FromOwlready2 attribute), 173
 __slots__ (owlapy.owl_ontology.IRI attribute), 154
 __slots__ (owlapy.owl_ontology.Ontology attribute), 171
 __slots__ (owlapy.owl_ontology.OWLAnnotationProperty attribute), 149
 __slots__ (owlapy.owl_ontology.OWLClass attribute), 155
 __slots__ (owlapy.owl_ontology.OWLClassAxiom attribute), 148
 __slots__ (owlapy.owl_ontology.OWLClassExpression attribute), 156
 __slots__ (owlapy.owl_ontology.OWLDataAllValuesFrom attribute), 161
 __slots__ (owlapy.owl_ontology.OWLDataExactCardinality attribute), 161
 __slots__ (owlapy.owl_ontology.OWLDataHasValue attribute), 162
 __slots__ (owlapy.owl_ontology.OWLDataIntersectionOf attribute), 150
 __slots__ (owlapy.owl_ontology.OWLDataMaxCardinality attribute), 162
 __slots__ (owlapy.owl_ontology.OWLDataMinCardinality attribute), 162
 __slots__ (owlapy.owl_ontology.OWLDataProperty attribute), 165
 __slots__ (owlapy.owl_ontology.OWLDataPropertyDomainAxiom attribute), 148
 __slots__ (owlapy.owl_ontology.OWLDataPropertyExpression attribute), 168
 __slots__ (owlapy.owl_ontology.OWLDataPropertyRangeAxiom attribute), 148
 __slots__ (owlapy.owl_ontology.OWLDataRestriction attribute), 164
 __slots__ (owlapy.owl_ontology.OWLDataSomeValuesFrom attribute), 160
 __slots__ (owlapy.owl_ontology.OWLDatatype attribute), 151
 __slots__ (owlapy.owl_ontology.OWLDatatypeRestriction attribute), 163
 __slots__ (owlapy.owl_ontology.OWLDataUnionOf attribute), 150
 __slots__ (owlapy.owl_ontology.OWLEquivalentClassesAxiom attribute), 147
 __slots__ (owlapy.owl_ontology.OWLFacetRestriction attribute), 165
 __slots__ (owlapy.owl_ontology.OWLIndividual attribute), 151
 __slots__ (owlapy.owl_ontology.OWLLiteral attribute), 152
 __slots__ (owlapy.owl_ontology.OWLNamedIndividual attribute), 151
 __slots__ (owlapy.owl_ontology.OWLObject attribute), 153
 __slots__ (owlapy.owl_ontology.OWLObjectAllValuesFrom attribute), 158
 __slots__ (owlapy.owl_ontology.OWLObjectComplementOf attribute), 156
 __slots__ (owlapy.owl_ontology.OWLObjectExactCardinality attribute), 159
 __slots__ (owlapy.owl_ontology.OWLObjectHasValue attribute), 160
 __slots__ (owlapy.owl_ontology.OWLObjectIntersectionOf attribute), 157
 __slots__ (owlapy.owl_ontology.OWLObjectInverseOf attribute), 167
 __slots__ (owlapy.owl_ontology.OWLObjectMaxCardinality attribute), 159
 __slots__ (owlapy.owl_ontology.OWLObjectMinCardinality attribute), 160
 __slots__ (owlapy.owl_ontology.OWLObjectOneOf attribute), 158
 __slots__ (owlapy.owl_ontology.OWLObjectProperty attribute), 165
 __slots__ (owlapy.owl_ontology.OWLObjectPropertyDomainAxiom attribute), 148
 __slots__ (owlapy.owl_ontology.OWLObjectPropertyExpression attribute), 167
 __slots__ (owlapy.owl_ontology.OWLObjectPropertyRangeAxiom attribute), 149
 __slots__ (owlapy.owl_ontology.OWLObjectRestriction attribute), 164
 __slots__ (owlapy.owl_ontology.OWLObjectSomeValuesFrom attribute), 157
 __slots__ (owlapy.owl_ontology.OWLObjectUnionOf attribute), 157
 __slots__ (owlapy.owl_ontology.OWLOntology attribute), 169
 __slots__ (owlapy.owl_ontology.OWLOntologyID attribute), 168
 __slots__ (owlapy.owl_ontology.OWLPropertyExpression attribute), 166
 __slots__ (owlapy.owl_ontology.OWLRestriction attribute), 164
 __slots__ (owlapy.owl_ontology.OWLSubClassOfAxiom attribute), 149
 __slots__ (owlapy.owl_ontology.ToOwlready2 attribute), 173
 __slots__ (owlapy.owl_property.IRI attribute), 203
 __slots__ (owlapy.owl_property.OWLDataProperty attribute), 207
 __slots__ (owlapy.owl_property.OWLDataPropertyExpression attribute), 206
 __slots__ (owlapy.owl_property.OWLEntity attribute), 203
 __slots__ (owlapy.owl_property.OWLObject attribute), 203
 __slots__ (owlapy.owl_property.OWLObjectInverseOf attribute), 207
 __slots__ (owlapy.owl_property.OWLObjectProperty attribute), 206

- __slots__ (owlapy.owl_property.OWLObjectPropertyExpression attribute), 205
- __slots__ (owlapy.owl_property.OWLProperty attribute), 206
- __slots__ (owlapy.owl_property.OWLPropertyExpression attribute), 205
- __slots__ (owlapy.owl_reasoner.FastInstanceCheckerReasoner attribute), 247
- __slots__ (owlapy.owl_reasoner.IRI attribute), 219
- __slots__ (owlapy.owl_reasoner.Ontology attribute), 224
- __slots__ (owlapy.owl_reasoner.OntologyManager attribute), 226
- __slots__ (owlapy.owl_reasoner.OntologyReasoner attribute), 240
- __slots__ (owlapy.owl_reasoner.OWLAxiom attribute), 220
- __slots__ (owlapy.owl_reasoner.OWLClass attribute), 218
- __slots__ (owlapy.owl_reasoner.OWLClassExpression attribute), 210
- __slots__ (owlapy.owl_reasoner.OWLDataAllValuesFrom attribute), 217
- __slots__ (owlapy.owl_reasoner.OWLDataHasValue attribute), 217
- __slots__ (owlapy.owl_reasoner.OWLDataIntersectionOf attribute), 221
- __slots__ (owlapy.owl_reasoner.OWLDataProperty attribute), 228
- __slots__ (owlapy.owl_reasoner.OWLDataPropertyExpression attribute), 230
- __slots__ (owlapy.owl_reasoner.OWLDataSomeValuesFrom attribute), 215
- __slots__ (owlapy.owl_reasoner.OWLDatatype attribute), 222
- __slots__ (owlapy.owl_reasoner.OWLDatatypeRestriction attribute), 216
- __slots__ (owlapy.owl_reasoner.OWLDataUnionOf attribute), 221
- __slots__ (owlapy.owl_reasoner.OWLFacetRestriction attribute), 216
- __slots__ (owlapy.owl_reasoner.OWLLiteral attribute), 230
- __slots__ (owlapy.owl_reasoner.OWLNamedIndividual attribute), 230
- __slots__ (owlapy.owl_reasoner.OWLObjectAllValuesFrom attribute), 212
- __slots__ (owlapy.owl_reasoner.OWLObjectCardinalityRestriction attribute), 214
- __slots__ (owlapy.owl_reasoner.OWLObjectComplementOf attribute), 211
- __slots__ (owlapy.owl_reasoner.OWLObjectExactCardinality attribute), 214
- __slots__ (owlapy.owl_reasoner.OWLObjectHasValue attribute), 213
- __slots__ (owlapy.owl_reasoner.OWLObjectIntersectionOf attribute), 211
- __slots__ (owlapy.owl_reasoner.OWLObjectInverseOf attribute), 229
- __slots__ (owlapy.owl_reasoner.OWLObjectMaxCardinality attribute), 214
- __slots__ (owlapy.owl_reasoner.OWLObjectMinCardinality attribute), 214
- __slots__ (owlapy.owl_reasoner.OWLObjectOneOf attribute), 212
- __slots__ (owlapy.owl_reasoner.OWLObjectProperty attribute), 228
- __slots__ (owlapy.owl_reasoner.OWLObjectPropertyExpression attribute), 227
- __slots__ (owlapy.owl_reasoner.OWLObjectSomeValuesFrom attribute), 211
- __slots__ (owlapy.owl_reasoner.OWLObjectUnionOf attribute), 211
- __slots__ (owlapy.owl_reasoner.OWLOntology attribute), 222
- __slots__ (owlapy.owl_reasoner.OWLPropertyExpression attribute), 229
- __slots__ (owlapy.owl_reasoner.OWLReasoner attribute), 233
- __slots__ (owlapy.owl_reasoner.OWLSubClassOfAxiom attribute), 220
- __slots__ (owlapy.owl_reasoner.SyncReasoner attribute), 253
- __slots__ (owlapy.owl_reasoner.ToOwlready2 attribute), 226
- __slots__ (owlapy.owlapi_adaptor.IRI attribute), 255
- __slots__ (owlapy.owlapi_adaptor.OWLClassExpression attribute), 254
- __slots__ (owlapy.owlapi_adaptor.OWLNamedIndividual attribute), 256
- __slots__ (owlapy.parser.IRI attribute), 260
- __slots__ (owlapy.parser.Namespaces attribute), 265
- __slots__ (owlapy.parser.OWLClass attribute), 269
- __slots__ (owlapy.parser.OWLClassExpression attribute), 270
- __slots__ (owlapy.parser.OWLDataAllValuesFrom attribute), 278
- __slots__ (owlapy.parser.OWLDataCardinalityRestriction attribute), 277
- __slots__ (owlapy.parser.OWLDataExactCardinality attribute), 273
- __slots__ (owlapy.parser.OWLDataHasValue attribute), 275
- __slots__ (owlapy.parser.OWLDataIntersectionOf attribute), 278
- __slots__ (owlapy.parser.OWLDataMaxCardinality attribute), 274
- __slots__ (owlapy.parser.OWLDataMinCardinality attribute), 274
- __slots__ (owlapy.parser.OWLDataProperty attribute), 265
- __slots__ (owlapy.parser.OWLDataSomeValuesFrom attribute), 273
- __slots__ (owlapy.parser.OWLDatatype attribute), 266
- __slots__ (owlapy.parser.OWLDatatypeRestriction attribute), 276
- __slots__ (owlapy.parser.OWLDataUnionOf attribute), 278
- __slots__ (owlapy.parser.OWLFacetRestriction attribute), 272
- __slots__ (owlapy.parser.OWLLiteral attribute), 262
- __slots__ (owlapy.parser.OWLNamedIndividual attribute), 261
- __slots__ (owlapy.parser.OWLObjectAllValuesFrom attribute), 277
- __slots__ (owlapy.parser.OWLObjectCardinalityRestriction attribute), 276
- __slots__ (owlapy.parser.OWLObjectComplementOf attribute), 271

- __slots__ (owlapy.parser.OwlObjectExactCardinality attribute), 271
- __slots__ (owlapy.parser.OwlObjectHasSelf attribute), 267
- __slots__ (owlapy.parser.OwlObjectHasValue attribute), 274
- __slots__ (owlapy.parser.OwlObjectIntersectionOf attribute), 267
- __slots__ (owlapy.parser.OwlObjectMaxCardinality attribute), 274
- __slots__ (owlapy.parser.OwlObjectMinCardinality attribute), 268
- __slots__ (owlapy.parser.OwlObjectOneOf attribute), 269
- __slots__ (owlapy.parser.OwlObjectProperty attribute), 264
- __slots__ (owlapy.parser.OwlObjectPropertyExpression attribute), 264
- __slots__ (owlapy.parser.OwlObjectSomeValuesFrom attribute), 268
- __slots__ (owlapy.parser.OwlObjectUnionOf attribute), 268
- __slots__ (owlapy.parser.OwlQuantifiedDataRestriction attribute), 272
- __slots__ (owlapy.parser.OwlQuantifiedObjectRestriction attribute), 272
- __slots__ (owlapy.providers.OwlDatatypeRestriction attribute), 286
- __slots__ (owlapy.providers.OwlFacetRestriction attribute), 287
- __slots__ (owlapy.providers.OwlLiteral attribute), 284
- __slots__ (owlapy.render.DLSyntaxObjectRenderer attribute), 308
- __slots__ (owlapy.render.IRI attribute), 290
- __slots__ (owlapy.render.ManchesterOwlSyntaxOwlObjectRenderer attribute), 308
- __slots__ (owlapy.render.OwlBooleanClassExpression attribute), 296
- __slots__ (owlapy.render.OwlClass attribute), 296
- __slots__ (owlapy.render.OwlClassExpression attribute), 295
- __slots__ (owlapy.render.OwlDataAllValuesFrom attribute), 301
- __slots__ (owlapy.render.OwlDataExactCardinality attribute), 302
- __slots__ (owlapy.render.OwlDataHasValue attribute), 301
- __slots__ (owlapy.render.OwlDataIntersectionOf attribute), 305
- __slots__ (owlapy.render.OwlDataMaxCardinality attribute), 302
- __slots__ (owlapy.render.OwlDataMinCardinality attribute), 302
- __slots__ (owlapy.render.OwlDataSomeValuesFrom attribute), 300
- __slots__ (owlapy.render.OwlDatatype attribute), 308
- __slots__ (owlapy.render.OwlDatatypeRestriction attribute), 307
- __slots__ (owlapy.render.OwlDataUnionOf attribute), 305
- __slots__ (owlapy.render.OwlEntity attribute), 294
- __slots__ (owlapy.render.OwlFacetRestriction attribute), 306
- __slots__ (owlapy.render.OwlLiteral attribute), 292
- __slots__ (owlapy.render.OwlNamedIndividual attribute), 291
- __slots__ (owlapy.render.OwlNaryBooleanClassExpression attribute), 303
- __slots__ (owlapy.render.OwlNaryDataRange attribute), 304
- __slots__ (owlapy.render.OwlObject attribute), 294
- __slots__ (owlapy.render.OwlObjectAllValuesFrom attribute), 298
- __slots__ (owlapy.render.OwlObjectComplementOf attribute), 298
- __slots__ (owlapy.render.OwlObjectExactCardinality attribute), 299
- __slots__ (owlapy.render.OwlObjectHasSelf attribute), 300
- __slots__ (owlapy.render.OwlObjectHasValue attribute), 306
- __slots__ (owlapy.render.OwlObjectIntersectionOf attribute), 298
- __slots__ (owlapy.render.OwlObjectInverseOf attribute), 294
- __slots__ (owlapy.render.OwlObjectMaxCardinality attribute), 300
- __slots__ (owlapy.render.OwlObjectMinCardinality attribute), 299
- __slots__ (owlapy.render.OwlObjectOneOf attribute), 307
- __slots__ (owlapy.render.OwlObjectSomeValuesFrom attribute), 297
- __slots__ (owlapy.render.OwlObjectUnionOf attribute), 298
- __slots__ (owlapy.render.OwlPropertyExpression attribute), 295
- __slots__ (owlapy.render.OwlRestriction attribute), 303
- __slots__ (owlapy.utils.EvaluatedDescriptionSet attribute), 332
- __slots__ (owlapy.utils.HasCardinality attribute), 313
- __slots__ (owlapy.utils.HasFiller attribute), 313
- __slots__ (owlapy.utils.HasIRI attribute), 313
- __slots__ (owlapy.utils.HasOperands attribute), 313
- __slots__ (owlapy.utils.OrderedOwlObject attribute), 333
- __slots__ (owlapy.utils.OwlClass attribute), 318
- __slots__ (owlapy.utils.OwlClassExpression attribute), 317
- __slots__ (owlapy.utils.OwlClassExpressionLengthMetric attribute), 331
- __slots__ (owlapy.utils.OwlDataAllValuesFrom attribute), 323
- __slots__ (owlapy.utils.OwlDataCardinalityRestriction attribute), 325
- __slots__ (owlapy.utils.OwlDataExactCardinality attribute), 322
- __slots__ (owlapy.utils.OwlDataHasValue attribute), 323
- __slots__ (owlapy.utils.OwlDataIntersectionOf attribute), 329
- __slots__ (owlapy.utils.OwlDataMaxCardinality attribute), 322

- `__slots__` (*owlapy.utils.OWLDataMinCardinality attribute*), 322
- `__slots__` (*owlapy.utils.OWLDataProperty attribute*), 317
- `__slots__` (*owlapy.utils.OWLDataSomeValuesFrom attribute*), 324
- `__slots__` (*owlapy.utils.OWLDatatype attribute*), 330
- `__slots__` (*owlapy.utils.OWLDatatypeRestriction attribute*), 327
- `__slots__` (*owlapy.utils.OWLDataUnionOf attribute*), 329
- `__slots__` (*owlapy.utils.OWLFacetRestriction attribute*), 327
- `__slots__` (*owlapy.utils.OWLLiteral attribute*), 314
- `__slots__` (*owlapy.utils.OWLNamedIndividual attribute*), 312
- `__slots__` (*owlapy.utils.OWLNaryBooleanClassExpression attribute*), 326
- `__slots__` (*owlapy.utils.OWLNaryDataRange attribute*), 329
- `__slots__` (*owlapy.utils.OWLObject attribute*), 330
- `__slots__` (*owlapy.utils.OWLObjectAllValuesFrom attribute*), 324
- `__slots__` (*owlapy.utils.OWLObjectCardinalityRestriction attribute*), 319
- `__slots__` (*owlapy.utils.OWLObjectComplementOf attribute*), 319
- `__slots__` (*owlapy.utils.OWLObjectExactCardinality attribute*), 321
- `__slots__` (*owlapy.utils.OWLObjectHasSelf attribute*), 321
- `__slots__` (*owlapy.utils.OWLObjectHasValue attribute*), 326
- `__slots__` (*owlapy.utils.OWLObjectIntersectionOf attribute*), 325
- `__slots__` (*owlapy.utils.OWLObjectInverseOf attribute*), 316
- `__slots__` (*owlapy.utils.OWLObjectMaxCardinality attribute*), 321
- `__slots__` (*owlapy.utils.OWLObjectMinCardinality attribute*), 320
- `__slots__` (*owlapy.utils.OWLObjectOneOf attribute*), 328
- `__slots__` (*owlapy.utils.OWLObjectProperty attribute*), 316
- `__slots__` (*owlapy.utils.OWLObjectSomeValuesFrom attribute*), 320
- `__slots__` (*owlapy.utils.OWLObjectUnionOf attribute*), 326
- `__slots__` (*owlapy.utils.OWLRestriction attribute*), 319
- `__slots__` (*owlapy.vocab.HasIRI attribute*), 335
- `__slots__` (*owlapy.vocab.IRI attribute*), 336
- `__slots__` (*owlapy.vocab.Namespaces attribute*), 337
- `__version__` (*in module owlapy*), 339

A

- `AbstractHierarchy` (*class in owlapy.owl_hierarchy*), 130
- `add_axiom()` (*owlapy.owl_ontology_manager.OntologyManager method*), 202
- `add_axiom()` (*owlapy.owl_ontology_manager.OWLOntologyManager method*), 200
- `add_axiom()` (*owlapy.owl_reasoner.OntologyManager method*), 227
- `AddImport` (*class in owlapy.owl_ontology_manager*), 201
- `all_data_property_values()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 249
- `all_data_property_values()` (*owlapy.owl_reasoner.OntologyReasoner method*), 243
- `all_data_property_values()` (*owlapy.owl_reasoner.OWLReasonerEx method*), 239
- `annotations()` (*owlapy.owl_axiom.OWLAxiom method*), 99
- `annotations()` (*owlapy.owl_ontology_manager.OWLAxiom method*), 182
- `annotations()` (*owlapy.owl_reasoner.OWLAxiom method*), 220
- `append()` (*owlapy.converter.Owl2SparqlConverter method*), 81
- `append_triple()` (*owlapy.converter.Owl2SparqlConverter method*), 81
- `apply_change()` (*owlapy.owl_ontology_manager.OntologyManager method*), 201
- `apply_change()` (*owlapy.owl_ontology_manager.OWLOntologyManager method*), 200
- `apply_change()` (*owlapy.owl_reasoner.OntologyManager method*), 227
- `as_anonymous_individual()` (*owlapy.owl_annotation.OWLAnnotationObject method*), 87
- `as_index()` (*in module owlapy.utils*), 334
- `as_intersection_of_min_max()` (*owlapy.class_expression.OWLDataExactCardinality method*), 61
- `as_intersection_of_min_max()` (*owlapy.class_expression.OWLObjectExactCardinality method*), 59
- `as_intersection_of_min_max()` (*owlapy.class_expression.restriction.OWLDataExactCardinality method*), 43
- `as_intersection_of_min_max()` (*owlapy.class_expression.restriction.OWLObjectExactCardinality method*), 38
- `as_intersection_of_min_max()` (*owlapy.converter.OWLDataExactCardinality method*), 67
- `as_intersection_of_min_max()` (*owlapy.converter.OWLObjectExactCardinality method*), 72
- `as_intersection_of_min_max()` (*owlapy.owl_ontology.OWLDataExactCardinality method*), 162
- `as_intersection_of_min_max()` (*owlapy.owl_ontology.OWLObjectExactCardinality method*), 159
- `as_intersection_of_min_max()` (*owlapy.owl_reasoner.OWLObjectExactCardinality method*), 214
- `as_intersection_of_min_max()` (*owlapy.parser.OWLDataExactCardinality method*), 273
- `as_intersection_of_min_max()` (*owlapy.parser.OWLObjectExactCardinality method*), 271
- `as_intersection_of_min_max()` (*owlapy.render.OWLDataExactCardinality method*), 302
- `as_intersection_of_min_max()` (*owlapy.render.OWLObjectExactCardinality method*), 299
- `as_intersection_of_min_max()` (*owlapy.utils.OWLDataExactCardinality method*), 322
- `as_intersection_of_min_max()` (*owlapy.utils.OWLObjectExactCardinality method*), 321
- `as_iri()` (*owlapy.class_expression.owl_class.IRI method*), 25

`as_iri()` (*owlapy.iri.IRI method*), 83
`as_iri()` (*owlapy.owl_annotation.OWLAnnotationObject method*), 87
`as_iri()` (*owlapy.owl_axiom.IRI method*), 96
`as_iri()` (*owlapy.owl_datatype.IRI method*), 120
`as_iri()` (*owlapy.owl_individual.IRI method*), 135
`as_iri()` (*owlapy.owl_ontology_manager.IRI method*), 177
`as_iri()` (*owlapy.owl_ontology.IRI method*), 154
`as_iri()` (*owlapy.owl_property.IRI method*), 204
`as_iri()` (*owlapy.owl_reasoner.IRI method*), 219
`as_iri()` (*owlapy.owlapi_adaptor.IRI method*), 256
`as_iri()` (*owlapy.parser.IRI method*), 261
`as_iri()` (*owlapy.render.IRI method*), 291
`as_iri()` (*owlapy.vocab.IRI method*), 336
`as_literal()` (*owlapy.class_expression.restriction.OWLLiteral method*), 34
`as_literal()` (*owlapy.converter.OWLLiteral method*), 77
`as_literal()` (*owlapy.iri.OWLAnnotationValue method*), 82
`as_literal()` (*owlapy.owl_annotation.OWLAnnotationValue method*), 87
`as_literal()` (*owlapy.owl_axiom.OWLAnnotationValue method*), 97
`as_literal()` (*owlapy.owl_axiom.OWLLiteral method*), 98
`as_literal()` (*owlapy.owl_literal.OWLAnnotationValue method*), 138
`as_literal()` (*owlapy.owl_literal.OWLLiteral method*), 142
`as_literal()` (*owlapy.owl_ontology.OWLLiteral method*), 153
`as_literal()` (*owlapy.owl_reasoner.OWLLiteral method*), 232
`as_literal()` (*owlapy.parser.OWLLiteral method*), 264
`as_literal()` (*owlapy.providers.OWLLiteral method*), 286
`as_literal()` (*owlapy.render.OWLLiteral method*), 293
`as_literal()` (*owlapy.utils.OWLLiteral method*), 315
`as_object_union_of()` (*owlapy.class_expression.OWLObjectOneOf method*), 62
`as_object_union_of()` (*owlapy.class_expression.restriction.OWLObjectOneOf method*), 41
`as_object_union_of()` (*owlapy.converter.OWLObjectOneOf method*), 66
`as_object_union_of()` (*owlapy.owl_ontology_manager.OWLObjectOneOf method*), 180
`as_object_union_of()` (*owlapy.owl_ontology.OWLObjectOneOf method*), 159
`as_object_union_of()` (*owlapy.owl_reasoner.OWLObjectOneOf method*), 213
`as_object_union_of()` (*owlapy.parser.OWLObjectOneOf method*), 270
`as_object_union_of()` (*owlapy.render.OWLObjectOneOf method*), 307
`as_object_union_of()` (*owlapy.utils.OWLObjectOneOf method*), 328
`as_pairwise_axioms()` (*owlapy.owl_axiom.OWLNaryAxiom method*), 101
`as_pairwise_axioms()` (*owlapy.owl_axiom.OWLNaryClassAxiom method*), 102
`as_pairwise_axioms()` (*owlapy.owl_axiom.OWLNaryIndividualAxiom method*), 103
`as_pairwise_axioms()` (*owlapy.owl_axiom.OWLNaryPropertyAxiom method*), 104
`as_pairwise_axioms()` (*owlapy.owl_ontology_manager.OWLNaryIndividualAxiom method*), 192
`as_pairwise_axioms()` (*owlapy.owl_ontology_manager.OWLNaryPropertyAxiom method*), 191
`as_query()` (*owlapy.converter.Owl2SparqlConverter method*), 81
`as_some_values_from()` (*owlapy.class_expression.OWLDataHasValue method*), 60
`as_some_values_from()` (*owlapy.class_expression.OWLObjectHasValue method*), 56
`as_some_values_from()` (*owlapy.class_expression.restriction.OWLDataHasValue method*), 44
`as_some_values_from()` (*owlapy.class_expression.restriction.OWLObjectHasValue method*), 40
`as_some_values_from()` (*owlapy.converter.OWLDataHasValue method*), 74
`as_some_values_from()` (*owlapy.converter.OWLObjectHasValue method*), 65
`as_some_values_from()` (*owlapy.owl_ontology_manager.OWLDataHasValue method*), 179
`as_some_values_from()` (*owlapy.owl_ontology_manager.OWLObjectHasValue method*), 181
`as_some_values_from()` (*owlapy.owl_ontology.OWLDataHasValue method*), 162
`as_some_values_from()` (*owlapy.owl_ontology.OWLObjectHasValue method*), 160
`as_some_values_from()` (*owlapy.owl_reasoner.OWLDataHasValue method*), 217
`as_some_values_from()` (*owlapy.owl_reasoner.OWLObjectHasValue method*), 213
`as_some_values_from()` (*owlapy.parser.OWLDataHasValue method*), 275
`as_some_values_from()` (*owlapy.parser.OWLObjectHasValue method*), 274
`as_some_values_from()` (*owlapy.render.OWLDataHasValue method*), 301
`as_some_values_from()` (*owlapy.render.OWLObjectHasValue method*), 306
`as_some_values_from()` (*owlapy.utils.OWLDataHasValue method*), 323
`as_some_values_from()` (*owlapy.utils.OWLObjectHasValue method*), 327
`as_str()` (*owlapy.class_expression.owl_class.IRI method*), 25
`as_str()` (*owlapy.iri.IRI method*), 83
`as_str()` (*owlapy.owl_axiom.IRI method*), 96
`as_str()` (*owlapy.owl_datatype.IRI method*), 120
`as_str()` (*owlapy.owl_individual.IRI method*), 135
`as_str()` (*owlapy.owl_ontology_manager.IRI method*), 177
`as_str()` (*owlapy.owl_ontology.IRI method*), 154

as_str() (owlapy.owl_property.IRI method), 204
 as_str() (owlapy.owl_reasoner.IRI method), 219
 as_str() (owlapy.owlapi_adaptor.IRI method), 256
 as_str() (owlapy.parser.IRI method), 261
 as_str() (owlapy.render.IRI method), 291
 as_str() (owlapy.vocab.IRI method), 336

B

BaseReasoner (class in owlapy.owl_reasoner), 239
 best() (owlapy.utils.EvaluatedDescriptionSet method), 333
 best_quality_value() (owlapy.utils.EvaluatedDescriptionSet method), 333
 BOOLEAN (owlapy.owl_literal.XSDVocabulary attribute), 139
 BOOLEAN (owlapy.vocab.XSDVocabulary attribute), 338
 BooleanOWLDatatype (in module owlapy.owl_literal), 142
 BooleanOWLDatatype (in module owlapy.owl_ontology), 151
 BooleanOWLDatatype (in module owlapy.parser), 262

C

cache_clear() (owlapy.owl_reasoner.LRUCache method), 233
 cache_clear() (owlapy.utils.LRUCache method), 335
 cache_info() (owlapy.owl_reasoner.LRUCache method), 233
 cache_info() (owlapy.utils.LRUCache method), 335
 ce (owlapy.converter.Owl2SparqlConverter attribute), 80
 children() (owlapy.owl_hierarchy.AbstractHierarchy method), 131
 class_expressions() (owlapy.owl_axiom.OwLNaryClassAxiom method), 102
 class_length (owlapy.utils.OwLClassExpressionLengthMetric attribute), 331
 classes_in_signature() (owlapy.owl_ontology_manager.Ontology method), 195
 classes_in_signature() (owlapy.owl_ontology_manager.OwLOntology method), 193
 classes_in_signature() (owlapy.owl_ontology.Ontology method), 171
 classes_in_signature() (owlapy.owl_ontology.OwLOntology method), 169
 classes_in_signature() (owlapy.owl_reasoner.Ontology method), 224
 classes_in_signature() (owlapy.owl_reasoner.OwLOntology method), 222
 ClassHierarchy (class in owlapy.owl_hierarchy), 132
 clean() (owlapy.utils.EvaluatedDescriptionSet method), 332
 cnt (owlapy.converter.Owl2SparqlConverter attribute), 80
 combine_nary_expressions() (in module owlapy.utils), 334
 ConceptOperandSorter (class in owlapy.utils), 333
 contains_named_equivalent_class() (owlapy.owl_axiom.OwLEquivalentClassesAxiom method), 102
 contains_named_equivalent_class() (owlapy.owl_ontology_manager.OwLEquivalentClassesAxiom method), 183
 contains_named_equivalent_class() (owlapy.owl_ontology.OwLEquivalentClassesAxiom method), 147
 contains_owl_nothing() (owlapy.owl_axiom.OwLEquivalentClassesAxiom method), 102
 contains_owl_nothing() (owlapy.owl_ontology_manager.OwLEquivalentClassesAxiom method), 183
 contains_owl_nothing() (owlapy.owl_ontology.OwLEquivalentClassesAxiom method), 148
 contains_owl_thing() (owlapy.owl_axiom.OwLEquivalentClassesAxiom method), 102
 contains_owl_thing() (owlapy.owl_ontology_manager.OwLEquivalentClassesAxiom method), 183
 contains_owl_thing() (owlapy.owl_ontology.OwLEquivalentClassesAxiom method), 148
 convert() (owlapy.converter.Owl2SparqlConverter method), 80
 convert_from_owlapi() (owlapy.owlapi_adaptor.OwLAPIAdaptor method), 257
 convert_to_owlapi() (owlapy.owlapi_adaptor.OwLAPIAdaptor method), 257
 converter (in module owlapy.converter), 81
 create() (owlapy.class_expression.owl_class.IRI static method), 25
 create() (owlapy.iri.IRI static method), 82
 create() (owlapy.owl_axiom.IRI static method), 95
 create() (owlapy.owl_datatype.IRI static method), 119
 create() (owlapy.owl_individual.IRI static method), 134
 create() (owlapy.owl_ontology_manager.IRI static method), 176
 create() (owlapy.owl_ontology.IRI static method), 154
 create() (owlapy.owl_property.IRI static method), 204
 create() (owlapy.owl_reasoner.IRI static method), 219
 create() (owlapy.owlapi_adaptor.IRI static method), 255
 create() (owlapy.parser.IRI static method), 260
 create() (owlapy.render.IRI static method), 290
 create() (owlapy.vocab.IRI static method), 336
 create_ontology() (owlapy.owl_ontology_manager.OntologyManager method), 201
 create_ontology() (owlapy.owl_ontology_manager.OwLOntologyManager method), 199
 create_ontology() (owlapy.owl_reasoner.OntologyManager method), 226
 current_variable (owlapy.converter.Owl2SparqlConverter property), 81

D

`data_all_values_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 332
`data_cardinality_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 332
`data_complement_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 332
`data_has_value_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 332
`data_intersection_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 332
`data_one_of_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 332
`data_properties_in_signature` () (*owlapy.owl_ontology_manager.Ontology* method), 195
`data_properties_in_signature` () (*owlapy.owl_ontology_manager.OWLOntology* method), 193
`data_properties_in_signature` () (*owlapy.owl_ontology.Ontology* method), 171
`data_properties_in_signature` () (*owlapy.owl_ontology.OWLOntology* method), 169
`data_properties_in_signature` () (*owlapy.owl_reasoner.Ontology* method), 224
`data_properties_in_signature` () (*owlapy.owl_reasoner.OWLOntology* method), 222
`data_property_domain_axioms` () (*owlapy.owl_ontology_manager.Ontology* method), 196
`data_property_domain_axioms` () (*owlapy.owl_ontology_manager.OWLOntology* method), 194
`data_property_domain_axioms` () (*owlapy.owl_ontology.Ontology* method), 172
`data_property_domain_axioms` () (*owlapy.owl_ontology.OWLOntology* method), 170
`data_property_domain_axioms` () (*owlapy.owl_reasoner.Ontology* method), 225
`data_property_domain_axioms` () (*owlapy.owl_reasoner.OWLOntology* method), 223
`data_property_domains` () (*owlapy.owl_hierarchy.OWLReasoner* method), 124
`data_property_domains` () (*owlapy.owl_reasoner.FastInstanceCheckerReasoner* method), 247
`data_property_domains` () (*owlapy.owl_reasoner.OntologyReasoner* method), 240
`data_property_domains` () (*owlapy.owl_reasoner.OWLReasoner* method), 233
`data_property_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 332
`data_property_range_axioms` () (*owlapy.owl_ontology_manager.Ontology* method), 196
`data_property_range_axioms` () (*owlapy.owl_ontology_manager.OWLOntology* method), 194
`data_property_range_axioms` () (*owlapy.owl_ontology.Ontology* method), 172
`data_property_range_axioms` () (*owlapy.owl_ontology.OWLOntology* method), 170
`data_property_range_axioms` () (*owlapy.owl_reasoner.Ontology* method), 225
`data_property_range_axioms` () (*owlapy.owl_reasoner.OWLOntology* method), 223
`data_property_ranges` () (*owlapy.owl_reasoner.FastInstanceCheckerReasoner* method), 247
`data_property_ranges` () (*owlapy.owl_reasoner.OWLReasonerEx* method), 239
`data_property_values` () (*owlapy.owl_hierarchy.OWLReasoner* method), 127
`data_property_values` () (*owlapy.owl_reasoner.FastInstanceCheckerReasoner* method), 249
`data_property_values` () (*owlapy.owl_reasoner.OntologyReasoner* method), 242
`data_property_values` () (*owlapy.owl_reasoner.OWLReasoner* method), 235
`data_some_values_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 331
`data_union_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 332
`datatype_length` (*owlapy.utils.OWLClassExpressionLengthMetric* attribute), 332
`DatatypePropertyHierarchy` (class in *owlapy.owl_hierarchy*), 133
`DATE` (*owlapy.owl_literal.XSDVocabulary* attribute), 139
`DATE` (*owlapy.vocab.XSDVocabulary* attribute), 338
`DATE_TIME` (*owlapy.owl_literal.XSDVocabulary* attribute), 139
`DATE_TIME` (*owlapy.vocab.XSDVocabulary* attribute), 338
`DATE_TIME_STAMP` (*owlapy.owl_literal.XSDVocabulary* attribute), 139
`DATE_TIME_STAMP` (*owlapy.vocab.XSDVocabulary* attribute), 338
`DateOWLDatatype` (in module *owlapy.owl_literal*), 142
`DateOWLDatatype` (in module *owlapy.owl_ontology*), 152
`DateOWLDatatype` (in module *owlapy.parser*), 262
`DateTimeOWLDatatype` (in module *owlapy.owl_literal*), 142
`DateTimeOWLDatatype` (in module *owlapy.owl_ontology*), 152
`DateTimeOWLDatatype` (in module *owlapy.parser*), 262
`DECIMAL` (*owlapy.owl_literal.XSDVocabulary* attribute), 139
`DECIMAL` (*owlapy.vocab.XSDVocabulary* attribute), 338
`different_individuals` () (*owlapy.owl_hierarchy.OWLReasoner* method), 126
`different_individuals` () (*owlapy.owl_reasoner.FastInstanceCheckerReasoner* method), 249
`different_individuals` () (*owlapy.owl_reasoner.OntologyReasoner* method), 242
`different_individuals` () (*owlapy.owl_reasoner.OWLReasoner* method), 234
`disjoint_classes` () (*owlapy.owl_hierarchy.OWLReasoner* method), 126
`disjoint_classes` () (*owlapy.owl_reasoner.FastInstanceCheckerReasoner* method), 248
`disjoint_classes` () (*owlapy.owl_reasoner.OntologyReasoner* method), 242
`disjoint_classes` () (*owlapy.owl_reasoner.OWLReasoner* method), 234
`disjoint_data_properties` () (*owlapy.owl_hierarchy.OWLReasoner* method), 128
`disjoint_data_properties` () (*owlapy.owl_reasoner.FastInstanceCheckerReasoner* method), 252
`disjoint_data_properties` () (*owlapy.owl_reasoner.OntologyReasoner* method), 245
`disjoint_data_properties` () (*owlapy.owl_reasoner.OWLReasoner* method), 237
`disjoint_object_properties` () (*owlapy.owl_hierarchy.OWLReasoner* method), 128
`disjoint_object_properties` () (*owlapy.owl_reasoner.FastInstanceCheckerReasoner* method), 251

`disjoint_object_properties()` (*owlapy.owl_reasoner.OntologyReasoner method*), 245
`disjoint_object_properties()` (*owlapy.owl_reasoner.OWLReasoner method*), 236
`DL_GRAMMAR` (*in module owlapy.parser*), 281
`dl_to_owl_expression()` (*in module owlapy*), 339
`dl_to_owl_expression()` (*in module owlapy.parser*), 283
`DLparser` (*in module owlapy.parser*), 283
`DLrenderer` (*in module owlapy.render*), 309
`DLSyntaxObjectRenderer` (*class in owlapy.render*), 308
`DLSyntaxParser` (*class in owlapy.parser*), 281
`DOUBLE` (*owlapy.owl_literal.XSDVocabulary attribute*), 139
`DOUBLE` (*owlapy.vocab.XSDVocabulary attribute*), 338
`DoubleOWLDatatype` (*in module owlapy.owl_literal*), 142
`DoubleOWLDatatype` (*in module owlapy.owl_ontology*), 151
`DoubleOWLDatatype` (*in module owlapy.parser*), 262
`download_external_files()` (*in module owlapy.static_funcs*), 309
`DURATION` (*owlapy.owl_literal.XSDVocabulary attribute*), 139
`DURATION` (*owlapy.vocab.XSDVocabulary attribute*), 338
`DurationOWLDatatype` (*in module owlapy.owl_literal*), 142
`DurationOWLDatatype` (*in module owlapy.owl_ontology*), 152
`DurationOWLDatatype` (*in module owlapy.parser*), 262

E

`equivalent_classes()` (*owlapy.owl_hierarchy.OWLReasoner method*), 125
`equivalent_classes()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 248
`equivalent_classes()` (*owlapy.owl_reasoner.OntologyReasoner method*), 241
`equivalent_classes()` (*owlapy.owl_reasoner.OWLReasoner method*), 234
`equivalent_classes_axioms()` (*owlapy.owl_ontology_manager.Ontology method*), 195
`equivalent_classes_axioms()` (*owlapy.owl_ontology_manager.OWLOntology method*), 194
`equivalent_classes_axioms()` (*owlapy.owl_ontology.Ontology method*), 171
`equivalent_classes_axioms()` (*owlapy.owl_ontology.OWLOntology method*), 170
`equivalent_classes_axioms()` (*owlapy.owl_reasoner.Ontology method*), 224
`equivalent_classes_axioms()` (*owlapy.owl_reasoner.OWLOntology method*), 223
`equivalent_data_properties()` (*owlapy.owl_hierarchy.OWLReasoner method*), 126
`equivalent_data_properties()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 251
`equivalent_data_properties()` (*owlapy.owl_reasoner.OntologyReasoner method*), 244
`equivalent_data_properties()` (*owlapy.owl_reasoner.OWLReasoner method*), 235
`equivalent_object_properties()` (*owlapy.owl_hierarchy.OWLReasoner method*), 126
`equivalent_object_properties()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 251
`equivalent_object_properties()` (*owlapy.owl_reasoner.OntologyReasoner method*), 244
`equivalent_object_properties()` (*owlapy.owl_reasoner.OWLReasoner method*), 235
`EvaluatedDescriptionSet` (*class in owlapy.utils*), 332

F

`FastInstanceCheckerReasoner` (*class in owlapy.owl_reasoner*), 246
`FLOAT` (*owlapy.owl_literal.XSDVocabulary attribute*), 139
`FLOAT` (*owlapy.vocab.XSDVocabulary attribute*), 338
`flush()` (*owlapy.owl_hierarchy.OWLReasoner method*), 127
`flush()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 250
`flush()` (*owlapy.owl_reasoner.OntologyReasoner method*), 243
`flush()` (*owlapy.owl_reasoner.OWLReasoner method*), 236
`for_all_de_morgan` (*owlapy.converter.Owl2SparqlConverter attribute*), 80
`forAll()` (*owlapy.converter.Owl2SparqlConverter method*), 81
`forAllDeMorgan()` (*owlapy.converter.Owl2SparqlConverter method*), 81
`FRACTION_DIGITS` (*owlapy.class_expression.OWLFacet attribute*), 57
`FRACTION_DIGITS` (*owlapy.class_expression.restriction.OWLFacet attribute*), 35
`FRACTION_DIGITS` (*owlapy.converter.OWLFacet attribute*), 79
`FRACTION_DIGITS` (*owlapy.owl_ontology.OWLFacet attribute*), 168
`FRACTION_DIGITS` (*owlapy.parser.OWLFacet attribute*), 267
`FRACTION_DIGITS` (*owlapy.providers.OWLFacet attribute*), 287
`FRACTION_DIGITS` (*owlapy.render.OWLFacet attribute*), 304
`FRACTION_DIGITS` (*owlapy.vocab.OWLFacet attribute*), 338
`from_str()` (*owlapy.class_expression.OWLFacet static method*), 57
`from_str()` (*owlapy.class_expression.restriction.OWLFacet static method*), 35
`from_str()` (*owlapy.converter.OWLFacet static method*), 78
`from_str()` (*owlapy.owl_ontology.OWLFacet static method*), 168
`from_str()` (*owlapy.parser.OWLFacet static method*), 266
`from_str()` (*owlapy.providers.OWLFacet static method*), 286

`from_str()` (*owlapy.render.OWLFacet static method*), 304
`from_str()` (*owlapy.vocab.OWLFacet static method*), 338
`FromOwlready2` (*class in owlapy.owl_ontology*), 173

G

`general_class_axioms()` (*owlapy.owl_ontology_manager.Ontology method*), 196
`general_class_axioms()` (*owlapy.owl_ontology_manager.OWL ontology method*), 194
`general_class_axioms()` (*owlapy.owl_ontology.Ontology method*), 172
`general_class_axioms()` (*owlapy.owl_ontology.OWL ontology method*), 170
`general_class_axioms()` (*owlapy.owl_reasoner.Ontology method*), 225
`general_class_axioms()` (*owlapy.owl_reasoner.OWL ontology method*), 223
`generic_visit()` (*owlapy.parser.DLSyntaxParser method*), 283
`generic_visit()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 281
`get_bottom_entity()` (*owlapy.owl_hierarchy.AbstractHierarchy class method*), 130
`get_bottom_entity()` (*owlapy.owl_hierarchy.ClassHierarchy class method*), 132
`get_bottom_entity()` (*owlapy.owl_hierarchy.DatatypePropertyHierarchy class method*), 133
`get_bottom_entity()` (*owlapy.owl_hierarchy.ObjectPropertyHierarchy class method*), 132
`get_cardinality()` (*owlapy.class_expression.OWLCardinalityRestriction method*), 53
`get_cardinality()` (*owlapy.class_expression.restriction.HasCardinality method*), 29
`get_cardinality()` (*owlapy.class_expression.restriction.OWLCardinalityRestriction method*), 37
`get_cardinality()` (*owlapy.meta_classes.HasCardinality method*), 85
`get_cardinality()` (*owlapy.utils.HasCardinality method*), 313
`get_class_expression()` (*owlapy.owl_axiom.OWLClassAssertionAxiom method*), 106
`get_class_expression()` (*owlapy.owl_axiom.OWLHasKeyAxiom method*), 101
`get_class_expression()` (*owlapy.owl_ontology_manager.OWLClassAssertionAxiom method*), 186
`get_class_expressions()` (*owlapy.owl_axiom.OWLDisjointUnionAxiom method*), 106
`get_class_expressions()` (*owlapy.owl_ontology_manager.OWLDisjointUnionAxiom method*), 184
`get_class_nnf()` (*owlapy.utils.NNF method*), 333
`get_data_range()` (*owlapy.owl_data_ranges.OWLDataComplementOf method*), 118
`get_data_range()` (*owlapy.owl_ontology.OWLDataComplementOf method*), 150
`get_data_range()` (*owlapy.owl_reasoner.OWLDataComplementOf method*), 221
`get_data_range()` (*owlapy.parser.OWLDataComplementOf method*), 278
`get_data_range()` (*owlapy.render.OWLDataComplementOf method*), 305
`get_data_range()` (*owlapy.utils.OWLDataComplementOf method*), 328
`get_datatype()` (*owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method*), 100
`get_datatype()` (*owlapy.class_expression.OWLDatatypeRestriction method*), 57
`get_datatype()` (*owlapy.class_expression.restriction.OWLDatatypeRestriction method*), 45
`get_datatype()` (*owlapy.class_expression.restriction.OWLLiteral method*), 34
`get_datatype()` (*owlapy.converter.OWLDatatypeRestriction method*), 66
`get_datatype()` (*owlapy.converter.OWLLiteral method*), 77
`get_datatype()` (*owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method*), 100
`get_datatype()` (*owlapy.owl_axiom.OWLLiteral method*), 98
`get_datatype()` (*owlapy.owl_literal.OWLLiteral method*), 142
`get_datatype()` (*owlapy.owl_ontology.OWLDatatypeRestriction method*), 163
`get_datatype()` (*owlapy.owl_ontology.OWLLiteral method*), 153
`get_datatype()` (*owlapy.owl_reasoner.OWLDatatypeRestriction method*), 216
`get_datatype()` (*owlapy.owl_reasoner.OWLLiteral method*), 232
`get_datatype()` (*owlapy.parser.OWLDatatypeRestriction method*), 276
`get_datatype()` (*owlapy.parser.OWLLiteral method*), 264
`get_datatype()` (*owlapy.providers.OWLDatatypeRestriction method*), 286
`get_datatype()` (*owlapy.providers.OWLLiteral method*), 286
`get_datatype()` (*owlapy.render.OWLDatatypeRestriction method*), 307
`get_datatype()` (*owlapy.render.OWLLiteral method*), 293
`get_datatype()` (*owlapy.utils.OWLDatatypeRestriction method*), 327
`get_datatype()` (*owlapy.utils.OWLLiteral method*), 315
`get_default()` (*owlapy.utils.OWLClassExpressionLengthMetric static method*), 332
`get_default_document_iri()` (*owlapy.owl_ontology.OWL ontology ID method*), 169
`get_domain()` (*owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method*), 109
`get_domain()` (*owlapy.owl_axiom.OWLPropertyDomainAxiom method*), 114
`get_domain()` (*owlapy.owl_ontology_manager.OWLPropertyDomainAxiom method*), 187
`get_entity()` (*owlapy.owl_axiom.OWLDeclarationAxiom method*), 100
`get_entity()` (*owlapy.owl_ontology_manager.OWLDeclarationAxiom method*), 186
`get_expression_length()` (*in module owlapy.utils*), 332
`get_facet()` (*owlapy.class_expression.OWLFacetRestriction method*), 58
`get_facet()` (*owlapy.class_expression.restriction.OWLFacetRestriction method*), 45
`get_facet()` (*owlapy.owl_ontology.OWLFacetRestriction method*), 165
`get_facet()` (*owlapy.owl_reasoner.OWLFacetRestriction method*), 216

`get_facet()` (*owlapy.parser.OWLFacetRestriction method*), 272
`get_facet()` (*owlapy.providers.OWLFacetRestriction method*), 287
`get_facet()` (*owlapy.render.OWLFacetRestriction method*), 306
`get_facet()` (*owlapy.utils.OWLFacetRestriction method*), 327
`get_facet_restrictions()` (*owlapy.class_expression.OWLDatatypeRestriction method*), 57
`get_facet_restrictions()` (*owlapy.class_expression.restriction.OWLDatatypeRestriction method*), 45
`get_facet_restrictions()` (*owlapy.converter.OWLDatatypeRestriction method*), 66
`get_facet_restrictions()` (*owlapy.owl_ontology.OWLDatatypeRestriction method*), 163
`get_facet_restrictions()` (*owlapy.owl_reasoner.OWLDatatypeRestriction method*), 216
`get_facet_restrictions()` (*owlapy.parser.OWLDatatypeRestriction method*), 276
`get_facet_restrictions()` (*owlapy.providers.OWLDatatypeRestriction method*), 286
`get_facet_restrictions()` (*owlapy.render.OWLDatatypeRestriction method*), 307
`get_facet_restrictions()` (*owlapy.utils.OWLDatatypeRestriction method*), 327
`get_facet_value()` (*owlapy.class_expression.OWLFacetRestriction method*), 58
`get_facet_value()` (*owlapy.class_expression.restriction.OWLFacetRestriction method*), 46
`get_facet_value()` (*owlapy.owl_ontology.OWLFacetRestriction method*), 165
`get_facet_value()` (*owlapy.owl_reasoner.OWLFacetRestriction method*), 216
`get_facet_value()` (*owlapy.parser.OWLFacetRestriction method*), 272
`get_facet_value()` (*owlapy.providers.OWLFacetRestriction method*), 287
`get_facet_value()` (*owlapy.render.OWLFacetRestriction method*), 306
`get_facet_value()` (*owlapy.utils.OWLFacetRestriction method*), 327
`get_filler()` (*owlapy.class_expression.OWLCardinalityRestriction method*), 53
`get_filler()` (*owlapy.class_expression.OWLHasValueRestriction method*), 52
`get_filler()` (*owlapy.class_expression.OWLQuantifiedDataRestriction method*), 55
`get_filler()` (*owlapy.class_expression.OWLQuantifiedObjectRestriction method*), 52
`get_filler()` (*owlapy.class_expression.restriction.HasFiller method*), 29
`get_filler()` (*owlapy.class_expression.restriction.OWLCardinalityRestriction method*), 37
`get_filler()` (*owlapy.class_expression.restriction.OWLHasValueRestriction method*), 36
`get_filler()` (*owlapy.class_expression.restriction.OWLQuantifiedDataRestriction method*), 41
`get_filler()` (*owlapy.class_expression.restriction.OWLQuantifiedObjectRestriction method*), 37
`get_filler()` (*owlapy.meta_classes.HasFiller method*), 85
`get_filler()` (*owlapy.owl_ontology_manager.OWLQuantifiedDataRestriction method*), 179
`get_filler()` (*owlapy.owl_ontology_manager.OWLQuantifiedObjectRestriction method*), 182
`get_filler()` (*owlapy.parser.OWLQuantifiedDataRestriction method*), 272
`get_filler()` (*owlapy.parser.OWLQuantifiedObjectRestriction method*), 272
`get_filler()` (*owlapy.utils.HasFiller method*), 313
`get_first_property()` (*owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom method*), 105
`get_first_property()` (*owlapy.owl_ontology_manager.OWLInverseObjectPropertiesAxiom method*), 191
`get_import_declaration()` (*owlapy.owl_ontology_manager.AddImport method*), 201
`get_individual()` (*owlapy.owl_axiom.OWLClassAssertionAxiom method*), 106
`get_individual()` (*owlapy.owl_ontology_manager.OWLClassAssertionAxiom method*), 186
`get_inverse()` (*owlapy.owl_ontology_manager.OWLObjectInverseOf method*), 198
`get_inverse()` (*owlapy.owl_ontology.OWLObjectInverseOf method*), 167
`get_inverse()` (*owlapy.owl_property.OWLObjectInverseOf method*), 207
`get_inverse()` (*owlapy.owl_reasoner.OWLObjectInverseOf method*), 229
`get_inverse()` (*owlapy.render.OWLObjectInverseOf method*), 294
`get_inverse()` (*owlapy.utils.OWLObjectInverseOf method*), 316
`get_inverse_property()` (*owlapy.class_expression.restriction.OWLObjectPropertyExpression method*), 31
`get_inverse_property()` (*owlapy.converter.OWLObjectProperty method*), 77
`get_inverse_property()` (*owlapy.owl_axiom.OWLObjectPropertyExpression method*), 91
`get_inverse_property()` (*owlapy.owl_hierarchy.OWLObjectProperty method*), 124
`get_inverse_property()` (*owlapy.owl_literal.OWLObjectProperty method*), 139
`get_inverse_property()` (*owlapy.owl_ontology_manager.OWLObjectInverseOf method*), 198
`get_inverse_property()` (*owlapy.owl_ontology_manager.OWLObjectProperty method*), 198
`get_inverse_property()` (*owlapy.owl_ontology.OWLObjectInverseOf method*), 167
`get_inverse_property()` (*owlapy.owl_ontology.OWLObjectProperty method*), 166
`get_inverse_property()` (*owlapy.owl_ontology.OWLObjectPropertyExpression method*), 167
`get_inverse_property()` (*owlapy.owl_property.OWLObjectInverseOf method*), 207
`get_inverse_property()` (*owlapy.owl_property.OWLObjectProperty method*), 206
`get_inverse_property()` (*owlapy.owl_property.OWLObjectPropertyExpression method*), 205
`get_inverse_property()` (*owlapy.owl_reasoner.OWLObjectInverseOf method*), 229
`get_inverse_property()` (*owlapy.owl_reasoner.OWLObjectProperty method*), 228
`get_inverse_property()` (*owlapy.owl_reasoner.OWLObjectPropertyExpression method*), 227
`get_inverse_property()` (*owlapy.parser.OWLObjectProperty method*), 264
`get_inverse_property()` (*owlapy.parser.OWLObjectPropertyExpression method*), 264
`get_inverse_property()` (*owlapy.render.OWLObjectInverseOf method*), 294
`get_inverse_property()` (*owlapy.utils.OWLObjectInverseOf method*), 316
`get_inverse_property()` (*owlapy.utils.OWLObjectProperty method*), 316

`get_literal()` (*owlapy.class_expression.restriction.OWLLiteral method*), 32
`get_literal()` (*owlapy.converter.OWLLiteral method*), 75
`get_literal()` (*owlapy.owl_axiom.OWLLiteral method*), 97
`get_literal()` (*owlapy.owl_literal.OWLLiteral method*), 140
`get_literal()` (*owlapy.owl_ontology.OWLLiteral method*), 152
`get_literal()` (*owlapy.owl_reasoner.OWLLiteral method*), 231
`get_literal()` (*owlapy.parser.OWLLiteral method*), 262
`get_literal()` (*owlapy.providers.OWLLiteral method*), 284
`get_literal()` (*owlapy.render.OWLLiteral method*), 292
`get_literal()` (*owlapy.utils.OWLLiteral method*), 314
`get_named_property()` (*owlapy.class_expression.restriction.OWLObjectPropertyExpression method*), 32
`get_named_property()` (*owlapy.converter.OWLObjectProperty method*), 77
`get_named_property()` (*owlapy.owl_axiom.OWLObjectPropertyExpression method*), 91
`get_named_property()` (*owlapy.owl_hierarchy.OWLObjectProperty method*), 123
`get_named_property()` (*owlapy.owl_literal.OWLObjectProperty method*), 139
`get_named_property()` (*owlapy.owl_ontology_manager.OWLObjectInverseOf method*), 198
`get_named_property()` (*owlapy.owl_ontology_manager.OWLObjectProperty method*), 198
`get_named_property()` (*owlapy.owl_ontology.OWLObjectInverseOf method*), 167
`get_named_property()` (*owlapy.owl_ontology.OWLObjectProperty method*), 165
`get_named_property()` (*owlapy.owl_ontology.OWLObjectPropertyExpression method*), 167
`get_named_property()` (*owlapy.owl_property.OWLObjectInverseOf method*), 207
`get_named_property()` (*owlapy.owl_property.OWLObjectProperty method*), 206
`get_named_property()` (*owlapy.owl_property.OWLObjectPropertyExpression method*), 205
`get_named_property()` (*owlapy.owl_reasoner.OWLObjectInverseOf method*), 229
`get_named_property()` (*owlapy.owl_reasoner.OWLObjectProperty method*), 228
`get_named_property()` (*owlapy.owl_reasoner.OWLObjectPropertyExpression method*), 227
`get_named_property()` (*owlapy.parser.OWLObjectProperty method*), 264
`get_named_property()` (*owlapy.parser.OWLObjectPropertyExpression method*), 264
`get_named_property()` (*owlapy.render.OWLObjectInverseOf method*), 294
`get_named_property()` (*owlapy.utils.OWLObjectInverseOf method*), 316
`get_named_property()` (*owlapy.utils.OWLObjectProperty method*), 316
`get_namespace()` (*owlapy.class_expression.owl_class.IRI method*), 26
`get_namespace()` (*owlapy.iri.IRI method*), 83
`get_namespace()` (*owlapy.owl_axiom.IRI method*), 96
`get_namespace()` (*owlapy.owl_datatype.IRI method*), 120
`get_namespace()` (*owlapy.owl_individual.IRI method*), 135
`get_namespace()` (*owlapy.owl_ontology_manager.IRI method*), 177
`get_namespace()` (*owlapy.owl_ontology.IRI method*), 155
`get_namespace()` (*owlapy.owl_property.IRI method*), 204
`get_namespace()` (*owlapy.owl_reasoner.IRI method*), 220
`get_namespace()` (*owlapy.owlapi_adaptor.IRI method*), 256
`get_namespace()` (*owlapy.parser.IRI method*), 261
`get_namespace()` (*owlapy.render.IRI method*), 291
`get_namespace()` (*owlapy.vocab.IRI method*), 337
`get_nnf()` (*owlapy.class_expression.class_expression.OWLAnonymousClassExpression method*), 20
`get_nnf()` (*owlapy.class_expression.class_expression.OWLClassExpression method*), 19
`get_nnf()` (*owlapy.class_expression.nary_boolean_expression.OWLClassExpression method*), 21
`get_nnf()` (*owlapy.class_expression.owl_class.OWLClass method*), 27
`get_nnf()` (*owlapy.class_expression.owl_class.OWLClassExpression method*), 24
`get_nnf()` (*owlapy.class_expression.OWLAnonymousClassExpression method*), 49
`get_nnf()` (*owlapy.class_expression.OWLClass method*), 50
`get_nnf()` (*owlapy.class_expression.OWLClassExpression method*), 48
`get_nnf()` (*owlapy.class_expression.restriction.OWLAnonymousClassExpression method*), 30
`get_nnf()` (*owlapy.class_expression.restriction.OWLClassExpression method*), 31
`get_nnf()` (*owlapy.converter.OWLClass method*), 68
`get_nnf()` (*owlapy.converter.OWLClassExpression method*), 68
`get_nnf()` (*owlapy.owl_axiom.OWLClass method*), 95
`get_nnf()` (*owlapy.owl_axiom.OWLClassExpression method*), 94
`get_nnf()` (*owlapy.owl_hierarchy.OWLClass method*), 123
`get_nnf()` (*owlapy.owl_ontology_manager.OWLClass method*), 179
`get_nnf()` (*owlapy.owl_ontology.OWLClass method*), 156
`get_nnf()` (*owlapy.owl_ontology.OWLClassExpression method*), 156
`get_nnf()` (*owlapy.owl_reasoner.OWLClass method*), 218
`get_nnf()` (*owlapy.owl_reasoner.OWLClassExpression method*), 210
`get_nnf()` (*owlapy.owlapi_adaptor.OWLClassExpression method*), 255
`get_nnf()` (*owlapy.parser.OWLClass method*), 269
`get_nnf()` (*owlapy.parser.OWLClassExpression method*), 271
`get_nnf()` (*owlapy.render.OWLClass method*), 297

`get_nnf()` (*owlapy.render.OWLClassExpression method*), 296
`get_nnf()` (*owlapy.utils.OWLClass method*), 318
`get_nnf()` (*owlapy.utils.OWLClassExpression method*), 317
`get_object()` (*owlapy.owl_axiom.OWLPropertyAssertionAxiom method*), 110
`get_object_complement_of()` (*owlapy.class_expression.class_expression.OWLAnonymousClassExpression method*), 19
`get_object_complement_of()` (*owlapy.class_expression.class_expression.OWLClassExpression method*), 19
`get_object_complement_of()` (*owlapy.class_expression.nary_boolean_expression.OWLClassExpression method*), 21
`get_object_complement_of()` (*owlapy.class_expression.owl_class.OWLClass method*), 26
`get_object_complement_of()` (*owlapy.class_expression.owl_class.OWLClassExpression method*), 24
`get_object_complement_of()` (*owlapy.class_expression.OWLAnonymousClassExpression method*), 48
`get_object_complement_of()` (*owlapy.class_expression.OWLClass method*), 50
`get_object_complement_of()` (*owlapy.class_expression.OWLClassExpression method*), 48
`get_object_complement_of()` (*owlapy.class_expression.restriction.OWLAnonymousClassExpression method*), 30
`get_object_complement_of()` (*owlapy.class_expression.restriction.OWLClassExpression method*), 31
`get_object_complement_of()` (*owlapy.converter.OWLClass method*), 68
`get_object_complement_of()` (*owlapy.converter.OWLClassExpression method*), 68
`get_object_complement_of()` (*owlapy.owl_axiom.OWLClass method*), 95
`get_object_complement_of()` (*owlapy.owl_axiom.OWLClassExpression method*), 94
`get_object_complement_of()` (*owlapy.owl_hierarchy.OWLClass method*), 123
`get_object_complement_of()` (*owlapy.owl_ontology_manager.OWLClass method*), 178
`get_object_complement_of()` (*owlapy.owl_ontology.OWLClass method*), 156
`get_object_complement_of()` (*owlapy.owl_ontology.OWLClassExpression method*), 156
`get_object_complement_of()` (*owlapy.owl_reasoner.OWLClass method*), 218
`get_object_complement_of()` (*owlapy.owl_reasoner.OWLClassExpression method*), 210
`get_object_complement_of()` (*owlapy.owlapi_adaptor.OWLClassExpression method*), 255
`get_object_complement_of()` (*owlapy.parser.OWLClass method*), 269
`get_object_complement_of()` (*owlapy.parser.OWLClassExpression method*), 270
`get_object_complement_of()` (*owlapy.render.OWLClass method*), 297
`get_object_complement_of()` (*owlapy.render.OWLClassExpression method*), 296
`get_object_complement_of()` (*owlapy.utils.OWLClass method*), 318
`get_object_complement_of()` (*owlapy.utils.OWLClassExpression method*), 317
`get_ontology()` (*owlapy.owl_ontology_manager.OWLOntologyChange method*), 199
`get_ontology_id()` (*owlapy.owl_ontology_manager.Ontology method*), 196
`get_ontology_id()` (*owlapy.owl_ontology_manager.OWLOntology method*), 195
`get_ontology_id()` (*owlapy.owl_ontology.Ontology method*), 172
`get_ontology_id()` (*owlapy.owl_ontology.OWLOntology method*), 171
`get_ontology_id()` (*owlapy.owl_reasoner.Ontology method*), 225
`get_ontology_id()` (*owlapy.owl_reasoner.OWLOntology method*), 224
`get_ontology_iri()` (*owlapy.owl_ontology.OWLOntologyID method*), 168
`get_operand()` (*owlapy.class_expression.class_expression.OWLObjectComplementOf method*), 20
`get_operand()` (*owlapy.class_expression.owl_class.OWLObjectComplementOf method*), 24
`get_operand()` (*owlapy.class_expression.OWLObjectComplementOf method*), 49
`get_operand()` (*owlapy.converter.OWLObjectComplementOf method*), 69
`get_operand()` (*owlapy.owl_ontology_manager.OWLObjectComplementOf method*), 181
`get_operand()` (*owlapy.owl_ontology.OWLObjectComplementOf method*), 157
`get_operand()` (*owlapy.owl_reasoner.OWLObjectComplementOf method*), 211
`get_operand()` (*owlapy.parser.OWLObjectComplementOf method*), 271
`get_operand()` (*owlapy.render.OWLObjectComplementOf method*), 298
`get_operand()` (*owlapy.utils.OWLObjectComplementOf method*), 319
`get_original_iri()` (*owlapy.owl_ontology_manager.Ontology method*), 197
`get_original_iri()` (*owlapy.owl_ontology.Ontology method*), 173
`get_original_iri()` (*owlapy.owl_reasoner.Ontology method*), 226
`get_owl_class()` (*owlapy.owl_axiom.OWLDisjointUnionAxiom method*), 106
`get_owl_class()` (*owlapy.owl_ontology_manager.OWLDisjointUnionAxiom method*), 184
`get_owl_disjoint_classes_axiom()` (*owlapy.owl_axiom.OWLDisjointUnionAxiom method*), 106
`get_owl_disjoint_classes_axiom()` (*owlapy.owl_ontology_manager.OWLDisjointUnionAxiom method*), 184
`get_owl_equivalent_classes_axiom()` (*owlapy.owl_axiom.OWLDisjointUnionAxiom method*), 106
`get_owl_equivalent_classes_axiom()` (*owlapy.owl_ontology_manager.OWLDisjointUnionAxiom method*), 184
`get_owl_ontology_manager()` (*owlapy.owl_ontology_manager.Ontology method*), 196
`get_owl_ontology_manager()` (*owlapy.owl_ontology_manager.OWLOntology method*), 195
`get_owl_ontology_manager()` (*owlapy.owl_ontology.Ontology method*), 172
`get_owl_ontology_manager()` (*owlapy.owl_ontology.OWLOntology method*), 171
`get_owl_ontology_manager()` (*owlapy.owl_reasoner.Ontology method*), 225
`get_owl_ontology_manager()` (*owlapy.owl_reasoner.OWLOntology method*), 224
`get_property()` (*owlapy.class_expression.OWLDataAllValuesFrom method*), 60
`get_property()` (*owlapy.class_expression.OWLDataCardinalityRestriction method*), 55
`get_property()` (*owlapy.class_expression.OWLDataHasValue method*), 60
`get_property()` (*owlapy.class_expression.OWLDataSomeValuesFrom method*), 59

`get_property()` (`owlapy.class_expression.OwlObjectAllValuesFrom` method), 56
`get_property()` (`owlapy.class_expression.OwlObjectCardinalityRestriction` method), 53
`get_property()` (`owlapy.class_expression.OwlObjectHasSelf` method), 54
`get_property()` (`owlapy.class_expression.OwlObjectHasValue` method), 56
`get_property()` (`owlapy.class_expression.OwlObjectRestriction` method), 52
`get_property()` (`owlapy.class_expression.OwlObjectSomeValuesFrom` method), 56
`get_property()` (`owlapy.class_expression.OwlRestriction` method), 51
`get_property()` (`owlapy.class_expression.restriction.OwlDataAllValuesFrom` method), 44
`get_property()` (`owlapy.class_expression.restriction.OwlDataCardinalityRestriction` method), 42
`get_property()` (`owlapy.class_expression.restriction.OwlDataHasValue` method), 44
`get_property()` (`owlapy.class_expression.restriction.OwlDataSomeValuesFrom` method), 43
`get_property()` (`owlapy.class_expression.restriction.OwlObjectAllValuesFrom` method), 39
`get_property()` (`owlapy.class_expression.restriction.OwlObjectCardinalityRestriction` method), 37
`get_property()` (`owlapy.class_expression.restriction.OwlObjectHasSelf` method), 40
`get_property()` (`owlapy.class_expression.restriction.OwlObjectHasValue` method), 40
`get_property()` (`owlapy.class_expression.restriction.OwlObjectRestriction` method), 36
`get_property()` (`owlapy.class_expression.restriction.OwlObjectSomeValuesFrom` method), 39
`get_property()` (`owlapy.class_expression.restriction.OwlRestriction` method), 35
`get_property()` (`owlapy.converter.OwlDataAllValuesFrom` method), 73
`get_property()` (`owlapy.converter.OwlDataCardinalityRestriction` method), 72
`get_property()` (`owlapy.converter.OwlDataHasValue` method), 74
`get_property()` (`owlapy.converter.OwlDataSomeValuesFrom` method), 73
`get_property()` (`owlapy.converter.OwlObjectAllValuesFrom` method), 70
`get_property()` (`owlapy.converter.OwlObjectCardinalityRestriction` method), 71
`get_property()` (`owlapy.converter.OwlObjectHasSelf` method), 72
`get_property()` (`owlapy.converter.OwlObjectHasValue` method), 65
`get_property()` (`owlapy.converter.OwlObjectSomeValuesFrom` method), 70
`get_property()` (`owlapy.owl_axiom.OwlAnnotation` method), 107
`get_property()` (`owlapy.owl_axiom.OwlAnnotationAssertionAxiom` method), 108
`get_property()` (`owlapy.owl_axiom.OwlAnnotationPropertyDomainAxiom` method), 109
`get_property()` (`owlapy.owl_axiom.OwlAnnotationPropertyRangeAxiom` method), 109
`get_property()` (`owlapy.owl_axiom.OwlPropertyAssertionAxiom` method), 110
`get_property()` (`owlapy.owl_axiom.OwlUnaryPropertyAxiom` method), 112
`get_property()` (`owlapy.owl_ontology_manager.OwlAnnotationAssertionAxiom` method), 184
`get_property()` (`owlapy.owl_ontology_manager.OwlDataHasValue` method), 179
`get_property()` (`owlapy.owl_ontology_manager.OwlObjectHasValue` method), 181
`get_property()` (`owlapy.owl_ontology.OwlDataAllValuesFrom` method), 161
`get_property()` (`owlapy.owl_ontology.OwlDataHasValue` method), 163
`get_property()` (`owlapy.owl_ontology.OwlDataSomeValuesFrom` method), 161
`get_property()` (`owlapy.owl_ontology.OwlObjectAllValuesFrom` method), 158
`get_property()` (`owlapy.owl_ontology.OwlObjectHasValue` method), 160
`get_property()` (`owlapy.owl_ontology.OwlObjectRestriction` method), 164
`get_property()` (`owlapy.owl_ontology.OwlObjectSomeValuesFrom` method), 158
`get_property()` (`owlapy.owl_ontology.OwlRestriction` method), 164
`get_property()` (`owlapy.owl_reasoner.OwlDataAllValuesFrom` method), 218
`get_property()` (`owlapy.owl_reasoner.OwlDataHasValue` method), 217
`get_property()` (`owlapy.owl_reasoner.OwlDataSomeValuesFrom` method), 215
`get_property()` (`owlapy.owl_reasoner.OwlObjectAllValuesFrom` method), 212
`get_property()` (`owlapy.owl_reasoner.OwlObjectCardinalityRestriction` method), 215
`get_property()` (`owlapy.owl_reasoner.OwlObjectHasValue` method), 213
`get_property()` (`owlapy.owl_reasoner.OwlObjectSomeValuesFrom` method), 211
`get_property()` (`owlapy.parser.OwlDataAllValuesFrom` method), 278
`get_property()` (`owlapy.parser.OwlDataCardinalityRestriction` method), 277
`get_property()` (`owlapy.parser.OwlDataHasValue` method), 275
`get_property()` (`owlapy.parser.OwlDataSomeValuesFrom` method), 273
`get_property()` (`owlapy.parser.OwlObjectAllValuesFrom` method), 277
`get_property()` (`owlapy.parser.OwlObjectCardinalityRestriction` method), 276
`get_property()` (`owlapy.parser.OwlObjectHasSelf` method), 267
`get_property()` (`owlapy.parser.OwlObjectHasValue` method), 274
`get_property()` (`owlapy.parser.OwlObjectSomeValuesFrom` method), 268
`get_property()` (`owlapy.render.OwlDataAllValuesFrom` method), 301
`get_property()` (`owlapy.render.OwlDataHasValue` method), 302
`get_property()` (`owlapy.render.OwlDataSomeValuesFrom` method), 300
`get_property()` (`owlapy.render.OwlObjectAllValuesFrom` method), 298
`get_property()` (`owlapy.render.OwlObjectHasSelf` method), 300
`get_property()` (`owlapy.render.OwlObjectHasValue` method), 306
`get_property()` (`owlapy.render.OwlObjectSomeValuesFrom` method), 297
`get_property()` (`owlapy.render.OwlRestriction` method), 304

`get_property()` (*owlapy.utils.OWLDataAllValuesFrom method*), 323
`get_property()` (*owlapy.utils.OWLDataCardinalityRestriction method*), 325
`get_property()` (*owlapy.utils.OWLDataHasValue method*), 323
`get_property()` (*owlapy.utils.OWLDataSomeValuesFrom method*), 324
`get_property()` (*owlapy.utils.OWLObjectAllValuesFrom method*), 324
`get_property()` (*owlapy.utils.OWLObjectCardinalityRestriction method*), 319
`get_property()` (*owlapy.utils.OWLObjectHasSelf method*), 321
`get_property()` (*owlapy.utils.OWLObjectHasValue method*), 326
`get_property()` (*owlapy.utils.OWLObjectSomeValuesFrom method*), 320
`get_property()` (*owlapy.utils.OWLRestriction method*), 320
`get_property_expressions()` (*owlapy.owl_axiom.OWLHasKeyAxiom method*), 101
`get_range()` (*owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method*), 109
`get_range()` (*owlapy.owl_axiom.OWLPropertyRangeAxiom method*), 115
`get_range()` (*owlapy.owl_ontology_manager.OWLPropertyRangeAxiom method*), 185
`get_remainder()` (*owlapy.class_expression.owl_class.IRI method*), 26
`get_remainder()` (*owlapy.iri.IRI method*), 83
`get_remainder()` (*owlapy.owl_axiom.IRI method*), 96
`get_remainder()` (*owlapy.owl_datatype.IRI method*), 120
`get_remainder()` (*owlapy.owl_individual.IRI method*), 135
`get_remainder()` (*owlapy.owl_ontology_manager.IRI method*), 177
`get_remainder()` (*owlapy.owl_ontology.IRI method*), 155
`get_remainder()` (*owlapy.owl_property.IRI method*), 205
`get_remainder()` (*owlapy.owl_reasoner.IRI method*), 220
`get_remainder()` (*owlapy.owlapi_adaptor.IRI method*), 256
`get_remainder()` (*owlapy.parser.IRI method*), 261
`get_remainder()` (*owlapy.render.IRI method*), 291
`get_remainder()` (*owlapy.vocab.IRI method*), 337
`get_root_ontology()` (*owlapy.owl_hierarchy.OWLReasoner method*), 130
`get_root_ontology()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 253
`get_root_ontology()` (*owlapy.owl_reasoner.OntologyReasoner method*), 246
`get_root_ontology()` (*owlapy.owl_reasoner.OWLReasoner method*), 238
`get_second_property()` (*owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom method*), 105
`get_second_property()` (*owlapy.owl_ontology_manager.OWLInverseObjectPropertiesAxiom method*), 191
`get_short_form()` (*owlapy.class_expression.owl_class.IRI method*), 25
`get_short_form()` (*owlapy.iri.IRI method*), 83
`get_short_form()` (*owlapy.owl_axiom.IRI method*), 96
`get_short_form()` (*owlapy.owl_datatype.IRI method*), 120
`get_short_form()` (*owlapy.owl_individual.IRI method*), 135
`get_short_form()` (*owlapy.owl_ontology_manager.IRI method*), 177
`get_short_form()` (*owlapy.owl_ontology.IRI method*), 155
`get_short_form()` (*owlapy.owl_property.IRI method*), 204
`get_short_form()` (*owlapy.owl_reasoner.IRI method*), 220
`get_short_form()` (*owlapy.owlapi_adaptor.IRI method*), 256
`get_short_form()` (*owlapy.parser.IRI method*), 261
`get_short_form()` (*owlapy.render.IRI method*), 291
`get_short_form()` (*owlapy.vocab.IRI method*), 337
`get_sub_class()` (*owlapy.owl_axiom.OWLSubClassOfAxiom method*), 105
`get_sub_class()` (*owlapy.owl_ontology_manager.OWLSubClassOfAxiom method*), 183
`get_sub_class()` (*owlapy.owl_ontology.OWLSubClassOfAxiom method*), 149
`get_sub_class()` (*owlapy.owl_reasoner.OWLSubClassOfAxiom method*), 220
`get_sub_property()` (*owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method*), 108
`get_sub_property()` (*owlapy.owl_axiom.OWLSubPropertyAxiom method*), 109
`get_sub_property()` (*owlapy.owl_ontology_manager.OWLSubPropertyAxiom method*), 185
`get_subject()` (*owlapy.owl_axiom.OWLAnnotationAssertionAxiom method*), 108
`get_subject()` (*owlapy.owl_axiom.OWLPropertyAssertionAxiom method*), 110
`get_subject()` (*owlapy.owl_ontology_manager.OWLAnnotationAssertionAxiom method*), 184
`get_super_class()` (*owlapy.owl_axiom.OWLSubClassOfAxiom method*), 105
`get_super_class()` (*owlapy.owl_ontology_manager.OWLSubClassOfAxiom method*), 183
`get_super_class()` (*owlapy.owl_ontology.OWLSubClassOfAxiom method*), 149
`get_super_class()` (*owlapy.owl_reasoner.OWLSubClassOfAxiom method*), 220
`get_super_property()` (*owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method*), 108
`get_super_property()` (*owlapy.owl_axiom.OWLSubPropertyAxiom method*), 110
`get_super_property()` (*owlapy.owl_ontology_manager.OWLSubPropertyAxiom method*), 185
`get_top_entity()` (*owlapy.owl_hierarchy.AbstractHierarchy class method*), 130
`get_top_entity()` (*owlapy.owl_hierarchy.ClassHierarchy class method*), 132
`get_top_entity()` (*owlapy.owl_hierarchy.DatatypePropertyHierarchy class method*), 133
`get_top_entity()` (*owlapy.owl_hierarchy.ObjectPropertyHierarchy class method*), 132
`get_top_level_cnf()` (*owlapy.utils.TopLevelCNF method*), 334

`get_top_level_dnf()` (*owlapy.utils.TopLevelDNF method*), 334
`get_value()` (*owlapy.owl_axiom.OWLAnnotation method*), 107
`get_value()` (*owlapy.owl_axiom.OWLAnnotationAssertionAxiom method*), 108
`get_value()` (*owlapy.owl_ontology_manager.OWLAnnotationAssertionAxiom method*), 184
`get_variable()` (*owlapy.converter.VariablesMapping method*), 79
`get_version_iri()` (*owlapy.owl_ontology.OWLOntologyID method*), 168
`grouping_vars` (*owlapy.converter.Owl2SparqlConverter attribute*), 80

H

`has_consistent_ontology()` (*owlapy.owlapi_adaptor.OWLAPIAdaptor method*), 257
`HasCardinality` (*class in owlapy.class_expression.restriction*), 29
`HasCardinality` (*class in owlapy.meta_classes*), 85
`HasCardinality` (*class in owlapy.utils*), 313
`HasFiller` (*class in owlapy.class_expression.restriction*), 29
`HasFiller` (*class in owlapy.meta_classes*), 84
`HasFiller` (*class in owlapy.utils*), 313
`HasIndex` (*class in owlapy.utils*), 333
`HasIRI` (*class in owlapy.meta_classes*), 84
`HasIRI` (*class in owlapy.owl_datatype*), 120
`HasIRI` (*class in owlapy.owl_hierarchy*), 123
`HasIRI` (*class in owlapy.owl_object*), 143
`HasIRI` (*class in owlapy.owl_ontology_manager*), 177
`HasIRI` (*class in owlapy.utils*), 312
`HasIRI` (*class in owlapy.vocab*), 335
`HasOperands` (*class in owlapy.class_expression.class_expression*), 18
`HasOperands` (*class in owlapy.class_expression.nary_boolean_expression*), 22
`HasOperands` (*class in owlapy.class_expression.restriction*), 29
`HasOperands` (*class in owlapy.meta_classes*), 84
`HasOperands` (*class in owlapy.owl_axiom*), 92
`HasOperands` (*class in owlapy.owl_data_ranges*), 117
`HasOperands` (*class in owlapy.utils*), 313
`having_conditions` (*owlapy.converter.Owl2SparqlConverter attribute*), 80
`HERMIT` (*owlapy.owl_reasoner.BaseReasoner attribute*), 239

I

`ind_data_properties()` (*owlapy.owl_reasoner.OWLReasonerEx method*), 240
`ind_object_properties()` (*owlapy.owl_reasoner.OWLReasonerEx method*), 240
`individuals()` (*owlapy.class_expression.OWLObjectOneOf method*), 61
`individuals()` (*owlapy.class_expression.restriction.OWLObjectOneOf method*), 40
`individuals()` (*owlapy.converter.OWLObjectOneOf method*), 65
`individuals()` (*owlapy.owl_axiom.OWLNaryIndividualAxiom method*), 103
`individuals()` (*owlapy.owl_ontology_manager.OWLNaryIndividualAxiom method*), 191
`individuals()` (*owlapy.owl_ontology_manager.OWLObjectOneOf method*), 180
`individuals()` (*owlapy.owl_ontology.OWLObjectOneOf method*), 158
`individuals()` (*owlapy.owl_reasoner.OWLObjectOneOf method*), 212
`individuals()` (*owlapy.parser.OWLObjectOneOf method*), 270
`individuals()` (*owlapy.render.OWLObjectOneOf method*), 307
`individuals()` (*owlapy.utils.OWLObjectOneOf method*), 328
`individuals_in_signature()` (*owlapy.owl_ontology_manager.Ontology method*), 195
`individuals_in_signature()` (*owlapy.owl_ontology_manager.OWLOntology method*), 193
`individuals_in_signature()` (*owlapy.owl_ontology.Ontology method*), 171
`individuals_in_signature()` (*owlapy.owl_ontology.OWLOntology method*), 169
`individuals_in_signature()` (*owlapy.owl_reasoner.Ontology method*), 224
`individuals_in_signature()` (*owlapy.owl_reasoner.OWLOntology method*), 223
`instances()` (*owlapy.owl_hierarchy.OWLReasoner method*), 127
`instances()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 250
`instances()` (*owlapy.owl_reasoner.OntologyReasoner method*), 243
`instances()` (*owlapy.owl_reasoner.OWLReasoner method*), 236
`instances()` (*owlapy.owl_reasoner.SyncReasoner method*), 253
`instances()` (*owlapy.owlapi_adaptor.OWLAPIAdaptor method*), 257
`INTEGER` (*owlapy.owl_literal.XSDVocabulary attribute*), 139
`INTEGER` (*owlapy.vocab.XSDVocabulary attribute*), 338
`IntegerOWLDatatype` (*in module owlapy.owl_literal*), 142
`IntegerOWLDatatype` (*in module owlapy.owl_ontology*), 151
`IntegerOWLDatatype` (*in module owlapy.parser*), 262
`IRI` (*class in owlapy.class_expression.owl_class*), 24
`IRI` (*class in owlapy.iri*), 82

IRI (class in owlapy.owl_axiom), 95
 IRI (class in owlapy.owl_datatype), 119
 IRI (class in owlapy.owl_individual), 134
 IRI (class in owlapy.owl_ontology), 154
 IRI (class in owlapy.owl_ontology_manager), 176
 IRI (class in owlapy.owl_property), 203
 IRI (class in owlapy.owl_reasoner), 219
 IRI (class in owlapy.owlapi_adaptor), 255
 IRI (class in owlapy.parser), 260
 IRI (class in owlapy.render), 290
 IRI (class in owlapy.vocab), 336
 iri (owlapy.class_expression.owl_class.OWLClass property), 26
 iri (owlapy.class_expression.OWLClass property), 49
 iri (owlapy.class_expression.restriction.OWLDatatype property), 34
 iri (owlapy.converter.OWLClass property), 67
 iri (owlapy.converter.OWLDatatype property), 78
 iri (owlapy.converter.OWLNamedIndividual property), 75
 iri (owlapy.meta_classes.HasIRI property), 84
 iri (owlapy.owl_axiom.OWLAnnotationProperty property), 107
 iri (owlapy.owl_axiom.OWLClass property), 94
 iri (owlapy.owl_axiom.OWLDatatype property), 92
 iri (owlapy.owl_axiom.OWLProperty property), 93
 iri (owlapy.owl_datatype.HasIRI property), 120
 iri (owlapy.owl_datatype.OWLDatatype property), 121
 iri (owlapy.owl_hierarchy.HasIRI property), 123
 iri (owlapy.owl_hierarchy.OWLClass property), 122
 iri (owlapy.owl_individual.OWLNamedIndividual property), 136
 iri (owlapy.owl_literal.OWLDatatype property), 138
 iri (owlapy.owl_object.HasIRI property), 143
 iri (owlapy.owl_ontology_manager.HasIRI property), 177
 iri (owlapy.owl_ontology_manager.OWLAnnotationProperty property), 185
 iri (owlapy.owl_ontology_manager.OWLClass property), 178
 iri (owlapy.owl_ontology_manager.OWLImportsDeclaration property), 200
 iri (owlapy.owl_ontology_manager.OWLNamedIndividual property), 193
 iri (owlapy.owl_ontology_manager.OWLProperty property), 199
 iri (owlapy.owl_ontology.OWLAnnotationProperty property), 149
 iri (owlapy.owl_ontology.OWLClass property), 155
 iri (owlapy.owl_ontology.OWLDatatype property), 151
 iri (owlapy.owl_ontology.OWLNamedIndividual property), 151
 iri (owlapy.owl_property.OWLProperty property), 206
 iri (owlapy.owl_reasoner.OWLClass property), 218
 iri (owlapy.owl_reasoner.OWLDatatype property), 222
 iri (owlapy.owl_reasoner.OWLNamedIndividual property), 230
 iri (owlapy.owlapi_adaptor.OWLNamedIndividual property), 256
 iri (owlapy.parser.OWLClass property), 269
 iri (owlapy.parser.OWLDatatype property), 266
 iri (owlapy.parser.OWLNamedIndividual property), 261
 iri (owlapy.render.OWLClass property), 296
 iri (owlapy.render.OWLDatatype property), 308
 iri (owlapy.render.OWLNamedIndividual property), 291
 iri (owlapy.utils.HasIRI property), 313
 iri (owlapy.utils.OWLClass property), 318
 iri (owlapy.utils.OWLDatatype property), 330
 iri (owlapy.utils.OWLNamedIndividual property), 312
 iri (owlapy.vocab.HasIRI property), 335
 is_annotated () (owlapy.owl_axiom.OWLAxiom method), 99
 is_annotated () (owlapy.owl_ontology_manager.OWLAxiom method), 182
 is_annotated () (owlapy.owl_reasoner.OWLAxiom method), 220
 is_annotation_axiom () (owlapy.owl_axiom.OWLAnnotationAxiom method), 107
 is_annotation_axiom () (owlapy.owl_axiom.OWLAxiom method), 99
 is_annotation_axiom () (owlapy.owl_ontology_manager.OWLAxiom method), 182
 is_annotation_axiom () (owlapy.owl_reasoner.OWLAxiom method), 220
 is_anonymous () (owlapy.class_expression.owl_class.OWLEntity method), 24
 is_anonymous () (owlapy.class_expression.restriction.OWLObject method), 35
 is_anonymous () (owlapy.converter.OWLEntity method), 78
 is_anonymous () (owlapy.owl_annotation.OWLObject method), 87
 is_anonymous () (owlapy.owl_axiom.OWLEntity method), 92
 is_anonymous () (owlapy.owl_axiom.OWLObject method), 91

`is_anonymous()` (*owlapy.owl_data_ranges.OWLObject method*), 117
`is_anonymous()` (*owlapy.owl_datatype.OWLEntity method*), 119
`is_anonymous()` (*owlapy.owl_individual.OWLEntity method*), 134
`is_anonymous()` (*owlapy.owl_individual.OWLObject method*), 134
`is_anonymous()` (*owlapy.owl_object.OWLEntity method*), 144
`is_anonymous()` (*owlapy.owl_object.OWLObject method*), 143
`is_anonymous()` (*owlapy.owl_ontology_manager.OWLObject method*), 178
`is_anonymous()` (*owlapy.owl_ontology_manager.OWLOntology method*), 195
`is_anonymous()` (*owlapy.owl_ontology.OWLObject method*), 154
`is_anonymous()` (*owlapy.owl_ontology.OWLOntology method*), 171
`is_anonymous()` (*owlapy.owl_ontology.OWLOntologyID method*), 169
`is_anonymous()` (*owlapy.owl_property.OWLEntity method*), 203
`is_anonymous()` (*owlapy.owl_property.OWLObject method*), 203
`is_anonymous()` (*owlapy.owl_reasoner.OWLOntology method*), 224
`is_anonymous()` (*owlapy.render.OWLEntity method*), 294
`is_anonymous()` (*owlapy.render.OWLObject method*), 294
`is_anonymous()` (*owlapy.utils.OWLObject method*), 330
`is_boolean()` (*owlapy.class_expression.restriction.OWLLiteral method*), 32
`is_boolean()` (*owlapy.converter.OWLLiteral method*), 75
`is_boolean()` (*owlapy.owl_axiom.OWLLiteral method*), 97
`is_boolean()` (*owlapy.owl_literal.OWLLiteral method*), 140
`is_boolean()` (*owlapy.owl_ontology.OWLLiteral method*), 152
`is_boolean()` (*owlapy.owl_reasoner.OWLLiteral method*), 231
`is_boolean()` (*owlapy.parser.OWLLiteral method*), 262
`is_boolean()` (*owlapy.providers.OWLLiteral method*), 284
`is_boolean()` (*owlapy.render.OWLLiteral method*), 292
`is_boolean()` (*owlapy.utils.OWLLiteral method*), 314
`is_child_of()` (*owlapy.owl_hierarchy.AbstractHierarchy method*), 131
`is_data_property_expression()` (*owlapy.class_expression.restriction.OWLDataPropertyExpression method*), 32
`is_data_property_expression()` (*owlapy.class_expression.restriction.OWLPropertyExpression method*), 31
`is_data_property_expression()` (*owlapy.owl_axiom.OWLDataPropertyExpression method*), 91
`is_data_property_expression()` (*owlapy.owl_axiom.OWLPropertyExpression method*), 92
`is_data_property_expression()` (*owlapy.owl_ontology.OWLDataPropertyExpression method*), 168
`is_data_property_expression()` (*owlapy.owl_ontology.OWLPropertyExpression method*), 166
`is_data_property_expression()` (*owlapy.owl_property.OWLDataPropertyExpression method*), 206
`is_data_property_expression()` (*owlapy.owl_property.OWLPropertyExpression method*), 205
`is_data_property_expression()` (*owlapy.owl_reasoner.OWLDataPropertyExpression method*), 230
`is_data_property_expression()` (*owlapy.owl_reasoner.OWLPropertyExpression method*), 229
`is_data_property_expression()` (*owlapy.render.OWLPropertyExpression method*), 295
`is_data_restriction()` (*owlapy.class_expression.OWLDataRestriction method*), 53
`is_data_restriction()` (*owlapy.class_expression.OWLRestriction method*), 51
`is_data_restriction()` (*owlapy.class_expression.restriction.OWLDataRestriction method*), 41
`is_data_restriction()` (*owlapy.class_expression.restriction.OWLRestriction method*), 35
`is_data_restriction()` (*owlapy.owl_ontology.OWLDataRestriction method*), 164
`is_data_restriction()` (*owlapy.owl_ontology.OWLRestriction method*), 164
`is_data_restriction()` (*owlapy.render.OWLRestriction method*), 304
`is_data_restriction()` (*owlapy.utils.OWLRestriction method*), 320
`is_date()` (*owlapy.class_expression.restriction.OWLLiteral method*), 33
`is_date()` (*owlapy.converter.OWLLiteral method*), 76
`is_date()` (*owlapy.owl_axiom.OWLLiteral method*), 98
`is_date()` (*owlapy.owl_literal.OWLLiteral method*), 141
`is_date()` (*owlapy.owl_ontology.OWLLiteral method*), 153
`is_date()` (*owlapy.owl_reasoner.OWLLiteral method*), 231
`is_date()` (*owlapy.parser.OWLLiteral method*), 263
`is_date()` (*owlapy.providers.OWLLiteral method*), 285
`is_date()` (*owlapy.render.OWLLiteral method*), 292
`is_date()` (*owlapy.utils.OWLLiteral method*), 315
`is_datetime()` (*owlapy.class_expression.restriction.OWLLiteral method*), 33
`is_datetime()` (*owlapy.converter.OWLLiteral method*), 76
`is_datetime()` (*owlapy.owl_axiom.OWLLiteral method*), 98
`is_datetime()` (*owlapy.owl_literal.OWLLiteral method*), 141
`is_datetime()` (*owlapy.owl_ontology.OWLLiteral method*), 153
`is_datetime()` (*owlapy.owl_reasoner.OWLLiteral method*), 231
`is_datetime()` (*owlapy.parser.OWLLiteral method*), 263
`is_datetime()` (*owlapy.providers.OWLLiteral method*), 285
`is_datetime()` (*owlapy.render.OWLLiteral method*), 293
`is_datetime()` (*owlapy.utils.OWLLiteral method*), 315
`is_double()` (*owlapy.class_expression.restriction.OWLLiteral method*), 33

`is_double()` (*owlapy.converter.OWLLiteral method*), 75
`is_double()` (*owlapy.owl_axiom.OWLLiteral method*), 97
`is_double()` (*owlapy.owl_literal.OWLLiteral method*), 140
`is_double()` (*owlapy.owl_ontology.OWLLiteral method*), 152
`is_double()` (*owlapy.owl_reasoner.OWLLiteral method*), 231
`is_double()` (*owlapy.parser.OWLLiteral method*), 262
`is_double()` (*owlapy.providers.OWLLiteral method*), 284
`is_double()` (*owlapy.render.OWLLiteral method*), 292
`is_double()` (*owlapy.utils.OWLLiteral method*), 314
`is_duration()` (*owlapy.class_expression.restriction.OWLLiteral method*), 33
`is_duration()` (*owlapy.converter.OWLLiteral method*), 76
`is_duration()` (*owlapy.owl_axiom.OWLLiteral method*), 98
`is_duration()` (*owlapy.owl_literal.OWLLiteral method*), 141
`is_duration()` (*owlapy.owl_ontology.OWLLiteral method*), 153
`is_duration()` (*owlapy.owl_reasoner.OWLLiteral method*), 232
`is_duration()` (*owlapy.parser.OWLLiteral method*), 263
`is_duration()` (*owlapy.providers.OWLLiteral method*), 285
`is_duration()` (*owlapy.render.OWLLiteral method*), 293
`is_duration()` (*owlapy.utils.OWLLiteral method*), 315
`is_integer()` (*owlapy.class_expression.restriction.OWLLiteral method*), 33
`is_integer()` (*owlapy.converter.OWLLiteral method*), 76
`is_integer()` (*owlapy.owl_axiom.OWLLiteral method*), 97
`is_integer()` (*owlapy.owl_literal.OWLLiteral method*), 141
`is_integer()` (*owlapy.owl_ontology.OWLLiteral method*), 152
`is_integer()` (*owlapy.owl_reasoner.OWLLiteral method*), 231
`is_integer()` (*owlapy.parser.OWLLiteral method*), 263
`is_integer()` (*owlapy.providers.OWLLiteral method*), 285
`is_integer()` (*owlapy.render.OWLLiteral method*), 292
`is_integer()` (*owlapy.utils.OWLLiteral method*), 314
`is_isolated()` (*owlapy.owl_hierarchy.OWLReasoner method*), 130
`is_isolated()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 247
`is_isolated()` (*owlapy.owl_reasoner.OntologyReasoner method*), 246
`is_isolated()` (*owlapy.owl_reasoner.OWLReasoner method*), 238
`is_literal()` (*owlapy.class_expression.restriction.OWLLiteral method*), 34
`is_literal()` (*owlapy.converter.OWLLiteral method*), 76
`is_literal()` (*owlapy.iri.OWLAnnotationValue method*), 82
`is_literal()` (*owlapy.owl_annotation.OWLAnnotationValue method*), 87
`is_literal()` (*owlapy.owl_axiom.OWLAnnotationValue method*), 97
`is_literal()` (*owlapy.owl_axiom.OWLLiteral method*), 98
`is_literal()` (*owlapy.owl_literal.OWLAnnotationValue method*), 138
`is_literal()` (*owlapy.owl_literal.OWLLiteral method*), 141
`is_literal()` (*owlapy.owl_ontology.OWLLiteral method*), 153
`is_literal()` (*owlapy.owl_reasoner.OWLLiteral method*), 232
`is_literal()` (*owlapy.parser.OWLLiteral method*), 263
`is_literal()` (*owlapy.providers.OWLLiteral method*), 285
`is_literal()` (*owlapy.render.OWLLiteral method*), 293
`is_literal()` (*owlapy.utils.OWLLiteral method*), 315
`is_logical_axiom()` (*owlapy.owl_axiom.OWLXiom method*), 99
`is_logical_axiom()` (*owlapy.owl_axiom.OWLLogicalXiom method*), 99
`is_logical_axiom()` (*owlapy.owl_ontology_manager.OWLXiom method*), 182
`is_logical_axiom()` (*owlapy.owl_reasoner.OWLXiom method*), 220
`is_nothing()` (*owlapy.class_expression.owl_class.IRI method*), 25
`is_nothing()` (*owlapy.iri.IRI method*), 83
`is_nothing()` (*owlapy.owl_axiom.IRI method*), 96
`is_nothing()` (*owlapy.owl_datatype.IRI method*), 119
`is_nothing()` (*owlapy.owl_individual.IRI method*), 135
`is_nothing()` (*owlapy.owl_ontology_manager.IRI method*), 176
`is_nothing()` (*owlapy.owl_ontology.IRI method*), 154
`is_nothing()` (*owlapy.owl_property.IRI method*), 204
`is_nothing()` (*owlapy.owl_reasoner.IRI method*), 219
`is_nothing()` (*owlapy.owlapi_adaptor.IRI method*), 255
`is_nothing()` (*owlapy.parser.IRI method*), 260
`is_nothing()` (*owlapy.render.IRI method*), 290
`is_nothing()` (*owlapy.vocab.IRI method*), 336
`is_object_property_expression()` (*owlapy.class_expression.restriction.OWLObjectPropertyExpression method*), 32
`is_object_property_expression()` (*owlapy.class_expression.restriction.OWLPropertyExpression method*), 31
`is_object_property_expression()` (*owlapy.owl_axiom.OWLObjectPropertyExpression method*), 91
`is_object_property_expression()` (*owlapy.owl_axiom.OWLPropertyExpression method*), 93

`is_object_property_expression()` (`owlapy.owl_ontology.OwlObjectPropertyExpression` method), 167
`is_object_property_expression()` (`owlapy.owl_ontology.OwlPropertyExpression` method), 166
`is_object_property_expression()` (`owlapy.owl_property.OwlObjectPropertyExpression` method), 206
`is_object_property_expression()` (`owlapy.owl_property.OwlPropertyExpression` method), 205
`is_object_property_expression()` (`owlapy.owl_reasoner.OwlObjectPropertyExpression` method), 228
`is_object_property_expression()` (`owlapy.owl_reasoner.OwlPropertyExpression` method), 229
`is_object_property_expression()` (`owlapy.parser.OwlObjectPropertyExpression` method), 264
`is_object_property_expression()` (`owlapy.render.OwlPropertyExpression` method), 295
`is_object_restriction()` (`owlapy.class_expression.OwlObjectRestriction` method), 52
`is_object_restriction()` (`owlapy.class_expression.OwlRestriction` method), 51
`is_object_restriction()` (`owlapy.class_expression.restriction.OwlObjectRestriction` method), 36
`is_object_restriction()` (`owlapy.class_expression.restriction.OwlRestriction` method), 35
`is_object_restriction()` (`owlapy.owl_ontology.OwlObjectRestriction` method), 164
`is_object_restriction()` (`owlapy.owl_ontology.OwlRestriction` method), 164
`is_object_restriction()` (`owlapy.render.OwlRestriction` method), 304
`is_object_restriction()` (`owlapy.utils.OwlRestriction` method), 320
`is_owl_nothing()` (`owlapy.class_expression.class_expression.OwlAnonymousClassExpression` method), 19
`is_owl_nothing()` (`owlapy.class_expression.class_expression.OwlClassExpression` method), 19
`is_owl_nothing()` (`owlapy.class_expression.nary_boolean_expression.OwlClassExpression` method), 21
`is_owl_nothing()` (`owlapy.class_expression.owl_class.OwlClass` method), 26
`is_owl_nothing()` (`owlapy.class_expression.owl_class.OwlClassExpression` method), 23
`is_owl_nothing()` (`owlapy.class_expression.OwlAnonymousClassExpression` method), 48
`is_owl_nothing()` (`owlapy.class_expression.OwlClass` method), 50
`is_owl_nothing()` (`owlapy.class_expression.OwlClassExpression` method), 48
`is_owl_nothing()` (`owlapy.class_expression.restriction.OwlAnonymousClassExpression` method), 30
`is_owl_nothing()` (`owlapy.class_expression.restriction.OwlClassExpression` method), 30
`is_owl_nothing()` (`owlapy.converter.OwlClass` method), 68
`is_owl_nothing()` (`owlapy.converter.OwlClassExpression` method), 68
`is_owl_nothing()` (`owlapy.owl_axiom.OwlClass` method), 94
`is_owl_nothing()` (`owlapy.owl_axiom.OwlClassExpression` method), 94
`is_owl_nothing()` (`owlapy.owl_hierarchy.OwlClass` method), 123
`is_owl_nothing()` (`owlapy.owl_ontology_manager.OwlClass` method), 178
`is_owl_nothing()` (`owlapy.owl_ontology.OwlClass` method), 155
`is_owl_nothing()` (`owlapy.owl_ontology.OwlClassExpression` method), 156
`is_owl_nothing()` (`owlapy.owl_reasoner.OwlClass` method), 218
`is_owl_nothing()` (`owlapy.owl_reasoner.OwlClassExpression` method), 210
`is_owl_nothing()` (`owlapy.owlapi_adaptor.OwlClassExpression` method), 255
`is_owl_nothing()` (`owlapy.parser.OwlClass` method), 269
`is_owl_nothing()` (`owlapy.parser.OwlClassExpression` method), 270
`is_owl_nothing()` (`owlapy.render.OwlClass` method), 297
`is_owl_nothing()` (`owlapy.render.OwlClassExpression` method), 296
`is_owl_nothing()` (`owlapy.utils.OwlClass` method), 318
`is_owl_nothing()` (`owlapy.utils.OwlClassExpression` method), 317
`is_owl_thing()` (`owlapy.class_expression.class_expression.OwlAnonymousClassExpression` method), 19
`is_owl_thing()` (`owlapy.class_expression.class_expression.OwlClassExpression` method), 19
`is_owl_thing()` (`owlapy.class_expression.nary_boolean_expression.OwlClassExpression` method), 21
`is_owl_thing()` (`owlapy.class_expression.owl_class.OwlClass` method), 26
`is_owl_thing()` (`owlapy.class_expression.owl_class.OwlClassExpression` method), 23
`is_owl_thing()` (`owlapy.class_expression.OwlAnonymousClassExpression` method), 48
`is_owl_thing()` (`owlapy.class_expression.OwlClass` method), 50
`is_owl_thing()` (`owlapy.class_expression.OwlClassExpression` method), 48
`is_owl_thing()` (`owlapy.class_expression.restriction.OwlAnonymousClassExpression` method), 30
`is_owl_thing()` (`owlapy.class_expression.restriction.OwlClassExpression` method), 30
`is_owl_thing()` (`owlapy.converter.OwlClass` method), 68
`is_owl_thing()` (`owlapy.converter.OwlClassExpression` method), 68
`is_owl_thing()` (`owlapy.owl_axiom.OwlClass` method), 94
`is_owl_thing()` (`owlapy.owl_axiom.OwlClassExpression` method), 93
`is_owl_thing()` (`owlapy.owl_hierarchy.OwlClass` method), 122
`is_owl_thing()` (`owlapy.owl_ontology_manager.OwlClass` method), 178
`is_owl_thing()` (`owlapy.owl_ontology.OwlClass` method), 155
`is_owl_thing()` (`owlapy.owl_ontology.OwlClassExpression` method), 156
`is_owl_thing()` (`owlapy.owl_reasoner.OwlClass` method), 218
`is_owl_thing()` (`owlapy.owl_reasoner.OwlClassExpression` method), 210
`is_owl_thing()` (`owlapy.owlapi_adaptor.OwlClassExpression` method), 254
`is_owl_thing()` (`owlapy.parser.OwlClass` method), 269
`is_owl_thing()` (`owlapy.parser.OwlClassExpression` method), 270
`is_owl_thing()` (`owlapy.render.OwlClass` method), 296
`is_owl_thing()` (`owlapy.render.OwlClassExpression` method), 295

`is_owl_thing()` (*owlapy.utils.OWLClass method*), 318
`is_owl_thing()` (*owlapy.utils.OWLClassExpression method*), 317
`is_owl_top_data_property()` (*owlapy.class_expression.restriction.OWLPropertyExpression method*), 31
`is_owl_top_data_property()` (*owlapy.converter.OWLDataProperty method*), 78
`is_owl_top_data_property()` (*owlapy.owl_axiom.OWLPropertyExpression method*), 93
`is_owl_top_data_property()` (*owlapy.owl_hierarchy.OWLDataProperty method*), 124
`is_owl_top_data_property()` (*owlapy.owl_literal.OWLDataProperty method*), 140
`is_owl_top_data_property()` (*owlapy.owl_ontology_manager.OWLDataProperty method*), 197
`is_owl_top_data_property()` (*owlapy.owl_ontology.OWLDataProperty method*), 165
`is_owl_top_data_property()` (*owlapy.owl_ontology.OWLPropertyExpression method*), 166
`is_owl_top_data_property()` (*owlapy.owl_property.OWLDataProperty method*), 208
`is_owl_top_data_property()` (*owlapy.owl_property.OWLPropertyExpression method*), 205
`is_owl_top_data_property()` (*owlapy.owl_reasoner.OWLDataProperty method*), 228
`is_owl_top_data_property()` (*owlapy.owl_reasoner.OWLPropertyExpression method*), 230
`is_owl_top_data_property()` (*owlapy.parser.OWLDataProperty method*), 265
`is_owl_top_data_property()` (*owlapy.render.OWLPropertyExpression method*), 295
`is_owl_top_data_property()` (*owlapy.utils.OWLDataProperty method*), 317
`is_owl_top_object_property()` (*owlapy.class_expression.restriction.OWLPropertyExpression method*), 31
`is_owl_top_object_property()` (*owlapy.converter.OWLObjectProperty method*), 77
`is_owl_top_object_property()` (*owlapy.owl_axiom.OWLPropertyExpression method*), 93
`is_owl_top_object_property()` (*owlapy.owl_hierarchy.OWLObjectProperty method*), 124
`is_owl_top_object_property()` (*owlapy.owl_literal.OWLObjectProperty method*), 139
`is_owl_top_object_property()` (*owlapy.owl_ontology_manager.OWLObjectProperty method*), 199
`is_owl_top_object_property()` (*owlapy.owl_ontology.OWLObjectProperty method*), 166
`is_owl_top_object_property()` (*owlapy.owl_ontology.OWLPropertyExpression method*), 166
`is_owl_top_object_property()` (*owlapy.owl_property.OWLObjectProperty method*), 206
`is_owl_top_object_property()` (*owlapy.owl_property.OWLPropertyExpression method*), 205
`is_owl_top_object_property()` (*owlapy.owl_reasoner.OWLObjectProperty method*), 228
`is_owl_top_object_property()` (*owlapy.owl_reasoner.OWLPropertyExpression method*), 229
`is_owl_top_object_property()` (*owlapy.parser.OWLObjectProperty method*), 265
`is_owl_top_object_property()` (*owlapy.render.OWLPropertyExpression method*), 295
`is_owl_top_object_property()` (*owlapy.utils.OWLObjectProperty method*), 316
`is_parent_of()` (*owlapy.owl_hierarchy.AbstractHierarchy method*), 131
`is_reserved_vocabulary()` (*owlapy.class_expression.owl_class.IRI method*), 25
`is_reserved_vocabulary()` (*owlapy.iri.IRI method*), 83
`is_reserved_vocabulary()` (*owlapy.owl_axiom.IRI method*), 96
`is_reserved_vocabulary()` (*owlapy.owl_datatype.IRI method*), 120
`is_reserved_vocabulary()` (*owlapy.owl_individual.IRI method*), 135
`is_reserved_vocabulary()` (*owlapy.owl_ontology_manager.IRI method*), 176
`is_reserved_vocabulary()` (*owlapy.owl_ontology.IRI method*), 154
`is_reserved_vocabulary()` (*owlapy.owl_property.IRI method*), 204
`is_reserved_vocabulary()` (*owlapy.owl_reasoner.IRI method*), 219
`is_reserved_vocabulary()` (*owlapy.owlapi_adaptor.IRI method*), 255
`is_reserved_vocabulary()` (*owlapy.parser.IRI method*), 261
`is_reserved_vocabulary()` (*owlapy.render.IRI method*), 290
`is_reserved_vocabulary()` (*owlapy.vocab.IRI method*), 336
`is_string()` (*owlapy.class_expression.restriction.OWLLiteral method*), 33
`is_string()` (*owlapy.converter.OWLLiteral method*), 76
`is_string()` (*owlapy.owl_axiom.OWLLiteral method*), 98
`is_string()` (*owlapy.owl_literal.OWLLiteral method*), 141
`is_string()` (*owlapy.owl_ontology.OWLLiteral method*), 152
`is_string()` (*owlapy.owl_reasoner.OWLLiteral method*), 231
`is_string()` (*owlapy.parser.OWLLiteral method*), 263
`is_string()` (*owlapy.providers.OWLLiteral method*), 285
`is_string()` (*owlapy.render.OWLLiteral method*), 292
`is_string()` (*owlapy.utils.OWLLiteral method*), 314
`is_sub_property_of()` (*owlapy.owl_hierarchy.DatatypePropertyHierarchy method*), 133
`is_sub_property_of()` (*owlapy.owl_hierarchy.ObjectPropertyHierarchy method*), 133
`is_subclass_of()` (*owlapy.owl_hierarchy.ClassHierarchy method*), 132
`is_thing()` (*owlapy.class_expression.owl_class.IRI method*), 25
`is_thing()` (*owlapy.iri.IRI method*), 83
`is_thing()` (*owlapy.owl_axiom.IRI method*), 96
`is_thing()` (*owlapy.owl_datatype.IRI method*), 119
`is_thing()` (*owlapy.owl_individual.IRI method*), 135
`is_thing()` (*owlapy.owl_ontology_manager.IRI method*), 176
`is_thing()` (*owlapy.owl_ontology.IRI method*), 154
`is_thing()` (*owlapy.owl_property.IRI method*), 204
`is_thing()` (*owlapy.owl_reasoner.IRI method*), 219

- `is_thing()` (*owlapy.owlapi_adaptor.IRI method*), 255
- `is_thing()` (*owlapy.parser.IRI method*), 261
- `is_thing()` (*owlapy.render.IRI method*), 290
- `is_thing()` (*owlapy.vocab.IRI method*), 336
- `is_using_triplestore()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 247
- `items` (*owlapy.utils.EvaluatedDescriptionSet attribute*), 332
- `items()` (*owlapy.owl_hierarchy.AbstractHierarchy method*), 131
- `iter_count()` (*in module owlapy.utils*), 334

K

- `KEY` (*owlapy.owl_reasoner.LRUCache attribute*), 232
- `KEY` (*owlapy.utils.LRUCache attribute*), 334

L

- `leaves()` (*owlapy.owl_hierarchy.AbstractHierarchy method*), 132
- `LENGTH` (*owlapy.class_expression.OWLFacet attribute*), 57
- `LENGTH` (*owlapy.class_expression.restriction.OWLFacet attribute*), 35
- `LENGTH` (*owlapy.converter.OWLFacet attribute*), 79
- `LENGTH` (*owlapy.owl_ontology.OWLFacet attribute*), 168
- `LENGTH` (*owlapy.parser.OWLFacet attribute*), 266
- `LENGTH` (*owlapy.providers.OWLFacet attribute*), 287
- `LENGTH` (*owlapy.render.OWLFacet attribute*), 304
- `LENGTH` (*owlapy.vocab.OWLFacet attribute*), 338
- `length()` (*owlapy.utils.OWLClassExpressionLengthMetric method*), 332
- `Literals` (*in module owlapy.class_expression.restriction*), 35
- `Literals` (*in module owlapy.owl_literal*), 140
- `load_ontology()` (*owlapy.owl_ontology_manager.OntologyManager method*), 201
- `load_ontology()` (*owlapy.owl_ontology_manager.OWLOntologyManager method*), 199
- `load_ontology()` (*owlapy.owl_reasoner.OntologyManager method*), 226
- `logger` (*in module owlapy.owl_ontology*), 168
- `logger` (*in module owlapy.owl_reasoner*), 233
- `LONG` (*owlapy.owl_literal.XSDVocabulary attribute*), 139
- `LONG` (*owlapy.vocab.XSDVocabulary attribute*), 338
- `LRUCache` (*class in owlapy.owl_reasoner*), 232
- `LRUCache` (*class in owlapy.utils*), 334

M

- `MANCHESTER_GRAMMAR` (*in module owlapy.parser*), 279
- `manchester_to_owl_expression()` (*in module owlapy*), 339
- `manchester_to_owl_expression()` (*in module owlapy.owlapi_adaptor*), 254
- `manchester_to_owl_expression()` (*in module owlapy.parser*), 283
- `ManchesterOWLSyntaxOWLObjectRenderer` (*class in owlapy.render*), 308
- `ManchesterOWLSyntaxParser` (*class in owlapy.parser*), 279
- `ManchesterParser` (*in module owlapy.parser*), 283
- `ManchesterRenderer` (*in module owlapy.render*), 309
- `map_concept()` (*owlapy.owl_ontology_manager.ToOwlready2 method*), 197
- `map_concept()` (*owlapy.owl_ontology.FromOwlready2 method*), 173
- `map_concept()` (*owlapy.owl_ontology.ToOwlready2 method*), 173
- `map_concept()` (*owlapy.owl_reasoner.ToOwlready2 method*), 226
- `map_datarange()` (*owlapy.owl_ontology_manager.ToOwlready2 method*), 197
- `map_datarange()` (*owlapy.owl_ontology.FromOwlready2 method*), 173
- `map_datarange()` (*owlapy.owl_ontology.ToOwlready2 method*), 173
- `map_datarange()` (*owlapy.owl_reasoner.ToOwlready2 method*), 226
- `map_object()` (*owlapy.owl_ontology_manager.ToOwlready2 method*), 197
- `map_object()` (*owlapy.owl_ontology.ToOwlready2 method*), 173
- `map_object()` (*owlapy.owl_reasoner.ToOwlready2 method*), 226
- `mapping` (*owlapy.converter.Owl2SparqlConverter attribute*), 80
- `MAX_EXCLUSIVE` (*owlapy.class_expression.OWLFacet attribute*), 57
- `MAX_EXCLUSIVE` (*owlapy.class_expression.restriction.OWLFacet attribute*), 35
- `MAX_EXCLUSIVE` (*owlapy.converter.OWLFacet attribute*), 79
- `MAX_EXCLUSIVE` (*owlapy.owl_ontology.OWLFacet attribute*), 168
- `MAX_EXCLUSIVE` (*owlapy.parser.OWLFacet attribute*), 266
- `MAX_EXCLUSIVE` (*owlapy.providers.OWLFacet attribute*), 287
- `MAX_EXCLUSIVE` (*owlapy.render.OWLFacet attribute*), 304
- `MAX_EXCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 338
- `MAX_INCLUSIVE` (*owlapy.class_expression.OWLFacet attribute*), 57

MAX_INCLUSIVE (*owlapy.class_expression.restriction.OWLFacet attribute*), 35
 MAX_INCLUSIVE (*owlapy.converter.OWLFacet attribute*), 79
 MAX_INCLUSIVE (*owlapy.owl_ontology.OWLFacet attribute*), 168
 MAX_INCLUSIVE (*owlapy.parser.OWLFacet attribute*), 266
 MAX_INCLUSIVE (*owlapy.providers.OWLFacet attribute*), 286
 MAX_INCLUSIVE (*owlapy.render.OWLFacet attribute*), 304
 MAX_INCLUSIVE (*owlapy.vocab.OWLFacet attribute*), 338
 MAX_LENGTH (*owlapy.class_expression.OWLFacet attribute*), 57
 MAX_LENGTH (*owlapy.class_expression.restriction.OWLFacet attribute*), 35
 MAX_LENGTH (*owlapy.converter.OWLFacet attribute*), 79
 MAX_LENGTH (*owlapy.owl_ontology.OWLFacet attribute*), 168
 MAX_LENGTH (*owlapy.parser.OWLFacet attribute*), 266
 MAX_LENGTH (*owlapy.providers.OWLFacet attribute*), 287
 MAX_LENGTH (*owlapy.render.OWLFacet attribute*), 304
 MAX_LENGTH (*owlapy.vocab.OWLFacet attribute*), 338
 maybe_add() (*owlapy.utils.EvaluatedDescriptionSet method*), 332
 measurer (*in module owlapy.utils*), 332
 MIN_EXCLUSIVE (*owlapy.class_expression.OWLFacet attribute*), 57
 MIN_EXCLUSIVE (*owlapy.class_expression.restriction.OWLFacet attribute*), 35
 MIN_EXCLUSIVE (*owlapy.converter.OWLFacet attribute*), 79
 MIN_EXCLUSIVE (*owlapy.owl_ontology.OWLFacet attribute*), 168
 MIN_EXCLUSIVE (*owlapy.parser.OWLFacet attribute*), 266
 MIN_EXCLUSIVE (*owlapy.providers.OWLFacet attribute*), 286
 MIN_EXCLUSIVE (*owlapy.render.OWLFacet attribute*), 304
 MIN_EXCLUSIVE (*owlapy.vocab.OWLFacet attribute*), 338
 MIN_INCLUSIVE (*owlapy.class_expression.OWLFacet attribute*), 57
 MIN_INCLUSIVE (*owlapy.class_expression.restriction.OWLFacet attribute*), 35
 MIN_INCLUSIVE (*owlapy.converter.OWLFacet attribute*), 78
 MIN_INCLUSIVE (*owlapy.owl_ontology.OWLFacet attribute*), 168
 MIN_INCLUSIVE (*owlapy.parser.OWLFacet attribute*), 266
 MIN_INCLUSIVE (*owlapy.providers.OWLFacet attribute*), 286
 MIN_INCLUSIVE (*owlapy.render.OWLFacet attribute*), 304
 MIN_INCLUSIVE (*owlapy.vocab.OWLFacet attribute*), 338
 MIN_LENGTH (*owlapy.class_expression.OWLFacet attribute*), 57
 MIN_LENGTH (*owlapy.class_expression.restriction.OWLFacet attribute*), 35
 MIN_LENGTH (*owlapy.converter.OWLFacet attribute*), 79
 MIN_LENGTH (*owlapy.owl_ontology.OWLFacet attribute*), 168
 MIN_LENGTH (*owlapy.parser.OWLFacet attribute*), 266
 MIN_LENGTH (*owlapy.providers.OWLFacet attribute*), 287
 MIN_LENGTH (*owlapy.render.OWLFacet attribute*), 304
 MIN_LENGTH (*owlapy.vocab.OWLFacet attribute*), 338
 modal_depth (*owlapy.converter.Owl2SparqlConverter property*), 80
 module
 owlapy, 18
 owlapy.class_expression, 18
 owlapy.class_expression.class_expression, 18
 owlapy.class_expression.nary_boolean_expression, 21
 owlapy.class_expression.owl_class, 23
 owlapy.class_expression.restriction, 27
 owlapy.converter, 63
 owlapy.entities, 63
 owlapy.iri, 81
 owlapy.meta_classes, 84
 owlapy.namespaces, 85
 owlapy.owl_annotation, 86
 owlapy.owl_axiom, 87
 owlapy.owl_data_ranges, 116
 owlapy.owl_datatype, 118
 owlapy.owl_hierarchy, 121
 owlapy.owl_individual, 133
 owlapy.owl_literal, 136
 owlapy.owl_object, 142
 owlapy.owl_ontology, 145
 owlapy.owl_ontology_manager, 174
 owlapy.owl_property, 202
 owlapy.owl_reasoner, 208
 owlapy.owlapi_adaptor, 254
 owlapy.parser, 257

- owlapy.providers, 283
- owlapy.render, 288
- owlapy.static_funcs, 309
- owlapy.utils, 310
- owlapy.vocab, 335
- more_general_roles() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 133
- more_general_roles() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 132
- more_special_roles() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 133
- more_special_roles() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 133
- most_general_roles() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 133
- most_general_roles() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 133
- most_special_roles() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 133
- most_special_roles() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 133
- move() (in module owlapy.static_funcs), 309

N

- named_classes() (owlapy.owl_axiom.OWLEquivalentClassesAxiom method), 102
- named_classes() (owlapy.owl_ontology_manager.OWLEquivalentClassesAxiom method), 183
- named_classes() (owlapy.owl_ontology.OWLEquivalentClassesAxiom method), 148
- named_individuals (owlapy.converter.Owl2SparqlConverter attribute), 80
- Namespaces (class in owlapy.iri), 82
- Namespaces (class in owlapy.namespaces), 86
- Namespaces (class in owlapy.parser), 265
- Namespaces (class in owlapy.vocab), 337
- new_count_var() (owlapy.converter.Owl2SparqlConverter method), 81
- new_individual_variable() (owlapy.converter.VariablesMapping method), 79
- new_property_variable() (owlapy.converter.VariablesMapping method), 79
- NEXT (owlapy.owl_reasoner.LRUCache attribute), 232
- NEXT (owlapy.utils.LRUCache attribute), 334
- NNF (class in owlapy.utils), 333
- ns (owlapy.iri.Namespaces property), 82
- ns (owlapy.namespaces.Namespaces property), 86
- ns (owlapy.parser.DLSyntaxParser attribute), 281
- ns (owlapy.parser.ManchesterOWLSyntaxParser attribute), 279
- ns (owlapy.parser.Namespaces property), 265
- ns (owlapy.vocab.Namespaces property), 337
- NUMERIC_DATATYPES (in module owlapy.owl_literal), 142

O

- o (owlapy.utils.OrderedOWLObject attribute), 333
- object_all_values_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 331
- object_cardinality_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 331
- object_complement_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 331
- object_has_self_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 331
- object_has_value_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 331
- object_intersection_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 331
- object_inverse_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 332
- object_one_of_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 331
- object_properties_in_signature() (owlapy.owl_ontology_manager.Ontology method), 195
- object_properties_in_signature() (owlapy.owl_ontology_manager.OWLOntology method), 193
- object_properties_in_signature() (owlapy.owl_ontology.Ontology method), 171
- object_properties_in_signature() (owlapy.owl_ontology.OWLOntology method), 169
- object_properties_in_signature() (owlapy.owl_reasoner.Ontology method), 224
- object_properties_in_signature() (owlapy.owl_reasoner.OWLOntology method), 222
- object_property_domain_axioms() (owlapy.owl_ontology_manager.Ontology method), 196
- object_property_domain_axioms() (owlapy.owl_ontology_manager.OWLOntology method), 194
- object_property_domain_axioms() (owlapy.owl_ontology.Ontology method), 172
- object_property_domain_axioms() (owlapy.owl_ontology.OWLOntology method), 170
- object_property_domain_axioms() (owlapy.owl_reasoner.Ontology method), 225
- object_property_domain_axioms() (owlapy.owl_reasoner.OWLOntology method), 223
- object_property_domains() (owlapy.owl_hierarchy.OWLReasoner method), 125
- object_property_domains() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 247
- object_property_domains() (owlapy.owl_reasoner.OntologyReasoner method), 241
- object_property_domains() (owlapy.owl_reasoner.OWLReasoner method), 233
- object_property_length (owlapy.utils.OWLClassExpressionLengthMetric attribute), 332
- object_property_range_axioms() (owlapy.owl_ontology_manager.Ontology method), 196
- object_property_range_axioms() (owlapy.owl_ontology_manager.OWLOntology method), 194

`object_property_range_axioms()` (*owlapy.owl_ontology.Ontology method*), 172
`object_property_range_axioms()` (*owlapy.owl_ontology.OWLOntology method*), 170
`object_property_range_axioms()` (*owlapy.owl_reasoner.Ontology method*), 225
`object_property_range_axioms()` (*owlapy.owl_reasoner.OWLOntology method*), 223
`object_property_ranges()` (*owlapy.owl_hierarchy.OWLReasoner method*), 125
`object_property_ranges()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 248
`object_property_ranges()` (*owlapy.owl_reasoner.OntologyReasoner method*), 241
`object_property_ranges()` (*owlapy.owl_reasoner.OWLReasoner method*), 234
`object_property_values()` (*owlapy.owl_hierarchy.OWLReasoner method*), 127
`object_property_values()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 249
`object_property_values()` (*owlapy.owl_reasoner.OntologyReasoner method*), 243
`object_property_values()` (*owlapy.owl_reasoner.OWLReasoner method*), 235
`object_some_values_length` (*owlapy.utils.OWLClassExpressionLengthMetric attribute*), 331
`object_union_length` (*owlapy.utils.OWLClassExpressionLengthMetric attribute*), 331
`ObjectPropertyHierarchy` (class in *owlapy.owl_hierarchy*), 132
`Ontology` (class in *owlapy.owl_ontology*), 171
`Ontology` (class in *owlapy.owl_ontology_manager*), 195
`Ontology` (class in *owlapy.owl_reasoner*), 224
`OntologyManager` (class in *owlapy.owl_ontology_manager*), 201
`OntologyManager` (class in *owlapy.owl_reasoner*), 226
`OntologyReasoner` (class in *owlapy.owl_reasoner*), 240
`operands()` (*owlapy.class_expression.class_expression.HasOperands method*), 18
`operands()` (*owlapy.class_expression.class_expression.OWLObjectComplementOf method*), 20
`operands()` (*owlapy.class_expression.nary_boolean_expression.HasOperands method*), 22
`operands()` (*owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression method*), 22
`operands()` (*owlapy.class_expression.owl_class.OWLObjectComplementOf method*), 24
`operands()` (*owlapy.class_expression.OWLDataOneOf method*), 54
`operands()` (*owlapy.class_expression.OWLNaryBooleanClassExpression method*), 50
`operands()` (*owlapy.class_expression.OWLObjectComplementOf method*), 49
`operands()` (*owlapy.class_expression.OWLObjectOneOf method*), 61
`operands()` (*owlapy.class_expression.restriction.HasOperands method*), 29
`operands()` (*owlapy.class_expression.restriction.OWLDataOneOf method*), 45
`operands()` (*owlapy.class_expression.restriction.OWLObjectOneOf method*), 41
`operands()` (*owlapy.converter.OWLDataOneOf method*), 74
`operands()` (*owlapy.converter.OWLObjectComplementOf method*), 69
`operands()` (*owlapy.converter.OWLObjectOneOf method*), 65
`operands()` (*owlapy.meta_classes.HasOperands method*), 84
`operands()` (*owlapy.owl_axiom.HasOperands method*), 92
`operands()` (*owlapy.owl_axiom.OWLHasKeyAxiom method*), 101
`operands()` (*owlapy.owl_data_ranges.HasOperands method*), 117
`operands()` (*owlapy.owl_data_ranges.OWLNaryDataRange method*), 117
`operands()` (*owlapy.owl_ontology_manager.OWLNaryBooleanClassExpression method*), 180
`operands()` (*owlapy.owl_ontology_manager.OWLObjectComplementOf method*), 181
`operands()` (*owlapy.owl_ontology_manager.OWLObjectOneOf method*), 180
`operands()` (*owlapy.owl_ontology.OWLDataOneOf method*), 163
`operands()` (*owlapy.owl_ontology.OWLObjectComplementOf method*), 157
`operands()` (*owlapy.owl_ontology.OWLObjectOneOf method*), 158
`operands()` (*owlapy.owl_reasoner.OWLDataOneOf method*), 215
`operands()` (*owlapy.owl_reasoner.OWLObjectComplementOf method*), 212
`operands()` (*owlapy.owl_reasoner.OWLObjectOneOf method*), 213
`operands()` (*owlapy.parser.OWLDataOneOf method*), 275
`operands()` (*owlapy.parser.OWLObjectComplementOf method*), 271
`operands()` (*owlapy.parser.OWLObjectOneOf method*), 270
`operands()` (*owlapy.render.OWLDataOneOf method*), 303
`operands()` (*owlapy.render.OWLNaryBooleanClassExpression method*), 303
`operands()` (*owlapy.render.OWLNaryDataRange method*), 304
`operands()` (*owlapy.render.OWLObjectComplementOf method*), 298
`operands()` (*owlapy.render.OWLObjectOneOf method*), 307
`operands()` (*owlapy.utils.HasOperands method*), 313
`operands()` (*owlapy.utils.OWLDataOneOf method*), 325
`operands()` (*owlapy.utils.OWLNaryBooleanClassExpression method*), 326
`operands()` (*owlapy.utils.OWLNaryDataRange method*), 329
`operands()` (*owlapy.utils.OWLObjectComplementOf method*), 319
`operands()` (*owlapy.utils.OWLObjectOneOf method*), 328
`OperandSetTransform` (class in *owlapy.utils*), 333
`operator` (*owlapy.class_expression.OWLFacet property*), 57
`operator` (*owlapy.class_expression.restriction.OWLFacet property*), 35
`operator` (*owlapy.converter.OWLFacet property*), 78

operator (*owlapy.owl_ontology.OwlFacet property*), 168
 operator (*owlapy.parser.OwlFacet property*), 266
 operator (*owlapy.providers.OwlFacet property*), 286
 operator (*owlapy.render.OwlFacet property*), 304
 operator (*owlapy.vocab.OwlFacet property*), 338
 OrderedOwlObject (*class in owlapy.utils*), 333
 Owl (*in module owlapy.namespaces*), 86
 Owl2SparqlConverter (*class in owlapy.converter*), 80
 Owl_BOTTOM_DATA_PROPERTY (*owlapy.class_expression.OwlRDFVocabulary attribute*), 62
 Owl_BOTTOM_DATA_PROPERTY (*owlapy.converter.OwlRDFVocabulary attribute*), 79
 Owl_BOTTOM_DATA_PROPERTY (*owlapy.owl_literal.OwlRDFVocabulary attribute*), 139
 Owl_BOTTOM_DATA_PROPERTY (*owlapy.parser.OwlRDFVocabulary attribute*), 267
 Owl_BOTTOM_DATA_PROPERTY (*owlapy.vocab.OwlRDFVocabulary attribute*), 337
 Owl_BOTTOM_OBJECT_PROPERTY (*owlapy.class_expression.OwlRDFVocabulary attribute*), 62
 Owl_BOTTOM_OBJECT_PROPERTY (*owlapy.converter.OwlRDFVocabulary attribute*), 79
 Owl_BOTTOM_OBJECT_PROPERTY (*owlapy.owl_literal.OwlRDFVocabulary attribute*), 139
 Owl_BOTTOM_OBJECT_PROPERTY (*owlapy.parser.OwlRDFVocabulary attribute*), 267
 Owl_BOTTOM_OBJECT_PROPERTY (*owlapy.vocab.OwlRDFVocabulary attribute*), 337
 Owl_CLASS (*owlapy.class_expression.OwlRDFVocabulary attribute*), 62
 Owl_CLASS (*owlapy.converter.OwlRDFVocabulary attribute*), 79
 Owl_CLASS (*owlapy.owl_literal.OwlRDFVocabulary attribute*), 138
 Owl_CLASS (*owlapy.parser.OwlRDFVocabulary attribute*), 267
 Owl_CLASS (*owlapy.vocab.OwlRDFVocabulary attribute*), 337
 owl_datatype_max_exclusive_restriction () (*in module owlapy.providers*), 287
 owl_datatype_max_inclusive_restriction () (*in module owlapy.providers*), 287
 owl_datatype_min_exclusive_restriction () (*in module owlapy.providers*), 287
 owl_datatype_min_inclusive_restriction () (*in module owlapy.providers*), 287
 owl_datatype_min_max_exclusive_restriction () (*in module owlapy.providers*), 287
 owl_datatype_min_max_inclusive_restriction () (*in module owlapy.providers*), 288
 owl_expression_to_dl () (*in module owlapy*), 339
 owl_expression_to_dl () (*in module owlapy.render*), 309
 owl_expression_to_manchester () (*in module owlapy*), 339
 owl_expression_to_manchester () (*in module owlapy.owlapi_adaptor*), 256
 owl_expression_to_manchester () (*in module owlapy.render*), 309
 owl_expression_to_sparql () (*in module owlapy*), 339
 owl_expression_to_sparql () (*in module owlapy.converter*), 81
 Owl_NAMED_INDIVIDUAL (*owlapy.class_expression.OwlRDFVocabulary attribute*), 62
 Owl_NAMED_INDIVIDUAL (*owlapy.converter.OwlRDFVocabulary attribute*), 79
 Owl_NAMED_INDIVIDUAL (*owlapy.owl_literal.OwlRDFVocabulary attribute*), 138
 Owl_NAMED_INDIVIDUAL (*owlapy.parser.OwlRDFVocabulary attribute*), 267
 Owl_NAMED_INDIVIDUAL (*owlapy.vocab.OwlRDFVocabulary attribute*), 337
 Owl_NOTHING (*owlapy.class_expression.OwlRDFVocabulary attribute*), 62
 Owl_NOTHING (*owlapy.converter.OwlRDFVocabulary attribute*), 79
 Owl_NOTHING (*owlapy.owl_literal.OwlRDFVocabulary attribute*), 138
 Owl_NOTHING (*owlapy.parser.OwlRDFVocabulary attribute*), 267
 Owl_NOTHING (*owlapy.vocab.OwlRDFVocabulary attribute*), 337
 Owl_THING (*owlapy.class_expression.OwlRDFVocabulary attribute*), 62
 Owl_THING (*owlapy.converter.OwlRDFVocabulary attribute*), 79
 Owl_THING (*owlapy.owl_literal.OwlRDFVocabulary attribute*), 138
 Owl_THING (*owlapy.parser.OwlRDFVocabulary attribute*), 267
 Owl_THING (*owlapy.vocab.OwlRDFVocabulary attribute*), 337
 Owl_TOP_DATA_PROPERTY (*owlapy.class_expression.OwlRDFVocabulary attribute*), 62
 Owl_TOP_DATA_PROPERTY (*owlapy.converter.OwlRDFVocabulary attribute*), 79
 Owl_TOP_DATA_PROPERTY (*owlapy.owl_literal.OwlRDFVocabulary attribute*), 139
 Owl_TOP_DATA_PROPERTY (*owlapy.parser.OwlRDFVocabulary attribute*), 267
 Owl_TOP_DATA_PROPERTY (*owlapy.vocab.OwlRDFVocabulary attribute*), 337
 Owl_TOP_OBJECT_PROPERTY (*owlapy.class_expression.OwlRDFVocabulary attribute*), 62
 Owl_TOP_OBJECT_PROPERTY (*owlapy.converter.OwlRDFVocabulary attribute*), 79
 Owl_TOP_OBJECT_PROPERTY (*owlapy.owl_literal.OwlRDFVocabulary attribute*), 138
 Owl_TOP_OBJECT_PROPERTY (*owlapy.parser.OwlRDFVocabulary attribute*), 267
 Owl_TOP_OBJECT_PROPERTY (*owlapy.vocab.OwlRDFVocabulary attribute*), 337
 OwlAnnotation (*class in owlapy.owl_axiom*), 107
 OwlAnnotationAssertionAxiom (*class in owlapy.owl_axiom*), 108
 OwlAnnotationAssertionAxiom (*class in owlapy.owl_ontology_manager*), 184
 OwlAnnotationAxiom (*class in owlapy.owl_axiom*), 107
 OwlAnnotationObject (*class in owlapy.owl_annotation*), 87
 OwlAnnotationProperty (*class in owlapy.owl_axiom*), 107
 OwlAnnotationProperty (*class in owlapy.owl_ontology*), 149

- OWLAnnotationProperty (class in owlapy.owl_ontology_manager), 184
- OWLAnnotationPropertyDomainAxiom (class in owlapy.owl_axiom), 109
- OWLAnnotationPropertyRangeAxiom (class in owlapy.owl_axiom), 109
- OWLAnnotationSubject (class in owlapy.iri), 82
- OWLAnnotationSubject (class in owlapy.owl_annotation), 87
- OWLAnnotationSubject (class in owlapy.owl_axiom), 96
- OWLAnnotationValue (class in owlapy.iri), 82
- OWLAnnotationValue (class in owlapy.owl_annotation), 87
- OWLAnnotationValue (class in owlapy.owl_axiom), 96
- OWLAnnotationValue (class in owlapy.owl_literal), 138
- OWLAnonymousClassExpression (class in owlapy.class_expression), 48
- OWLAnonymousClassExpression (class in owlapy.class_expression.class_expression), 19
- OWLAnonymousClassExpression (class in owlapy.class_expression.restriction), 30
- OWLAPIAdaptor (class in owlapy.owlapi_adaptor), 256
- owlapy
 - module, 18
- owlapy.class_expression
 - module, 18
- owlapy.class_expression.class_expression
 - module, 18
- owlapy.class_expression.nary_boolean_expression
 - module, 21
- owlapy.class_expression.owl_class
 - module, 23
- owlapy.class_expression.restriction
 - module, 27
- owlapy.converter
 - module, 63
- owlapy.entities
 - module, 63
- owlapy.iri
 - module, 81
- owlapy.meta_classes
 - module, 84
- owlapy.namespaces
 - module, 85
- owlapy.owl_annotation
 - module, 86
- owlapy.owl_axiom
 - module, 87
- owlapy.owl_data_ranges
 - module, 116
- owlapy.owl_datatype
 - module, 118
- owlapy.owl_hierarchy
 - module, 121
- owlapy.owl_individual
 - module, 133
- owlapy.owl_literal
 - module, 136
- owlapy.owl_object
 - module, 142
- owlapy.owl_ontology
 - module, 145
- owlapy.owl_ontology_manager
 - module, 174
- owlapy.owl_property
 - module, 202
- owlapy.owl_reasoner
 - module, 208
- owlapy.owlapi_adaptor
 - module, 254
- owlapy.parser
 - module, 257
- owlapy.providers
 - module, 283
- owlapy.render
 - module, 288

- owlapy.static_funcs
 - module, 309
- owlapy.utils
 - module, 310
- owlapy.vocab
 - module, 335
- OWLAsymmetricObjectPropertyAxiom (class in owlapy.owl_axiom), 112
- OWLAsymmetricObjectPropertyAxiom (class in owlapy.owl_ontology_manager), 187
- OWLAxiom (class in owlapy.owl_axiom), 99
- OWLAxiom (class in owlapy.owl_ontology_manager), 182
- OWLAxiom (class in owlapy.owl_reasoner), 220
- OWLBooleanClassExpression (class in owlapy.class_expression), 49
- OWLBooleanClassExpression (class in owlapy.class_expression.class_expression), 20
- OWLBooleanClassExpression (class in owlapy.class_expression.nary_boolean_expression), 22
- OWLBooleanClassExpression (class in owlapy.render), 296
- OWLBottomDataProperty (in module owlapy.owl_hierarchy), 123
- OWLBottomDataProperty (in module owlapy.owl_literal), 142
- OWLBottomObjectProperty (in module owlapy.owl_hierarchy), 123
- OWLBottomObjectProperty (in module owlapy.owl_literal), 142
- OWLCardinalityRestriction (class in owlapy.class_expression), 53
- OWLCardinalityRestriction (class in owlapy.class_expression.restriction), 36
- OWLClass (class in owlapy.class_expression), 49
- OWLClass (class in owlapy.class_expression.owl_class), 26
- OWLClass (class in owlapy.converter), 67
- OWLClass (class in owlapy.owl_axiom), 94
- OWLClass (class in owlapy.owl_hierarchy), 122
- OWLClass (class in owlapy.owl_ontology), 155
- OWLClass (class in owlapy.owl_ontology_manager), 178
- OWLClass (class in owlapy.owl_reasoner), 218
- OWLClass (class in owlapy.parser), 268
- OWLClass (class in owlapy.render), 296
- OWLClass (class in owlapy.utils), 318
- OWLClassAssertionAxiom (class in owlapy.owl_axiom), 106
- OWLClassAssertionAxiom (class in owlapy.owl_ontology_manager), 185
- OWLClassAxiom (class in owlapy.owl_axiom), 100
- OWLClassAxiom (class in owlapy.owl_ontology), 148
- OWLClassExpression (class in owlapy.class_expression), 48
- OWLClassExpression (class in owlapy.class_expression.class_expression), 19
- OWLClassExpression (class in owlapy.class_expression.nary_boolean_expression), 21
- OWLClassExpression (class in owlapy.class_expression.owl_class), 23
- OWLClassExpression (class in owlapy.class_expression.restriction), 30
- OWLClassExpression (class in owlapy.converter), 68
- OWLClassExpression (class in owlapy.owl_axiom), 93
- OWLClassExpression (class in owlapy.owl_ontology), 156
- OWLClassExpression (class in owlapy.owl_reasoner), 210
- OWLClassExpression (class in owlapy.owlapi_adaptor), 254
- OWLClassExpression (class in owlapy.parser), 270
- OWLClassExpression (class in owlapy.render), 295
- OWLClassExpression (class in owlapy.utils), 317
- OWLClassExpressionLengthMetric (class in owlapy.utils), 330
- OWLDataAllValuesFrom (class in owlapy.class_expression), 59
- OWLDataAllValuesFrom (class in owlapy.class_expression.restriction), 43
- OWLDataAllValuesFrom (class in owlapy.converter), 73
- OWLDataAllValuesFrom (class in owlapy.owl_ontology), 161
- OWLDataAllValuesFrom (class in owlapy.owl_reasoner), 217
- OWLDataAllValuesFrom (class in owlapy.parser), 277
- OWLDataAllValuesFrom (class in owlapy.render), 300
- OWLDataAllValuesFrom (class in owlapy.utils), 323
- OWLDataCardinalityRestriction (class in owlapy.class_expression), 55
- OWLDataCardinalityRestriction (class in owlapy.class_expression.restriction), 41
- OWLDataCardinalityRestriction (class in owlapy.converter), 72
- OWLDataCardinalityRestriction (class in owlapy.parser), 276
- OWLDataCardinalityRestriction (class in owlapy.utils), 325
- OWLDataComplementOf (class in owlapy.owl_data_ranges), 118
- OWLDataComplementOf (class in owlapy.owl_ontology), 150
- OWLDataComplementOf (class in owlapy.owl_reasoner), 221
- OWLDataComplementOf (class in owlapy.parser), 278
- OWLDataComplementOf (class in owlapy.render), 305

OWLDataComplementOf (class in owlapy.utils), 328
 OWLDataExactCardinality (class in owlapy.class_expression), 61
 OWLDataExactCardinality (class in owlapy.class_expression.restriction), 42
 OWLDataExactCardinality (class in owlapy.converter), 67
 OWLDataExactCardinality (class in owlapy.owl_ontology), 161
 OWLDataExactCardinality (class in owlapy.parser), 273
 OWLDataExactCardinality (class in owlapy.render), 302
 OWLDataExactCardinality (class in owlapy.utils), 322
 OWLDataHasValue (class in owlapy.class_expression), 60
 OWLDataHasValue (class in owlapy.class_expression.restriction), 44
 OWLDataHasValue (class in owlapy.converter), 74
 OWLDataHasValue (class in owlapy.owl_ontology), 162
 OWLDataHasValue (class in owlapy.owl_ontology_manager), 179
 OWLDataHasValue (class in owlapy.owl_reasoner), 217
 OWLDataHasValue (class in owlapy.parser), 275
 OWLDataHasValue (class in owlapy.render), 301
 OWLDataHasValue (class in owlapy.utils), 322
 OWLDataIntersectionOf (class in owlapy.owl_data_ranges), 117
 OWLDataIntersectionOf (class in owlapy.owl_ontology), 150
 OWLDataIntersectionOf (class in owlapy.owl_reasoner), 221
 OWLDataIntersectionOf (class in owlapy.parser), 278
 OWLDataIntersectionOf (class in owlapy.render), 305
 OWLDataIntersectionOf (class in owlapy.utils), 329
 OWLDataMaxCardinality (class in owlapy.class_expression), 61
 OWLDataMaxCardinality (class in owlapy.class_expression.restriction), 42
 OWLDataMaxCardinality (class in owlapy.converter), 67
 OWLDataMaxCardinality (class in owlapy.owl_ontology), 162
 OWLDataMaxCardinality (class in owlapy.parser), 274
 OWLDataMaxCardinality (class in owlapy.render), 302
 OWLDataMaxCardinality (class in owlapy.utils), 322
 OWLDataMinCardinality (class in owlapy.class_expression), 60
 OWLDataMinCardinality (class in owlapy.class_expression.restriction), 42
 OWLDataMinCardinality (class in owlapy.converter), 66
 OWLDataMinCardinality (class in owlapy.owl_ontology), 162
 OWLDataMinCardinality (class in owlapy.parser), 274
 OWLDataMinCardinality (class in owlapy.render), 302
 OWLDataMinCardinality (class in owlapy.utils), 322
 OWLDataOneOf (class in owlapy.class_expression), 54
 OWLDataOneOf (class in owlapy.class_expression.restriction), 44
 OWLDataOneOf (class in owlapy.converter), 74
 OWLDataOneOf (class in owlapy.owl_ontology), 163
 OWLDataOneOf (class in owlapy.owl_reasoner), 215
 OWLDataOneOf (class in owlapy.parser), 275
 OWLDataOneOf (class in owlapy.render), 302
 OWLDataOneOf (class in owlapy.utils), 324
 OWLDataProperty (class in owlapy.converter), 77
 OWLDataProperty (class in owlapy.owl_hierarchy), 124
 OWLDataProperty (class in owlapy.owl_literal), 140
 OWLDataProperty (class in owlapy.owl_ontology), 165
 OWLDataProperty (class in owlapy.owl_ontology_manager), 197
 OWLDataProperty (class in owlapy.owl_property), 207
 OWLDataProperty (class in owlapy.owl_reasoner), 228
 OWLDataProperty (class in owlapy.parser), 265
 OWLDataProperty (class in owlapy.utils), 317
 OWLDataPropertyAssertionAxiom (class in owlapy.owl_axiom), 111
 OWLDataPropertyAssertionAxiom (class in owlapy.owl_ontology_manager), 188
 OWLDataPropertyAxiom (class in owlapy.owl_axiom), 99
 OWLDataPropertyCharacteristicAxiom (class in owlapy.owl_axiom), 114
 OWLDataPropertyCharacteristicAxiom (class in owlapy.owl_ontology_manager), 188
 OWLDataPropertyDomainAxiom (class in owlapy.owl_axiom), 115
 OWLDataPropertyDomainAxiom (class in owlapy.owl_ontology), 148
 OWLDataPropertyExpression (class in owlapy.class_expression.restriction), 32
 OWLDataPropertyExpression (class in owlapy.owl_axiom), 91
 OWLDataPropertyExpression (class in owlapy.owl_ontology), 167
 OWLDataPropertyExpression (class in owlapy.owl_property), 206
 OWLDataPropertyExpression (class in owlapy.owl_reasoner), 230
 OWLDataPropertyRangeAxiom (class in owlapy.owl_axiom), 116
 OWLDataPropertyRangeAxiom (class in owlapy.owl_ontology), 148

OWLDataRange (class in owlapy.class_expression.restriction), 32
 OWLDataRange (class in owlapy.owl_axiom), 92
 OWLDataRange (class in owlapy.owl_data_ranges), 117
 OWLDataRange (class in owlapy.owl_datatype), 119
 OWLDataRange (class in owlapy.owl_ontology), 150
 OWLDataRange (class in owlapy.owl_reasoner), 221
 OWLDataRange (class in owlapy.parser), 279
 OWLDataRange (class in owlapy.utils), 329
 OWLDataRestriction (class in owlapy.class_expression), 52
 OWLDataRestriction (class in owlapy.class_expression.restriction), 41
 OWLDataRestriction (class in owlapy.owl_ontology), 164
 OWLDataSomeValuesFrom (class in owlapy.class_expression), 59
 OWLDataSomeValuesFrom (class in owlapy.class_expression.restriction), 43
 OWLDataSomeValuesFrom (class in owlapy.converter), 72
 OWLDataSomeValuesFrom (class in owlapy.owl_ontology), 160
 OWLDataSomeValuesFrom (class in owlapy.owl_reasoner), 215
 OWLDataSomeValuesFrom (class in owlapy.parser), 273
 OWLDataSomeValuesFrom (class in owlapy.render), 300
 OWLDataSomeValuesFrom (class in owlapy.utils), 324
 OWLDatatype (class in owlapy.class_expression.restriction), 34
 OWLDatatype (class in owlapy.converter), 78
 OWLDatatype (class in owlapy.owl_axiom), 92
 OWLDatatype (class in owlapy.owl_datatype), 121
 OWLDatatype (class in owlapy.owl_literal), 138
 OWLDatatype (class in owlapy.owl_ontology), 150
 OWLDatatype (class in owlapy.owl_reasoner), 221
 OWLDatatype (class in owlapy.parser), 266
 OWLDatatype (class in owlapy.render), 307
 OWLDatatype (class in owlapy.utils), 330
 OWLDatatypeDefinitionAxiom (class in owlapy.owl_axiom), 100
 OWLDatatypeRestriction (class in owlapy.class_expression), 57
 OWLDatatypeRestriction (class in owlapy.class_expression.restriction), 45
 OWLDatatypeRestriction (class in owlapy.converter), 66
 OWLDatatypeRestriction (class in owlapy.owl_ontology), 163
 OWLDatatypeRestriction (class in owlapy.owl_reasoner), 216
 OWLDatatypeRestriction (class in owlapy.parser), 276
 OWLDatatypeRestriction (class in owlapy.providers), 286
 OWLDatatypeRestriction (class in owlapy.render), 306
 OWLDatatypeRestriction (class in owlapy.utils), 327
 OWLDataUnionOf (class in owlapy.owl_data_ranges), 118
 OWLDataUnionOf (class in owlapy.owl_ontology), 150
 OWLDataUnionOf (class in owlapy.owl_reasoner), 221
 OWLDataUnionOf (class in owlapy.parser), 278
 OWLDataUnionOf (class in owlapy.render), 305
 OWLDataUnionOf (class in owlapy.utils), 329
 OWLDeclarationAxiom (class in owlapy.owl_axiom), 100
 OWLDeclarationAxiom (class in owlapy.owl_ontology_manager), 186
 OWLDifferentIndividualsAxiom (class in owlapy.owl_axiom), 103
 OWLDifferentIndividualsAxiom (class in owlapy.owl_ontology_manager), 192
 OWLDisjointClassesAxiom (class in owlapy.owl_axiom), 102
 OWLDisjointClassesAxiom (class in owlapy.owl_ontology_manager), 192
 OWLDisjointDataPropertiesAxiom (class in owlapy.owl_axiom), 105
 OWLDisjointDataPropertiesAxiom (class in owlapy.owl_ontology_manager), 189
 OWLDisjointObjectPropertiesAxiom (class in owlapy.owl_axiom), 104
 OWLDisjointObjectPropertiesAxiom (class in owlapy.owl_ontology_manager), 190
 OWLDisjointUnionAxiom (class in owlapy.owl_axiom), 106
 OWLDisjointUnionAxiom (class in owlapy.owl_ontology_manager), 183
 OWLEntity (class in owlapy.class_expression.owl_class), 24
 OWLEntity (class in owlapy.converter), 78
 OWLEntity (class in owlapy.owl_axiom), 91
 OWLEntity (class in owlapy.owl_datatype), 119
 OWLEntity (class in owlapy.owl_individual), 134
 OWLEntity (class in owlapy.owl_object), 144
 OWLEntity (class in owlapy.owl_property), 203
 OWLEntity (class in owlapy.render), 294
 OWLEquivalentClassesAxiom (class in owlapy.owl_axiom), 102
 OWLEquivalentClassesAxiom (class in owlapy.owl_ontology), 147
 OWLEquivalentClassesAxiom (class in owlapy.owl_ontology_manager), 183

OWLEquivalentDataPropertiesAxiom (class in owlapy.owl_axiom), 105
 OWLEquivalentDataPropertiesAxiom (class in owlapy.owl_ontology_manager), 190
 OWLEquivalentObjectPropertiesAxiom (class in owlapy.owl_axiom), 104
 OWLEquivalentObjectPropertiesAxiom (class in owlapy.owl_ontology_manager), 190
 OWLFacet (class in owlapy.class_expression), 57
 OWLFacet (class in owlapy.class_expression.restriction), 35
 OWLFacet (class in owlapy.converter), 78
 OWLFacet (class in owlapy.owl_ontology), 168
 OWLFacet (class in owlapy.parser), 266
 OWLFacet (class in owlapy.providers), 286
 OWLFacet (class in owlapy.render), 304
 OWLFacet (class in owlapy.vocab), 338
 OWLFacetRestriction (class in owlapy.class_expression), 58
 OWLFacetRestriction (class in owlapy.class_expression.restriction), 45
 OWLFacetRestriction (class in owlapy.owl_ontology), 165
 OWLFacetRestriction (class in owlapy.owl_reasoner), 216
 OWLFacetRestriction (class in owlapy.parser), 272
 OWLFacetRestriction (class in owlapy.providers), 287
 OWLFacetRestriction (class in owlapy.render), 306
 OWLFacetRestriction (class in owlapy.utils), 327
 OWLFunctionalDataPropertyAxiom (class in owlapy.owl_axiom), 114
 OWLFunctionalDataPropertyAxiom (class in owlapy.owl_ontology_manager), 188
 OWLFunctionalObjectPropertyAxiom (class in owlapy.owl_axiom), 112
 OWLFunctionalObjectPropertyAxiom (class in owlapy.owl_ontology_manager), 189
 OWLHasKeyAxiom (class in owlapy.owl_axiom), 101
 OWLHasValueRestriction (class in owlapy.class_expression), 52
 OWLHasValueRestriction (class in owlapy.class_expression.restriction), 36
 OWLImportsDeclaration (class in owlapy.owl_ontology_manager), 200
 OWLIndividual (class in owlapy.class_expression.restriction), 34
 OWLIndividual (class in owlapy.owl_axiom), 95
 OWLIndividual (class in owlapy.owl_individual), 135
 OWLIndividual (class in owlapy.owl_ontology), 151
 OWLIndividual (class in owlapy.owl_ontology_manager), 193
 OWLIndividualAxiom (class in owlapy.owl_axiom), 99
 OWLInverseFunctionalObjectPropertyAxiom (class in owlapy.owl_axiom), 112
 OWLInverseFunctionalObjectPropertyAxiom (class in owlapy.owl_ontology_manager), 189
 OWLInverseObjectPropertiesAxiom (class in owlapy.owl_axiom), 104
 OWLInverseObjectPropertiesAxiom (class in owlapy.owl_ontology_manager), 190
 OWLIrreflexiveObjectPropertyAxiom (class in owlapy.owl_axiom), 113
 OWLIrreflexiveObjectPropertyAxiom (class in owlapy.owl_ontology_manager), 189
 OWLLiteral (class in owlapy.class_expression.restriction), 32
 OWLLiteral (class in owlapy.converter), 75
 OWLLiteral (class in owlapy.owl_axiom), 97
 OWLLiteral (class in owlapy.owl_literal), 140
 OWLLiteral (class in owlapy.owl_ontology), 152
 OWLLiteral (class in owlapy.owl_reasoner), 230
 OWLLiteral (class in owlapy.parser), 262
 OWLLiteral (class in owlapy.providers), 284
 OWLLiteral (class in owlapy.render), 291
 OWLLiteral (class in owlapy.utils), 314
 OWLLogicalAxiom (class in owlapy.owl_axiom), 99
 OWLNamedIndividual (class in owlapy.converter), 75
 OWLNamedIndividual (class in owlapy.owl_individual), 136
 OWLNamedIndividual (class in owlapy.owl_ontology), 151
 OWLNamedIndividual (class in owlapy.owl_ontology_manager), 192
 OWLNamedIndividual (class in owlapy.owl_reasoner), 230
 OWLNamedIndividual (class in owlapy.owlapi_adaptor), 256
 OWLNamedIndividual (class in owlapy.parser), 261
 OWLNamedIndividual (class in owlapy.render), 291
 OWLNamedIndividual (class in owlapy.utils), 312
 OWLNamedObject (class in owlapy.owl_object), 144
 OWLNaryAxiom (class in owlapy.owl_axiom), 101
 OWLNaryBooleanClassExpression (class in owlapy.class_expression), 50
 OWLNaryBooleanClassExpression (class in owlapy.class_expression.nary_boolean_expression), 22
 OWLNaryBooleanClassExpression (class in owlapy.owl_ontology_manager), 180
 OWLNaryBooleanClassExpression (class in owlapy.render), 303
 OWLNaryBooleanClassExpression (class in owlapy.utils), 326
 OWLNaryClassAxiom (class in owlapy.owl_axiom), 101

OWLNaryDataRange (class in owlapy.owl_data_ranges), 117
 OWLNaryDataRange (class in owlapy.render), 304
 OWLNaryDataRange (class in owlapy.utils), 329
 OWLNaryIndividualAxiom (class in owlapy.owl_axiom), 103
 OWLNaryIndividualAxiom (class in owlapy.owl_ontology_manager), 191
 OWLNaryPropertyAxiom (class in owlapy.owl_axiom), 103
 OWLNaryPropertyAxiom (class in owlapy.owl_ontology_manager), 191
 OWLNegativeDataPropertyAssertionAxiom (class in owlapy.owl_axiom), 111
 OWLNegativeObjectPropertyAssertionAxiom (class in owlapy.owl_axiom), 111
 OWLNothing (in module owlapy.class_expression), 62
 OWLNothing (in module owlapy.owl_axiom), 95
 OWLNothing (in module owlapy.owl_hierarchy), 123
 OWLNothing (in module owlapy.utils), 319
 OWLObject (class in owlapy.class_expression.restriction), 34
 OWLObject (class in owlapy.owl_annotation), 86
 OWLObject (class in owlapy.owl_axiom), 91
 OWLObject (class in owlapy.owl_data_ranges), 116
 OWLObject (class in owlapy.owl_individual), 134
 OWLObject (class in owlapy.owl_object), 143
 OWLObject (class in owlapy.owl_ontology), 153
 OWLObject (class in owlapy.owl_ontology_manager), 177
 OWLObject (class in owlapy.owl_property), 203
 OWLObject (class in owlapy.render), 294
 OWLObject (class in owlapy.utils), 330
 OWLObjectAllValuesFrom (class in owlapy.class_expression), 56
 OWLObjectAllValuesFrom (class in owlapy.class_expression.restriction), 39
 OWLObjectAllValuesFrom (class in owlapy.converter), 70
 OWLObjectAllValuesFrom (class in owlapy.owl_ontology), 158
 OWLObjectAllValuesFrom (class in owlapy.owl_reasoner), 212
 OWLObjectAllValuesFrom (class in owlapy.parser), 277
 OWLObjectAllValuesFrom (class in owlapy.render), 297
 OWLObjectAllValuesFrom (class in owlapy.utils), 324
 OWLObjectCardinalityRestriction (class in owlapy.class_expression), 53
 OWLObjectCardinalityRestriction (class in owlapy.class_expression.restriction), 37
 OWLObjectCardinalityRestriction (class in owlapy.converter), 70
 OWLObjectCardinalityRestriction (class in owlapy.owl_reasoner), 214
 OWLObjectCardinalityRestriction (class in owlapy.parser), 276
 OWLObjectCardinalityRestriction (class in owlapy.utils), 318
 OWLObjectComplementOf (class in owlapy.class_expression), 49
 OWLObjectComplementOf (class in owlapy.class_expression.class_expression), 20
 OWLObjectComplementOf (class in owlapy.class_expression.owl_class), 24
 OWLObjectComplementOf (class in owlapy.converter), 69
 OWLObjectComplementOf (class in owlapy.owl_ontology), 156
 OWLObjectComplementOf (class in owlapy.owl_ontology_manager), 181
 OWLObjectComplementOf (class in owlapy.owl_reasoner), 211
 OWLObjectComplementOf (class in owlapy.parser), 271
 OWLObjectComplementOf (class in owlapy.render), 298
 OWLObjectComplementOf (class in owlapy.utils), 319
 OWLObjectExactCardinality (class in owlapy.class_expression), 58
 OWLObjectExactCardinality (class in owlapy.class_expression.restriction), 38
 OWLObjectExactCardinality (class in owlapy.converter), 71
 OWLObjectExactCardinality (class in owlapy.owl_ontology), 159
 OWLObjectExactCardinality (class in owlapy.owl_reasoner), 214
 OWLObjectExactCardinality (class in owlapy.parser), 271
 OWLObjectExactCardinality (class in owlapy.render), 299
 OWLObjectExactCardinality (class in owlapy.utils), 321
 OWLObjectHasSelf (class in owlapy.class_expression), 53
 OWLObjectHasSelf (class in owlapy.class_expression.restriction), 39
 OWLObjectHasSelf (class in owlapy.converter), 72
 OWLObjectHasSelf (class in owlapy.parser), 267
 OWLObjectHasSelf (class in owlapy.render), 300
 OWLObjectHasSelf (class in owlapy.utils), 321
 OWLObjectHasValue (class in owlapy.class_expression), 56
 OWLObjectHasValue (class in owlapy.class_expression.restriction), 40
 OWLObjectHasValue (class in owlapy.converter), 65
 OWLObjectHasValue (class in owlapy.owl_ontology), 160
 OWLObjectHasValue (class in owlapy.owl_ontology_manager), 181
 OWLObjectHasValue (class in owlapy.owl_reasoner), 213

OWLObjectHasValue (class in owlapy.parser), 273
 OWLObjectHasValue (class in owlapy.render), 306
 OWLObjectHasValue (class in owlapy.utils), 326
 OWLObjectIntersectionOf (class in owlapy.class_expression), 51
 OWLObjectIntersectionOf (class in owlapy.class_expression.nary_boolean_expression), 22
 OWLObjectIntersectionOf (class in owlapy.class_expression.restriction), 29
 OWLObjectIntersectionOf (class in owlapy.converter), 69
 OWLObjectIntersectionOf (class in owlapy.owl_ontology), 157
 OWLObjectIntersectionOf (class in owlapy.owl_reasoner), 211
 OWLObjectIntersectionOf (class in owlapy.parser), 267
 OWLObjectIntersectionOf (class in owlapy.render), 298
 OWLObjectIntersectionOf (class in owlapy.utils), 325
 OWLObjectInverseOf (class in owlapy.owl_ontology), 166
 OWLObjectInverseOf (class in owlapy.owl_ontology_manager), 197
 OWLObjectInverseOf (class in owlapy.owl_property), 207
 OWLObjectInverseOf (class in owlapy.owl_reasoner), 229
 OWLObjectInverseOf (class in owlapy.render), 294
 OWLObjectInverseOf (class in owlapy.utils), 315
 OWLObjectMaxCardinality (class in owlapy.class_expression), 58
 OWLObjectMaxCardinality (class in owlapy.class_expression.restriction), 38
 OWLObjectMaxCardinality (class in owlapy.converter), 71
 OWLObjectMaxCardinality (class in owlapy.owl_ontology), 159
 OWLObjectMaxCardinality (class in owlapy.owl_reasoner), 214
 OWLObjectMaxCardinality (class in owlapy.parser), 274
 OWLObjectMaxCardinality (class in owlapy.render), 299
 OWLObjectMaxCardinality (class in owlapy.utils), 321
 OWLObjectMinCardinality (class in owlapy.class_expression), 58
 OWLObjectMinCardinality (class in owlapy.class_expression.restriction), 38
 OWLObjectMinCardinality (class in owlapy.converter), 71
 OWLObjectMinCardinality (class in owlapy.owl_ontology), 159
 OWLObjectMinCardinality (class in owlapy.owl_reasoner), 213
 OWLObjectMinCardinality (class in owlapy.parser), 268
 OWLObjectMinCardinality (class in owlapy.render), 299
 OWLObjectMinCardinality (class in owlapy.utils), 320
 OWLObjectOneOf (class in owlapy.class_expression), 61
 OWLObjectOneOf (class in owlapy.class_expression.restriction), 40
 OWLObjectOneOf (class in owlapy.converter), 65
 OWLObjectOneOf (class in owlapy.owl_ontology), 158
 OWLObjectOneOf (class in owlapy.owl_ontology_manager), 180
 OWLObjectOneOf (class in owlapy.owl_reasoner), 212
 OWLObjectOneOf (class in owlapy.parser), 269
 OWLObjectOneOf (class in owlapy.render), 307
 OWLObjectOneOf (class in owlapy.utils), 328
 OWLObjectParser (class in owlapy.owl_object), 144
 OWLObjectParser (class in owlapy.parser), 265
 OWLObjectProperty (class in owlapy.converter), 77
 OWLObjectProperty (class in owlapy.owl_hierarchy), 123
 OWLObjectProperty (class in owlapy.owl_literal), 139
 OWLObjectProperty (class in owlapy.owl_ontology), 165
 OWLObjectProperty (class in owlapy.owl_ontology_manager), 198
 OWLObjectProperty (class in owlapy.owl_property), 206
 OWLObjectProperty (class in owlapy.owl_reasoner), 228
 OWLObjectProperty (class in owlapy.parser), 264
 OWLObjectProperty (class in owlapy.utils), 316
 OWLObjectPropertyAssertionAxiom (class in owlapy.owl_axiom), 111
 OWLObjectPropertyAssertionAxiom (class in owlapy.owl_ontology_manager), 186
 OWLObjectPropertyAssertionAxiom (class in owlapy.owl_axiom), 99
 OWLObjectPropertyCharacteristicAxiom (class in owlapy.owl_axiom), 112
 OWLObjectPropertyCharacteristicAxiom (class in owlapy.owl_ontology_manager), 189
 OWLObjectPropertyDomainAxiom (class in owlapy.owl_axiom), 115
 OWLObjectPropertyDomainAxiom (class in owlapy.owl_ontology), 148
 OWLObjectPropertyExpression (class in owlapy.class_expression.restriction), 31
 OWLObjectPropertyExpression (class in owlapy.owl_axiom), 91
 OWLObjectPropertyExpression (class in owlapy.owl_ontology), 167
 OWLObjectPropertyExpression (class in owlapy.owl_property), 205
 OWLObjectPropertyExpression (class in owlapy.owl_reasoner), 227
 OWLObjectPropertyExpression (class in owlapy.parser), 264
 OWLObjectPropertyRangeAxiom (class in owlapy.owl_axiom), 115

OWLObjectPropertyRangeAxiom (class in owlapy.owl_ontology), 149
 OWLObjectPropertyRangeAxiom (class in owlapy.owl_ontology_manager), 182
 OWLObjectRenderer (class in owlapy.owl_object), 143
 OWLObjectRenderer (class in owlapy.render), 293
 OWLObjectRestriction (class in owlapy.class_expression), 52
 OWLObjectRestriction (class in owlapy.class_expression.restriction), 36
 OWLObjectRestriction (class in owlapy.owl_ontology), 164
 OWLObjectSomeValuesFrom (class in owlapy.class_expression), 55
 OWLObjectSomeValuesFrom (class in owlapy.class_expression.restriction), 38
 OWLObjectSomeValuesFrom (class in owlapy.converter), 69
 OWLObjectSomeValuesFrom (class in owlapy.owl_ontology), 157
 OWLObjectSomeValuesFrom (class in owlapy.owl_reasoner), 210
 OWLObjectSomeValuesFrom (class in owlapy.parser), 268
 OWLObjectSomeValuesFrom (class in owlapy.render), 297
 OWLObjectSomeValuesFrom (class in owlapy.utils), 320
 OWLObjectUnionOf (class in owlapy.class_expression), 50
 OWLObjectUnionOf (class in owlapy.class_expression.nary_boolean_expression), 22
 OWLObjectUnionOf (class in owlapy.class_expression.restriction), 29
 OWLObjectUnionOf (class in owlapy.converter), 69
 OWLObjectUnionOf (class in owlapy.owl_axiom), 95
 OWLObjectUnionOf (class in owlapy.owl_ontology), 157
 OWLObjectUnionOf (class in owlapy.owl_reasoner), 211
 OWLObjectUnionOf (class in owlapy.parser), 268
 OWLObjectUnionOf (class in owlapy.render), 298
 OWLObjectUnionOf (class in owlapy.utils), 326
 OWLOntology (class in owlapy.owl_ontology), 169
 OWLOntology (class in owlapy.owl_ontology_manager), 193
 OWLOntology (class in owlapy.owl_reasoner), 222
 OWLOntologyChange (class in owlapy.owl_ontology_manager), 199
 OWLOntologyID (class in owlapy.owl_ontology), 168
 OWLOntologyManager (class in owlapy.owl_ontology_manager), 199
 OWLProperty (class in owlapy.owl_axiom), 93
 OWLProperty (class in owlapy.owl_ontology_manager), 199
 OWLProperty (class in owlapy.owl_property), 206
 OWLPropertyAssertionAxiom (class in owlapy.owl_axiom), 110
 OWLPropertyAxiom (class in owlapy.owl_axiom), 99
 OWLPropertyDomainAxiom (class in owlapy.owl_axiom), 114
 OWLPropertyDomainAxiom (class in owlapy.owl_ontology_manager), 187
 OWLPropertyExpression (class in owlapy.class_expression.restriction), 31
 OWLPropertyExpression (class in owlapy.owl_axiom), 92
 OWLPropertyExpression (class in owlapy.owl_ontology), 166
 OWLPropertyExpression (class in owlapy.owl_property), 205
 OWLPropertyExpression (class in owlapy.owl_reasoner), 229
 OWLPropertyExpression (class in owlapy.render), 295
 OWLPropertyRange (class in owlapy.class_expression.class_expression), 18
 OWLPropertyRange (class in owlapy.class_expression.restriction), 32
 OWLPropertyRange (class in owlapy.owl_data_ranges), 117
 OWLPropertyRange (class in owlapy.utils), 329
 OWLPropertyRangeAxiom (class in owlapy.owl_axiom), 114
 OWLPropertyRangeAxiom (class in owlapy.owl_ontology_manager), 185
 OWLQuantifiedDataRestriction (class in owlapy.class_expression), 54
 OWLQuantifiedDataRestriction (class in owlapy.class_expression.restriction), 41
 OWLQuantifiedDataRestriction (class in owlapy.owl_ontology_manager), 179
 OWLQuantifiedDataRestriction (class in owlapy.parser), 272
 OWLQuantifiedObjectRestriction (class in owlapy.class_expression), 51
 OWLQuantifiedObjectRestriction (class in owlapy.class_expression.restriction), 37
 OWLQuantifiedObjectRestriction (class in owlapy.owl_ontology_manager), 182
 OWLQuantifiedObjectRestriction (class in owlapy.parser), 272
 OWLQuantifiedRestriction (class in owlapy.class_expression), 51
 OWLQuantifiedRestriction (class in owlapy.class_expression.restriction), 36
 OWLRDFVocabulary (class in owlapy.class_expression), 62
 OWLRDFVocabulary (class in owlapy.converter), 79
 OWLRDFVocabulary (class in owlapy.owl_literal), 138
 OWLRDFVocabulary (class in owlapy.parser), 267
 OWLRDFVocabulary (class in owlapy.vocab), 337
 OWLREADY2_FACET_KEYS (in module owlapy.owl_ontology), 173
 OWLReasoner (class in owlapy.owl_hierarchy), 124
 OWLReasoner (class in owlapy.owl_reasoner), 233

OWLReasonerEx (class in owlapy.owl_reasoner), 239
 OWLReflexiveObjectPropertyAxiom (class in owlapy.owl_axiom), 113
 OWLReflexiveObjectPropertyAxiom (class in owlapy.owl_ontology_manager), 188
 OWLRestriction (class in owlapy.class_expression), 51
 OWLRestriction (class in owlapy.class_expression.restriction), 35
 OWLRestriction (class in owlapy.owl_ontology), 164
 OWLRestriction (class in owlapy.render), 303
 OWLRestriction (class in owlapy.utils), 319
 OWLSameIndividualAxiom (class in owlapy.owl_axiom), 103
 OWLSameIndividualAxiom (class in owlapy.owl_ontology_manager), 192
 OWLSubAnnotationPropertyOfAxiom (class in owlapy.owl_axiom), 108
 OWLSubClassOfAxiom (class in owlapy.owl_axiom), 105
 OWLSubClassOfAxiom (class in owlapy.owl_ontology), 149
 OWLSubClassOfAxiom (class in owlapy.owl_ontology_manager), 182
 OWLSubClassOfAxiom (class in owlapy.owl_reasoner), 220
 OWLSubDataPropertyOfAxiom (class in owlapy.owl_axiom), 110
 OWLSubObjectPropertyOfAxiom (class in owlapy.owl_axiom), 110
 OWLSubPropertyAxiom (class in owlapy.owl_axiom), 109
 OWLSubPropertyAxiom (class in owlapy.owl_ontology_manager), 185
 OWLSymmetricObjectPropertyAxiom (class in owlapy.owl_axiom), 113
 OWLSymmetricObjectPropertyAxiom (class in owlapy.owl_ontology_manager), 186
 OWLThing (in module owlapy.class_expression), 62
 OWLThing (in module owlapy.owl_axiom), 95
 OWLThing (in module owlapy.owl_hierarchy), 123
 OWLThing (in module owlapy.owl_ontology), 156
 OWLThing (in module owlapy.owl_ontology_manager), 178
 OWLThing (in module owlapy.utils), 320
 OWLTopDataProperty (in module owlapy.owl_hierarchy), 123
 OWLTopDataProperty (in module owlapy.owl_literal), 142
 OWLTopObjectProperty (in module owlapy.owl_hierarchy), 123
 OWLTopObjectProperty (in module owlapy.owl_literal), 142
 OWLTransitiveObjectPropertyAxiom (class in owlapy.owl_axiom), 113
 OWLTransitiveObjectPropertyAxiom (class in owlapy.owl_ontology_manager), 187
 OWLUnaryPropertyAxiom (class in owlapy.owl_axiom), 112

P

parent (owlapy.converter.Owl2SparqlConverter attribute), 80
 parent_var (owlapy.converter.Owl2SparqlConverter attribute), 80
 parents () (owlapy.owl_hierarchy.AbstractHierarchy method), 131
 parse_boolean () (owlapy.class_expression.restriction.OWLLiteral method), 32
 parse_boolean () (owlapy.converter.OWLLiteral method), 75
 parse_boolean () (owlapy.owl_axiom.OWLLiteral method), 97
 parse_boolean () (owlapy.owl_literal.OWLLiteral method), 140
 parse_boolean () (owlapy.owl_ontology.OWLLiteral method), 152
 parse_boolean () (owlapy.owl_reasoner.OWLLiteral method), 231
 parse_boolean () (owlapy.parser.OWLLiteral method), 262
 parse_boolean () (owlapy.providers.OWLLiteral method), 284
 parse_boolean () (owlapy.render.OWLLiteral method), 292
 parse_boolean () (owlapy.utils.OWLLiteral method), 314
 parse_date () (owlapy.class_expression.restriction.OWLLiteral method), 33
 parse_date () (owlapy.converter.OWLLiteral method), 76
 parse_date () (owlapy.owl_axiom.OWLLiteral method), 98
 parse_date () (owlapy.owl_literal.OWLLiteral method), 141
 parse_date () (owlapy.owl_ontology.OWLLiteral method), 153
 parse_date () (owlapy.owl_reasoner.OWLLiteral method), 231
 parse_date () (owlapy.parser.OWLLiteral method), 263
 parse_date () (owlapy.providers.OWLLiteral method), 285
 parse_date () (owlapy.render.OWLLiteral method), 293
 parse_date () (owlapy.utils.OWLLiteral method), 315
 parse_datetime () (owlapy.class_expression.restriction.OWLLiteral method), 33
 parse_datetime () (owlapy.converter.OWLLiteral method), 76
 parse_datetime () (owlapy.owl_axiom.OWLLiteral method), 98
 parse_datetime () (owlapy.owl_literal.OWLLiteral method), 141
 parse_datetime () (owlapy.owl_ontology.OWLLiteral method), 153
 parse_datetime () (owlapy.owl_reasoner.OWLLiteral method), 232
 parse_datetime () (owlapy.parser.OWLLiteral method), 263
 parse_datetime () (owlapy.providers.OWLLiteral method), 285

parse_datetime() (owlapy.render.OWLLiteral method), 293
 parse_datetime() (owlapy.utils.OWLLiteral method), 315
 parse_double() (owlapy.class_expression.restriction.OWLLiteral method), 33
 parse_double() (owlapy.converter.OWLLiteral method), 76
 parse_double() (owlapy.owl_axiom.OWLLiteral method), 97
 parse_double() (owlapy.owl_literal.OWLLiteral method), 141
 parse_double() (owlapy.owl_ontology.OWLLiteral method), 152
 parse_double() (owlapy.owl_reasoner.OWLLiteral method), 231
 parse_double() (owlapy.parser.OWLLiteral method), 262
 parse_double() (owlapy.providers.OWLLiteral method), 284
 parse_double() (owlapy.render.OWLLiteral method), 292
 parse_double() (owlapy.utils.OWLLiteral method), 314
 parse_duration() (owlapy.class_expression.restriction.OWLLiteral method), 33
 parse_duration() (owlapy.converter.OWLLiteral method), 76
 parse_duration() (owlapy.owl_axiom.OWLLiteral method), 98
 parse_duration() (owlapy.owl_literal.OWLLiteral method), 141
 parse_duration() (owlapy.owl_ontology.OWLLiteral method), 153
 parse_duration() (owlapy.owl_reasoner.OWLLiteral method), 232
 parse_duration() (owlapy.parser.OWLLiteral method), 263
 parse_duration() (owlapy.providers.OWLLiteral method), 285
 parse_duration() (owlapy.render.OWLLiteral method), 293
 parse_duration() (owlapy.utils.OWLLiteral method), 315
 parse_expression() (owlapy.owl_object.OWLObjectParser method), 144
 parse_expression() (owlapy.parser.DLSyntaxParser method), 281
 parse_expression() (owlapy.parser.ManchesterOWLSyntaxParser method), 279
 parse_expression() (owlapy.parser.OWLObjectParser method), 265
 parse_integer() (owlapy.class_expression.restriction.OWLLiteral method), 33
 parse_integer() (owlapy.converter.OWLLiteral method), 76
 parse_integer() (owlapy.owl_axiom.OWLLiteral method), 97
 parse_integer() (owlapy.owl_literal.OWLLiteral method), 141
 parse_integer() (owlapy.owl_ontology.OWLLiteral method), 152
 parse_integer() (owlapy.owl_reasoner.OWLLiteral method), 231
 parse_integer() (owlapy.parser.OWLLiteral method), 263
 parse_integer() (owlapy.providers.OWLLiteral method), 285
 parse_integer() (owlapy.render.OWLLiteral method), 292
 parse_integer() (owlapy.utils.OWLLiteral method), 314
 parse_string() (owlapy.class_expression.restriction.OWLLiteral method), 33
 parse_string() (owlapy.converter.OWLLiteral method), 76
 parse_string() (owlapy.owl_axiom.OWLLiteral method), 98
 parse_string() (owlapy.owl_literal.OWLLiteral method), 141
 parse_string() (owlapy.owl_ontology.OWLLiteral method), 153
 parse_string() (owlapy.owl_reasoner.OWLLiteral method), 231
 parse_string() (owlapy.parser.OWLLiteral method), 263
 parse_string() (owlapy.providers.OWLLiteral method), 285
 parse_string() (owlapy.render.OWLLiteral method), 292
 parse_string() (owlapy.utils.OWLLiteral method), 314
 PATTERN (owlapy.class_expression.OWLFacet attribute), 57
 PATTERN (owlapy.class_expression.restriction.OWLFacet attribute), 35
 PATTERN (owlapy.converter.OWLFacet attribute), 79
 PATTERN (owlapy.owl_ontology.OWLFacet attribute), 168
 PATTERN (owlapy.parser.OWLFacet attribute), 266
 PATTERN (owlapy.providers.OWLFacet attribute), 287
 PATTERN (owlapy.render.OWLFacet attribute), 304
 PATTERN (owlapy.vocab.OWLFacet attribute), 338
 peek() (in module owlapy.converter), 79
 PELLET (owlapy.owl_reasoner.BaseReasoner attribute), 239
 prefix (owlapy.iri.Namespaces property), 82
 prefix (owlapy.namespaces.Namespaces property), 86
 prefix (owlapy.parser.Namespaces property), 265
 prefix (owlapy.vocab.Namespaces property), 337
 PREV (owlapy.owl_reasoner.LRUCache attribute), 232
 PREV (owlapy.utils.LRUCache attribute), 334
 process() (owlapy.converter.Owl2SparqlConverter method), 81
 properties (owlapy.converter.Owl2SparqlConverter attribute), 80
 properties() (owlapy.owl_axiom.OwLNaryPropertyAxiom method), 104
 properties() (owlapy.owl_ontology_manager.OwLNaryPropertyAxiom method), 191

R

RDF (in module owlapy.namespaces), 86
RDFS (in module owlapy.namespaces), 86
RDFS_LITERAL (owlapy.class_expression.OWL RDF Vocabulary attribute), 62
RDFS_LITERAL (owlapy.converter.OWL RDF Vocabulary attribute), 79
RDFS_LITERAL (owlapy.owl_literal.OWL RDF Vocabulary attribute), 139
RDFS_LITERAL (owlapy.parser.OWL RDF Vocabulary attribute), 267
RDFS_LITERAL (owlapy.vocab.OWL RDF Vocabulary attribute), 338
reminder (owlapy.class_expression.owl_class.IRI property), 25
reminder (owlapy.class_expression.owl_class.OWLClass property), 26
reminder (owlapy.class_expression.OWLClass property), 50
reminder (owlapy.converter.OWLClass property), 67
reminder (owlapy.iri.IRI property), 83
reminder (owlapy.owl_axiom.IRI property), 96
reminder (owlapy.owl_axiom.OWLClass property), 94
reminder (owlapy.owl_datatype.IRI property), 120
reminder (owlapy.owl_hierarchy.OWLClass property), 122
reminder (owlapy.owl_individual.IRI property), 135
reminder (owlapy.owl_ontology_manager.IRI property), 177
reminder (owlapy.owl_ontology_manager.OWLClass property), 178
reminder (owlapy.owl_ontology.IRI property), 155
reminder (owlapy.owl_ontology.OWLClass property), 155
reminder (owlapy.owl_property.IRI property), 204
reminder (owlapy.owl_reasoner.IRI property), 219
reminder (owlapy.owl_reasoner.OWLClass property), 218
reminder (owlapy.owlapi_adaptor.IRI property), 256
reminder (owlapy.parser.IRI property), 261
reminder (owlapy.parser.OWLClass property), 269
reminder (owlapy.render.IRI property), 291
reminder (owlapy.render.OWLClass property), 296
reminder (owlapy.utils.OWLClass property), 318
reminder (owlapy.vocab.IRI property), 337
remove_axiom() (owlapy.owl_ontology_manager.OntologyManager method), 202
remove_axiom() (owlapy.owl_ontology_manager.OWL OntologyManager method), 200
remove_axiom() (owlapy.owl_reasoner.OntologyManager method), 227
render() (owlapy.converter.Owl2SparqlConverter method), 80
render() (owlapy.owl_object.OWLObjectRenderer method), 144
render() (owlapy.render.DLSyntaxObjectRenderer method), 308
render() (owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer method), 309
render() (owlapy.render.OWLObjectRenderer method), 293
reset() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 247
restrict() (owlapy.owl_hierarchy.AbstractHierarchy static method), 131
restrict_and_copy() (owlapy.owl_hierarchy.AbstractHierarchy method), 131
Restriction_Literals (in module owlapy.providers), 287
RESULT (owlapy.owl_reasoner.LRUCache attribute), 232
RESULT (owlapy.utils.LRUCache attribute), 335
roots() (owlapy.owl_hierarchy.AbstractHierarchy method), 132

S

same_individuals() (owlapy.owl_hierarchy.OWLReasoner method), 126
same_individuals() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 249
same_individuals() (owlapy.owl_reasoner.OntologyReasoner method), 242
same_individuals() (owlapy.owl_reasoner.OWLReasoner method), 235
save_ontology() (owlapy.owl_ontology_manager.OntologyManager method), 202
save_ontology() (owlapy.owl_ontology_manager.OWL OntologyManager method), 200
save_ontology() (owlapy.owl_reasoner.OntologyManager method), 227
save_world() (owlapy.owl_ontology_manager.OntologyManager method), 202
save_world() (owlapy.owl_reasoner.OntologyManager method), 227
sentinel (owlapy.owl_reasoner.LRUCache attribute), 232
sentinel (owlapy.utils.LRUCache attribute), 334, 335
set_short_form_provider() (owlapy.owl_object.OWLObjectRenderer method), 143
set_short_form_provider() (owlapy.render.DLSyntaxObjectRenderer method), 308
set_short_form_provider() (owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer method), 308
set_short_form_provider() (owlapy.render.OWLObjectRenderer method), 293
siblings() (owlapy.owl_hierarchy.AbstractHierarchy method), 131
simplify() (owlapy.utils.OperandSetTransform method), 333
slots (owlapy.parser.DLSyntaxParser attribute), 281

slots (*owlapy.parser.ManchesterOWLSyntaxParser attribute*), 279
 sort () (*owlapy.utils.ConceptOperandSorter method*), 333
 sparql (*owlapy.converter.Owl2SparqlConverter attribute*), 80
 stack_parent () (*owlapy.converter.Owl2SparqlConverter method*), 80
 stack_variable () (*owlapy.converter.Owl2SparqlConverter method*), 80
 str (*owlapy.class_expression.owl_class.IRI property*), 25
 str (*owlapy.class_expression.owl_class.OWLClass property*), 26
 str (*owlapy.class_expression.OWLClass property*), 50
 str (*owlapy.class_expression.restriction.OWLDatatype property*), 34
 str (*owlapy.converter.OWLClass property*), 67
 str (*owlapy.converter.OWLDatatype property*), 78
 str (*owlapy.converter.OWLNamedIndividual property*), 75
 str (*owlapy.iri.IRI property*), 83
 str (*owlapy.meta_classes.HasIRI property*), 84
 str (*owlapy.owl_axiom.IRI property*), 96
 str (*owlapy.owl_axiom.OWLAnnotationProperty property*), 107
 str (*owlapy.owl_axiom.OWLClass property*), 94
 str (*owlapy.owl_axiom.OWLDatatype property*), 92
 str (*owlapy.owl_axiom.OWLProperty property*), 93
 str (*owlapy.owl_datatype.HasIRI property*), 120
 str (*owlapy.owl_datatype.IRI property*), 120
 str (*owlapy.owl_datatype.OWLDatatype property*), 121
 str (*owlapy.owl_hierarchy.HasIRI property*), 123
 str (*owlapy.owl_hierarchy.OWLClass property*), 122
 str (*owlapy.owl_individual.IRI property*), 135
 str (*owlapy.owl_individual.OWLNamedIndividual property*), 136
 str (*owlapy.owl_literal.OWLDatatype property*), 138
 str (*owlapy.owl_object.HasIRI property*), 143
 str (*owlapy.owl_ontology_manager.HasIRI property*), 177
 str (*owlapy.owl_ontology_manager.IRI property*), 177
 str (*owlapy.owl_ontology_manager.OWLAnnotationProperty property*), 185
 str (*owlapy.owl_ontology_manager.OWLClass property*), 178
 str (*owlapy.owl_ontology_manager.OWLImportsDeclaration property*), 201
 str (*owlapy.owl_ontology_manager.OWLNamedIndividual property*), 193
 str (*owlapy.owl_ontology_manager.OWLProperty property*), 199
 str (*owlapy.owl_ontology.IRI property*), 155
 str (*owlapy.owl_ontology.OWLAnnotationProperty property*), 149
 str (*owlapy.owl_ontology.OWLClass property*), 155
 str (*owlapy.owl_ontology.OWLDatatype property*), 151
 str (*owlapy.owl_ontology.OWLNamedIndividual property*), 151
 str (*owlapy.owl_property.IRI property*), 204
 str (*owlapy.owl_property.OWLProperty property*), 206
 str (*owlapy.owl_reasoner.IRI property*), 219
 str (*owlapy.owl_reasoner.OWLClass property*), 218
 str (*owlapy.owl_reasoner.OWLDatatype property*), 222
 str (*owlapy.owl_reasoner.OWLNamedIndividual property*), 230
 str (*owlapy.owlapi_adaptor.IRI property*), 256
 str (*owlapy.owlapi_adaptor.OWLNamedIndividual property*), 256
 str (*owlapy.parser.IRI property*), 261
 str (*owlapy.parser.OWLClass property*), 269
 str (*owlapy.parser.OWLDatatype property*), 266
 str (*owlapy.parser.OWLNamedIndividual property*), 262
 str (*owlapy.render.IRI property*), 291
 str (*owlapy.render.OWLClass property*), 296
 str (*owlapy.render.OWLDatatype property*), 308
 str (*owlapy.render.OWLNamedIndividual property*), 291
 str (*owlapy.utils.HasIRI property*), 313
 str (*owlapy.utils.OWLClass property*), 318
 str (*owlapy.utils.OWLDatatype property*), 330
 str (*owlapy.utils.OWLNamedIndividual property*), 312
 str (*owlapy.vocab.HasIRI property*), 335
 str (*owlapy.vocab.IRI property*), 336
 STRING (*owlapy.owl_literal.XSDVocabulary attribute*), 139
 STRING (*owlapy.vocab.XSDVocabulary attribute*), 338
 StringOWLDatatype (*in module owlapy.owl_literal*), 142
 StringOWLDatatype (*in module owlapy.owl_ontology*), 152
 StringOWLDatatype (*in module owlapy.parser*), 262
 sub_classes () (*owlapy.owl_hierarchy.ClassHierarchy method*), 132

- sub_classes() (owlapy.owl_hierarchy.OWLReasoner method), 128
- sub_classes() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 250
- sub_classes() (owlapy.owl_reasoner.OntologyReasoner method), 243
- sub_classes() (owlapy.owl_reasoner.OWLReasoner method), 236
- sub_data_properties() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 133
- sub_data_properties() (owlapy.owl_hierarchy.OWLReasoner method), 128
- sub_data_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 252
- sub_data_properties() (owlapy.owl_reasoner.OntologyReasoner method), 245
- sub_data_properties() (owlapy.owl_reasoner.OWLReasoner method), 237
- sub_object_properties() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 132
- sub_object_properties() (owlapy.owl_hierarchy.OWLReasoner method), 129
- sub_object_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 253
- sub_object_properties() (owlapy.owl_reasoner.OntologyReasoner method), 246
- sub_object_properties() (owlapy.owl_reasoner.OWLReasoner method), 237
- super_classes() (owlapy.owl_hierarchy.ClassHierarchy method), 132
- super_classes() (owlapy.owl_hierarchy.OWLReasoner method), 130
- super_classes() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 250
- super_classes() (owlapy.owl_reasoner.OntologyReasoner method), 244
- super_classes() (owlapy.owl_reasoner.OWLReasoner method), 238
- super_data_properties() (owlapy.owl_hierarchy.DatatypePropertyHierarchy method), 133
- super_data_properties() (owlapy.owl_hierarchy.OWLReasoner method), 129
- super_data_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 252
- super_data_properties() (owlapy.owl_reasoner.OntologyReasoner method), 245
- super_data_properties() (owlapy.owl_reasoner.OWLReasoner method), 237
- super_object_properties() (owlapy.owl_hierarchy.ObjectPropertyHierarchy method), 132
- super_object_properties() (owlapy.owl_hierarchy.OWLReasoner method), 129
- super_object_properties() (owlapy.owl_reasoner.FastInstanceCheckerReasoner method), 252
- super_object_properties() (owlapy.owl_reasoner.OntologyReasoner method), 246
- super_object_properties() (owlapy.owl_reasoner.OWLReasoner method), 238
- symbolic_form(owlapy.class_expression.OWLFacet property), 57
- symbolic_form(owlapy.class_expression.restriction.OWLFacet property), 35
- symbolic_form(owlapy.converter.OWLFacet property), 78
- symbolic_form(owlapy.owl_ontology.OWLFacet property), 168
- symbolic_form(owlapy.parser.OWLFacet property), 266
- symbolic_form(owlapy.providers.OWLFacet property), 286
- symbolic_form(owlapy.render.OWLFacet property), 304
- symbolic_form(owlapy.vocab.OWLFacet property), 338
- SyncReasoner (class in owlapy.owl_reasoner), 253

T

- TIME_DATATYPES (in module owlapy.owl_literal), 142
- to_python() (owlapy.class_expression.restriction.OWLLiteral method), 34
- to_python() (owlapy.converter.OWLLiteral method), 77
- to_python() (owlapy.owl_axiom.OWLLiteral method), 98
- to_python() (owlapy.owl_literal.OWLLiteral method), 142
- to_python() (owlapy.owl_ontology.OWLLiteral method), 153
- to_python() (owlapy.owl_reasoner.OWLLiteral method), 232
- to_python() (owlapy.parser.OWLLiteral method), 264
- to_python() (owlapy.providers.OWLLiteral method), 286
- to_python() (owlapy.render.OWLLiteral method), 293
- to_python() (owlapy.utils.OWLLiteral method), 315
- to_string_id() (owlapy.class_expression.owl_class.OWLEntity method), 24
- to_string_id() (owlapy.converter.OWLEntity method), 78
- to_string_id() (owlapy.owl_axiom.OWLEntity method), 91
- to_string_id() (owlapy.owl_datatype.OWLEntity method), 119
- to_string_id() (owlapy.owl_individual.OWLEntity method), 134
- to_string_id() (owlapy.owl_object.OWLEntity method), 144
- to_string_id() (owlapy.owl_property.OWLEntity method), 203
- to_string_id() (owlapy.render.OWLEntity method), 294
- ToOwlready2 (class in owlapy.owl_ontology), 173
- ToOwlready2 (class in owlapy.owl_ontology_manager), 197
- ToOwlready2 (class in owlapy.owl_reasoner), 226
- TopLevelCNF (class in owlapy.utils), 334
- TopLevelDNF (class in owlapy.utils), 334
- TopOWLDatatype (in module owlapy.converter), 77
- TopOWLDatatype (in module owlapy.owl_literal), 142
- TOTAL_DIGITS (owlapy.class_expression.OWLFacet attribute), 57

TOTAL_DIGITS (owlapy.class_expression.restriction.OwLFacet attribute), 35
TOTAL_DIGITS (owlapy.converter.OwLFacet attribute), 79
TOTAL_DIGITS (owlapy.owl_ontology.OwLFacet attribute), 168
TOTAL_DIGITS (owlapy.parser.OwLFacet attribute), 266
TOTAL_DIGITS (owlapy.providers.OwLFacet attribute), 287
TOTAL_DIGITS (owlapy.render.OwLFacet attribute), 304
TOTAL_DIGITS (owlapy.vocab.OwLFacet attribute), 338
triple() (owlapy.converter.Owl2SparqlConverter method), 81
type_index (owlapy.class_expression.class_expression.OwLObjectComplementOf attribute), 20
type_index (owlapy.class_expression.nary_boolean_expression.OwLObjectIntersectionOf attribute), 23
type_index (owlapy.class_expression.nary_boolean_expression.OwLObjectUnionOf attribute), 22
type_index (owlapy.class_expression.owl_class.IRI attribute), 25
type_index (owlapy.class_expression.owl_class.OwLClass attribute), 26
type_index (owlapy.class_expression.owl_class.OwLObjectComplementOf attribute), 24
type_index (owlapy.class_expression.OwLClass attribute), 49
type_index (owlapy.class_expression.OwLDataAllValuesFrom attribute), 60
type_index (owlapy.class_expression.OwLDataExactCardinality attribute), 61
type_index (owlapy.class_expression.OwLDataHasValue attribute), 60
type_index (owlapy.class_expression.OwLDataMaxCardinality attribute), 61
type_index (owlapy.class_expression.OwLDataMinCardinality attribute), 61
type_index (owlapy.class_expression.OwLDataOneOf attribute), 54
type_index (owlapy.class_expression.OwLDataSomeValuesFrom attribute), 59
type_index (owlapy.class_expression.OwLDatatypeRestriction attribute), 57
type_index (owlapy.class_expression.OwLFacetRestriction attribute), 58
type_index (owlapy.class_expression.OwLObjectAllValuesFrom attribute), 56
type_index (owlapy.class_expression.OwLObjectComplementOf attribute), 49
type_index (owlapy.class_expression.OwLObjectExactCardinality attribute), 59
type_index (owlapy.class_expression.OwLObjectHasSelf attribute), 54
type_index (owlapy.class_expression.OwLObjectHasValue attribute), 56
type_index (owlapy.class_expression.OwLObjectIntersectionOf attribute), 51
type_index (owlapy.class_expression.OwLObjectMaxCardinality attribute), 58
type_index (owlapy.class_expression.OwLObjectMinCardinality attribute), 58
type_index (owlapy.class_expression.OwLObjectOneOf attribute), 61
type_index (owlapy.class_expression.OwLObjectSomeValuesFrom attribute), 55
type_index (owlapy.class_expression.OwLObjectUnionOf attribute), 51
type_index (owlapy.class_expression.restriction.OwLDataAllValuesFrom attribute), 43
type_index (owlapy.class_expression.restriction.OwLDataExactCardinality attribute), 43
type_index (owlapy.class_expression.restriction.OwLDataHasValue attribute), 44
type_index (owlapy.class_expression.restriction.OwLDataMaxCardinality attribute), 42
type_index (owlapy.class_expression.restriction.OwLDataMinCardinality attribute), 42
type_index (owlapy.class_expression.restriction.OwLDataOneOf attribute), 44
type_index (owlapy.class_expression.restriction.OwLDataSomeValuesFrom attribute), 43
type_index (owlapy.class_expression.restriction.OwLDatatype attribute), 34
type_index (owlapy.class_expression.restriction.OwLDatatypeRestriction attribute), 45
type_index (owlapy.class_expression.restriction.OwLFacetRestriction attribute), 45
type_index (owlapy.class_expression.restriction.OwLLiteral attribute), 32
type_index (owlapy.class_expression.restriction.OwLObjectAllValuesFrom attribute), 39
type_index (owlapy.class_expression.restriction.OwLObjectExactCardinality attribute), 38
type_index (owlapy.class_expression.restriction.OwLObjectHasSelf attribute), 40
type_index (owlapy.class_expression.restriction.OwLObjectHasValue attribute), 40
type_index (owlapy.class_expression.restriction.OwLObjectIntersectionOf attribute), 29
type_index (owlapy.class_expression.restriction.OwLObjectMaxCardinality attribute), 38
type_index (owlapy.class_expression.restriction.OwLObjectMinCardinality attribute), 38
type_index (owlapy.class_expression.restriction.OwLObjectOneOf attribute), 40
type_index (owlapy.class_expression.restriction.OwLObjectSomeValuesFrom attribute), 39
type_index (owlapy.class_expression.restriction.OwLObjectUnionOf attribute), 30
type_index (owlapy.converter.OwLClass attribute), 67
type_index (owlapy.converter.OwLDataAllValuesFrom attribute), 73
type_index (owlapy.converter.OwLDataExactCardinality attribute), 67
type_index (owlapy.converter.OwLDataHasValue attribute), 74
type_index (owlapy.converter.OwLDataMaxCardinality attribute), 67
type_index (owlapy.converter.OwLDataMinCardinality attribute), 66
type_index (owlapy.converter.OwLDataOneOf attribute), 74
type_index (owlapy.converter.OwLDataProperty attribute), 77
type_index (owlapy.converter.OwLDataSomeValuesFrom attribute), 73
type_index (owlapy.converter.OwLDatatype attribute), 78
type_index (owlapy.converter.OwLDatatypeRestriction attribute), 66
type_index (owlapy.converter.OwLLiteral attribute), 75

type_index (owlapy.converter.OWLNamedIndividual attribute), 75
 type_index (owlapy.converter.OWLObjectAllValuesFrom attribute), 70
 type_index (owlapy.converter.OWLObjectComplementOf attribute), 69
 type_index (owlapy.converter.OWLObjectExactCardinality attribute), 71
 type_index (owlapy.converter.OWLObjectHasSelf attribute), 72
 type_index (owlapy.converter.OWLObjectHasValue attribute), 65
 type_index (owlapy.converter.OWLObjectIntersectionOf attribute), 69
 type_index (owlapy.converter.OWLObjectMaxCardinality attribute), 71
 type_index (owlapy.converter.OWLObjectMinCardinality attribute), 71
 type_index (owlapy.converter.OWLObjectOneOf attribute), 65
 type_index (owlapy.converter.OWLObjectProperty attribute), 77
 type_index (owlapy.converter.OWLObjectSomeValuesFrom attribute), 70
 type_index (owlapy.converter.OWLObjectUnionOf attribute), 69
 type_index (owlapy.iri.IRI attribute), 82
 type_index (owlapy.owl_axiom.IRI attribute), 95
 type_index (owlapy.owl_axiom.OWLClass attribute), 94
 type_index (owlapy.owl_axiom.OWLDatatype attribute), 92
 type_index (owlapy.owl_axiom.OWLLiteral attribute), 97
 type_index (owlapy.owl_axiom.OWLObjectUnionOf attribute), 95
 type_index (owlapy.owl_data_ranges.OWLDataComplementOf attribute), 118
 type_index (owlapy.owl_data_ranges.OWLDataIntersectionOf attribute), 118
 type_index (owlapy.owl_data_ranges.OWLDataUnionOf attribute), 118
 type_index (owlapy.owl_datatype.IRI attribute), 119
 type_index (owlapy.owl_datatype.OWLDatatype attribute), 121
 type_index (owlapy.owl_hierarchy.OWLClass attribute), 122
 type_index (owlapy.owl_hierarchy.OWLDataProperty attribute), 124
 type_index (owlapy.owl_hierarchy.OWLObjectProperty attribute), 123
 type_index (owlapy.owl_individual.IRI attribute), 134
 type_index (owlapy.owl_individual.OWLNamedIndividual attribute), 136
 type_index (owlapy.owl_literal.OWLDataProperty attribute), 140
 type_index (owlapy.owl_literal.OWLDatatype attribute), 138
 type_index (owlapy.owl_literal.OWLLiteral attribute), 140
 type_index (owlapy.owl_literal.OWLObjectProperty attribute), 139
 type_index (owlapy.owl_ontology_manager.IRI attribute), 176
 type_index (owlapy.owl_ontology_manager.OWLClass attribute), 178
 type_index (owlapy.owl_ontology_manager.OWLDataHasValue attribute), 179
 type_index (owlapy.owl_ontology_manager.OWLDataProperty attribute), 197
 type_index (owlapy.owl_ontology_manager.OWLNamedIndividual attribute), 193
 type_index (owlapy.owl_ontology_manager.OWLObjectComplementOf attribute), 181
 type_index (owlapy.owl_ontology_manager.OWLObjectHasValue attribute), 181
 type_index (owlapy.owl_ontology_manager.OWLObjectInverseOf attribute), 198
 type_index (owlapy.owl_ontology_manager.OWLObjectOneOf attribute), 180
 type_index (owlapy.owl_ontology_manager.OWLObjectProperty attribute), 198
 type_index (owlapy.owl_ontology_manager.OWLOntology attribute), 193
 type_index (owlapy.owl_ontology.IRI attribute), 154
 type_index (owlapy.owl_ontology.OWLClass attribute), 155
 type_index (owlapy.owl_ontology.OWLDataAllValuesFrom attribute), 161
 type_index (owlapy.owl_ontology.OWLDataComplementOf attribute), 150
 type_index (owlapy.owl_ontology.OWLDataExactCardinality attribute), 161
 type_index (owlapy.owl_ontology.OWLDataHasValue attribute), 162
 type_index (owlapy.owl_ontology.OWLDataIntersectionOf attribute), 150
 type_index (owlapy.owl_ontology.OWLDataMaxCardinality attribute), 162
 type_index (owlapy.owl_ontology.OWLDataMinCardinality attribute), 162
 type_index (owlapy.owl_ontology.OWLDataOneOf attribute), 163
 type_index (owlapy.owl_ontology.OWLDataProperty attribute), 165
 type_index (owlapy.owl_ontology.OWLDataSomeValuesFrom attribute), 160
 type_index (owlapy.owl_ontology.OWLDatatype attribute), 151
 type_index (owlapy.owl_ontology.OWLDatatypeRestriction attribute), 163
 type_index (owlapy.owl_ontology.OWLDataUnionOf attribute), 150
 type_index (owlapy.owl_ontology.OWLFacetRestriction attribute), 165
 type_index (owlapy.owl_ontology.OWLLiteral attribute), 152
 type_index (owlapy.owl_ontology.OWLNamedIndividual attribute), 151
 type_index (owlapy.owl_ontology.OWLObjectAllValuesFrom attribute), 158
 type_index (owlapy.owl_ontology.OWLObjectComplementOf attribute), 157
 type_index (owlapy.owl_ontology.OWLObjectExactCardinality attribute), 159
 type_index (owlapy.owl_ontology.OWLObjectHasValue attribute), 160
 type_index (owlapy.owl_ontology.OWLObjectIntersectionOf attribute), 157
 type_index (owlapy.owl_ontology.OWLObjectInverseOf attribute), 167

type_index (owlapy.owl_ontology.OWLObjectMaxCardinality attribute), 159
 type_index (owlapy.owl_ontology.OWLObjectMinCardinality attribute), 160
 type_index (owlapy.owl_ontology.OWLObjectOneOf attribute), 158
 type_index (owlapy.owl_ontology.OWLObjectProperty attribute), 165
 type_index (owlapy.owl_ontology.OWLObjectSomeValuesFrom attribute), 157
 type_index (owlapy.owl_ontology.OWLObjectUnionOf attribute), 157
 type_index (owlapy.owl_ontology.OWLOntology attribute), 169
 type_index (owlapy.owl_property.IRI attribute), 204
 type_index (owlapy.owl_property.OWLDataProperty attribute), 208
 type_index (owlapy.owl_property.OWLObjectInverseOf attribute), 207
 type_index (owlapy.owl_property.OWLObjectProperty attribute), 206
 type_index (owlapy.owl_reasoner.IRI attribute), 219
 type_index (owlapy.owl_reasoner.OWLClass attribute), 218
 type_index (owlapy.owl_reasoner.OWLDataAllValuesFrom attribute), 217
 type_index (owlapy.owl_reasoner.OWLDataComplementOf attribute), 221
 type_index (owlapy.owl_reasoner.OWLDataHasValue attribute), 217
 type_index (owlapy.owl_reasoner.OWLDataIntersectionOf attribute), 221
 type_index (owlapy.owl_reasoner.OWLDataOneOf attribute), 215
 type_index (owlapy.owl_reasoner.OWLDataProperty attribute), 228
 type_index (owlapy.owl_reasoner.OWLDataSomeValuesFrom attribute), 215
 type_index (owlapy.owl_reasoner.OWLDatatype attribute), 222
 type_index (owlapy.owl_reasoner.OWLDatatypeRestriction attribute), 216
 type_index (owlapy.owl_reasoner.OWLDataUnionOf attribute), 221
 type_index (owlapy.owl_reasoner.OWLFacetRestriction attribute), 216
 type_index (owlapy.owl_reasoner.OWLLiteral attribute), 231
 type_index (owlapy.owl_reasoner.OWLNamedIndividual attribute), 230
 type_index (owlapy.owl_reasoner.OWLObjectAllValuesFrom attribute), 212
 type_index (owlapy.owl_reasoner.OWLObjectComplementOf attribute), 211
 type_index (owlapy.owl_reasoner.OWLObjectExactCardinality attribute), 214
 type_index (owlapy.owl_reasoner.OWLObjectHasValue attribute), 213
 type_index (owlapy.owl_reasoner.OWLObjectIntersectionOf attribute), 211
 type_index (owlapy.owl_reasoner.OWLObjectInverseOf attribute), 229
 type_index (owlapy.owl_reasoner.OWLObjectMaxCardinality attribute), 214
 type_index (owlapy.owl_reasoner.OWLObjectMinCardinality attribute), 214
 type_index (owlapy.owl_reasoner.OWLObjectOneOf attribute), 212
 type_index (owlapy.owl_reasoner.OWLObjectProperty attribute), 228
 type_index (owlapy.owl_reasoner.OWLObjectSomeValuesFrom attribute), 211
 type_index (owlapy.owl_reasoner.OWLObjectUnionOf attribute), 211
 type_index (owlapy.owl_reasoner.OWLOntology attribute), 222
 type_index (owlapy.owlapi_adaptor.IRI attribute), 255
 type_index (owlapy.owlapi_adaptor.OWLNamedIndividual attribute), 256
 type_index (owlapy.parser.IRI attribute), 260
 type_index (owlapy.parser.OWLClass attribute), 269
 type_index (owlapy.parser.OWLDataAllValuesFrom attribute), 278
 type_index (owlapy.parser.OWLDataComplementOf attribute), 278
 type_index (owlapy.parser.OWLDataExactCardinality attribute), 273
 type_index (owlapy.parser.OWLDataHasValue attribute), 275
 type_index (owlapy.parser.OWLDataIntersectionOf attribute), 278
 type_index (owlapy.parser.OWLDataMaxCardinality attribute), 274
 type_index (owlapy.parser.OWLDataMinCardinality attribute), 275
 type_index (owlapy.parser.OWLDataOneOf attribute), 275
 type_index (owlapy.parser.OWLDataProperty attribute), 265
 type_index (owlapy.parser.OWLDataSomeValuesFrom attribute), 273
 type_index (owlapy.parser.OWLDatatype attribute), 266
 type_index (owlapy.parser.OWLDatatypeRestriction attribute), 276
 type_index (owlapy.parser.OWLDataUnionOf attribute), 278
 type_index (owlapy.parser.OWLFacetRestriction attribute), 272
 type_index (owlapy.parser.OWLLiteral attribute), 262
 type_index (owlapy.parser.OWLNamedIndividual attribute), 261
 type_index (owlapy.parser.OWLObjectAllValuesFrom attribute), 277
 type_index (owlapy.parser.OWLObjectComplementOf attribute), 271
 type_index (owlapy.parser.OWLObjectExactCardinality attribute), 271
 type_index (owlapy.parser.OWLObjectHasSelf attribute), 267
 type_index (owlapy.parser.OWLObjectHasValue attribute), 274
 type_index (owlapy.parser.OWLObjectIntersectionOf attribute), 268
 type_index (owlapy.parser.OWLObjectMaxCardinality attribute), 274
 type_index (owlapy.parser.OWLObjectMinCardinality attribute), 268
 type_index (owlapy.parser.OWLObjectOneOf attribute), 270

`type_index (owlapy.parser.OwlObjectProperty attribute), 264`
`type_index (owlapy.parser.OwlObjectSomeValuesFrom attribute), 268`
`type_index (owlapy.parser.OwlObjectUnionOf attribute), 268`
`type_index (owlapy.providers.OwlDatatypeRestriction attribute), 286`
`type_index (owlapy.providers.OwlFacetRestriction attribute), 287`
`type_index (owlapy.providers.OwlLiteral attribute), 284`
`type_index (owlapy.render.IRI attribute), 290`
`type_index (owlapy.render.OwlClass attribute), 296`
`type_index (owlapy.render.OwlDataAllValuesFrom attribute), 301`
`type_index (owlapy.render.OwlDataComplementOf attribute), 305`
`type_index (owlapy.render.OwlDataExactCardinality attribute), 302`
`type_index (owlapy.render.OwlDataHasValue attribute), 301`
`type_index (owlapy.render.OwlDataIntersectionOf attribute), 306`
`type_index (owlapy.render.OwlDataMaxCardinality attribute), 302`
`type_index (owlapy.render.OwlDataMinCardinality attribute), 302`
`type_index (owlapy.render.OwlDataOneOf attribute), 303`
`type_index (owlapy.render.OwlDataSomeValuesFrom attribute), 300`
`type_index (owlapy.render.OwlDatatype attribute), 308`
`type_index (owlapy.render.OwlDatatypeRestriction attribute), 307`
`type_index (owlapy.render.OwlDataUnionOf attribute), 305`
`type_index (owlapy.render.OwlFacetRestriction attribute), 306`
`type_index (owlapy.render.OwlLiteral attribute), 292`
`type_index (owlapy.render.OwlNamedIndividual attribute), 291`
`type_index (owlapy.render.OwlObjectAllValuesFrom attribute), 298`
`type_index (owlapy.render.OwlObjectComplementOf attribute), 298`
`type_index (owlapy.render.OwlObjectExactCardinality attribute), 299`
`type_index (owlapy.render.OwlObjectHasSelf attribute), 300`
`type_index (owlapy.render.OwlObjectHasValue attribute), 306`
`type_index (owlapy.render.OwlObjectIntersectionOf attribute), 298`
`type_index (owlapy.render.OwlObjectInverseOf attribute), 294`
`type_index (owlapy.render.OwlObjectMaxCardinality attribute), 300`
`type_index (owlapy.render.OwlObjectMinCardinality attribute), 299`
`type_index (owlapy.render.OwlObjectOneOf attribute), 307`
`type_index (owlapy.render.OwlObjectSomeValuesFrom attribute), 297`
`type_index (owlapy.render.OwlObjectUnionOf attribute), 298`
`type_index (owlapy.utils.HasIndex attribute), 333`
`type_index (owlapy.utils.OwlClass attribute), 318`
`type_index (owlapy.utils.OwlDataAllValuesFrom attribute), 323`
`type_index (owlapy.utils.OwlDataComplementOf attribute), 328`
`type_index (owlapy.utils.OwlDataExactCardinality attribute), 322`
`type_index (owlapy.utils.OwlDataHasValue attribute), 323`
`type_index (owlapy.utils.OwlDataIntersectionOf attribute), 329`
`type_index (owlapy.utils.OwlDataMaxCardinality attribute), 322`
`type_index (owlapy.utils.OwlDataMinCardinality attribute), 322`
`type_index (owlapy.utils.OwlDataOneOf attribute), 325`
`type_index (owlapy.utils.OwlDataProperty attribute), 317`
`type_index (owlapy.utils.OwlDataSomeValuesFrom attribute), 324`
`type_index (owlapy.utils.OwlDatatype attribute), 330`
`type_index (owlapy.utils.OwlDatatypeRestriction attribute), 327`
`type_index (owlapy.utils.OwlDataUnionOf attribute), 329`
`type_index (owlapy.utils.OwlFacetRestriction attribute), 327`
`type_index (owlapy.utils.OwlLiteral attribute), 314`
`type_index (owlapy.utils.OwlNamedIndividual attribute), 312`
`type_index (owlapy.utils.OwlObjectAllValuesFrom attribute), 324`
`type_index (owlapy.utils.OwlObjectComplementOf attribute), 319`
`type_index (owlapy.utils.OwlObjectExactCardinality attribute), 321`
`type_index (owlapy.utils.OwlObjectHasSelf attribute), 321`
`type_index (owlapy.utils.OwlObjectHasValue attribute), 326`
`type_index (owlapy.utils.OwlObjectIntersectionOf attribute), 325`
`type_index (owlapy.utils.OwlObjectInverseOf attribute), 316`
`type_index (owlapy.utils.OwlObjectMaxCardinality attribute), 321`
`type_index (owlapy.utils.OwlObjectMinCardinality attribute), 320`
`type_index (owlapy.utils.OwlObjectOneOf attribute), 328`
`type_index (owlapy.utils.OwlObjectProperty attribute), 316`
`type_index (owlapy.utils.OwlObjectSomeValuesFrom attribute), 320`
`type_index (owlapy.utils.OwlObjectUnionOf attribute), 326`
`type_index (owlapy.vocab.IRI attribute), 336`
`types () (owlapy.owl_hierarchy.OwlReasoner method), 129`

`types()` (*owlapy.owl_reasoner.FastInstanceCheckerReasoner method*), 251
`types()` (*owlapy.owl_reasoner.OntologyReasoner method*), 246
`types()` (*owlapy.owl_reasoner.OWLReasoner method*), 238

U

`update_isolated_ontology()` (*owlapy.owl_reasoner.OntologyReasoner method*), 240
`update_isolated_ontology()` (*owlapy.owl_reasoner.SyncReasoner method*), 253

V

`values()` (*owlapy.class_expression.OWLDataOneOf method*), 54
`values()` (*owlapy.class_expression.restriction.OWLDataOneOf method*), 45
`values()` (*owlapy.converter.OWLDataOneOf method*), 74
`values()` (*owlapy.owl_ontology.OWLDataOneOf method*), 163
`values()` (*owlapy.owl_reasoner.OWLDataOneOf method*), 215
`values()` (*owlapy.parser.OWLDataOneOf method*), 275
`values()` (*owlapy.render.OWLDataOneOf method*), 303
`values()` (*owlapy.utils.OWLDataOneOf method*), 325
`variable_entities` (*owlapy.converter.Owl2SparqlConverter attribute*), 80
`variables` (*owlapy.converter.Owl2SparqlConverter attribute*), 80
`VariablesMapping` (*class in owlapy.converter*), 79
`visit_abbreviated_iri()` (*owlapy.parser.DLSyntaxParser method*), 283
`visit_abbreviated_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 281
`visit_boolean_literal()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_boolean_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_cardinality_res()` (*owlapy.parser.DLSyntaxParser method*), 281
`visit_cardinality_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_class_expression()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_class_expression()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_class_iri()` (*owlapy.parser.DLSyntaxParser method*), 283
`visit_class_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_data_cardinality_res()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_data_cardinality_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_data_intersection()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_data_intersection()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_data_parentheses()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_data_parentheses()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_data_primary()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_data_primary()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_data_property_iri()` (*owlapy.parser.DLSyntaxParser method*), 283
`visit_data_property_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 281
`visit_data_some_only_res()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_data_some_only_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_data_union()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_data_union()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_data_value_res()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_data_value_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_datatype()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_datatype()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_datatype_iri()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_datatype_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_datatype_restriction()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_datatype_restriction()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_date_literal()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_date_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_datetime_literal()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_datetime_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_decimal_literal()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_decimal_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_duration_literal()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_duration_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_facet()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_facet()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_facet_restriction()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_facet_restriction()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_facet_restrictions()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_facet_restrictions()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_float_literal()` (*owlapy.parser.DLSyntaxParser method*), 282

`visit_float_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_full_iri()` (*owlapy.parser.DLSyntaxParser method*), 283
`visit_full_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 281
`visit_has_self()` (*owlapy.parser.DLSyntaxParser method*), 281
`visit_has_self()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_individual_iri()` (*owlapy.parser.DLSyntaxParser method*), 283
`visit_individual_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_individual_list()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_individual_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_integer_literal()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_integer_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_intersection()` (*owlapy.parser.DLSyntaxParser method*), 281
`visit_intersection()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_iri()` (*owlapy.parser.DLSyntaxParser method*), 283
`visit_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 281
`visit_literal()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_literal_list()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_literal_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_non_negative_integer()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_non_negative_integer()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_object_property()` (*owlapy.parser.DLSyntaxParser method*), 281
`visit_object_property()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_object_property_iri()` (*owlapy.parser.DLSyntaxParser method*), 283
`visit_object_property_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_parentheses()` (*owlapy.parser.DLSyntaxParser method*), 283
`visit_parentheses()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 281
`visit_primary()` (*owlapy.parser.DLSyntaxParser method*), 281
`visit_primary()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_quoted_string()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_quoted_string()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_simple_iri()` (*owlapy.parser.DLSyntaxParser method*), 283
`visit_simple_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 281
`visit_some_only_res()` (*owlapy.parser.DLSyntaxParser method*), 281
`visit_some_only_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_string_literal_language()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_string_literal_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_string_literal_no_language()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_string_literal_no_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_typed_literal()` (*owlapy.parser.DLSyntaxParser method*), 282
`visit_typed_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 280
`visit_union()` (*owlapy.parser.DLSyntaxParser method*), 281
`visit_union()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279
`visit_value_res()` (*owlapy.parser.DLSyntaxParser method*), 281
`visit_value_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 279

W

`worst()` (*owlapy.utils.EvaluatedDescriptionSet method*), 333

X

XSD (in module *owlapy.namespaces*), 86
XSDVocabulary (class in *owlapy.owl_literal*), 139
XSDVocabulary (class in *owlapy.vocab*), 338