
OWLAPY

Release 0.1.2

Ontolearn Team

Apr 17, 2024

Contents:

1	About owlapy	1
1.1	What is owlapy?	2
1.2	What does owlapy have to offer?	2
1.3	How to install?	2
2	Usage	3
2.1	Atomic Classes	3
2.2	Object Property	3
2.3	Complex class expressions	4
2.4	Convert to SPARQL, DL or Manchester syntax	4
3	owlapy	5
3.1	Subpackages	5
3.2	Submodules	41
3.3	Package Contents	103
	Python Module Index	105
	Index	106

OWLAPY¹: Representation of OWL objects in python.

1 About owlapy

Version: owlapy 1.0.0

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

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

² <https://dice-research.org/>

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

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 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](https://github.com/dice-group/Ontolearn)⁴ and [OntoSample](https://github.com/alkidbaci/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?

- Represent every notation in [OWL 2 Structural Specification and Functional-Style Syntax](https://www.w3.org/TR/owl2-syntax/)⁶ including:
 - Entities, Literals, and Anonymous Individuals
 - 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
```

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

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

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

2 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:



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*”.

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

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
)
```

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(expression=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/
```

(continues on next page)

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

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

(continued from previous page)

```
↪family#','hasChild')),filler=OWLClass(IRI('http://example.com/family#','male'))))

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

3 owlapy

3.1 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

Module Contents

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.

class owlapy.class_expression.class_expression.OWLClassExpression

Bases: *owlapy.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

Module Contents

Classes

<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

```
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  $1 \leq i \leq n$ . (https://www.w3.org/TR/owl2-syntax/#Union\_of\_Class\_Expressions)  
__slots__ = '_operands'  
  
type_index: Final = 3002  
  
class owlapy.class_expression.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

Module Contents

Classes

<i>OWLClass</i>	An OWL 2 named Class. Classes can be understood as sets of individuals.
-----------------	-------------------------------------------------------------------------

```
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>)

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

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

```
type_index: Final = 1001
```

```
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

Module Contents

Classes

<i>OWLRestriction</i>	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
<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 a) 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 ≤ <i>i</i> ≤ <i>n</i> .
<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>l</i> _{<i>t</i>}) consists of a data property expression DPE and a literal <i>l</i> _{<i>t</i>} ,
<i>OWLDataOneOf</i>	An enumeration of literals <i>DataOneOf</i> (<i>l</i> _{<i>t</i>} ₁ ... <i>l</i> _{<i>t</i>} _{<i>n</i>}) contains exactly the explicitly specified literals <i>l</i> _{<i>t</i>} _{<i>i</i>} with
<i>OWLDatatypeRestriction</i>	A datatype restriction <i>DatatypeRestriction</i> (DT F ₁ <i>l</i> _{<i>t</i>} ₁ ... F _{<i>n</i>} <i>l</i> _{<i>t</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.

Attributes

Literals

`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 $(= 1 R C) = \geq 1 R C$ and $\leq 1 R C$.

```
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 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.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.data_ranges.OWLDataRange)

Bases: *OWLQuantifiedRestriction[owlapy.data_ranges.OWLDataRange], OWLDataRe-*
striction

Represents a quantified data restriction.

__slots__ = ()

get_filler() → *owlapy.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.data_ranges.OWLDataRange)

Bases: *OWLCardinalityRestriction[owlapy.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.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.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.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.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 (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.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 l_{t_i}, 1 ≤ i ≤ n, such that each tuple (l_{t₁} , ..., l_{t_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 l_t) consists of a data property expression DPE and a literal l_t, and it contains all those individuals that are connected by DPE to l_t. Each such class expression can be seen as a syntactic shortcut for the class expression **DataSomeValuesFrom**(DPE **DataOneOf**(l_t)). (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.OWLObjectOneOf (
 values: owlapy.owl_literal.OWLLiteral | Iterable[owlapy.owl_literal.OWLLiteral])

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

Bases: owlapy.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 , l_i) (multiple pairs are interpreted conjunctively), where l_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.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).

```

Package Contents

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

continues on next page

Table 1 – continued from previous page

<i>OWLObjectIntersectionOf</i>	An intersection class expression <i>ObjectIntersectionOf</i> (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 <i>ObjectHasSelf</i> (OPE) consists of an object property expression OPE,
<i>OWLDataOneOf</i>	An enumeration of literals <i>DataOneOf</i> (lt1 ... ltn) contains exactly the explicitly specified literals lti with
<i>OWLQuantifiedDataRestriction</i>	Represents a quantified data restriction.
<i>OWLDataCardinalityRestriction</i>	Represents Data Property Cardinality Restrictions.
<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>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 ... Fn ltn) 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 <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>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>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 1 – continued from previous page

<i>OWLDataExactCardinality</i>	An exact cardinality expression <code>ObjectExactCardinality(n OPE CE)</code> consists of a nonnegative integer n , 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>OWLRDFVocabulary</i>	Enumerations for OWL/RDF vocabulary.

Attributes

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

class owlapy.class_expression.OWLClassExpression

Bases: *owlapy.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: *OWLAnonymousClassExpression*

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)

    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

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

    type_index: Final = 1001

    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.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.data_ranges.OWLDataRange)

Bases: *OWLQuantifiedRestriction*[*owlapy.data_ranges.OWLDataRange*], *OWLDataRestriction*

Represents a quantified data restriction.

__slots__ = ()

get_filler () → *owlapy.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.data_ranges.OWLDataRange)

Bases: *OWLCardinalityRestriction*[*owlapy.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.data_ranges.OWLDataRange*

A datatype restriction `DatatypeRestriction(DT F1 l1t1 ... 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 `liti`. (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
```

```
MIN_INCLUSIVE: Final = ( 'minInclusive', '>=' )
```

```

MIN_EXCLUSIVE: Final = ('minExclusive', '>')
MAX_INCLUSIVE: Final = ('maxInclusive', '<=')
MAX_EXCLUSIVE: Final = ('maxExclusive', '<')
LENGTH: Final = ('length', 'length')
MIN_LENGTH: Final = ('minLength', 'minLength')
MAX_LENGTH: Final = ('maxLength', 'maxLength')
PATTERN: Final = ('pattern', 'pattern')
TOTAL_DIGITS: Final = ('totalDigits', 'totalDigits')
FRACTION_DIGITS: Final = ('fractionDigits', 'fractionDigits')

static from_str(name: str) → OWLFacet

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

```
class owlapy.class_expression.OwlDataSomeValuesFrom (
    property: owlapy.owl_property.OwlDataPropertyExpression,
    filler: owlapy.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 l_i, $1 \leq i \leq n$, such that the tuple (l₁ , ..., l_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.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 (*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 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.OWLDataMinCardinality(cardinality: int,
property: owlapy.owl_property.OWLDataPropertyExpression,
filler: owlapy.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.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.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.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.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.data_ranges

OWL data ranges

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

`DataRange` := `Datatype` | `DataIntersectionOf` | `DataUnionOf` | `DataComplementOf` | `DataOneOf` | `DatatypeRestriction`

Package Contents

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

```
class owlapy.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.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.data_ranges.OWLPropertyRange
    Bases: owlapy.owl_object.OWLObject
    OWL Objects that can be the ranges of properties.

class owlapy.data_ranges.OWLDataRange
    Bases: OWLPropertyRange
    Represents a DataRange in the OWL 2 Specification.

class owlapy.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.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 ≤ i ≤ 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.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.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 () → *OWLDataRange*

Returns

The wrapped data range.

__repr__ ()

Return repr(self).

__eq__ (other)

Return self==value.

__hash__ ()

Return hash(self).

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.

3.2 Submodules

owlapy.converter

Format converter.

Module Contents

Classes

<i>VariablesMapping</i>	Helper class for owl-to-sparql conversion.
<i>Owl2SparqlConverter</i>	Convert owl (owlapy model class expressions) to SPARQL.

Functions

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

Attributes

<i>converter</i>

`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.

property `modal_depth`

property `current_variable`

```

__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

convert (root_variable: str, ce: owlapy.class_expression.OWLClassExpression,
        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]

abstract render (*e*)

stack_variable (*var*)

stack_parent (*parent: owlapy.class_expression.OWLClassExpression*)

abstract process (*ce: owlapy.class_expression.OWLClassExpression*)

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, *count*: bool = False, *values*: Iterable[owlapy.owl_individual.OWLNamedIndividual] | None = None, *named_individuals*: bool = False) → str

root_variable: the variable that will be projected *ce*: the class expression to be transformed to a SPARQL query *count*: True, counts the results ; False, projects the individuals *values*: positive or negative examples from a class expression problem *named_individuals*: if set to True, the generated SPARQL query will return only entities that are instances of owl:NamedIndividual

owlapy.converter.converter

owlapy.converter.owl_expression_to_sparql (*root_variable*: str = '?x', *expression*: owlapy.class_expression.OWLClassExpression = None, *values*: Iterable[owlapy.owl_individual.OWLNamedIndividual] | None = None, *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 *named_individuals*: if set to True, the generated SPARQL query will return only entities that are instances of owl:NamedIndividual

owlapy.has

Has index

Module Contents

Classes

<i>HasIndex</i>	Interface for types with an index; this is used to group objects by type when sorting.
-----------------	----------------------------------------------------------------------------------------

class owlapy.has.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.

owlapy.iri

OWL IRI

Module Contents

Classes

<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
------------	----------------------------------------------------

```
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.
    property str: str
        Returns: The string that specifies the IRI.
    property reminder: str
        Returns: The string corresponding to the reminder of the IRI.
    __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.

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.

Module Contents

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.

class owlapy.meta_classes.**HasIRI**

Simple class to access the IRI.

abstract property **iri**: *IRI*

Gets the IRI of this object.

Returns

The IRI of this object.

abstract property **str**: str

Gets the string representation of this object

Returns

The IRI as string

__slots__ = ()

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.

Module Contents

Classes

<i>Namespaces</i>	Namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup
-------------------	---------------------------------------------------------------------------------------------------------

Attributes

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

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

property ns: **str**

property prefix: **str**

__slots__ = ('_prefix', '_ns')

__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

Module Contents

Classes

<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.

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

Module Contents

Classes

<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.
<i>OWLDatatypeDefinitionAxiom</i>	A datatype definition <i>DatatypeDefinition(DT DR)</i> defines a new datatype DT as being semantically
<i>OWLHasKeyAxiom</i>	A key axiom <i>HasKey(CE (OPE1 ... OPEm) (DPE1 ... DPEn))</i> 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 <i>EquivalentClasses(CE1 ... CEn)</i> states that all of the class expressions CE _i ,
<i>OWLDisjointClassesAxiom</i>	A disjoint classes axiom <i>DisjointClasses(CE1 ... CEn)</i> states that all of the class expressions CE _i , 1 ≤ i ≤ 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 <i>DifferentIndividuals(a1 ... an)</i> states that all of the individuals a _i ,
<i>OWLSameIndividualAxiom</i>	An individual equality axiom <i>SameIndividual(a1 ... an)</i> states that all of the individuals a _i , 1 ≤ i ≤ 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 <i>EquivalentObjectProperties(OPE1 ... OPEn)</i> states that all of the object
<i>OWLDisjointObjectPropertiesAxiom</i>	A disjoint object properties axiom <i>DisjointObjectProperties(OPE1 ... OPEn)</i> states that all of the object
<i>OWLInverseObjectPropertiesAxiom</i>	An inverse object properties axiom <i>InverseObjectProperties(OPE1 OPE2)</i> states that the object property
<i>OWLEquivalentDataPropertiesAxiom</i>	An equivalent data properties axiom <i>EquivalentDataProperties(DPE1 ... DPEn)</i> states that all the data property
<i>OWLDisjointDataPropertiesAxiom</i>	A disjoint data properties axiom <i>DisjointDataProperties(DPE1 ... DPEn)</i> states that all of the data property
<i>OWLSubClassOfAxiom</i>	A subclass axiom <i>SubClassOf(CE1 CE2)</i> states that the class expression CE1 is a subclass of the class

continues on next page

Table 2 – continued from previous page

<i>OWLDisjointUnionAxiom</i>	A disjoint union axiom <code>DisjointUnion(C CE1 ... CEn)</code> states that a class <i>C</i> is a disjoint union of the class
<i>OWLClassAssertionAxiom</i>	A class assertion <code>ClassAssertion(CE a)</code> states that the individual <i>a</i> is an instance of the class expression <i>CE</i> .
<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 <i>as</i> — an IRI or an
<i>OWLSubAnnotationPropertyOfAxiom</i>	An annotation subproperty axiom <code>SubAnnotationPropertyOf(AP1 AP2)</code> states that the annotation property <i>AP1</i> 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.
<i>OWLSubObjectPropertyOfAxiom</i>	Object subproperty axioms are analogous to subclass axioms, and they come in two forms.
<i>OWLSubDataPropertyOfAxiom</i>	A data subproperty axiom <code>SubDataPropertyOf(DPE1 DPE2)</code> states that the data property expression <i>DPE1</i> is a
<i>OWLPropertyAssertionAxiom</i>	Base class for Property Assertion axioms.
<i>OWLObjectPropertyAssertionAxiom</i>	A positive object property assertion <code>ObjectPropertyAssertion(OPE a1 a2)</code> states that the individual <i>a1</i> is
<i>OWLNegativeObjectPropertyAssertionAxiom</i>	A negative object property assertion <code>NegativeObjectPropertyAssertion(OPE a1 a2)</code> states that the individual <i>a1</i>
<i>OWLDatapropertyAssertionAxiom</i>	A positive data property assertion <code>DataPropertyAssertion(DPE a lt)</code> states that the individual <i>a</i> is connected
<i>OWLNegativeDataPropertyAssertionAxiom</i>	A negative data property assertion <code>NegativeDataPropertyAssertion(DPE a lt)</code> states that the individual <i>a</i> 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 <code>FunctionalObjectProperty(OPE)</code> states that
<i>OWLASymmetricObjectPropertyAxiom</i>	An object property asymmetry axiom <code>AsymmetricObjectProperty(OPE)</code> states that
<i>OWLInverseFunctionalObjectPropertyAxiom</i>	An object property inverse functionality axiom <code>InverseFunctionalObjectProperty(OPE)</code>
<i>OWLIrreflexiveObjectPropertyAxiom</i>	An object property irreflexivity axiom <code>IrreflexiveObjectProperty(OPE)</code> states that the
<i>OWLReflexiveObjectPropertyAxiom</i>	An object property reflexivity axiom <code>ReflexiveObjectProperty(OPE)</code> states that the
<i>OWLSymmetricObjectPropertyAxiom</i>	An object property symmetry axiom <code>SymmetricObjectProperty(OPE)</code> states that

continues on next page

Table 2 – continued from previous page

<i>OWLTransitiveObjectPropertyAxiom</i>	An object property transitivity axiom TransitiveObjectProperty(OPE) states that the
<i>OWLDataPropertyCharacteristicAxiom</i>	Base interface for Functional data property axiom.
<i>OWLFunctionalDataPropertyAxiom</i>	A data property functionality axiom FunctionalDataProperty(DPE) 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 ObjectPropertyDomain(OPE CE) states that the domain of the
<i>OWLDataPropertyDomainAxiom</i>	A data property domain axiom DataPropertyDomain(DPE CE) states that the domain of the
<i>OWLObjectPropertyRangeAxiom</i>	An object property range axiom ObjectPropertyRange(OPE CE) states that the range of the object property
<i>OWLDataPropertyRangeAxiom</i>	A data property range axiom DataPropertyRange(DPE DR) states that the range of the data property

```
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.OWLDataPropertyAxiom(
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLPropertyAxiom
    The base interface for data property axioms.

__slots__ = ()

class owlapy.owl_axiom.OWLIndividualAxiom(
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLLogicalAxiom
    The base interface for individual axioms.

__slots__ = ()

class owlapy.owl_axiom.OWLClassAxiom(annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLLogicalAxiom
    The base interface for class axioms.

__slots__ = ()

class owlapy.owl_axiom.OWLDeclarationAxiom(entity: owlapy.owl_object.OWLEntity,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLAxiom
    Represents a Declaration axiom in the OWL 2 Specification. A declaration axiom declares an entity in an ontology.
    It doesn't affect the logical meaning of the ontology.

__slots__ = '_entity'

get_entity() → owlapy.owl_object.OWLEntity

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.owl_axiom.OWLDatatypeDefinitionAxiom(
    datatype: owlapy.owl_datatype.OWLDatatype, datarange: owlapy.owl_datatype.OWLDataRange,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLLogicalAxiom
    A datatype definition DatatypeDefinition( DT DR ) defines a new datatype DT as being semantically equivalent to
    the data range DR; the latter must be a unary data range. This axiom allows one to use the defined datatype DT as
    a synonym for DR — that is, in any expression in the ontology containing such an axiom, DT can be replaced with
    DR without affecting the meaning of the ontology.
    (https://www.w3.org/TR/owl2-syntax/#Datatype\_Definitions)

__slots__ = ('_datatype', '_datarange')

get_datatype() → owlapy.owl_datatype.OWLDatatype

```

get_datarange () → owlapy.owl_datatype.OWLDataRange

__eq__ (other)

Return self==value.

__hash__ ()

Return hash(self).

__repr__ ()

Return repr(self).

```
class owlapy.owl_axiom.OWLHasKeyAxiom (  
    class_expression: owlapy.class_expression.OWLClassExpression,  
    property_expressions: List[owlapy.owl_property.OWLPropertyExpression],  
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLLogicalAxiom*, *owlapy.meta_classes.HasOperands[owlapy.owl_property.OWLPropertyExpression]*

A key axiom `HasKey(CE (OPE1 ... OPEm) (DPE1 ... DPEn))` states that each (named) instance of the class expression CE is uniquely identified by the object property expressions OPEi and/or the data property expressions DPEj — that is, no two distinct (named) instances of CE can coincide on the values of all object property expressions OPEi and all data property expressions DPEj. In each such axiom in an OWL ontology, m or n (or both) must be larger than zero. A key axiom of the form `HasKey(owl:Thing (OPE))` is similar to the axiom `InverseFunctionalObjectProperty(OPE)`, the main differences being that the former axiom is applicable only to individuals that are explicitly named in an ontology, while the latter axiom is also applicable to anonymous individuals and individuals whose existence is implied by existential quantification.

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

__slots__ = ('_class_expression', '_property_expressions')

get_class_expression () → owlapy.class_expression.OWLClassExpression

get_property_expressions () → List[owlapy.owl_property.OWLPropertyExpression]

operands () → Iterable[owlapy.owl_property.OWLPropertyExpression]

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

Returns

The operands.

__eq__ (other)

Return self==value.

__hash__ ()

Return hash(self).

__repr__ ()

Return repr(self).

```
class owlapy.owl_axiom.OWLNaryAxiom (annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *Generic[_C]*, *OWLAxiom*

Represents an axiom that contains two or more operands that could also be represented with multiple pairwise axioms.

Parameters

_C – Class of contained objects.

```

__slots__ = ()

abstract as_pairwise_axioms () → Iterable[OWLNaryAxiom[_C]]

class owlapy.owl_axiom.OWLNaryClassAxiom (
    class_expressions: List[owlapy.class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLClassAxiom, OWLNaryAxiom[owlapy.class_expression.OWLClassExpression]

    Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
    axioms.

    __slots__ = '_class_expressions'

    class_expressions () → Iterable[owlapy.class_expression.OWLClassExpression]
        Gets all of the top level class expressions that appear in this axiom.

        Returns
            Sorted stream of class expressions that appear in the axiom.

    as_pairwise_axioms () → Iterable[OWLNaryClassAxiom]
        Gets this axiom as a set of pairwise axioms; if the axiom contains only two operands, the axiom itself is
        returned unchanged, including its annotations.

        Returns
            This axiom as a set of pairwise axioms.

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

    __repr__ ()
        Return repr(self).

class owlapy.owl_axiom.OWLEquivalentClassesAxiom (
    class_expressions: List[owlapy.class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryClassAxiom

    An equivalent classes axiom EquivalentClasses( CE1 ... CEn ) states that all of the class expressions CEi, 1 ≤ i ≤
    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_axiom.OWLDisjointClassesAxiom(
    class_expressions: List[owlapy.class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLNaryClassAxiom*

A disjoint classes axiom `DisjointClasses(CE1 ... CEn)` states that all of the class expressions CE_i , $1 \leq i \leq n$, are pairwise disjoint; that is, no individual can be at the same time an instance of both CE_i and CE_j for $i \neq j$.

(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 , $1 \leq i \leq n$, are different from each other; that is, no individuals ai and aj with $i \neq j$ can be derived to be equal. This axiom can be used to axiomatize the unique name assumption — the assumption that all different individual names denote different individuals. (https://www.w3.org/TR/owl2-syntax/#Individual_Inequality)

```
__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 , $1 \leq i \leq n$, are equal to each other. This axiom allows one to use each ai as a synonym for each aj — that is, in any expression in the ontology containing such an axiom, ai can be replaced with aj without affecting the meaning of the ontology.

(https://www.w3.org/TR/owl2-syntax/#Individual_Equality)


```

__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 expression *OPE1* is an inverse of the object property expression *OPE2*. Thus, if an individual *x* is connected by *OPE1* to an individual *y*, then *y* is also connected by *OPE2* to *x*, and vice versa.

(https://www.w3.org/TR/owl2-syntax/#Inverse_Object_Properties_2)

__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 ... DPE_n*) states that all the data property expressions *DPE_i*, $1 \leq i \leq n$, are semantically equivalent to each other. This axiom allows one to use each *DPE_i* as a synonym for each *DPE_j* — that is, in any expression in the ontology containing such an axiom, *DPE_i* can be replaced with *DPE_j* 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 ... DPE_n*) states that all of the data property expressions *DPE_i*, $1 \leq i \leq n$, are pairwise disjoint; that is, no individual *x* can be connected to a literal *y* by both

DPE_i and *DPE_j* for $i \neq j$.

(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 ≤ i ≤ 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.

    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

    __slots__ = '_iri'

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 ex-
    pression 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_datatype

OWL Datatype

Module Contents

Classes

OWLDatatype

Datatypes are entities that refer to sets of data values. Thus, datatypes are analogous to classes,

```
class owlapy.owl_datatype.OWLDatatype(iri: owlapy.iri.IRI | owlapy.meta_classes.HasIRI)
```

Bases: *owlapy.owl_object.OWLEntity, owlapy.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>)

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

__slots__ = *'_iri'*

type_index: *Final* = 4001

owlapy.owl_individual

OWL Individuals

Module Contents

Classes

<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.

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)

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

```
__slots__ = '_iri'

type_index: Final = 1005
```

`owlapy.owl_literal`

OWL Literals

Module Contents

Classes

<i>OWLLiteral</i>	Literals represent data values such as particular strings or integers. They are analogous to typed RDF
-------------------	--------------------------------------------------------------------------------------------------------

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>

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

Module Contents

Classes

<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.

```
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

Module Contents

Classes

<code>OWLontologyID</code>	An object that identifies an ontology. Since OWL 2, ontologies do not have to have an ontology IRI, or if they
<code>OWLontology</code>	Represents an OWL 2 Ontology in the OWL 2 specification.

```
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.

Module Contents

Classes

<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.

```
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.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.

abstract 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.

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.

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

__slots__ = **'_iri'**

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.
```

owlapy.owl_property

OWL Properties

Module Contents

Classes

<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 marker interface 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 (Object-InverseOf). An inverse object property expression
<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification. Data properties connect individuals with literals.

```
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

Bases: *OWLPropertyExpression*, *owlapy.owl_object.OWLEntity*

A marker interface for properties that aren't expression i.e. named properties. By definition, properties are either data properties or object properties.

__slots__ = ()

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)

property str: str

Gets the string representation of this object

Returns

The IRI as string

property iri: str

Gets the IRI of this object.

Returns

The IRI of this object.

__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.OWLObjectProperty (iri: *owlapy.iri.IRI*)

Bases: *OWLObjectPropertyExpression*, *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)

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

__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

Module Contents

Classes

OWLReasoner

An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of

class `owlapy.owl_reasoner.OWLReasoner` (*ontology*: `owlapy.owl_ontology.OWLOntology`)

An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of a particular ontology - the “root” ontology.

`__slots__ = ()`

abstract `data_property_domains` (*pe*: `owlapy.owl_property.OWLDataProperty`,
direct: *bool* = *False*) → `Iterable[owlapy.class_expression.OWLClassExpression]`

Gets the class expressions that are the direct or indirect domains of this property with respect to the imports closure of the root ontology.

Parameters

- **pe** – The property expression whose domains are to be retrieved.
- **direct** – Specifies if the direct domains should be retrieved (True), or if all domains should be retrieved (False).

Returns

Let $N = \text{equivalent_classes}(\text{DataSomeValuesFrom}(\text{pe} \text{ rdfs:Literal}))$. 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`) \rightarrow `Iterable[owlapy.class_expression.OWLClassExpression]`

Gets the class expressions that are the direct or indirect ranges of this property with respect to the imports closure of the root ontology.

Parameters

- **pe** – The property expression whose ranges are to be retrieved.
- **direct** – Specifies if the direct ranges should be retrieved (True), or if all ranges should be retrieved (False).

Returns

Let `N = 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`) \rightarrow `Iterable[owlapy.class_expression.OWLClassExpression]`

Gets the class expressions that are equivalent to the specified class expression with respect to the set of reasoner axioms.

Parameters

- **ce** – The class expression whose equivalent classes are to be retrieved.
- **only_named** – Whether to only retrieve named equivalent classes or also complex class expressions.

Returns

All class expressions `C` where the root ontology imports closure entails `EquivalentClasses(ce C)`. If `ce` is not a class name (i.e. it is an anonymous class expression) and there are no such classes `C` then there will be no result. If `ce` is unsatisfiable with respect to the set of reasoner axioms then `owl:Nothing`, i.e. the bottom node, will be returned.

abstract `disjoint_classes` (*ce*: `owlapy.class_expression.OWLClassExpression`,
only_named: `bool = True`) \rightarrow `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`)
 \rightarrow `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 is_using_triplestore ()

Return *True* if this reasoner is using a triplestore to retrieve instances.

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.

owlapy.parser

String to OWL parsers.

Module Contents

Classes

<i>ManchesterOWLSyntaxParser</i>	Manchester Syntax parser to parse strings to OWLClassExpressions.
<i>DLSyntaxParser</i>	Description Logic Syntax parser to parse strings to OWLClassExpressions.

Functions

<i>dl_to_owl_expression</i> (<i>dl_expression</i> , <i>namespace</i>)
<i>manchester_to_owl_expression</i> (<i>manchester_ex</i> ...)

Attributes

MANCHESTER_GRAMMAR

DL_GRAMMAR

DLparser

ManchesterParser

`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.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.data_ranges.OWLDataRange*

visit_data_intersection (*node*, *children*) → *owlapy.data_ranges.OWLDataRange*

visit_literal_list (*node*, *children*) → *owlapy.class_expression.OWLDataOneOf*

visit_data_parentheses (*node*, *children*) → *owlapy.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.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.data_ranges.OWLDataRange*
visit_data_intersection (*node*, *children*) → *owlapy.data_ranges.OWLDataRange*
visit_literal_list (*node*, *children*) → *owlapy.class_expression.OWLDataOneOf*
visit_data_parentheses (*node*, *children*) → *owlapy.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.

Module Contents

Functions

<code>owl_datatype_max_exclusive_restriction</code>	Create a max exclusive restriction.
<code>owl_datatype_min_exclusive_restriction</code>	Create a min exclusive restriction.
<code>owl_datatype_max_inclusive_restriction</code>	Create a max inclusive restriction.
<code>owl_datatype_min_inclusive_restriction</code>	Create a min inclusive restriction.
<code>owl_datatype_min_max_exclusive_restric</code>	Create a min-max exclusive restriction.
<code>owl_datatype_min_max_inclusive_restric</code>	Create a min-max inclusive restriction.

Attributes

Restriction_Literals

`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.

Module Contents

Classes

<i>DLSyntaxObjectRenderer</i>	DL Syntax renderer for OWL Objects.
<i>ManchesterOWLSyntaxOWLObjectRenderer</i>	Manchester Syntax renderer for OWL Objects

Functions

```
owl_expression_to_dl( $\rightarrow$  str)
```

```
owl_expression_to_manchester( $\rightarrow$  str)
```

Attributes

```
DLrenderer
```

```
ManchesterRenderer
```

```
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])  
     $\rightarrow$  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)  $\rightarrow$  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])  
     $\rightarrow$  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.util

Owlapy utils.

Module Contents

Classes

<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>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> (→ owlapy.has.HasIndex)	Cast OWL Object to HasIndex.

class owlapy.util.**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.util.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.util.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.util.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.util.**combine_nary_expressions** (*ce: owlapy.class_expression.OWLClassExpression*)
→ *owlapy.class_expression.OWLClassExpression*

owlapy.util.**combine_nary_expressions** (*ce: owlapy.data_ranges.OWLDataRange*)
→ *owlapy.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.util.iter_count (i: Iterable) → int`

Count the number of elements in an iterable.

`owlapy.util.as_index (o: owlapy.owl_object.OWLObject) → owlapy.has.HasIndex`

Cast OWL Object to HasIndex.

class `owlapy.util.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.

Module Contents

Classes

<code>OWLRDFVocabulary</code>	Enumerations for OWL/RDF vocabulary.
<code>XSDVocabulary</code>	Enumerations for XSD vocabulary.
<code>OWLFacet</code>	Enumerations for OWL facets.

```

class owlapy.vocab.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.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

```

```

MIN_INCLUSIVE: Final = ('minInclusive', '>=')
MIN_EXCLUSIVE: Final = ('minExclusive', '>')
MAX_INCLUSIVE: Final = ('maxInclusive', '<=')
MAX_EXCLUSIVE: Final = ('maxExclusive', '<')
LENGTH: Final = ('length', 'length')
MIN_LENGTH: Final = ('minLength', 'minLength')
MAX_LENGTH: Final = ('maxLength', 'maxLength')
PATTERN: Final = ('pattern', 'pattern')
TOTAL_DIGITS: Final = ('totalDigits', 'totalDigits')
FRACTION_DIGITS: Final = ('fractionDigits', 'fractionDigits')

static from_str(name: str) → OWLFacet

```

3.3 Package Contents

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_expression, namespace)</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

Attributes

<code>__version__</code>

`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 (root_variable: str = '?x',
    expression: owlapy.class_expression.OWLClassExpression = None,
    values: Iterable[owlapy.owl_individual.OWLNamedIndividual] | None = None,
    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 named_individuals: if set to True, the generated SPARQL query will return only entities that are instances of owl:NamedIndividual

```
owlapy.__version__ = '0.1.3'
```


Python Module Index

O

- owlapy, 5
- owlapy.class_expression, 5
- owlapy.class_expression.class_expression,
5
- owlapy.class_expression.nary_boolean_expression,
7
- owlapy.class_expression.owl_class, 9
- owlapy.class_expression.restriction, 10
- owlapy.converter, 41
- owlapy.data_ranges, 39
- owlapy.entities, 41
- owlapy.has, 44
- owlapy.iri, 44
- owlapy.meta_classes, 46
- owlapy.namespaces, 47
- owlapy.owl_annotation, 49
- owlapy.owl_axiom, 50
- owlapy.owl_datatype, 69
- owlapy.owl_individual, 70
- owlapy.owl_literal, 71
- owlapy.owl_object, 74
- owlapy.owl_ontology, 76
- owlapy.owl_ontology_manager, 79
- owlapy.owl_property, 81
- owlapy.owl_reasoner, 85
- owlapy.parser, 91
- owlapy.providers, 96
- owlapy.render, 97
- owlapy.util, 99
- owlapy.vocab, 101

Index

Non-alphabetical

`__contains__()` (*owlapy.converter.VariablesMapping* method), 42
`__contains__()` (*owlapy.util.LRUCache* method), 101
`__eq__()` (*owlapy.class_expression.class_expression.OwlObjectComplementOf* method), 7
`__eq__()` (*owlapy.class_expression.nary_boolean_expression.OwlNaryBooleanClassExpression* method), 8
`__eq__()` (*owlapy.class_expression.OwlDataAllValuesFrom* method), 36
`__eq__()` (*owlapy.class_expression.OwlDataCardinalityRestriction* method), 31
`__eq__()` (*owlapy.class_expression.OwlDataHasValue* method), 36
`__eq__()` (*owlapy.class_expression.OwlDataOneOf* method), 31
`__eq__()` (*owlapy.class_expression.OwlDataSomeValuesFrom* method), 35
`__eq__()` (*owlapy.class_expression.OwlDatatypeRestriction* method), 33
`__eq__()` (*owlapy.class_expression.OwlFacetRestriction* method), 34
`__eq__()` (*owlapy.class_expression.OwlHasValueRestriction* method), 28
`__eq__()` (*owlapy.class_expression.OwlNaryBooleanClassExpression* method), 27
`__eq__()` (*owlapy.class_expression.OwlObjectAllValuesFrom* method), 32
`__eq__()` (*owlapy.class_expression.OwlObjectCardinalityRestriction* method), 30
`__eq__()` (*owlapy.class_expression.OwlObjectComplementOf* method), 25
`__eq__()` (*owlapy.class_expression.OwlObjectHasSelf* method), 30
`__eq__()` (*owlapy.class_expression.OwlObjectOneOf* method), 38
`__eq__()` (*owlapy.class_expression.OwlObjectSomeValuesFrom* method), 32
`__eq__()` (*owlapy.class_expression.restriction.OwlDataAllValuesFrom* method), 20
`__eq__()` (*owlapy.class_expression.restriction.OwlDataCardinalityRestriction* method), 18
`__eq__()` (*owlapy.class_expression.restriction.OwlDataHasValue* method), 20
`__eq__()` (*owlapy.class_expression.restriction.OwlDataOneOf* method), 21
`__eq__()` (*owlapy.class_expression.restriction.OwlDataSomeValuesFrom* method), 19
`__eq__()` (*owlapy.class_expression.restriction.OwlDatatypeRestriction* method), 22
`__eq__()` (*owlapy.class_expression.restriction.OwlFacetRestriction* method), 22
`__eq__()` (*owlapy.class_expression.restriction.OwlHasValueRestriction* method), 12
`__eq__()` (*owlapy.class_expression.restriction.OwlObjectAllValuesFrom* method), 16
`__eq__()` (*owlapy.class_expression.restriction.OwlObjectCardinalityRestriction* method), 14
`__eq__()` (*owlapy.class_expression.restriction.OwlObjectHasSelf* method), 16
`__eq__()` (*owlapy.class_expression.restriction.OwlObjectOneOf* method), 17
`__eq__()` (*owlapy.class_expression.restriction.OwlObjectSomeValuesFrom* method), 15
`__eq__()` (*owlapy.data_ranges.OwlDataComplementOf* method), 41
`__eq__()` (*owlapy.data_ranges.OwlNaryDataRange* method), 40
`__eq__()` (*owlapy.data_ranges.OwlObject* method), 39
`__eq__()` (*owlapy.has.HasIndex* method), 44
`__eq__()` (*owlapy.iri.IRI* method), 45
`__eq__()` (*owlapy.namespaces.Namespaces* method), 48
`__eq__()` (*owlapy.owl_axiom.OwlAnnotation* method), 60
`__eq__()` (*owlapy.owl_axiom.OwlAnnotationAssertionAxiom* method), 61
`__eq__()` (*owlapy.owl_axiom.OwlAnnotationPropertyDomainAxiom* method), 62
`__eq__()` (*owlapy.owl_axiom.OwlAnnotationPropertyRangeAxiom* method), 62
`__eq__()` (*owlapy.owl_axiom.OwlClassAssertionAxiom* method), 60
`__eq__()` (*owlapy.owl_axiom.OwlDataPropertyCharacteristicAxiom* method), 67
`__eq__()` (*owlapy.owl_axiom.OwlDatatypeDefinitionAxiom* method), 54
`__eq__()` (*owlapy.owl_axiom.OwlDeclarationAxiom* method), 53
`__eq__()` (*owlapy.owl_axiom.OwlDisjointUnionAxiom* method), 59
`__eq__()` (*owlapy.owl_axiom.OwlHasKeyAxiom* method), 54
`__eq__()` (*owlapy.owl_axiom.OwlNaryClassAxiom* method), 55
`__eq__()` (*owlapy.owl_axiom.OwlNaryIndividualAxiom* method), 56
`__eq__()` (*owlapy.owl_axiom.OwlNaryPropertyAxiom* method), 57
`__eq__()` (*owlapy.owl_axiom.OwlObjectPropertyCharacteristicAxiom* method), 65
`__eq__()` (*owlapy.owl_axiom.OwlPropertyAssertionAxiom* method), 64
`__eq__()` (*owlapy.owl_axiom.OwlPropertyDomainAxiom* method), 68
`__eq__()` (*owlapy.owl_axiom.OwlPropertyRangeAxiom* method), 68
`__eq__()` (*owlapy.owl_axiom.OwlSubAnnotationPropertyOfAxiom* method), 62
`__eq__()` (*owlapy.owl_axiom.OwlSubClassOfAxiom* method), 59
`__eq__()` (*owlapy.owl_axiom.OwlSubPropertyAxiom* method), 63
`__eq__()` (*owlapy.owl_object.OwlNamedObject* method), 75
`__eq__()` (*owlapy.owl_object.OwlObject* method), 74
`__eq__()` (*owlapy.owl_ontology.OwlOntologyID* method), 76
`__eq__()` (*owlapy.owl_property.OwlObjectInverseOf* method), 84
`__eq__()` (*owlapy.util.OrderedOwlObject* method), 100

__getitem__() (owlapy.converter.VariablesMapping method), 42
__getitem__() (owlapy.util.LRUCache method), 101
__hash__() (owlapy.class_expression.class_expression.OwlObjectComplementOf method), 7
__hash__() (owlapy.class_expression.nary_boolean_expression.OwlNaryBooleanClassExpression method), 8
__hash__() (owlapy.class_expression.OwlDataAllValuesFrom method), 36
__hash__() (owlapy.class_expression.OwlDataCardinalityRestriction method), 31
__hash__() (owlapy.class_expression.OwlDataHasValue method), 37
__hash__() (owlapy.class_expression.OwlDataOneOf method), 31
__hash__() (owlapy.class_expression.OwlDataSomeValuesFrom method), 35
__hash__() (owlapy.class_expression.OwlDatatypeRestriction method), 33
__hash__() (owlapy.class_expression.OwlFacetRestriction method), 34
__hash__() (owlapy.class_expression.OwlHasValueRestriction method), 29
__hash__() (owlapy.class_expression.OwlNaryBooleanClassExpression method), 27
__hash__() (owlapy.class_expression.OwlObjectAllValuesFrom method), 32
__hash__() (owlapy.class_expression.OwlObjectCardinalityRestriction method), 30
__hash__() (owlapy.class_expression.OwlObjectComplementOf method), 25
__hash__() (owlapy.class_expression.OwlObjectHasSelf method), 30
__hash__() (owlapy.class_expression.OwlObjectOneOf method), 38
__hash__() (owlapy.class_expression.OwlObjectSomeValuesFrom method), 32
__hash__() (owlapy.class_expression.restriction.OwlDataAllValuesFrom method), 20
__hash__() (owlapy.class_expression.restriction.OwlDataCardinalityRestriction method), 18
__hash__() (owlapy.class_expression.restriction.OwlDataHasValue method), 21
__hash__() (owlapy.class_expression.restriction.OwlDataOneOf method), 21
__hash__() (owlapy.class_expression.restriction.OwlDataSomeValuesFrom method), 19
__hash__() (owlapy.class_expression.restriction.OwlDatatypeRestriction method), 22
__hash__() (owlapy.class_expression.restriction.OwlFacetRestriction method), 22
__hash__() (owlapy.class_expression.restriction.OwlHasValueRestriction method), 12
__hash__() (owlapy.class_expression.restriction.OwlObjectAllValuesFrom method), 16
__hash__() (owlapy.class_expression.restriction.OwlObjectCardinalityRestriction method), 14
__hash__() (owlapy.class_expression.restriction.OwlObjectHasSelf method), 16
__hash__() (owlapy.class_expression.restriction.OwlObjectOneOf method), 17
__hash__() (owlapy.class_expression.restriction.OwlObjectSomeValuesFrom method), 15
__hash__() (owlapy.data_ranges.OwlDataComplementOf method), 41
__hash__() (owlapy.data_ranges.OwlNaryDataRange method), 40
__hash__() (owlapy.data_ranges.OwlObject method), 39
__hash__() (owlapy.iri.IRI method), 45
__hash__() (owlapy.namespaces.Namespaces method), 48
__hash__() (owlapy.owl_axiom.OwlAnnotation method), 60
__hash__() (owlapy.owl_axiom.OwlAnnotationAssertionAxiom method), 61
__hash__() (owlapy.owl_axiom.OwlAnnotationPropertyDomainAxiom method), 62
__hash__() (owlapy.owl_axiom.OwlAnnotationPropertyRangeAxiom method), 62
__hash__() (owlapy.owl_axiom.OwlClassAssertionAxiom method), 60
__hash__() (owlapy.owl_axiom.OwlDataPropertyCharacteristicAxiom method), 67
__hash__() (owlapy.owl_axiom.OwlDatatypeDefinitionAxiom method), 54
__hash__() (owlapy.owl_axiom.OwlDeclarationAxiom method), 53
__hash__() (owlapy.owl_axiom.OwlDisjointUnionAxiom method), 59
__hash__() (owlapy.owl_axiom.OwlHasKeyAxiom method), 54
__hash__() (owlapy.owl_axiom.OwlNaryClassAxiom method), 55
__hash__() (owlapy.owl_axiom.OwlNaryIndividualAxiom method), 56
__hash__() (owlapy.owl_axiom.OwlNaryPropertyAxiom method), 57
__hash__() (owlapy.owl_axiom.OwlObjectPropertyCharacteristicAxiom method), 65
__hash__() (owlapy.owl_axiom.OwlPropertyAssertionAxiom method), 64
__hash__() (owlapy.owl_axiom.OwlPropertyDomainAxiom method), 68
__hash__() (owlapy.owl_axiom.OwlPropertyRangeAxiom method), 68
__hash__() (owlapy.owl_axiom.OwlSubAnnotationPropertyOfAxiom method), 62
__hash__() (owlapy.owl_axiom.OwlSubClassOfAxiom method), 59
__hash__() (owlapy.owl_axiom.OwlSubPropertyAxiom method), 63
__hash__() (owlapy.owl_object.OwlNamedObject method), 75
__hash__() (owlapy.owl_object.OwlObject method), 74
__hash__() (owlapy.owl_property.OwlObjectInverseOf method), 84
__lt__() (owlapy.owl_object.OwlNamedObject method), 75
__lt__() (owlapy.util.OrderedOwlObject method), 100
__repr__() (owlapy.class_expression.class_expression.OwlObjectComplementOf method), 7
__repr__() (owlapy.class_expression.nary_boolean_expression.OwlNaryBooleanClassExpression method), 8
__repr__() (owlapy.class_expression.OwlDataAllValuesFrom method), 36
__repr__() (owlapy.class_expression.OwlDataCardinalityRestriction method), 31
__repr__() (owlapy.class_expression.OwlDataHasValue method), 36
__repr__() (owlapy.class_expression.OwlDataOneOf method), 31

```

__repr__() (owlapy.class_expression.OWLDataSomeValuesFrom method), 35
__repr__() (owlapy.class_expression.OWLDatatypeRestriction method), 33
__repr__() (owlapy.class_expression.OWLFacetRestriction method), 34
__repr__() (owlapy.class_expression.OWLNaryBooleanClassExpression method), 27
__repr__() (owlapy.class_expression.OWLObjectAllValuesFrom method), 32
__repr__() (owlapy.class_expression.OWLObjectCardinalityRestriction method), 30
__repr__() (owlapy.class_expression.OWLObjectComplementOf method), 25
__repr__() (owlapy.class_expression.OWLObjectHasSelf method), 30
__repr__() (owlapy.class_expression.OWLObjectHasValue method), 33
__repr__() (owlapy.class_expression.OWLObjectOneOf method), 38
__repr__() (owlapy.class_expression.OWLObjectSomeValuesFrom method), 32
__repr__() (owlapy.class_expression.restriction.OWLDataAllValuesFrom method), 20
__repr__() (owlapy.class_expression.restriction.OWLDataCardinalityRestriction method), 18
__repr__() (owlapy.class_expression.restriction.OWLDataHasValue method), 20
__repr__() (owlapy.class_expression.restriction.OWLDataOneOf method), 21
__repr__() (owlapy.class_expression.restriction.OWLDataSomeValuesFrom method), 19
__repr__() (owlapy.class_expression.restriction.OWLDatatypeRestriction method), 22
__repr__() (owlapy.class_expression.restriction.OWLFacetRestriction method), 22
__repr__() (owlapy.class_expression.restriction.OWLObjectAllValuesFrom method), 16
__repr__() (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction method), 14
__repr__() (owlapy.class_expression.restriction.OWLObjectHasSelf method), 16
__repr__() (owlapy.class_expression.restriction.OWLObjectHasValue method), 17
__repr__() (owlapy.class_expression.restriction.OWLObjectOneOf method), 17
__repr__() (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom method), 15
__repr__() (owlapy.data_ranges.OWLDataComplementOf method), 41
__repr__() (owlapy.data_ranges.OWLNaryDataRange method), 40
__repr__() (owlapy.data_ranges.OWLObject method), 39
__repr__() (owlapy.iri.IRI method), 45
__repr__() (owlapy.namespaces.Namespaces method), 48
__repr__() (owlapy.owl_axiom.OWLAnnotation method), 60
__repr__() (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 61
__repr__() (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 62
__repr__() (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 62
__repr__() (owlapy.owl_axiom.OWLClassAssertionAxiom method), 60
__repr__() (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom method), 67
__repr__() (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method), 54
__repr__() (owlapy.owl_axiom.OWLDeclarationAxiom method), 53
__repr__() (owlapy.owl_axiom.OWLDisjointUnionAxiom method), 59
__repr__() (owlapy.owl_axiom.OWLHasKeyAxiom method), 54
__repr__() (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom method), 58
__repr__() (owlapy.owl_axiom.OWLNaryClassAxiom method), 55
__repr__() (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 56
__repr__() (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 57
__repr__() (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom method), 65
__repr__() (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 64
__repr__() (owlapy.owl_axiom.OWLPropertyDomainAxiom method), 68
__repr__() (owlapy.owl_axiom.OWLPropertyRangeAxiom method), 68
__repr__() (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 62
__repr__() (owlapy.owl_axiom.OWLSubClassOfAxiom method), 59
__repr__() (owlapy.owl_axiom.OWLSubPropertyAxiom method), 63
__repr__() (owlapy.owl_object.OWLNamedObject method), 75
__repr__() (owlapy.owl_object.OWLObject method), 74
__repr__() (owlapy.owl_ontology.OWLOntologyID method), 76
__repr__() (owlapy.owl_property.OWLObjectInverseOf method), 84
__setitem__() (owlapy.util.LRUCache method), 101
__slots__ (owlapy.class_expression.class_expression.OWLBooleanClassExpression attribute), 7
__slots__ (owlapy.class_expression.class_expression.OWLClassExpression attribute), 6
__slots__ (owlapy.class_expression.class_expression.OWLObjectComplementOf attribute), 7
__slots__ (owlapy.class_expression.nary_boolean_expression.OWLNaryBooleanClassExpression attribute), 8
__slots__ (owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf attribute), 8
__slots__ (owlapy.class_expression.nary_boolean_expression.OWLObjectUnionOf attribute), 8
__slots__ (owlapy.class_expression.owl_class.OWLClass attribute), 9
__slots__ (owlapy.class_expression.OWLBooleanClassExpression attribute), 25
__slots__ (owlapy.class_expression.OWLCardinalityRestriction attribute), 29
__slots__ (owlapy.class_expression.OWLClass attribute), 26
__slots__ (owlapy.class_expression.OWLClassExpression attribute), 24
__slots__ (owlapy.class_expression.OWLDataAllValuesFrom attribute), 36
__slots__ (owlapy.class_expression.OWLDataCardinalityRestriction attribute), 31

```

__slots__ (owlapy.class_expression.OWLDataExactCardinality attribute), 37
 __slots__ (owlapy.class_expression.OWLDataHasValue attribute), 36
 __slots__ (owlapy.class_expression.OWLDataMaxCardinality attribute), 37
 __slots__ (owlapy.class_expression.OWLDataMinCardinality attribute), 37
 __slots__ (owlapy.class_expression.OWLDataRestriction attribute), 29
 __slots__ (owlapy.class_expression.OWLDataSomeValuesFrom attribute), 35
 __slots__ (owlapy.class_expression.OWLDatatypeRestriction attribute), 33
 __slots__ (owlapy.class_expression.OWLFacetRestriction attribute), 34
 __slots__ (owlapy.class_expression.OWLHasValueRestriction attribute), 28
 __slots__ (owlapy.class_expression.OWLNaryBooleanClassExpression attribute), 27
 __slots__ (owlapy.class_expression.OWLObjectAllValuesFrom attribute), 32
 __slots__ (owlapy.class_expression.OWLObjectCardinalityRestriction attribute), 29
 __slots__ (owlapy.class_expression.OWLObjectComplementOf attribute), 25
 __slots__ (owlapy.class_expression.OWLObjectExactCardinality attribute), 35
 __slots__ (owlapy.class_expression.OWLObjectHasSelf attribute), 30
 __slots__ (owlapy.class_expression.OWLObjectHasValue attribute), 33
 __slots__ (owlapy.class_expression.OWLObjectIntersectionOf attribute), 27
 __slots__ (owlapy.class_expression.OWLObjectMaxCardinality attribute), 35
 __slots__ (owlapy.class_expression.OWLObjectMinCardinality attribute), 34
 __slots__ (owlapy.class_expression.OWLObjectOneOf attribute), 38
 __slots__ (owlapy.class_expression.OWLObjectRestriction attribute), 28
 __slots__ (owlapy.class_expression.OWLObjectSomeValuesFrom attribute), 32
 __slots__ (owlapy.class_expression.OWLObjectUnionOf attribute), 27
 __slots__ (owlapy.class_expression.OWLQuantifiedDataRestriction attribute), 31
 __slots__ (owlapy.class_expression.OWLQuantifiedObjectRestriction attribute), 28
 __slots__ (owlapy.class_expression.OWLQuantifiedRestriction attribute), 28
 __slots__ (owlapy.class_expression.OWLRestriction attribute), 27
 __slots__ (owlapy.class_expression.restriction.OWLCardinalityRestriction attribute), 13
 __slots__ (owlapy.class_expression.restriction.OWLDataAllValuesFrom attribute), 20
 __slots__ (owlapy.class_expression.restriction.OWLDataCardinalityRestriction attribute), 18
 __slots__ (owlapy.class_expression.restriction.OWLDataExactCardinality attribute), 19
 __slots__ (owlapy.class_expression.restriction.OWLDataHasValue attribute), 20
 __slots__ (owlapy.class_expression.restriction.OWLDataMaxCardinality attribute), 19
 __slots__ (owlapy.class_expression.restriction.OWLDataMinCardinality attribute), 18
 __slots__ (owlapy.class_expression.restriction.OWLDataRestriction attribute), 17
 __slots__ (owlapy.class_expression.restriction.OWLDataSomeValuesFrom attribute), 19
 __slots__ (owlapy.class_expression.restriction.OWLDatatypeRestriction attribute), 21
 __slots__ (owlapy.class_expression.restriction.OWLFacetRestriction attribute), 22
 __slots__ (owlapy.class_expression.restriction.OWLHasValueRestriction attribute), 12
 __slots__ (owlapy.class_expression.restriction.OWLObjectAllValuesFrom attribute), 15
 __slots__ (owlapy.class_expression.restriction.OWLObjectCardinalityRestriction attribute), 14
 __slots__ (owlapy.class_expression.restriction.OWLObjectExactCardinality attribute), 15
 __slots__ (owlapy.class_expression.restriction.OWLObjectHasSelf attribute), 16
 __slots__ (owlapy.class_expression.restriction.OWLObjectHasValue attribute), 16
 __slots__ (owlapy.class_expression.restriction.OWLObjectMaxCardinality attribute), 14
 __slots__ (owlapy.class_expression.restriction.OWLObjectMinCardinality attribute), 14
 __slots__ (owlapy.class_expression.restriction.OWLObjectOneOf attribute), 17
 __slots__ (owlapy.class_expression.restriction.OWLObjectRestriction attribute), 12
 __slots__ (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom attribute), 15
 __slots__ (owlapy.class_expression.restriction.OWLQuantifiedDataRestriction attribute), 18
 __slots__ (owlapy.class_expression.restriction.OWLQuantifiedObjectRestriction attribute), 13
 __slots__ (owlapy.class_expression.restriction.OWLQuantifiedRestriction attribute), 13
 __slots__ (owlapy.class_expression.restriction.OWLRestriction attribute), 12
 __slots__ (owlapy.converter.Owl2SparqlConverter attribute), 42
 __slots__ (owlapy.converter.VariablesMapping attribute), 42
 __slots__ (owlapy.data_ranges.HasOperands attribute), 40
 __slots__ (owlapy.data_ranges.OWLDataIntersectionOf attribute), 40
 __slots__ (owlapy.data_ranges.OWLDataUnionOf attribute), 41
 __slots__ (owlapy.data_ranges.OWLNaryDataRange attribute), 40
 __slots__ (owlapy.data_ranges.OWLObject attribute), 39
 __slots__ (owlapy.iri.IRI attribute), 45
 __slots__ (owlapy.meta_classes.HasCardinality attribute), 47
 __slots__ (owlapy.meta_classes.HasFiller attribute), 47
 __slots__ (owlapy.meta_classes.HasIRI attribute), 46
 __slots__ (owlapy.meta_classes.HasOperands attribute), 47
 __slots__ (owlapy.namespaces.Namespaces attribute), 48
 __slots__ (owlapy.owl_annotation.OWLAnnotationObject attribute), 49
 __slots__ (owlapy.owl_annotation.OWLAnnotationSubject attribute), 49

__slots__ (owlapy.owl_annotation.OWLAnnotationValue attribute), 49
 __slots__ (owlapy.owl_axiom.OWLAnnotation attribute), 60
 __slots__ (owlapy.owl_axiom.OWLAnnotationAssertionAxiom attribute), 61
 __slots__ (owlapy.owl_axiom.OWLAnnotationAxiom attribute), 61
 __slots__ (owlapy.owl_axiom.OWLAnnotationProperty attribute), 60
 __slots__ (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom attribute), 62
 __slots__ (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom attribute), 62
 __slots__ (owlapy.owl_axiom.OWLAsymmetricObjectPropertyAxiom attribute), 66
 __slots__ (owlapy.owl_axiom.OWLAxiom attribute), 52
 __slots__ (owlapy.owl_axiom.OWLClassAssertionAxiom attribute), 59
 __slots__ (owlapy.owl_axiom.OWLClassAxiom attribute), 53
 __slots__ (owlapy.owl_axiom.OWLDataPropertyAssertionAxiom attribute), 64
 __slots__ (owlapy.owl_axiom.OWLDataPropertyAxiom attribute), 53
 __slots__ (owlapy.owl_axiom.OWLDataPropertyCharacteristicAxiom attribute), 67
 __slots__ (owlapy.owl_axiom.OWLDataPropertyDomainAxiom attribute), 68
 __slots__ (owlapy.owl_axiom.OWLDataPropertyRangeAxiom attribute), 69
 __slots__ (owlapy.owl_axiom.OWLDatatypeDefinitionAxiom attribute), 53
 __slots__ (owlapy.owl_axiom.OWLDeclarationAxiom attribute), 53
 __slots__ (owlapy.owl_axiom.OWLDifferentIndividualsAxiom attribute), 56
 __slots__ (owlapy.owl_axiom.OWLDisjointClassesAxiom attribute), 56
 __slots__ (owlapy.owl_axiom.OWLDisjointDataPropertiesAxiom attribute), 58
 __slots__ (owlapy.owl_axiom.OWLDisjointObjectPropertiesAxiom attribute), 57
 __slots__ (owlapy.owl_axiom.OWLDisjointUnionAxiom attribute), 59
 __slots__ (owlapy.owl_axiom.OWLEquivalentClassesAxiom attribute), 55
 __slots__ (owlapy.owl_axiom.OWLEquivalentDataPropertiesAxiom attribute), 58
 __slots__ (owlapy.owl_axiom.OWLEquivalentObjectPropertiesAxiom attribute), 57
 __slots__ (owlapy.owl_axiom.OWLFunctionalDataPropertyAxiom attribute), 67
 __slots__ (owlapy.owl_axiom.OWLFunctionalObjectPropertyAxiom attribute), 65
 __slots__ (owlapy.owl_axiom.OWLHasKeyAxiom attribute), 54
 __slots__ (owlapy.owl_axiom.OWLIndividualAxiom attribute), 53
 __slots__ (owlapy.owl_axiom.OWLInverseFunctionalObjectPropertyAxiom attribute), 66
 __slots__ (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom attribute), 58
 __slots__ (owlapy.owl_axiom.OWLIrreflexiveObjectPropertyAxiom attribute), 66
 __slots__ (owlapy.owl_axiom.OWLLogicalAxiom attribute), 52
 __slots__ (owlapy.owl_axiom.OWLNaryAxiom attribute), 54
 __slots__ (owlapy.owl_axiom.OWLNaryClassAxiom attribute), 55
 __slots__ (owlapy.owl_axiom.OWLNaryIndividualAxiom attribute), 56
 __slots__ (owlapy.owl_axiom.OWLNaryPropertyAxiom attribute), 57
 __slots__ (owlapy.owl_axiom.OWLNegativeDataPropertyAssertionAxiom attribute), 65
 __slots__ (owlapy.owl_axiom.OWLNegativeObjectPropertyAssertionAxiom attribute), 64
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyAssertionAxiom attribute), 64
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyAxiom attribute), 52
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyCharacteristicAxiom attribute), 65
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyDomainAxiom attribute), 68
 __slots__ (owlapy.owl_axiom.OWLObjectPropertyRangeAxiom attribute), 69
 __slots__ (owlapy.owl_axiom.OWLPropertyAssertionAxiom attribute), 63
 __slots__ (owlapy.owl_axiom.OWLPropertyAxiom attribute), 52
 __slots__ (owlapy.owl_axiom.OWLPropertyDomainAxiom attribute), 67
 __slots__ (owlapy.owl_axiom.OWLPropertyRangeAxiom attribute), 68
 __slots__ (owlapy.owl_axiom.OWLReflexiveObjectPropertyAxiom attribute), 66
 __slots__ (owlapy.owl_axiom.OWLSameIndividualAxiom attribute), 56
 __slots__ (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom attribute), 61
 __slots__ (owlapy.owl_axiom.OWLSubClassOfAxiom attribute), 58
 __slots__ (owlapy.owl_axiom.OWLSubDataPropertyOfAxiom attribute), 63
 __slots__ (owlapy.owl_axiom.OWLSubObjectPropertyOfAxiom attribute), 63
 __slots__ (owlapy.owl_axiom.OWLSubPropertyAxiom attribute), 63
 __slots__ (owlapy.owl_axiom.OWLSymmetricObjectPropertyAxiom attribute), 67
 __slots__ (owlapy.owl_axiom.OWLTransitiveObjectPropertyAxiom attribute), 67
 __slots__ (owlapy.owl_axiom.OWLUnaryPropertyAxiom attribute), 65
 __slots__ (owlapy.owl_datatype.OWLDatatype attribute), 70
 __slots__ (owlapy.owl_individual.OWLIndividual attribute), 70
 __slots__ (owlapy.owl_individual.OWLNamedIndividual attribute), 70
 __slots__ (owlapy.owl_literal.OWLLiteral attribute), 72
 __slots__ (owlapy.owl_object.OWLEntity attribute), 75
 __slots__ (owlapy.owl_object.OWLNamedObject attribute), 75
 __slots__ (owlapy.owl_object.OWLObject attribute), 74
 __slots__ (owlapy.owl_ontology_manager.AddImport attribute), 80
 __slots__ (owlapy.owl_ontology_manager.OWLImportsDeclaration attribute), 80

- `__slots__` (*owlapy.owl_ontology_manager.OwlOntologyChange* attribute), 79
- `__slots__` (*owlapy.owl_ontology.OwlOntology* attribute), 77
- `__slots__` (*owlapy.owl_ontology.OwlOntologyID* attribute), 76
- `__slots__` (*owlapy.owl_property.OwlDataProperty* attribute), 84
- `__slots__` (*owlapy.owl_property.OwlDataPropertyExpression* attribute), 82
- `__slots__` (*owlapy.owl_property.OwlObjectInverseOf* attribute), 83
- `__slots__` (*owlapy.owl_property.OwlObjectProperty* attribute), 83
- `__slots__` (*owlapy.owl_property.OwlObjectPropertyExpression* attribute), 82
- `__slots__` (*owlapy.owl_property.OwlProperty* attribute), 82
- `__slots__` (*owlapy.owl_property.OwlPropertyExpression* attribute), 81
- `__slots__` (*owlapy.owl_reasoner.OwlReasoner* attribute), 85
- `__slots__` (*owlapy.render.DLSyntaxObjectRenderer* attribute), 98
- `__slots__` (*owlapy.render.ManchesterOWLSyntaxOwlObjectRenderer* attribute), 98
- `__slots__` (*owlapy.util.OrderedOwlObject* attribute), 99
- `__version__` (in module *owlapy*), 104

A

- `add_axiom()` (*owlapy.owl_ontology_manager.OwlOntologyManager* method), 80
- `AddImport` (class in *owlapy.owl_ontology_manager*), 80
- `annotations()` (*owlapy.owl_axiom.OwlAxiom* method), 52
- `append()` (*owlapy.converter.Owl2SparqlConverter* method), 43
- `append_triple()` (*owlapy.converter.Owl2SparqlConverter* method), 43
- `apply_change()` (*owlapy.owl_ontology_manager.OwlOntologyManager* method), 79
- `as_anonymous_individual()` (*owlapy.owl_annotation.OwlAnnotationObject* method), 49
- `as_index()` (in module *owlapy.util*), 101
- `as_intersection_of_min_max()` (*owlapy.class_expression.OwlDataExactCardinality* method), 37
- `as_intersection_of_min_max()` (*owlapy.class_expression.OwlObjectExactCardinality* method), 35
- `as_intersection_of_min_max()` (*owlapy.class_expression.restriction.OwlDataExactCardinality* method), 19
- `as_intersection_of_min_max()` (*owlapy.class_expression.restriction.OwlObjectExactCardinality* method), 15
- `as_iri()` (*owlapy.iri.IRI* method), 45
- `as_iri()` (*owlapy.owl_annotation.OwlAnnotationObject* method), 49
- `as_literal()` (*owlapy.owl_annotation.OwlAnnotationValue* method), 49
- `as_literal()` (*owlapy.owl_literal.OwlLiteral* method), 73
- `as_object_union_of()` (*owlapy.class_expression.OwlObjectOneOf* method), 38
- `as_object_union_of()` (*owlapy.class_expression.restriction.OwlObjectOneOf* method), 17
- `as_pairwise_axioms()` (*owlapy.owl_axiom.OwlNaryAxiom* method), 55
- `as_pairwise_axioms()` (*owlapy.owl_axiom.OwlNaryClassAxiom* method), 55
- `as_pairwise_axioms()` (*owlapy.owl_axiom.OwlNaryIndividualAxiom* method), 56
- `as_pairwise_axioms()` (*owlapy.owl_axiom.OwlNaryPropertyAxiom* method), 57
- `as_query()` (*owlapy.converter.Owl2SparqlConverter* method), 43
- `as_some_values_from()` (*owlapy.class_expression.OwlDataHasValue* method), 37
- `as_some_values_from()` (*owlapy.class_expression.OwlObjectHasValue* method), 33
- `as_some_values_from()` (*owlapy.class_expression.restriction.OwlDataHasValue* method), 21
- `as_some_values_from()` (*owlapy.class_expression.restriction.OwlObjectHasValue* method), 17
- `as_str()` (*owlapy.iri.IRI* method), 46

B

- `BOOLEAN` (*owlapy.vocab.XSDVocabulary* attribute), 102
- `BooleanOwlDatatype` (in module *owlapy.owl_literal*), 73

C

- `cache_clear()` (*owlapy.util.LRUCache* method), 101
- `cache_info()` (*owlapy.util.LRUCache* method), 101
- `ce` (*owlapy.converter.Owl2SparqlConverter* attribute), 43
- `class_expressions()` (*owlapy.owl_axiom.OwlNaryClassAxiom* method), 55
- `classes_in_signature()` (*owlapy.owl_ontology.OwlOntology* method), 77
- `cnt` (*owlapy.converter.Owl2SparqlConverter* attribute), 43
- `combine_nary_expressions()` (in module *owlapy.util*), 100
- `contains_named_equivalent_class()` (*owlapy.owl_axiom.OwlEquivalentClassesAxiom* method), 55
- `contains_owl_nothing()` (*owlapy.owl_axiom.OwlEquivalentClassesAxiom* method), 55
- `contains_owl_thing()` (*owlapy.owl_axiom.OwlEquivalentClassesAxiom* method), 55
- `convert()` (*owlapy.converter.Owl2SparqlConverter* method), 43
- `converter` (in module *owlapy.converter*), 44
- `create()` (*owlapy.iri.IRI* static method), 45
- `create_ontology()` (*owlapy.owl_ontology_manager.OwlOntologyManager* method), 79
- `current_variable` (*owlapy.converter.Owl2SparqlConverter* property), 42

D

`data_properties_in_signature()` (*owlapy.owl_ontology.OWLOntology method*), 77
`data_property_domain_axioms()` (*owlapy.owl_ontology.OWLOntology method*), 78
`data_property_domains()` (*owlapy.owl_reasoner.OWLReasoner method*), 85
`data_property_range_axioms()` (*owlapy.owl_ontology.OWLOntology method*), 78
`data_property_values()` (*owlapy.owl_reasoner.OWLReasoner method*), 87
`DATE` (*owlapy.vocab.XSDVocabulary attribute*), 102
`DATE_TIME` (*owlapy.vocab.XSDVocabulary attribute*), 102
`DATE_TIME_STAMP` (*owlapy.vocab.XSDVocabulary attribute*), 102
`DateOWLDatatype` (*in module owlapy.owl_literal*), 74
`DateTimeOWLDatatype` (*in module owlapy.owl_literal*), 74
`DECIMAL` (*owlapy.vocab.XSDVocabulary attribute*), 102
`different_individuals()` (*owlapy.owl_reasoner.OWLReasoner method*), 86
`disjoint_classes()` (*owlapy.owl_reasoner.OWLReasoner method*), 86
`disjoint_data_properties()` (*owlapy.owl_reasoner.OWLReasoner method*), 89
`disjoint_object_properties()` (*owlapy.owl_reasoner.OWLReasoner method*), 88
`DL_GRAMMAR` (*in module owlapy.parser*), 94
`dl_to_owl_expression()` (*in module owlapy*), 103
`dl_to_owl_expression()` (*in module owlapy.parser*), 96
`DLparser` (*in module owlapy.parser*), 96
`DRenderer` (*in module owlapy.render*), 99
`DLSyntaxObjectRenderer` (*class in owlapy.render*), 98
`DLSyntaxParser` (*class in owlapy.parser*), 94
`DOUBLE` (*owlapy.vocab.XSDVocabulary attribute*), 102
`DoubleOWLDatatype` (*in module owlapy.owl_literal*), 73
`DURATION` (*owlapy.vocab.XSDVocabulary attribute*), 102
`DurationOWLDatatype` (*in module owlapy.owl_literal*), 74

E

`equivalent_classes()` (*owlapy.owl_reasoner.OWLReasoner method*), 86
`equivalent_classes_axioms()` (*owlapy.owl_ontology.OWLOntology method*), 77
`equivalent_data_properties()` (*owlapy.owl_reasoner.OWLReasoner method*), 87
`equivalent_object_properties()` (*owlapy.owl_reasoner.OWLReasoner method*), 87

F

`FLOAT` (*owlapy.vocab.XSDVocabulary attribute*), 102
`flush()` (*owlapy.owl_reasoner.OWLReasoner method*), 88
`FRACTION_DIGITS` (*owlapy.class_expression.OWLFacet attribute*), 34
`FRACTION_DIGITS` (*owlapy.vocab.OWLFacet attribute*), 103
`from_str()` (*owlapy.class_expression.OWLFacet static method*), 34
`from_str()` (*owlapy.vocab.OWLFacet static method*), 103

G

`general_class_axioms()` (*owlapy.owl_ontology.OWLOntology method*), 77
`generic_visit()` (*owlapy.parser.DLSyntaxParser method*), 96
`generic_visit()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 94
`get_cardinality()` (*owlapy.class_expression.OWLCardinalityRestriction method*), 29
`get_cardinality()` (*owlapy.class_expression.restriction.OWLCardinalityRestriction method*), 13
`get_cardinality()` (*owlapy.meta_classes.HasCardinality method*), 47
`get_class_expression()` (*owlapy.owl_axiom.OWLClassAssertionAxiom method*), 59
`get_class_expression()` (*owlapy.owl_axiom.OWLHasKeyAxiom method*), 54
`get_class_expressions()` (*owlapy.owl_axiom.OWLDisjointUnionAxiom method*), 59
`get_class_nnf()` (*owlapy.util.NNF method*), 100
`get_data_range()` (*owlapy.data_ranges.OWLDataComplementOf method*), 41
`get_datarange()` (*owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method*), 53
`get_datatype()` (*owlapy.class_expression.OWLDatatypeRestriction method*), 33
`get_datatype()` (*owlapy.class_expression.restriction.OWLDatatypeRestriction method*), 22
`get_datatype()` (*owlapy.owl_axiom.OWLDatatypeDefinitionAxiom method*), 53
`get_datatype()` (*owlapy.owl_literal.OWLLiteral method*), 73
`get_default_document_iri()` (*owlapy.owl_ontology.OWLOntologyID method*), 76
`get_domain()` (*owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method*), 62
`get_domain()` (*owlapy.owl_axiom.OWLPropertyDomainAxiom method*), 68
`get_entity()` (*owlapy.owl_axiom.OWLDeclarationAxiom method*), 53
`get_facet()` (*owlapy.class_expression.OWLFacetRestriction method*), 34
`get_facet()` (*owlapy.class_expression.restriction.OWLFacetRestriction method*), 22
`get_facet_restrictions()` (*owlapy.class_expression.OWLDatatypeRestriction method*), 33

`get_facet_restrictions()` (*owlapy.class_expression.restriction.OWLDatatypeRestriction method*), 22
`get_facet_value()` (*owlapy.class_expression.OWLFacetRestriction method*), 34
`get_facet_value()` (*owlapy.class_expression.restriction.OWLFacetRestriction method*), 22
`get_filler()` (*owlapy.class_expression.OWLCardinalityRestriction method*), 29
`get_filler()` (*owlapy.class_expression.OWLHasValueRestriction method*), 29
`get_filler()` (*owlapy.class_expression.OWLQuantifiedDataRestriction method*), 31
`get_filler()` (*owlapy.class_expression.OWLQuantifiedObjectRestriction method*), 28
`get_filler()` (*owlapy.class_expression.restriction.OWLCardinalityRestriction method*), 13
`get_filler()` (*owlapy.class_expression.restriction.OWLHasValueRestriction method*), 12
`get_filler()` (*owlapy.class_expression.restriction.OWLQuantifiedDataRestriction method*), 18
`get_filler()` (*owlapy.class_expression.restriction.OWLQuantifiedObjectRestriction method*), 13
`get_filler()` (*owlapy.meta_classes.HasFiller method*), 47
`get_first_property()` (*owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom method*), 58
`get_import_declaration()` (*owlapy.owl_ontology_manager.AddImport method*), 81
`get_individual()` (*owlapy.owl_axiom.OWLClassAssertionAxiom method*), 59
`get_inverse()` (*owlapy.owl_property.OWLObjectInverseOf method*), 83
`get_inverse_property()` (*owlapy.owl_property.OWLObjectInverseOf method*), 83
`get_inverse_property()` (*owlapy.owl_property.OWLObjectProperty method*), 83
`get_inverse_property()` (*owlapy.owl_property.OWLObjectPropertyExpression method*), 82
`get_literal()` (*owlapy.owl_literal.OWLLiteral method*), 72
`get_named_property()` (*owlapy.owl_property.OWLObjectInverseOf method*), 84
`get_named_property()` (*owlapy.owl_property.OWLObjectProperty method*), 83
`get_named_property()` (*owlapy.owl_property.OWLObjectPropertyExpression method*), 82
`get_namespace()` (*owlapy.iri.IRI method*), 46
`get_nnf()` (*owlapy.class_expression.class_expression.OWLAnonymousClassExpression method*), 6
`get_nnf()` (*owlapy.class_expression.class_expression.OWLClassExpression method*), 6
`get_nnf()` (*owlapy.class_expression.owl_class.OWLClass method*), 10
`get_nnf()` (*owlapy.class_expression.OWLAnonymousClassExpression method*), 25
`get_nnf()` (*owlapy.class_expression.OWLClass method*), 26
`get_nnf()` (*owlapy.class_expression.OWLClassExpression method*), 24
`get_object()` (*owlapy.owl_axiom.OWLPropertyAssertionAxiom method*), 64
`get_object_complement_of()` (*owlapy.class_expression.class_expression.OWLAnonymousClassExpression method*), 6
`get_object_complement_of()` (*owlapy.class_expression.class_expression.OWLClassExpression method*), 6
`get_object_complement_of()` (*owlapy.class_expression.owl_class.OWLClass method*), 9
`get_object_complement_of()` (*owlapy.class_expression.OWLAnonymousClassExpression method*), 25
`get_object_complement_of()` (*owlapy.class_expression.OWLClass method*), 26
`get_object_complement_of()` (*owlapy.class_expression.OWLClassExpression method*), 24
`get_ontology()` (*owlapy.owl_ontology_manager.OWLOntologyChange method*), 79
`get_ontology_id()` (*owlapy.owl_ontology.OWLOntology method*), 78
`get_ontology_iri()` (*owlapy.owl_ontology.OWLOntologyID method*), 76
`get_operand()` (*owlapy.class_expression.class_expression.OWLObjectComplementOf method*), 7
`get_operand()` (*owlapy.class_expression.OWLObjectComplementOf method*), 25
`get_owl_class()` (*owlapy.owl_axiom.OWLDisjointUnionAxiom method*), 59
`get_owl_disjoint_classes_axiom()` (*owlapy.owl_axiom.OWLDisjointUnionAxiom method*), 59
`get_owl_equivalent_classes_axiom()` (*owlapy.owl_axiom.OWLDisjointUnionAxiom method*), 59
`get_owl_ontology_manager()` (*owlapy.owl_ontology.OWLOntology method*), 78
`get_property()` (*owlapy.class_expression.OWLDataAllValuesFrom method*), 36
`get_property()` (*owlapy.class_expression.OWLDataCardinalityRestriction method*), 31
`get_property()` (*owlapy.class_expression.OWLDataHasValue method*), 37
`get_property()` (*owlapy.class_expression.OWLDataSomeValuesFrom method*), 36
`get_property()` (*owlapy.class_expression.OWLObjectAllValuesFrom method*), 32
`get_property()` (*owlapy.class_expression.OWLObjectCardinalityRestriction method*), 30
`get_property()` (*owlapy.class_expression.OWLObjectHasSelf method*), 30
`get_property()` (*owlapy.class_expression.OWLObjectHasValue method*), 33
`get_property()` (*owlapy.class_expression.OWLObjectRestriction method*), 28
`get_property()` (*owlapy.class_expression.OWLObjectSomeValuesFrom method*), 32
`get_property()` (*owlapy.class_expression.OWLRestriction method*), 27
`get_property()` (*owlapy.class_expression.restriction.OWLDataAllValuesFrom method*), 20
`get_property()` (*owlapy.class_expression.restriction.OWLDataCardinalityRestriction method*), 18
`get_property()` (*owlapy.class_expression.restriction.OWLDataHasValue method*), 21
`get_property()` (*owlapy.class_expression.restriction.OWLDataSomeValuesFrom method*), 20
`get_property()` (*owlapy.class_expression.restriction.OWLObjectAllValuesFrom method*), 16
`get_property()` (*owlapy.class_expression.restriction.OWLObjectCardinalityRestriction method*), 14
`get_property()` (*owlapy.class_expression.restriction.OWLObjectHasSelf method*), 16
`get_property()` (*owlapy.class_expression.restriction.OWLObjectHasValue method*), 16
`get_property()` (*owlapy.class_expression.restriction.OWLObjectRestriction method*), 13
`get_property()` (*owlapy.class_expression.restriction.OWLObjectSomeValuesFrom method*), 15
`get_property()` (*owlapy.class_expression.restriction.OWLRestriction method*), 12

get_property() (owlapy.owl_axiom.OWLAnnotation method), 60
 get_property() (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 61
 get_property() (owlapy.owl_axiom.OWLAnnotationPropertyDomainAxiom method), 62
 get_property() (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 62
 get_property() (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 64
 get_property() (owlapy.owl_axiom.OWLUnaryPropertyAxiom method), 65
 get_property_expressions() (owlapy.owl_axiom.OWLHasKeyAxiom method), 54
 get_range() (owlapy.owl_axiom.OWLAnnotationPropertyRangeAxiom method), 62
 get_range() (owlapy.owl_axiom.OWLPropertyRangeAxiom method), 68
 get_remainder() (owlapy.iri.IRI method), 46
 get_root_ontology() (owlapy.owl_reasoner.OWLReasoner method), 90
 get_second_property() (owlapy.owl_axiom.OWLInverseObjectPropertiesAxiom method), 58
 get_short_form() (owlapy.iri.IRI method), 46
 get_sub_class() (owlapy.owl_axiom.OWLSubClassOfAxiom method), 59
 get_sub_property() (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 61
 get_sub_property() (owlapy.owl_axiom.OWLSubPropertyAxiom method), 63
 get_subject() (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 61
 get_subject() (owlapy.owl_axiom.OWLPropertyAssertionAxiom method), 64
 get_super_class() (owlapy.owl_axiom.OWLSubClassOfAxiom method), 59
 get_super_property() (owlapy.owl_axiom.OWLSubAnnotationPropertyOfAxiom method), 61
 get_super_property() (owlapy.owl_axiom.OWLSubPropertyAxiom method), 63
 get_top_level_cnf() (owlapy.util.TopLevelCNF method), 100
 get_top_level_dnf() (owlapy.util.TopLevelDNF method), 100
 get_value() (owlapy.owl_axiom.OWLAnnotation method), 60
 get_value() (owlapy.owl_axiom.OWLAnnotationAssertionAxiom method), 61
 get_variable() (owlapy.converter.VariablesMapping method), 42
 get_version_iri() (owlapy.owl_ontology.OWLOntologyID method), 76
 grouping_vars (owlapy.converter.Owl2SparqlConverter attribute), 43

H

HasCardinality (class in owlapy.meta_classes), 47
 HasFiller (class in owlapy.meta_classes), 47
 HasIndex (class in owlapy.has), 44
 HasIRI (class in owlapy.meta_classes), 46
 HasOperands (class in owlapy.data_ranges), 40
 HasOperands (class in owlapy.meta_classes), 46
 having_conditions (owlapy.converter.Owl2SparqlConverter attribute), 43

I

individuals() (owlapy.class_expression.OWLObjectOneOf method), 38
 individuals() (owlapy.class_expression.restriction.OWLObjectOneOf method), 17
 individuals() (owlapy.owl_axiom.OWLNaryIndividualAxiom method), 56
 individuals_in_signature() (owlapy.owl_ontology.OWLOntology method), 77
 instances() (owlapy.owl_reasoner.OWLReasoner method), 88
 INTEGER (owlapy.vocab.XSDVocabulary attribute), 102
 IntegerOWLDatatype (in module owlapy.owl_literal), 73
 IRI (class in owlapy.iri), 45
 iri (owlapy.class_expression.owl_class.OWLClass property), 9
 iri (owlapy.class_expression.OWLClass property), 26
 iri (owlapy.meta_classes.HasIRI property), 46
 iri (owlapy.owl_axiom.OWLAnnotationProperty property), 60
 iri (owlapy.owl_datatype.OWLDatatype property), 69
 iri (owlapy.owl_individual.OWLNamedIndividual property), 70
 iri (owlapy.owl_ontology_manager.OWLImportsDeclaration property), 80
 iri (owlapy.owl_property.OWLDataProperty property), 84
 iri (owlapy.owl_property.OWLObjectProperty property), 83
 is_annotated() (owlapy.owl_axiom.OWLAxiom method), 52
 is_annotation_axiom() (owlapy.owl_axiom.OWLAnnotationAxiom method), 61
 is_annotation_axiom() (owlapy.owl_axiom.OWLAxiom method), 52
 is_anonymous() (owlapy.data_ranges.OWLObject method), 39
 is_anonymous() (owlapy.owl_object.OWLEntity method), 75
 is_anonymous() (owlapy.owl_object.OWLObject method), 74
 is_anonymous() (owlapy.owl_ontology.OWLOntology method), 78
 is_anonymous() (owlapy.owl_ontology.OWLOntologyID method), 76
 is_boolean() (owlapy.owl_literal.OWLLiteral method), 72
 is_data_property_expression() (owlapy.owl_property.OWLDataPropertyExpression method), 82
 is_data_property_expression() (owlapy.owl_property.OWLPropertyExpression method), 81

`is_data_restriction()` (*owlapy.class_expression.OWLDataRestriction method*), 29
`is_data_restriction()` (*owlapy.class_expression.OWLRestriction method*), 27
`is_data_restriction()` (*owlapy.class_expression.restriction.OWLDataRestriction method*), 18
`is_data_restriction()` (*owlapy.class_expression.restriction.OWLRestriction method*), 12
`is_date()` (*owlapy.owl_literal.OWLLiteral method*), 72
`is_datetime()` (*owlapy.owl_literal.OWLLiteral method*), 73
`is_double()` (*owlapy.owl_literal.OWLLiteral method*), 72
`is_duration()` (*owlapy.owl_literal.OWLLiteral method*), 73
`is_integer()` (*owlapy.owl_literal.OWLLiteral method*), 72
`is_isolated()` (*owlapy.owl_reasoner.OWLReasoner method*), 90
`is_literal()` (*owlapy.owl_annotation.OWLAnnotationValue method*), 49
`is_literal()` (*owlapy.owl_literal.OWLLiteral method*), 73
`is_logical_axiom()` (*owlapy.owl_axiom.OWLAXiom method*), 52
`is_logical_axiom()` (*owlapy.owl_axiom.OWLLogicalAxiom method*), 52
`is_nothing()` (*owlapy.iri.IRI method*), 45
`is_object_property_expression()` (*owlapy.owl_property.OWLObjectPropertyExpression method*), 82
`is_object_property_expression()` (*owlapy.owl_property.OWLPropertyExpression method*), 81
`is_object_restriction()` (*owlapy.class_expression.OWLObjectRestriction method*), 28
`is_object_restriction()` (*owlapy.class_expression.OWLRestriction method*), 27
`is_object_restriction()` (*owlapy.class_expression.restriction.OWLObjectRestriction method*), 13
`is_object_restriction()` (*owlapy.class_expression.restriction.OWLRestriction method*), 12
`is_owl_nothing()` (*owlapy.class_expression.class_expression.OWLAnonymousClassExpression method*), 6
`is_owl_nothing()` (*owlapy.class_expression.class_expression.OWLClassExpression method*), 6
`is_owl_nothing()` (*owlapy.class_expression.owl_class.OWLClass method*), 9
`is_owl_nothing()` (*owlapy.class_expression.OWLAnonymousClassExpression method*), 24
`is_owl_nothing()` (*owlapy.class_expression.OWLClass method*), 26
`is_owl_nothing()` (*owlapy.class_expression.OWLClassExpression method*), 24
`is_owl_thing()` (*owlapy.class_expression.class_expression.OWLAnonymousClassExpression method*), 6
`is_owl_thing()` (*owlapy.class_expression.class_expression.OWLClassExpression method*), 6
`is_owl_thing()` (*owlapy.class_expression.owl_class.OWLClass method*), 9
`is_owl_thing()` (*owlapy.class_expression.OWLAnonymousClassExpression method*), 25
`is_owl_thing()` (*owlapy.class_expression.OWLClass method*), 26
`is_owl_thing()` (*owlapy.class_expression.OWLClassExpression method*), 24
`is_owl_top_data_property()` (*owlapy.owl_property.OWLDataProperty method*), 84
`is_owl_top_data_property()` (*owlapy.owl_property.OWLPropertyExpression method*), 82
`is_owl_top_object_property()` (*owlapy.owl_property.OWLObjectProperty method*), 83
`is_owl_top_object_property()` (*owlapy.owl_property.OWLPropertyExpression method*), 81
`is_reserved_vocabulary()` (*owlapy.iri.IRI method*), 45
`is_string()` (*owlapy.owl_literal.OWLLiteral method*), 72
`is_thing()` (*owlapy.iri.IRI method*), 45
`is_using_triplestore()` (*owlapy.owl_reasoner.OWLReasoner method*), 90
`iter_count()` (*in module owlapy.util*), 100

K

`KEY` (*owlapy.util.LRUCache attribute*), 101

L

`LENGTH` (*owlapy.class_expression.OWLFacet attribute*), 34
`LENGTH` (*owlapy.vocab.OWLFacet attribute*), 103
Literals (*in module owlapy.class_expression.restriction*), 12
Literals (*in module owlapy.owl_literal*), 71
`load_ontology()` (*owlapy.owl_ontology_manager.OWLOntologyManager method*), 79
`LONG` (*owlapy.vocab.XSDVocabulary attribute*), 102
`LRUCache` (*class in owlapy.util*), 101

M

`MANCHESTER_GRAMMAR` (*in module owlapy.parser*), 92
`manchester_to_owl_expression()` (*in module owlapy*), 103
`manchester_to_owl_expression()` (*in module owlapy.parser*), 96
`ManchesterOWLSyntaxOWLObjectRenderer` (*class in owlapy.render*), 98
`ManchesterOWLSyntaxParser` (*class in owlapy.parser*), 92
`ManchesterParser` (*in module owlapy.parser*), 96
`ManchesterRenderer` (*in module owlapy.render*), 99
`mapping` (*owlapy.converter.Owl2SparqlConverter attribute*), 43
`MAX_EXCLUSIVE` (*owlapy.class_expression.OWLFacet attribute*), 34
`MAX_EXCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 103

MAX_INCLUSIVE (*owlapy.class_expression.OwLFacet attribute*), 34
 MAX_INCLUSIVE (*owlapy.vocab.OwLFacet attribute*), 103
 MAX_LENGTH (*owlapy.class_expression.OwLFacet attribute*), 34
 MAX_LENGTH (*owlapy.vocab.OwLFacet attribute*), 103
 MIN_EXCLUSIVE (*owlapy.class_expression.OwLFacet attribute*), 33
 MIN_EXCLUSIVE (*owlapy.vocab.OwLFacet attribute*), 103
 MIN_INCLUSIVE (*owlapy.class_expression.OwLFacet attribute*), 33
 MIN_INCLUSIVE (*owlapy.vocab.OwLFacet attribute*), 102
 MIN_LENGTH (*owlapy.class_expression.OwLFacet attribute*), 34
 MIN_LENGTH (*owlapy.vocab.OwLFacet attribute*), 103
 modal_depth (*owlapy.converter.Owl2SparqlConverter property*), 42
 module
 owlapy, 5
 owlapy.class_expression, 5
 owlapy.class_expression.class_expression, 5
 owlapy.class_expression.nary_boolean_expression, 7
 owlapy.class_expression.owl_class, 9
 owlapy.class_expression.restriction, 10
 owlapy.converter, 41
 owlapy.data_ranges, 39
 owlapy.entities, 41
 owlapy.has, 44
 owlapy.iri, 44
 owlapy.meta_classes, 46
 owlapy.namespaces, 47
 owlapy.owl_annotation, 49
 owlapy.owl_axiom, 50
 owlapy.owl_datatype, 69
 owlapy.owl_individual, 70
 owlapy.owl_literal, 71
 owlapy.owl_object, 74
 owlapy.owl_ontology, 76
 owlapy.owl_ontology_manager, 79
 owlapy.owl_property, 81
 owlapy.owl_reasoner, 85
 owlapy.parser, 91
 owlapy.providers, 96
 owlapy.render, 97
 owlapy.util, 99
 owlapy.vocab, 101

N

named_classes() (*owlapy.owl_axiom.OwLEquivalentClassesAxiom method*), 55
 Namespaces (*class in owlapy.namespaces*), 48
 new_count_var() (*owlapy.converter.Owl2SparqlConverter method*), 43
 new_individual_variable() (*owlapy.converter.VariablesMapping method*), 42
 new_property_variable() (*owlapy.converter.VariablesMapping method*), 42
 NEXT (*owlapy.util.LRUCache attribute*), 101
 NNF (*class in owlapy.util*), 100
 ns (*owlapy.namespaces.Namespaces property*), 48
 ns (*owlapy.parser.DLSyntaxParser attribute*), 94
 ns (*owlapy.parser.ManchesterOWLSyntaxParser attribute*), 92
 NUMERIC_DATATYPES (*in module owlapy.owl_literal*), 74

O

o (*owlapy.util.OrderedOWLObject attribute*), 99, 100
 object_properties_in_signature() (*owlapy.owl_ontology.OwLOntology method*), 77
 object_property_domain_axioms() (*owlapy.owl_ontology.OwLOntology method*), 78
 object_property_domains() (*owlapy.owl_reasoner.OwLReasoner method*), 85
 object_property_range_axioms() (*owlapy.owl_ontology.OwLOntology method*), 78
 object_property_ranges() (*owlapy.owl_reasoner.OwLReasoner method*), 86
 object_property_values() (*owlapy.owl_reasoner.OwLReasoner method*), 87
 operands() (*owlapy.class_expression.class_expression.OwLObjectComplementOf method*), 7
 operands() (*owlapy.class_expression.nary_boolean_expression.OwLNaryBooleanClassExpression method*), 8
 operands() (*owlapy.class_expression.OwLDataOneOf method*), 30
 operands() (*owlapy.class_expression.OwLNaryBooleanClassExpression method*), 27
 operands() (*owlapy.class_expression.OwLObjectComplementOf method*), 25

- `operands()` (*owlapy.class_expression.OwLObjectOneOf* method), 38
- `operands()` (*owlapy.class_expression.restriction.OwLDataOneOf* method), 21
- `operands()` (*owlapy.class_expression.restriction.OwLObjectOneOf* method), 17
- `operands()` (*owlapy.data_ranges.HasOperands* method), 40
- `operands()` (*owlapy.data_ranges.OwLNaryDataRange* method), 40
- `operands()` (*owlapy.meta_classes.HasOperands* method), 47
- `operands()` (*owlapy.owl_axiom.OwLHasKeyAxiom* method), 54
- `operator` (*owlapy.class_expression.OwLFacet* property), 33
- `operator` (*owlapy.vocab.OwLFacet* property), 102
- `OrderedOwLObject` (class in *owlapy.util*), 99
- `OwL` (in module *owlapy.namespaces*), 48
- `OwL2SparqlConverter` (class in *owlapy.converter*), 42
- `OwL_BOTTOM_DATA_PROPERTY` (*owlapy.class_expression.OwLRDFVocabulary* attribute), 39
- `OwL_BOTTOM_DATA_PROPERTY` (*owlapy.vocab.OwLRDFVocabulary* attribute), 102
- `OwL_BOTTOM_OBJECT_PROPERTY` (*owlapy.class_expression.OwLRDFVocabulary* attribute), 39
- `OwL_BOTTOM_OBJECT_PROPERTY` (*owlapy.vocab.OwLRDFVocabulary* attribute), 102
- `OwL_CLASS` (*owlapy.class_expression.OwLRDFVocabulary* attribute), 38
- `OwL_CLASS` (*owlapy.vocab.OwLRDFVocabulary* attribute), 102
- `owl_datatype_max_exclusive_restriction()` (in module *owlapy.providers*), 97
- `owl_datatype_max_inclusive_restriction()` (in module *owlapy.providers*), 97
- `owl_datatype_min_exclusive_restriction()` (in module *owlapy.providers*), 97
- `owl_datatype_min_inclusive_restriction()` (in module *owlapy.providers*), 97
- `owl_datatype_min_max_exclusive_restriction()` (in module *owlapy.providers*), 97
- `owl_datatype_min_max_inclusive_restriction()` (in module *owlapy.providers*), 97
- `owl_expression_to_dl()` (in module *owlapy*), 103
- `owl_expression_to_dl()` (in module *owlapy.render*), 99
- `owl_expression_to_manchester()` (in module *owlapy*), 103
- `owl_expression_to_manchester()` (in module *owlapy.render*), 99
- `owl_expression_to_sparql()` (in module *owlapy*), 103
- `owl_expression_to_sparql()` (in module *owlapy.converter*), 44
- `OwL_NAMED_INDIVIDUAL` (*owlapy.class_expression.OwLRDFVocabulary* attribute), 38
- `OwL_NAMED_INDIVIDUAL` (*owlapy.vocab.OwLRDFVocabulary* attribute), 102
- `OwL_NOTHING` (*owlapy.class_expression.OwLRDFVocabulary* attribute), 38
- `OwL_NOTHING` (*owlapy.vocab.OwLRDFVocabulary* attribute), 102
- `OwL_THING` (*owlapy.class_expression.OwLRDFVocabulary* attribute), 38
- `OwL_THING` (*owlapy.vocab.OwLRDFVocabulary* attribute), 102
- `OwL_TOP_DATA_PROPERTY` (*owlapy.class_expression.OwLRDFVocabulary* attribute), 39
- `OwL_TOP_DATA_PROPERTY` (*owlapy.vocab.OwLRDFVocabulary* attribute), 102
- `OwL_TOP_OBJECT_PROPERTY` (*owlapy.class_expression.OwLRDFVocabulary* attribute), 39
- `OwL_TOP_OBJECT_PROPERTY` (*owlapy.vocab.OwLRDFVocabulary* attribute), 102
- `OwLAnnotation` (class in *owlapy.owl_axiom*), 60
- `OwLAnnotationAssertionAxiom` (class in *owlapy.owl_axiom*), 61
- `OwLAnnotationAxiom` (class in *owlapy.owl_axiom*), 60
- `OwLAnnotationObject` (class in *owlapy.owl_annotation*), 49
- `OwLAnnotationProperty` (class in *owlapy.owl_axiom*), 60
- `OwLAnnotationPropertyDomainAxiom` (class in *owlapy.owl_axiom*), 62
- `OwLAnnotationPropertyRangeAxiom` (class in *owlapy.owl_axiom*), 62
- `OwLAnnotationSubject` (class in *owlapy.owl_annotation*), 49
- `OwLAnnotationValue` (class in *owlapy.owl_annotation*), 49
- `OwLAnonymousClassExpression` (class in *owlapy.class_expression*), 24
- `OwLAnonymousClassExpression` (class in *owlapy.class_expression.class_expression*), 6
- `owlapy`
 - module, 5
- `owlapy.class_expression`
 - module, 5
- `owlapy.class_expression.class_expression`
 - module, 5
- `owlapy.class_expression.nary_boolean_expression`
 - module, 7
- `owlapy.class_expression.owl_class`
 - module, 9
- `owlapy.class_expression.restriction`
 - module, 10
- `owlapy.converter`
 - module, 41
- `owlapy.data_ranges`
 - module, 39
- `owlapy.entities`

- module, 41
- owlapy.has
 - module, 44
- owlapy.iri
 - module, 44
- owlapy.meta_classes
 - module, 46
- owlapy.namespaces
 - module, 47
- owlapy.owl_annotation
 - module, 49
- owlapy.owl_axiom
 - module, 50
- owlapy.owl_datatype
 - module, 69
- owlapy.owl_individual
 - module, 70
- owlapy.owl_literal
 - module, 71
- owlapy.owl_object
 - module, 74
- owlapy.owl_ontology
 - module, 76
- owlapy.owl_ontology_manager
 - module, 79
- owlapy.owl_property
 - module, 81
- owlapy.owl_reasoner
 - module, 85
- owlapy.parser
 - module, 91
- owlapy.providers
 - module, 96
- owlapy.render
 - module, 97
- owlapy.util
 - module, 99
- owlapy.vocab
 - module, 101
- OWLAsymmetricObjectPropertyAxiom (*class in owlapy.owl_axiom*), 65
- OWLAxiom (*class in owlapy.owl_axiom*), 52
- OWLBooleanClassExpression (*class in owlapy.class_expression*), 25
- OWLBooleanClassExpression (*class in owlapy.class_expression.class_expression*), 7
- OWLBottomDataProperty (*in module owlapy.owl_literal*), 73
- OWLBottomObjectProperty (*in module owlapy.owl_literal*), 73
- OWLCardinalityRestriction (*class in owlapy.class_expression*), 29
- OWLCardinalityRestriction (*class in owlapy.class_expression.restriction*), 13
- OWLClass (*class in owlapy.class_expression*), 26
- OWLClass (*class in owlapy.class_expression.owl_class*), 9
- OWLClassAssertionAxiom (*class in owlapy.owl_axiom*), 59
- OWLClassAxiom (*class in owlapy.owl_axiom*), 53
- OWLClassExpression (*class in owlapy.class_expression*), 24
- OWLClassExpression (*class in owlapy.class_expression.class_expression*), 5
- OWLDataAllValuesFrom (*class in owlapy.class_expression*), 36
- OWLDataAllValuesFrom (*class in owlapy.class_expression.restriction*), 20
- OWLDataCardinalityRestriction (*class in owlapy.class_expression*), 31
- OWLDataCardinalityRestriction (*class in owlapy.class_expression.restriction*), 18
- OWLDataComplementOf (*class in owlapy.data_ranges*), 41
- OWLDataExactCardinality (*class in owlapy.class_expression*), 37
- OWLDataExactCardinality (*class in owlapy.class_expression.restriction*), 19
- OWLDataHasValue (*class in owlapy.class_expression*), 36
- OWLDataHasValue (*class in owlapy.class_expression.restriction*), 20
- OWLDataIntersectionOf (*class in owlapy.data_ranges*), 40
- OWLDataMaxCardinality (*class in owlapy.class_expression*), 37
- OWLDataMaxCardinality (*class in owlapy.class_expression.restriction*), 19
- OWLDataMinCardinality (*class in owlapy.class_expression*), 37
- OWLDataMinCardinality (*class in owlapy.class_expression.restriction*), 18
- OWLDataOneOf (*class in owlapy.class_expression*), 30

OWLDataOneOf (class in owlapy.class_expression.restriction), 21
 OWLDataProperty (class in owlapy.owl_property), 84
 OWLDataPropertyAssertionAxiom (class in owlapy.owl_axiom), 64
 OWLDataPropertyAxiom (class in owlapy.owl_axiom), 53
 OWLDataPropertyCharacteristicAxiom (class in owlapy.owl_axiom), 67
 OWLDataPropertyDomainAxiom (class in owlapy.owl_axiom), 68
 OWLDataPropertyExpression (class in owlapy.owl_property), 82
 OWLDataPropertyRangeAxiom (class in owlapy.owl_axiom), 69
 OWLDataRange (class in owlapy.data_ranges), 40
 OWLDataRestriction (class in owlapy.class_expression), 29
 OWLDataRestriction (class in owlapy.class_expression.restriction), 17
 OWLDataSomeValuesFrom (class in owlapy.class_expression), 35
 OWLDataSomeValuesFrom (class in owlapy.class_expression.restriction), 19
 OWLDatatype (class in owlapy.owl_datatype), 69
 OWLDatatypeDefinitionAxiom (class in owlapy.owl_axiom), 53
 OWLDatatypeRestriction (class in owlapy.class_expression), 33
 OWLDatatypeRestriction (class in owlapy.class_expression.restriction), 21
 OWLDataUnionOf (class in owlapy.data_ranges), 41
 OWLDeclarationAxiom (class in owlapy.owl_axiom), 53
 OWLDifferentIndividualsAxiom (class in owlapy.owl_axiom), 56
 OWLDisjointClassesAxiom (class in owlapy.owl_axiom), 55
 OWLDisjointDataPropertiesAxiom (class in owlapy.owl_axiom), 58
 OWLDisjointObjectPropertiesAxiom (class in owlapy.owl_axiom), 57
 OWLDisjointUnionAxiom (class in owlapy.owl_axiom), 59
 OWLEntity (class in owlapy.owl_object), 75
 OWLEquivalentClassesAxiom (class in owlapy.owl_axiom), 55
 OWLEquivalentDataPropertiesAxiom (class in owlapy.owl_axiom), 58
 OWLEquivalentObjectPropertiesAxiom (class in owlapy.owl_axiom), 57
 OWLFacet (class in owlapy.class_expression), 33
 OWLFacet (class in owlapy.vocab), 102
 OWLFacetRestriction (class in owlapy.class_expression), 34
 OWLFacetRestriction (class in owlapy.class_expression.restriction), 22
 OWLFunctionalDataPropertyAxiom (class in owlapy.owl_axiom), 67
 OWLFunctionalObjectPropertyAxiom (class in owlapy.owl_axiom), 65
 OWLHasKeyAxiom (class in owlapy.owl_axiom), 54
 OWLHasValueRestriction (class in owlapy.class_expression), 28
 OWLHasValueRestriction (class in owlapy.class_expression.restriction), 12
 OWLImportsDeclaration (class in owlapy.owl_ontology_manager), 80
 OWLIndividual (class in owlapy.owl_individual), 70
 OWLIndividualAxiom (class in owlapy.owl_axiom), 53
 OWLInverseFunctionalObjectPropertyAxiom (class in owlapy.owl_axiom), 66
 OWLInverseObjectPropertiesAxiom (class in owlapy.owl_axiom), 57
 OWLIrreflexiveObjectPropertyAxiom (class in owlapy.owl_axiom), 66
 OWLLiteral (class in owlapy.owl_literal), 72
 OWLLogicalAxiom (class in owlapy.owl_axiom), 52
 OWLNamedIndividual (class in owlapy.owl_individual), 70
 OWLNamedObject (class in owlapy.owl_object), 75
 OWLNaryAxiom (class in owlapy.owl_axiom), 54
 OWLNaryBooleanClassExpression (class in owlapy.class_expression), 26
 OWLNaryBooleanClassExpression (class in owlapy.class_expression.nary_boolean_expression), 8
 OWLNaryClassAxiom (class in owlapy.owl_axiom), 55
 OWLNaryDataRange (class in owlapy.data_ranges), 40
 OWLNaryIndividualAxiom (class in owlapy.owl_axiom), 56
 OWLNaryPropertyAxiom (class in owlapy.owl_axiom), 57
 OWLNegativeDataPropertyAssertionAxiom (class in owlapy.owl_axiom), 65
 OWLNegativeObjectPropertyAssertionAxiom (class in owlapy.owl_axiom), 64
 OWLNothing (in module owlapy.class_expression), 39
 OWLObject (class in owlapy.data_ranges), 39
 OWLObject (class in owlapy.owl_object), 74
 OWLObjectAllValuesFrom (class in owlapy.class_expression), 32
 OWLObjectAllValuesFrom (class in owlapy.class_expression.restriction), 15
 OWLObjectCardinalityRestriction (class in owlapy.class_expression), 29
 OWLObjectCardinalityRestriction (class in owlapy.class_expression.restriction), 14
 OWLObjectComplementOf (class in owlapy.class_expression), 25
 OWLObjectComplementOf (class in owlapy.class_expression.class_expression), 7
 OWLObjectExactCardinality (class in owlapy.class_expression), 35
 OWLObjectExactCardinality (class in owlapy.class_expression.restriction), 15
 OWLObjectHasSelf (class in owlapy.class_expression), 30

OWLObjectHasSelf (class in owlapy.class_expression.restriction), 16
 OWLObjectHasValue (class in owlapy.class_expression), 32
 OWLObjectHasValue (class in owlapy.class_expression.restriction), 16
 OWLObjectIntersectionOf (class in owlapy.class_expression), 27
 OWLObjectIntersectionOf (class in owlapy.class_expression.nary_boolean_expression), 8
 OWLObjectInverseOf (class in owlapy.owl_property), 83
 OWLObjectMaxCardinality (class in owlapy.class_expression), 34
 OWLObjectMaxCardinality (class in owlapy.class_expression.restriction), 14
 OWLObjectMinCardinality (class in owlapy.class_expression), 34
 OWLObjectMinCardinality (class in owlapy.class_expression.restriction), 14
 OWLObjectOneOf (class in owlapy.class_expression), 38
 OWLObjectOneOf (class in owlapy.class_expression.restriction), 17
 OWLObjectParser (class in owlapy.owl_object), 75
 OWLObjectProperty (class in owlapy.owl_property), 82
 OWLObjectPropertyAssertionAxiom (class in owlapy.owl_axiom), 64
 OWLObjectPropertyAxiom (class in owlapy.owl_axiom), 52
 OWLObjectPropertyCharacteristicAxiom (class in owlapy.owl_axiom), 65
 OWLObjectPropertyDomainAxiom (class in owlapy.owl_axiom), 68
 OWLObjectPropertyExpression (class in owlapy.owl_property), 82
 OWLObjectPropertyRangeAxiom (class in owlapy.owl_axiom), 69
 OWLObjectRenderer (class in owlapy.owl_object), 74
 OWLObjectRestriction (class in owlapy.class_expression), 28
 OWLObjectRestriction (class in owlapy.class_expression.restriction), 12
 OWLObjectSomeValuesFrom (class in owlapy.class_expression), 31
 OWLObjectSomeValuesFrom (class in owlapy.class_expression.restriction), 15
 OWLObjectUnionOf (class in owlapy.class_expression), 27
 OWLObjectUnionOf (class in owlapy.class_expression.nary_boolean_expression), 8
 OWLOntology (class in owlapy.owl_ontology), 76
 OWLOntologyChange (class in owlapy.owl_ontology_manager), 79
 OWLOntologyID (class in owlapy.owl_ontology), 76
 OWLOntologyManager (class in owlapy.owl_ontology_manager), 79
 OWLProperty (class in owlapy.owl_property), 82
 OWLPropertyAssertionAxiom (class in owlapy.owl_axiom), 63
 OWLPropertyAxiom (class in owlapy.owl_axiom), 52
 OWLPropertyDomainAxiom (class in owlapy.owl_axiom), 67
 OWLPropertyExpression (class in owlapy.owl_property), 81
 OWLPropertyRange (class in owlapy.data_ranges), 40
 OWLPropertyRangeAxiom (class in owlapy.owl_axiom), 68
 OWLQuantifiedDataRestriction (class in owlapy.class_expression), 31
 OWLQuantifiedDataRestriction (class in owlapy.class_expression.restriction), 18
 OWLQuantifiedObjectRestriction (class in owlapy.class_expression), 28
 OWLQuantifiedObjectRestriction (class in owlapy.class_expression.restriction), 13
 OWLQuantifiedRestriction (class in owlapy.class_expression), 28
 OWLQuantifiedRestriction (class in owlapy.class_expression.restriction), 13
 OWLRDFVocabulary (class in owlapy.class_expression), 38
 OWLRDFVocabulary (class in owlapy.vocab), 101
 OWLReasoner (class in owlapy.owl_reasoner), 85
 OWLReflexiveObjectPropertyAxiom (class in owlapy.owl_axiom), 66
 OWLRestriction (class in owlapy.class_expression), 27
 OWLRestriction (class in owlapy.class_expression.restriction), 12
 OWLSameIndividualAxiom (class in owlapy.owl_axiom), 56
 OWLSubAnnotationPropertyOfAxiom (class in owlapy.owl_axiom), 61
 OWLSubClassOfAxiom (class in owlapy.owl_axiom), 58
 OWLSubDataPropertyOfAxiom (class in owlapy.owl_axiom), 63
 OWLSubObjectPropertyOfAxiom (class in owlapy.owl_axiom), 63
 OWLSubPropertyAxiom (class in owlapy.owl_axiom), 62
 OWLSymmetricObjectPropertyAxiom (class in owlapy.owl_axiom), 66
 OWLThing (in module owlapy.class_expression), 39
 OWLTopDataProperty (in module owlapy.owl_literal), 73
 OWLTopObjectProperty (in module owlapy.owl_literal), 73
 OWLTransitiveObjectPropertyAxiom (class in owlapy.owl_axiom), 67
 OWLUnaryPropertyAxiom (class in owlapy.owl_axiom), 65

P

parent (owlapy.converter.Owl2SparqlConverter attribute), 43
 parent_var (owlapy.converter.Owl2SparqlConverter attribute), 43
 parse_boolean() (owlapy.owl_literal.OWLLiteral method), 72

parse_date() (owlapy.owl_literal.OWLLiteral method), 73
 parse_datetime() (owlapy.owl_literal.OWLLiteral method), 73
 parse_double() (owlapy.owl_literal.OWLLiteral method), 72
 parse_duration() (owlapy.owl_literal.OWLLiteral method), 73
 parse_expression() (owlapy.owl_object.OWLObjectParser method), 75
 parse_expression() (owlapy.parser.DLSyntaxParser method), 94
 parse_expression() (owlapy.parser.ManchesterOWLSyntaxParser method), 92
 parse_integer() (owlapy.owl_literal.OWLLiteral method), 72
 parse_string() (owlapy.owl_literal.OWLLiteral method), 72
 PATTERN (owlapy.class_expression.OWLFacet attribute), 34
 PATTERN (owlapy.vocab.OWLFacet attribute), 103
 peek() (in module owlapy.converter), 42
 prefix (owlapy.namespaces.Namespaces property), 48
 PREV (owlapy.util.LRUCache attribute), 101
 process() (owlapy.converter.Owl2SparqlConverter method), 43
 properties (owlapy.converter.Owl2SparqlConverter attribute), 43
 properties() (owlapy.owl_axiom.OWLNaryPropertyAxiom method), 57

R

RDF (in module owlapy.namespaces), 48
 RDFS (in module owlapy.namespaces), 48
 RDFS_LITERAL (owlapy.class_expression.OWLRDFVocabulary attribute), 39
 RDFS_LITERAL (owlapy.vocab.OWLRDFVocabulary attribute), 102
 reminder (owlapy.class_expression.owl_class.OWLClass property), 9
 reminder (owlapy.class_expression.OWLClass property), 26
 reminder (owlapy.iri.IRI property), 45
 remove_axiom() (owlapy.owl_ontology_manager.OWLOntologyManager method), 80
 render() (owlapy.converter.Owl2SparqlConverter method), 43
 render() (owlapy.owl_object.OWLObjectRenderer method), 75
 render() (owlapy.render.DLSyntaxObjectRenderer method), 98
 render() (owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer method), 98
 Restriction_Literals (in module owlapy.providers), 97
 RESULT (owlapy.util.LRUCache attribute), 101

S

same_individuals() (owlapy.owl_reasoner.OWLReasoner method), 87
 save_ontology() (owlapy.owl_ontology_manager.OWLOntologyManager method), 80
 sentinel (owlapy.util.LRUCache attribute), 101
 set_short_form_provider() (owlapy.owl_object.OWLObjectRenderer method), 74
 set_short_form_provider() (owlapy.render.DLSyntaxObjectRenderer method), 98
 set_short_form_provider() (owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer method), 98
 slots (owlapy.parser.DLSyntaxParser attribute), 94
 slots (owlapy.parser.ManchesterOWLSyntaxParser attribute), 92
 sparql (owlapy.converter.Owl2SparqlConverter attribute), 43
 stack_parent() (owlapy.converter.Owl2SparqlConverter method), 43
 stack_variable() (owlapy.converter.Owl2SparqlConverter method), 43
 str (owlapy.class_expression.owl_class.OWLClass property), 9
 str (owlapy.class_expression.OWLClass property), 26
 str (owlapy.iri.IRI property), 45
 str (owlapy.meta_classes.HasIRI property), 46
 str (owlapy.owl_axiom.OWLAnnotationProperty property), 60
 str (owlapy.owl_datatype.OWLDatatype property), 70
 str (owlapy.owl_individual.OWLNamedIndividual property), 70
 str (owlapy.owl_ontology_manager.OWLImportsDeclaration property), 80
 str (owlapy.owl_property.OWLDataProperty property), 84
 str (owlapy.owl_property.OWLObjectProperty property), 82
 STRING (owlapy.vocab.XSDVocabulary attribute), 102
 StringOWLDatatype (in module owlapy.owl_literal), 74
 sub_classes() (owlapy.owl_reasoner.OWLReasoner method), 88
 sub_data_properties() (owlapy.owl_reasoner.OWLReasoner method), 89
 sub_object_properties() (owlapy.owl_reasoner.OWLReasoner method), 89
 super_classes() (owlapy.owl_reasoner.OWLReasoner method), 90
 super_data_properties() (owlapy.owl_reasoner.OWLReasoner method), 89
 super_object_properties() (owlapy.owl_reasoner.OWLReasoner method), 90
 symbolic_form (owlapy.class_expression.OWLFacet property), 33
 symbolic_form (owlapy.vocab.OWLFacet property), 102

T

TIME_DATATYPES (in module owlapy.owl_literal), 74
 to_python() (owlapy.owl_literal.OWLLiteral method), 73
 to_string_id() (owlapy.owl_object.OWLEntity method), 75
 TopLevelCNF (class in owlapy.util), 100
 TopLevelDNF (class in owlapy.util), 100
 TopOWLDatatype (in module owlapy.owl_literal), 74
 TOTAL_DIGITS (owlapy.class_expression.OWLFacet attribute), 34
 TOTAL_DIGITS (owlapy.vocab.OWLFacet attribute), 103
 triple() (owlapy.converter.Owl2SparqlConverter method), 43
 type_index (owlapy.class_expression.class_expression.OWLObjectComplementOf attribute), 7
 type_index (owlapy.class_expression.nary_boolean_expression.OWLObjectIntersectionOf attribute), 9
 type_index (owlapy.class_expression.nary_boolean_expression.OWLObjectUnionOf attribute), 8
 type_index (owlapy.class_expression.owl_class.OWLClass attribute), 9
 type_index (owlapy.class_expression.OWLClass attribute), 26
 type_index (owlapy.class_expression.OWLDataAllValuesFrom attribute), 36
 type_index (owlapy.class_expression.OWLDataExactCardinality attribute), 37
 type_index (owlapy.class_expression.OWLDataHasValue attribute), 36
 type_index (owlapy.class_expression.OWLDataMaxCardinality attribute), 37
 type_index (owlapy.class_expression.OWLDataMinCardinality attribute), 37
 type_index (owlapy.class_expression.OWLDataOneOf attribute), 30
 type_index (owlapy.class_expression.OWLDataSomeValuesFrom attribute), 35
 type_index (owlapy.class_expression.OWLDatatypeRestriction attribute), 33
 type_index (owlapy.class_expression.OWLFacetRestriction attribute), 34
 type_index (owlapy.class_expression.OWLObjectAllValuesFrom attribute), 32
 type_index (owlapy.class_expression.OWLObjectComplementOf attribute), 25
 type_index (owlapy.class_expression.OWLObjectExactCardinality attribute), 35
 type_index (owlapy.class_expression.OWLObjectHasSelf attribute), 30
 type_index (owlapy.class_expression.OWLObjectHasValue attribute), 33
 type_index (owlapy.class_expression.OWLObjectIntersectionOf attribute), 27
 type_index (owlapy.class_expression.OWLObjectMaxCardinality attribute), 35
 type_index (owlapy.class_expression.OWLObjectMinCardinality attribute), 34
 type_index (owlapy.class_expression.OWLObjectOneOf attribute), 38
 type_index (owlapy.class_expression.OWLObjectSomeValuesFrom attribute), 32
 type_index (owlapy.class_expression.OWLObjectUnionOf attribute), 27
 type_index (owlapy.class_expression.restriction.OWLDataAllValuesFrom attribute), 20
 type_index (owlapy.class_expression.restriction.OWLDataExactCardinality attribute), 19
 type_index (owlapy.class_expression.restriction.OWLDataHasValue attribute), 20
 type_index (owlapy.class_expression.restriction.OWLDataMaxCardinality attribute), 19
 type_index (owlapy.class_expression.restriction.OWLDataMinCardinality attribute), 18
 type_index (owlapy.class_expression.restriction.OWLDataOneOf attribute), 21
 type_index (owlapy.class_expression.restriction.OWLDataSomeValuesFrom attribute), 19
 type_index (owlapy.class_expression.restriction.OWLDatatypeRestriction attribute), 21
 type_index (owlapy.class_expression.restriction.OWLFacetRestriction attribute), 22
 type_index (owlapy.class_expression.restriction.OWLObjectAllValuesFrom attribute), 16
 type_index (owlapy.class_expression.restriction.OWLObjectExactCardinality attribute), 15
 type_index (owlapy.class_expression.restriction.OWLObjectHasSelf attribute), 16
 type_index (owlapy.class_expression.restriction.OWLObjectHasValue attribute), 16
 type_index (owlapy.class_expression.restriction.OWLObjectMaxCardinality attribute), 14
 type_index (owlapy.class_expression.restriction.OWLObjectMinCardinality attribute), 14
 type_index (owlapy.class_expression.restriction.OWLObjectOneOf attribute), 17
 type_index (owlapy.class_expression.restriction.OWLObjectSomeValuesFrom attribute), 15
 type_index (owlapy.data_ranges.OWLDataComplementOf attribute), 41
 type_index (owlapy.data_ranges.OWLDataIntersectionOf attribute), 40
 type_index (owlapy.data_ranges.OWLDataUnionOf attribute), 41
 type_index (owlapy.has.HasIndex attribute), 44
 type_index (owlapy.iri.IRI attribute), 45
 type_index (owlapy.owl_datatype.OWLDatatype attribute), 70
 type_index (owlapy.owl_individual.OWLNamedIndividual attribute), 71
 type_index (owlapy.owl_literal.OWLLiteral attribute), 72
 type_index (owlapy.owl_ontology.OWLOntology attribute), 77
 type_index (owlapy.owl_property.OWLDataProperty attribute), 84
 type_index (owlapy.owl_property.OWLObjectInverseOf attribute), 83
 type_index (owlapy.owl_property.OWLObjectProperty attribute), 83
 types() (owlapy.owl_reasoner.OWLReasoner method), 90

V

`values()` (*owlapy.class_expression.OWLDataOneOf method*), 30
`values()` (*owlapy.class_expression.restriction.OWLDataOneOf method*), 21
`variable_entities` (*owlapy.converter.Owl2SparqlConverter attribute*), 43
`variables` (*owlapy.converter.Owl2SparqlConverter attribute*), 43
`VariablesMapping` (*class in owlapy.converter*), 42
`visit_abbreviated_iri()` (*owlapy.parser.DLSyntaxParser method*), 96
`visit_abbreviated_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 94
`visit_boolean_literal()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_boolean_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_cardinality_res()` (*owlapy.parser.DLSyntaxParser method*), 94
`visit_cardinality_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_class_expression()` (*owlapy.parser.DLSyntaxParser method*), 94
`visit_class_expression()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_class_iri()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_class_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_data_cardinality_res()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_data_cardinality_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_data_intersection()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_data_intersection()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_data_parentheses()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_data_parentheses()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_data_primary()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_data_primary()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_data_property_iri()` (*owlapy.parser.DLSyntaxParser method*), 96
`visit_data_property_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_data_some_only_res()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_data_some_only_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_data_union()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_data_union()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_data_value_res()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_data_value_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_datatype()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_datatype()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_datatype_iri()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_datatype_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_datatype_restriction()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_datatype_restriction()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_date_literal()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_date_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_datetime_literal()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_datetime_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_decimal_literal()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_decimal_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_duration_literal()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_duration_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_facet()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_facet()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_facet_restriction()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_facet_restriction()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_facet_restrictions()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_facet_restrictions()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_float_literal()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_float_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_full_iri()` (*owlapy.parser.DLSyntaxParser method*), 96
`visit_full_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 94
`visit_has_self()` (*owlapy.parser.DLSyntaxParser method*), 94
`visit_has_self()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_individual_iri()` (*owlapy.parser.DLSyntaxParser method*), 96
`visit_individual_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_individual_list()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_individual_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_integer_literal()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_integer_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_intersection()` (*owlapy.parser.DLSyntaxParser method*), 94
`visit_intersection()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_iri()` (*owlapy.parser.DLSyntaxParser method*), 96
`visit_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 94

`visit_literal()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_literal_list()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_literal_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_non_negative_integer()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_non_negative_integer()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_object_property()` (*owlapy.parser.DLSyntaxParser method*), 94
`visit_object_property()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_object_property_iri()` (*owlapy.parser.DLSyntaxParser method*), 96
`visit_object_property_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_parentheses()` (*owlapy.parser.DLSyntaxParser method*), 96
`visit_parentheses()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 94
`visit_primary()` (*owlapy.parser.DLSyntaxParser method*), 94
`visit_primary()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_quoted_string()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_quoted_string()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_simple_iri()` (*owlapy.parser.DLSyntaxParser method*), 96
`visit_simple_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 94
`visit_some_only_res()` (*owlapy.parser.DLSyntaxParser method*), 94
`visit_some_only_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_string_literal_language()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_string_literal_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_string_literal_no_language()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_string_literal_no_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_typed_literal()` (*owlapy.parser.DLSyntaxParser method*), 95
`visit_typed_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 93
`visit_union()` (*owlapy.parser.DLSyntaxParser method*), 94
`visit_union()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92
`visit_value_res()` (*owlapy.parser.DLSyntaxParser method*), 94
`visit_value_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 92

X

XSD (*in module owlapy.namespaces*), 48
XSDVocabulary (*class in owlapy.vocab*), 102