

Contents:

1	OWLAPY	1
2	owlapy	1
2.1	Subpackages	1
2.2	Submodules	59
2.3	Package Contents	82
	Python Module Index	83
	Index	84

---

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

1 OWLAPY

placeholder

2 owlapy

2.1 Subpackages

owlapy.model

@TODO: CD: This is not a python code. We should refactor this model module.

---

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

## Submodules

### `owlapy.model.providers`

OWL Datatype restriction constructors.

## Module Contents

### Functions

<code>OWLDatatypeMaxExclusiveRestriction(...)</code>	Create a max exclusive restriction.
<code>OWLDatatypeMinExclusiveRestriction(...)</code>	Create a min exclusive restriction.
<code>OWLDatatypeMaxInclusiveRestriction(...)</code>	Create a max inclusive restriction.
<code>OWLDatatypeMinInclusiveRestriction(...)</code>	Create a min inclusive restriction.
<code>OWLDatatypeMinMaxExclusiveRestriction(...)</code>	Create a min-max exclusive restriction.
<code>OWLDatatypeMinMaxInclusiveRestriction(...)</code>	Create a min-max inclusive restriction.

### Attributes

`Restriction_Literals`

`owlapy.model.providers.Restriction_Literals`

`owlapy.model.providers.OWLDatatypeMaxExclusiveRestriction (max_: Restriction_Literals) → owlapy.model.OWLDatatypeRestriction`  
Create a max exclusive restriction.

`owlapy.model.providers.OWLDatatypeMinExclusiveRestriction (min_: Restriction_Literals) → owlapy.model.OWLDatatypeRestriction`  
Create a min exclusive restriction.

`owlapy.model.providers.OWLDatatypeMaxInclusiveRestriction (max_: Restriction_Literals) → owlapy.model.OWLDatatypeRestriction`  
Create a max inclusive restriction.

`owlapy.model.providers.OWLDatatypeMinInclusiveRestriction (min_: Restriction_Literals) → owlapy.model.OWLDatatypeRestriction`  
Create a min inclusive restriction.

`owlapy.model.providers.OWLDatatypeMinMaxExclusiveRestriction (min_: Restriction_Literals, max_: Restriction_Literals) → owlapy.model.OWLDatatypeRestriction`  
Create a min-max exclusive restriction.

`owlapy.model.providers.OWLDatatypeMinMaxInclusiveRestriction (min_: Restriction_Literals, max_: Restriction_Literals) → owlapy.model.OWLDatatypeRestriction`  
Create a min-max inclusive restriction.

## Package Contents

### Classes

<i>OWLRDFVocabulary</i>	Enumerations for OWL/RDF vocabulary.
<i>XSDVocabulary</i>	Enumerations for XSD vocabulary.
<i>OWLFacet</i>	Enumerations for OWL facets.
<i>OWLObject</i>	Base interface for OWL objects
<i>OWLEntity</i>	Represents Entities in the OWL 2 Specification.
<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.
<i>IRI</i>	An IRI, consisting of a namespace and a remainder.
<i>HasIndex</i>	Interface for types with an index; this is used to group objects by type when sorting.
<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>OWLClassExpression</i>	An OWL 2 Class Expression.
<i>OWLObjectComplementOf</i>	Represents an ObjectComplementOf class expression in the OWL 2 Specification.
<i>OWLAnonymousClassExpression</i>	A Class Expression which is not a named Class.
<i>OWLBooleanClassExpression</i>	Represent an anonymous boolean class expression.
<i>OWLPropertyRange</i>	OWL Objects that can be the ranges of properties.
<i>OWLDaRange</i>	Represents a DaRange in the OWL 2 Specification.
<i>OWLClass</i>	An OWL 2 named Class
<i>OWLObjectPropertyExpression</i>	A high level interface to describe different types of object properties.
<i>OWLProperty</i>	A marker interface for properties that aren't expression i.e. named properties. By definition, properties
<i>OWLPropertyExpression</i>	Represents a property or possibly the inverse of a property.
<i>OWLDaPropertyExpression</i>	A high level interface to describe different types of data properties.
<i>OWLDaProperty</i>	Represents a Data Property in the OWL 2 Specification.
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification.
<i>OWLRestriction</i>	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
<i>OWLDaRestriction</i>	Represents a Data Property Restriction in the OWL 2 specification.
<i>OWLObjectRestriction</i>	Represents a Object Property Restriction in the OWL 2 specification.
<i>OWLHasValueRestriction</i>	OWLHasValueRestriction.
<i>OWLQuantifiedRestriction</i>	Represents a quantified restriction.
<i>OWLQuantifiedObjectRestriction</i>	Represents a quantified object restriction.
<i>OWLObjectSomeValuesFrom</i>	Represents an ObjectSomeValuesFrom class expression in the OWL 2 Specification.

continues on next page

Table 1 – continued from previous page

<i>OWLObjectAllValuesFrom</i>	Represents an ObjectAllValuesFrom class expression in the OWL 2 Specification.
<i>OWLNaryBooleanClassExpression</i>	OWLNaryBooleanClassExpression.
<i>OWLObjectUnionOf</i>	Represents an ObjectUnionOf class expression in the OWL 2 Specification.
<i>OWLObjectIntersectionOf</i>	Represents an OWLObjectIntersectionOf class expression in the OWL 2 Specification.
<i>HasCardinality</i>	An interface to objects that have a cardinality.
<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>OWLObjectMinCardinality</i>	Represents a ObjectMinCardinality restriction in the OWL 2 Specification.
<i>OWLObjectMaxCardinality</i>	Represents a ObjectMaxCardinality restriction in the OWL 2 Specification.
<i>OWLObjectExactCardinality</i>	Represents an ObjectExactCardinality restriction in the OWL 2 Specification.
<i>OWLObjectHasSelf</i>	Represents an ObjectHasSelf class expression in the OWL 2 Specification.
<i>OWLIndividual</i>	Represents a named or anonymous individual.
<i>OWLObjectHasValue</i>	Represents an ObjectHasValue class expression in the OWL 2 Specification.
<i>OWLObjectOneOf</i>	Represents an ObjectOneOf class expression in the OWL 2 Specification.
<i>OWLNamedIndividual</i>	Represents a Named Individual in the OWL 2 Specification.
<i>OWLOntologyID</i>	An object that identifies an ontology. Since OWL 2, ontologies do not have to have an ontology IRI, or if they
<i>OWLAxiom</i>	Represents Axioms in the OWL 2 Specification.
<i>OWLDatatype</i>	Represents a Datatype (named data range) in the OWL 2 Specification.
<i>OWLDatatypeRestriction</i>	Represents a DatatypeRestriction data range in the OWL 2 Specification.
<i>OWLFacetRestriction</i>	A facet restriction is used to restrict a particular datatype.
<i>OWLLiteral</i>	Represents a Literal in the OWL 2 Specification.
<i>OWLQuantifiedDataRestriction</i>	Represents a quantified data restriction.
<i>OWLDataCardinalityRestriction</i>	Represents Data Property Cardinality Restrictions in the OWL 2 specification.
<i>OWLDataAllValuesFrom</i>	Represents DataAllValuesFrom class expressions in the OWL 2 Specification.
<i>OWLDataComplementOf</i>	Represents DataComplementOf in the OWL 2 Specification.
<i>OWLDataExactCardinality</i>	Represents DataExactCardinality restrictions in the OWL 2 Specification.
<i>OWLDataHasValue</i>	Represents DataHasValue restrictions in the OWL 2 Specification.
<i>OWLDataMaxCardinality</i>	Represents DataMaxCardinality restrictions in the OWL 2 Specification.
<i>OWLDataMinCardinality</i>	Represents DataMinCardinality restrictions in the OWL 2 Specification.
<i>OWLDataOneOf</i>	Represents DataOneOf in the OWL 2 Specification.

continues on next page

Table 1 – continued from previous page

<i>OWLDataSomeValuesFrom</i>	Represents a DataSomeValuesFrom restriction in the OWL 2 Specification.
<i>OWLNaryDataRange</i>	OWLNaryDataRange.
<i>OWLDataUnionOf</i>	Represents a DataUnionOf data range in the OWL 2 Specification.
<i>OWLDataIntersectionOf</i>	Represents DataIntersectionOf in the OWL 2 Specification.
<i>OWLImportsDeclaration</i>	Represents an import statement in an ontology.
<i>OWLLogicalAxiom</i>	A base interface of all axioms that affect the logical meaning of an ontology. This excludes declaration axioms
<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>	Represents a DatatypeDefinition axiom in the OWL 2 Specification.
<i>OWLHasKeyAxiom</i>	Represents a HasKey axiom in the OWL 2 Specification.
<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>	Represents an EquivalentClasses axiom in the OWL 2 Specification.
<i>OWLDisjointClassesAxiom</i>	Represents a DisjointClasses axiom in the OWL 2 Specification.
<i>OWLNaryIndividualAxiom</i>	Represents an axiom that contains two or more operands that could also be represented with
<i>OWLDifferentIndividualsAxiom</i>	Represents a DifferentIndividuals axiom in the OWL 2 Specification.
<i>OWLSameIndividualAxiom</i>	Represents a SameIndividual axiom in the OWL 2 Specification.
<i>OWLNaryPropertyAxiom</i>	Represents an axiom that contains two or more operands that could also be represented with
<i>OWLEquivalentObjectPropertiesAxiom</i>	Represents EquivalentObjectProperties axioms in the OWL 2 Specification.
<i>OWLDisjointObjectPropertiesAxiom</i>	Represents DisjointObjectProperties axioms in the OWL 2 Specification.
<i>OWLInverseObjectPropertiesAxiom</i>	Represents InverseObjectProperties axioms in the OWL 2 Specification.
<i>OWLEquivalentDataPropertiesAxiom</i>	Represents EquivalentDataProperties axioms in the OWL 2 Specification.
<i>OWLDisjointDataPropertiesAxiom</i>	Represents DisjointDataProperties axioms in the OWL 2 Specification.
<i>OWLSubClassOfAxiom</i>	Represents an SubClassOf axiom in the OWL 2 Specification.
<i>OWLDisjointUnionAxiom</i>	Represents a DisjointUnion axiom in the OWL 2 Specification.

continues on next page

Table 1 – continued from previous page

<i>OWLClassAssertionAxiom</i>	Represents ClassAssertion axioms in the OWL 2 Specification.
<i>OWLAnnotationAxiom</i>	A super interface for annotation axioms.
<i>OWLAnnotationProperty</i>	Represents an AnnotationProperty 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>OWLAnnotationAssertionAxiom</i>	Represents AnnotationAssertion axioms in the OWL 2 specification.
<i>OWLSubAnnotationPropertyOfAxiom</i>	Represents an SubAnnotationPropertyOf axiom in the OWL 2 specification.
<i>OWLAnnotationPropertyDomainAxiom</i>	Represents an AnnotationPropertyDomain axiom in the OWL 2 specification.
<i>OWLAnnotationPropertyRangeAxiom</i>	Represents an AnnotationPropertyRange axiom in the OWL 2 specification.
<i>OWLSubPropertyAxiom</i>	Base interface for object and data sub-property axioms.
<i>OWLSubObjectPropertyOfAxiom</i>	Represents a SubObjectPropertyOf axiom in the OWL 2 specification.
<i>OWLSubDataPropertyOfAxiom</i>	Represents a SubDataPropertyOf axiom in the OWL 2 specification.
<i>OWLPropertyAssertionAxiom</i>	Represents a PropertyAssertion axiom in the OWL 2 specification.
<i>OWLObjectPropertyAssertionAxiom</i>	Represents an ObjectPropertyAssertion axiom in the OWL 2 specification.
<i>OWLNegativeObjectPropertyAssertionAxiom</i>	Represents a NegativeObjectPropertyAssertion axiom in the OWL 2 specification.
<i>OWLDataPropertyAssertionAxiom</i>	Represents an DataPropertyAssertion axiom in the OWL 2 specification.
<i>OWLNegativeDataPropertyAssertionAxiom</i>	Represents an NegativeDataPropertyAssertion axiom in the OWL 2 specification.
<i>OWLUnaryPropertyAxiom</i>	Unary property axiom.
<i>OWLObjectPropertyCharacteristicAxiom</i>	Base interface for functional object property axiom.
<i>OWLFunctionalObjectPropertyAxiom</i>	Represents FunctionalObjectProperty axioms in the OWL 2 specification.
<i>OWLASymmetricObjectPropertyAxiom</i>	Represents AsymmetricObjectProperty axioms in the OWL 2 specification.
<i>OWLInverseFunctionalObjectPropertyAxiom</i>	Represents InverseFunctionalObjectProperty axioms in the OWL 2 specification.
<i>OWLIrreflexiveObjectPropertyAxiom</i>	Represents IrreflexiveObjectProperty axioms in the OWL 2 specification.
<i>OWLReflexiveObjectPropertyAxiom</i>	Represents ReflexiveObjectProperty axioms in the OWL 2 specification.
<i>OWLSymmetricObjectPropertyAxiom</i>	Represents SymmetricObjectProperty axioms in the OWL 2 specification.
<i>OWLTransitiveObjectPropertyAxiom</i>	Represents TransitiveObjectProperty axioms in the OWL 2 specification.
<i>OWLDataPropertyCharacteristicAxiom</i>	Base interface for Functional data property axiom.
<i>OWLFunctionalDataPropertyAxiom</i>	Represents FunctionalDataProperty axioms in the OWL 2 specification.
<i>OWLPropertyDomainAxiom</i>	Represents ObjectPropertyDomain axioms in the OWL 2 specification.

continues on next page

Table 1 – continued from previous page

<i>OWLPropertyRangeAxiom</i>	Represents ObjectPropertyRange axioms in the OWL 2 specification.
<i>OWLObjectPropertyDomainAxiom</i>	Represents a ObjectPropertyDomain axiom in the OWL 2 Specification.
<i>OWLDataPropertyDomainAxiom</i>	Represents a DataPropertyDomain axiom in the OWL 2 Specification.
<i>OWLObjectPropertyRangeAxiom</i>	Represents a ObjectPropertyRange axiom in the OWL 2 Specification.
<i>OWLDataPropertyRangeAxiom</i>	Represents a DataPropertyRange axiom in the OWL 2 Specification.
<i>OWLOntology</i>	Represents an OWL 2 Ontology in the OWL 2 specification.
<i>OWLOntologyChange</i>	Represents an ontology change.
<i>AddImport</i>	Represents an ontology change where an import statement is added to an ontology.
<i>OWLOntologyManager</i>	An OWLOntologyManager manages a set of ontologies. It is the main point for creating, loading and accessing
<i>OWLReasoner</i>	An OWLReasoner reasons over a set of axioms (the set of reasoner axioms) that is based on the imports closure of

## Functions

<i>MOVE(*args)</i>	"Move" an imported class to the current module by setting the classes <code>__module__</code> attribute.
--------------------	--

## Attributes

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

```
class owlapy.model.OWLRDFVocabulary (namespace: owlapy.namespaces.Namespaces,  
    remainder: str)  
    Bases: _Vocabulary, enum.Enum  
    Enumerations for OWL/RDF vocabulary.  
    OWL_THING = ()  
    OWL_NOHING = ()  
    OWL_CLASS = ()  
    OWL_NAMED_INDIVIDUAL = ()  
    OWL_TOP_OBJECT_PROPERTY = ()  
    OWL_BOTTOM_OBJECT_PROPERTY = ()
```



```

OWL_TOP_DATA_PROPERTY = ()

OWL_BOTTOM_DATA_PROPERTY = ()

RDFS_LITERAL = ()

class owlapy.model.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.model.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
```

```
owlapy.model.MOVE(*args)
```

“Move” an imported class to the current module by setting the classes `__module__` attribute.

This is useful for documentation purposes to hide internal packages in sphinx.

#### Parameters

**args** – List of classes to move.

```
class owlapy.model.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.model.OWLEntity
```

Bases: *OWLNamedObject*

Represents Entities in the OWL 2 Specification.

```
__slots__ = ()
```

```
to_string_id() → str
```

```
is_anonymous() → bool
```

```
class owlapy.model.OWLAnnotationObject
```

Bases: *owlapy.owlobject.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.model.OWLAnnotationSubject
```

Bases: *OWLAnnotationObject*

A marker interface for annotation subjects, which can either be IRIs or anonymous individuals

```

__slots__ = ()

class owlapy.model.OWLAnnotationValue
    Bases: OWLAnnotationObject

    A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.

    __slots__ = ()

    is_literal() → bool

        Returns
            true if the annotation value is a literal

    as_literal() → OWLLiteral | None

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

class owlapy.model.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.

**class** owlapy.model.**HasIndex**

Bases: Protocol

Interface for types with an index; this is used to group objects by type when sorting.

**type\_index**: ClassVar[int]

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

Return self==value.

**class** owlapy.model.**HasIRI**

Simple class to access the IRI.

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

**abstract get\_iri** () → *IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**class** owlapy.model.**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.model.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.model.OWLClassExpression
    Bases: OWLPropertyRange
    An OWL 2 Class Expression.

__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.model.OWLObjectComplementOf (op: OWLClassExpression)
    Bases: OWLBooleanClassExpression, owlapy.has.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.model.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.model.OWLBooleanClassExpression
    Bases: OWLANonymousClassExpression
    Represent an anonymous boolean class expression.
    __slots__ = ()

class owlapy.model.OWLPropertyRange
    Bases: owlapy.owlobject.OWLObject
    OWL Objects that can be the ranges of properties.

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

class owlapy.model.OWLClass (iri: IRI)
    Bases: OWLClassExpression, owlapy.owlobject.OWLEntity
    An OWL 2 named Class
    property str
    property reminder: str
        The reminder of the IRI
    __slots__ = ('_iri', '_is_nothing', '_is_thing')
    type_index: Final = 1001
    get_iri () → IRI
        Gets the IRI of this object.
        Returns
            The IRI of this object.
    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 () → 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.model.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.model.OWLProperty

Bases: *OWLPropertyExpression*, *owlapy.owlobject.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.model.OWLPropertyExpression

Bases: *owlapy.owlobject.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.model.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.model.OWLDataProperty(iri: owlapy.iri.IRI)
```

Bases: *OWLDataPropertyExpression*, *OWLProperty*

Represents a Data Property in the OWL 2 Specification.

```
__slots__ = '_iri'
```

```
type_index: Final = 1004
```

```
get_iri() → owlapy.iri.IRI
```

Gets the IRI of this object.

#### Returns

The IRI of this object.

```
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.model.OWLObjectProperty(iri: owlapy.iri.IRI | str)
```

Bases: *OWLObjectPropertyExpression*, *OWLProperty*

Represents an Object Property in the OWL 2 Specification.

```
property str: str
```

```
property iri: str
```

```
__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.

**get\_iri()** → *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**is\_owl\_top\_object\_property()** → bool

Determines if this is the owl:topObjectProperty.

**Returns**

topObjectProperty.

**Return type**

True if this property is the owl

owlapy.model.Literals

**class owlapy.model.OWLRestriction**

Bases: *owlapy.owl\_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.model.OWLDataRestriction**

Bases: *OWLRestriction*

Represents a Data Property Restriction in the OWL 2 specification.

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

**is\_data\_restriction()** → bool

Determines if this is a data restriction.

**Returns**

True if this is a data restriction.

**class owlapy.model.OWLObjectRestriction**

Bases: *OWLRestriction*

Represents a 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.model.**OWLHasValueRestriction**(value: *\_T*)  
Bases: *Generic[\_T]*, *OWLRestriction*, *owlapy.has.HasFiller[\_T]*  
OWLHasValueRestriction.

**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.model.**OWLQuantifiedRestriction**  
Bases: *Generic[\_T]*, *OWLRestriction*, *owlapy.has.HasFiller[\_T]*  
Represents a quantified restriction.

**Parameters**  
*\_T* – value type

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

**class** owlapy.model.**OWLQuantifiedObjectRestriction**(  
*filler: owlapy.owl\_class\_expression.OWLClassExpression*)  
Bases: *OWLQuantifiedRestriction[owlapy.owl\_class\_expression.OWLClassExpression]*, *OWLObjectRestriction*  
Represents a quantified object restriction.

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

**get\_filler**() → *owlapy.owl\_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.model.OwlObjectSomeValuesFrom (
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.owl_class_expression.OwlClassExpression)
Bases: OwlQuantifiedObjectRestriction
Represents an ObjectSomeValuesFrom class expression in the OWL 2 Specification.
__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.model.OwlObjectAllValuesFrom (
    property: owlapy.owl_property.OwlObjectPropertyExpression,
    filler: owlapy.owl_class_expression.OwlClassExpression)
Bases: OwlQuantifiedObjectRestriction
Represents an ObjectAllValuesFrom class expression in the OWL 2 Specification.
__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.model.OwlNaryBooleanClassExpression (
    operands: Iterable[owlapy.owl_class_expression.OwlClassExpression])
Bases: owlapy.owl_class_expression.OwlBooleanClassExpression, owlapy.has.HasOperands[owlapy.owl_class_expression.OwlClassExpression]
OwlNaryBooleanClassExpression.
__slots__ = ()

```

**operands** () → Iterable[*owlapy.owl\_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.model.**OWLObjectUnionOf** (  
     *operands: Iterable[owlapy.owl\_class\_expression.OWLClassExpression]*)  
 Bases: *OWLNaryBooleanClassExpression*  
 Represents an ObjectUnionOf class expression in the OWL 2 Specification.

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

**type\_index**: Final = 3002

**class** owlapy.model.**OWLObjectIntersectionOf** (  
     *operands: Iterable[owlapy.owl\_class\_expression.OWLClassExpression]*)  
 Bases: *OWLNaryBooleanClassExpression*  
 Represents an OWLObjectIntersectionOf class expression in the OWL 2 Specification.

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

**type\_index**: Final = 3001

**class** owlapy.model.**HasCardinality**  
 An interface to objects that have a cardinality.

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

**abstract get\_cardinality** () → int  
 Gets the cardinality of a restriction.

**Returns**  
 The cardinality. A non-negative integer.

**class** owlapy.model.**OWLCardinalityRestriction** (*cardinality: int, filler: \_F*)  
 Bases: Generic[\_F], *OWLQuantifiedRestriction[\_F]*, *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.model.OwlObjectCardinalityRestriction (cardinality: int,  
    property: owlapy.owl_property.OwlObjectPropertyExpression,  
    filler: owlapy.owl_class_expression.OwlClassExpression)
```

Bases: *OwlCardinalityRestriction[owlapy.owl\_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.model.OwlObjectMinCardinality (cardinality: int,  
    property: owlapy.owl_property.OwlObjectPropertyExpression,  
    filler: owlapy.owl_class_expression.OwlClassExpression)
```

Bases: *OwlObjectCardinalityRestriction*

Represents a ObjectMinCardinality restriction in the OWL 2 Specification.

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

```
type_index: Final = 3008
```

```
class owlapy.model.OwlObjectMaxCardinality (cardinality: int,  
    property: owlapy.owl_property.OwlObjectPropertyExpression,  
    filler: owlapy.owl_class_expression.OwlClassExpression)
```

Bases: *OwlObjectCardinalityRestriction*

Represents a ObjectMaxCardinality restriction in the OWL 2 Specification.

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

```
type_index: Final = 3010
```

```
class owlapy.model.OwlObjectExactCardinality (cardinality: int,  
    property: owlapy.owl_property.OwlObjectPropertyExpression,  
    filler: owlapy.owl_class_expression.OwlClassExpression)
```

Bases: *OwlObjectCardinalityRestriction*

Represents an ObjectExactCardinality restriction in the OWL 2 Specification.

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

**type\_index:** Final = 3009

**as\_intersection\_of\_min\_max()** → *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.model.**OWLObjectHasSelf**(  
    *property: owlapy.owl\_property.OWLObjectPropertyExpression*)

Bases: *OWLObjectRestriction*

Represents an ObjectHasSelf class expression in the OWL 2 Specification.

**\_\_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.model.**OWLIndividual**

Bases: *owlapy.owl\_object.OWLObject*

Represents a named or anonymous individual.

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

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

Bases: *OWLHasValueRestriction[OWLIndividual], OWLObjectRestriction*

Represents an ObjectHasValue class expression in the OWL 2 Specification.

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

**type\_index:** Final = 3007

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

**Returns**

Property being restricted.

**as\_some\_values\_from()** → *owlapy.owl\_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.model.OwlObjectOneOf (values: *OwlIndividual* | *Iterable*[*OwlIndividual*])

Bases: *owlapy.owl\_class\_expression.OwlAnonymousClassExpression*, *owlapy.has.HasOperands*[*OwlIndividual*]

Represents an ObjectOneOf class expression in the OWL 2 Specification.

```

__slots__ = '_values'

type_index: Final = 3004

```

**individuals** () → *Iterable*[*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*[*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.owl\_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.model.OwlNamedIndividual (iri: *owlapy.iri.IRI* | *str*)

Bases: *OwlIndividual*, *owlapy.owlobject.OwlEntity*

Represents a Named Individual in the OWL 2 Specification.

```

property iri
property str

__slots__ = '_iri'

type_index: Final = 1005

```

**get\_iri** () → *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.



```
class owlapy.model.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.model.OWLXiom (annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *owlapy.owlobject.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.model.OWLDatatype (iri: owlapy.iri.IRI | owlapy.has.HasIRI)
```

Bases: *owlapy.owlobject.OWLEntity*, *owlapy.owl\_class\_expression.OWLDataRange*

Represents a Datatype (named data range) in the OWL 2 Specification.

```

__slots__ = '_iri'

type_index: Final = 4001

get_iri() → owlapy.iri.IRI
    Gets the IRI of this object.

    Returns
    The IRI of this object.

class owlapy.model.OWLDatatypeRestriction (type_: OWLDatatype,
    facet_restrictions: OWLFacetRestriction | Iterable[OWLFacetRestriction])
    Bases: owlapy.owl_class_expression.OWLDataRange
    Represents a DatatypeRestriction data range in the OWL 2 Specification.
    __slots__ = ('_type', '_facet_restrictions')
    type_index: Final = 4006
    get_datatype() → OWLDatatype
    get_facet_restrictions() → Sequence[OWLFacetRestriction]
    __eq__(other)
        Return self==value.
    __hash__()
        Return hash(self).
    __repr__()
        Return repr(self).

class owlapy.model.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() → OWLLiteral
    __eq__(other)
        Return self==value.
    __hash__()
        Return hash(self).
    __repr__()
        Return repr(self).

class owlapy.model.OWLLiteral
    Bases: owlapy.owl_annotation.OWLAnnotationValue
    Represents a Literal in the OWL 2 Specification.

```

**\_\_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()** → *OWLDatatype*  
Gets the OWLDatatype which types this literal.

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

**class** owlapy.model.OWLQuantifiedDataRestriction(  
    *filler: owlapy.owl\_class\_expression.OWLDataRange*)  
Bases: *OWLQuantifiedRestriction[owlapy.owl\_class\_expression.OWLDataRange], OWLDataRestriction*  
Represents a quantified data restriction.

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

**get\_filler()** → *owlapy.owl\_class\_expression.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.model.OWLDataCardinalityRestriction(*cardinality: int,*  
    *property: owlapy.owl\_property.OWLDataPropertyExpression,*  
    *filler: owlapy.owl\_class\_expression.OWLDataRange*)  
Bases: *OWLCardinalityRestriction[owlapy.owl\_class\_expression.OWLDataRange], OWLQuantifiedDataRestriction, OWLDataRestriction*

Represents Data Property Cardinality Restrictions in the OWL 2 specification.

**\_\_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.model.OWLDataAllValuesFrom(  
    property: owlapy.owl_property.OWLDataPropertyExpression,  
    filler: owlapy.owl_class_expression.OWLDataRange)
```

Bases: *OWLQuantifiedDataRestriction*

Represents DataAllValuesFrom class expressions in the OWL 2 Specification.

**\_\_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.model.OWLDataComplementOf(  
    data_range: owlapy.owl_class_expression.OWLDataRange)
```

Bases: *owlapy.owl\_class\_expression.OWLDataRange*

Represents DataComplementOf in the OWL 2 Specification.

**type\_index: Final** = 4002

**get\_data\_range**() → *owlapy.owl\_class\_expression.OWLDataRange*

**Returns**

The wrapped data range.

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

Return repr(self).

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

Return self==value.

```

__hash__()
    Return hash(self).

class owlapy.model.OWLDataExactCardinality (cardinality: int,
        property: owlapy.owl_property.OWLDataPropertyExpression,
        filler: owlapy.owl_class_expression.OWLDataRange)
    Bases: OWLDataCardinalityRestriction
    Represents DataExactCardinality restrictions in the OWL 2 Specification.
    __slots__ = ('_cardinality', '_filler', '_property')
    type_index: Final = 3016

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

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

class owlapy.model.OWLDataHasValue (
    property: owlapy.owl_property.OWLDataPropertyExpression, value: OWLLiteral)
    Bases: OWLHasValueRestriction[OWLLiteral], OWLDataRestriction
    Represents DataHasValue restrictions in the OWL 2 Specification.
    __slots__ = '_property'
    type_index: Final = 3014

    __repr__()
        Return repr(self).

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    as_some_values_from() → owlapy.owl_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.model.OWLDataMaxCardinality (cardinality: int,
        property: owlapy.owl_property.OWLDataPropertyExpression,
        filler: owlapy.owl_class_expression.OWLDataRange)
    Bases: OWLDataCardinalityRestriction
    Represents DataMaxCardinality restrictions in the OWL 2 Specification.
    __slots__ = ('_cardinality', '_filler', '_property')
    type_index: Final = 3017

```

```

class owlapy.model.OWLDataMinCardinality (cardinality: int,
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_class_expression.OWLDataRange)
    Bases: OWLDataCardinalityRestriction
    Represents DataMinCardinality restrictions in the OWL 2 Specification.
    __slots__ = ('_cardinality', '_filler', '_property')
    type_index: Final = 3015

class owlapy.model.OWLDataOneOf (values: OWLLiteral | Iterable[OWLLiteral])
    Bases: owlapy.owl_class_expression.OWLDataRange, owlapy.has.HasOperands[OWLLiteral]
    Represents DataOneOf in the OWL 2 Specification.
    type_index: Final = 4003

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

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

    operands () → Iterable[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.model.OWLDataSomeValuesFrom (
    property: owlapy.owl_property.OWLDataPropertyExpression,
    filler: owlapy.owl_class_expression.OWLDataRange)
    Bases: OWLQuantifiedDataRestriction
    Represents a DataSomeValuesFrom restriction in the OWL 2 Specification.
    __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.model.**OWLNaryDataRange** (  
    *operands: Iterable[owlapy.owl\_class\_expression.OWLDataRange]*)

Bases: *owlapy.owl\_class\_expression.OWLDataRange*, *owlapy.has.HasOperands[owlapy.owl\_class\_expression.OWLDataRange]*

OWLNaryDataRange.

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

**operands** () → *Iterable[owlapy.owl\_class\_expression.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.model.**OWLDataUnionOf** (  
    *operands: Iterable[owlapy.owl\_class\_expression.OWLDataRange]*)

Bases: *OWLNaryDataRange*

Represents a DataUnionOf data range in the OWL 2 Specification.

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

**type\_index: Final** = 4005

**class** owlapy.model.**OWLDataIntersectionOf** (  
    *operands: Iterable[owlapy.owl\_class\_expression.OWLDataRange]*)

Bases: *OWLNaryDataRange*

Represents DataIntersectionOf in the OWL 2 Specification.

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

**type\_index: Final** = 4004

**class** owlapy.model.**OWLImportsDeclaration** (*import\_iri: owlapy.iri.IRI*)

Bases: *owlapy.has.HasIRI*

Represents an import statement in an ontology.

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

**get\_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.

```
class owlapy.model.OWLLogicalAxiom (annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLAxiom*

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.model.OWLPropertyAxiom (annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLLogicalAxiom*

The base interface for property axioms.

```
__slots__ = ()
```

```
class owlapy.model.OWLObjectPropertyAxiom (
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLPropertyAxiom*

The base interface for object property axioms.

```
__slots__ = ()
```

```
class owlapy.model.OWLDataPropertyAxiom (
    annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLPropertyAxiom*

The base interface for data property axioms.

```
__slots__ = ()
```

```
class owlapy.model.OWLIndividualAxiom (annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLLogicalAxiom*

The base interface for individual axioms.

```
__slots__ = ()
```

```
class owlapy.model.OWLClassAxiom (annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: *OWLLogicalAxiom*

The base interface for class axioms.

```
__slots__ = ()
```

```
class owlapy.model.OWLDeclarationAxiom (entity: owlapy.owlobject.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.owlobject.OWLEntity

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.model.OWLDatatypeDefinitionAxiom(datatype: OWLDatatype,
    datarange: owlapy.owl_class_expression.OWLDataRange,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLLogicalAxiom
    Represents a DatatypeDefinition axiom in the OWL 2 Specification.

    __slots__ = ('_datatype', '_datarange')

    get_datatype() → OWLDatatype

    get_datarange() → owlapy.owl_class_expression.OWLDataRange

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

class owlapy.model.OWLHasKeyAxiom(
    class_expression: owlapy.owl_class_expression.OWLClassExpression,
    property_expressions: List[owlapy.owl_property.OWLPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLLogicalAxiom, owlapy.has.HasOperands[owlapy.owl_property.OWLPropertyExpression]
    Represents a HasKey axiom in the OWL 2 Specification.

    __slots__ = ('_class_expression', '_property_expressions')

    get_class_expression() → owlapy.owl_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.model.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.model.OWLNaryClassAxiom (
    class_expressions: List[owlapy.owl_class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLClassAxiom, OWLNaryAxiom[owlapy.owl_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.owl_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.model.OWLEquivalentClassesAxiom (
    class_expressions: List[owlapy.owl_class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryClassAxiom

    Represents an EquivalentClasses axiom in the OWL 2 Specification.

__slots__ = ()

contains_named_equivalent_class () → bool

```

```

contains_owl_nothing () → bool

contains_owl_thing () → bool

named_classes () → Iterable[owlapy.owl_class_expression.OWLClass]

class owlapy.model.OWLDisjointClassesAxiom (
    class_expressions: List[owlapy.owl_class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryClassAxiom
    Represents a DisjointClasses axiom in the OWL 2 Specification.
    __slots__ = ()

class owlapy.model.OWLNaryIndividualAxiom (individuals: List[OWLIndividual],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLIndividualAxiom, OWLNaryAxiom[OWLIndividual]
    Represents an axiom that contains two or more operands that could also be represented with multiple pairwise
    individual axioms.
    __slots__ = '_individuals'
    individuals () → Iterable[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.model.OWLDifferentIndividualsAxiom (individuals: List[OWLIndividual],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryIndividualAxiom
    Represents a DifferentIndividuals axiom in the OWL 2 Specification.
    __slots__ = ()

class owlapy.model.OWLSameIndividualAxiom (individuals: List[OWLIndividual],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryIndividualAxiom
    Represents a SameIndividual axiom in the OWL 2 Specification.
    __slots__ = ()

```

```

class owlapy.model.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.model.OWLEquivalentObjectPropertiesAxiom (
    properties: List[owlapy.owl_property.OWLObjectPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
    OWLObjectPropertyAxiom

    Represents EquivalentObjectProperties axioms in the OWL 2 Specification.

    __slots__ = ()

class owlapy.model.OWLDisjointObjectPropertiesAxiom (
    properties: List[owlapy.owl_property.OWLObjectPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
    OWLObjectPropertyAxiom

    Represents DisjointObjectProperties axioms in the OWL 2 Specification.

    __slots__ = ()

class owlapy.model.OWLInverseObjectPropertiesAxiom (
    first: owlapy.owl_property.OWLObjectPropertyExpression,
    second: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLObjectPropertyExpression],
    OWLObjectPropertyAxiom

    Represents InverseObjectProperties axioms in the OWL 2 Specification.

    __slots__ = ('_first', '_second')

    get_first_property () → owlapy.owl_property.OWLObjectPropertyExpression

    get_second_property () → owlapy.owl_property.OWLObjectPropertyExpression

```

```

__repr__()
    Return repr(self).

class owlapy.model.OWLEquivalentDataPropertiesAxiom(
    properties: List[owlapy.owl_property.OWLDataPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLDataPropertyExpression],
        OWLDataPropertyAxiom

Represents EquivalentDataProperties axioms in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.OWLDisjointDataPropertiesAxiom(
    properties: List[owlapy.owl_property.OWLDataPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLNaryPropertyAxiom[owlapy.owl_property.OWLDataPropertyExpression],
        OWLDataPropertyAxiom

Represents DisjointDataProperties axioms in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.OWLSubClassOfAxiom(
    sub_class: owlapy.owl_class_expression.OWLClassExpression,
    super_class: owlapy.owl_class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLClassAxiom

Represents an SubClassOf axiom in the OWL 2 Specification.

__slots__ = ('_sub_class', '_super_class')

get_sub_class() → owlapy.owl_class_expression.OWLClassExpression

get_super_class() → owlapy.owl_class_expression.OWLClassExpression

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.model.OWLDisjointUnionAxiom(cls_: owlapy.owl_class_expression.OWLClass,
    class_expressions: List[owlapy.owl_class_expression.OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLClassAxiom

Represents a DisjointUnion axiom in the OWL 2 Specification.

__slots__ = ('_cls', '_class_expressions')

get_owl_class() → owlapy.owl_class_expression.OWLClass

get_class_expressions() → Iterable[owlapy.owl_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.model.OWLClassAssertionAxiom (individual: OWLIndividual,
      class_expression: owlapy.owl_class_expression.OWLClassExpression,
      annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLIndividualAxiom
    Represents ClassAssertion axioms in the OWL 2 Specification.

    __slots__ = ('_individual', '_class_expression')

    get_individual() → OWLIndividual

    get_class_expression() → owlapy.owl_class_expression.OWLClassExpression

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

class owlapy.model.OWLAnnotationAxiom (annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLAxiom
    A super interface for annotation axioms.

    __slots__ = ()

    is_annotation_axiom() → bool

class owlapy.model.OWLAnnotationProperty (iri: owlapy.iri.IRI)
    Bases: owlapy.owl_property.OWLProperty
    Represents an AnnotationProperty in the OWL 2 specification.

    __slots__ = '_iri'

    get_iri() → owlapy.iri.IRI
        Gets the IRI of this object.

        Returns
        The IRI of this object.

class owlapy.model.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.model.OWLAnnotationAssertionAxiom(
    subject: owlapy.owl_annotation.OWLAnnotationSubject, annotation: OWLAnnotation)
    Bases: OWLAnnotationAxiom

    Represents AnnotationAssertion axioms in the OWL 2 specification.

    __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.model.OWLSubAnnotationPropertyOfAxiom(
    sub_property: OWLAnnotationProperty, super_property: OWLAnnotationProperty,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLAnnotationAxiom
Represents an SubAnnotationPropertyOf axiom in the OWL 2 specification.
__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.model.OWLAnnotationPropertyDomainAxiom(
    property_: OWLAnnotationProperty, domain: owlapy.iri.IRI,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLAnnotationAxiom
Represents an AnnotationPropertyDomain axiom in the OWL 2 specification.
__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.model.OWLAnnotationPropertyRangeAxiom(
    property_: OWLAnnotationProperty, range_: owlapy.iri.IRI,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLAnnotationAxiom
Represents an AnnotationPropertyRange axiom in the OWL 2 specification.
__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.model.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.model.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
    Represents a SubObjectPropertyOf axiom in the OWL 2 specification.

    __slots__ = ()

class owlapy.model.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
    Represents a SubDataPropertyOf axiom in the OWL 2 specification.

    __slots__ = ()

class owlapy.model.OWLPropertyAssertionAxiom (subject: OWLIndividual, property_: _P,
    object_: _C, annotations: Iterable[OWLAnnotation] | None = None)
    Bases: Generic[_P, _C], OWLIndividualAxiom
    Represents a PropertyAssertion axiom in the OWL 2 specification.

    __slots__ = ('_subject', '_property', '_object')

    get_subject() → OWLIndividual

    get_property() → _P

```

```

get_object() → _C

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.model.OwlObjectPropertyAssertionAxiom (subject: OwlIndividual,
    property_: owlapy.owl_property.OwlObjectPropertyExpression, object_: OwlIndividual,
    annotations: Iterable[OwlAnnotation] | None = None)
    Bases: OwlPropertyAssertionAxiom[owlapy.owl_property.OwlObjectPropertyExpression,
    OwlIndividual]
    Represents an ObjectPropertyAssertion axiom in the OWL 2 specification.

    __slots__ = ()

class owlapy.model.OwlNegativeObjectPropertyAssertionAxiom (
    subject: OwlIndividual, property_: owlapy.owl_property.OwlObjectPropertyExpression,
    object_: OwlIndividual, annotations: Iterable[OwlAnnotation] | None = None)
    Bases: OwlPropertyAssertionAxiom[owlapy.owl_property.OwlObjectPropertyExpression,
    OwlIndividual]
    Represents a NegativeObjectPropertyAssertion axiom in the OWL 2 specification.

    __slots__ = ()

class owlapy.model.OwlDataPropertyAssertionAxiom (subject: OwlIndividual,
    property_: owlapy.owl_property.OwlDataPropertyExpression, object_: OwlLiteral,
    annotations: Iterable[OwlAnnotation] | None = None)
    Bases: OwlPropertyAssertionAxiom[owlapy.owl_property.OwlDataPropertyExpression,
    OwlLiteral]
    Represents an DataPropertyAssertion axiom in the OWL 2 specification.

    __slots__ = ()

class owlapy.model.OwlNegativeDataPropertyAssertionAxiom (subject: OwlIndividual,
    property_: owlapy.owl_property.OwlDataPropertyExpression, object_: OwlLiteral,
    annotations: Iterable[OwlAnnotation] | None = None)
    Bases: OwlPropertyAssertionAxiom[owlapy.owl_property.OwlDataPropertyExpression,
    OwlLiteral]
    Represents an NegativeDataPropertyAssertion axiom in the OWL 2 specification.

    __slots__ = ()

class owlapy.model.OwlUnaryPropertyAxiom (property_: _P,
    annotations: Iterable[OwlAnnotation] | None = None)
    Bases: Generic[_P], OwlPropertyAxiom
    Unary property axiom.

    __slots__ = '_property'

    get_property() → _P

```

```

class owlapy.model.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.model.OWLFunctionalObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLObjectPropertyCharacteristicAxiom

Represents FunctionalObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.OWLAsymmetricObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLObjectPropertyCharacteristicAxiom

Represents AsymmetricObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.OWLInverseFunctionalObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLObjectPropertyCharacteristicAxiom

Represents InverseFunctionalObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.OWLIrreflexiveObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLObjectPropertyCharacteristicAxiom

Represents IrreflexiveObjectProperty axioms in the OWL 2 specification.

__slots__ = ()

class owlapy.model.OWLReflexiveObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLObjectPropertyCharacteristicAxiom

Represents ReflexiveObjectProperty axioms in the OWL 2 specification.

```

```

__slots__ = ()

class owlapy.model.OWLSymmetricObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLObjectPropertyCharacteristicAxiom
    Represents SymmetricObjectProperty axioms in the OWL 2 specification.
    __slots__ = ()

class owlapy.model.OWLTransitiveObjectPropertyAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLObjectPropertyCharacteristicAxiom
    Represents TransitiveObjectProperty axioms in the OWL 2 specification.
    __slots__ = ()

class owlapy.model.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.model.OWLFunctionalDataPropertyAxiom(
    property_: owlapy.owl_property.OWLDataPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLDataPropertyCharacteristicAxiom
    Represents FunctionalDataProperty axioms in the OWL 2 specification.
    __slots__ = ()

class owlapy.model.OWLPropertyDomainAxiom(property_: _P,
    domain: owlapy.owl_class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: Generic[_P], OWLUnaryPropertyAxiom[_P]
    Represents ObjectPropertyDomain axioms in the OWL 2 specification.
    __slots__ = '_domain'

    get_domain() → owlapy.owl_class_expression.OWLClassExpression

```

```

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.model.OWLPropertyRangeAxiom(property_: _P, range_: _R,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: Generic[_P, _R], OWLUnaryPropertyAxiom[_P]
    Represents ObjectPropertyRange axioms in the OWL 2 specification.

    __slots__ = '_range'

    get_range() → _R

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.model.OWLObjectPropertyDomainAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    domain: owlapy.owl_class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLPropertyDomainAxiom[owlapy.owl_property.OWLObjectPropertyExpression]
    Represents a ObjectPropertyDomain axiom in the OWL 2 Specification.

    __slots__ = ()

class owlapy.model.OWLDataPropertyDomainAxiom(
    property_: owlapy.owl_property.OWLDataPropertyExpression,
    domain: owlapy.owl_class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLPropertyDomainAxiom[owlapy.owl_property.OWLDataPropertyExpression]
    Represents a DataPropertyDomain axiom in the OWL 2 Specification.

    __slots__ = ()

class owlapy.model.OWLObjectPropertyRangeAxiom(
    property_: owlapy.owl_property.OWLObjectPropertyExpression,
    range_: owlapy.owl_class_expression.OWLClassExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLPropertyRangeAxiom[owlapy.owl_property.OWLObjectPropertyExpression,
    owlapy.owl_class_expression.OWLClassExpression]
    Represents a ObjectPropertyRange axiom in the OWL 2 Specification.

    __slots__ = ()

```

```

class owlapy.model.OWLDataPropertyRangeAxiom (
    property_: owlapy.owl_property.OWLDataPropertyExpression,
    range_: owlapy.owl_class_expression.OWLDataRange,
    annotations: Iterable[OWLAnnotation] | None = None)
Bases: OWLPropertyRangeAxiom[owlapy.owl_property.OWLDataPropertyExpression,
owlapy.owl_class_expression.OWLDataRange]
Represents a DataPropertyRange axiom in the OWL 2 Specification.
__slots__ = ()

```

```

class owlapy.model.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.owl_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[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.owl_class_expression.OWLClass)
    → Iterable[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[*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[*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[*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[*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[*OWLObjectPropertyRangeAxiom*]

Gets the OWLObjectPropertyRangeAxiom objects where the property is equal to the specified property.

**Parameters**

**property** – The property which is equal to the property of the retrieved axioms.

**Returns**

The axioms matching the search.

**abstract get\_owl\_ontology\_manager** () → *\_M*

Gets the manager that manages this ontology.

**abstract get\_ontology\_id** () → *OWLOntologyID*

Gets the OWLOntologyID belonging to this object.

**Returns**

The OWLOntologyID.



**is\_anonymous** () → bool

Check whether this ontology does contain an IRI or not.

**class** owlapy.model.OwlOntologyChange (ontology: OwlOntology)

Represents an ontology change.

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

**get\_ontology** () → OwlOntology

Gets the ontology that the change is/was applied to.

**Returns**

The ontology that the change is applicable to.

**class** owlapy.model.AddImport (ontology: OwlOntology,  
import\_declaration: OwlImportsDeclaration)

Bases: OwlOntologyChange

Represents an ontology change where an import statement is added to an ontology.

**\_\_slots\_\_** = ('\_ont', '\_declaration')

**get\_import\_declaration** () → OwlImportsDeclaration

Gets the import declaration that the change pertains to.

**Returns**

The import declaration.

**class** owlapy.model.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) → 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) → 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*: *OWLOntology*, *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*: *OWLOntology*, *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*: *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.model.OWLReasoner** (*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.owl\_class\_expression.OWLClassExpression*]

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

### Parameters

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

### Returns

Let  $N = \text{equivalent\_classes}(\text{DataSomeValuesFrom}(\text{pe} \text{ rdfs:Literal}))$ . If *direct* is True: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of  $\text{super\_classes}(\text{DataSomeValuesFrom}(\text{pe} \text{ rdfs:Literal}), \text{true})$ . If *direct* is False: then the result of  $\text{super\_classes}(\text{DataSomeValuesFrom}(\text{pe} \text{ rdfs:Literal}), \text{false})$  together with  $N$  if  $N$  is non-empty. (Note, *rdfs:Literal* is the top datatype).

**abstract object\_property\_domains** (*pe*: *owlapy.owl\_property.OWLObjectProperty*,  
*direct*: *bool* = *False*) → *Iterable*[*owlapy.owl\_class\_expression.OWLClassExpression*]

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

### Parameters

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

### Returns

Let  $N = \text{equivalent\_classes}(\text{ObjectSomeValuesFrom}(\text{pe owl:Thing}))$ . If **direct** is True: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{pe owl:Thing}), \text{true})$ . If **direct** is False: then the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{pe owl:Thing}), \text{false})$  together with  $N$  if  $N$  is non-empty.

**abstract object\_property\_ranges** (*pe: owlapy.owl\_property.OWLObjectProperty*,  
*direct: bool = False*)  $\rightarrow \text{Iterable}[\text{owlapy.owl\_class\_expression.OWLClassExpression}]$

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

### Parameters

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

### Returns

Let  $N = \text{equivalent\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe}) \text{owl:Thing}))$ . If **direct** is True: then if  $N$  is not empty then the return value is  $N$ , else the return value is the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe}) \text{owl:Thing}), \text{true})$ . If **direct** is False: then the result of  $\text{super\_classes}(\text{ObjectSomeValuesFrom}(\text{ObjectInverseOf}(\text{pe}) \text{owl:Thing}), \text{false})$  together with  $N$  if  $N$  is non-empty.

**abstract equivalent\_classes** (*ce: owlapy.owl\_class\_expression.OWLClassExpression*,  
*only\_named: bool = True*)  $\rightarrow \text{Iterable}[\text{owlapy.owl\_class\_expression.OWLClassExpression}]$

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

### Parameters

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

### Returns

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

**abstract disjoint\_classes** (*ce: owlapy.owl\_class\_expression.OWLClassExpression*,  
*only\_named: bool = True*)  $\rightarrow \text{Iterable}[\text{owlapy.owl\_class\_expression.OWLClassExpression}]$

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: OWLNamedIndividual*)  
→ `Iterable[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: OWLNamedIndividual*) → `Iterable[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: owlpy.owl\_property.OWLObjectPropertyExpression*)  
→ `Iterable[owlpy.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: owlpy.owl\_property.OWLDataProperty*)  
→ `Iterable[owlpy.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: OWLNamedIndividual, pe: owlpy.owl\_property.OWLDataProperty, direct: bool = True*) → `Iterable[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*: *OWLNamedIndividual*,  
*pe*: *owlapy.owl\_property.OWLObjectPropertyExpression*, *direct*: *bool = True*)  
→ *Iterable[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.owl\_class\_expression.OWLClassExpression*, *direct*: *bool = False*)  
→ *Iterable[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.owl\_class\_expression.OWLClassExpression*, *direct*: *bool = False*,  
*only\_named*: *bool = True*) → *Iterable[owlapy.owl\_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: OWLNamedIndividual*, *direct: bool = False*) → *Iterable[owlapy.owl\_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** () → *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.owl\_class\_expression.OWLClassExpression*,  
*direct: bool = False, only\_named: bool = True*)  
→ *Iterable[owlapy.owl\_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.model.OWLThing: Final`

`owlapy.model.OWLNothing: Final`

`owlapy.model.OWLTopObjectProperty: Final`

`owlapy.model.OWLBottomObjectProperty: Final`

`owlapy.model.OWLTopDataProperty: Final`

`owlapy.model.OWLBottomDataProperty: Final`

`owlapy.model.DoubleOWLDatatype: Final`

`owlapy.model.IntegerOWLDatatype: Final`

`owlapy.model.BooleanOWLDatatype: Final`

`owlapy.model.StringOWLDatatype: Final`

`owlapy.model.DateOWLDatatype: Final`

`owlapy.model.DateTimeOWLDatatype: Final`

`owlapy.model.DurationOWLDatatype: Final`

`owlapy.model.TopOWLDatatype: Final`

`owlapy.model.NUMERIC_DATATYPES: Final[Set[OWLDatatype]]`

`owlapy.model.TIME_DATATYPES: Final[Set[OWLDatatype]]`



`owlapy.owl2sparql`

OWL-to-SPARQL converter.

## Submodules

`owlapy.owl2sparql.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</i> (x)	Peek the last element of an array.
<i>owl_expression_to_sparql</i> (→ str)	Convert an OWL Class Expression ( <a href="https://www.w3.org/TR/owl2-syntax/#Class_Expressions">https://www.w3.org/TR/owl2-syntax/#Class_Expressions</a> ) into a SPARQL query

### Attributes

<i>converter</i>
------------------

`owlapy.owl2sparql.converter.peek(x)`

Peek the last element of an array.

#### Returns

The last element `arr[-1]`.

**class** `owlapy.owl2sparql.converter.VariablesMapping`

Helper class for owl-to-sparql conversion.

**\_\_slots\_\_** = ('class\_cnt', 'prop\_cnt', 'ind\_cnt', 'dict')

**get\_variable** (*e*: `owlapy.model.OWLEntity`) → str

**new\_individual\_variable** () → str

```

new_property_variable () → str

__contains__ (item: owlapy.model.OWLEntity) → bool

__getitem__ (item: owlapy.model.OWLEntity) → str

class owlapy.owl2sparql.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.model.OWLClassExpression

    sparql: List[str]

    variables: List[str]

    parent: List[owlapy.model.OWLClassExpression]

    parent_var: List[str]

    variable_entities: Set[owlapy.model.OWLEntity]

    properties: Dict[int, List[owlapy.model.OWLEntity]]

    mapping: VariablesMapping

    grouping_vars: Dict[owlapy.model.OWLClassExpression, Set[str]]

    having_conditions: Dict[owlapy.model.OWLClassExpression, Set[str]]

    cnt: int

    convert (root_variable: str, ce: owlapy.model.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.model.OWLClassExpression)

```

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

**new\_count\_var** () → str

**append\_triple** (*subject, predicate, object\_*)

**append** (*frag*)

**triple** (*subject, predicate, object\_*)

**as\_query** (*root\_variable: str, ce: owlapy.model.OWLClassExpression, count: bool = False,*  
*values: Iterable[owlapy.model.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.owl2sparql.converter.converter

owlapy.owl2sparql.converter.owl\_expression\_to\_sparql (*root\_variable: str = '?x',*  
*expression: owlapy.model.OWLClassExpression = None,*  
*values: Iterable[owlapy.model.OWLNamedIndividual] | None = None,*  
*named\_individuals: bool = False*) → str

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

values: positive or negative examples from a class expression problem. Unclear named\_individuals: if set to True, the generated SPARQL query will return only entities that are instances of owl:NamedIndividual

## 2.2 Submodules

owlapy.has

### Module Contents

#### Classes

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

**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.

**class** owlapy.has.HasIRI

Simple class to access the IRI.

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

**abstract** **get\_iri**() → *IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**class** owlapy.has.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.has.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

**owlapy.iri**

## Module Contents

### Classes

---

*IRI*

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.namespaces**

Namespaces.

## **Module Contents**

### **Classes**

<i>Namespaces</i>	A Namespace and its prefix.
-------------------	-----------------------------

### **Attributes**

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

**class** owlapy.namespaces.**Namespaces** (*prefix: str, ns: str*)

A Namespace and its prefix.

**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**

## 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.owlobject.OWLObject*

A marker interface for the values (objects) of annotations.

```
__slots__ = ()
```

```
as_iri() → IRI | None
```

#### Returns

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

```
as_anonymous_individual()
```

#### Returns

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

**class** owlapy.owl\_annotation.OWLAnnotationSubject

Bases: *OWLAnnotationObject*

A marker interface for annotation subjects, which can either be IRIs or anonymous individuals

```
__slots__ = ()
```

**class** owlapy.owl\_annotation.OWLAnnotationValue

Bases: *OWLAnnotationObject*

A marker interface for annotation values, which can either be an IRI (URI), Literal or Anonymous Individual.

```
__slots__ = ()
```

**is\_literal()** → bool

**Returns**

true if the annotation value is a literal

**as\_literal()** → *OWLLiteral* | None

**Returns**

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

**owlapy.owl\_class\_expression**

## Module Contents

### Classes

<i>OWLPropertyRange</i>	OWL Objects that can be the ranges of properties.
<i>OWLDataRange</i>	Represents a DataRange in the OWL 2 Specification.
<i>OWLClassExpression</i>	An OWL 2 Class Expression.
<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

**class** owlapy.owl\_class\_expression.**OWLPropertyRange**

Bases: *owlapy.owlobject.OWLObject*

OWL Objects that can be the ranges of properties.

**class** owlapy.owl\_class\_expression.**OWLDataRange**

Bases: *OWLPropertyRange*

Represents a DataRange in the OWL 2 Specification.

**class** owlapy.owl\_class\_expression.**OWLClassExpression**

Bases: *OWLPropertyRange*

An OWL 2 Class Expression.

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

**abstract is\_owl\_thing()** → bool

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

**Returns**

Thing.

**Return type**

True if this expression is owl

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

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



```

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

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

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

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

class owlapy.owl_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.owl_class_expression.OWLBooleanClassExpression
    Bases: OWLAnonymousClassExpression
    Represent an anonymous boolean class expression.

    __slots__ = ()

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

    __slots__ = '_operand'

    type_index: Final = 3003

```

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

**Returns**

The wrapped expression.

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

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

**Returns**

The operands.

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

Return repr(self).

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

Return self==value.

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

Return hash(self).

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

Bases: *OWLClassExpression*, owlapy.owlobject.*OWLEntity*

An OWL 2 named Class

**property str**

**property reminder: str**

The reminder of the IRI

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

**type\_index: Final = 1001**

**get\_iri** () → *IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**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** () → *OWLObjectComplementOf*

Gets the object complement of this class expression.

**Returns**

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

**get\_nnf()** → *OWLClass*

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

**Returns**

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

**owlapy.owl\_property**

## 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.
<i>OWLObjectInverseOf</i>	Represents the inverse of a property expression (ObjectInverseOf). This can be used to refer to the inverse of
<i>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification.

**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.owlobject.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.

**property** str: str

**property** iri: str

**\_\_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.

**get\_iri** () → *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**

The IRI of this object.

**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). 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.

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

**type\_index**: **Final** = 1003

**get\_inverse** () → *OWLObjectProperty*

Gets the property expression that this is the inverse of.

**Returns**

The object property expression such that this object property expression is an inverse of it.

**get\_inverse\_property** () → *OWLObjectProperty*

Obtains the property that corresponds to the inverse of this property.

**Returns**

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

**get\_named\_property** () → *OWLObjectProperty*

Get the named object property used in this property expression.

**Returns**

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

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

Return repr(self).

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

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

**class** owlapy.owl\_property.OWLDataProperty (*iri: owlapy.iri.IRI*)

Bases: *OWLDataPropertyExpression*, *OWLProperty*

Represents a Data Property in the OWL 2 Specification.

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

**type\_index**: Final = 1004

**get\_iri** () → *owlapy.iri.IRI*

Gets the IRI of this object.

**Returns**  
The IRI of this object.

**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.owlobject

## 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.owlobject.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.owlobject.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.owlobject.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.owlobject.OWLNamedObject
    Bases: OWLObject, owlapy.has.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.owlobject.OWLEntity
    Bases: OWLNamedObject
    Represents Entities in the OWL 2 Specification.

    __slots__ = ()

```

```

to_string_id() → str
is_anonymous() → bool

```

## owlapy.parser

String to OWL parsers.

## Module Contents

### Classes

<i>ManchesterOWLSyntaxParser</i>	Manchester Syntax parser to parse strings to OWLClass-Expressions.
<i>DLSyntaxParser</i>	Description Logic Syntax parser to parse strings to OWL-ClassExpressions.

### Functions

<i>dl_to_owl_expression</i> (dl_expression)
<i>manchester_to_owl_expression</i> (manchester_ex

### Attributes

<i>MANCHESTER_GRAMMAR</i>
<i>DL_GRAMMAR</i>
<i>DLparser</i>
<i>ManchesterParser</i>

owlapy.parser.**MANCHESTER\_GRAMMAR**

```

class owlapy.parser.ManchesterOWLSyntaxParser (
    namespace: str | owlapy.namespaces.Namespaces | None = None, grammar=None)
    Bases: parsimonious.nodes.NodeVisitor, owlapy.owlobject.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.model.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.model.OWLClassExpression`

**visit\_intersection** (*node, children*) → `owlapy.model.OWLClassExpression`

**visit\_primary** (*node, children*) → `owlapy.model.OWLClassExpression`

**visit\_some\_only\_res** (*node, children*) → `owlapy.model.OWLQuantifiedObjectRestriction`

**visit\_cardinality\_res** (*node, children*) → `owlapy.model.OWLObjectCardinalityRestriction`

**visit\_value\_res** (*node, children*) → `owlapy.model.OWLObjectHasValue`

**visit\_has\_self** (*node, children*) → `owlapy.model.OWLObjectHasSelf`

**visit\_object\_property** (*node, children*) → `owlapy.model.OWLObjectPropertyExpression`

**visit\_class\_expression** (*node, children*) → `owlapy.model.OWLClassExpression`

**visit\_individual\_list** (*node, children*) → `owlapy.model.OWLObjectOneOf`

**visit\_data\_primary** (*node, children*) → `owlapy.model.OWLDataRange`

**visit\_data\_some\_only\_res** (*node, children*) → `owlapy.model.OWLQuantifiedDataRestriction`

**visit\_data\_cardinality\_res** (*node, children*) → `owlapy.model.OWLDataCardinalityRestriction`

**visit\_data\_value\_res** (*node, children*) → `owlapy.model.OWLDataHasValue`

**visit\_data\_union** (*node, children*) → `owlapy.model.OWLDataRange`

**visit\_data\_intersection** (*node, children*) → `owlapy.model.OWLDataRange`

**visit\_literal\_list** (*node, children*) → `owlapy.model.OWLDataOneOf`

**visit\_data\_parentheses** (*node, children*) → `owlapy.model.OWLDataRange`

**visit\_datatype\_restriction** (*node, children*) → `owlapy.model.OWLDatatypeRestriction`

**visit\_facet\_restrictions** (*node, children*) → `List[owlapy.model.OWLFacetRestriction]`

**visit\_facet\_restriction** (*node, children*) → `owlapy.model.OWLFacetRestriction`

**visit\_literal** (*node, children*) → `owlapy.model.OWLLiteral`

**visit\_typed\_literal** (*node, children*) → `owlapy.model.OWLLiteral`

**abstract visit\_string\_literal\_language** (*node, children*)

**visit\_string\_literal\_no\_language** (*node, children*) → `owlapy.model.OWLLiteral`

```

visit_quoted_string (node, children) → str
visit_float_literal (node, children) → owlapy.model.OWLLiteral
visit_decimal_literal (node, children) → owlapy.model.OWLLiteral
visit_integer_literal (node, children) → owlapy.model.OWLLiteral
visit_boolean_literal (node, children) → owlapy.model.OWLLiteral
visit_datetime_literal (node, children) → owlapy.model.OWLLiteral
visit_duration_literal (node, children) → owlapy.model.OWLLiteral
visit_date_literal (node, children) → owlapy.model.OWLLiteral
visit_non_negative_integer (node, children) → int
visit_datatype_iri (node, children) → str
visit_datatype (node, children) → owlapy.model.OWLDatatype
visit_facet (node, children) → owlapy.vocab.OWLFacet
visit_class_iri (node, children) → owlapy.model.OWLClass
visit_individual_iri (node, children) → owlapy.model.OWLNamedIndividual
visit_object_property_iri (node, children) → owlapy.model.OWLObjectProperty
visit_data_property_iri (node, children) → owlapy.model.OWLDataProperty
visit_iri (node, children) → owlapy.model.IRI
visit_full_iri (node, children) → owlapy.model.IRI
abstract visit_abbreviated_iri (node, children)
visit_simple_iri (node, children) → owlapy.model.IRI
visit_parentheses (node, children) → owlapy.model.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.owlobject.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.model.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.model.OWLClassExpression

visit_intersection (node, children) → owlapy.model.OWLClassExpression

visit_primary (node, children) → owlapy.model.OWLClassExpression

visit_some_only_res (node, children) → owlapy.model.OWLQuantifiedObjectRestriction

visit_cardinality_res (node, children) → owlapy.model.OWLObjectCardinalityRestriction

visit_value_res (node, children) → owlapy.model.OWLObjectHasValue

visit_has_self (node, children) → owlapy.model.OWLObjectHasSelf

visit_object_property (node, children) → owlapy.model.OWLObjectPropertyExpression

visit_class_expression (node, children) → owlapy.model.OWLClassExpression

visit_individual_list (node, children) → owlapy.model.OWLObjectOneOf

visit_data_primary (node, children) → owlapy.model.OWLDataRange

visit_data_some_only_res (node, children) → owlapy.model.OWLQuantifiedDataRestriction

visit_data_cardinality_res (node, children) → owlapy.model.OWLDataCardinalityRestriction

visit_data_value_res (node, children) → owlapy.model.OWLDataHasValue

visit_data_union (node, children) → owlapy.model.OWLDataRange

visit_data_intersection (node, children) → owlapy.model.OWLDataRange

visit_literal_list (node, children) → owlapy.model.OWLDataOneOf

visit_data_parentheses (node, children) → owlapy.model.OWLDataRange

visit_datatype_restriction (node, children) → owlapy.model.OWLDatatypeRestriction

visit_facet_restrictions (node, children) → List[owlapy.model.OWLFacetRestriction]

visit_facet_restriction (node, children) → owlapy.model.OWLFacetRestriction

visit_literal (node, children) → owlapy.model.OWLLiteral

visit_typed_literal (node, children) → owlapy.model.OWLLiteral

abstract visit_string_literal_language (node, children)

```

**visit\_string\_literal\_no\_language** (*node, children*) → *owlapy.model.OWLLiteral*  
**visit\_quoted\_string** (*node, children*) → *str*  
**visit\_float\_literal** (*node, children*) → *owlapy.model.OWLLiteral*  
**visit\_decimal\_literal** (*node, children*) → *owlapy.model.OWLLiteral*  
**visit\_integer\_literal** (*node, children*) → *owlapy.model.OWLLiteral*  
**visit\_boolean\_literal** (*node, children*) → *owlapy.model.OWLLiteral*  
**visit\_datetime\_literal** (*node, children*) → *owlapy.model.OWLLiteral*  
**visit\_duration\_literal** (*node, children*) → *owlapy.model.OWLLiteral*  
**visit\_date\_literal** (*node, children*) → *owlapy.model.OWLLiteral*  
**visit\_non\_negative\_integer** (*node, children*) → *int*  
**visit\_datatype\_iri** (*node, children*) → *str*  
**visit\_datatype** (*node, children*) → *owlapy.model.OWLDatatype*  
**visit\_facet** (*node, children*) → *owlapy.vocab.OWLFacet*  
**visit\_class\_iri** (*node, children*) → *owlapy.model.OWLClass*  
**visit\_individual\_iri** (*node, children*) → *owlapy.model.OWLNamedIndividual*  
**visit\_object\_property\_iri** (*node, children*) → *owlapy.model.OWLObjectProperty*  
**visit\_data\_property\_iri** (*node, children*) → *owlapy.model.OWLDataProperty*  
**visit\_iri** (*node, children*) → *owlapy.model.IRI*  
**visit\_full\_iri** (*node, children*) → *owlapy.model.IRI*  
**abstract visit\_abbreviated\_iri** (*node, children*)  
**visit\_simple\_iri** (*node, children*) → *owlapy.model.IRI*  
**visit\_parentheses** (*node, children*) → *owlapy.model.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*)

`owlapy.parser.manchester_to_owl_expression` (*manchester\_expression: str*)

## 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

<i>owl_expression_to_dl</i> ( $\rightarrow$ str)
<i>owl_expression_to_manchester</i> ( $\rightarrow$ str)

### Attributes

<i>DLrenderer</i>
<i>ManchesterRenderer</i>

```
class owlapy.render.DLSyntaxObjectRenderer (
    short_form_provider: Callable[[owlapy.model.OWLEntity], str] = _simple_short_form_provider)
    Bases: owlapy.owlobject.OWLObjectRenderer
    DL Syntax renderer for OWL Objects.
    __slots__ = '_sfp'
    set_short_form_provider (short_form_provider: Callable[[owlapy.model.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.model.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.model.OWLEntity], str] = _simple_short_form_provider,
    no_render_thing=False)

Bases: owlapy.owlobject.OWLObjectRenderer

Manchester Syntax renderer for OWL Objects

__slots__ = ('_sfp', '_no_render_thing')

set_short_form_provider (short_form_provider: Callable[[owlapy.model.OWLEntity], str])
    → None
    Configure a short form provider that shortens the OWL objects during rendering.

    Parameters
        short_form_provider – Short form provider.

render (o: owlapy.model.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.model.OWLObject) → str

owlapy.render.owl_expression_to_manchester (o: owlapy.model.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

<code>combine_nary_expressions(...)</code>	Shortens an OWLClassExpression or OWLDataRange by combining all nested nary expressions of the same type.
<code>iter_count(→ int)</code>	Count the number of elements in an iterable.
<code>as_index(→ owlapy.has.HasIndex)</code>	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.model.OWLClassExpression, negated: bool = False*)  
→ *owlapy.model.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.model.OWLClassExpression*) → *owlapy.model.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.model.OWLClassExpression*) → *owlapy.model.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.model.OWLClassExpression*)

→ *owlapy.model.OWLClassExpression*

**owlapy.util.combine\_nary\_expressions** (*ce: owlapy.model.OWLDataRange*)

→ *owlapy.model.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.model.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

<a href="#"><i>OWLRDFVocabulary</i></a>	Enumerations for OWL/RDF vocabulary.
<a href="#"><i>XSDVocabulary</i></a>	Enumerations for XSD vocabulary.
<a href="#"><i>OWLFacet</i></a>	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

```

## 2.3 Package Contents

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

## Python Module Index

### O

- `owlapy`, 1
- `owlapy.has`, 59
- `owlapy.iri`, 60
- `owlapy.model`, 1
- `owlapy.model.providers`, 2
- `owlapy.namespaces`, 62
- `owlapy.owl2sparql`, 57
- `owlapy.owl2sparql.converter`, 57
- `owlapy.owl_annotation`, 63
- `owlapy.owl_class_expression`, 64
- `owlapy.owl_property`, 67
- `owlapy.owlobject`, 70
- `owlapy.parser`, 72
- `owlapy.render`, 77
- `owlapy.util`, 78
- `owlapy.vocab`, 81

# Index

## Non-alphabetical

`__contains__()` (`owlapy.owl2sparql.converter.VariablesMapping` method), 58  
`__contains__()` (`owlapy.util.LRUCache` method), 80  
`__eq__()` (`owlapy.has.HasIndex` method), 59  
`__eq__()` (`owlapy.iri.IRI` method), 61  
`__eq__()` (`owlapy.model.HasIndex` method), 12  
`__eq__()` (`owlapy.model.IRI` method), 11  
`__eq__()` (`owlapy.model.OWLAnnotation` method), 40  
`__eq__()` (`owlapy.model.OWLAnnotationAssertionAxiom` method), 40  
`__eq__()` (`owlapy.model.OWLAnnotationPropertyDomainAxiom` method), 41  
`__eq__()` (`owlapy.model.OWLAnnotationPropertyRangeAxiom` method), 41  
`__eq__()` (`owlapy.model.OWLClassAssertionAxiom` method), 39  
`__eq__()` (`owlapy.model.OWLDataAllValuesFrom` method), 29  
`__eq__()` (`owlapy.model.OWLDataCardinalityRestriction` method), 29  
`__eq__()` (`owlapy.model.OWLDataComplementOf` method), 29  
`__eq__()` (`owlapy.model.OWLDataHasValue` method), 30  
`__eq__()` (`owlapy.model.OWLDataOneOf` method), 31  
`__eq__()` (`owlapy.model.OWLDataPropertyCharacteristicAxiom` method), 45  
`__eq__()` (`owlapy.model.OWLDataSomeValuesFrom` method), 31  
`__eq__()` (`owlapy.model.OWLDatatypeDefinitionAxiom` method), 34  
`__eq__()` (`owlapy.model.OWLDatatypeRestriction` method), 26  
`__eq__()` (`owlapy.model.OWLDeclarationAxiom` method), 34  
`__eq__()` (`owlapy.model.OWLDisjointUnionAxiom` method), 39  
`__eq__()` (`owlapy.model.OWLFacetRestriction` method), 26  
`__eq__()` (`owlapy.model.OWLHasKeyAxiom` method), 34  
`__eq__()` (`owlapy.model.OWLHasValueRestriction` method), 19  
`__eq__()` (`owlapy.model.OWLNaryBooleanClassExpression` method), 21  
`__eq__()` (`owlapy.model.OWLNaryClassAxiom` method), 35  
`__eq__()` (`owlapy.model.OWLNaryDataRange` method), 32  
`__eq__()` (`owlapy.model.OWLNaryIndividualAxiom` method), 36  
`__eq__()` (`owlapy.model.OWLNaryPropertyAxiom` method), 37  
`__eq__()` (`owlapy.model.OWLObject` method), 10  
`__eq__()` (`owlapy.model.OWLObjectAllValuesFrom` method), 20  
`__eq__()` (`owlapy.model.OWLObjectCardinalityRestriction` method), 22  
`__eq__()` (`owlapy.model.OWLObjectComplementOf` method), 14  
`__eq__()` (`owlapy.model.OWLObjectHasSelf` method), 23  
`__eq__()` (`owlapy.model.OWLObjectOneOf` method), 24  
`__eq__()` (`owlapy.model.OWLObjectPropertyCharacteristicAxiom` method), 44  
`__eq__()` (`owlapy.model.OWLObjectSomeValuesFrom` method), 20  
`__eq__()` (`owlapy.model.OWLOntologyID` method), 25  
`__eq__()` (`owlapy.model.OWLPropertyAssertionAxiom` method), 43  
`__eq__()` (`owlapy.model.OWLPropertyDomainAxiom` method), 45  
`__eq__()` (`owlapy.model.OWLPropertyRangeAxiom` method), 46  
`__eq__()` (`owlapy.model.OWLSubAnnotationPropertyOfAxiom` method), 41  
`__eq__()` (`owlapy.model.OWLSubClassOfAxiom` method), 38  
`__eq__()` (`owlapy.model.OWLSubPropertyAxiom` method), 42  
`__eq__()` (`owlapy.namespaces.Namespaces` method), 63  
`__eq__()` (`owlapy.owl_class_expression.OWLObjectComplementOf` method), 66  
`__eq__()` (`owlapy.owl_property.OWLObjectInverseOf` method), 69  
`__eq__()` (`owlapy.owlobject.OWLNamedObject` method), 71  
`__eq__()` (`owlapy.owlobject.OWLObject` method), 70  
`__eq__()` (`owlapy.util.OrderedOWLObject` method), 79  
`__getitem__()` (`owlapy.owl2sparql.converter.VariablesMapping` method), 58  
`__getitem__()` (`owlapy.util.LRUCache` method), 80  
`__hash__()` (`owlapy.iri.IRI` method), 61  
`__hash__()` (`owlapy.model.IRI` method), 11  
`__hash__()` (`owlapy.model.OWLAnnotation` method), 40  
`__hash__()` (`owlapy.model.OWLAnnotationAssertionAxiom` method), 40  
`__hash__()` (`owlapy.model.OWLAnnotationPropertyDomainAxiom` method), 41  
`__hash__()` (`owlapy.model.OWLAnnotationPropertyRangeAxiom` method), 41  
`__hash__()` (`owlapy.model.OWLClassAssertionAxiom` method), 39  
`__hash__()` (`owlapy.model.OWLDataAllValuesFrom` method), 29  
`__hash__()` (`owlapy.model.OWLDataCardinalityRestriction` method), 29  
`__hash__()` (`owlapy.model.OWLDataComplementOf` method), 29

\_\_hash\_\_() (owlapy.model.OWLDataHasValue method), 30  
\_\_hash\_\_() (owlapy.model.OWLDataOneOf method), 31  
\_\_hash\_\_() (owlapy.model.OWLDataPropertyCharacteristicAxiom method), 45  
\_\_hash\_\_() (owlapy.model.OWLDataSomeValuesFrom method), 31  
\_\_hash\_\_() (owlapy.model.OWLDatatypeDefinitionAxiom method), 34  
\_\_hash\_\_() (owlapy.model.OWLDatatypeRestriction method), 26  
\_\_hash\_\_() (owlapy.model.OWLDeclarationAxiom method), 34  
\_\_hash\_\_() (owlapy.model.OWLDisjointUnionAxiom method), 39  
\_\_hash\_\_() (owlapy.model.OWLFacetRestriction method), 26  
\_\_hash\_\_() (owlapy.model.OWLHasKeyAxiom method), 34  
\_\_hash\_\_() (owlapy.model.OWLHasValueRestriction method), 19  
\_\_hash\_\_() (owlapy.model.OWLNaryBooleanClassExpression method), 21  
\_\_hash\_\_() (owlapy.model.OWLNaryClassAxiom method), 35  
\_\_hash\_\_() (owlapy.model.OWLNaryDataRange method), 32  
\_\_hash\_\_() (owlapy.model.OWLNaryIndividualAxiom method), 36  
\_\_hash\_\_() (owlapy.model.OWLNaryPropertyAxiom method), 37  
\_\_hash\_\_() (owlapy.model.OWLObject method), 10  
\_\_hash\_\_() (owlapy.model.OWLObjectAllValuesFrom method), 20  
\_\_hash\_\_() (owlapy.model.OWLObjectCardinalityRestriction method), 22  
\_\_hash\_\_() (owlapy.model.OWLObjectComplementOf method), 14  
\_\_hash\_\_() (owlapy.model.OWLObjectHasSelf method), 23  
\_\_hash\_\_() (owlapy.model.OWLObjectOneOf method), 24  
\_\_hash\_\_() (owlapy.model.OWLObjectPropertyCharacteristicAxiom method), 44  
\_\_hash\_\_() (owlapy.model.OWLObjectSomeValuesFrom method), 20  
\_\_hash\_\_() (owlapy.model.OWLPropertyAssertionAxiom method), 43  
\_\_hash\_\_() (owlapy.model.OWLPropertyDomainAxiom method), 46  
\_\_hash\_\_() (owlapy.model.OWLPropertyRangeAxiom method), 46  
\_\_hash\_\_() (owlapy.model.OWLSubAnnotationPropertyOfAxiom method), 41  
\_\_hash\_\_() (owlapy.model.OWLSubClassOfAxiom method), 38  
\_\_hash\_\_() (owlapy.model.OWLSubPropertyAxiom method), 42  
\_\_hash\_\_() (owlapy.namespaces.Namespaces method), 62  
\_\_hash\_\_() (owlapy.owl\_class\_expression.OWLObjectComplementOf method), 66  
\_\_hash\_\_() (owlapy.owl\_property.OWLObjectInverseOf method), 70  
\_\_hash\_\_() (owlapy.owlobject.OWLNamedObject method), 71  
\_\_hash\_\_() (owlapy.owlobject.OWLObject method), 70  
\_\_lt\_\_() (owlapy.owlobject.OWLNamedObject method), 71  
\_\_lt\_\_() (owlapy.util.OrderedOWLObject method), 79  
\_\_repr\_\_() (owlapy.iri.IRI method), 61  
\_\_repr\_\_() (owlapy.model.IRI method), 11  
\_\_repr\_\_() (owlapy.model.OWLAnnotation method), 40  
\_\_repr\_\_() (owlapy.model.OWLAnnotationAssertionAxiom method), 40  
\_\_repr\_\_() (owlapy.model.OWLAnnotationPropertyDomainAxiom method), 41  
\_\_repr\_\_() (owlapy.model.OWLAnnotationPropertyRangeAxiom method), 42  
\_\_repr\_\_() (owlapy.model.OWLClassAssertionAxiom method), 39  
\_\_repr\_\_() (owlapy.model.OWLDataAllValuesFrom method), 29  
\_\_repr\_\_() (owlapy.model.OWLDataCardinalityRestriction method), 29  
\_\_repr\_\_() (owlapy.model.OWLDataComplementOf method), 29  
\_\_repr\_\_() (owlapy.model.OWLDataHasValue method), 30  
\_\_repr\_\_() (owlapy.model.OWLDataOneOf method), 31  
\_\_repr\_\_() (owlapy.model.OWLDataPropertyCharacteristicAxiom method), 45  
\_\_repr\_\_() (owlapy.model.OWLDataSomeValuesFrom method), 31  
\_\_repr\_\_() (owlapy.model.OWLDatatypeDefinitionAxiom method), 34  
\_\_repr\_\_() (owlapy.model.OWLDatatypeRestriction method), 26  
\_\_repr\_\_() (owlapy.model.OWLDeclarationAxiom method), 34  
\_\_repr\_\_() (owlapy.model.OWLDisjointUnionAxiom method), 39  
\_\_repr\_\_() (owlapy.model.OWLFacetRestriction method), 26  
\_\_repr\_\_() (owlapy.model.OWLHasKeyAxiom method), 34  
\_\_repr\_\_() (owlapy.model.OWLInverseObjectPropertiesAxiom method), 37  
\_\_repr\_\_() (owlapy.model.OWLNaryBooleanClassExpression method), 21  
\_\_repr\_\_() (owlapy.model.OWLNaryClassAxiom method), 35  
\_\_repr\_\_() (owlapy.model.OWLNaryDataRange method), 32  
\_\_repr\_\_() (owlapy.model.OWLNaryIndividualAxiom method), 36  
\_\_repr\_\_() (owlapy.model.OWLNaryPropertyAxiom method), 37  
\_\_repr\_\_() (owlapy.model.OWLObject method), 10  
\_\_repr\_\_() (owlapy.model.OWLObjectAllValuesFrom method), 20  
\_\_repr\_\_() (owlapy.model.OWLObjectCardinalityRestriction method), 22  
\_\_repr\_\_() (owlapy.model.OWLObjectComplementOf method), 14  
\_\_repr\_\_() (owlapy.model.OWLObjectHasSelf method), 23

\_\_repr\_\_() (owlapy.model.OwlObjectHasValue method), 23  
\_\_repr\_\_() (owlapy.model.OwlObjectOneOf method), 24  
\_\_repr\_\_() (owlapy.model.OwlObjectPropertyCharacteristicAxiom method), 44  
\_\_repr\_\_() (owlapy.model.OwlObjectSomeValuesFrom method), 20  
\_\_repr\_\_() (owlapy.model.OwlOntologyID method), 25  
\_\_repr\_\_() (owlapy.model.OwlPropertyAssertionAxiom method), 43  
\_\_repr\_\_() (owlapy.model.OwlPropertyDomainAxiom method), 46  
\_\_repr\_\_() (owlapy.model.OwlPropertyRangeAxiom method), 46  
\_\_repr\_\_() (owlapy.model.OwlSubAnnotationPropertyOfAxiom method), 41  
\_\_repr\_\_() (owlapy.model.OwlSubClassOfAxiom method), 38  
\_\_repr\_\_() (owlapy.model.OwlSubPropertyAxiom method), 42  
\_\_repr\_\_() (owlapy.namespaces.Namespaces method), 62  
\_\_repr\_\_() (owlapy.owl\_class\_expression.OwlObjectComplementOf method), 66  
\_\_repr\_\_() (owlapy.owl\_property.OwlObjectInverseOf method), 69  
\_\_repr\_\_() (owlapy.owlobject.OwlNamedObject method), 71  
\_\_repr\_\_() (owlapy.owlobject.OwlObject method), 70  
\_\_setitem\_\_() (owlapy.util.LRUCache method), 80  
\_\_slots\_\_ (owlapy.has.HasFiller attribute), 60  
\_\_slots\_\_ (owlapy.has.HasIRI attribute), 60  
\_\_slots\_\_ (owlapy.has.HasOperands attribute), 60  
\_\_slots\_\_ (owlapy.iri.IRI attribute), 61  
\_\_slots\_\_ (owlapy.model.AddImport attribute), 49  
\_\_slots\_\_ (owlapy.model.HasCardinality attribute), 21  
\_\_slots\_\_ (owlapy.model.HasFiller attribute), 13  
\_\_slots\_\_ (owlapy.model.HasIRI attribute), 12  
\_\_slots\_\_ (owlapy.model.HasOperands attribute), 12  
\_\_slots\_\_ (owlapy.model.IRI attribute), 11  
\_\_slots\_\_ (owlapy.model.OwlAnnotation attribute), 39  
\_\_slots\_\_ (owlapy.model.OwlAnnotationAssertionAxiom attribute), 40  
\_\_slots\_\_ (owlapy.model.OwlAnnotationAxiom attribute), 39  
\_\_slots\_\_ (owlapy.model.OwlAnnotationObject attribute), 10  
\_\_slots\_\_ (owlapy.model.OwlAnnotationProperty attribute), 39  
\_\_slots\_\_ (owlapy.model.OwlAnnotationPropertyDomainAxiom attribute), 41  
\_\_slots\_\_ (owlapy.model.OwlAnnotationPropertyRangeAxiom attribute), 41  
\_\_slots\_\_ (owlapy.model.OwlAnnotationSubject attribute), 10  
\_\_slots\_\_ (owlapy.model.OwlAnnotationValue attribute), 11  
\_\_slots\_\_ (owlapy.model.OwlAsymmetricObjectPropertyAxiom attribute), 44  
\_\_slots\_\_ (owlapy.model.OwlAxiom attribute), 25  
\_\_slots\_\_ (owlapy.model.OwlBooleanClassExpression attribute), 15  
\_\_slots\_\_ (owlapy.model.OwlCardinalityRestriction attribute), 21  
\_\_slots\_\_ (owlapy.model.OwlClass attribute), 15  
\_\_slots\_\_ (owlapy.model.OwlClassAssertionAxiom attribute), 39  
\_\_slots\_\_ (owlapy.model.OwlClassAxiom attribute), 33  
\_\_slots\_\_ (owlapy.model.OwlClassExpression attribute), 13  
\_\_slots\_\_ (owlapy.model.OwlData.AllValuesFrom attribute), 29  
\_\_slots\_\_ (owlapy.model.OwlDataCardinalityRestriction attribute), 29  
\_\_slots\_\_ (owlapy.model.OwlDataExactCardinality attribute), 30  
\_\_slots\_\_ (owlapy.model.OwlDataHasValue attribute), 30  
\_\_slots\_\_ (owlapy.model.OwlDataIntersectionOf attribute), 32  
\_\_slots\_\_ (owlapy.model.OwlDataMaxCardinality attribute), 30  
\_\_slots\_\_ (owlapy.model.OwlDataMinCardinality attribute), 31  
\_\_slots\_\_ (owlapy.model.OwlDataProperty attribute), 17  
\_\_slots\_\_ (owlapy.model.OwlDataPropertyAssertionAxiom attribute), 43  
\_\_slots\_\_ (owlapy.model.OwlDataPropertyAxiom attribute), 33  
\_\_slots\_\_ (owlapy.model.OwlDataPropertyCharacteristicAxiom attribute), 45  
\_\_slots\_\_ (owlapy.model.OwlDataPropertyDomainAxiom attribute), 46  
\_\_slots\_\_ (owlapy.model.OwlDataPropertyExpression attribute), 17  
\_\_slots\_\_ (owlapy.model.OwlDataPropertyRangeAxiom attribute), 47  
\_\_slots\_\_ (owlapy.model.OwlDataRestriction attribute), 18  
\_\_slots\_\_ (owlapy.model.OwlDataSomeValuesFrom attribute), 31  
\_\_slots\_\_ (owlapy.model.OwlDatatype attribute), 25  
\_\_slots\_\_ (owlapy.model.OwlDatatypeDefinitionAxiom attribute), 34  
\_\_slots\_\_ (owlapy.model.OwlDatatypeRestriction attribute), 26  
\_\_slots\_\_ (owlapy.model.OwlDataUnionOf attribute), 32  
\_\_slots\_\_ (owlapy.model.OwlDeclarationAxiom attribute), 33  
\_\_slots\_\_ (owlapy.model.OwlDifferentIndividualsAxiom attribute), 36  
\_\_slots\_\_ (owlapy.model.OwlDisjointClassesAxiom attribute), 36  
\_\_slots\_\_ (owlapy.model.OwlDisjointDataPropertiesAxiom attribute), 38

\_\_slots\_\_ (owlapy.model.OWLDisjointObjectPropertiesAxiom attribute), 37  
 \_\_slots\_\_ (owlapy.model.OWLDisjointUnionAxiom attribute), 38  
 \_\_slots\_\_ (owlapy.model.OWLEntity attribute), 10  
 \_\_slots\_\_ (owlapy.model.OWLEquivalentClassesAxiom attribute), 35  
 \_\_slots\_\_ (owlapy.model.OWLEquivalentDataPropertiesAxiom attribute), 38  
 \_\_slots\_\_ (owlapy.model.OWLEquivalentObjectPropertiesAxiom attribute), 37  
 \_\_slots\_\_ (owlapy.model.OWLFacetRestriction attribute), 26  
 \_\_slots\_\_ (owlapy.model.OWLFunctionalDataPropertyAxiom attribute), 45  
 \_\_slots\_\_ (owlapy.model.OWLFunctionalObjectPropertyAxiom attribute), 44  
 \_\_slots\_\_ (owlapy.model.OWLHasKeyAxiom attribute), 34  
 \_\_slots\_\_ (owlapy.model.OWLHasValueRestriction attribute), 19  
 \_\_slots\_\_ (owlapy.model.OWLImportsDeclaration attribute), 32  
 \_\_slots\_\_ (owlapy.model.OWLIndividual attribute), 23  
 \_\_slots\_\_ (owlapy.model.OWLIndividualAxiom attribute), 33  
 \_\_slots\_\_ (owlapy.model.OWLInverseFunctionalObjectPropertyAxiom attribute), 44  
 \_\_slots\_\_ (owlapy.model.OWLInverseObjectPropertiesAxiom attribute), 37  
 \_\_slots\_\_ (owlapy.model.OWLIrreflexiveObjectPropertyAxiom attribute), 44  
 \_\_slots\_\_ (owlapy.model.OWLLiteral attribute), 26  
 \_\_slots\_\_ (owlapy.model.OWLLogicalAxiom attribute), 33  
 \_\_slots\_\_ (owlapy.model.OWLNamedIndividual attribute), 24  
 \_\_slots\_\_ (owlapy.model.OWLNaryAxiom attribute), 35  
 \_\_slots\_\_ (owlapy.model.OWLNaryBooleanClassExpression attribute), 20  
 \_\_slots\_\_ (owlapy.model.OWLNaryClassAxiom attribute), 35  
 \_\_slots\_\_ (owlapy.model.OWLNaryDataRange attribute), 32  
 \_\_slots\_\_ (owlapy.model.OWLNaryIndividualAxiom attribute), 36  
 \_\_slots\_\_ (owlapy.model.OWLNaryPropertyAxiom attribute), 37  
 \_\_slots\_\_ (owlapy.model.OWLNegativeDataPropertyAssertionAxiom attribute), 43  
 \_\_slots\_\_ (owlapy.model.OWLNegativeObjectPropertyAssertionAxiom attribute), 43  
 \_\_slots\_\_ (owlapy.model.OWLObject attribute), 10  
 \_\_slots\_\_ (owlapy.model.OWLObjectAllValuesFrom attribute), 20  
 \_\_slots\_\_ (owlapy.model.OWLObjectCardinalityRestriction attribute), 22  
 \_\_slots\_\_ (owlapy.model.OWLObjectComplementOf attribute), 14  
 \_\_slots\_\_ (owlapy.model.OWLObjectExactCardinality attribute), 22  
 \_\_slots\_\_ (owlapy.model.OWLObjectHasSelf attribute), 23  
 \_\_slots\_\_ (owlapy.model.OWLObjectHasValue attribute), 23  
 \_\_slots\_\_ (owlapy.model.OWLObjectIntersectionOf attribute), 21  
 \_\_slots\_\_ (owlapy.model.OWLObjectMaxCardinality attribute), 22  
 \_\_slots\_\_ (owlapy.model.OWLObjectMinCardinality attribute), 22  
 \_\_slots\_\_ (owlapy.model.OWLObjectOneOf attribute), 24  
 \_\_slots\_\_ (owlapy.model.OWLObjectProperty attribute), 17  
 \_\_slots\_\_ (owlapy.model.OWLObjectPropertyAssertionAxiom attribute), 43  
 \_\_slots\_\_ (owlapy.model.OWLObjectPropertyAxiom attribute), 33  
 \_\_slots\_\_ (owlapy.model.OWLObjectPropertyCharacteristicAxiom attribute), 44  
 \_\_slots\_\_ (owlapy.model.OWLObjectPropertyDomainAxiom attribute), 46  
 \_\_slots\_\_ (owlapy.model.OWLObjectPropertyExpression attribute), 16  
 \_\_slots\_\_ (owlapy.model.OWLObjectPropertyRangeAxiom attribute), 46  
 \_\_slots\_\_ (owlapy.model.OWLObjectRestriction attribute), 18  
 \_\_slots\_\_ (owlapy.model.OWLObjectSomeValuesFrom attribute), 20  
 \_\_slots\_\_ (owlapy.model.OWLObjectUnionOf attribute), 21  
 \_\_slots\_\_ (owlapy.model.OWLOntology attribute), 47  
 \_\_slots\_\_ (owlapy.model.OWLOntologyChange attribute), 49  
 \_\_slots\_\_ (owlapy.model.OWLOntologyID attribute), 25  
 \_\_slots\_\_ (owlapy.model.OWLProperty attribute), 16  
 \_\_slots\_\_ (owlapy.model.OWLPropertyAssertionAxiom attribute), 42  
 \_\_slots\_\_ (owlapy.model.OWLPropertyAxiom attribute), 33  
 \_\_slots\_\_ (owlapy.model.OWLPropertyDomainAxiom attribute), 45  
 \_\_slots\_\_ (owlapy.model.OWLPropertyExpression attribute), 16  
 \_\_slots\_\_ (owlapy.model.OWLPropertyRangeAxiom attribute), 46  
 \_\_slots\_\_ (owlapy.model.OWLQuantifiedDataRestriction attribute), 28  
 \_\_slots\_\_ (owlapy.model.OWLQuantifiedObjectRestriction attribute), 19  
 \_\_slots\_\_ (owlapy.model.OWLQuantifiedRestriction attribute), 19  
 \_\_slots\_\_ (owlapy.model.OWLReasoner attribute), 50  
 \_\_slots\_\_ (owlapy.model.OWLReflexiveObjectPropertyAxiom attribute), 44  
 \_\_slots\_\_ (owlapy.model.OWLRestriction attribute), 18  
 \_\_slots\_\_ (owlapy.model.OWLSameIndividualAxiom attribute), 36  
 \_\_slots\_\_ (owlapy.model.OWLSubAnnotationPropertyOfAxiom attribute), 41  
 \_\_slots\_\_ (owlapy.model.OWLSubClassOfAxiom attribute), 38  
 \_\_slots\_\_ (owlapy.model.OWLSubDataPropertyOfAxiom attribute), 42



- \_\_slots\_\_ (owlapy.model.OWLSubObjectPropertyOfAxiom attribute), 42
- \_\_slots\_\_ (owlapy.model.OWLSubPropertyAxiom attribute), 42
- \_\_slots\_\_ (owlapy.model.OWLSymmetricObjectPropertyAxiom attribute), 45
- \_\_slots\_\_ (owlapy.model.OWLTransitiveObjectPropertyAxiom attribute), 45
- \_\_slots\_\_ (owlapy.model.OWLUnaryPropertyAxiom attribute), 43
- \_\_slots\_\_ (owlapy.namespaces.Namespaces attribute), 62
- \_\_slots\_\_ (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58
- \_\_slots\_\_ (owlapy.owl2sparql.converter.VariablesMapping attribute), 57
- \_\_slots\_\_ (owlapy.owl\_annotation.OWLAnnotationObject attribute), 63
- \_\_slots\_\_ (owlapy.owl\_annotation.OWLAnnotationSubject attribute), 63
- \_\_slots\_\_ (owlapy.owl\_annotation.OWLAnnotationValue attribute), 63
- \_\_slots\_\_ (owlapy.owl\_class\_expression.OWLBooleanClassExpression attribute), 65
- \_\_slots\_\_ (owlapy.owl\_class\_expression.OWLClass attribute), 66
- \_\_slots\_\_ (owlapy.owl\_class\_expression.OWLClassExpression attribute), 64
- \_\_slots\_\_ (owlapy.owl\_class\_expression.OWLObjectComplementOf attribute), 65
- \_\_slots\_\_ (owlapy.owl\_property.OWLDataProperty attribute), 70
- \_\_slots\_\_ (owlapy.owl\_property.OWLDataPropertyExpression attribute), 68
- \_\_slots\_\_ (owlapy.owl\_property.OWLObjectInverseOf attribute), 69
- \_\_slots\_\_ (owlapy.owl\_property.OWLObjectProperty attribute), 68
- \_\_slots\_\_ (owlapy.owl\_property.OWLObjectPropertyExpression attribute), 68
- \_\_slots\_\_ (owlapy.owl\_property.OWLProperty attribute), 68
- \_\_slots\_\_ (owlapy.owl\_property.OWLPropertyExpression attribute), 67
- \_\_slots\_\_ (owlapy.owlobject.OWLEntity attribute), 71
- \_\_slots\_\_ (owlapy.owlobject.OWLNamedObject attribute), 71
- \_\_slots\_\_ (owlapy.owlobject.OWLObject attribute), 70
- \_\_slots\_\_ (owlapy.render.DLSyntaxObjectRenderer attribute), 77
- \_\_slots\_\_ (owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer attribute), 78
- \_\_slots\_\_ (owlapy.util.OrderedOWLObject attribute), 79
- \_\_version\_\_ (in module owlapy), 82

## A

- add\_axiom() (owlapy.model.OWLOntologyManager method), 50
- AddImport (class in owlapy.model), 49
- annotations() (owlapy.model.OWLXiom method), 25
- append() (owlapy.owl2sparql.converter.Owl2SparqlConverter method), 59
- append\_triple() (owlapy.owl2sparql.converter.Owl2SparqlConverter method), 59
- apply\_change() (owlapy.model.OWLOntologyManager method), 49
- as\_anonymous\_individual() (owlapy.model.OWLAnnotationObject method), 10
- as\_anonymous\_individual() (owlapy.owl\_annotation.OWLAnnotationObject method), 63
- as\_index() (in module owlapy.util), 80
- as\_intersection\_of\_min\_max() (owlapy.model.OWLDataExactCardinality method), 30
- as\_intersection\_of\_min\_max() (owlapy.model.OWLObjectExactCardinality method), 23
- as\_iri() (owlapy.iri.IRI method), 61
- as\_iri() (owlapy.model.IRI method), 12
- as\_iri() (owlapy.model.OWLAnnotationObject method), 10
- as\_iri() (owlapy.owl\_annotation.OWLAnnotationObject method), 63
- as\_literal() (owlapy.model.OWLAnnotationValue method), 11
- as\_literal() (owlapy.model.OWLLiteral method), 28
- as\_literal() (owlapy.owl\_annotation.OWLAnnotationValue method), 64
- as\_object\_union\_of() (owlapy.model.OWLObjectOneOf method), 24
- as\_pairwise\_axioms() (owlapy.model.OWLNaryAxiom method), 35
- as\_pairwise\_axioms() (owlapy.model.OWLNaryClassAxiom method), 35
- as\_pairwise\_axioms() (owlapy.model.OWLNaryIndividualAxiom method), 36
- as\_pairwise\_axioms() (owlapy.model.OWLNaryPropertyAxiom method), 37
- as\_query() (owlapy.owl2sparql.converter.Owl2SparqlConverter method), 59
- as\_some\_values\_from() (owlapy.model.OWLDataHasValue method), 30
- as\_some\_values\_from() (owlapy.model.OWLObjectHasValue method), 23
- as\_str() (owlapy.iri.IRI method), 61
- as\_str() (owlapy.model.IRI method), 12

## B

- BOOLEAN (owlapy.model.XSDVocabulary attribute), 9
- BOOLEAN (owlapy.vocab.XSDVocabulary attribute), 81
- BooleanOWLDatatype (in module owlapy.model), 56

## C

- cache\_clear() (owlapy.util.LRUCache method), 80



cache\_info() (owlapy.util.LRUCache method), 80  
 ce (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58  
 class\_expressions() (owlapy.model.OwLNaryClassAxiom method), 35  
 classes\_in\_signature() (owlapy.model.OwLOntology method), 47  
 cnt (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58  
 combine\_nary\_expressions() (in module owlapy.util), 80  
 contains\_named\_equivalent\_class() (owlapy.model.OwLEquivalentClassesAxiom method), 35  
 contains\_owl\_nothing() (owlapy.model.OwLEquivalentClassesAxiom method), 35  
 contains\_owl\_thing() (owlapy.model.OwLEquivalentClassesAxiom method), 36  
 convert() (owlapy.owl2sparql.converter.Owl2SparqlConverter method), 58  
 converter (in module owlapy.owl2sparql.converter), 59  
 create() (owlapy.iri.IRI static method), 61  
 create() (owlapy.model.IRI static method), 11  
 create\_ontology() (owlapy.model.OwLOntologyManager method), 49  
 current\_variable (owlapy.owl2sparql.converter.Owl2SparqlConverter property), 58

## D

data\_properties\_in\_signature() (owlapy.model.OwLOntology method), 47  
 data\_property\_domain\_axioms() (owlapy.model.OwLOntology method), 48  
 data\_property\_domains() (owlapy.model.OwLReasoner method), 50  
 data\_property\_range\_axioms() (owlapy.model.OwLOntology method), 48  
 data\_property\_values() (owlapy.model.OwLReasoner method), 52  
 DATE (owlapy.model.XSDVocabulary attribute), 9  
 DATE (owlapy.vocab.XSDVocabulary attribute), 81  
 DATE\_TIME (owlapy.model.XSDVocabulary attribute), 9  
 DATE\_TIME (owlapy.vocab.XSDVocabulary attribute), 81  
 DATE\_TIME\_STAMP (owlapy.model.XSDVocabulary attribute), 9  
 DATE\_TIME\_STAMP (owlapy.vocab.XSDVocabulary attribute), 82  
 DateOWLDatatype (in module owlapy.model), 56  
 DateTimeOWLDatatype (in module owlapy.model), 56  
 DECIMAL (owlapy.model.XSDVocabulary attribute), 9  
 DECIMAL (owlapy.vocab.XSDVocabulary attribute), 81  
 different\_individuals() (owlapy.model.OwLReasoner method), 52  
 disjoint\_classes() (owlapy.model.OwLReasoner method), 51  
 disjoint\_data\_properties() (owlapy.model.OwLReasoner method), 54  
 disjoint\_object\_properties() (owlapy.model.OwLReasoner method), 54  
 DL\_GRAMMAR (in module owlapy.parser), 74  
 dl\_to\_owl\_expression() (in module owlapy.parser), 76  
 DLparser (in module owlapy.parser), 76  
 DLrenderer (in module owlapy.render), 78  
 DLSyntaxObjectRenderer (class in owlapy.render), 77  
 DLSyntaxParser (class in owlapy.parser), 74  
 DOUBLE (owlapy.model.XSDVocabulary attribute), 9  
 DOUBLE (owlapy.vocab.XSDVocabulary attribute), 81  
 DoubleOWLDatatype (in module owlapy.model), 56  
 DURATION (owlapy.model.XSDVocabulary attribute), 9  
 DURATION (owlapy.vocab.XSDVocabulary attribute), 82  
 DurationOWLDatatype (in module owlapy.model), 56

## E

equivalent\_classes() (owlapy.model.OwLReasoner method), 51  
 equivalent\_classes\_axioms() (owlapy.model.OwLOntology method), 47  
 equivalent\_data\_properties() (owlapy.model.OwLReasoner method), 52  
 equivalent\_object\_properties() (owlapy.model.OwLReasoner method), 52

## F

FLOAT (owlapy.model.XSDVocabulary attribute), 9  
 FLOAT (owlapy.vocab.XSDVocabulary attribute), 81  
 flush() (owlapy.model.OwLReasoner method), 53  
 FRACTION\_DIGITS (owlapy.model.OwLFacet attribute), 9  
 FRACTION\_DIGITS (owlapy.vocab.OwLFacet attribute), 82  
 from\_str() (owlapy.model.OwLFacet static method), 10  
 from\_str() (owlapy.vocab.OwLFacet static method), 82

## G

general\_class\_axioms() (owlapy.model.OwLOntology method), 48

`generic_visit()` (*owlapy.parser.DLSyntaxParser method*), 76  
`generic_visit()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`get_cardinality()` (*owlapy.model.HasCardinality method*), 21  
`get_cardinality()` (*owlapy.model.OWLCardinalityRestriction method*), 21  
`get_class_expression()` (*owlapy.model.OWLClassAssertionAxiom method*), 39  
`get_class_expression()` (*owlapy.model.OWLHasKeyAxiom method*), 34  
`get_class_expressions()` (*owlapy.model.OWLDisjointUnionAxiom method*), 38  
`get_class_nnf()` (*owlapy.util.NNF method*), 79  
`get_data_range()` (*owlapy.model.OWLDataComplementOf method*), 29  
`get_datarange()` (*owlapy.model.OWLDatatypeDefinitionAxiom method*), 34  
`get_datatype()` (*owlapy.model.OWLDatatypeDefinitionAxiom method*), 34  
`get_datatype()` (*owlapy.model.OWLDatatypeRestriction method*), 26  
`get_datatype()` (*owlapy.model.OWLLiteral method*), 28  
`get_default_document_iri()` (*owlapy.model.OWLOntologyID method*), 25  
`get_domain()` (*owlapy.model.OWLAnnotationPropertyDomainAxiom method*), 41  
`get_domain()` (*owlapy.model.OWLPropertyDomainAxiom method*), 45  
`get_entity()` (*owlapy.model.OWLDeclarationAxiom method*), 33  
`get_facet()` (*owlapy.model.OWLFacetRestriction method*), 26  
`get_facet_restrictions()` (*owlapy.model.OWLDatatypeRestriction method*), 26  
`get_facet_value()` (*owlapy.model.OWLFacetRestriction method*), 26  
`get_filler()` (*owlapy.has.HasFiller method*), 60  
`get_filler()` (*owlapy.model.HasFiller method*), 13  
`get_filler()` (*owlapy.model.OWLCardinalityRestriction method*), 21  
`get_filler()` (*owlapy.model.OWLHasValueRestriction method*), 19  
`get_filler()` (*owlapy.model.OWLQuantifiedDataRestriction method*), 28  
`get_filler()` (*owlapy.model.OWLQuantifiedObjectRestriction method*), 19  
`get_first_property()` (*owlapy.model.OWLInverseObjectPropertiesAxiom method*), 37  
`get_import_declaration()` (*owlapy.model.AddImport method*), 49  
`get_individual()` (*owlapy.model.OWLClassAssertionAxiom method*), 39  
`get_inverse()` (*owlapy.owl\_property.OWLObjectInverseOf method*), 69  
`get_inverse_property()` (*owlapy.model.OWLObjectProperty method*), 17  
`get_inverse_property()` (*owlapy.model.OWLObjectPropertyExpression method*), 16  
`get_inverse_property()` (*owlapy.owl\_property.OWLObjectInverseOf method*), 69  
`get_inverse_property()` (*owlapy.owl\_property.OWLObjectProperty method*), 69  
`get_inverse_property()` (*owlapy.owl\_property.OWLObjectPropertyExpression method*), 68  
`get_iri()` (*owlapy.has.HasIRI method*), 60  
`get_iri()` (*owlapy.model.HasIRI method*), 12  
`get_iri()` (*owlapy.model.OWLAnnotationProperty method*), 39  
`get_iri()` (*owlapy.model.OWLClass method*), 15  
`get_iri()` (*owlapy.model.OWLDataProperty method*), 17  
`get_iri()` (*owlapy.model.OWLDatatype method*), 26  
`get_iri()` (*owlapy.model.OWLImportsDeclaration method*), 32  
`get_iri()` (*owlapy.model.OWLNamedIndividual method*), 24  
`get_iri()` (*owlapy.model.OWLObjectProperty method*), 18  
`get_iri()` (*owlapy.owl\_class\_expression.OWLClass method*), 66  
`get_iri()` (*owlapy.owl\_property.OWLDataProperty method*), 70  
`get_iri()` (*owlapy.owl\_property.OWLObjectProperty method*), 69  
`get_literal()` (*owlapy.model.OWLLiteral method*), 27  
`get_named_property()` (*owlapy.model.OWLObjectProperty method*), 17  
`get_named_property()` (*owlapy.model.OWLObjectPropertyExpression method*), 16  
`get_named_property()` (*owlapy.owl\_property.OWLObjectInverseOf method*), 69  
`get_named_property()` (*owlapy.owl\_property.OWLObjectProperty method*), 68  
`get_named_property()` (*owlapy.owl\_property.OWLObjectPropertyExpression method*), 68  
`get_namespace()` (*owlapy.iri.IRI method*), 62  
`get_namespace()` (*owlapy.model.IRI method*), 12  
`get_nnf()` (*owlapy.model.OWLAnonymousClassExpression method*), 14  
`get_nnf()` (*owlapy.model.OWLClass method*), 15  
`get_nnf()` (*owlapy.model.OWLClassExpression method*), 13  
`get_nnf()` (*owlapy.owl\_class\_expression.OWLAnonymousClassExpression method*), 65  
`get_nnf()` (*owlapy.owl\_class\_expression.OWLClass method*), 66  
`get_nnf()` (*owlapy.owl\_class\_expression.OWLClassExpression method*), 65  
`get_object()` (*owlapy.model.OWLPropertyAssertionAxiom method*), 42  
`get_object_complement_of()` (*owlapy.model.OWLAnonymousClassExpression method*), 14  
`get_object_complement_of()` (*owlapy.model.OWLClass method*), 15  
`get_object_complement_of()` (*owlapy.model.OWLClassExpression method*), 13  
`get_object_complement_of()` (*owlapy.owl\_class\_expression.OWLAnonymousClassExpression method*), 65  
`get_object_complement_of()` (*owlapy.owl\_class\_expression.OWLClass method*), 66  
`get_object_complement_of()` (*owlapy.owl\_class\_expression.OWLClassExpression method*), 64

`get_ontology()` (*owlapy.model.OWLontologyChange method*), 49  
`get_ontology_id()` (*owlapy.model.OWLontology method*), 48  
`get_ontology_iri()` (*owlapy.model.OWLontologyID method*), 25  
`get_operand()` (*owlapy.model.OWLObjectComplementOf method*), 14  
`get_operand()` (*owlapy.owl\_class\_expression.OWLObjectComplementOf method*), 65  
`get_owl_class()` (*owlapy.model.OWLDisjointUnionAxiom method*), 38  
`get_owl_disjoint_classes_axiom()` (*owlapy.model.OWLDisjointUnionAxiom method*), 38  
`get_owl_equivalent_classes_axiom()` (*owlapy.model.OWLDisjointUnionAxiom method*), 38  
`get_owl_ontology_manager()` (*owlapy.model.OWLontology method*), 48  
`get_property()` (*owlapy.model.OWLAnnotation method*), 40  
`get_property()` (*owlapy.model.OWLAnnotationAssertionAxiom method*), 40  
`get_property()` (*owlapy.model.OWLAnnotationPropertyDomainAxiom method*), 41  
`get_property()` (*owlapy.model.OWLAnnotationPropertyRangeAxiom method*), 41  
`get_property()` (*owlapy.model.OWLDataAllValuesFrom method*), 29  
`get_property()` (*owlapy.model.OWLDataCardinalityRestriction method*), 29  
`get_property()` (*owlapy.model.OWLDataHasValue method*), 30  
`get_property()` (*owlapy.model.OWLDataSomeValuesFrom method*), 31  
`get_property()` (*owlapy.model.OWLObjectAllValuesFrom method*), 20  
`get_property()` (*owlapy.model.OWLObjectCardinalityRestriction method*), 22  
`get_property()` (*owlapy.model.OWLObjectHasSelf method*), 23  
`get_property()` (*owlapy.model.OWLObjectHasValue method*), 23  
`get_property()` (*owlapy.model.OWLObjectRestriction method*), 19  
`get_property()` (*owlapy.model.OWLObjectSomeValuesFrom method*), 20  
`get_property()` (*owlapy.model.OWLPropertyAssertionAxiom method*), 42  
`get_property()` (*owlapy.model.OWLRestriction method*), 18  
`get_property()` (*owlapy.model.OWLUnaryPropertyAxiom method*), 43  
`get_property_expressions()` (*owlapy.model.OWLHasKeyAxiom method*), 34  
`get_range()` (*owlapy.model.OWLAnnotationPropertyRangeAxiom method*), 41  
`get_range()` (*owlapy.model.OWLPropertyRangeAxiom method*), 46  
`get_remainder()` (*owlapy.iri.IRI method*), 62  
`get_remainder()` (*owlapy.model.IRI method*), 12  
`get_root_ontology()` (*owlapy.model.OWLReasoner method*), 55  
`get_second_property()` (*owlapy.model.OWLInverseObjectPropertiesAxiom method*), 37  
`get_short_form()` (*owlapy.iri.IRI method*), 61  
`get_short_form()` (*owlapy.model.IRI method*), 12  
`get_sub_class()` (*owlapy.model.OWLSubClassOfAxiom method*), 38  
`get_sub_property()` (*owlapy.model.OWLSubAnnotationPropertyOfAxiom method*), 41  
`get_sub_property()` (*owlapy.model.OWLSubPropertyAxiom method*), 42  
`get_subject()` (*owlapy.model.OWLAnnotationAssertionAxiom method*), 40  
`get_subject()` (*owlapy.model.OWLPropertyAssertionAxiom method*), 42  
`get_super_class()` (*owlapy.model.OWLSubClassOfAxiom method*), 38  
`get_super_property()` (*owlapy.model.OWLSubAnnotationPropertyOfAxiom method*), 41  
`get_super_property()` (*owlapy.model.OWLSubPropertyAxiom method*), 42  
`get_top_level_cnf()` (*owlapy.util.TopLevelCNF method*), 79  
`get_top_level_dnf()` (*owlapy.util.TopLevelDNF method*), 79  
`get_value()` (*owlapy.model.OWLAnnotation method*), 40  
`get_value()` (*owlapy.model.OWLAnnotationAssertionAxiom method*), 40  
`get_variable()` (*owlapy.owl2sparql.converter.VariablesMapping method*), 57  
`get_version_iri()` (*owlapy.model.OWLontologyID method*), 25  
`grouping_vars` (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58

## H

`HasCardinality` (*class in owlapy.model*), 21  
`HasFiller` (*class in owlapy.has*), 60  
`HasFiller` (*class in owlapy.model*), 13  
`HasIndex` (*class in owlapy.has*), 59  
`HasIndex` (*class in owlapy.model*), 12  
`HasIRI` (*class in owlapy.has*), 59  
`HasIRI` (*class in owlapy.model*), 12  
`HasOperands` (*class in owlapy.has*), 60  
`HasOperands` (*class in owlapy.model*), 12  
`having_conditions` (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58

## I

`individuals()` (*owlapy.model.OWLNaryIndividualAxiom method*), 36  
`individuals()` (*owlapy.model.OWLObjectOneOf method*), 24  
`individuals_in_signature()` (*owlapy.model.OWLontology method*), 47

instances() (owlapy.model.OWLReasoner method), 53  
 INTEGER (owlapy.model.XSDVocabulary attribute), 9  
 INTEGER (owlapy.vocab.XSDVocabulary attribute), 81  
 IntegerOWLDatatype (in module owlapy.model), 56  
 IRI (class in owlapy.iri), 60  
 IRI (class in owlapy.model), 11  
 iri (owlapy.model.OWLNamedIndividual property), 24  
 iri (owlapy.model.OWLObjectProperty property), 17  
 iri (owlapy.owl\_property.OWLObjectProperty property), 68  
 is\_annotated() (owlapy.model.OWL Axiom method), 25  
 is\_annotation\_axiom() (owlapy.model.OWLAnnotationAxiom method), 39  
 is\_annotation\_axiom() (owlapy.model.OWL Axiom method), 25  
 is\_anonymous() (owlapy.model.OWLEntity method), 10  
 is\_anonymous() (owlapy.model.OWLObject method), 10  
 is\_anonymous() (owlapy.model.OWLOntology method), 48  
 is\_anonymous() (owlapy.model.OWLOntologyID method), 25  
 is\_anonymous() (owlapy.owlobject.OWLEntity method), 72  
 is\_anonymous() (owlapy.owlobject.OWLObject method), 71  
 is\_boolean() (owlapy.model.OWLLiteral method), 27  
 is\_data\_property\_expression() (owlapy.model.OWLDataPropertyExpression method), 17  
 is\_data\_property\_expression() (owlapy.model.OWLPropertyExpression method), 16  
 is\_data\_property\_expression() (owlapy.owl\_property.OWLDataPropertyExpression method), 68  
 is\_data\_property\_expression() (owlapy.owl\_property.OWLPropertyExpression method), 67  
 is\_data\_restriction() (owlapy.model.OWLDataRestriction method), 18  
 is\_data\_restriction() (owlapy.model.OWLRestriction method), 18  
 is\_date() (owlapy.model.OWLLiteral method), 27  
 is\_datetime() (owlapy.model.OWLLiteral method), 27  
 is\_double() (owlapy.model.OWLLiteral method), 27  
 is\_duration() (owlapy.model.OWLLiteral method), 28  
 is\_integer() (owlapy.model.OWLLiteral method), 27  
 is\_isolated() (owlapy.model.OWLReasoner method), 55  
 is\_literal() (owlapy.model.OWLAnnotationValue method), 11  
 is\_literal() (owlapy.model.OWLLiteral method), 28  
 is\_literal() (owlapy.owl\_annotation.OWLAnnotationValue method), 63  
 is\_logical\_axiom() (owlapy.model.OWL Axiom method), 25  
 is\_logical\_axiom() (owlapy.model.OWLLogicalAxiom method), 33  
 is\_nothing() (owlapy.iri.IRI method), 61  
 is\_nothing() (owlapy.model.IRI method), 11  
 is\_object\_property\_expression() (owlapy.model.OWLObjectPropertyExpression method), 16  
 is\_object\_property\_expression() (owlapy.model.OWLPropertyExpression method), 16  
 is\_object\_property\_expression() (owlapy.owl\_property.OWLObjectPropertyExpression method), 68  
 is\_object\_property\_expression() (owlapy.owl\_property.OWLPropertyExpression method), 67  
 is\_object\_restriction() (owlapy.model.OWLObjectRestriction method), 18  
 is\_object\_restriction() (owlapy.model.OWLRestriction method), 18  
 is\_owl\_nothing() (owlapy.model.OWLAnonymousClassExpression method), 14  
 is\_owl\_nothing() (owlapy.model.OWLClass method), 15  
 is\_owl\_nothing() (owlapy.model.OWLClassExpression method), 13  
 is\_owl\_nothing() (owlapy.owl\_class\_expression.OWLAnonymousClassExpression method), 65  
 is\_owl\_nothing() (owlapy.owl\_class\_expression.OWLClass method), 66  
 is\_owl\_nothing() (owlapy.owl\_class\_expression.OWLClassExpression method), 64  
 is\_owl\_thing() (owlapy.model.OWLAnonymousClassExpression method), 14  
 is\_owl\_thing() (owlapy.model.OWLClass method), 15  
 is\_owl\_thing() (owlapy.model.OWLClassExpression method), 13  
 is\_owl\_thing() (owlapy.owl\_class\_expression.OWLAnonymousClassExpression method), 65  
 is\_owl\_thing() (owlapy.owl\_class\_expression.OWLClass method), 66  
 is\_owl\_thing() (owlapy.owl\_class\_expression.OWLClassExpression method), 64  
 is\_owl\_top\_data\_property() (owlapy.model.OWLDataProperty method), 17  
 is\_owl\_top\_data\_property() (owlapy.model.OWLPropertyExpression method), 16  
 is\_owl\_top\_data\_property() (owlapy.owl\_property.OWLDataProperty method), 70  
 is\_owl\_top\_data\_property() (owlapy.owl\_property.OWLPropertyExpression method), 67  
 is\_owl\_top\_object\_property() (owlapy.model.OWLObjectProperty method), 18  
 is\_owl\_top\_object\_property() (owlapy.model.OWLPropertyExpression method), 16  
 is\_owl\_top\_object\_property() (owlapy.owl\_property.OWLObjectProperty method), 69  
 is\_owl\_top\_object\_property() (owlapy.owl\_property.OWLPropertyExpression method), 67  
 is\_reserved\_vocabulary() (owlapy.iri.IRI method), 61  
 is\_reserved\_vocabulary() (owlapy.model.IRI method), 11  
 is\_string() (owlapy.model.OWLLiteral method), 27  
 is\_thing() (owlapy.iri.IRI method), 61

`is_thing()` (*owlapy.model.IRI method*), 11  
`is_using_triplestore()` (*owlapy.model.OWLReasoner method*), 56  
`iter_count()` (*in module owlapy.util*), 80

## K

`KEY` (*owlapy.util.LRUCache attribute*), 80

## L

`LENGTH` (*owlapy.model.OWLFacet attribute*), 9  
`LENGTH` (*owlapy.vocab.OWLFacet attribute*), 82  
`Literals` (*in module owlapy.model*), 18  
`load_ontology()` (*owlapy.model.OWLOntologyManager method*), 49  
`LONG` (*owlapy.model.XSDVocabulary attribute*), 9  
`LONG` (*owlapy.vocab.XSDVocabulary attribute*), 81  
`LRUCache` (*class in owlapy.util*), 80

## M

`MANCHESTER_GRAMMAR` (*in module owlapy.parser*), 72  
`manchester_to_owl_expression()` (*in module owlapy.parser*), 76  
`ManchesterOWLSyntaxOWLObjectRenderer` (*class in owlapy.render*), 77  
`ManchesterOWLSyntaxParser` (*class in owlapy.parser*), 72  
`ManchesterParser` (*in module owlapy.parser*), 76  
`ManchesterRenderer` (*in module owlapy.render*), 78  
`mapping` (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58  
`MAX_EXCLUSIVE` (*owlapy.model.OWLFacet attribute*), 9  
`MAX_EXCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 82  
`MAX_INCLUSIVE` (*owlapy.model.OWLFacet attribute*), 9  
`MAX_INCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 82  
`MAX_LENGTH` (*owlapy.model.OWLFacet attribute*), 9  
`MAX_LENGTH` (*owlapy.vocab.OWLFacet attribute*), 82  
`MIN_EXCLUSIVE` (*owlapy.model.OWLFacet attribute*), 9  
`MIN_EXCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 82  
`MIN_INCLUSIVE` (*owlapy.model.OWLFacet attribute*), 9  
`MIN_INCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 82  
`MIN_LENGTH` (*owlapy.model.OWLFacet attribute*), 9  
`MIN_LENGTH` (*owlapy.vocab.OWLFacet attribute*), 82  
`modal_depth` (*owlapy.owl2sparql.converter.Owl2SparqlConverter property*), 58  
`module`

- `owlapy`, 1
- `owlapy.has`, 59
- `owlapy.iri`, 60
- `owlapy.model`, 1
- `owlapy.model.providers`, 2
- `owlapy.namespaces`, 62
- `owlapy.owl2sparql`, 57
- `owlapy.owl2sparql.converter`, 57
- `owlapy.owl_annotation`, 63
- `owlapy.owl_class_expression`, 64
- `owlapy.owl_property`, 67
- `owlapy.owlobject`, 70
- `owlapy.parser`, 72
- `owlapy.render`, 77
- `owlapy.util`, 78
- `owlapy.vocab`, 81

`MOVE()` (*in module owlapy.model*), 10

## N

`named_classes()` (*owlapy.model.OWLEquivalentClassesAxiom method*), 36  
`Namespaces` (*class in owlapy.namespaces*), 62  
`new_count_var()` (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 59  
`new_individual_variable()` (*owlapy.owl2sparql.converter.VariablesMapping method*), 57  
`new_property_variable()` (*owlapy.owl2sparql.converter.VariablesMapping method*), 57  
`NEXT` (*owlapy.util.LRUCache attribute*), 80  
`NNF` (*class in owlapy.util*), 79  
`ns` (*owlapy.namespaces.Namespaces property*), 62



ns (*owlapy.parser.DLSyntaxParser* attribute), 75  
 ns (*owlapy.parser.ManchesterOWLSyntaxParser* attribute), 72  
 NUMERIC\_DATATYPES (in module *owlapy.model*), 56

## O

- o (*owlapy.util.OrderedOWLObject* attribute), 79
- object\_properties\_in\_signature() (*owlapy.model.OWLOntology* method), 47
- object\_property\_domain\_axioms() (*owlapy.model.OWLOntology* method), 48
- object\_property\_domains() (*owlapy.model.OWLReasoner* method), 50
- object\_property\_range\_axioms() (*owlapy.model.OWLOntology* method), 48
- object\_property\_ranges() (*owlapy.model.OWLReasoner* method), 51
- object\_property\_values() (*owlapy.model.OWLReasoner* method), 53
- operands() (*owlapy.has.HasOperands* method), 60
- operands() (*owlapy.model.HasOperands* method), 13
- operands() (*owlapy.model.OWLDataOneOf* method), 31
- operands() (*owlapy.model.OWLHasKeyAxiom* method), 34
- operands() (*owlapy.model.OWLNaryBooleanClassExpression* method), 20
- operands() (*owlapy.model.OWLNaryDataRange* method), 32
- operands() (*owlapy.model.OWLObjectComplementOf* method), 14
- operands() (*owlapy.model.OWLObjectOneOf* method), 24
- operands() (*owlapy.owl\_class\_expression.OWLObjectComplementOf* method), 66
- operator (*owlapy.model.OWLFacet* property), 9
- operator (*owlapy.vocab.OWLFacet* property), 82
- OrderedOWLObject (class in *owlapy.util*), 79
- OWL (in module *owlapy.namespaces*), 63
- Owl2SparqlConverter (class in *owlapy.owl2sparql.converter*), 58
- OWL\_BOTTOM\_DATA\_PROPERTY (*owlapy.model.OWLRDFVocabulary* attribute), 9
- OWL\_BOTTOM\_DATA\_PROPERTY (*owlapy.vocab.OWLRDFVocabulary* attribute), 81
- OWL\_BOTTOM\_OBJECT\_PROPERTY (*owlapy.model.OWLRDFVocabulary* attribute), 8
- OWL\_BOTTOM\_OBJECT\_PROPERTY (*owlapy.vocab.OWLRDFVocabulary* attribute), 81
- OWL\_CLASS (*owlapy.model.OWLRDFVocabulary* attribute), 8
- OWL\_CLASS (*owlapy.vocab.OWLRDFVocabulary* attribute), 81
- owl\_expression\_to\_dl() (in module *owlapy.render*), 78
- owl\_expression\_to\_manchester() (in module *owlapy.render*), 78
- owl\_expression\_to\_sparql() (in module *owlapy.owl2sparql.converter*), 59
- OWL\_NAMED\_INDIVIDUAL (*owlapy.model.OWLRDFVocabulary* attribute), 8
- OWL\_NAMED\_INDIVIDUAL (*owlapy.vocab.OWLRDFVocabulary* attribute), 81
- OWL\_NOTHING (*owlapy.model.OWLRDFVocabulary* attribute), 8
- OWL\_NOTHING (*owlapy.vocab.OWLRDFVocabulary* attribute), 81
- OWL\_THING (*owlapy.model.OWLRDFVocabulary* attribute), 8
- OWL\_THING (*owlapy.vocab.OWLRDFVocabulary* attribute), 81
- OWL\_TOP\_DATA\_PROPERTY (*owlapy.model.OWLRDFVocabulary* attribute), 8
- OWL\_TOP\_DATA\_PROPERTY (*owlapy.vocab.OWLRDFVocabulary* attribute), 81
- OWL\_TOP\_OBJECT\_PROPERTY (*owlapy.model.OWLRDFVocabulary* attribute), 8
- OWL\_TOP\_OBJECT\_PROPERTY (*owlapy.vocab.OWLRDFVocabulary* attribute), 81
- OWLAnnotation (class in *owlapy.model*), 39
- OWLAnnotationAssertionAxiom (class in *owlapy.model*), 40
- OWLAnnotationAxiom (class in *owlapy.model*), 39
- OWLAnnotationObject (class in *owlapy.model*), 10
- OWLAnnotationObject (class in *owlapy.owl\_annotation*), 63
- OWLAnnotationProperty (class in *owlapy.model*), 39
- OWLAnnotationPropertyDomainAxiom (class in *owlapy.model*), 41
- OWLAnnotationPropertyRangeAxiom (class in *owlapy.model*), 41
- OWLAnnotationSubject (class in *owlapy.model*), 10
- OWLAnnotationSubject (class in *owlapy.owl\_annotation*), 63
- OWLAnnotationValue (class in *owlapy.model*), 11
- OWLAnnotationValue (class in *owlapy.owl\_annotation*), 63
- OWLANonymousClassExpression (class in *owlapy.model*), 14
- OWLANonymousClassExpression (class in *owlapy.owl\_class\_expression*), 65
- owlapy
  - module, 1
- owlapy.has
  - module, 59
- owlapy.iri
  - module, 60
- owlapy.model
  - module, 1

- owlapy.model.providers
  - module, 2
- owlapy.namespaces
  - module, 62
- owlapy.owl2sparql
  - module, 57
- owlapy.owl2sparql.converter
  - module, 57
- owlapy.owl\_annotation
  - module, 63
- owlapy.owl\_class\_expression
  - module, 64
- owlapy.owl\_property
  - module, 67
- owlapy.owlobject
  - module, 70
- owlapy.parser
  - module, 72
- owlapy.render
  - module, 77
- owlapy.util
  - module, 78
- owlapy.vocab
  - module, 81
- OWLAsymmetricObjectPropertyAxiom (class in owlapy.model), 44
- OWLAxiom (class in owlapy.model), 25
- OWLBooleanClassExpression (class in owlapy.model), 14
- OWLBooleanClassExpression (class in owlapy.owl\_class\_expression), 65
- OWLBottomDataProperty (in module owlapy.model), 56
- OWLBottomObjectProperty (in module owlapy.model), 56
- OWLCardinalityRestriction (class in owlapy.model), 21
- OWLClass (class in owlapy.model), 15
- OWLClass (class in owlapy.owl\_class\_expression), 66
- OWLClassAssertionAxiom (class in owlapy.model), 39
- OWLClassAxiom (class in owlapy.model), 33
- OWLClassExpression (class in owlapy.model), 13
- OWLClassExpression (class in owlapy.owl\_class\_expression), 64
- OWLDataAllValuesFrom (class in owlapy.model), 29
- OWLDataCardinalityRestriction (class in owlapy.model), 28
- OWLDataComplementOf (class in owlapy.model), 29
- OWLDataExactCardinality (class in owlapy.model), 30
- OWLDataHasValue (class in owlapy.model), 30
- OWLDataIntersectionOf (class in owlapy.model), 32
- OWLDataMaxCardinality (class in owlapy.model), 30
- OWLDataMinCardinality (class in owlapy.model), 30
- OWLDataOneOf (class in owlapy.model), 31
- OWLDataProperty (class in owlapy.model), 17
- OWLDataProperty (class in owlapy.owl\_property), 70
- OWLDataPropertyAssertionAxiom (class in owlapy.model), 43
- OWLDataPropertyAxiom (class in owlapy.model), 33
- OWLDataPropertyCharacteristicAxiom (class in owlapy.model), 45
- OWLDataPropertyDomainAxiom (class in owlapy.model), 46
- OWLDataPropertyExpression (class in owlapy.model), 17
- OWLDataPropertyExpression (class in owlapy.owl\_property), 68
- OWLDataPropertyRangeAxiom (class in owlapy.model), 46
- OWLDataRange (class in owlapy.model), 15
- OWLDataRange (class in owlapy.owl\_class\_expression), 64
- OWLDataRestriction (class in owlapy.model), 18
- OWLDataSomeValuesFrom (class in owlapy.model), 31
- OWLDatatype (class in owlapy.model), 25
- OWLDatatypeDefinitionAxiom (class in owlapy.model), 34
- OWLDatatypeMaxExclusiveRestriction() (in module owlapy.model.providers), 2
- OWLDatatypeMaxInclusiveRestriction() (in module owlapy.model.providers), 2
- OWLDatatypeMinExclusiveRestriction() (in module owlapy.model.providers), 2
- OWLDatatypeMinInclusiveRestriction() (in module owlapy.model.providers), 2
- OWLDatatypeMinMaxExclusiveRestriction() (in module owlapy.model.providers), 2
- OWLDatatypeMinMaxInclusiveRestriction() (in module owlapy.model.providers), 2
- OWLDatatypeRestriction (class in owlapy.model), 26

OWLDataUnionOf (class in owlapy.model), 32  
 OWLDeclarationAxiom (class in owlapy.model), 33  
 OWLDifferentIndividualsAxiom (class in owlapy.model), 36  
 OWLDisjointClassesAxiom (class in owlapy.model), 36  
 OWLDisjointDataPropertiesAxiom (class in owlapy.model), 38  
 OWLDisjointObjectPropertiesAxiom (class in owlapy.model), 37  
 OWLDisjointUnionAxiom (class in owlapy.model), 38  
 OWLEntity (class in owlapy.model), 10  
 OWLEntity (class in owlapy.owlobject), 71  
 OWLEquivalentClassesAxiom (class in owlapy.model), 35  
 OWLEquivalentDataPropertiesAxiom (class in owlapy.model), 38  
 OWLEquivalentObjectPropertiesAxiom (class in owlapy.model), 37  
 OWLFacet (class in owlapy.model), 9  
 OWLFacet (class in owlapy.vocab), 82  
 OWLFacetRestriction (class in owlapy.model), 26  
 OWLFunctionalDataPropertyAxiom (class in owlapy.model), 45  
 OWLFunctionalObjectPropertyAxiom (class in owlapy.model), 44  
 OWLHasKeyAxiom (class in owlapy.model), 34  
 OWLHasValueRestriction (class in owlapy.model), 19  
 OWLImportsDeclaration (class in owlapy.model), 32  
 OWLIndividual (class in owlapy.model), 23  
 OWLIndividualAxiom (class in owlapy.model), 33  
 OWLInverseFunctionalObjectPropertyAxiom (class in owlapy.model), 44  
 OWLInverseObjectPropertiesAxiom (class in owlapy.model), 37  
 OWLIrreflexiveObjectPropertyAxiom (class in owlapy.model), 44  
 OWLLiteral (class in owlapy.model), 26  
 OWLLogicalAxiom (class in owlapy.model), 33  
 OWLNamedIndividual (class in owlapy.model), 24  
 OWLNamedObject (class in owlapy.owlobject), 71  
 OWLNaryAxiom (class in owlapy.model), 35  
 OWLNaryBooleanClassExpression (class in owlapy.model), 20  
 OWLNaryClassAxiom (class in owlapy.model), 35  
 OWLNaryDataRange (class in owlapy.model), 32  
 OWLNaryIndividualAxiom (class in owlapy.model), 36  
 OWLNaryPropertyAxiom (class in owlapy.model), 36  
 OWLNegativeDataPropertyAssertionAxiom (class in owlapy.model), 43  
 OWLNegativeObjectPropertyAssertionAxiom (class in owlapy.model), 43  
 OWLNothing (in module owlapy.model), 56  
 OWLObject (class in owlapy.model), 10  
 OWLObject (class in owlapy.owlobject), 70  
 OWLObjectAllValuesFrom (class in owlapy.model), 20  
 OWLObjectCardinalityRestriction (class in owlapy.model), 22  
 OWLObjectComplementOf (class in owlapy.model), 13  
 OWLObjectComplementOf (class in owlapy.owl\_class\_expression), 65  
 OWLObjectExactCardinality (class in owlapy.model), 22  
 OWLObjectHasSelf (class in owlapy.model), 23  
 OWLObjectHasValue (class in owlapy.model), 23  
 OWLObjectIntersectionOf (class in owlapy.model), 21  
 OWLObjectInverseOf (class in owlapy.owl\_property), 69  
 OWLObjectMaxCardinality (class in owlapy.model), 22  
 OWLObjectMinCardinality (class in owlapy.model), 22  
 OWLObjectOneOf (class in owlapy.model), 24  
 OWLObjectParser (class in owlapy.owlobject), 71  
 OWLObjectProperty (class in owlapy.model), 17  
 OWLObjectProperty (class in owlapy.owl\_property), 68  
 OWLObjectPropertyAssertionAxiom (class in owlapy.model), 43  
 OWLObjectPropertyAxiom (class in owlapy.model), 33  
 OWLObjectPropertyCharacteristicAxiom (class in owlapy.model), 43  
 OWLObjectPropertyDomainAxiom (class in owlapy.model), 46  
 OWLObjectPropertyExpression (class in owlapy.model), 15  
 OWLObjectPropertyExpression (class in owlapy.owl\_property), 68  
 OWLObjectPropertyRangeAxiom (class in owlapy.model), 46  
 OWLObjectRenderer (class in owlapy.owlobject), 71  
 OWLObjectRestriction (class in owlapy.model), 18  
 OWLObjectSomeValuesFrom (class in owlapy.model), 19  
 OWLObjectUnionOf (class in owlapy.model), 21  
 OWLOntology (class in owlapy.model), 47  
 OWLOntologyChange (class in owlapy.model), 49



OWLOntologyID (class in owlapy.model), 24  
 OWLOntologyManager (class in owlapy.model), 49  
 OWLProperty (class in owlapy.model), 16  
 OWLProperty (class in owlapy.owl\_property), 68  
 OWLPropertyAssertionAxiom (class in owlapy.model), 42  
 OWLPropertyAxiom (class in owlapy.model), 33  
 OWLPropertyDomainAxiom (class in owlapy.model), 45  
 OWLPropertyExpression (class in owlapy.model), 16  
 OWLPropertyExpression (class in owlapy.owl\_property), 67  
 OWLPropertyRange (class in owlapy.model), 15  
 OWLPropertyRange (class in owlapy.owl\_class\_expression), 64  
 OWLPropertyRangeAxiom (class in owlapy.model), 46  
 OWLQuantifiedDataRestriction (class in owlapy.model), 28  
 OWLQuantifiedObjectRestriction (class in owlapy.model), 19  
 OWLQuantifiedRestriction (class in owlapy.model), 19  
 OWLRDFVocabulary (class in owlapy.model), 8  
 OWLRDFVocabulary (class in owlapy.vocab), 81  
 OWLReasoner (class in owlapy.model), 50  
 OWLReflexiveObjectPropertyAxiom (class in owlapy.model), 44  
 OWLRestriction (class in owlapy.model), 18  
 OWLSameIndividualAxiom (class in owlapy.model), 36  
 OWLSubAnnotationPropertyOfAxiom (class in owlapy.model), 40  
 OWLSubClassOfAxiom (class in owlapy.model), 38  
 OWLSubDataPropertyOfAxiom (class in owlapy.model), 42  
 OWLSubObjectPropertyOfAxiom (class in owlapy.model), 42  
 OWLSubPropertyAxiom (class in owlapy.model), 42  
 OWLSymmetricObjectPropertyAxiom (class in owlapy.model), 45  
 OWLThing (in module owlapy.model), 56  
 OWLTopDataProperty (in module owlapy.model), 56  
 OWLTopObjectProperty (in module owlapy.model), 56  
 OWLTransitiveObjectPropertyAxiom (class in owlapy.model), 45  
 OWLUnaryPropertyAxiom (class in owlapy.model), 43

## P

parent (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58  
 parent\_var (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58  
 parse\_boolean() (owlapy.model.OwLLiteral method), 27  
 parse\_date() (owlapy.model.OwLLiteral method), 27  
 parse\_datetime() (owlapy.model.OwLLiteral method), 28  
 parse\_double() (owlapy.model.OwLLiteral method), 27  
 parse\_duration() (owlapy.model.OwLLiteral method), 28  
 parse\_expression() (owlapy.owlobject.OwLObjectParser method), 71  
 parse\_expression() (owlapy.parser.DLSyntaxParser method), 75  
 parse\_expression() (owlapy.parser.ManchesterOWLSyntaxParser method), 73  
 parse\_integer() (owlapy.model.OwLLiteral method), 27  
 parse\_string() (owlapy.model.OwLLiteral method), 27  
 PATTERN (owlapy.model.OwLFacet attribute), 9  
 PATTERN (owlapy.vocab.OwLFacet attribute), 82  
 peek() (in module owlapy.owl2sparql.converter), 57  
 prefix (owlapy.namespaces.Namespaces property), 62  
 PREV (owlapy.util.LRUCache attribute), 80  
 process() (owlapy.owl2sparql.converter.Owl2SparqlConverter method), 58  
 properties (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58  
 properties() (owlapy.model.OwLNaryPropertyAxiom method), 37

## R

RDF (in module owlapy.namespaces), 63  
 RDFS (in module owlapy.namespaces), 63  
 RDFS\_LITERAL (owlapy.model.OwLRDFVocabulary attribute), 9  
 RDFS\_LITERAL (owlapy.vocab.OwLRDFVocabulary attribute), 81  
 reminder (owlapy.iri.IRI property), 61  
 reminder (owlapy.model.IRI property), 11  
 reminder (owlapy.model.OwLClass property), 15  
 reminder (owlapy.owl\_class\_expression.OwLClass property), 66  
 remove\_axiom() (owlapy.model.OwLOntologyManager method), 50  
 render() (owlapy.owl2sparql.converter.Owl2SparqlConverter method), 58  
 render() (owlapy.owlobject.OwLObjectRenderer method), 71

`render()` (*owlapy.render.DLSyntaxObjectRenderer method*), 77  
`render()` (*owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer method*), 78  
`Restriction_Literals` (in module *owlapy.model.providers*), 2  
`RESULT` (*owlapy.util.LRUCache attribute*), 80

## S

`same_individuals()` (*owlapy.model.OWLReasoner method*), 52  
`save_ontology()` (*owlapy.model.OWLOntologyManager method*), 50  
`sentinel` (*owlapy.util.LRUCache attribute*), 80  
`set_short_form_provider()` (*owlapy.owlobject.OWLObjectRenderer method*), 71  
`set_short_form_provider()` (*owlapy.render.DLSyntaxObjectRenderer method*), 77  
`set_short_form_provider()` (*owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer method*), 78  
`slots` (*owlapy.parser.DLSyntaxParser attribute*), 74  
`slots` (*owlapy.parser.ManchesterOWLSyntaxParser attribute*), 72  
`sparql` (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58  
`stack_parent()` (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 58  
`stack_variable()` (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 58  
`str` (*owlapy.iri.IRI property*), 61  
`str` (*owlapy.model.IRI property*), 11  
`str` (*owlapy.model.OWLClass property*), 15  
`str` (*owlapy.model.OWLNamedIndividual property*), 24  
`str` (*owlapy.model.OWLObjectProperty property*), 17  
`str` (*owlapy.owl\_class\_expression.OWLClass property*), 66  
`str` (*owlapy.owl\_property.OWLObjectProperty property*), 68  
`STRING` (*owlapy.model.XSDVocabulary attribute*), 9  
`STRING` (*owlapy.vocab.XSDVocabulary attribute*), 81  
`StringOWLDatatype` (in module *owlapy.model*), 56  
`sub_classes()` (*owlapy.model.OWLReasoner method*), 53  
`sub_data_properties()` (*owlapy.model.OWLReasoner method*), 54  
`sub_object_properties()` (*owlapy.model.OWLReasoner method*), 55  
`super_classes()` (*owlapy.model.OWLReasoner method*), 56  
`super_data_properties()` (*owlapy.model.OWLReasoner method*), 54  
`super_object_properties()` (*owlapy.model.OWLReasoner method*), 55  
`symbolic_form` (*owlapy.model.OWLFacet property*), 9  
`symbolic_form` (*owlapy.vocab.OWLFacet property*), 82

## T

`TIME_DATATYPES` (in module *owlapy.model*), 56  
`to_python()` (*owlapy.model.OWLLiteral method*), 28  
`to_string_id()` (*owlapy.model.OWLEntity method*), 10  
`to_string_id()` (*owlapy.owlobject.OWLEntity method*), 71  
`TopLevelCNF` (class in *owlapy.util*), 79  
`TopLevelDNF` (class in *owlapy.util*), 79  
`TopOWLDatatype` (in module *owlapy.model*), 56  
`TOTAL_DIGITS` (*owlapy.model.OWLFacet attribute*), 9  
`TOTAL_DIGITS` (*owlapy.vocab.OWLFacet attribute*), 82  
`triple()` (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 59  
`type_index` (*owlapy.has.HasIndex attribute*), 59  
`type_index` (*owlapy.iri.IRI attribute*), 61  
`type_index` (*owlapy.model.HasIndex attribute*), 12  
`type_index` (*owlapy.model.IRI attribute*), 11  
`type_index` (*owlapy.model.OWLClass attribute*), 15  
`type_index` (*owlapy.model.OWLDataAllValuesFrom attribute*), 29  
`type_index` (*owlapy.model.OWLDataComplementOf attribute*), 29  
`type_index` (*owlapy.model.OWLDataExactCardinality attribute*), 30  
`type_index` (*owlapy.model.OWLDataHasValue attribute*), 30  
`type_index` (*owlapy.model.OWLDataIntersectionOf attribute*), 32  
`type_index` (*owlapy.model.OWLDataMaxCardinality attribute*), 30  
`type_index` (*owlapy.model.OWLDataMinCardinality attribute*), 31  
`type_index` (*owlapy.model.OWLDataOneOf attribute*), 31  
`type_index` (*owlapy.model.OWLDataProperty attribute*), 17  
`type_index` (*owlapy.model.OWLDataSomeValuesFrom attribute*), 31  
`type_index` (*owlapy.model.OWLDatatype attribute*), 26  
`type_index` (*owlapy.model.OWLDatatypeRestriction attribute*), 26  
`type_index` (*owlapy.model.OWLDataUnionOf attribute*), 32  
`type_index` (*owlapy.model.OWLFacetRestriction attribute*), 26  
`type_index` (*owlapy.model.OWLLiteral attribute*), 27

type\_index (*owlapy.model.OWLNamedIndividual* attribute), 24  
 type\_index (*owlapy.model.OWLObjectAllValuesFrom* attribute), 20  
 type\_index (*owlapy.model.OWLObjectComplementOf* attribute), 14  
 type\_index (*owlapy.model.OWLObjectExactCardinality* attribute), 23  
 type\_index (*owlapy.model.OWLObjectHasSelf* attribute), 23  
 type\_index (*owlapy.model.OWLObjectHasValue* attribute), 23  
 type\_index (*owlapy.model.OWLObjectIntersectionOf* attribute), 21  
 type\_index (*owlapy.model.OWLObjectMaxCardinality* attribute), 22  
 type\_index (*owlapy.model.OWLObjectMinCardinality* attribute), 22  
 type\_index (*owlapy.model.OWLObjectOneOf* attribute), 24  
 type\_index (*owlapy.model.OWLObjectProperty* attribute), 17  
 type\_index (*owlapy.model.OWLObjectSomeValuesFrom* attribute), 20  
 type\_index (*owlapy.model.OWLObjectUnionOf* attribute), 21  
 type\_index (*owlapy.model.OWLOntology* attribute), 47  
 type\_index (*owlapy.owl\_class\_expression.OWLClass* attribute), 66  
 type\_index (*owlapy.owl\_class\_expression.OWLObjectComplementOf* attribute), 65  
 type\_index (*owlapy.owl\_property.OWLDataProperty* attribute), 70  
 type\_index (*owlapy.owl\_property.OWLObjectInverseOf* attribute), 69  
 type\_index (*owlapy.owl\_property.OWLObjectProperty* attribute), 68  
 types () (*owlapy.model.OWLReasoner* method), 55

## V

values () (*owlapy.model.OWLDataOneOf* method), 31  
 variable\_entities (*owlapy.owl2sparql.converter.Owl2SparqlConverter* attribute), 58  
 variables (*owlapy.owl2sparql.converter.Owl2SparqlConverter* attribute), 58  
 VariablesMapping (*class in owlapy.owl2sparql.converter*), 57  
 visit\_abbreviated\_iri () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_abbreviated\_iri () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_boolean\_literal () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_boolean\_literal () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_cardinality\_res () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_cardinality\_res () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_class\_expression () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_class\_expression () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_class\_iri () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_class\_iri () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_data\_cardinality\_res () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_data\_cardinality\_res () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_data\_intersection () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_data\_intersection () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_data\_parentheses () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_data\_parentheses () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_data\_primary () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_data\_primary () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_data\_property\_iri () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_data\_property\_iri () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_data\_some\_only\_res () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_data\_some\_only\_res () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_data\_union () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_data\_union () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_data\_value\_res () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_data\_value\_res () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_datatype () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_datatype () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_datatype\_iri () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_datatype\_iri () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_datatype\_restriction () (*owlapy.parser.DLSyntaxParser* method), 75  
 visit\_datatype\_restriction () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 73  
 visit\_date\_literal () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_date\_literal () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_datetime\_literal () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_datetime\_literal () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_decimal\_literal () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_decimal\_literal () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_duration\_literal () (*owlapy.parser.DLSyntaxParser* method), 76  
 visit\_duration\_literal () (*owlapy.parser.ManchesterOWLSyntaxParser* method), 74  
 visit\_facet () (*owlapy.parser.DLSyntaxParser* method), 76

`visit_facet()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_facet_restriction()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_facet_restriction()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_facet_restrictions()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_facet_restrictions()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_float_literal()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_float_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_full_iri()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_full_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_has_self()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_has_self()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_individual_iri()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_individual_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_individual_list()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_individual_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_integer_literal()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_integer_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_intersection()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_intersection()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_iri()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_literal()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_literal_list()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_literal_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_non_negative_integer()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_non_negative_integer()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_object_property()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_object_property()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_object_property_iri()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_object_property_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_parentheses()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_parentheses()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_primary()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_primary()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_quoted_string()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_quoted_string()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_simple_iri()` (*owlapy.parser.DLSyntaxParser method*), 76  
`visit_simple_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 74  
`visit_some_only_res()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_some_only_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_string_literal_language()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_string_literal_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_string_literal_no_language()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_string_literal_no_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_typed_literal()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_typed_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_union()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_union()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73  
`visit_value_res()` (*owlapy.parser.DLSyntaxParser method*), 75  
`visit_value_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 73

## X

XSD (in module *owlapy.namespaces*), 63  
XSDVocabulary (class in *owlapy.model*), 9  
XSDVocabulary (class in *owlapy.vocab*), 81