

Contents:

1	OWLAPY	1
2	owlapy	1
2.1	Subpackages	1
2.2	Submodules	59
2.3	Package Contents	72
	Python Module Index	73
	Index	74

---

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>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>HasIRI</i>	Simple class to access the IRI.
<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>HasOperands</i>	An interface to objects that have a collection of operands.
<i>OWLPropertyRange</i>	OWL Objects that can be the ranges of properties.
<i>OWLDataRange</i>	Represents a 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>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.
<i>OWLClass</i>	An OWL 2 named Class
<i>OWLPropertyExpression</i>	Represents a property or possibly the inverse of a property.
<i>OWLRestriction</i>	Represents an Object Property Restriction or Data Property Restriction in the OWL 2 specification.
<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>OWLDataProperty</i>	Represents a Data Property in the OWL 2 Specification.
<i>OWLObjectProperty</i>	Represents an Object Property in the OWL 2 Specification.
<i>OWLObjectInverseOf</i>	Represents the inverse of a property expression (Object-InverseOf). This can be used to refer to the inverse of
<i>OWLDataRestriction</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>HasFiller</i>	An interface to objects that have a filler.
<i>OWLHasValueRestriction</i>	OWLHasValueRestriction.
<i>OWLQuantifiedRestriction</i>	Represents a quantified restriction.

continues on next page

Table 1 – continued from previous page

<i>OWLQuantifiedObjectRestriction</i>	Represents a quantified object restriction.
<i>OWLObjectSomeValuesFrom</i>	Represents an ObjectSomeValuesFrom class expression in the OWL 2 Specification.
<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.

continues on next page

Table 1 – continued from previous page

<i>OWLDataMinCardinality</i>	Represents DataMinCardinality restrictions in the OWL 2 Specification.
<i>OWLDataOneOf</i>	Represents DataOneOf in the OWL 2 Specification.
<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.

continues on next page

Table 1 – continued from previous page

<i>OWLDisjointUnionAxiom</i>	Represents a DisjointUnion axiom in the OWL 2 Specification.
<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.

continues on next page

Table 1 – continued from previous page

<i>OWLPropertyDomainAxiom</i>	Represents ObjectPropertyDomain axioms in the OWL 2 specification.
<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_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.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.OWLAnnotationObject
```

Bases: *OWLObject*

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

```
__slots__ = ()
```

```
as_iri () → owlapy.model._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 () → owlapy.model.OWLLiteral | None

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

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.IRI (namespace: str | owlapy.namespaces.Namespaces, remainder: str)
    Bases:          owlapy.model._base.OWLAnnotationSubject,          owlapy.model._base.
    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.model.Literals

**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.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.OWLPropertyRange
    Bases: _base.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.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.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.OWLObjectComplementOf(*op: OWLClassExpression*)

Bases: *OWLBooleanClassExpression*, *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.OWLNamedObject

Bases: *\_base.OWLObject*, *\_iri.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.model.OWLEntity
    Bases: OWLNamedObject
    Represents Entities in the OWL 2 Specification.
    __slots__ = ()
    to_string_id() → str
    is_anonymous() → bool

class owlapy.model.OWLClass(iri: _iri.IRI)
    Bases: OWLClassExpression, 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.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.OWLPropertyExpression

Bases: `_base.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.OWLRestriction

Bases: *OWLAnonymousClassExpression*

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

`__slots__ = ()`

**abstract** `get_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.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.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.OWLProperty
    Bases: OWLPropertyExpression, 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.OWLDataProperty (iri: _iri.IRI)
    Bases: OWLDataPropertyExpression, OWLProperty
    Represents a Data Property in the OWL 2 Specification.
    __slots__ = '_iri'

    type_index: Final = 1004

    get_iri () → _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: *\_iri.IRI*)

Bases: *OWLObjectPropertyExpression, OWLProperty*

Represents an Object Property in the OWL 2 Specification.

**property** str: 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** () → *\_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.model.**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.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** () → *OWLObjectPropertyExpression*

**Returns**

Property being restricted.

```

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.OWLHasValueRestriction(value: _T)
    Bases: Generic[_T], OWLRestriction, 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, HasFiller[_T]

    Represents a quantified restriction.

        Parameters
            _T – value type

    __slots__ = ()

class owlapy.model.OWLQuantifiedObjectRestriction(filler: OWLClassExpression)
    Bases: OWLQuantifiedRestriction[OWLClassExpression], OWLObjectRestriction

    Represents a quantified object restriction.

    __slots__ = ()

```

**get\_filler()** → *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: OWLObjectPropertyExpression,
      filler: 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()** → *OWLObjectPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.model.OwlObjectAllValuesFrom (property: OWLObjectPropertyExpression,
      filler: 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()** → *OWLObjectPropertyExpression*

**Returns**

Property being restricted.

```
class owlapy.model.OwlNaryBooleanClassExpression (
      operands: Iterable[OWLClassExpression])
```

Bases: *OWLBooleanClassExpression*, *HasOperands*[*OWLClassExpression*]

OWLNaryBooleanClassExpression.

```

__slots__ = ()

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.OwlObjectUnionOf (operands: Iterable[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[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: OwlObjectPropertyExpression, filler: OwlClassExpression)
```

Bases: *OwlCardinalityRestriction*[*OwlClassExpression*], *OwlQuantifiedObjectRestriction*

Represents Object Property Cardinality Restrictions in the OWL 2 specification.

```
__slots__ = ()
```

**get\_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: OwlObjectPropertyExpression, filler: 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: OwlObjectPropertyExpression, filler: 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: OwlObjectPropertyExpression, filler: 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: *OWLObjectPropertyExpression*)

Bases: *OWLObjectRestriction*

Represents an ObjectHasSelf class expression in the OWL 2 Specification.

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

**type\_index**: **Final** = 3011

**get\_property**() → *OWLObjectPropertyExpression*

**Returns**

Property being restricted.

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

Return self==value.

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

Return hash(self).

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

Return repr(self).

**class** owlapy.model.**OWLIndividual**

Bases: *\_base.OWLObject*

Represents a named or anonymous individual.

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

**class** owlapy.model.**OWLObjectHasValue** (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**() → *OWLObjectPropertyExpression*

**Returns**

Property being restricted.

**as\_some\_values\_from**() → *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: OWLANonymousClassExpression, 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 () → 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: _iri.IRI)
    Bases: OWLIndividual, OWLEntity
    Represents a Named Individual in the OWL 2 Specification.

    property iri

    property str

    __slots__ = '_iri'

    type_index: Final = 1005

    get_iri () → _iri.IRI
        Gets the IRI of this object.

        Returns
            The IRI of this object.

```

```
class owlapy.model.OWLOntologyID (ontology_iri: _iri.IRI | None = None,
                                   version_iri: _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 () → _iri.IRI | None
```

Gets the ontology IRI.

#### Returns

Ontology IRI. If the ontology is anonymous, it will return None.

```
get_version_iri () → _iri.IRI | None
```

Gets the version IRI.

#### Returns

Version IRI or None.

```
get_default_document_iri () → _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.OWLAxiomAxiom (annotations: Iterable[OWLAnnotation] | None = None)
```

Bases: `_base.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: _iri.IRI | _iri.HasIRI)
```

Bases: `OWLEntity`, `OWLDataRange`

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

```

__slots__ = '_iri'

type_index: Final = 4001

get_iri() → _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: 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: \_base.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: \_base.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: OWLDataRange*)  
Bases: *OWLQuantifiedRestriction*[*OWLDataRange*], *OWLDataRestriction*  
Represents a quantified data restriction.

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

**get\_filler()** → *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: OWLDataPropertyExpression*, *filler: OWLDataRange*)  
Bases: *OWLCardinalityRestriction*[*OWLDataRange*], *OWLQuantifiedDataRestriction*,  
*OWLDataRestriction*  
Represents Data Property Cardinality Restrictions in the OWL 2 specification.

```

__slots__ = ()

get_property () → OWLDataPropertyExpression

    Returns
        Property being restricted.

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

class owlapy.model.OWLDataAllValuesFrom (property: OWLDataPropertyExpression,
    filler: 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 () → OWLDataPropertyExpression

        Returns
            Property being restricted.

class owlapy.model.OWLDataComplementOf (data_range: OWLDataRange)
    Bases: OWLDataRange
    Represents DataComplementOf in the OWL 2 Specification.

    type_index: Final = 4002

    get_data_range () → OWLDataRange

        Returns
            The wrapped data range.

    __repr__ ()
        Return repr(self).

    __eq__ (other)
        Return self==value.

    __hash__ ()
        Return hash(self).

```

```

class owlapy.model.OWLDataExactCardinality (cardinality: int,
      property: OWLDataPropertyExpression, filler: 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: 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() → 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() → OWLDataPropertyExpression

    Returns
      Property being restricted.

class owlapy.model.OWLDataMaxCardinality (cardinality: int,
      property: OWLDataPropertyExpression, filler: 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: OWLDataPropertyExpression, filler: 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: OWLDataRange, 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: OWLDataPropertyExpression,
    filler: 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 () → OWLDataPropertyExpression
        Returns
            Property being restricted.

class owlapy.model.OWLNaryDataRange (operands: Iterable[OWLDataRange])
    Bases: OWLDataRange, HasOperands[OWLDataRange]
    OWLNaryDataRange.

```



```

__slots__ = ()

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

    Returns
        The operands.

__repr__ ()
    Return repr(self).

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

class owlapy.model.OWLDataUnionOf (operands: Iterable[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[OWLDataRange])
    Bases: OWLNaryDataRange
    Represents DataIntersectionOf in the OWL 2 Specification.

    __slots__ = '_operands'

    type_index: Final = 4004

class owlapy.model.OWLImportsDeclaration (import_iri: _iri.IRI)
    Bases: _iri.HasIRI
    Represents an import statement in an ontology.

    __slots__ = '_iri'

    get_iri () → _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: 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() → OWLEntity

    __eq__(other)
        Return self==value.

    __hash__()
        Return hash(self).

    __repr__()
        Return repr(self).

class owlapy.model.OWLDatatypeDefinitionAxiom (datatype: OWLDatatype,
    datarange: 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() → OWLDataRange

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

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

__slots__ = ('_class_expression', '_property_expressions')

get_class_expression() → OWLClassExpression

get_property_expressions() → List[OWLPropertyExpression]

operands() → Iterable[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[OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLClassAxiom, OWLNaryAxiom[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[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[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[OWLClass]

class owlapy.model.OWLDisjointClassesAxiom (class_expressions: List[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[OWLObjectPropertyExpression],
    annotations: Iterable[OWLAnnotation] | None = None)

Bases: OWLNaryPropertyAxiom[OWLObjectPropertyExpression], OWLObjectPropertyAx-
iom

Represents EquivalentObjectProperties axioms in the OWL 2 Specification.

__slots__ = ()

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

Bases: OWLNaryPropertyAxiom[OWLObjectPropertyExpression], OWLObjectPropertyAx-
iom

Represents DisjointObjectProperties axioms in the OWL 2 Specification.

__slots__ = ()

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

Bases: OWLNaryPropertyAxiom[OWLObjectPropertyExpression], OWLObjectPropertyAx-
iom

Represents InverseObjectProperties axioms in the OWL 2 Specification.

__slots__ = ('_first', '_second')

get_first_property () → OWLObjectPropertyExpression

get_second_property () → OWLObjectPropertyExpression

__repr__ ()
    Return repr(self).

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

Bases: OWLNaryPropertyAxiom[OWLDataPropertyExpression], OWLDataPropertyAxiom

Represents EquivalentDataProperties axioms in the OWL 2 Specification.

__slots__ = ()

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

Bases: OWLNaryPropertyAxiom[OWLDataPropertyExpression], OWLDataPropertyAxiom

Represents DisjointDataProperties axioms in the OWL 2 Specification.

__slots__ = ()

class owlapy.model.OWLSubClassOfAxiom (sub_class: OWLClassExpression,
    super_class: 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() → OWLClassExpression

get_super_class() → OWLClassExpression

__eq__(other)
    Return self==value.

__hash__()
    Return hash(self).

__repr__()
    Return repr(self).

class owlapy.model.OWLDisjointUnionAxiom(cls_: OWLClass,
    class_expressions: List[OWLClassExpression],
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLClassAxiom
    Represents a DisjointUnion axiom in the OWL 2 Specification.
    __slots__ = ('_cls', '_class_expressions')
    get_owl_class() → OWLClass
    get_class_expressions() → Iterable[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: 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() → 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: _iri.IRI)
    Bases: OWLProperty
    Represents an AnnotationProperty in the OWL 2 specification.

    __slots__ = '_iri'

    get_iri() → _iri.IRI
        Gets the IRI of this object.

        Returns
            The IRI of this object.

class owlapy.model.OWLAnnotation (property: OWLAnnotationProperty,
    value: _base.OWLAnnotationValue)
    Bases: _base.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() → _base.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: _base.OWLAnnotationSubject,
    annotation: OWLAnnotation)
    Bases: OWLAnnotationAxiom
    Represents AnnotationAssertion axioms in the OWL 2 specification.

```



```

__slots__ = ('_subject', '_annotation')

get_subject () → _base.OWLAnnotationSubject
    Gets the subject of this object.

    Returns
        The subject.

get_property () → OWLAnnotationProperty
    Gets the property.

    Returns
        The property.

get_value () → _base.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: _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 () → _iri.IRI

__eq__ (other)
    Return self==value.

__hash__ ()
    Return hash(self).

__repr__ ()
    Return repr(self).

class owlapy.model.OWLAnnotationPropertyRangeAxiom (
    property_: OWLAnnotationProperty, range_: _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 () → _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: OWLObjectPropertyExpression, super_property: OWLObjectPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLSubPropertyAxiom[OWLObjectPropertyExpression], OWLObjectPropertyAxiom
    Represents a SubObjectPropertyOf axiom in the OWL 2 specification.
    __slots__ = ()

class owlapy.model.OWLSubDataPropertyOfAxiom (sub_property: OWLDataPropertyExpression,
    super_property: OWLDataPropertyExpression,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLSubPropertyAxiom[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_: OWLObjectPropertyExpression, object_: OWLIndividual,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLPropertyAssertionAxiom[OWLObjectPropertyExpression, OWLIndividual]
    Represents an ObjectPropertyAssertion axiom in the OWL 2 specification.
    __slots__ = ()

class owlapy.model.OWLNegativeObjectPropertyAssertionAxiom (
    subject: OWLIndividual, property_: OWLObjectPropertyExpression, object_: OWLIndividual,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLPropertyAssertionAxiom[OWLObjectPropertyExpression, OWLIndividual]
    Represents a NegativeObjectPropertyAssertion axiom in the OWL 2 specification.
    __slots__ = ()

```

```

class owlapy.model.OWLDataPropertyAssertionAxiom (subject: OWLIndividual,
    property_: OWLDataPropertyExpression, object_: OWLLiteral,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLPropertyAssertionAxiom[OWLDataPropertyExpression, OWLLiteral]
    Represents an DataPropertyAssertion axiom in the OWL 2 specification.
    __slots__ = ()

class owlapy.model.OWLNegativeDataPropertyAssertionAxiom (subject: OWLIndividual,
    property_: OWLDataPropertyExpression, object_: OWLLiteral,
    annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLPropertyAssertionAxiom[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_: OWLObjectPropertyExpression, annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLUnaryPropertyAxiom[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_: OWLObjectPropertyExpression, annotations: Iterable[OWLAnnotation] | None = None)
    Bases: OWLObjectPropertyCharacteristicAxiom
    Represents FunctionalObjectProperty axioms in the OWL 2 specification.
    __slots__ = ()

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

```

```

__slots__ = ()

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

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

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

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

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

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

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

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

```

```

__slots__ = ()

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

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

class owlapy.model.OWLOntology
    Bases: _base.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[OWLClass]
        Gets the classes in the signature of this object.

        Returns
            Classes in the signature of this object.

    abstract data_properties_in_signature () → Iterable[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[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*: *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*: *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*: *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*: *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*: *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: _iri.IRI) → OWLOntology
        Creates a new (empty) ontology that that has the specified ontology IRI (and no version IRI).

    Parameters
        iri – The IRI of the ontology to be created.

    Returns
        The newly created ontology, or if an ontology with the specified IRI already exists then this
        existing ontology will be returned.

    abstract load_ontology (iri: _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*: *\_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*: *OWLDataProperty*, *direct*: *bool* = *False*)  
→ *Iterable[OWLClassExpression]*

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

**Parameters**

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

**Returns**

Let *N* = *equivalent\_classes(DataSomeValuesFrom(pe rdfs:Literal))*. If *direct* is *True*: then if *N* is not empty then the return value is *N*, else the return value is the result of *super\_classes(DataSomeValuesFrom(pe rdfs:Literal), true)*. If *direct* is *False*: then the result of

super\_classes(DataSomeValuesFrom(pe rdfs:Literal), false) together with N if N is non-empty.  
(Note, rdfs:Literal is the top datatype).

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

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

#### Parameters

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

#### Returns

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

**abstract object\_property\_ranges** (pe: *OWLObjectProperty*, direct: bool = False)  
→ Iterable[*OWLClassExpression*]

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

#### Parameters

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

#### Returns

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

**abstract equivalent\_classes** (ce: *OWLClassExpression*, only\_named: bool = True)  
→ Iterable[*OWLClassExpression*]

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

#### Parameters

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

#### Returns

All class expressions C where the root ontology imports closure entails EquivalentClasses(ce C). If ce is not a class name (i.e. it is an anonymous class expression) and there are no such

classes C then there will be no result. If ce is unsatisfiable with respect to the set of reasoner axioms then owl:Nothing, i.e. the bottom node, will be returned.

**abstract disjoint\_classes** (ce: *OWLClassExpression*, only\_named: bool = True)  
→ Iterable[*OWLClassExpression*]

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

**Parameters**

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

**Returns**

All class expressions D where the set of reasoner axioms entails EquivalentClasses(D Object-ComplementOf(ce)) or StrictSubClassOf(D ObjectComplementOf(ce)).

**abstract different\_individuals** (ind: *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: *OWLObjectPropertyExpression*)  
→ Iterable[*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: *OWLDataProperty*)  
→ Iterable[*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  $\text{EquivalentDataProperties}(dp\ e)$ . 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*: *OWLDataProperty*, *direct*: *bool = True*)  $\rightarrow$  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  $\text{DataPropertyAssertion}(pe\ ind\ l)$ .

**abstract object\_property\_values** (*ind*: *OWLNamedIndividual*, *pe*: *OWLObjectPropertyExpression*, *direct*: *bool = True*)  $\rightarrow$  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  $\text{ObjectPropertyAssertion}(pe\ ind\ j)$ .

**abstract flush** ()  $\rightarrow$  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*: *OWLClassExpression*, *direct*: *bool = False*)  $\rightarrow$  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  $\text{DirectClassAssertion}(ce, j)$ . If **direct** is False, each named individual  $j$  where the set of reasoner axioms entails  $\text{ClassAssertion}(ce, j)$ . If  $ce$  is unsatisfiable with respect to the set of reasoner axioms then nothing returned.

**abstract sub\_classes** (*ce*: *OWLClassExpression*, *direct*: *bool* = *False*, *only\_named*: *bool* = *True*)  
→ Iterable[*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*: *OWLObjectPropertyExpression*)  
→ Iterable[*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*: *OWLDataProperty*) → Iterable[*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*: *OWLDataProperty*, *direct*: *bool* = *False*)  
→ Iterable[*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: *OWLDataProperty*, direct: bool = False)  
→ Iterable[*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: *OWLObjectPropertyExpression*, direct: bool = False)  
→ Iterable[*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: *OWLObjectPropertyExpression*, direct: bool = False)  
→ Iterable[*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[*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()`  $\rightarrow$  *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: OWLClassExpression, direct: bool = False, only_named: bool = True)`  $\rightarrow$  `Iterable[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(x)</i>	Peek the last element of an array.
<i>owl_expression_to_sparql(→ str)</i>	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)

owlapy.owl2sparql.converter.converter

owlapy.owl2sparql.converter.owl_expression_to_sparql (root_variable: str = '?x',
          ce: owlapy.model.OWLClassExpression = None, count: bool = False,
          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) into a SPARQL
query

```

## 2.2 Submodules

**owlapy.io**

Abstract renderer and parser classes.

### Module Contents

#### Classes

<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.

**class** owlapy.io.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: owlapy.model.OWLObject*) → str

Render OWL Object to string.

**Parameters**

o – OWL Object.

**Returns**

String rendition of OWL object.

**class** owlapy.io.OWLObjectParser

Abstract class with a parse method to parse a string to an OWL Object.

**abstract parse\_expression** (*expression\_str: str*) → *owlapy.model.OWLObject*

Parse a string to an OWL Object.

**Parameters**

**expression\_str** (*str*) – Expression string.

**Returns**

The OWL Object which is represented by the string.

**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.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.io.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.io.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

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

### Functions

<code>owl_expression_to_dl(→ str)</code>
<code>owl_expression_to_manchester(→ str)</code>

### Attributes

<code>DLrenderer</code>
<code>ManchesterRenderer</code>

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

```
class owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer (
    short_form_provider: Callable[[owlapy.model.OWLEntity], str] = _simple_short_form_provider,
    no_render_thing=False)
```

Bases: *owlapy.io.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

<i>combine_nary_expressions(...)</i>	Shortens an OWLClassExpression or OWLDataRange by combining all nested nary expressions of the same type.
<i>iter_count</i> (→ int)	Count the number of elements in an iterable.
<i>as_index</i> (→ owlapy.model.HasIndex)	Cast OWL Object to HasIndex.

**class** owlapy.util.**OrderedOWLObject** (*o: \_HasIndex*)

Holder of OWL Objects that can be used for Python sorted.

The Ordering is dependent on the type\_index of the impl. classes recursively followed by all components of the OWL Object.

o

OWL object.

**\_\_slots\_\_** = ('o', '\_chain')

**o:** **\_HasIndex**

**\_\_lt\_\_** (*other*)

Return self<value.

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

Return self==value.

**class** owlapy.util.**NNF**

This class contains functions to transform a Class Expression into Negation Normal Form.

**abstract** **get\_class\_nnf** (*ce: owlapy.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.model.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

<i>OWLRFDFVocabulary</i>	Enumerations for OWL/RDF vocabulary.
<i>XSDVocabulary</i>	Enumerations for XSD vocabulary.
<i>OWLFacet</i>	Enumerations for OWL facets.

```

class owlapy.vocab.OWLRFDFVocabulary(namespace: owlapy.namespaces.Namespaces,
    remainder: str)

```

Bases: `_Vocabulary`, `enum.Enum`

Enumerations for OWL/RDF vocabulary.

```
OWL_THING = ()
```

```
OWL_NOTHING = ()
```

```
OWL_CLASS = ()
```

```
OWL_NAMED_INDIVIDUAL = ()
```

```
OWL_TOP_OBJECT_PROPERTY = ()
```

```
OWL_BOTTOM_OBJECT_PROPERTY = ()
```

```
OWL_TOP_DATA_PROPERTY = ()
```

```
OWL_BOTTOM_DATA_PROPERTY = ()
```

```
RDFS_LITERAL = ()
```

```

class owlapy.vocab.XSDVocabulary(remainder: str)

```

Bases: `_Vocabulary`, `enum.Enum`

Enumerations for XSD vocabulary.

```

DECIMAL: Final = 'decimal'

INTEGER: Final = 'integer'

LONG: Final = 'long'

DOUBLE: Final = 'double'

FLOAT: Final = 'float'

BOOLEAN: Final = 'boolean'

STRING: Final = 'string'

DATE: Final = 'date'

DATE_TIME: Final = 'dateTime'

DATE_TIME_STAMP: Final = 'dateTimeStamp'

DURATION: Final = 'duration'

class owlapy.vocab.OWLFacet (remainder: str, symbolic_form: str,
    operator: Callable[[_X, _X], bool])
    Bases: _Vocabulary, enum.Enum
    Enumerations for OWL facets.

    property symbolic_form

    property operator

    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.io`, [59](#)
- `owlapy.model`, [1](#)
- `owlapy.model.providers`, [2](#)
- `owlapy.namespaces`, [60](#)
- `owlapy.owl2sparql`, [57](#)
- `owlapy.owl2sparql.converter`, [57](#)
- `owlapy.parser`, [61](#)
- `owlapy.render`, [66](#)
- `owlapy.util`, [67](#)
- `owlapy.vocab`, [70](#)

# Index

## Non-alphabetical

`__contains__()` (*owlapy.owl2sparql.converter.VariablesMapping method*), 58  
`__contains__()` (*owlapy.util.LRUCache method*), 70  
`__eq__()` (*owlapy.model.HasIndex method*), 12  
`__eq__()` (*owlapy.model.IRI method*), 11  
`__eq__()` (*owlapy.model.OWLAnnotation method*), 40  
`__eq__()` (*owlapy.model.OWLAnnotationAssertionAxiom method*), 41  
`__eq__()` (*owlapy.model.OWLAnnotationPropertyDomainAxiom method*), 42  
`__eq__()` (*owlapy.model.OWLAnnotationPropertyRangeAxiom method*), 42  
`__eq__()` (*owlapy.model.OWLClassAssertionAxiom method*), 39  
`__eq__()` (*owlapy.model.OWLDataAllValuesFrom method*), 30  
`__eq__()` (*owlapy.model.OWLDataCardinalityRestriction method*), 30  
`__eq__()` (*owlapy.model.OWLDataComplementOf method*), 30  
`__eq__()` (*owlapy.model.OWLDataHasValue method*), 31  
`__eq__()` (*owlapy.model.OWLDataOneOf method*), 32  
`__eq__()` (*owlapy.model.OWLDataPropertyCharacteristicAxiom method*), 45  
`__eq__()` (*owlapy.model.OWLDataSomeValuesFrom method*), 32  
`__eq__()` (*owlapy.model.OWLDatatypeDefinitionAxiom method*), 35  
`__eq__()` (*owlapy.model.OWLDatatypeRestriction method*), 27  
`__eq__()` (*owlapy.model.OWLDeclarationAxiom method*), 34  
`__eq__()` (*owlapy.model.OWLDisjointUnionAxiom method*), 39  
`__eq__()` (*owlapy.model.OWLFacetRestriction method*), 27  
`__eq__()` (*owlapy.model.OWLHasKeyAxiom method*), 35  
`__eq__()` (*owlapy.model.OWLHasValueRestriction method*), 20  
`__eq__()` (*owlapy.model.OWLNamedObject method*), 14  
`__eq__()` (*owlapy.model.OWLNaryBooleanClassExpression method*), 22  
`__eq__()` (*owlapy.model.OWLNaryClassAxiom method*), 36  
`__eq__()` (*owlapy.model.OWLNaryDataRange method*), 33  
`__eq__()` (*owlapy.model.OWLNaryIndividualAxiom method*), 37  
`__eq__()` (*owlapy.model.OWLNaryPropertyAxiom method*), 37  
`__eq__()` (*owlapy.model.OWLObject method*), 10  
`__eq__()` (*owlapy.model.OWLObjectAllValuesFrom method*), 21  
`__eq__()` (*owlapy.model.OWLObjectCardinalityRestriction method*), 23  
`__eq__()` (*owlapy.model.OWLObjectComplementOf method*), 14  
`__eq__()` (*owlapy.model.OWLObjectHasSelf method*), 24  
`__eq__()` (*owlapy.model.OWLObjectInverseOf method*), 19  
`__eq__()` (*owlapy.model.OWLObjectOneOf method*), 25  
`__eq__()` (*owlapy.model.OWLObjectPropertyCharacteristicAxiom method*), 44  
`__eq__()` (*owlapy.model.OWLObjectSomeValuesFrom method*), 21  
`__eq__()` (*owlapy.model.OWLOntologyID method*), 26  
`__eq__()` (*owlapy.model.OWLPropertyAssertionAxiom method*), 43  
`__eq__()` (*owlapy.model.OWLPropertyDomainAxiom method*), 46  
`__eq__()` (*owlapy.model.OWLPropertyRangeAxiom method*), 46  
`__eq__()` (*owlapy.model.OWLSubAnnotationPropertyOfAxiom method*), 41  
`__eq__()` (*owlapy.model.OWLSubClassOfAxiom method*), 39  
`__eq__()` (*owlapy.model.OWLSubPropertyAxiom method*), 42  
`__eq__()` (*owlapy.namespaces.Namespaces method*), 61  
`__eq__()` (*owlapy.util.OrderedOWLObject method*), 68  
`__getitem__()` (*owlapy.owl2sparql.converter.VariablesMapping method*), 58  
`__getitem__()` (*owlapy.util.LRUCache method*), 70  
`__hash__()` (*owlapy.model.IRI method*), 11  
`__hash__()` (*owlapy.model.OWLAnnotation method*), 40  
`__hash__()` (*owlapy.model.OWLAnnotationAssertionAxiom method*), 41  
`__hash__()` (*owlapy.model.OWLAnnotationPropertyDomainAxiom method*), 42  
`__hash__()` (*owlapy.model.OWLAnnotationPropertyRangeAxiom method*), 42  
`__hash__()` (*owlapy.model.OWLClassAssertionAxiom method*), 39  
`__hash__()` (*owlapy.model.OWLDataAllValuesFrom method*), 30  
`__hash__()` (*owlapy.model.OWLDataCardinalityRestriction method*), 30  
`__hash__()` (*owlapy.model.OWLDataComplementOf method*), 30  
`__hash__()` (*owlapy.model.OWLDataHasValue method*), 31  
`__hash__()` (*owlapy.model.OWLDataOneOf method*), 32  
`__hash__()` (*owlapy.model.OWLDataPropertyCharacteristicAxiom method*), 45  
`__hash__()` (*owlapy.model.OWLDataSomeValuesFrom method*), 32  
`__hash__()` (*owlapy.model.OWLDatatypeDefinitionAxiom method*), 35

\_\_hash\_\_ () (owlapy.model.OWLDatatypeRestriction method), 27  
\_\_hash\_\_ () (owlapy.model.OWLDeclarationAxiom method), 34  
\_\_hash\_\_ () (owlapy.model.OWLDisjointUnionAxiom method), 39  
\_\_hash\_\_ () (owlapy.model.OWLFacetRestriction method), 27  
\_\_hash\_\_ () (owlapy.model.OWLHasKeyAxiom method), 35  
\_\_hash\_\_ () (owlapy.model.OWLHasValueRestriction method), 20  
\_\_hash\_\_ () (owlapy.model.OWLNamedObject method), 14  
\_\_hash\_\_ () (owlapy.model.OWLNaryBooleanClassExpression method), 22  
\_\_hash\_\_ () (owlapy.model.OWLNaryClassAxiom method), 36  
\_\_hash\_\_ () (owlapy.model.OWLNaryDataRange method), 33  
\_\_hash\_\_ () (owlapy.model.OWLNaryIndividualAxiom method), 37  
\_\_hash\_\_ () (owlapy.model.OWLNaryPropertyAxiom method), 37  
\_\_hash\_\_ () (owlapy.model.OWLObject method), 10  
\_\_hash\_\_ () (owlapy.model.OWLObjectAllValuesFrom method), 21  
\_\_hash\_\_ () (owlapy.model.OWLObjectCardinalityRestriction method), 23  
\_\_hash\_\_ () (owlapy.model.OWLObjectComplementOf method), 14  
\_\_hash\_\_ () (owlapy.model.OWLObjectHasSelf method), 24  
\_\_hash\_\_ () (owlapy.model.OWLObjectInverseOf method), 19  
\_\_hash\_\_ () (owlapy.model.OWLObjectOneOf method), 25  
\_\_hash\_\_ () (owlapy.model.OWLObjectPropertyCharacteristicAxiom method), 44  
\_\_hash\_\_ () (owlapy.model.OWLObjectSomeValuesFrom method), 21  
\_\_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), 39  
\_\_hash\_\_ () (owlapy.model.OWLSubPropertyAxiom method), 42  
\_\_hash\_\_ () (owlapy.namespaces.Namespaces method), 61  
\_\_lt\_\_ () (owlapy.model.OWLNamedObject method), 14  
\_\_lt\_\_ () (owlapy.util.OrderedOWLObject method), 68  
\_\_repr\_\_ () (owlapy.model.IRI method), 11  
\_\_repr\_\_ () (owlapy.model.OWLAnnotation method), 40  
\_\_repr\_\_ () (owlapy.model.OWLAnnotationAssertionAxiom method), 41  
\_\_repr\_\_ () (owlapy.model.OWLAnnotationPropertyDomainAxiom method), 42  
\_\_repr\_\_ () (owlapy.model.OWLAnnotationPropertyRangeAxiom method), 42  
\_\_repr\_\_ () (owlapy.model.OWLClassAssertionAxiom method), 39  
\_\_repr\_\_ () (owlapy.model.OWLDataAllValuesFrom method), 30  
\_\_repr\_\_ () (owlapy.model.OWLDataCardinalityRestriction method), 30  
\_\_repr\_\_ () (owlapy.model.OWLDataComplementOf method), 30  
\_\_repr\_\_ () (owlapy.model.OWLDataHasValue method), 31  
\_\_repr\_\_ () (owlapy.model.OWLDataOneOf method), 32  
\_\_repr\_\_ () (owlapy.model.OWLDataPropertyCharacteristicAxiom method), 45  
\_\_repr\_\_ () (owlapy.model.OWLDataSomeValuesFrom method), 32  
\_\_repr\_\_ () (owlapy.model.OWLDatatypeDefinitionAxiom method), 35  
\_\_repr\_\_ () (owlapy.model.OWLDatatypeRestriction method), 27  
\_\_repr\_\_ () (owlapy.model.OWLDeclarationAxiom method), 34  
\_\_repr\_\_ () (owlapy.model.OWLDisjointUnionAxiom method), 39  
\_\_repr\_\_ () (owlapy.model.OWLFacetRestriction method), 27  
\_\_repr\_\_ () (owlapy.model.OWLHasKeyAxiom method), 35  
\_\_repr\_\_ () (owlapy.model.OWLInverseObjectPropertiesAxiom method), 38  
\_\_repr\_\_ () (owlapy.model.OWLNamedObject method), 15  
\_\_repr\_\_ () (owlapy.model.OWLNaryBooleanClassExpression method), 22  
\_\_repr\_\_ () (owlapy.model.OWLNaryClassAxiom method), 36  
\_\_repr\_\_ () (owlapy.model.OWLNaryDataRange method), 33  
\_\_repr\_\_ () (owlapy.model.OWLNaryIndividualAxiom method), 37  
\_\_repr\_\_ () (owlapy.model.OWLNaryPropertyAxiom method), 37  
\_\_repr\_\_ () (owlapy.model.OWLObject method), 10  
\_\_repr\_\_ () (owlapy.model.OWLObjectAllValuesFrom method), 21  
\_\_repr\_\_ () (owlapy.model.OWLObjectCardinalityRestriction method), 23  
\_\_repr\_\_ () (owlapy.model.OWLObjectComplementOf method), 14  
\_\_repr\_\_ () (owlapy.model.OWLObjectHasSelf method), 24  
\_\_repr\_\_ () (owlapy.model.OWLObjectHasValue method), 24  
\_\_repr\_\_ () (owlapy.model.OWLObjectInverseOf method), 19  
\_\_repr\_\_ () (owlapy.model.OWLObjectOneOf method), 25  
\_\_repr\_\_ () (owlapy.model.OWLObjectPropertyCharacteristicAxiom method), 44  
\_\_repr\_\_ () (owlapy.model.OWLObjectSomeValuesFrom method), 21  
\_\_repr\_\_ () (owlapy.model.OWLOntologyID method), 26  
\_\_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), 39  
\_\_repr\_\_() (owlapy.model.OWLSubPropertyAxiom method), 42  
\_\_repr\_\_() (owlapy.namespaces.Namespaces method), 60  
\_\_setitem\_\_() (owlapy.util.LRUCache method), 70  
\_\_slots\_\_ (owlapy.model.AddImport attribute), 49  
\_\_slots\_\_ (owlapy.model.HasCardinality attribute), 22  
\_\_slots\_\_ (owlapy.model.HasFiller attribute), 20  
\_\_slots\_\_ (owlapy.model.HasIRI attribute), 11  
\_\_slots\_\_ (owlapy.model.HasOperands attribute), 12  
\_\_slots\_\_ (owlapy.model.IRI attribute), 11  
\_\_slots\_\_ (owlapy.model.OWLAnnotation attribute), 40  
\_\_slots\_\_ (owlapy.model.OWLAnnotationAssertionAxiom attribute), 40  
\_\_slots\_\_ (owlapy.model.OWLAnnotationAxiom attribute), 40  
\_\_slots\_\_ (owlapy.model.OWLAnnotationObject attribute), 10  
\_\_slots\_\_ (owlapy.model.OWLAnnotationProperty attribute), 40  
\_\_slots\_\_ (owlapy.model.OWLAnnotationPropertyDomainAxiom attribute), 41  
\_\_slots\_\_ (owlapy.model.OWLAnnotationPropertyRangeAxiom attribute), 42  
\_\_slots\_\_ (owlapy.model.OWLAnnotationSubject attribute), 10  
\_\_slots\_\_ (owlapy.model.OWLAnnotationValue attribute), 10  
\_\_slots\_\_ (owlapy.model.OWLAsymmetricObjectPropertyAxiom attribute), 44  
\_\_slots\_\_ (owlapy.model.OWLClass attribute), 26  
\_\_slots\_\_ (owlapy.model.OWLBooleanClassExpression attribute), 14  
\_\_slots\_\_ (owlapy.model.OWLCardinalityRestriction attribute), 22  
\_\_slots\_\_ (owlapy.model.OWLClass attribute), 15  
\_\_slots\_\_ (owlapy.model.OWLClassAssertionAxiom attribute), 39  
\_\_slots\_\_ (owlapy.model.OWLClassAxiom attribute), 34  
\_\_slots\_\_ (owlapy.model.OWLClassExpression attribute), 13  
\_\_slots\_\_ (owlapy.model.OWLData.AllValuesFrom attribute), 30  
\_\_slots\_\_ (owlapy.model.OWLDataCardinalityRestriction attribute), 29  
\_\_slots\_\_ (owlapy.model.OWLDataExactCardinality attribute), 31  
\_\_slots\_\_ (owlapy.model.OWLDataHasValue attribute), 31  
\_\_slots\_\_ (owlapy.model.OWLDataIntersectionOf attribute), 33  
\_\_slots\_\_ (owlapy.model.OWLDataMaxCardinality attribute), 31  
\_\_slots\_\_ (owlapy.model.OWLDataMinCardinality attribute), 31  
\_\_slots\_\_ (owlapy.model.OWLDataProperty attribute), 17  
\_\_slots\_\_ (owlapy.model.OWLDataPropertyAssertionAxiom attribute), 44  
\_\_slots\_\_ (owlapy.model.OWLDataPropertyAxiom attribute), 34  
\_\_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), 19  
\_\_slots\_\_ (owlapy.model.OWLDataSomeValuesFrom attribute), 32  
\_\_slots\_\_ (owlapy.model.OWLDatatype attribute), 26  
\_\_slots\_\_ (owlapy.model.OWLDatatypeDefinitionAxiom attribute), 34  
\_\_slots\_\_ (owlapy.model.OWLDatatypeRestriction attribute), 27  
\_\_slots\_\_ (owlapy.model.OWLDataUnionOf attribute), 33  
\_\_slots\_\_ (owlapy.model.OWLDeclarationAxiom attribute), 34  
\_\_slots\_\_ (owlapy.model.OWLDifferentIndividualsAxiom attribute), 37  
\_\_slots\_\_ (owlapy.model.OWLDisjointClassesAxiom attribute), 36  
\_\_slots\_\_ (owlapy.model.OWLDisjointDataPropertiesAxiom attribute), 38  
\_\_slots\_\_ (owlapy.model.OWLDisjointObjectPropertiesAxiom attribute), 38  
\_\_slots\_\_ (owlapy.model.OWLDisjointUnionAxiom attribute), 39  
\_\_slots\_\_ (owlapy.model.OWLEntity attribute), 15  
\_\_slots\_\_ (owlapy.model.OWLEquivalentClassesAxiom attribute), 36  
\_\_slots\_\_ (owlapy.model.OWLEquivalentDataPropertiesAxiom attribute), 38  
\_\_slots\_\_ (owlapy.model.OWLEquivalentObjectPropertiesAxiom attribute), 38  
\_\_slots\_\_ (owlapy.model.OWLFacetRestriction attribute), 27  
\_\_slots\_\_ (owlapy.model.OWLFunctionalDataPropertyAxiom attribute), 46  
\_\_slots\_\_ (owlapy.model.OWLFunctionalObjectPropertyAxiom attribute), 44  
\_\_slots\_\_ (owlapy.model.OWLHasKeyAxiom attribute), 35  
\_\_slots\_\_ (owlapy.model.OWLHasValueRestriction attribute), 20  
\_\_slots\_\_ (owlapy.model.OWLImportsDeclaration attribute), 33  
\_\_slots\_\_ (owlapy.model.OWLIndividual attribute), 24  
\_\_slots\_\_ (owlapy.model.OWLIndividualAxiom attribute), 34

- \_\_slots\_\_ (owlapy.model.OWLInverseFunctionalObjectPropertyAxiom attribute), 45
- \_\_slots\_\_ (owlapy.model.OWLInverseObjectPropertiesAxiom attribute), 38
- \_\_slots\_\_ (owlapy.model.OWLIrreflexiveObjectPropertyAxiom attribute), 45
- \_\_slots\_\_ (owlapy.model.OWLLiteral attribute), 27
- \_\_slots\_\_ (owlapy.model.OWLLogicalAxiom attribute), 33
- \_\_slots\_\_ (owlapy.model.OWLNamedIndividual attribute), 25
- \_\_slots\_\_ (owlapy.model.OWLNamedObject attribute), 14
- \_\_slots\_\_ (owlapy.model.OWLNaryAxiom attribute), 35
- \_\_slots\_\_ (owlapy.model.OWLNaryBooleanClassExpression attribute), 21
- \_\_slots\_\_ (owlapy.model.OWLNaryClassAxiom attribute), 36
- \_\_slots\_\_ (owlapy.model.OWLNaryDataRange attribute), 32
- \_\_slots\_\_ (owlapy.model.OWLNaryIndividualAxiom attribute), 36
- \_\_slots\_\_ (owlapy.model.OWLNaryPropertyAxiom attribute), 37
- \_\_slots\_\_ (owlapy.model.OWLNegativeDataPropertyAssertionAxiom attribute), 44
- \_\_slots\_\_ (owlapy.model.OWLNegativeObjectPropertyAssertionAxiom attribute), 43
- \_\_slots\_\_ (owlapy.model.OWLObject attribute), 10
- \_\_slots\_\_ (owlapy.model.OWLObjectAllValuesFrom attribute), 21
- \_\_slots\_\_ (owlapy.model.OWLObjectCardinalityRestriction attribute), 23
- \_\_slots\_\_ (owlapy.model.OWLObjectComplementOf attribute), 14
- \_\_slots\_\_ (owlapy.model.OWLObjectExactCardinality attribute), 23
- \_\_slots\_\_ (owlapy.model.OWLObjectHasSelf attribute), 24
- \_\_slots\_\_ (owlapy.model.OWLObjectHasValue attribute), 24
- \_\_slots\_\_ (owlapy.model.OWLObjectIntersectionOf attribute), 22
- \_\_slots\_\_ (owlapy.model.OWLObjectInverseOf attribute), 18
- \_\_slots\_\_ (owlapy.model.OWLObjectMaxCardinality attribute), 23
- \_\_slots\_\_ (owlapy.model.OWLObjectMinCardinality attribute), 23
- \_\_slots\_\_ (owlapy.model.OWLObjectOneOf attribute), 25
- \_\_slots\_\_ (owlapy.model.OWLObjectProperty attribute), 18
- \_\_slots\_\_ (owlapy.model.OWLObjectPropertyAssertionAxiom attribute), 43
- \_\_slots\_\_ (owlapy.model.OWLObjectPropertyAxiom attribute), 34
- \_\_slots\_\_ (owlapy.model.OWLObjectPropertyCharacteristicAxiom attribute), 44
- \_\_slots\_\_ (owlapy.model.OWLObjectPropertyDomainAxiom attribute), 46
- \_\_slots\_\_ (owlapy.model.OWLObjectPropertyExpression attribute), 17
- \_\_slots\_\_ (owlapy.model.OWLObjectPropertyRangeAxiom attribute), 47
- \_\_slots\_\_ (owlapy.model.OWLObjectRestriction attribute), 19
- \_\_slots\_\_ (owlapy.model.OWLObjectSomeValuesFrom attribute), 21
- \_\_slots\_\_ (owlapy.model.OWLObjectUnionOf attribute), 22
- \_\_slots\_\_ (owlapy.model.OWLOntology attribute), 47
- \_\_slots\_\_ (owlapy.model.OWLOntologyChange attribute), 49
- \_\_slots\_\_ (owlapy.model.OWLOntologyID attribute), 26
- \_\_slots\_\_ (owlapy.model.OWLProperty attribute), 17
- \_\_slots\_\_ (owlapy.model.OWLPropertyAssertionAxiom attribute), 43
- \_\_slots\_\_ (owlapy.model.OWLPropertyAxiom attribute), 34
- \_\_slots\_\_ (owlapy.model.OWLPropertyDomainAxiom attribute), 46
- \_\_slots\_\_ (owlapy.model.OWLPropertyExpression attribute), 16
- \_\_slots\_\_ (owlapy.model.OWLPropertyRangeAxiom attribute), 46
- \_\_slots\_\_ (owlapy.model.OWLQuantifiedDataRestriction attribute), 29
- \_\_slots\_\_ (owlapy.model.OWLQuantifiedObjectRestriction attribute), 20
- \_\_slots\_\_ (owlapy.model.OWLQuantifiedRestriction attribute), 20
- \_\_slots\_\_ (owlapy.model.OWLReasoner attribute), 50
- \_\_slots\_\_ (owlapy.model.OWLReflexiveObjectPropertyAxiom attribute), 45
- \_\_slots\_\_ (owlapy.model.OWLRestriction attribute), 16
- \_\_slots\_\_ (owlapy.model.OWLSameIndividualAxiom attribute), 37
- \_\_slots\_\_ (owlapy.model.OWLSubAnnotationPropertyOfAxiom attribute), 41
- \_\_slots\_\_ (owlapy.model.OWLSubClassOfAxiom attribute), 38
- \_\_slots\_\_ (owlapy.model.OWLSubDataPropertyOfAxiom attribute), 43
- \_\_slots\_\_ (owlapy.model.OWLSubObjectPropertyOfAxiom attribute), 43
- \_\_slots\_\_ (owlapy.model.OWLSubPropertyAxiom attribute), 42
- \_\_slots\_\_ (owlapy.model.OWLSymmetricObjectPropertyAxiom attribute), 45
- \_\_slots\_\_ (owlapy.model.OWLTransitiveObjectPropertyAxiom attribute), 45
- \_\_slots\_\_ (owlapy.model.OWLUnaryPropertyAxiom attribute), 44
- \_\_slots\_\_ (owlapy.namespaces.Namespaces attribute), 60
- \_\_slots\_\_ (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58
- \_\_slots\_\_ (owlapy.owl2sparql.converter.VariablesMapping attribute), 57
- \_\_slots\_\_ (owlapy.render.DLSyntaxObjectRenderer attribute), 66
- \_\_slots\_\_ (owlapy.render.ManchesterOWLSyntaxOWLObjectRenderer attribute), 67
- \_\_slots\_\_ (owlapy.util.OrderedOWLObject attribute), 68
- \_\_version\_\_ (in module owlapy), 72

## A

`add_axiom()` (*owlapy.model.OWLOntologyManager method*), 50  
`AddImport` (*class in owlapy.model*), 49  
`annotations()` (*owlapy.model.OWLAXiom method*), 26  
`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_index()` (*in module owlapy.util*), 69  
`as_intersection_of_min_max()` (*owlapy.model.OWLDataExactCardinality method*), 31  
`as_intersection_of_min_max()` (*owlapy.model.OWLObjectExactCardinality method*), 23  
`as_iri()` (*owlapy.model.IRI method*), 12  
`as_iri()` (*owlapy.model.OWLAnnotationObject method*), 10  
`as_literal()` (*owlapy.model.OWLAnnotationValue method*), 11  
`as_literal()` (*owlapy.model.OWLLiteral method*), 29  
`as_object_union_of()` (*owlapy.model.OWLObjectOneOf method*), 25  
`as_pairwise_axioms()` (*owlapy.model.OWLNaryAxiom method*), 35  
`as_pairwise_axioms()` (*owlapy.model.OWLNaryClassAxiom method*), 36  
`as_pairwise_axioms()` (*owlapy.model.OWLNaryIndividualAxiom method*), 37  
`as_pairwise_axioms()` (*owlapy.model.OWLNaryPropertyAxiom method*), 37  
`as_query()` (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 59  
`as_some_values_from()` (*owlapy.model.OWLDataHasValue method*), 31  
`as_some_values_from()` (*owlapy.model.OWLObjectHasValue method*), 24  
`as_str()` (*owlapy.model.IRI method*), 12

## B

`BOOLEAN` (*owlapy.model.XSDVocabulary attribute*), 9  
`BOOLEAN` (*owlapy.vocab.XSDVocabulary attribute*), 71  
`BooleanOWLDatatype` (*in module owlapy.model*), 56

## C

`cache_clear()` (*owlapy.util.LRUCache method*), 70  
`cache_info()` (*owlapy.util.LRUCache method*), 70  
`ce` (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58  
`class_expressions()` (*owlapy.model.OWLNaryClassAxiom method*), 36  
`classes_in_signature()` (*owlapy.model.OWLOntology method*), 47  
`cnt` (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58  
`combine_nary_expressions()` (*in module owlapy.util*), 69  
`contains_named_equivalent_class()` (*owlapy.model.OWLEquivalentClassesAxiom method*), 36  
`contains_owl_nothing()` (*owlapy.model.OWLEquivalentClassesAxiom method*), 36  
`contains_owl_thing()` (*owlapy.model.OWLEquivalentClassesAxiom method*), 36  
`convert()` (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 58  
`converter` (*in module owlapy.owl2sparql.converter*), 59  
`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*), 53  
`DATE` (*owlapy.model.XSDVocabulary attribute*), 9  
`DATE` (*owlapy.vocab.XSDVocabulary attribute*), 71  
`DATE_TIME` (*owlapy.model.XSDVocabulary attribute*), 9  
`DATE_TIME` (*owlapy.vocab.XSDVocabulary attribute*), 71  
`DATE_TIME_STAMP` (*owlapy.model.XSDVocabulary attribute*), 9  
`DATE_TIME_STAMP` (*owlapy.vocab.XSDVocabulary attribute*), 71  
`DateOWLDatatype` (*in module owlapy.model*), 56  
`DateTimeOWLDatatype` (*in module owlapy.model*), 56  
`DECIMAL` (*owlapy.model.XSDVocabulary attribute*), 9  
`DECIMAL` (*owlapy.vocab.XSDVocabulary attribute*), 70  
`different_individuals()` (*owlapy.model.OWLReasoner method*), 52  
`disjoint_classes()` (*owlapy.model.OWLReasoner method*), 52  
`disjoint_data_properties()` (*owlapy.model.OWLReasoner method*), 54



`disjoint_object_properties()` (*owlapy.model.OWLReasoner method*), 54  
`DL_GRAMMAR` (*in module owlapy.parser*), 63  
`dl_to_owl_expression()` (*in module owlapy.parser*), 66  
`DLparser` (*in module owlapy.parser*), 65  
`DLrenderer` (*in module owlapy.render*), 67  
`DLSyntaxObjectRenderer` (*class in owlapy.render*), 66  
`DLSyntaxParser` (*class in owlapy.parser*), 64  
`DOUBLE` (*owlapy.model.XSDVocabulary attribute*), 9  
`DOUBLE` (*owlapy.vocab.XSDVocabulary attribute*), 71  
`DoubleOWLDatatype` (*in module owlapy.model*), 56  
`DURATION` (*owlapy.model.XSDVocabulary attribute*), 9  
`DURATION` (*owlapy.vocab.XSDVocabulary attribute*), 71  
`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*), 71  
`flush()` (*owlapy.model.OWLReasoner method*), 53  
`FRACTION_DIGITS` (*owlapy.model.OWLFacet attribute*), 9  
`FRACTION_DIGITS` (*owlapy.vocab.OWLFacet attribute*), 71  
`from_str()` (*owlapy.model.OWLFacet static method*), 10  
`from_str()` (*owlapy.vocab.OWLFacet static method*), 71

## G

`general_class_axioms()` (*owlapy.model.OWLOntology method*), 48  
`generic_visit()` (*owlapy.parser.DLSyntaxParser method*), 65  
`generic_visit()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`get_cardinality()` (*owlapy.model.HasCardinality method*), 22  
`get_cardinality()` (*owlapy.model.OWLCardinalityRestriction method*), 22  
`get_class_expression()` (*owlapy.model.OWLClassAssertionAxiom method*), 39  
`get_class_expression()` (*owlapy.model.OWLHasKeyAxiom method*), 35  
`get_class_expressions()` (*owlapy.model.OWLDisjointUnionAxiom method*), 39  
`get_class_nnf()` (*owlapy.util.NNF method*), 68  
`get_data_range()` (*owlapy.model.OWLDataComplementOf method*), 30  
`get_datarange()` (*owlapy.model.OWLDatatypeDefinitionAxiom method*), 35  
`get_datatype()` (*owlapy.model.OWLDatatypeDefinitionAxiom method*), 35  
`get_datatype()` (*owlapy.model.OWLDatatypeRestriction method*), 27  
`get_datatype()` (*owlapy.model.OWLLiteral method*), 29  
`get_default_document_iri()` (*owlapy.model.OWLOntologyID method*), 26  
`get_domain()` (*owlapy.model.OWLAnnotationPropertyDomainAxiom method*), 42  
`get_domain()` (*owlapy.model.OWLPropertyDomainAxiom method*), 46  
`get_entity()` (*owlapy.model.OWLDeclarationAxiom method*), 34  
`get_facet()` (*owlapy.model.OWLFacetRestriction method*), 27  
`get_facet_restrictions()` (*owlapy.model.OWLDatatypeRestriction method*), 27  
`get_facet_value()` (*owlapy.model.OWLFacetRestriction method*), 27  
`get_filler()` (*owlapy.model.HasFiller method*), 20  
`get_filler()` (*owlapy.model.OWLCardinalityRestriction method*), 22  
`get_filler()` (*owlapy.model.OWLHasValueRestriction method*), 20  
`get_filler()` (*owlapy.model.OWLQuantifiedDataRestriction method*), 29  
`get_filler()` (*owlapy.model.OWLQuantifiedObjectRestriction method*), 20  
`get_first_property()` (*owlapy.model.OWLInverseObjectPropertiesAxiom method*), 38  
`get_import_declaration()` (*owlapy.model.AddImport method*), 49  
`get_individual()` (*owlapy.model.OWLClassAssertionAxiom method*), 39  
`get_inverse()` (*owlapy.model.OWLObjectInverseOf method*), 18  
`get_inverse_property()` (*owlapy.model.OWLObjectInverseOf method*), 19  
`get_inverse_property()` (*owlapy.model.OWLObjectProperty method*), 18  
`get_inverse_property()` (*owlapy.model.OWLObjectPropertyExpression method*), 17  
`get_iri()` (*owlapy.model.HasIRI method*), 11  
`get_iri()` (*owlapy.model.OWLAnnotationProperty method*), 40  
`get_iri()` (*owlapy.model.OWLClass method*), 15

`get_iri()` (*owlapy.model.OWLDataProperty method*), 17  
`get_iri()` (*owlapy.model.OWLDatatype method*), 27  
`get_iri()` (*owlapy.model.OWLImportsDeclaration method*), 33  
`get_iri()` (*owlapy.model.OWLNamedIndividual method*), 25  
`get_iri()` (*owlapy.model.OWLObjectProperty method*), 18  
`get_literal()` (*owlapy.model.OWLLiteral method*), 28  
`get_named_property()` (*owlapy.model.OWLObjectInverseOf method*), 19  
`get_named_property()` (*owlapy.model.OWLObjectProperty method*), 18  
`get_named_property()` (*owlapy.model.OWLObjectPropertyExpression method*), 17  
`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_object()` (*owlapy.model.OWLPropertyAssertionAxiom method*), 43  
`get_object_complement_of()` (*owlapy.model.OWLAnonymousClassExpression method*), 13  
`get_object_complement_of()` (*owlapy.model.OWLClass method*), 15  
`get_object_complement_of()` (*owlapy.model.OWLClassExpression method*), 13  
`get_ontology()` (*owlapy.model.OWLOntologyChange method*), 49  
`get_ontology_id()` (*owlapy.model.OWLOntology method*), 49  
`get_ontology_iri()` (*owlapy.model.OWLOntologyID method*), 26  
`get_operand()` (*owlapy.model.OWLObjectComplementOf method*), 14  
`get_owl_class()` (*owlapy.model.OWLDisjointUnionAxiom method*), 39  
`get_owl_disjoint_classes_axiom()` (*owlapy.model.OWLDisjointUnionAxiom method*), 39  
`get_owl_equivalent_classes_axiom()` (*owlapy.model.OWLDisjointUnionAxiom method*), 39  
`get_owl_ontology_manager()` (*owlapy.model.OWLOntology method*), 48  
`get_property()` (*owlapy.model.OWLAnnotation method*), 40  
`get_property()` (*owlapy.model.OWLAnnotationAssertionAxiom method*), 41  
`get_property()` (*owlapy.model.OWLAnnotationPropertyDomainAxiom method*), 41  
`get_property()` (*owlapy.model.OWLAnnotationPropertyRangeAxiom method*), 42  
`get_property()` (*owlapy.model.OWLDataAllValuesFrom method*), 30  
`get_property()` (*owlapy.model.OWLDataCardinalityRestriction method*), 30  
`get_property()` (*owlapy.model.OWLDataHasValue method*), 31  
`get_property()` (*owlapy.model.OWLDataSomeValuesFrom method*), 32  
`get_property()` (*owlapy.model.OWLObjectAllValuesFrom method*), 21  
`get_property()` (*owlapy.model.OWLObjectCardinalityRestriction method*), 23  
`get_property()` (*owlapy.model.OWLObjectHasSelf method*), 24  
`get_property()` (*owlapy.model.OWLObjectHasValue method*), 24  
`get_property()` (*owlapy.model.OWLObjectRestriction method*), 19  
`get_property()` (*owlapy.model.OWLObjectSomeValuesFrom method*), 21  
`get_property()` (*owlapy.model.OWLPropertyAssertionAxiom method*), 43  
`get_property()` (*owlapy.model.OWLRestriction method*), 16  
`get_property()` (*owlapy.model.OWLUnaryPropertyAxiom method*), 44  
`get_property_expressions()` (*owlapy.model.OWLHasKeyAxiom method*), 35  
`get_range()` (*owlapy.model.OWLAnnotationPropertyRangeAxiom method*), 42  
`get_range()` (*owlapy.model.OWLPropertyRangeAxiom method*), 46  
`get_remainder()` (*owlapy.model.IRI method*), 12  
`get_root_ontology()` (*owlapy.model.OWLReasoner method*), 56  
`get_second_property()` (*owlapy.model.OWLInverseObjectPropertiesAxiom method*), 38  
`get_short_form()` (*owlapy.model.IRI method*), 12  
`get_sub_class()` (*owlapy.model.OWLSubClassOfAxiom method*), 39  
`get_sub_property()` (*owlapy.model.OWLSubAnnotationPropertyOfAxiom method*), 41  
`get_sub_property()` (*owlapy.model.OWLSubPropertyAxiom method*), 42  
`get_subject()` (*owlapy.model.OWLAnnotationAssertionAxiom method*), 41  
`get_subject()` (*owlapy.model.OWLPropertyAssertionAxiom method*), 43  
`get_super_class()` (*owlapy.model.OWLSubClassOfAxiom method*), 39  
`get_super_property()` (*owlapy.model.OWLSubAnnotationPropertyOfAxiom method*), 41  
`get_super_property()` (*owlapy.model.OWLSubPropertyAxiom method*), 42  
`get_top_level_cnf()` (*owlapy.util.TopLevelCNF method*), 69  
`get_top_level_dnf()` (*owlapy.util.TopLevelDNF method*), 69  
`get_value()` (*owlapy.model.OWLAnnotation method*), 40  
`get_value()` (*owlapy.model.OWLAnnotationAssertionAxiom method*), 41  
`get_variable()` (*owlapy.owl2sparql.converter.VariablesMapping method*), 58  
`get_version_iri()` (*owlapy.model.OWLOntologyID method*), 26  
`grouping_vars` (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58

## H

`HasCardinality` (*class in owlapy.model*), 22



HasFiller (class in owlapy.model), 19  
 HasIndex (class in owlapy.model), 12  
 HasIRI (class in owlapy.model), 11  
 HasOperands (class in owlapy.model), 12  
 having\_conditions (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58

## I

individuals() (owlapy.model.OwLNaryIndividualAxiom method), 37  
 individuals() (owlapy.model.OwLObjectOneOf method), 25  
 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), 71  
 IntegerOWLDatatype (in module owlapy.model), 56  
 IRI (class in owlapy.model), 11  
 iri (owlapy.model.OwLNamedIndividual property), 25  
 is\_annotated() (owlapy.model.OwLAxiom method), 26  
 is\_annotation\_axiom() (owlapy.model.OwLAnnotationAxiom method), 40  
 is\_annotation\_axiom() (owlapy.model.OwLAxiom method), 26  
 is\_anonymous() (owlapy.model.OwLEntity method), 15  
 is\_anonymous() (owlapy.model.OwLObject method), 10  
 is\_anonymous() (owlapy.model.OwLOntology method), 49  
 is\_anonymous() (owlapy.model.OwLOntologyID method), 26  
 is\_boolean() (owlapy.model.OwLLiteral method), 28  
 is\_data\_property\_expression() (owlapy.model.OwLDataPropertyExpression method), 17  
 is\_data\_property\_expression() (owlapy.model.OwLPropertyExpression method), 16  
 is\_data\_restriction() (owlapy.model.OwLDataRestriction method), 19  
 is\_data\_restriction() (owlapy.model.OwLRestriction method), 16  
 is\_date() (owlapy.model.OwLLiteral method), 28  
 is\_datetime() (owlapy.model.OwLLiteral method), 28  
 is\_double() (owlapy.model.OwLLiteral method), 28  
 is\_duration() (owlapy.model.OwLLiteral method), 29  
 is\_integer() (owlapy.model.OwLLiteral method), 28  
 is\_isolated() (owlapy.model.OwLReasoner method), 56  
 is\_literal() (owlapy.model.OwLAnnotationValue method), 10  
 is\_literal() (owlapy.model.OwLLiteral method), 29  
 is\_logical\_axiom() (owlapy.model.OwLAxiom method), 26  
 is\_logical\_axiom() (owlapy.model.OwLLogicalAxiom method), 33  
 is\_nothing() (owlapy.model.IRI method), 11  
 is\_object\_property\_expression() (owlapy.model.OwLObjectPropertyExpression method), 17  
 is\_object\_property\_expression() (owlapy.model.OwLPropertyExpression method), 16  
 is\_object\_restriction() (owlapy.model.OwLObjectRestriction method), 19  
 is\_object\_restriction() (owlapy.model.OwLRestriction method), 16  
 is\_owl\_nothing() (owlapy.model.OwLAnonymousClassExpression method), 13  
 is\_owl\_nothing() (owlapy.model.OwLClass method), 15  
 is\_owl\_nothing() (owlapy.model.OwLClassExpression method), 13  
 is\_owl\_thing() (owlapy.model.OwLAnonymousClassExpression method), 13  
 is\_owl\_thing() (owlapy.model.OwLClass method), 15  
 is\_owl\_thing() (owlapy.model.OwLClassExpression method), 13  
 is\_owl\_top\_data\_property() (owlapy.model.OwLDataProperty method), 17  
 is\_owl\_top\_data\_property() (owlapy.model.OwLPropertyExpression method), 16  
 is\_owl\_top\_object\_property() (owlapy.model.OwLObjectProperty method), 18  
 is\_owl\_top\_object\_property() (owlapy.model.OwLPropertyExpression method), 16  
 is\_reserved\_vocabulary() (owlapy.model.IRI method), 12  
 is\_string() (owlapy.model.OwLLiteral method), 28  
 is\_thing() (owlapy.model.IRI method), 11  
 is\_using\_triplestore() (owlapy.model.OwLReasoner method), 56  
 iter\_count() (in module owlapy.util), 69

## K

KEY (owlapy.util.LRUCache attribute), 69

## L

LENGTH (owlapy.model.OwLFacet attribute), 9  
 LENGTH (owlapy.vocab.OwLFacet attribute), 71  
 Literals (in module owlapy.model), 12

`load_ontology()` (*owlapy.model.OWLOntologyManager method*), 49  
`LONG` (*owlapy.model.XSDVocabulary attribute*), 9  
`LONG` (*owlapy.vocab.XSDVocabulary attribute*), 71  
`LRUCache` (*class in owlapy.util*), 69

## M

`MANCHESTER_GRAMMAR` (*in module owlapy.parser*), 61  
`manchester_to_owl_expression()` (*in module owlapy.parser*), 66  
`ManchesterOWLSyntaxOWLObjectRenderer` (*class in owlapy.render*), 67  
`ManchesterOWLSyntaxParser` (*class in owlapy.parser*), 62  
`ManchesterParser` (*in module owlapy.parser*), 66  
`ManchesterRenderer` (*in module owlapy.render*), 67  
`mapping` (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58  
`MAX_EXCLUSIVE` (*owlapy.model.OWLFacet attribute*), 9  
`MAX_EXCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 71  
`MAX_INCLUSIVE` (*owlapy.model.OWLFacet attribute*), 9  
`MAX_INCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 71  
`MAX_LENGTH` (*owlapy.model.OWLFacet attribute*), 9  
`MAX_LENGTH` (*owlapy.vocab.OWLFacet attribute*), 71  
`MIN_EXCLUSIVE` (*owlapy.model.OWLFacet attribute*), 9  
`MIN_EXCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 71  
`MIN_INCLUSIVE` (*owlapy.model.OWLFacet attribute*), 9  
`MIN_INCLUSIVE` (*owlapy.vocab.OWLFacet attribute*), 71  
`MIN_LENGTH` (*owlapy.model.OWLFacet attribute*), 9  
`MIN_LENGTH` (*owlapy.vocab.OWLFacet attribute*), 71  
`modal_depth` (*owlapy.owl2sparql.converter.Owl2SparqlConverter property*), 58  
`module`

- `owlapy`, 1
- `owlapy.io`, 59
- `owlapy.model`, 1
- `owlapy.model.providers`, 2
- `owlapy.namespaces`, 60
- `owlapy.owl2sparql`, 57
- `owlapy.owl2sparql.converter`, 57
- `owlapy.parser`, 61
- `owlapy.render`, 66
- `owlapy.util`, 67
- `owlapy.vocab`, 70

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

## N

`named_classes()` (*owlapy.model.OWLEquivalentClassesAxiom method*), 36  
`Namespaces` (*class in owlapy.namespaces*), 60  
`new_count_var()` (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 59  
`new_individual_variable()` (*owlapy.owl2sparql.converter.VariablesMapping method*), 58  
`new_property_variable()` (*owlapy.owl2sparql.converter.VariablesMapping method*), 58  
`NEXT` (*owlapy.util.LRUCache attribute*), 69  
`NNF` (*class in owlapy.util*), 68  
`ns` (*owlapy.namespaces.Namespaces property*), 60  
`ns` (*owlapy.parser.DLSyntaxParser attribute*), 64  
`ns` (*owlapy.parser.ManchesterOWLSyntaxParser attribute*), 62  
`NUMERIC_DATATYPES` (*in module owlapy.model*), 57

## O

`o` (*owlapy.util.OrderedOWLObject attribute*), 68  
`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*), 51  
`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.model.HasOperands method*), 12  
`operands()` (*owlapy.model.OWLDataOneOf method*), 32  
`operands()` (*owlapy.model.OWLHasKeyAxiom method*), 35  
`operands()` (*owlapy.model.OWLNaryBooleanClassExpression method*), 22  
`operands()` (*owlapy.model.OWLNaryDataRange method*), 33

- `operands()` (*owlapy.model.OwLObjectComplementOf method*), 14
- `operands()` (*owlapy.model.OwLObjectOneOf method*), 25
- `operator` (*owlapy.model.OwLFacet property*), 9
- `operator` (*owlapy.vocab.OwLFacet property*), 71
- `OrderedOwLObject` (*class in owlapy.util*), 68
- `OwL` (*in module owlapy.namespaces*), 61
- `OwL2SparqlConverter` (*class in owlapy.owl2sparql.converter*), 58
- `OwL_BOTTOM_DATA_PROPERTY` (*owlapy.model.OwLRDFVocabulary attribute*), 9
- `OwL_BOTTOM_DATA_PROPERTY` (*owlapy.vocab.OwLRDFVocabulary attribute*), 70
- `OwL_BOTTOM_OBJECT_PROPERTY` (*owlapy.model.OwLRDFVocabulary attribute*), 8
- `OwL_BOTTOM_OBJECT_PROPERTY` (*owlapy.vocab.OwLRDFVocabulary attribute*), 70
- `OwL_CLASS` (*owlapy.model.OwLRDFVocabulary attribute*), 8
- `OwL_CLASS` (*owlapy.vocab.OwLRDFVocabulary attribute*), 70
- `owl_expression_to_dl()` (*in module owlapy.render*), 67
- `owl_expression_to_manchester()` (*in module owlapy.render*), 67
- `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*), 70
- `OwL_NOTHING` (*owlapy.model.OwLRDFVocabulary attribute*), 8
- `OwL_NOTHING` (*owlapy.vocab.OwLRDFVocabulary attribute*), 70
- `OwL_THING` (*owlapy.model.OwLRDFVocabulary attribute*), 8
- `OwL_THING` (*owlapy.vocab.OwLRDFVocabulary attribute*), 70
- `OwL_TOP_DATA_PROPERTY` (*owlapy.model.OwLRDFVocabulary attribute*), 8
- `OwL_TOP_DATA_PROPERTY` (*owlapy.vocab.OwLRDFVocabulary attribute*), 70
- `OwL_TOP_OBJECT_PROPERTY` (*owlapy.model.OwLRDFVocabulary attribute*), 8
- `OwL_TOP_OBJECT_PROPERTY` (*owlapy.vocab.OwLRDFVocabulary attribute*), 70
- `OwLAnnotation` (*class in owlapy.model*), 40
- `OwLAnnotationAssertionAxiom` (*class in owlapy.model*), 40
- `OwLAnnotationAxiom` (*class in owlapy.model*), 40
- `OwLAnnotationObject` (*class in owlapy.model*), 10
- `OwLAnnotationProperty` (*class in owlapy.model*), 40
- `OwLAnnotationPropertyDomainAxiom` (*class in owlapy.model*), 41
- `OwLAnnotationPropertyRangeAxiom` (*class in owlapy.model*), 42
- `OwLAnnotationSubject` (*class in owlapy.model*), 10
- `OwLAnnotationValue` (*class in owlapy.model*), 10
- `OwLAnonymousClassExpression` (*class in owlapy.model*), 13
- `owlapy`
  - `module`, 1
- `owlapy.io`
  - `module`, 59
- `owlapy.model`
  - `module`, 1
- `owlapy.model.providers`
  - `module`, 2
- `owlapy.namespaces`
  - `module`, 60
- `owlapy.owl2sparql`
  - `module`, 57
- `owlapy.owl2sparql.converter`
  - `module`, 57
- `owlapy.parser`
  - `module`, 61
- `owlapy.render`
  - `module`, 66
- `owlapy.util`
  - `module`, 67
- `owlapy.vocab`
  - `module`, 70
- `OwLAsymmetricObjectPropertyAxiom` (*class in owlapy.model*), 44
- `OwLAxiom` (*class in owlapy.model*), 26
- `OwLBooleanClassExpression` (*class in owlapy.model*), 14
- `OwLBottomDataProperty` (*in module owlapy.model*), 56
- `OwLBottomObjectProperty` (*in module owlapy.model*), 56
- `OwLCardinalityRestriction` (*class in owlapy.model*), 22
- `OwLClass` (*class in owlapy.model*), 15
- `OwLClassAssertionAxiom` (*class in owlapy.model*), 39
- `OwLClassAxiom` (*class in owlapy.model*), 34
- `OwLClassExpression` (*class in owlapy.model*), 13

OWLDataAllValuesFrom (class in owlapy.model), 30  
 OWLDataCardinalityRestriction (class in owlapy.model), 29  
 OWLDataComplementOf (class in owlapy.model), 30  
 OWLDataExactCardinality (class in owlapy.model), 30  
 OWLDataHasValue (class in owlapy.model), 31  
 OWLDataIntersectionOf (class in owlapy.model), 33  
 OWLDataMaxCardinality (class in owlapy.model), 31  
 OWLDataMinCardinality (class in owlapy.model), 31  
 OWLDataOneOf (class in owlapy.model), 32  
 OWLDataProperty (class in owlapy.model), 17  
 OWLDataPropertyAssertionAxiom (class in owlapy.model), 43  
 OWLDataPropertyAxiom (class in owlapy.model), 34  
 OWLDataPropertyCharacteristicAxiom (class in owlapy.model), 45  
 OWLDataPropertyDomainAxiom (class in owlapy.model), 46  
 OWLDataPropertyExpression (class in owlapy.model), 17  
 OWLDataPropertyRangeAxiom (class in owlapy.model), 47  
 OWLDataRange (class in owlapy.model), 13  
 OWLDataRestriction (class in owlapy.model), 19  
 OWLDataSomeValuesFrom (class in owlapy.model), 32  
 OWLDatatype (class in owlapy.model), 26  
 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), 27  
 OWLDataUnionOf (class in owlapy.model), 33  
 OWLDeclarationAxiom (class in owlapy.model), 34  
 OWLDifferentIndividualsAxiom (class in owlapy.model), 37  
 OWLDisjointClassesAxiom (class in owlapy.model), 36  
 OWLDisjointDataPropertiesAxiom (class in owlapy.model), 38  
 OWLDisjointObjectPropertiesAxiom (class in owlapy.model), 38  
 OWLDisjointUnionAxiom (class in owlapy.model), 39  
 OWLEntity (class in owlapy.model), 15  
 OWLEquivalentClassesAxiom (class in owlapy.model), 36  
 OWLEquivalentDataPropertiesAxiom (class in owlapy.model), 38  
 OWLEquivalentObjectPropertiesAxiom (class in owlapy.model), 37  
 OWLFacet (class in owlapy.model), 9  
 OWLFacet (class in owlapy.vocab), 71  
 OWLFacetRestriction (class in owlapy.model), 27  
 OWLFunctionalDataPropertyAxiom (class in owlapy.model), 45  
 OWLFunctionalObjectPropertyAxiom (class in owlapy.model), 44  
 OWLHasKeyAxiom (class in owlapy.model), 35  
 OWLHasValueRestriction (class in owlapy.model), 20  
 OWLImportsDeclaration (class in owlapy.model), 33  
 OWLIndividual (class in owlapy.model), 24  
 OWLIndividualAxiom (class in owlapy.model), 34  
 OWLInverseFunctionalObjectPropertyAxiom (class in owlapy.model), 45  
 OWLInverseObjectPropertiesAxiom (class in owlapy.model), 38  
 OWLIrreflexiveObjectPropertyAxiom (class in owlapy.model), 45  
 OWLLiteral (class in owlapy.model), 27  
 OWLLogicalAxiom (class in owlapy.model), 33  
 OWLNamedIndividual (class in owlapy.model), 25  
 OWLNamedObject (class in owlapy.model), 14  
 OWLNaryAxiom (class in owlapy.model), 35  
 OWLNaryBooleanClassExpression (class in owlapy.model), 21  
 OWLNaryClassAxiom (class in owlapy.model), 35  
 OWLNaryDataRange (class in owlapy.model), 32  
 OWLNaryIndividualAxiom (class in owlapy.model), 36  
 OWLNaryPropertyAxiom (class in owlapy.model), 37  
 OWLNegativeDataPropertyAssertionAxiom (class in owlapy.model), 44  
 OWLNegativeObjectPropertyAssertionAxiom (class in owlapy.model), 43  
 OWLNothing (in module owlapy.model), 56  
 OWLObject (class in owlapy.model), 10  
 OWLObjectAllValuesFrom (class in owlapy.model), 21  
 OWLObjectCardinalityRestriction (class in owlapy.model), 23

OWLObjectComplementOf (class in owlapy.model), 14  
 OWLObjectExactCardinality (class in owlapy.model), 23  
 OWLObjectHasSelf (class in owlapy.model), 24  
 OWLObjectHasValue (class in owlapy.model), 24  
 OWLObjectIntersectionOf (class in owlapy.model), 22  
 OWLObjectInverseOf (class in owlapy.model), 18  
 OWLObjectMaxCardinality (class in owlapy.model), 23  
 OWLObjectMinCardinality (class in owlapy.model), 23  
 OWLObjectOneOf (class in owlapy.model), 24  
 OWLObjectParser (class in owlapy.io), 60  
 OWLObjectProperty (class in owlapy.model), 18  
 OWLObjectPropertyAssertionAxiom (class in owlapy.model), 43  
 OWLObjectPropertyAxiom (class in owlapy.model), 34  
 OWLObjectPropertyCharacteristicAxiom (class in owlapy.model), 44  
 OWLObjectPropertyDomainAxiom (class in owlapy.model), 46  
 OWLObjectPropertyExpression (class in owlapy.model), 16  
 OWLObjectPropertyRangeAxiom (class in owlapy.model), 47  
 OWLObjectRenderer (class in owlapy.io), 59  
 OWLObjectRestriction (class in owlapy.model), 19  
 OWLObjectSomeValuesFrom (class in owlapy.model), 21  
 OWLObjectUnionOf (class in owlapy.model), 22  
 OWLOntology (class in owlapy.model), 47  
 OWLOntologyChange (class in owlapy.model), 49  
 OWLOntologyID (class in owlapy.model), 25  
 OWLOntologyManager (class in owlapy.model), 49  
 OWLProperty (class in owlapy.model), 17  
 OWLPropertyAssertionAxiom (class in owlapy.model), 43  
 OWLPropertyAxiom (class in owlapy.model), 33  
 OWLPropertyDomainAxiom (class in owlapy.model), 46  
 OWLPropertyExpression (class in owlapy.model), 16  
 OWLPropertyRange (class in owlapy.model), 12  
 OWLPropertyRangeAxiom (class in owlapy.model), 46  
 OWLQuantifiedDataRestriction (class in owlapy.model), 29  
 OWLQuantifiedObjectRestriction (class in owlapy.model), 20  
 OWLQuantifiedRestriction (class in owlapy.model), 20  
 OWLRDFVocabulary (class in owlapy.model), 8  
 OWLRDFVocabulary (class in owlapy.vocab), 70  
 OWLReasoner (class in owlapy.model), 50  
 OWLReflexiveObjectPropertyAxiom (class in owlapy.model), 45  
 OWLRestriction (class in owlapy.model), 16  
 OWLSameIndividualAxiom (class in owlapy.model), 37  
 OWLSubAnnotationPropertyOfAxiom (class in owlapy.model), 41  
 OWLSubClassOfAxiom (class in owlapy.model), 38  
 OWLSubDataPropertyOfAxiom (class in owlapy.model), 43  
 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), 44

## P

parent (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58  
 parent\_var (owlapy.owl2sparql.converter.Owl2SparqlConverter attribute), 58  
 parse\_boolean() (owlapy.model.OWLLiteral method), 28  
 parse\_date() (owlapy.model.OWLLiteral method), 28  
 parse\_datetime() (owlapy.model.OWLLiteral method), 29  
 parse\_double() (owlapy.model.OWLLiteral method), 28  
 parse\_duration() (owlapy.model.OWLLiteral method), 29  
 parse\_expression() (owlapy.io.OwlObjectParser method), 60  
 parse\_expression() (owlapy.parser.DLSyntaxParser method), 64  
 parse\_expression() (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 parse\_integer() (owlapy.model.OWLLiteral method), 28  
 parse\_string() (owlapy.model.OWLLiteral method), 28  
 PATTERN (owlapy.model.OWLFacet attribute), 9

PATTERN (*owlapy.vocab.OWLFacet attribute*), 71  
 peek () (*in module owlapy.owl2sparql.converter*), 57  
 prefix (*owlapy.namespaces.Namespaces property*), 60  
 PREV (*owlapy.util.LRUCache attribute*), 69  
 process () (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 59  
 properties (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58  
 properties () (*owlapy.model.OwLNaryPropertyAxiom method*), 37

## R

RDF (*in module owlapy.namespaces*), 61  
 RDFS (*in module owlapy.namespaces*), 61  
 RDFS\_LITERAL (*owlapy.model.OwLRDFVocabulary attribute*), 9  
 RDFS\_LITERAL (*owlapy.vocab.OwLRDFVocabulary attribute*), 70  
 reminder (*owlapy.model.IRI property*), 11  
 reminder (*owlapy.model.OwLClass property*), 15  
 remove\_axiom () (*owlapy.model.OwLOntologyManager method*), 50  
 render () (*owlapy.io.OwLObjectRenderer method*), 59  
 render () (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 58  
 render () (*owlapy.render.DLSyntaxObjectRenderer method*), 66  
 render () (*owlapy.render.ManchesterOWLSyntaxOwLObjectRenderer method*), 67  
 Restriction\_Literals (*in module owlapy.model.providers*), 2  
 RESULT (*owlapy.util.LRUCache attribute*), 69

## S

same\_individuals () (*owlapy.model.OwLReasoner method*), 52  
 save\_ontology () (*owlapy.model.OwLOntologyManager method*), 50  
 sentinel (*owlapy.util.LRUCache attribute*), 69  
 set\_short\_form\_provider () (*owlapy.io.OwLObjectRenderer method*), 59  
 set\_short\_form\_provider () (*owlapy.render.DLSyntaxObjectRenderer method*), 66  
 set\_short\_form\_provider () (*owlapy.render.ManchesterOWLSyntaxOwLObjectRenderer method*), 67  
 slots (*owlapy.parser.DLSyntaxParser attribute*), 64  
 slots (*owlapy.parser.ManchesterOWLSyntaxParser attribute*), 62  
 sparql (*owlapy.owl2sparql.converter.Owl2SparqlConverter attribute*), 58  
 stack\_parent () (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 59  
 stack\_variable () (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 59  
 str (*owlapy.model.IRI property*), 11  
 str (*owlapy.model.OwLClass property*), 15  
 str (*owlapy.model.OwLNamedIndividual property*), 25  
 str (*owlapy.model.OwLObjectProperty property*), 18  
 STRING (*owlapy.model.XSDVocabulary attribute*), 9  
 STRING (*owlapy.vocab.XSDVocabulary attribute*), 71  
 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*), 55  
 super\_object\_properties () (*owlapy.model.OwLReasoner method*), 55  
 symbolic\_form (*owlapy.model.OwLFacet property*), 9  
 symbolic\_form (*owlapy.vocab.OwLFacet property*), 71

## T

TIME\_DATATYPES (*in module owlapy.model*), 57  
 to\_python () (*owlapy.model.OwLLiteral method*), 29  
 to\_string\_id () (*owlapy.model.OwLEntity method*), 15  
 TopLevelCNF (*class in owlapy.util*), 68  
 TopLevelDNF (*class in owlapy.util*), 69  
 TopOWLDatatype (*in module owlapy.model*), 56  
 TOTAL\_DIGITS (*owlapy.model.OwLFacet attribute*), 9  
 TOTAL\_DIGITS (*owlapy.vocab.OwLFacet attribute*), 71  
 triple () (*owlapy.owl2sparql.converter.Owl2SparqlConverter method*), 59  
 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*), 30  
 type\_index (*owlapy.model.OwLDataComplementOf attribute*), 30



type\_index (owlapy.model.OWLDataExactCardinality attribute), 31  
 type\_index (owlapy.model.OWLDataHasValue attribute), 31  
 type\_index (owlapy.model.OWLDataIntersectionOf attribute), 33  
 type\_index (owlapy.model.OWLDataMaxCardinality attribute), 31  
 type\_index (owlapy.model.OWLDataMinCardinality attribute), 32  
 type\_index (owlapy.model.OWLDataOneOf attribute), 32  
 type\_index (owlapy.model.OWLDataProperty attribute), 17  
 type\_index (owlapy.model.OWLDataSomeValuesFrom attribute), 32  
 type\_index (owlapy.model.OWLDatatype attribute), 27  
 type\_index (owlapy.model.OWLDatatypeRestriction attribute), 27  
 type\_index (owlapy.model.OWLDataUnionOf attribute), 33  
 type\_index (owlapy.model.OWLFacetRestriction attribute), 27  
 type\_index (owlapy.model.OWLLiteral attribute), 28  
 type\_index (owlapy.model.OWLNamedIndividual attribute), 25  
 type\_index (owlapy.model.OWLObjectAllValuesFrom attribute), 21  
 type\_index (owlapy.model.OWLObjectComplementOf attribute), 14  
 type\_index (owlapy.model.OWLObjectExactCardinality attribute), 23  
 type\_index (owlapy.model.OWLObjectHasSelf attribute), 24  
 type\_index (owlapy.model.OWLObjectHasValue attribute), 24  
 type\_index (owlapy.model.OWLObjectIntersectionOf attribute), 22  
 type\_index (owlapy.model.OWLObjectInverseOf attribute), 18  
 type\_index (owlapy.model.OWLObjectMaxCardinality attribute), 23  
 type\_index (owlapy.model.OWLObjectMinCardinality attribute), 23  
 type\_index (owlapy.model.OWLObjectOneOf attribute), 25  
 type\_index (owlapy.model.OWLObjectProperty attribute), 18  
 type\_index (owlapy.model.OWLObjectSomeValuesFrom attribute), 21  
 type\_index (owlapy.model.OWLObjectUnionOf attribute), 22  
 type\_index (owlapy.model.OWLOntology attribute), 47  
 types () (owlapy.model.OWLReasoner method), 55

## V

values () (owlapy.model.OWLDataOneOf method), 32  
 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), 65  
 visit\_abbreviated\_iri () (owlapy.parser.ManchesterOWLSyntaxParser method), 63  
 visit\_boolean\_literal () (owlapy.parser.DLSyntaxParser method), 65  
 visit\_boolean\_literal () (owlapy.parser.ManchesterOWLSyntaxParser method), 63  
 visit\_cardinality\_res () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_cardinality\_res () (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 visit\_class\_expression () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_class\_expression () (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 visit\_class\_iri () (owlapy.parser.DLSyntaxParser method), 65  
 visit\_class\_iri () (owlapy.parser.ManchesterOWLSyntaxParser method), 63  
 visit\_data\_cardinality\_res () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_data\_cardinality\_res () (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 visit\_data\_intersection () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_data\_intersection () (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 visit\_data\_parentheses () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_data\_parentheses () (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 visit\_data\_primary () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_data\_primary () (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 visit\_data\_property\_iri () (owlapy.parser.DLSyntaxParser method), 65  
 visit\_data\_property\_iri () (owlapy.parser.ManchesterOWLSyntaxParser method), 63  
 visit\_data\_some\_only\_res () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_data\_some\_only\_res () (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 visit\_data\_union () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_data\_union () (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 visit\_data\_value\_res () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_data\_value\_res () (owlapy.parser.ManchesterOWLSyntaxParser method), 62  
 visit\_datatype () (owlapy.parser.DLSyntaxParser method), 65  
 visit\_datatype () (owlapy.parser.ManchesterOWLSyntaxParser method), 63  
 visit\_datatype\_iri () (owlapy.parser.DLSyntaxParser method), 65  
 visit\_datatype\_iri () (owlapy.parser.ManchesterOWLSyntaxParser method), 63  
 visit\_datatype\_restriction () (owlapy.parser.DLSyntaxParser method), 64  
 visit\_datatype\_restriction () (owlapy.parser.ManchesterOWLSyntaxParser method), 62

`visit_date_literal()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_date_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_datetime_literal()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_datetime_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_decimal_literal()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_decimal_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_duration_literal()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_duration_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_facet()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_facet()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_facet_restriction()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_facet_restriction()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_facet_restrictions()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_facet_restrictions()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_float_literal()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_float_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_full_iri()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_full_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_has_self()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_has_self()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_individual_iri()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_individual_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_individual_list()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_individual_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_integer_literal()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_integer_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_intersection()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_intersection()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_iri()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_literal()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_literal_list()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_literal_list()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_non_negative_integer()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_non_negative_integer()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_object_property()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_object_property()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_object_property_iri()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_object_property_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_parentheses()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_parentheses()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_primary()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_primary()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_quoted_string()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_quoted_string()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_simple_iri()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_simple_iri()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_some_only_res()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_some_only_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_string_literal_language()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_string_literal_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_string_literal_no_language()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_string_literal_no_language()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_typed_literal()` (*owlapy.parser.DLSyntaxParser method*), 65  
`visit_typed_literal()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 63  
`visit_union()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_union()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62  
`visit_value_res()` (*owlapy.parser.DLSyntaxParser method*), 64  
`visit_value_res()` (*owlapy.parser.ManchesterOWLSyntaxParser method*), 62

## X

XSD (*in module owlapy.namespaces*), 61  
XSDVocabulary (*class in owlapy.model*), 9  
XSDVocabulary (*class in owlapy.vocab*), 70