
Take Your Terms from Ontologies (tyto)

Release 1.0

Bryan Bartley

Apr 22, 2022

CONTENTS

| | | |
|----------|--|-----------|
| 1 | Examples using dynamic attributes | 3 |
| 2 | Example use with subscripts | 5 |
| 3 | Query Backend | 7 |
| 4 | Support for SBOL Ontologies | 9 |
| 5 | Support for Other Ontologies | 11 |
| 6 | Inference | 13 |
| 6.1 | API Reference | 13 |
| 7 | Indices and tables | 31 |
| | Python Module Index | 33 |
| | Index | 35 |

Tyto is a tool that supports standardized annotation practices using terms from controlled vocabularies and ontologies. Tyto allows a user to reference ontology terms in their code using human-readable labels rather than uniform resource identifiers (URIs), thus rendering code more readable and easier to understand.

EXAMPLES USING DYNAMIC ATTRIBUTES

Tyto allows the user to reference an ontology term as an attribute on an Ontology instance. In the following example, the term `promoter` is referenced from an Ontology instance representing the [Sequence Ontology](#).

```
>>> SO.promoter
'https://identifiers.org/SO:0000167'
```

Some ontology terms have spaces, such as the term `functional entity` from the Systems Biology Ontology. In these cases, replace the space with an underscore:

```
>>> SBO.functional_entity
'https://identifiers.org/SBO:0000241'
```


EXAMPLE USE WITH SUBSCRIPTS

Some ontology terms may have special characters, especially in the NCBI Taxonomy. In this case, dynamic attributes cannot be used as it would result in invalid Python symbols. Instead, use subscripting:

```
>>> NCBITaxon['Escherichia coli 0157:H-']  
'https://identifiers.org/taxonomy:183192'
```


QUERY BACKEND

Tyto's back-end dynamically queries ontology lookup services for the URI that corresponds to the attribute name or subscript. This dynamic querying approach is in keeping with the principles of the semantic web, in which knowledge is interlinked and distributed across the web rather than concentrated in isolated resources. In this way, Tyto provides up-to-date access to hundreds of ontologies without the need for packaging a large amount of data with the distribution.

Currently Tyto supports queries to [Ontobee](#) and the [EBI Ontology Lookup Service](#). In addition, it includes an extensible framework so that users may add their own REST services or SPARQL endpoints. It is also possible to query local OWL files for offline work.

SUPPORT FOR SBOL ONTOLOGIES

Tyto was originally developed to support use of the [Synthetic Biology Open Language](#) which uses ontologies as a source of standardized terms for annotating synthetic biology data. Tyto provides “out-of-the-box” support for most of the ontologies used in SBOL:

- SO: sequence ontology
- SBO: systems biology ontology
- NCBITaxon: NCBI taxonomy
- NCIT: National Cancer Institute Thesaurus
- OM: Ontology of Units of Measure

Note: SBOL identifiers standardize on the `identifiers.org` namespace. By default most ontology lookup services standardize on the `purl.org` namespace. Tyto’s built-in `Ontology` interfaces automatically translate from `purl.org` to `identifiers.org` namespaces.

SUPPORT FOR OTHER ONTOLOGIES

Ontobee and EBI Ontology Lookup Service host hundreds of ontologies. Even though Tyto has built-in support for only a few of these, it is easy to define your own `Ontology` interface.

First identify the URI associated with the ontology you wish to use. To do this, use the `get_ontologies` method on your `Endpoint` instance to get a dictionary of available ontologies:

```
>>> for uri, ontology in tyto.EBIOntologyLookupService.get_ontologies().items():
...     print(uri, ontology)
...
http://purl.obolibrary.org/obo/aeo.owl aeo
http://purl.allotrope.org/voc/afo/merged-OLS/REC/2019/05/10 afo
http://purl.obolibrary.org/obo/agro-edit.owl agro
.
.
.
```

Once you have identified the URI of your desired ontology, instantiate an `Ontology`, specifying its URI and the lookup service:

```
>>> from tyto import EBIOntologyLookupService, Ontology
>>> KISAO = Ontology(uri='http://www.biomodels.net/kisao/KISAO_FULL#',
↳ endpoints=[EBIOntologyLookupService])
>>> KISAO.Gillespie_direct_algorithm
'http://www.biomodels.net/kisao/KISAO#KISAO_0000029'
```


INFERENCE

Tyto can be used to reason about the relationships between terms in an ontology.

For example:

```
>>> S0.inducible_promoter.is_a(S0.promoter)
True
>>> S0.inducible_promoter.is_a(S0.ribosome_entry_site)
False
```

Other supported inference methods include:

```
term1.is_a(term2)
term1.is_descendant_of(term2)
term1.is_ancestor_of(term2)
term1.is_child_of(term2)
term1.is_parent_of(term2)
```

6.1 API Reference

This page contains auto-generated API reference documentation¹.

6.1.1 tyto

Subpackages

`tyto.endpoint`

Submodules

`tyto.endpoint.endpoint`

Module Contents

¹ Created with sphinx-autoapi

Classes

| | |
|------------------------------------|--|
| <i>QueryBackend</i> | Helper class that provides a standard way to create an ABC using |
| <i>SPARQLBuilder</i> | Mixin class that provides SPARQL queries to SPARQLEndpoint and GraphEndpoint classes |
| <i>Endpoint</i> | Helper class that provides a standard way to create an ABC using |
| <i>RESTEndpoint</i> | Class for issuing and handling HTTP requests |
| <i>SPARQLEndpoint</i> | Class which issues SPARQL queries to an endpoint |
| <i>GraphEndpoint</i> | Class for querying a local graph from a file |
| <i>OntobeeEndpoint</i> | Class which issues SPARQL queries to an endpoint |
| <i>EBIOntologyLookupServiceAPI</i> | Class for issuing and handling HTTP requests |
| <i>PUG_REST</i> | Class for issuing and handling HTTP requests |

Attributes

| | |
|---------------------------------|--|
| <i>Ontobee</i> | Endpoint instance representing Ontobee. Ontobee is the default linked data server for most OBO Foundry library ontologies, but is also been used for many non-OBO ontologies. |
| <i>EBIOntologyLookupService</i> | The Ontology Lookup Service (OLS) is a repository for biomedical ontologies that aims to provide a single point of access to the latest ontology versions. Hosted by the European Bioinformatics Institute |
| <i>PubChemAPI</i> | |

class tyto.endpoint.endpoint.**QueryBackend**

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

abstract `get_term_by_uri(self, ontology: Ontology, uri: str)`

abstract `get_uri_by_term(self, ontology: Ontology, term: str)`

class tyto.endpoint.endpoint.**SPARQLBuilder**

Mixin class that provides SPARQL queries to SPARQLEndpoint and GraphEndpoint classes

get_term_by_uri(*self*, *ontology*, *uri*)

Query for a term by its URI

Parameters

- **uri** – The URI for the term
- **ontology** – The Ontology to query

Uri URI

Ontology Ontology

get_uri_by_term(self, ontology: *Ontology*, term: str) → str

Query for the URI associated with the given an ontology term (e.g., “promoter”) :param term: The ontology term :term: str :param ontology: The ontology to query :ontology: *Ontology*

is_child_of(self, ontology: *Ontology*, child_uri: str, parent_uri: str) → bool

is_parent_of(self, ontology: *Ontology*, parent_uri: str, child_uri: str) → bool

is_ancestor_of(self, ontology: *Ontology*, ancestor_uri: str, descendant_uri: str) → bool

is_descendant_of(self, ontology: *Ontology*, descendant_uri: str, ancestor_uri: str) → bool

get_ontologies(self)

class tyto.endpoint.endpoint.**Endpoint**(url)

Bases: *QueryBackend*, abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

class tyto.endpoint.endpoint.**RESTEndpoint**(url)

Bases: *QueryBackend*, abc.ABC

Class for issuing and handling HTTP requests

_get_request(self, ontology: *Ontology*, request: str)

class tyto.endpoint.endpoint.**SPARQLEndpoint**(url)

Bases: *SPARQLBuilder*, *Endpoint*

Class which issues SPARQL queries to an endpoint

query(self, ontology, sparql, err_msg)

Issues SPARQL query

convert(self, response)

Converts standard SPARQL query JSON into a flat list.

See <https://www.w3.org/TR/2013/REC-sparql11-results-json-20130321/>

class tyto.endpoint.endpoint.**GraphEndpoint**(file_path)

Bases: *SPARQLBuilder*, *Endpoint*

Class for querying a local graph from a file

is_loaded(self)

load(self)

query(self, ontology, sparql, err_msg)

convert(self, response)

Extracts and flattens queried variables from rdflib response into a list

class tyto.endpoint.endpoint.**OntobeeEndpoint**

Bases: *SPARQLEndpoint*

Class which issues SPARQL queries to an endpoint

query(self, ontology, sparql, err_msg)

Issues SPARQL query

class tyto.endpoint.endpoint.**EBIOntologyLookupServiceAPI**

Bases: *RESTEndpoint*

Class for issuing and handling HTTP requests

_load_ontology_ids(*self*)

_get_request(*self*, *ontology*: *Ontology*, *get_request*: *str*)

get_term_by_uri(*self*, *ontology*: *Ontology*, *uri*: *str*)

get_uri_by_term(*self*, *ontology*: *Ontology*, *term*: *str*)

get_parents(*self*, *ontology*: *Ontology*, *uri*: *str*)

get_children(*self*, *ontology*: *Ontology*, *uri*: *str*)

get_descendants(*self*, *ontology*: *Ontology*, *uri*: *str*)

get_ancestors(*self*, *ontology*: *Ontology*, *uri*: *str*)

is_parent_of(*self*, *ontology*: *Ontology*, *parent_uri*: *str*, *child_uri*: *str*) → bool

is_child_of(*self*, *ontology*: *Ontology*, *child_uri*: *str*, *parent_uri*: *str*) → bool

is_descendant_of(*self*, *ontology*: *Ontology*, *descendant_uri*: *str*, *ancestor*: *str*) → bool

is_ancestor_of(*self*, *ontology*: *Ontology*, *ancestor_uri*: *str*, *descendant_uri*: *str*) → bool

get_ontologies(*self*)

convert(*self*, *response*)

query(*self*, *query*)

class tyto.endpoint.endpoint.**PUG_REST**

Bases: *RESTEndpoint*

Class for issuing and handling HTTP requests

get_term_by_uri(*self*, *ontology*: *Ontology*, *uri*: *str*)

get_uri_by_term(*self*, *ontology*: *Ontology*, *term*: *str*)

tyto.endpoint.endpoint.Ontobee

Endpoint instance representing Ontobee. Ontobee is the default linked data server for most OBO Foundry library ontologies, but is also been used for many non-OBO ontologies.

tyto.endpoint.endpoint.EBIOntologyLookupService

The Ontology Lookup Service (OLS) is a repository for biomedical ontologies that aims to provide a single point of access to the latest ontology versions. Hosted by the European Bioinformatics Institute

tyto.endpoint.endpoint.PubChemAPI

Package Contents

Classes

| | |
|------------------------------------|--|
| <i>QueryBackend</i> | Helper class that provides a standard way to create an ABC using |
| <i>SPARQLBuilder</i> | Mixin class that provides SPARQL queries to SPARQLEndpoint and GraphEndpoint classes |
| <i>Endpoint</i> | Helper class that provides a standard way to create an ABC using |
| <i>RESTEndpoint</i> | Class for issuing and handling HTTP requests |
| <i>SPARQLEndpoint</i> | Class which issues SPARQL queries to an endpoint |
| <i>GraphEndpoint</i> | Class for querying a local graph from a file |
| <i>OntobeeEndpoint</i> | Class which issues SPARQL queries to an endpoint |
| <i>EBIOntologyLookupServiceAPI</i> | Class for issuing and handling HTTP requests |
| <i>PUG_REST</i> | Class for issuing and handling HTTP requests |

Attributes

| | |
|---------------------------------|--|
| <i>Ontobee</i> | Endpoint instance representing Ontobee. Ontobee is the default linked data server for most OBO Foundry library ontologies, but is also been used for many non-OBO ontologies. |
| <i>EBIOntologyLookupService</i> | The Ontology Lookup Service (OLS) is a repository for biomedical ontologies that aims to provide a single point of access to the latest ontology versions. Hosted by the European Bioinformatics Institute |
| <i>PubChemAPI</i> | |

class tyto.endpoint.QueryBackend

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

abstract `get_term_by_uri(self, ontology: Ontology, uri: str)`

abstract `get_uri_by_term(self, ontology: Ontology, term: str)`

class tyto.endpoint.SPARQLBuilder

Mixin class that provides SPARQL queries to SPARQLEndpoint and GraphEndpoint classes

get_term_by_uri(self, ontology, uri)

Query for a term by its URI

Parameters

- **uri** – The URI for the term
- **ontology** – The Ontology to query

Uri URI

Ontology Ontology

get_uri_by_term(self, ontology: [Ontology](#), term: str) → str

Query for the URI associated with the given an ontology term (e.g., “promoter”) :param term: The ontology term :term: str :param ontology: The ontology to query :ontology: [Ontology](#)

is_child_of(self, ontology: [Ontology](#), child_uri: str, parent_uri: str) → bool

is_parent_of(self, ontology: [Ontology](#), parent_uri: str, child_uri: str) → bool

is_ancestor_of(self, ontology: [Ontology](#), ancestor_uri: str, descendant_uri: str) → bool

is_descendant_of(self, ontology: [Ontology](#), descendant_uri: str, ancestor_uri: str) → bool

get_ontologies(self)

class tyto.endpoint.**Endpoint**(url)

Bases: [QueryBackend](#), abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

class tyto.endpoint.**RESTEndpoint**(url)

Bases: [QueryBackend](#), abc.ABC

Class for issuing and handling HTTP requests

_get_request(self, ontology: [Ontology](#), request: str)

class tyto.endpoint.**SPARQLEndpoint**(url)

Bases: [SPARQLBuilder](#), [Endpoint](#)

Class which issues SPARQL queries to an endpoint

query(self, ontology, sparql, err_msg)

Issues SPARQL query

convert(self, response)

Converts standard SPARQL query JSON into a flat list.

See <https://www.w3.org/TR/2013/REC-sparql11-results-json-20130321/>

class tyto.endpoint.**GraphEndpoint**(file_path)

Bases: [SPARQLBuilder](#), [Endpoint](#)

Class for querying a local graph from a file

is_loaded(self)

load(self)

query(self, ontology, sparql, err_msg)

convert(self, response)

Extracts and flattens queried variables from rdflib response into a list

class tyto.endpoint.**OntobeeEndpoint**

Bases: [SPARQLEndpoint](#)

Class which issues SPARQL queries to an endpoint

query(self, ontology, sparql, err_msg)

Issues SPARQL query

```
class tyto.endpoint.EBIOntologyLookupServiceAPI
```

Bases: [RESTEndpoint](#)

Class for issuing and handling HTTP requests

```
_load_ontology_ids(self)
```

```
_get_request(self, ontology: Ontology, get_request: str)
```

```
get_term_by_uri(self, ontology: Ontology, uri: str)
```

```
get_uri_by_term(self, ontology: Ontology, term: str)
```

```
get_parents(self, ontology: Ontology, uri: str)
```

```
get_children(self, ontology: Ontology, uri: str)
```

```
get_descendants(self, ontology: Ontology, uri: str)
```

```
get_ancestors(self, ontology: Ontology, uri: str)
```

```
is_parent_of(self, ontology: Ontology, parent_uri: str, child_uri: str) → bool
```

```
is_child_of(self, ontology: Ontology, child_uri: str, parent_uri: str) → bool
```

```
is_descendant_of(self, ontology: Ontology, descendant_uri: str, ancestor: str) → bool
```

```
is_ancestor_of(self, ontology: Ontology, ancestor_uri: str, descendant_uri: str) → bool
```

```
get_ontologies(self)
```

```
convert(self, response)
```

```
query(self, query)
```

```
class tyto.endpoint.PUG_REST
```

Bases: [RESTEndpoint](#)

Class for issuing and handling HTTP requests

```
get_term_by_uri(self, ontology: Ontology, uri: str)
```

```
get_uri_by_term(self, ontology: Ontology, term: str)
```

```
tyto.endpoint.Ontobee
```

Endpoint instance representing Ontobee. Ontobee is the default linked data server for most OBO Foundry library ontologies, but is also been used for many non-OBO ontologies.

```
tyto.endpoint.EBIOntologyLookupService
```

The Ontology Lookup Service (OLS) is a repository for biomedical ontologies that aims to provide a single point of access to the latest ontology versions. Hosted by the European Bioinformatics Institute

```
tyto.endpoint.PubChemAPI
```

Submodules

`tyto.edam`

Module Contents

`tyto.edam.EDAM`

EDAM (EMBRACE Data and Methods) is an ontology of common bioinformatics operations, topics, types of data including identifiers, and formats. EDAM comprises common concepts (shared within the bioinformatics community) that apply to semantic annotation of resources.

`tyto.ncbi_taxon`

Module Contents

`tyto.ncbi_taxon.NCBITaxon`

Ontology instance for NCBI Taxonomy

`tyto.ncbi_taxon._sanitize_uri`

`tyto.ncbi_taxon._reverse_sanitize_uri`

`tyto.ncit`

Module Contents

`tyto.ncit.NCIT`

Ontology instance for National Cancer Institute Thesaurus

`tyto.ncit._sanitize_uri`

`tyto.ncit._reverse_sanitize_uri`

`tyto.om`

Module Contents

`tyto.om.OM`

Ontology instance for Ontology of Units of Measure

`tyto.om._sanitize_term`

`tyto.om._reverse_sanitize_term`

`tyto.pubchem`

Module Contents

`tyto.pubchem.PubChem`

`tyto.sbo`

Module Contents

`tyto.sbo.SBO`

Ontology instance for Systems Biology Ontology

`tyto.sbo._sanitize_uri`

`tyto.sbo._reverse_sanitize_uri`

`tyto.sbol2`

Module Contents

`tyto.sbol2.SBOL2`

`tyto.sbol3`

Module Contents

`tyto.sbol3.SBOL3`

`tyto.so`

Module Contents

`tyto.so.SO`

Ontology instance for Sequence Ontology

`tyto.so._sanitize_uri`

`tyto.so._reverse_sanitize_uri`

`tyto.so._sanitize_term`

tyto.tyto

Module Contents

Classes

| | |
|-----------------|--|
| <i>Ontology</i> | The Ontology class provides an abstraction layer for accessing ontologies, and a |
| <i>URI</i> | The URI class wraps the Python string primitive type, enabling the use of inference methods on the represented uniform resource identifier |
| <i>Term</i> | The URI class wraps the Python string primitive type, enabling the use of inference methods on the represented uniform resource identifier |

Functions

| | |
|--|--|
| <i>installation_path</i> (relative_path) | |
| <i>multi_replace</i> (target_uri, old_namespaces, new_namespace) | |
| <i>configure_cache_size</i> (maxsize=1000) | Set the size of the in-memory cache in order to optimize performance and frequency of queries over the network |

Attributes

| | |
|---------------|--|
| <i>LOGGER</i> | |
|---------------|--|

tyto.tyto.LOGGER

class tyto.tyto.**Ontology**(path=None, endpoints=None, uri=None)

The Ontology class provides an abstraction layer for accessing ontologies, and a back-end query dispatcher for interfacing with RESTful services, SPARQL endpoints, or local triple stores.

Parameters

- **path** (*str*, *optional*) – A path to a local ontology file, defaults to None
- **endpoints** (*list*, *optional*) – A list of zero or more Endpoint objects that provide a query interface to an ontology resource, defaults to None
- **uri** – The URI of the ontology

:type str

__getattr__(self, name)

Enables use of ontology terms as dynamic attributes, e.g., SO.promoter

`_handler(self, method_name, exception, *args)`

Dispatches queries through Endpoints

`get_term_by_uri(self, uri)`

Provides the ontology term (rdfs:label) associated with the given URI.

Parameters **uri** (*str*) – A uniform resource identifier corresponding to an ontology term

Returns A human-readable term or label that corresponds to the given identifier

Return type *string*

`get_uri_by_term(self, term)`

Provides the URI associated with the given ontology term (rdfs:label). The `__getattr__` and `__getitem__` methods delegate to this method.

Parameters **term** (*str*) – an ontology term

Returns A human-readable term or label that corresponds to the given identifier

Return type *URI*

`_sanitize_uri(self, uri)`

Some Ontology instances may override this method to translate a URI from purl to identifiers.org namespaces

`_reverse_sanitize_uri(self, uri)`

Some Ontology instances may override this method to reverse-translate a URI from identifiers.org back into purl namespace

`_sanitize_term(self, term)`

Some Ontology instances may override this method in order to convert a Pythonic representation of a label into a more human-readable representation, such as replacing underscores with spaces

`_reverse_sanitize_term(self, term)`

Some Ontology instances may override this method to undo the conversion done by `_sanitize_term` and return a Pythonic label from free text label

`__getitem__(self, key)`

Enables use of an ontology term as a subscript. This method is a useful alternative to dynamic attributes in case a term contains special characters

Returns A uniform resource identifier associated with the provided term

Return type *URI*

`class tyto.tyto.URI`

Bases: *str*

The URI class wraps the Python string primitive type, enabling the use of inference methods on the represented uniform resource identifier

Parameters

- **value** (*str*) – A string value representing a uniform resource identifier
- **ontology** (*Ontology*) – links a term to a particular Ontology instance

`is_child_of(self, parent_uri: str)`

Determine whether this URI is an immediate subclass or subtype of the argument URI

Parameters **parent_uri** (*str*) – URI corresponding to the putative parent term

Return type bool

is_parent_of(*self*, *child_uri*: str)

Determine whether this URI is an immediate superclass or supertype of the argument URI

Parameters **parent_uri** (str) – URI corresponding to the putative parent term

Return type bool

is_descendant_of(*self*, *ancestor_uri*: str)

Determine whether this URI is a taxonomic subcategory of the argument URI

Parameters **ancestor_uri** (str) – URI corresponding to the putative ancestor

Return type bool

is_ancestor_of(*self*, *descendant_uri*: str)

Determine whether this URI is a taxonomic superclass or supertype of the argument URI

Parameters **descendant_uri** (str) – URI corresponding to the putative descendant

Return type bool

is_subtype_of(*self*, *supertype*: str)

Alias of is_descendant_of. Determines whether this URI is a derivative subclass or subtype of the argument URI

Parameters **supertype** (str) – URI corresponding to the putative supertype

Return type bool

is_supertype_of(*self*, *subtype*: str)

Alias of is_ancestor_of. Determines whether this URI is a superclass or supertype of the argument URI

Parameters **subtype** (str) – URI corresponding to the putative subtype

Return type bool

is_a(*self*, *term*: URI)

get_parents(*self*)

get_children(*self*)

get_ancestors(*self*)

get_descendants(*self*)

class tyto.tyto.Term

Bases: [URI](#)

The URI class wraps the Python string primitive type, enabling the use of inference methods on the represented uniform resource identifier

Parameters

- **value** (str) – A string value representing a uniform resource identifier
- **ontology** ([Ontology](#)) – links a term to a particular Ontology instance

__repr__(*self*)

Return repr(self).

tyto.tyto.installation_path(*relative_path*)

```
tyto.tyto.multi_replace(target_uri, old_namespaces, new_namespace)
```

```
tyto.tyto.configure_cache_size(maxsize=1000)
```

Set the size of the in-memory cache in order to optimize performance and frequency of queries over the network

Parameters `maxsize` (*int*) – The maximum number of cached query results

Package Contents

Classes

| | |
|-----------------|--|
| <i>Ontology</i> | The Ontology class provides an abstraction layer for accessing ontologies, and a |
| <i>URI</i> | The URI class wraps the Python string primitive type, enabling the use of inference methods on the represented uniform resource identifier |

Functions

| | |
|--|--|
| <i>configure_cache_size</i> (maxsize=1000) | Set the size of the in-memory cache in order to optimize performance and frequency of queries over the network |
|--|--|

Attributes

| | |
|---------------------------------|---|
| <i>Ontobee</i> | Endpoint instance representing Ontobee. Ontobee is the default linked data server for most OBO Foundry library ontologies, but is also been used for many non-OBO ontologies. |
| <i>EBIOntologyLookupService</i> | The Ontology Lookup Service (OLS) is a repository for biomedical ontologies that aims to provide a single point of access to the latest ontology versions. Hosted by the European Bioinformatics Institute |
| <i>PubChemAPI</i> | |
| <i>SBO</i> | Ontology instance for Systems Biology Ontology |
| <i>SO</i> | Ontology instance for Sequence Ontology |
| <i>NCIT</i> | Ontology instance for National Cancer Institute Thesaurus |
| <i>OM</i> | Ontology instance for Ontology of Units of Measure |
| <i>NCBITaxon</i> | Ontology instance for NCBI Taxonomy |
| <i>SBOL2</i> | |
| <i>SBOL3</i> | |
| <i>EDAM</i> | EDAM (EMBRACE Data and Methods) is an ontology of common bioinformatics operations, topics, types of data including identifiers, and formats. EDAM comprises common concepts (shared within the bioinformatics community) that apply to semantic annotation of resources. |
| <i>PubChem</i> | |

class tyto.**Ontology**(*path=None, endpoints=None, uri=None*)

The Ontology class provides an abstraction layer for accessing ontologies, and a back-end query dispatcher for interfacing with RESTful services, SPARQL endpoints, or local triple stores.

Parameters

- **path** (*str, optional*) – A path to a local ontology file, defaults to None
- **endpoints** (*list, optional*) – A list of zero or more Endpoint objects that provide a query interface to an ontology resource, defaults to None
- **uri** – The URI of the ontology

:type str

__getattr__(*self, name*)

Enables use of ontology terms as dynamic attributes, e.g., SO.promoter

_handler(*self, method_name, exception, *args*)

Dispatches queries through Endpoints

get_term_by_uri(*self, uri*)

Provides the ontology term (rdfs:label) associated with the given URI.

Parameters **uri** (*str*) – A uniform resource identifier corresponding to an ontology term

Returns A human-readable term or label that corresponds to the given identifier

Return type `string`

get_uri_by_term(*self*, *term*)

Provides the URI associated with the given ontology term (rdfs:label). The `__getattr__` and `__getitem__` methods delegate to this method.

Parameters **term** (*str*) – an ontology term

Returns A human-readable term or label that corresponds to the given identifier

Return type `URI`

_sanitize_uri(*self*, *uri*)

Some Ontology instances may override this method to translate a URI from purl to identifiers.org namespaces

_reverse_sanitize_uri(*self*, *uri*)

Some Ontology instances may override this method to reverse-translate a URI from identifiers.org back into purl namespace

_sanitize_term(*self*, *term*)

Some Ontology instances may override this method in order to convert a Pythonic representation of a label into a more human-readable representation, such as replacing underscores with spaces

_reverse_sanitize_term(*self*, *term*)

Some Ontology instances may override this method to undo the conversion done by `_sanitize_term` and return a Pythonic label from free text label

__getitem__(*self*, *key*)

Enables use of an ontology term as a subscript. This method is a useful alternative to dynamic attributes in case a term contains special characters

Returns A uniform resource identifier associated with the provided term

Return type `URI`

class `tyto.URI`

Bases: `str`

The URI class wraps the Python string primitive type, enabling the use of inference methods on the represented uniform resource identifier

Parameters

- **value** (*str*) – A string value representing a uniform resource identifier
- **ontology** (*Ontology*) – links a term to a particular Ontology instance

is_child_of(*self*, *parent_uri*: *str*)

Determine whether this URI is an immediate subclass or subtype of the argument URI

Parameters **parent_uri** (*str*) – URI corresponding to the putative parent term

Return type `bool`

is_parent_of(*self*, *child_uri*: *str*)

Determine whether this URI is an immediate superclass or supertype of the argument URI

Parameters **parent_uri** (*str*) – URI corresponding to the putative parent term

Return type `bool`

is_descendant_of(*self*, *ancestor_uri*: *str*)

Determine whether this URI is a taxonomic subcategory of the argument URI

Parameters **ancestor_uri** (*str*) – URI corresponding to the putative ancestor

Return type bool

is_ancestor_of(*self*, *descendant_uri*: *str*)

Determine whether this URI is a taxonomic superclass or supertype of the argument URI

Parameters **descendant_uri** (*str*) – URI corresponding to the putative descendant

Return type bool

is_subtype_of(*self*, *supertype*: *str*)

Alias of `is_descendant_of`. Determines whether this URI is a derivative subclass or subtype of the argument URI

Parameters **supertype** (*str*) – URI corresponding to the putative supertype

Return type bool

is_supertype_of(*self*, *subtype*: *str*)

Alias of `is_ancestor_of`. Determines whether this URI is a superclass or supertype of the argument URI

Parameters **subtype** (*str*) – URI corresponding to the putative subtype

Return type bool

is_a(*self*, *term*: [URI](#))

get_parents(*self*)

get_children(*self*)

get_ancestors(*self*)

get_descendants(*self*)

tyto.configure_cache_size(*maxsize*=1000)

Set the size of the in-memory cache in order to optimize performance and frequency of queries over the network

Parameters **maxsize** (*int*) – The maximum number of cached query results

tyto.Ontobee

Endpoint instance representing Ontobee. Ontobee is the default linked data server for most OBO Foundry library ontologies, but is also been used for many non-OBO ontologies.

tyto.EBIOntologyLookupService

The Ontology Lookup Service (OLS) is a repository for biomedical ontologies that aims to provide a single point of access to the latest ontology versions. Hosted by the European Bioinformatics Institute

tyto.PubChemAPI

tyto.SBO

Ontology instance for Systems Biology Ontology

tyto.SO

Ontology instance for Sequence Ontology

tyto.NCIT

Ontology instance for National Cancer Institute Thesaurus

tyto.**OM**

Ontology instance for Ontology of Units of Measure

tyto.**NCBITaxon**

Ontology instance for NCBI Taxonomy

tyto.**SBOL2**

tyto.**SBOL3**

tyto.**EDAM**

EDAM (EMBRACE Data and Methods) is an ontology of common bioinformatics operations, topics, types of data including identifiers, and formats. EDAM comprises common concepts (shared within the bioinformatics community) that apply to semantic annotation of resources.

tyto.**PubChem**

6.1.2 App

Created on 26 May 2021

@author: gokselmisirli

Module Contents

Classes

Measure

Identified

Attributes

sbol3

prov

om

App.**sbol3**

App.**prov**

App.**om**

class App.**Measure**

Bases: Thing

label = Measure

```
class App.Identified
  Bases: Thing
  label = Identified
```

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

a

App, [29](#)

t

tyto, [13](#)

tyto.edam, [20](#)

tyto.endpoint, [13](#)

tyto.endpoint.endpoint, [13](#)

tyto.ncbi_taxon, [20](#)

tyto.ncit, [20](#)

tyto.om, [20](#)

tyto.pubchem, [21](#)

tyto.sbo, [21](#)

tyto.sbol2, [21](#)

tyto.sbol3, [21](#)

tyto.so, [21](#)

tyto.tyto, [22](#)

Symbols

[__getattr__\(\) \(tyto.Ontology method\), 26](#)
[__getattr__\(\) \(tyto.tyto.Ontology method\), 22](#)
[__getitem__\(\) \(tyto.Ontology method\), 27](#)
[__getitem__\(\) \(tyto.tyto.Ontology method\), 23](#)
[__repr__\(\) \(tyto.tyto.Term method\), 24](#)
[_get_request\(\) \(tyto.endpoint.EBIOntologyLookupServiceAPI module\), 19](#)
[_get_request\(\) \(tyto.endpoint.RESTEndpoint method\), 18](#)
[_get_request\(\) \(tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method\), 16](#)
[_get_request\(\) \(tyto.endpoint.endpoint.RESTEndpoint method\), 15](#)
[_handler\(\) \(tyto.Ontology method\), 26](#)
[_handler\(\) \(tyto.tyto.Ontology method\), 22](#)
[_load_ontology_ids\(\) \(tyto.endpoint.EBIOntologyLookupServiceAPI method\), 19](#)
[_load_ontology_ids\(\) \(tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method\), 16](#)
[_reverse_sanitize_term \(in module tyto.om\), 20](#)
[_reverse_sanitize_term\(\) \(tyto.Ontology method\), 27](#)
[_reverse_sanitize_term\(\) \(tyto.tyto.Ontology method\), 23](#)
[_reverse_sanitize_uri \(in module tyto.ncbi_taxon\), 20](#)
[_reverse_sanitize_uri \(in module tyto.ncit\), 20](#)
[_reverse_sanitize_uri \(in module tyto.sbo\), 21](#)
[_reverse_sanitize_uri \(in module tyto.so\), 21](#)
[_reverse_sanitize_uri\(\) \(tyto.Ontology method\), 27](#)
[_reverse_sanitize_uri\(\) \(tyto.tyto.Ontology method\), 23](#)
[_sanitize_term \(in module tyto.om\), 20](#)
[_sanitize_term \(in module tyto.so\), 21](#)
[_sanitize_term\(\) \(tyto.Ontology method\), 27](#)
[_sanitize_term\(\) \(tyto.tyto.Ontology method\), 23](#)
[_sanitize_uri \(in module tyto.ncbi_taxon\), 20](#)
[_sanitize_uri \(in module tyto.ncit\), 20](#)
[_sanitize_uri \(in module tyto.sbo\), 21](#)

[_sanitize_uri \(in module tyto.so\), 21](#)
[_sanitize_uri\(\) \(tyto.Ontology method\), 27](#)
[_sanitize_uri\(\) \(tyto.tyto.Ontology method\), 23](#)

A

App

module, 29

C

[configure_cache_size\(\) \(in module tyto\), 28](#)
[configure_cache_size\(\) \(in module tyto.tyto\), 25](#)
[convert\(\) \(tyto.endpoint.EBIOntologyLookupServiceAPI method\), 19](#)
[convert\(\) \(tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method\), 16](#)
[convert\(\) \(tyto.endpoint.endpoint.GraphEndpoint method\), 15](#)
[convert\(\) \(tyto.endpoint.endpoint.SPARQLEndpoint method\), 15](#)
[convert\(\) \(tyto.endpoint.GraphEndpoint method\), 18](#)
[convert\(\) \(tyto.endpoint.SPARQLEndpoint method\), 18](#)

E

[EBIOntologyLookupService \(in module tyto\), 28](#)
[EBIOntologyLookupService \(in module tyto.endpoint\), 19](#)
[EBIOntologyLookupService \(in module tyto.endpoint.endpoint\), 16](#)
[EBIOntologyLookupServiceAPI \(class in tyto.endpoint\), 18](#)
[EBIOntologyLookupServiceAPI \(class in tyto.endpoint.endpoint\), 15](#)
[EDAM \(in module tyto\), 29](#)
[EDAM \(in module tyto.edam\), 20](#)
[Endpoint \(class in tyto.endpoint\), 18](#)
[Endpoint \(class in tyto.endpoint.endpoint\), 15](#)

G

[get_ancestors\(\) \(tyto.endpoint.EBIOntologyLookupServiceAPI method\), 19](#)
[get_ancestors\(\) \(tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method\), 16](#)

[get_ancestors\(\)](#) (*tyto.tyto.URI method*), 24
[get_ancestors\(\)](#) (*tyto.URI method*), 28
[get_children\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19
[get_children\(\)](#) (*tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method*), 16
[get_children\(\)](#) (*tyto.tyto.URI method*), 24
[get_children\(\)](#) (*tyto.URI method*), 28
[get_descendants\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19
[get_descendants\(\)](#) (*tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method*), 16
[get_descendants\(\)](#) (*tyto.tyto.URI method*), 24
[get_descendants\(\)](#) (*tyto.URI method*), 28
[get_ontologies\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19
[get_ontologies\(\)](#) (*tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method*), 16
[get_ontologies\(\)](#) (*tyto.endpoint.endpoint.SPARQLBuilder method*), 15
[get_ontologies\(\)](#) (*tyto.endpoint.SPARQLBuilder method*), 18
[get_parents\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19
[get_parents\(\)](#) (*tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method*), 16
[get_parents\(\)](#) (*tyto.tyto.URI method*), 24
[get_parents\(\)](#) (*tyto.URI method*), 28
[get_term_by_uri\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19
[get_term_by_uri\(\)](#) (*tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method*), 16
[get_term_by_uri\(\)](#) (*tyto.endpoint.endpoint.PUG_REST method*), 16
[get_term_by_uri\(\)](#) (*tyto.endpoint.endpoint.QueryBackend method*), 14
[get_term_by_uri\(\)](#) (*tyto.endpoint.endpoint.SPARQLBuilder method*), 14
[get_term_by_uri\(\)](#) (*tyto.endpoint.PUG_REST method*), 19
[get_term_by_uri\(\)](#) (*tyto.endpoint.QueryBackend method*), 17
[get_term_by_uri\(\)](#) (*tyto.endpoint.SPARQLBuilder method*), 17
[get_term_by_uri\(\)](#) (*tyto.Ontology method*), 26
[get_term_by_uri\(\)](#) (*tyto.tyto.Ontology method*), 23
[get_uri_by_term\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19
[get_uri_by_term\(\)](#) (*tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method*), 16
[get_uri_by_term\(\)](#) (*tyto.endpoint.endpoint.PUG_REST method*), 16
[get_uri_by_term\(\)](#) (*tyto.endpoint.endpoint.QueryBackend method*), 14
[get_uri_by_term\(\)](#) (*tyto.endpoint.endpoint.SPARQLBuilder method*), 14
[get_uri_by_term\(\)](#) (*tyto.endpoint.PUG_REST method*), 19
[get_uri_by_term\(\)](#) (*tyto.endpoint.QueryBackend method*), 17
[get_uri_by_term\(\)](#) (*tyto.endpoint.SPARQLBuilder method*), 18
[get_uri_by_term\(\)](#) (*tyto.Ontology method*), 27
[get_uri_by_term\(\)](#) (*tyto.tyto.Ontology method*), 23
[GraphEndpoint](#) (class in *tyto.endpoint*), 18
[GraphEndpoint](#) (class in *tyto.endpoint.endpoint*), 15
[I](#)
[Identified](#) (class in *App*), 29
[installation_path\(\)](#) (in module *tyto.tyto*), 24
[lookup\(\)](#) (*tyto.tyto.URI method*), 24
[is_a\(\)](#) (*tyto.URI method*), 28
[is_ancestor_of\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19
[is_ancestor_of\(\)](#) (*tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method*), 16
[is_ancestor_of\(\)](#) (*tyto.endpoint.endpoint.SPARQLBuilder method*), 15
[is_ancestor_of\(\)](#) (*tyto.endpoint.SPARQLBuilder method*), 18
[is_ancestor_of\(\)](#) (*tyto.tyto.URI method*), 24
[is_ancestor_of\(\)](#) (*tyto.URI method*), 28
[is_child_of\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19
[is_child_of\(\)](#) (*tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method*), 16
[is_child_of\(\)](#) (*tyto.endpoint.endpoint.SPARQLBuilder method*), 15
[is_child_of\(\)](#) (*tyto.endpoint.SPARQLBuilder method*), 18
[is_child_of\(\)](#) (*tyto.tyto.URI method*), 23
[is_child_of\(\)](#) (*tyto.URI method*), 27
[is_descendant_of\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19
[is_descendant_of\(\)](#) (*tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI method*), 16
[is_descendant_of\(\)](#) (*tyto.endpoint.endpoint.SPARQLBuilder method*), 15
[is_descendant_of\(\)](#) (*tyto.endpoint.SPARQLBuilder method*), 18
[is_descendant_of\(\)](#) (*tyto.tyto.URI method*), 24
[is_descendant_of\(\)](#) (*tyto.URI method*), 27
[is_loaded\(\)](#) (*tyto.endpoint.endpoint.GraphEndpoint method*), 15
[is_loaded\(\)](#) (*tyto.endpoint.GraphEndpoint method*), 18
[is_parent_of\(\)](#) (*tyto.endpoint.EBIOntologyLookupServiceAPI method*), 19

- `is_parent_of()` (`tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI` method), 16
- `is_parent_of()` (`tyto.endpoint.endpoint.SPARQLBuilder` method), 15
- `is_parent_of()` (`tyto.endpoint.SPARQLBuilder` method), 18
- `is_parent_of()` (`tyto.tyto.URI` method), 24
- `is_parent_of()` (`tyto.URI` method), 27
- `is_subtype_of()` (`tyto.tyto.URI` method), 24
- `is_subtype_of()` (`tyto.URI` method), 28
- `is_supertype_of()` (`tyto.tyto.URI` method), 24
- `is_supertype_of()` (`tyto.URI` method), 28
- ## L
- `label` (`App`.Identified attribute), 30
- `label` (`App`.Measure attribute), 29
- `load()` (`tyto.endpoint.endpoint.GraphEndpoint` method), 15
- `load()` (`tyto.endpoint.GraphEndpoint` method), 18
- `LOGGER` (in module `tyto.tyto`), 22
- ## M
- `Measure` (class in `App`), 29
- module
- `App`, 29
 - `tyto`, 13
 - `tyto.edam`, 20
 - `tyto.endpoint`, 13
 - `tyto.endpoint.endpoint`, 13
 - `tyto.ncbi_taxon`, 20
 - `tyto.ncit`, 20
 - `tyto.om`, 20
 - `tyto.pubchem`, 21
 - `tyto.sbo`, 21
 - `tyto.sbol2`, 21
 - `tyto.sbol3`, 21
 - `tyto.so`, 21
 - `tyto.tyto`, 22
- `multi_replace()` (in module `tyto.tyto`), 24
- ## N
- `NCBITaxon` (in module `tyto`), 29
- `NCBITaxon` (in module `tyto.ncbi_taxon`), 20
- `NCIT` (in module `tyto`), 28
- `NCIT` (in module `tyto.ncit`), 20
- ## O
- `om` (in module `App`), 29
- `OM` (in module `tyto`), 28
- `OM` (in module `tyto.om`), 20
- `Ontobee` (in module `tyto`), 28
- `Ontobee` (in module `tyto.endpoint`), 19
- `Ontobee` (in module `tyto.endpoint.endpoint`), 16
- `OntobeeEndpoint` (class in `tyto.endpoint`), 18
- `OntobeeEndpoint` (class in `tyto.endpoint.endpoint`), 15
- `Ontology` (class in `tyto`), 26
- `Ontology` (class in `tyto.tyto`), 22
- ## P
- `prov` (in module `App`), 29
- `PubChem` (in module `tyto`), 29
- `PubChem` (in module `tyto.pubchem`), 21
- `PubChemAPI` (in module `tyto`), 28
- `PubChemAPI` (in module `tyto.endpoint`), 19
- `PubChemAPI` (in module `tyto.endpoint.endpoint`), 16
- `PUG_REST` (class in `tyto.endpoint`), 19
- `PUG_REST` (class in `tyto.endpoint.endpoint`), 16
- ## Q
- `query()` (`tyto.endpoint.EBIOntologyLookupServiceAPI` method), 19
- `query()` (`tyto.endpoint.endpoint.EBIOntologyLookupServiceAPI` method), 16
- `query()` (`tyto.endpoint.endpoint.GraphEndpoint` method), 15
- `query()` (`tyto.endpoint.endpoint.OntobeeEndpoint` method), 15
- `query()` (`tyto.endpoint.endpoint.SPARQLEndpoint` method), 15
- `query()` (`tyto.endpoint.GraphEndpoint` method), 18
- `query()` (`tyto.endpoint.OntobeeEndpoint` method), 18
- `query()` (`tyto.endpoint.SPARQLEndpoint` method), 18
- `QueryBackend` (class in `tyto.endpoint`), 17
- `QueryBackend` (class in `tyto.endpoint.endpoint`), 14
- ## R
- `RESTEndpoint` (class in `tyto.endpoint`), 18
- `RESTEndpoint` (class in `tyto.endpoint.endpoint`), 15
- ## S
- `SBO` (in module `tyto`), 28
- `SBO` (in module `tyto.sbo`), 21
- `SBOL2` (in module `tyto`), 29
- `SBOL2` (in module `tyto.sbol2`), 21
- `sbo13` (in module `App`), 29
- `SBOL3` (in module `tyto`), 29
- `SBOL3` (in module `tyto.sbol3`), 21
- `SO` (in module `tyto`), 28
- `SO` (in module `tyto.so`), 21
- `SPARQLBuilder` (class in `tyto.endpoint`), 17
- `SPARQLBuilder` (class in `tyto.endpoint.endpoint`), 14
- `SPARQLEndpoint` (class in `tyto.endpoint`), 18
- `SPARQLEndpoint` (class in `tyto.endpoint.endpoint`), 15
- ## T
- `Term` (class in `tyto.tyto`), 24

tyto
 module, 13
tyto.edam
 module, 20
tyto.endpoint
 module, 13
tyto.endpoint.endpoint
 module, 13
tyto.ncbi_taxon
 module, 20
tyto.ncit
 module, 20
tyto.om
 module, 20
tyto.pubchem
 module, 21
tyto.sbo
 module, 21
tyto.sbol2
 module, 21
tyto.sbol3
 module, 21
tyto.so
 module, 21
tyto.tyto
 module, 22

U

URI (*class in tyto*), 27
URI (*class in tyto.tyto*), 23