
Schema Salad

Peter Amstutz and Common Workflow Language Project contributors

Aug 08, 2023

CONTENTS

1	Installation	3
2	Commands	5
3	Codegen Examples	7
4	Quick Start	9
5	Documentation	11
6	Rationale	13
6.1	Command Line Options	13
6.2	How to add new types to the local Typeshed	16
6.3	API Reference	16
6.4	Indices and tables	142
	Python Module Index	143
	Index	145

Salad is a schema language for describing JSON or YAML structured linked data documents. Salad schema describes rules for preprocessing, structural validation, and hyperlink checking for documents described by a Salad schema. Salad supports rich data modeling with inheritance, template specialization, object identifiers, object references, documentation generation, code generation, and transformation to [RDF](#). Salad provides a bridge between document and record oriented data modeling and the Semantic Web.

The Schema Salad library is Python 3.6+ only.

INSTALLATION

```
pip3 install schema_salad
```

If you intend to use the *schema-salad-tool --codegen=python* feature, please include the *[pycodegen]* extra:

```
pip3 install schema_salad[pycodegen]
```

To install from source:

```
git clone https://github.com/common-workflow-language/schema_salad
cd schema_salad
pip3 install .
# or pip3 install .[pycodegen] if needed
```


COMMANDS

Schema salad can be used as a command line tool or imported as a Python module:

```
$ schema-salad-tool
usage: schema-salad-tool [-h] [--rdf-serializer RDF_SERIALIZER] [--skip-schemas]
                        [--strict-foreign-properties] [--print-jsonld-context]
                        [--print-rdfs] [--print-avro] [--print-rdf] [--print-pre]
                        [--print-index] [--print-metadata] [--print-inheritance-dot]
                        [--print-fieldrefs-dot] [--codegen language] [--codegen-target
↳ CODEGEN_TARGET]
                        [--codegen-examples directory] [--codegen-package dotted.package]
                        [--codegen-copyright copyright_string] [--print-oneline]
                        [--print-doc] [--strict | --non-strict]
                        [--verbose | --quiet | --debug] [--only ONLY] [--redirect REDIRECT]
                        [--brand BRAND] [--brandlink BRANDLINK] [--brandstyle BRANDSTYLE]
                        [--brandinverse] [--primetype PRIMTYPE] [--version]
                        [schema] [document]

$ python
>>> import schema_salad
```

Validate a schema:

```
$ schema-salad-tool myschema.yml
```

Validate a document using a schema:

```
$ schema-salad-tool myschema.yml mydocument.yml
```

Generate HTML documentation:

```
$ schema-salad-tool --print-doc myschema.yml > myschema.html
$ # or
$ schema-salad-doc myschema.yml > myschema.html
```

Get JSON-LD context:

```
$ schema-salad-tool --print-jsonld-context myschema.yml mydocument.yml
```

Convert a document to JSON-LD:

```
$ schema-salad-tool --print-pre myschema.yml mydocument.yml > mydocument.jsonld
```

Schema Salad

Generate Python classes for loading/generating documents described by the schema (Requires the *[pycodegen]* extra):

```
$ schema-salad-tool --codegen=python myschema.yml > myschema.py
```

Display inheritance relationship between classes as a graphviz 'dot' file and render as SVG:

```
$ schema-salad-tool --print-inheritance-dot myschema.yml | dot -Tsvg > myschema.svg
```

CODEGEN EXAMPLES

These are helpful to see how to use the output of *schema-salad-tool --codegen* in different languages for loading and/or creating/editing/saving objects, using the [CWL v1.2 schema](#) as an example.

Language	Repository	Serialization Example Deserialization Example	
Python	https://github.com/common-workflow-language/cwl-utils/	<code>create_cwl_from_objects.py</code>	<code>load_document()</code>
Java	https://github.com/common-workflow-language/cwljava/	(Not yet implemented)	<code>PackedWorkflow-ClassTest.java</code>
Type-Script	https://github.com/common-workflow-lab/cwl-ts-auto	Creating, editing, and saving CWL docs with TypeScript	Loading CWL documents with TypeScript
.Net	https://github.com/common-workflow-lab/CWLDotNet	Creating, editing, and saving CWL docs with .Net	Loading CWL documents with .Net
C++	https://github.com/common-workflow-lab/cwl-cpp-auto	<code>cwl_output_example.cpp</code>	(Not yet implemented)
D	https://github.com/common-workflow-lab/cwl-d-auto	How to use	How to use

QUICK START

Let's say you have a 'basket' record that can contain items measured either by weight or by count. Here's an example:

```
basket:
- product: bananas
  price: 0.39
  per: pound
  weight: 1
- product: cucumbers
  price: 0.79
  per: item
  count: 3
```

We want to validate that all the expected fields are present, the measurement is known, and that "count" cannot be a fractional value. Here is an example schema to do that:

```
- name: Product
  doc: |
    The base type for a product. This is an abstract type, so it
    can't be used directly, but can be used to define other types.
  type: record
  abstract: true
  fields:
    product: string
    price: float

- name: ByWeight
  doc: |
    A product, sold by weight. Products may be sold by pound or by
    kilogram. Weights may be fractional.
  type: record
  extends: Product
  fields:
    per:
      type:
        type: enum
        symbols:
          - pound
          - kilogram
      jsonldPredicate: '#per'
    weight: float
```

(continues on next page)

```
- name: ByCount
  doc: |
    A product, sold by count. The count must be a integer value.
  type: record
  extends: Product
  fields:
    per:
      type:
        type: enum
        symbols:
          - item
      jsonldPredicate: '#per'
    count: int

- name: Basket
  doc: |
    A basket of products. The 'documentRoot' field indicates it is a
    valid starting point for a document. The 'basket' field will
    validate subtypes of 'Product' (ByWeight and ByCount).
  type: record
  documentRoot: true
  fields:
    basket:
      type:
        type: array
        items: Product
```

You can check the schema and document in `schema_salad/tests/basket_schema.yml` and `schema_salad/tests/basket.yml`:

```
$ schema-salad-tool basket_schema.yml basket.yml
Document `basket.yml` is valid
```

DOCUMENTATION

See the [specification](#) and the [metaschema](#) (salad schema for itself). For an example application of Schema Salad see the [Common Workflow Language](#).

RATIONALE

The JSON data model is a popular way to represent structured data. It is attractive because of its relative simplicity and is a natural fit with the standard types of many programming languages. However, this simplicity comes at the cost that basic JSON lacks expressive features useful for working with complex data structures and document formats, such as schemas, object references, and namespaces.

JSON-LD is a W3C standard providing a way to describe how to interpret a JSON document as Linked Data by means of a “context”. JSON-LD provides a powerful solution for representing object references and namespaces in JSON based on standard web URIs, but is not itself a schema language. Without a schema providing a well defined structure, it is difficult to process an arbitrary JSON-LD document as idiomatic JSON because there are many ways to express the same data that are logically equivalent but structurally distinct.

Several schema languages exist for describing and validating JSON data, such as JSON Schema and Apache Avro data serialization system, however none understand linked data. As a result, to fully take advantage of JSON-LD to build the next generation of linked data applications, one must maintain separate JSON schema, JSON-LD context, RDF schema, and human documentation, despite significant overlap of content and obvious need for these documents to stay synchronized.

Schema Salad is designed to address this gap. It provides a schema language and processing rules for describing structured JSON content permitting URI resolution and strict document validation. The schema language supports linked data through annotations that describe the linked data interpretation of the content, enables generation of JSON-LD context and RDF schema, and production of RDF triples by applying the JSON-LD context. The schema language also provides for robust support of inline documentation.

6.1 Command Line Options

6.1.1 schema-salad-tool

```
usage: schema-salad-tool [-h] [--rdf-serializer RDF_SERIALIZER]
                        [--skip-schemas] [--strict-foreign-properties]
                        [--print-jsonld-context] [--print-rdfs]
                        [--print-avro] [--print-rdf] [--print-pre]
                        [--print-index] [--print-metadata]
                        [--print-inheritance-dot] [--print-fieldrefs-dot]
                        [--codegen language]
                        [--codegen-target CODEGEN_TARGET]
                        [--codegen-examples directory]
                        [--codegen-package dotted.package]
                        [--codegen-copyright copyright_string]
                        [--codegen-parser-info parser_info] [--print-online]
```

(continues on next page)

```
[--print-doc] [--strict | --non-strict]
[--verbose | --quiet | --debug] [--only ONLY]
[--redirect REDIRECT] [--brand BRAND]
[--brandlink BRANDLINK] [--brandstyle BRANDSTYLE]
[--brandinverse] [--primetype PRIMTYPE] [--version]
[schema] [document ...]
```

schema**document****-h, --help**

show this help message and exit

--rdf-serializer <rdf_serializer>

Output RDF serialization format used by `--print-rdf`(one of turtle (default), n3, nt, xml)

--skip-schemas

If specified, ignore \$schemas sections.

--strict-foreign-properties

Strict checking of foreign properties

--print-jsonld-context

Print JSON-LD context for schema

--print-rdfs

Print RDF schema

--print-avro

Print Avro schema

--print-rdf

Print corresponding RDF graph for document

--print-pre

Print document after preprocessing

--print-index

Print node index

--print-metadata

Print document metadata

--print-inheritance-dot

Print graphviz file of inheritance

--print-fieldrefs-dot

Print graphviz file of field refs

--codegen <language>

Generate classes in target language, currently supported: python, java, typescript, dotnet, cpp, dlang

--codegen-target <codegen_target>

Defaults to sys.stdout for Python/C++/Dlang and ./ for Java/TypeScript/.Net

-
- codegen-examples** <directory>
Directory of example documents for test case generation (Java/TypeScript/.Net/Dlang only).
 - codegen-package** <dotted.package>
Optional override of the package name which is other derived from the base URL (Java/TypeScript/.Net/Dlang only).
 - codegen-copyright** <copyright_string>
Optional copyright of the input schema.
 - codegen-parser-info** <parser_info>
Optional parser name which is accessible via resulted parser API (Python and Dlang only)
 - print-oneline**
Print each error message in oneline
 - print-doc**
Print HTML schema documentation page
 - strict**
Strict validation (unrecognized or out of place fields are error)
 - non-strict**
Lenient validation (ignore unrecognized fields)
 - verbose**
Default logging
 - quiet**
Only print warnings and errors.
 - debug**
Print even more logging
 - only** <only>
Use with `-print-doc`, document only listed types
 - redirect** <redirect>
Use with `-print-doc`, override default link for type
 - brand** <brand>
Use with `-print-doc`, set the 'brand' text in nav bar
 - brandlink** <brandlink>
Use with `-print-doc`, set the link for 'brand' in nav bar
 - brandstyle** <brandstyle>
Use with `-print-doc`, HTML code to link to an external style sheet
 - brandinverse**
Use with `-print-doc`
 - primetype** <primetype>
Use with `-print-doc`, link to use for primitive types (string, int etc)
 - version, -v**
Print version

6.1.2 schema-salad-doc

```
usage: schema-salad-doc [-h] [--only ONLY] [--redirect REDIRECT]
                        [--brand BRAND] [--brandlink BRANDLINK]
                        [--brandstyle BRANDSTYLE] [--brandinverse]
                        [--primetype PRIMETYPE] [--debug]
                        schema
```

schema

-h, --help

show this help message and exit

--only <only>

--redirect <redirect>

--brand <brand>

--brandlink <brandlink>

--brandstyle <brandstyle>

--brandinverse

--primetype <primetype>

--debug

6.2 How to add new types to the local Typedshed

If when running `make mypy` you receive errors about modules that can't be found you may need to add type stubs for new modules to the `mypy-stubs/` directory.

```
stubgen -o mypy-stubs module_name
make mypy
```

Note: the module name is not always the name of the PyPI package (`CacheControl` vs `cachecontrol`).

Stubs are just that, you will still need to annotate whichever functions you call.

Oftentimes it is simpler to comment out imports in the `.pyi` stubs that are not needed yet. The goal is represent the public API, or at least the part we use.

6.3 API Reference

This page contains auto-generated API reference documentation¹.

¹ Created with `sphinx-autoapi`

6.3.1 schema_salad

A schema language for describing JSON or YAML structured linked data documents.

Subpackages

`schema_salad.avro`

Submodules

`schema_salad.avro.schema`

Contains the Schema classes.

A schema may be one of:

A record, mapping field names to field value data; An enum, containing one of a small set of symbols; An array of values, all of the same schema; A union of other schemas; A unicode string; A 32-bit signed int; A 64-bit signed long; A 32-bit floating-point float; A 64-bit floating-point double; A boolean; or Null.

Module Contents

Classes

<i>Schema</i>	Base class for all Schema classes.
<i>Name</i>	Class to describe Avro name.
<i>Names</i>	Track name set and default namespace during parsing.
<i>NamedSchema</i>	Named Schemas specified in NAMED_TYPES.
<i>Field</i>	
<i>PrimitiveSchema</i>	Valid primitive types are in PRIMITIVE_TYPES.
<i>EnumSchema</i>	Named Schemas specified in NAMED_TYPES.
<i>ArraySchema</i>	Base class for all Schema classes.
<i>UnionSchema</i>	Base class for all Schema classes.
<i>RecordSchema</i>	Named Schemas specified in NAMED_TYPES.

Functions

<i>get_other_props</i> (all_props, reserved_props)	Retrieve the non-reserved properties from a dictionary of properties.
<i>make_avsc_object</i> (json_data[, names])	Build Avro Schema from data parsed out of JSON string.
<i>is_subtype</i> (existing, new)	Check if a new type specification is compatible with an existing type spec.

Attributes

PRIMITIVE_TYPES

NAMED_TYPES

VALID_TYPES

SCHEMA_RESERVED_PROPS

JsonDataType

AtomicPropType

PropType

PropsType

FIELD_RESERVED_PROPS

VALID_FIELD_SORT_ORDERS

```
schema_salad.avro.schema.PRIMITIVE_TYPES = ('null', 'boolean', 'string', 'int', 'long',  
'float', 'double')
```

```
schema_salad.avro.schema.NAMED_TYPES = ('enum', 'record')
```

```
schema_salad.avro.schema.VALID_TYPES
```

```
schema_salad.avro.schema.SCHEMA_RESERVED_PROPS = ('type', 'name', 'namespace', 'fields',  
'items', 'symbols', 'doc')
```

```
schema_salad.avro.schema.JsonDataType
```

```
schema_salad.avro.schema.AtomicPropType
```

```
schema_salad.avro.schema.PropType
```

```
schema_salad.avro.schema.PropsType
```

```
schema_salad.avro.schema.FIELD_RESERVED_PROPS = ('default', 'name', 'doc', 'order',  
'type')
```

```
schema_salad.avro.schema.VALID_FIELD_SORT_ORDERS = ('ascending', 'descending', 'ignore')
```

```
exception schema_salad.avro.schema.AvroException(msg, sl=None, children=None,  
bullet_for_children="")
```

Bases: *schema_salad.exceptions.SchemaException*

Indicates error with the provided schema definition.

Parameters

- **msg** (*str*) –
- **sl** (*Optional*[*schema_salad.sourceline.SourceLine*]) –
- **children** (*Optional*[*Sequence*[*SchemaSaladException*]]) –
- **bullet_for_children** (*str*) –

exception `schema_salad.avro.schema.SchemaParseException`(*msg*, *sl=None*, *children=None*, *bullet_for_children=""*)

Bases: *AvroException*

Indicates error with the provided schema definition.

Parameters

- **msg** (*str*) –
- **sl** (*Optional*[*schema_salad.sourceline.SourceLine*]) –
- **children** (*Optional*[*Sequence*[*SchemaSaladException*]]) –
- **bullet_for_children** (*str*) –

class `schema_salad.avro.schema.Schema`(*atype*, *other_props=None*)

Base class for all Schema classes.

Parameters

- **atype** (*str*) –
- **other_props** (*Optional*[*PropsType*]) –

property props: *PropsType*

Return type
PropsType

get_prop(*key*)

Parameters
key (*str*) –

Return type
Optional[*PropType*]

set_prop(*key*, *value*)

Parameters

- **key** (*str*) –
- **value** (*Optional*[*PropType*]) –

Return type
None

class `schema_salad.avro.schema.Name`(*name_attr=None*, *space_attr=None*, *default_space=None*)

Class to describe Avro name.

Parameters

- **name_attr** (*Optional*[*str*]) –
- **space_attr** (*Optional*[*str*]) –

- **default_space** (*Optional[str]*) –

property fullname: `str | None`

Return type
Optional[str]

get_space()
 Back out a namespace from full name.

Return type
Optional[str]

class `schema_salad.avro.schema.Names`(*default_namespace=None*)

Track name set and default namespace during parsing.

Parameters
default_namespace (*Optional[str]*) –

has_name(*name_attr, space_attr*)

Parameters

- **name_attr** (*str*) –
- **space_attr** (*Optional[str]*) –

Return type
bool

get_name(*name_attr, space_attr*)
 Fetch the stored schema for the given namespace.

Parameters

- **name_attr** (*str*) –
- **space_attr** (*Optional[str]*) –

Return type
Optional[NamedSchema]

add_name(*name_attr, space_attr, new_schema*)
 Add a new schema object to the name set.

Parameters

- **name_attr** (*str*) – name value read in schema
- **space_attr** (*Optional[str]*) – namespace value read in schema.
- **new_schema** (*NamedSchema*) –

Returns
 the Name that was just added.

Return type
Name

class `schema_salad.avro.schema.NamedSchema`(*atype, name, namespace=None, names=None, other_props=None*)

Bases: *Schema*

Named Schemas specified in NAMED_TYPES.

Parameters

- **atype** (*str*) –
- **name** (*str*) –
- **namespace** (*Optional[str]*) –
- **names** (*Optional[Names]*) –
- **other_props** (*Optional[PropsType]*) –

property name: **str**

Return type

str

```
class schema_salad.avro.schema.Field(atype, name, has_default, default=None, order=None, names=None,
                                     doc=None, other_props=None)
```

Parameters

- **atype** (*JsonDataType*) –
- **name** (*str*) –
- **has_default** (*bool*) –
- **default** (*Optional[Any]*) –
- **order** (*Optional[str]*) –
- **names** (*Optional[Names]*) –
- **doc** (*Optional[Union[str, List[str]]]*) –
- **other_props** (*Optional[PropsType]*) –

property default: **Any | None**

Return type

Optional[Any]

get_prop(*key*)

Parameters

key (*str*) –

Return type

Optional[PropType]

set_prop(*key, value*)

Parameters

- **key** (*str*) –
- **value** (*Optional[PropType]*) –

Return type

None

```
class schema_salad.avro.schema.PrimitiveSchema(atype, other_props=None)
```

Bases: *Schema*

Valid primitive types are in `PRIMITIVE_TYPES`.

Parameters

- **atype** (*str*) –
- **other_props** (*Optional[PropsType]*) –

```
class schema_salad.avro.schema.EnumSchema(name, namespace, symbols, names=None, doc=None,
                                          other_props=None)
```

Bases: *NamedSchema*

Named Schemas specified in NAMED_TYPES.

Parameters

- **name** (*str*) –
- **namespace** (*Optional[str]*) –
- **symbols** (*List[str]*) –
- **names** (*Optional[Names]*) –
- **doc** (*Optional[Union[str, List[str]]]*) –
- **other_props** (*Optional[PropsType]*) –

property symbols: **List[str]**

Return type

List[str]

```
class schema_salad.avro.schema.ArraySchema(items, names, other_props=None)
```

Bases: *Schema*

Base class for all Schema classes.

Parameters

- **items** (*JsonDataType*) –
- **names** (*Names*) –
- **other_props** (*Optional[PropsType]*) –

property items: *Schema*

Return type

Schema

```
class schema_salad.avro.schema.UnionSchema(schemas, names)
```

Bases: *Schema*

Base class for all Schema classes.

Parameters

- **schemas** (*List[JsonDataType]*) –
- **names** (*Names*) –

property schemas: `List[Schema]`

Return type

`List[Schema]`

class `schema_salad.avro.schema.RecordSchema`(*name, namespace, fields, names, schema_type='record', doc=None, other_props=None*)

Bases: `NamedSchema`

Named Schemas specified in NAMED_TYPES.

Parameters

- **name** (*str*) –
- **namespace** (*Optional[str]*) –
- **fields** (*List[PropsType]*) –
- **names** (*Names*) –
- **schema_type** (*str*) –
- **doc** (*Optional[Union[str, List[str]]]*) –
- **other_props** (*Optional[PropsType]*) –

property fields: `List[Field]`

Return type

`List[Field]`

static `make_field_objects`(*field_data, names*)

We're going to need to make message parameters too.

Parameters

- **field_data** (*List[PropsType]*) –
- **names** (*Names*) –

Return type

`List[Field]`

`schema_salad.avro.schema.get_other_props`(*all_props, reserved_props*)

Retrieve the non-reserved properties from a dictionary of properties.

Parameters

- **reserved_props** (*Tuple[str, Ellipsis]*) – The set of reserved properties to exclude
- **all_props** (*PropsType*) –

Return type

`Optional[PropsType]`

`schema_salad.avro.schema.make_avsc_object`(*json_data, names=None*)

Build Avro Schema from data parsed out of JSON string.

Parameters

- **names** (*Optional[Names]*) – A Name object (tracks seen names and default space)
- **json_data** (*JsonDataType*) –

Return type

Schema

`schema_salad.avro.schema.is_subtype(existing, new)`

Check if a new type specification is compatible with an existing type spec.

Parameters

- **existing** (*PropType*) –
- **new** (*PropType*) –

Return type

bool

`schema_salad.tests`

Submodules

`schema_salad.tests.conftest`

Module Contents

Functions

isolated_cache()

Clear the `schema_salad` metaschema cache.

`schema_salad.tests.conftest.isolated_cache()`

Clear the `schema_salad` metaschema cache.

Auto-loaded (see `autouse`) fixture, loaded per test (function scope). Prevents issues when running multiple tests that load metaschemas multiple times or in parallel (*pytest-parallel*, *pytest-xdist*, etc).

Return type

None

`schema_salad.tests.matcher`

Module Contents

Classes

JsonDiffMatcher

Raise `AssertionError` with a readable JSON diff when `not __eq__()`.

Functions

StripYAMLComments(yml)

class `schema_salad.tests.matcher.JsonDiffMatcher`(*expected*)

Raise AssertionError with a readable JSON diff when not `__eq__()`.

Used with `assert_called_with()` so it's possible for a human to see the differences between expected and actual call arguments that include non-trivial data structures.

Parameters

expected (*Any*) –

`__eq__`(*actual*)

Return self==value.

Parameters

actual (*Any*) –

Return type

bool

`schema_salad.tests.matcher.StripYAMLComments`(*yml*)

Parameters

yml (*str*) –

Return type

Any

`schema_salad.tests.test_avro_names`

Avro related tests.

Module Contents

Functions

test_avro_loading()

Confirm conversion of SALAD style names to avro.

`schema_salad.tests.test_avro_names.test_avro_loading`()

Confirm conversion of SALAD style names to avro.

Return type

None

`schema_salad.tests.test_cg`

Module Contents

Functions

<code>test_load()</code>	
<code>test_err()</code>	
<code>test_include()</code>	
<code>test_import()</code>	
<code>test_import2()</code>	
<code>test_err2()</code>	
<code>test_idmap()</code>	
<code>test_idmap2()</code>	
<code>test_load_pt()</code>	
<code>test_shortcode()</code>	Test shortcode() function.
<code>metaschema_pre()</code>	Prep-parsed schema for testing.
<code>test_load_metaschema(metaschema_pre)</code>	
<code>test_load_by_yaml_metaschema(metaschema_pre)</code>	
<code>test_load_cwlschema()</code>	

Attributes

<code>maxDiff</code>

`schema_salad.tests.test_cg.test_load()`

Return type

None

`schema_salad.tests.test_cg.test_err()`

Return type

None

`schema_salad.tests.test_cg.test_include()`

Return type

None

schema_salad.tests.test_cg.test_import()

Return type

None

schema_salad.tests.test_cg.maxDiff

schema_salad.tests.test_cg.test_import2()

Return type

None

schema_salad.tests.test_cg.test_err2()

Return type

None

schema_salad.tests.test_cg.test_idmap()

Return type

None

schema_salad.tests.test_cg.test_idmap2()

Return type

None

schema_salad.tests.test_cg.test_load_pt()

Return type

None

schema_salad.tests.test_cg.test_shortcode()

Test shortcode() function.

Return type

None

schema_salad.tests.test_cg.metaschema_pre()

Prep-parsed schema for testing.

Return type

Any

schema_salad.tests.test_cg.test_load_metaschema(*metaschema_pre*)**Parameters****metaschema_pre** (*Any*) –**Return type**

None

schema_salad.tests.test_cg.test_load_by_yaml_metaschema(*metaschema_pre*)**Parameters****metaschema_pre** (*Any*) –**Return type**

None

`schema_salad.tests.test_cg.test_load_cwlschema()`

Return type

None

`schema_salad.tests.test_cli_args`

test different sets of command line arguments

Module Contents

Functions

`captured_output()`

`test_version()`

`test_empty_input()`

`schema_salad.tests.test_cli_args.captured_output()`

Return type

Iterator[Tuple[io.StringIO, io.StringIO]]

`schema_salad.tests.test_cli_args.test_version()`

Return type

None

`schema_salad.tests.test_cli_args.test_empty_input()`

Return type

None

`schema_salad.tests.test_cpp_codegen`

Test C++ code generation.

Module Contents

Functions

`test_cwl_cpp_gen(tmp_path)`

End to end test of C++ generator using the CWL v1.0 schema.

`cpp_codegen(file_uri, target)`

Help using the C++ code generation function.

`schema_salad.tests.test_cpp_codegen.test_cwl_cpp_gen(tmp_path)`

End to end test of C++ generator using the CWL v1.0 schema.

Parameters

tmp_path (*pathlib.Path*) –

Return type

None

`schema_salad.tests.test_cpp_codegen.cpp_codegen(file_uri, target)`

Help using the C++ code generation function.

Parameters

- **file_uri** (*str*) –
- **target** (*pathlib.Path*) –

Return type

None

`schema_salad.tests.test_cwl11`

Ensure codegen-produced parsers accept \$schemas directives

run individually as `py.test -k test_cwl11`

Module Contents

Functions

<code>cwl_v1_2_schema(tmp_path_factory)</code>	
<code>load_cwl(cwl_v1_2_schema, src)</code>	
<code>test_secondaryFiles(cwl_v1_2_schema)</code>	secondaryFiles
<code>test_outputBinding(cwl_v1_2_schema)</code>	secondaryFiles
<code>test_yaml_tab_error(cwl_v1_2_schema)</code>	Tabs in the file.

Attributes

<code>test_dir_name</code>
<code>SchemaType</code>

`schema_salad.tests.test_cwl11.test_dir_name = 'tests/'`

`schema_salad.tests.test_cwl11.SchemaType`

`schema_salad.tests.test_cwl11.cwl_v1_2_schema(tmp_path_factory)`

Parameters

`tmp_path_factory` (`_pytest.tmpdir.TempPathFactory`) –

Return type

Generator[SchemaType, None, None]

`schema_salad.tests.test_cwl11.load_cwl(cwl_v1_2_schema, src)`

Parameters

- `cwl_v1_2_schema` (`SchemaType`) –
- `src` (`str`) –

Return type

Tuple[Any, Dict[str, Any]]

`schema_salad.tests.test_cwl11.test_secondaryFiles(cwl_v1_2_schema)`

secondaryFiles

Parameters

`cwl_v1_2_schema` (`SchemaType`) –

Return type

None

`schema_salad.tests.test_cwl11.test_outputBinding(cwl_v1_2_schema)`

secondaryFiles

Parameters

`cwl_v1_2_schema` (`SchemaType`) –

Return type

None

`schema_salad.tests.test_cwl11.test_yaml_tab_error(cwl_v1_2_schema)`

Tabs in the file.

Parameters

`cwl_v1_2_schema` (`SchemaType`) –

Return type

None

`schema_salad.tests.test_dlang_codegen`

Test D code generation.

Module Contents

Functions

<code>test_cwl_dlang_gen(tmp_path)</code>	End to end test of D generator using the CWL v1.0 schema.
<code>dlang_codegen(file_uri, target)</code>	Help using the D code generation function.

`schema_salad.tests.test_dlang_codegen.test_cwl_dlang_gen(tmp_path)`

End to end test of D generator using the CWL v1.0 schema.

Parameters

tmp_path (*pathlib.Path*) –

Return type

None

`schema_salad.tests.test_dlang_codegen.dlang_codegen(file_uri, target)`

Help using the D code generation function.

Parameters

- **file_uri** (*str*) –
- **target** (*pathlib.Path*) –

Return type

None

`schema_salad.tests.test_dotnet_codegen`

Module Contents

Functions

<code>test_cwl_gen(tmp_path)</code>
<code>test_meta_schema_gen(tmp_path)</code>
<code>test_class_field(tmp_path)</code>
<code>get_data_uri(resource_path)</code>
<code>dotnet_codegen(file_uri, target[, examples])</code>

Attributes

`cwl_file_uri`

`metaschema_file_uri`

`schema_salad.tests.test_dotnet_codegen.test_cwl_gen(tmp_path)`

Parameters

`tmp_path` (`pathlib.Path`) –

Return type

None

`schema_salad.tests.test_dotnet_codegen.test_meta_schema_gen(tmp_path)`

Parameters

`tmp_path` (`pathlib.Path`) –

Return type

None

`schema_salad.tests.test_dotnet_codegen.test_class_field(tmp_path)`

Parameters

`tmp_path` (`pathlib.Path`) –

Return type

None

`schema_salad.tests.test_dotnet_codegen.get_data_uri(resource_path)`

Parameters

`resource_path` (`str`) –

Return type

`str`

`schema_salad.tests.test_dotnet_codegen.cwl_file_uri`

`schema_salad.tests.test_dotnet_codegen.metaschema_file_uri`

`schema_salad.tests.test_dotnet_codegen.dotnet_codegen(file_uri, target, examples=None)`

Parameters

- `file_uri` (`str`) –
- `target` (`pathlib.Path`) –
- `examples` (`Optional[pathlib.Path]`) –

Return type

None

schema_salad.tests.test_errors

Tests of helpful error messages.

Module Contents**Functions**

<code>test_errors()</code>	
<code>test_error_message1()</code>	
<code>test_error_message2()</code>	
<code>test_error_message3()</code>	
<code>test_error_message4()</code>	
<code>test_error_message5()</code>	
<code>test_error_message7()</code>	
<code>test_error_message8()</code>	
<code>test_error_message9()</code>	
<code>test_error_message10()</code>	
<code>test_error_message11()</code>	
<code>test_error_message15()</code>	
<code>test_errors_previously_defined_dict_key()</code>	
<code>test_bad_schema()</code>	
<code>test_bad_schema2()</code>	
<code>test_namespaces_type()</code>	Confirm helpful error message when \$namespaces is the wrong type.
<code>test_namespaces_undeclared(caplog)</code>	Confirm warning message a namespace is used but not declared.
<code>test_not_a_namespace1(caplog)</code>	Confirm no warning when relative id contains a colon but prefix doesn't look like a namespace.
<code>test_not_a_namespace2(caplog)</code>	Confirm no warning when relative id contains a colon but prefix doesn't look like a namespace.
<code>test_not_a_namespace3(caplog)</code>	Confirm no warning when relative id starts with a colon.
<code>test_schemas_type()</code>	Confirm helpful error message when \$schemas is the wrong type.

`schema_salad.tests.test_errors.test_errors()`

Return type

None

`schema_salad.tests.test_errors.test_error_message1()`

Return type

None

`schema_salad.tests.test_errors.test_error_message2()`

Return type

None

`schema_salad.tests.test_errors.test_error_message3()`

Return type

None

`schema_salad.tests.test_errors.test_error_message4()`

Return type

None

`schema_salad.tests.test_errors.test_error_message5()`

Return type

None

`schema_salad.tests.test_errors.test_error_message7()`

Return type

None

`schema_salad.tests.test_errors.test_error_message8()`

Return type

None

`schema_salad.tests.test_errors.test_error_message9()`

Return type

None

`schema_salad.tests.test_errors.test_error_message10()`

Return type

None

`schema_salad.tests.test_errors.test_error_message11()`

Return type

None

`schema_salad.tests.test_errors.test_error_message15()`

Return type

None

`schema_salad.tests.test_errors.test_errors_previously_defined_dict_key()`

Return type

None

`schema_salad.tests.test_errors.test_bad_schema()`

Return type

None

`schema_salad.tests.test_errors.test_bad_schema2()`

Return type

None

`schema_salad.tests.test_errors.test_namespaces_type()`

Confirm helpful error message when \$namespaces is the wrong type.

Return type

None

`schema_salad.tests.test_errors.test_namespaces_undeclared(caplog)`

Confirm warning message a namespace is used but not declared.

Parameters

caplog (*pytest.LogCaptureFixture*) –

Return type

None

`schema_salad.tests.test_errors.test_not_a_namespace1(caplog)`

Confirm no warning when relative id contains a colon but prefix doesn't look like a namespace.

Parameters

caplog (*pytest.LogCaptureFixture*) –

Return type

None

`schema_salad.tests.test_errors.test_not_a_namespace2(caplog)`

Confirm no warning when relative id contains a colon but prefix doesn't look like a namespace.

Parameters

caplog (*pytest.LogCaptureFixture*) –

Return type

None

`schema_salad.tests.test_errors.test_not_a_namespace3(caplog)`

Confirm no warning when relative id starts with a colon.

Parameters

caplog (*pytest.LogCaptureFixture*) –

Return type

None

`schema_salad.tests.test_errors.test_schemas_type()`

Confirm helpful error message when \$schemas is the wrong type.

Return type

None

`schema_salad.tests.test_examples`

Test examples.

Module Contents

Functions

<code>test_schemas()</code>	
<code>test_bad_schemas(caplog)</code>	Test that bad \$schemas refs don't stop parsing.
<code>test_skip_bad_schemas(caplog)</code>	Test that (bad) \$schemas refs are properly skipped.
<code>test_self_validate()</code>	
<code>test_print_rdf()</code>	Test --print-rdf.
<code>test_print_rdf_invalid_external_ref()</code>	Test --print-rdf when document references unfetchable external schema.
<code>test_print_pre_schema()</code>	Test --print-pre only schema.
<code>test_print_pre()</code>	Test --print-pre.
<code>test_print_schema_index()</code>	Test --print-index only with a schema.
<code>test_print_index()</code>	Test --print-index.
<code>test_print_schema_metadata()</code>	Test --print-metadata only for a schema.
<code>test_print_metadata()</code>	Test --print-metadata.
<code>test_schema_salad_doc_online_doc()</code>	Test schema-salad-doc when the 1st type has only a single doc line.
<code>test_avro_regression()</code>	
<code>test_jsonld_ctx()</code>	
<code>test_idmap()</code>	
<code>test_scoped_ref()</code>	
<code>test_examples()</code>	
<code>test_yaml_float_test()</code>	
<code>test_typedsl_ref()</code>	
<code>test_nested_typedsl_ref()</code>	
<code>test_secondaryFile_dsl_ref()</code>	
<code>test_scoped_id()</code>	
<code>test_rdf_datetime()</code>	Affirm that datetime objects can be serialized in <code>makerdf()</code> .
<code>test_yaml_datetime()</code>	Affirm that <code>yaml_no_ts</code> prevents the creation of datetime objects.

continues on next page

Table 1 – continued from previous page

<code>test_subscoped_id()</code>	
<code>test_mixin()</code>	
<code>test_fragment()</code>	
<code>test_file_uri()</code>	
<code>test_sourceline()</code>	
<code>test_cmap()</code>	
<code>test_blank_node_id()</code>	
<code>test_can_use_Any()</code>	Test that 'type: Any' can be used
<code>test_nullable_links()</code>	

`schema_salad.tests.test_examples.test_schemas()`

Return type

None

`schema_salad.tests.test_examples.test_bad_schemas(caplog)`

Test that bad \$schemas refs don't stop parsing.

Parameters

caplog (*pytest.LogCaptureFixture*) –

Return type

None

`schema_salad.tests.test_examples.test_skip_bad_schemas(caplog)`

Test that (bad) \$schemas refs are properly skipped.

Parameters

caplog (*pytest.LogCaptureFixture*) –

Return type

None

`schema_salad.tests.test_examples.test_self_validate()`

Return type

None

`schema_salad.tests.test_examples.test_print_rdf()`

Test –print-rdf.

Return type

None

`schema_salad.tests.test_examples.test_print_rdf_invalid_external_ref()`

Test –print-rdf when document references unfetchable external schema.

Return type

None

`schema_salad.tests.test_examples.test_print_pre_schema()`

Test `-print-pre` only schema.

Return type

None

`schema_salad.tests.test_examples.test_print_pre()`

Test `-print-pre`.

Return type

None

`schema_salad.tests.test_examples.test_print_schema_index()`

Test `-print-index` only with a schema.

Return type

None

`schema_salad.tests.test_examples.test_print_index()`

Test `-print-index`.

Return type

None

`schema_salad.tests.test_examples.test_print_schema_metadata()`

Test `-print-metadata` only for a schema.

Return type

None

`schema_salad.tests.test_examples.test_print_metadata()`

Test `-print-metadata`.

Return type

None

`schema_salad.tests.test_examples.test_schema_salad_doc_online_doc()`

Test `schema-salad-doc` when the 1st type has only a single doc line.

Return type

None

`schema_salad.tests.test_examples.test_avro_regression()`

Return type

None

`schema_salad.tests.test_examples.test_jsonld_ctx()`

Return type

None

`schema_salad.tests.test_examples.test_idmap()`

Return type

None

`schema_salad.tests.test_examples.test_scoped_ref()`

Return type

None

`schema_salad.tests.test_examples.test_examples()`

Return type

None

`schema_salad.tests.test_examples.test_yaml_float_test()`

Return type

None

`schema_salad.tests.test_examples.test_typedsl_ref()`

Return type

None

`schema_salad.tests.test_examples.test_nested_typedsl_ref()`

Return type

None

`schema_salad.tests.test_examples.test_secondaryFile_dsl_ref()`

Return type

None

`schema_salad.tests.test_examples.test_scoped_id()`

Return type

None

`schema_salad.tests.test_examples.test_rdf_datetime()`

Affirm that datetime objects can be serialized in makerdf().

Return type

None

`schema_salad.tests.test_examples.test_yaml_datetime()`

Affirm that `yaml_no_ts` prevents the creation of datetime objects.

Return type

None

`schema_salad.tests.test_examples.test_subscoped_id()`

Return type

None

`schema_salad.tests.test_examples.test_mixin()`

Return type

None

`schema_salad.tests.test_examples.test_fragment()`

Return type

None

`schema_salad.tests.test_examples.test_file_uri()`

Return type

None

`schema_salad.tests.test_examples.test_sourceline()`

Return type

None

`schema_salad.tests.test_examples.test_cmap()`

Return type

None

`schema_salad.tests.test_examples.test_blank_node_id()`

Return type

None

`schema_salad.tests.test_examples.test_can_use_Any()`

Test that 'type: Any' can be used

Return type

None

`schema_salad.tests.test_examples.test_nullable_links()`

Return type

None

`schema_salad.tests.test_fetch`

Module Contents

Classes

<code>testFetcher</code>	Fetch resources from URIs.
<code>CWLTestFetcher</code>	Fetch resources from URIs.

Functions

<code>test_fetcher()</code>
<code>test_cache()</code>

class `schema_salad.tests.test_fetch.testFetcher`(*cache*, *session*)

Bases: `schema_salad.fetcher.Fetcher`

Fetch resources from URIs.

Parameters

- **cache** (`schema_salad.utils.CacheType`) –
- **session** (`Optional[requests.sessions.Session]`) –

fetch_text(*url*, *content_types=None*)

Retrieve the given resource as a string.

Parameters

- **url** (*str*) –
- **content_types** (*Optional[List[str]*) –

Return type

str

check_exists(*url*)

Check if the given resource exists.

Parameters

url (*str*) –

Return type

bool

urljoin(*base*, *url*)

Parameters

- **base** (*str*) –
- **url** (*str*) –

Return type

str

class `schema_salad.tests.test_fetch.CWLTTestFetcher`(*cache*, *session*)

Bases: `schema_salad.fetcher.Fetcher`

Fetch resources from URIs.

Parameters

- **cache** (`schema_salad.utils.CacheType`) –
- **session** (*Optional[requests.sessions.Session]*) –

fetch_text(*url*, *content_types=None*)

Retrieve the given resource as a string.

Parameters

- **url** (*str*) –
- **content_types** (*Optional[List[str]*) –

Return type

str

check_exists(*url*)

Check if the given resource exists.

Parameters

url (*str*) –

Return type

bool

`urljoin(base, url)`

Parameters

- **base** (*str*) –
- **url** (*str*) –

Return type

str

`schema_salad.tests.test_fetch.test_fetcher()`

Return type

None

`schema_salad.tests.test_fetch.test_cache()`

Return type

None

`schema_salad.tests.test_fp`

Module Contents

Functions

`test_fp()`

`schema_salad.tests.test_fp.test_fp()`

Return type

None

`schema_salad.tests.test_java_codegen`

Module Contents

Functions

`test_cwl_gen(tmp_path)`

`test_meta_schema_gen(tmp_path)`

`java_codegen(file_uri, target[, examples])`

`schema_salad.tests.test_java_codegen.test_cwl_gen(tmp_path)`

Parameters

`tmp_path` (*pathlib.Path*) –

Return type

None

`schema_salad.tests.test_java_codegen.test_meta_schema_gen(tmp_path)`

Parameters

`tmp_path` (*pathlib.Path*) –

Return type

None

`schema_salad.tests.test_java_codegen.java_codegen(file_uri, target, examples=None)`

Parameters

- `file_uri` (*str*) –
- `target` (*pathlib.Path*) –
- `examples` (*Optional[pathlib.Path]*) –

Return type

None

`schema_salad.tests.test_makedoc`

Test schema-salad-doc.

(also known as `schema-salad-tool --print-doc`)

For convenience, tests are checking exact strings. In the event of changes in the “mistune” package, `makedoc.py`, or other changes, feel free to modify the test strings as long as the new HTML renders the same way in typical browsers.

Likewise, if the schema-salad metaschema changes and it is missing one or more of the features tested below, then please copy those old features to a new file and update the affected tests to use those new file(s).

Module Contents

Functions

<code>test_schema_salad_inherit_docs()</code>	Test schema-salad-doc when types inherit and override values from parent types.
<code>generate_doc([schema_data])</code>	Avoid error when calling fixture directly.
<code>fixture_metaschema_doc()</code>	Pytest Fixture of the rendered HTML for the metaschema schema.
<code>test_doc_fenced_code_contents_preserved()</code>	Fenced code contents are not interpreted as Markdown definitions and converted into erroneous HTML.
<code>test_doc_headings_target_anchor(metaschema_doc)</code>	Doc headers must have an id and section link.
<code>test_doc_render_table_of_contents(metaschema_doc)</code>	The special Table of Contents token must be replaced with a rendered table.
<code>test_plain_links_autolinked(metaschema_doc)</code>	Plain links should be treated as if they were wrapped in angle brackets.
<code>test_embedded_html_unescaped()</code>	Raw HTML shouldn't get escaped.
<code>test_multiline_list_entries_word_spacing(metaschema_doc)</code>	Hanging indents in Markdown lists don't lead to wordsmushing.
<code>test_multiline_list_entries_without_indentation(metaschema_doc)</code>	Hanging indents are not required in Markdown lists.
<code>test_detect_changes_in_html(metaschema_doc, tmp_path)</code>	Catch all for changes in HTML output, please adjust if the changes are innocent.

`schema_salad.tests.test_makedoc.test_schema_salad_inherit_docs()`

Test schema-salad-doc when types inherit and override values from parent types.

Return type

None

`schema_salad.tests.test_makedoc.generate_doc(schema_data=None)`

Avoid error when calling fixture directly.

Parameters

`schema_data` (Optional[str]) –

Return type

str

`schema_salad.tests.test_makedoc.fixture_metaschema_doc()`

Pytest Fixture of the rendered HTML for the metaschema schema.

Return type

str

`schema_salad.tests.test_makedoc.test_doc_fenced_code_contents_preserved()`

Fenced code contents are not interpreted as Markdown definitions and converted into erroneous HTML.

An example of problem case is when a definition looks like a Markdown list (e.g.: a YAML array). It must not be converted into HTML contents with list tags. However, special characters (e.g.: <, >) must still be escaped, otherwise they will not be correctly rendered within an HTML <pre><code> block.

Return type

None

`schema_salad.tests.test_makedoc.test_doc_headings_target_anchor(metaschema_doc)`

Doc headers must have an id and section link.

Parameters

`metaschema_doc` (str) –

Return type

None

`schema_salad.tests.test_makedoc.test_doc_render_table_of_contents(metaschema_doc)`

The special Table of Contents token must be replaced with a rendered table.

Parameters`metaschema_doc` (*str*) –**Return type**

None

`schema_salad.tests.test_makedoc.test_plain_links_autolinked(metaschema_doc)`

Plan links should be treated as if they were wrapped in angle brackets.

Parameters`metaschema_doc` (*str*) –**Return type**

None

`schema_salad.tests.test_makedoc.test_embedded_html_unescaped()`

Raw HTML shouldn't get escaped.

Return type

None

`schema_salad.tests.test_makedoc.test_multiline_list_entries_word_spacing(metaschema_doc)`

Hanging indents in Markdown lists don't lead to wordsmushing.

Parameters`metaschema_doc` (*str*) –**Return type**

None

`schema_salad.tests.test_makedoc.test_multiline_list_entries_without_indention(metaschema_doc)`

Hanging indents are not required in Markdown lists.

Parameters`metaschema_doc` (*str*) –**Return type**

None

`schema_salad.tests.test_makedoc.test_detect_changes_in_html(metaschema_doc, tmp_path)`

Catch all for changes in HTML output, please adjust if the changes are innocent.

Parameters

- `metaschema_doc` (*str*) –
- `tmp_path` (*pathlib.Path*) –

Return type

None

`schema_salad.tests.test_misc`

Module Contents

Functions

`test_misc()`

`test_load_schema_cache()`

`schema_salad.tests.test_misc.test_misc()`

Return type

None

`schema_salad.tests.test_misc.test_load_schema_cache()`

Return type

None

`schema_salad.tests.test_pickling`

Tests to ensure that mypyc compiled classes are still pickleable.

See https://mypyc.readthedocs.io/en/latest/differences_from_python.html#pickling-and-copying-objects

Module Contents

Functions

`test_recordschema_pickle()`

Targeted test of pickling a RecordSchema.

`test_extend_and_specialize_enums(tmp_path)`

`schema_salad.tests.test_pickling.test_recordschema_pickle()`

Targeted test of pickling a RecordSchema.

Return type

None

`schema_salad.tests.test_pickling.test_extend_and_specialize_enums(tmp_path)`

Parameters

`tmp_path` (*pathlib.Path*) –

Return type

None

`schema_salad.tests.test_print_online`

Module Contents

Functions

`test_print_online()`

`test_print_online_for_invalid_yaml()`

`test_print_online_for_errors_in_the_same_line()`

`test_print_online_for_errors_in_resolve_ref()`

`test_for_invalid_yaml1()`

`test_for_invalid_yaml2()`

`schema_salad.tests.test_print_online.test_print_online()`

Return type

None

`schema_salad.tests.test_print_online.test_print_online_for_invalid_yaml()`

Return type

None

`schema_salad.tests.test_print_online.test_print_online_for_errors_in_the_same_line()`

Return type

None

`schema_salad.tests.test_print_online.test_print_online_for_errors_in_resolve_ref()`

Return type

None

`schema_salad.tests.test_print_online.test_for_invalid_yaml1()`

Return type

None

`schema_salad.tests.test_print_online.test_for_invalid_yaml2()`

Return type

None

`schema_salad.tests.test_python_codegen`

Module Contents

Functions

<code>test_safe_identifiers()</code>	Affirm correct construction of identifiers safe for Python.
<code>test_cwl_gen(tmp_path)</code>	
<code>test_meta_schema_gen(tmp_path)</code>	
<code>test_meta_schema_gen_up_to_date(tmp_path)</code>	
<code>test_meta_schema_gen_no_base(tmp_path)</code>	
<code>python_codegen(file_uri, target[, parser_info, package])</code>	
<code>test_default_parser_info(tmp_path)</code>	
<code>test_parser_info(tmp_path)</code>	
<code>test_use_of_package_for_parser_info(tmp_path)</code>	
<code>test_graph_property()</code>	Test the RDFLib Graph representation of the <code>\$schemas</code> directive.
<code>test_graph_property_cache()</code>	Test that LoadingOptions properly cache the <code>\$schemas</code> RDFLib Graph representations.
<code>test_graph_property_empty_schema()</code>	Test that an empty RDFLib Graph is returned when not <code>\$schemas</code> directive is present.

`schema_salad.tests.test_python_codegen.test_safe_identifiers()`

Affirm correct construction of identifiers safe for Python.

Return type

None

`schema_salad.tests.test_python_codegen.test_cwl_gen(tmp_path)`

Parameters

`tmp_path` (`pathlib.Path`) –

Return type

None

`schema_salad.tests.test_python_codegen.test_meta_schema_gen(tmp_path)`

Parameters

`tmp_path` (`pathlib.Path`) –

Return type

None

`schema_salad.tests.test_python_codegen.test_meta_schema_gen_up_to_date(tmp_path)`

Parameters**tmp_path** (*pathlib.Path*) –**Return type**

None

schema_salad.tests.test_python_codegen.test_meta_schema_gen_no_base(*tmp_path*)**Parameters****tmp_path** (*pathlib.Path*) –**Return type**

None

schema_salad.tests.test_python_codegen.python_codegen(*file_uri*, *target*, *parser_info=None*,
package=None)**Parameters**

- **file_uri** (*str*) –
- **target** (*pathlib.Path*) –
- **parser_info** (*Optional[str]*) –
- **package** (*Optional[str]*) –

Return type

None

schema_salad.tests.test_python_codegen.test_default_parser_info(*tmp_path*)**Parameters****tmp_path** (*pathlib.Path*) –**Return type**

None

schema_salad.tests.test_python_codegen.test_parser_info(*tmp_path*)**Parameters****tmp_path** (*pathlib.Path*) –**Return type**

None

schema_salad.tests.test_python_codegen.test_use_of_package_for_parser_info(*tmp_path*)**Parameters****tmp_path** (*pathlib.Path*) –**Return type**

None

schema_salad.tests.test_python_codegen.test_graph_property()

Test the RDFLib Graph representation of the *\$schemas* directive.**Return type**

None

schema_salad.tests.test_python_codegen.test_graph_property_cache()

Test that LoadingOptions properly cache the *\$schemas* RDFLib Graph representations.

Return type

None

`schema_salad.tests.test_python_codegen.test_graph_property_empty_schema()`

Test that an empty RDFLib Graph is returned when not *\$schemas* directive is present.

Return type

None

`schema_salad.tests.test_real_cwl`

Checks loading of some real world tools and workflows found in the wild (e.g. dockstore)

run individually as `py.test -k tests/test_real_cwl.py`

Module Contents

Classes

TestRealWorldCWL

Attributes

test_dir_name

`schema_salad.tests.test_real_cwl.test_dir_name = 'tests/test_real_cwl/'`

`class schema_salad.tests.test_real_cwl.TestRealWorldCWL`

`document_loader: schema_salad.ref_resolver.Loader`

`avsc_names: schema_salad.avro.schema.Names | schema_salad.avro.schema.SchemaParseException | None`

`schema_metadata: Dict[str, Any] | None`

`metaschema_loader: schema_salad.ref_resolver.Loader | None`

`classmethod setup_class()`

Return type

None

`load_cwl(src)`

Parameters

`src (str)` –

Return type

None

test_topmed_single_doc()

TOPMed Variant Calling Pipeline CWL1

Return type

None

test_h3agatk_WES()

H3ABioNet GATK Germline Workflow

Return type

None

test_h3agatk_SNP()

H3ABioNet SNPs Workflow

Return type

None

test_icgc_pancan()

ICGC PanCan

Return type

None

schema_salad.tests.test_ref_resolver

Test the ref_resolver module.

Module Contents

Functions

<code>is_fs_case_sensitive(path)</code>	
<code>tmp_dir_fixture(request)</code>	
<code>test_loader_initialisation_for_HOME_env_var(tmp_dir_fixture)</code>	
<code>test_loader_initialisation_for_TMP_env_var(tmp_dir_fixture)</code>	
<code>test_loader_initialisation_with_neither_TMP_env_var(tmp_dir_fixture)</code>	
<code>test_loader_initialisation_disable_doc_cache(tmp_dir_fixture)</code>	
<code>test_DefaultFetcher_urljoin_win32(tmp_dir_fixture)</code>	
<code>test_DefaultFetcher_urljoin_linux(tmp_dir_fixture)</code>	
<code>test_import_list()</code>	
<code>test_fetch_inject_id()</code>	
<code>test_attachments()</code>	
<code>test_check_exists_follows_redirects()</code>	
<code>test_resolve_missing_step_id(caplog)</code>	From issue #cwltool/issues/1635. A Workflow with a Step without

`schema_salad.tests.test_ref_resolver.is_fs_case_sensitive(path)`

Parameters

path (*str*) –

Return type

bool

`schema_salad.tests.test_ref_resolver.tmp_dir_fixture(request)`

Parameters

request (*pytest.fixtures.FixtureRequest*) –

Return type

str

`schema_salad.tests.test_ref_resolver.test_loader_initialisation_for_HOME_env_var(tmp_dir_fixture)`

Parameters

tmp_dir_fixture (*str*) –

Return type

None

`schema_salad.tests.test_ref_resolver.test_loader_initialisation_for_TMP_env_var(tmp_dir_fixture)`

Parameters**tmp_dir_fixture** (*str*) –**Return type**

None

schema_salad.tests.test_ref_resolver.test_loader_initialisation_with_neither_TMP_HOME_set(*tmp_dir_fixture*)**Parameters****tmp_dir_fixture** (*str*) –**Return type**

None

schema_salad.tests.test_ref_resolver.test_loader_initialisation_disable_doc_cache(*tmp_dir_fixture*)**Parameters****tmp_dir_fixture** (*str*) –**Return type**

None

schema_salad.tests.test_ref_resolver.test_DefaultFetcher_urljoin_win32(*tmp_dir_fixture*)**Parameters****tmp_dir_fixture** (*str*) –**Return type**

None

schema_salad.tests.test_ref_resolver.test_DefaultFetcher_urljoin_linux(*tmp_dir_fixture*)**Parameters****tmp_dir_fixture** (*str*) –**Return type**

None

schema_salad.tests.test_ref_resolver.test_import_list()

Return type

None

schema_salad.tests.test_ref_resolver.test_fetch_inject_id()

Return type

None

schema_salad.tests.test_ref_resolver.test_attachments()

Return type

None

schema_salad.tests.test_ref_resolver.test_check_exists_follows_redirects()

Return type

None

schema_salad.tests.test_ref_resolver.test_resolve_missing_step_id(*caplog*)

From issue [#cwltool/issues/1635](#). A Workflow with a Step without the name attribute must raise a ValidationException that contains the SourceLine data.

Schema Salad

Parameters

caplog (*Any*) –

Return type

None

`schema_salad.tests.test_schema`

Module Contents

Functions

```
test_extend_and_specialize_enums(tmp_path)
```

Attributes

```
cwl_file_uri
```

`schema_salad.tests.test_schema.cwl_file_uri`

`schema_salad.tests.test_schema.test_extend_and_specialize_enums`(*tmp_path*)

Parameters

tmp_path (*pathlib.Path*) –

Return type

None

`schema_salad.tests.test_schemas_directive`

Checks for accepting \$schemas directive

run individually as `py.test -k tests/test_schemas_directive.py`

Module Contents

Classes

```
TestSchemasDirective
```

Ensure codegen-produced parsers accept \$schemas directives

Attributes

```
test_dir_name
```

```
schema_salad.tests.test_schemas_directive.test_dir_name = 'tests/'
```

```
class schema_salad.tests.test_schemas_directive.TestSchemasDirective
```

```
    Ensure codegen-produced parsers accept $schemas directives
```

```
    document_loader: schema_salad.ref_resolver.Loader
```

```
    avsc_names: schema_salad.avro.schema.Names |  
               schema_salad.avro.schema.SchemaParseException | None
```

```
    schema_metadata: Dict[str, Any] | None
```

```
    metaschema_loader: schema_salad.ref_resolver.Loader | None
```

```
    classmethod setup_class()
```

```
        Return type
```

```
        None
```

```
    load_cwl(src)
```

```
        Parameters
```

```
        src (str) –
```

```
        Return type
```

```
        Tuple[Any, Dict[str, Any]]
```

```
    test_dollarsign_schema()
```

```
        EDAM.owl as a schema
```

```
        Return type
```

```
        None
```

```
schema_salad.tests.test_subtypes
```

Confirm subtypes.

Module Contents

Functions

<i>test_subtypes</i> (old, new, result)	Test <code>is_subtype()</code> function.
<i>test_avro_loading_subtype</i> ()	Confirm conversion of SALAD style names to avro when overriding.
<i>test_avro_loading_subtype_bad</i> ()	Confirm subtype error when overriding incorrectly.

Attributes

types

```
schema_salad.tests.test_subtypes.types = ([('int', 'float', 'double'), 'int', True),  
(['int', 'float', 'double'], ['int'], True),...
```

`schema_salad.tests.test_subtypes.test_subtypes`(*old, new, result*)

Test `is_subtype()` function.

Parameters

- **old** (`schema_salad.avro.schema.PropertyType`) –
- **new** (`schema_salad.avro.schema.PropertyType`) –
- **result** (`bool`) –

Return type

None

`schema_salad.tests.test_subtypes.test_avro_loading_subtype()`

Confirm conversion of SALAD style names to avro when overriding.

Return type

None

`schema_salad.tests.test_subtypes.test_avro_loading_subtype_bad()`

Confirm subtype error when overriding incorrectly.

Return type

None

`schema_salad.tests.test_typescript_codegen`

Module Contents

Functions

`test_cwl_gen`(*tmp_path*)

`test_meta_schema_gen`(*tmp_path*)

`test_class_field`(*tmp_path*)

`get_data_uri`(*resource_path*)

`typescript_codegen`(*file_uri, target[, examples]*)

Attributes

`cwl_file_uri`

`metaschema_file_uri`

`schema_salad.tests.test_typescript_codegen.test_cwl_gen(tmp_path)`

Parameters

`tmp_path` (`pathlib.Path`) –

Return type

None

`schema_salad.tests.test_typescript_codegen.test_meta_schema_gen(tmp_path)`

Parameters

`tmp_path` (`pathlib.Path`) –

Return type

None

`schema_salad.tests.test_typescript_codegen.test_class_field(tmp_path)`

Parameters

`tmp_path` (`pathlib.Path`) –

Return type

None

`schema_salad.tests.test_typescript_codegen.get_data_uri(resource_path)`

Parameters

`resource_path` (`str`) –

Return type

`str`

`schema_salad.tests.test_typescript_codegen.cwl_file_uri`

`schema_salad.tests.test_typescript_codegen.metaschema_file_uri`

`schema_salad.tests.test_typescript_codegen.typescript_codegen(file_uri, target, examples=None)`

Parameters

- `file_uri` (`str`) –
- `target` (`pathlib.Path`) –
- `examples` (`Optional[pathlib.Path]`) –

Return type

None

`schema_salad.tests.util`

Module Contents

Functions

<code>get_data(filename)</code>	Get the file path for a given schema file name.
<code>get_data_uri(resource_path)</code>	Get the file URI for tests.

Attributes

`cwl_file_uri`

`metaschema_file_uri`

`basket_file_uri`

`schema_salad.tests.util.get_data(filename)`

Get the file path for a given schema file name.

It is able to find file names in the `schema_salad` namespace, but also able to load schema files from the `tests` directory.

Parameters

filename (*str*) –

Return type

Optional[*str*]

`schema_salad.tests.util.get_data_uri(resource_path)`

Get the file URI for tests.

Parameters

resource_path (*str*) –

Return type

str

`schema_salad.tests.util.cwl_file_uri`

`schema_salad.tests.util.metaschema_file_uri`

`schema_salad.tests.util.basket_file_uri`

Submodules

`schema_salad.__main__`

Default entry point for the schema-salad module.

`schema_salad.codegen`

Generate language specific loaders for a particular SALAD schema.

Module Contents

Functions

<code>codegen</code> (<i>lang</i> , <i>i</i> , <i>schema_metadata</i> , <i>loader</i> [, <i>target</i> , ...])	Generate classes with loaders for the given Schema Salad description.
---	---

Attributes

`FIELD_SORT_ORDER`

`schema_salad.codegen.FIELD_SORT_ORDER = ['id', 'class', 'name']`

`schema_salad.codegen.codegen`(*lang*, *i*, *schema_metadata*, *loader*, *target=None*, *examples=None*, *package=None*, *copyright=None*, *parser_info=None*)

Generate classes with loaders for the given Schema Salad description.

Parameters

- **lang** (*str*) –
- **i** (*List[Dict[str, str]]*) –
- **schema_metadata** (*Dict[str, Any]*) –
- **loader** (`schema_salad.ref_resolver.Loader`) –
- **target** (*Optional[str]*) –
- **examples** (*Optional[str]*) –
- **package** (*Optional[str]*) –
- **copyright** (*Optional[str]*) –
- **parser_info** (*Optional[str]*) –

Return type

None

`schema_salad.codegen_base`

Base class for the generation of loaders from schema-salad definitions.

Module Contents

Classes

<code>TypeDef</code>	Schema Salad type description.
<code>CodeGenBase</code>	Abstract base class for schema salad code generators.

```
class schema_salad.codegen_base.TypeDef(name, init, is_uri=False, scoped_id=False, ref_scope=0,  
                                         loader_type=None, instance_type=None, abstract=False)
```

Schema Salad type description.

Parameters

- **name** (*str*) –
- **init** (*str*) –
- **is_uri** (*bool*) –
- **scoped_id** (*bool*) –
- **ref_scope** (*Optional[int]*) –
- **loader_type** (*Optional[str]*) –
- **instance_type** (*Optional[str]*) –
- **abstract** (*bool*) –

```
__slots__ = ['name', 'init', 'is_uri', 'scoped_id', 'ref_scope', 'loader_type',  
            'instance_type', 'abstract']
```

```
class schema_salad.codegen_base.CodeGenBase
```

Abstract base class for schema salad code generators.

```
declare_type(declared_type)
```

Add this type to our collection, if needed.

Parameters

declared_type (*TypeDef*) –

Return type

TypeDef

```
add_vocab(name, uri)
```

Add the given name as an abbreviation for the given URI.

Parameters

- **name** (*str*) –
- **uri** (*str*) –

Return type

None

abstract prologue()

Trigger to generate the prologue code.

Return type

None

abstract static safe_name(name)

Generate a safe version of the given name.

Parameters

name (*str*) –

Return type

str

abstract begin_class(classname, extends, doc, abstract, field_names, idfield, optional_fields)

Produce the header for the given class.

Parameters

- **classname** (*str*) –
- **extends** (*MutableSequence[str]*) –
- **doc** (*str*) –
- **abstract** (*bool*) –
- **field_names** (*MutableSequence[str]*) –
- **idfield** (*str*) –
- **optional_fields** (*Set[str]*) –

Return type

None

abstract end_class(classname, field_names)

Signal that we are done with this class.

Parameters

- **classname** (*str*) –
- **field_names** (*List[str]*) –

Return type

None

abstract type_loader(type_declaration)

Parse the given type declaration and declare its components.

Parameters

type_declaration (*Union[List[Any], Dict[str, Any]]*) –

Return type

TypeDef

abstract declare_field(name, fieldtype, doc, optional, subscope)

Output the code to load the given field.

Parameters

- **name** (*str*) –
- **fieldtype** (*TypeDef*) –

- **doc** (*Optional[str]*) –
- **optional** (*bool*) –
- **subscope** (*str*) –

Return type

None

abstract declare_id_field(*name, fieldtype, doc, optional*)

Output the code to handle the given ID field.

Parameters

- **name** (*str*) –
- **fieldtype** (*TypeDef*) –
- **doc** (*str*) –
- **optional** (*bool*) –

Return type

None

abstract uri_loader(*inner, scoped_id, vocab_term, ref_scope*)

Construct the TypeDef for the given URI loader.

Parameters

- **inner** (*TypeDef*) –
- **scoped_id** (*bool*) –
- **vocab_term** (*bool*) –
- **ref_scope** (*Optional[int]*) –

Return type*TypeDef***abstract idmap_loader**(*field, inner, map_subject, map_predicate*)

Construct the TypeDef for the given mapped ID loader.

Parameters

- **field** (*str*) –
- **inner** (*TypeDef*) –
- **map_subject** (*str*) –
- **map_predicate** (*Optional[str]*) –

Return type*TypeDef***abstract typedsl_loader**(*inner, ref_scope*)

Construct the TypeDef for the given DSL loader.

Parameters

- **inner** (*TypeDef*) –
- **ref_scope** (*Optional[int]*) –

Return type*TypeDef*

abstract epilogue(*root_loader*)

Trigger to generate the epilogue code.

Parameters

root_loader (*TypeDef*) –

Return type

None

abstract secondaryfilesdsl_loader(*inner*)

Construct the TypeDef for secondary files.

Parameters

inner (*TypeDef*) –

Return type

TypeDef

schema_salad.cpp_codegen

C++17 code generator for a given Schema Salad definition.

Currently only supports emitting YAML from the C++ objects, not yet parsing YAML into C++ objects.

The generated code requires the libyaml-cpp library & headers

To see an example of usage, look at `schema_salad/tests/codegen/cwl.cpp` which can be combined with the CWL V1.0 schema as shown below:

```
schema-salad-tool --codegen cpp          schema_salad/tests/test_schema/
↳ CommonWorkflowLanguage.yml           > cwl_v1_0.h

g++ --std=c++20 -I. -lyaml-cpp schema_salad/tests/codegen/cwl.cpp -o cwl-v1_0-test
./cwl-v1_0-test

# g++ versions older than version 10 may need "--std=c++2a" instead of "--std=c++20"
```

Module Contents**Classes**

<i>ClassDefinition</i>	
<i>FieldDefinition</i>	
<i>EnumDefinition</i>	
<i>CppCodeGen</i>	Generation of C++ code for a given Schema Salad definition.

Functions

<code>replaceKeywords(s)</code>	Rename keywords that are reserved in C++.
<code>safename(name)</code>	Create a C++ safe name.
<code>safename2(name)</code>	Create a namespaced safename.
<code>split_name(s)</code>	
<code>split_field(s)</code>	
<code>isPrimitiveType(v)</code>	
<code>hasFieldValue(e, f, v)</code>	
<code>isRecordSchema(v)</code>	
<code>isEnumSchema(v)</code>	
<code>isArray(v)</code>	
<code>pred(i)</code>	
<code>isArraySchema(v)</code>	

`schema_salad.cpp_codegen.replaceKeywords(s)`

Rename keywords that are reserved in C++.

Parameters

s (*str*) –

Return type

str

`schema_salad.cpp_codegen.safename(name)`

Create a C++ safe name.

Parameters

name (*str*) –

Return type

str

`schema_salad.cpp_codegen.safename2(name)`

Create a namespaced safename.

Parameters

name (*Dict[str, str]*) –

Return type

str

`schema_salad.cpp_codegen.split_name(s)`

Parameters

s (*str*) –

Return type

Tuple[str, str]

schema_salad.cpp_codegen.split_field(*s*)

Parameters

s (*str*) –

Return type

Tuple[str, str, str]

class schema_salad.cpp_codegen.ClassDefinition(*name*)

Parameters

name (*str*) –

writeFwdDeclaration(*target*, *fullInd*, *ind*)

Parameters

- **target** (*IO*[*str*]) –
- **fullInd** (*str*) –
- **ind** (*str*) –

Return type

None

writeDefinition(*target*, *fullInd*, *ind*)

Parameters

- **target** (*IO*[*Any*]) –
- **fullInd** (*str*) –
- **ind** (*str*) –

Return type

None

writeImplDefinition(*target*, *fullInd*, *ind*)

Parameters

- **target** (*IO*[*str*]) –
- **fullInd** (*str*) –
- **ind** (*str*) –

Return type

None

class schema_salad.cpp_codegen.FieldDefinition(*name*, *typeStr*, *optional*)

Parameters

- **name** (*str*) –
- **typeStr** (*str*) –
- **optional** (*bool*) –

writeDefinition(*target*, *fullInd*, *ind*, *namespace*)

Write a C++ definition for the class field.

Parameters

- **target** (*IO*[*Any*]) –
- **fullInd** (*str*) –
- **ind** (*str*) –
- **namespace** (*str*) –

Return type

None

class `schema_salad.cpp_codegen.EnumDefinition`(*name*, *values*)

Parameters

- **name** (*str*) –
- **values** (*List*[*str*]) –

writeDefinition(*target*, *ind*)

Parameters

- **target** (*IO*[*str*]) –
- **ind** (*str*) –

Return type

None

`schema_salad.cpp_codegen.isPrimitiveType`(*v*)

Parameters

v (*Any*) –

Return type

`bool`

`schema_salad.cpp_codegen.hasFieldValue`(*e*, *f*, *v*)

Parameters

- **e** (*Any*) –
- **f** (*str*) –
- **v** (*Any*) –

Return type

`bool`

`schema_salad.cpp_codegen.isRecordSchema`(*v*)

Parameters

v (*Any*) –

Return type

`bool`

`schema_salad.cpp_codegen.isEnumSchema(v)`

Parameters

v (*Any*) –

Return type

`bool`

`schema_salad.cpp_codegen.isArray(v)`

Parameters

v (*Any*) –

Return type

`bool`

`schema_salad.cpp_codegen.pred(i)`

Parameters

i (*Any*) –

Return type

`bool`

`schema_salad.cpp_codegen.isArraySchema(v)`

Parameters

v (*Any*) –

Return type

`bool`

class `schema_salad.cpp_codegen.CppCodeGen`(*base, target, examples, package, copyright*)

Bases: `schema_salad.codegen_base.CodeGenBase`

Generation of C++ code for a given Schema Salad definition.

Parameters

- **base** (*str*) –
- **target** (*IO[str]*) –
- **examples** (*Optional[str]*) –
- **package** (*str*) –
- **copyright** (*Optional[str]*) –

convertTypeToCpp(*type_declaration*)

Convert a Schema Salad type to a C++ type.

Parameters

type_declaration (*Union[List[Any], Dict[str, Any], str]*) –

Return type

`str`

epilogue(*root_loader*)

Trigger to generate the epilogue code.

Parameters

root_loader (*Optional[schema_salad.codegen_base.TypeDef]*) –

Return type

None

parseRecordField(*field*)

Parameters

field (*Dict*[*str*, *Any*]) –

Return type

FieldDefinition

parseRecordSchema(*stype*)

Parameters

stype (*Dict*[*str*, *Any*]) –

Return type

None

parseEnum(*stype*)

Parameters

stype (*Dict*[*str*, *Any*]) –

Return type

str

parse(*items*)

Parameters

items (*List*[*Dict*[*str*, *Any*]]) –

Return type

None

`schema_salad.dlang_codegen`

D code generator for a given schema salad definition.

Module Contents

Classes

DLangCodeGen

Generation of D code for a given Schema Salad definition.

class `schema_salad.dlang_codegen.DLangCodeGen`(*base*, *target*, *examples*, *package*, *copyright_*, *parser_info*, *salad_version*)

Bases: `schema_salad.codegen_base.CodeGenBase`

Generation of D code for a given Schema Salad definition.

Parameters

- **base** (*str*) –

- **target** (*IO[str]*) –
- **examples** (*Optional[str]*) –
- **package** (*str*) –
- **copyright_** (*Optional[str]*) –
- **parser_info** (*Optional[str]*) –
- **salad_version** (*str*) –

prologue()

Trigger to generate the prologue code.

Return type

None

epilogue(*root_loader*)

Trigger to generate the epilogue code.

Parameters

root_loader (*schema_salad.codegen_base.TypeDef*) –

Return type

None

static safe_name(*name*)

Generate a safe version of the given name.

Parameters

name (*str*) –

Return type

str

to_doc_comment(*doc*)

Return an embedded documentation comments for a given string.

Parameters

doc (*Union[None, str, List[str]]*) –

Return type

str

parse_record_field_type(*type_*, *jsonld_pred*)

Return an annotation string and a type string.

Parameters

- **type_** (*Any*) –
- **jsonld_pred** (*Union[None, str, Dict[str, Any]]*) –

Return type

Tuple[str, str]

parse_record_field(*field*, *parent_name=None*)

Return a declaration string for a given record field.

Parameters

- **field** (*Dict[str, Any]*) –
- **parent_name** (*Optional[str]*) –

Return type

str

parse_record_schema(*stype*)

Return a declaration string for a given record schema.

Parameters

stype (*Dict[str, Any]*) –

Return type

str

parse_enum(*stype*)

Return a declaration string for a given enum schema.

Parameters

stype (*Dict[str, Any]*) –

Return type

str

parse(*items*)

Generate D code from items and write it to target.

Parameters

items (*List[Dict[str, Any]]*) –

Return type

None

schema_salad.dotnet_codegen

DotNet code generator for a given schema salad definition.

Module Contents

Classes

DotNetCodeGen

Generation of TypeScript code for a given Schema Salad definition.

Functions

doc_to_doc_string(doc[, indent_level])

Generate a documentation string from a schema salad doc field.

Attributes

prims

`schema_salad.dotnet_codegen.doc_to_doc_string(doc, indent_level=0)`

Generate a documentation string from a schema salad doc field.

Parameters

- **doc** (*Optional[str]*) –
- **indent_level** (*int*) –

Return type

str

`schema_salad.dotnet_codegen.prims`

class `schema_salad.dotnet_codegen.DotNetCodeGen`(*base, examples, target, package*)

Bases: `schema_salad.codegen_base.CodeGenBase`

Generation of TypeScript code for a given Schema Salad definition.

Parameters

- **base** (*str*) –
- **examples** (*Optional[str]*) –
- **target** (*Optional[str]*) –
- **package** (*str*) –

prologue()

Trigger to generate the prologue code.

Return type

None

static safe_name(*name*)

Generate a safe version of the given name.

Parameters

name (*str*) –

Return type

str

begin_class(*classname, extends, doc, abstract, field_names, idfield, optional_fields*)

Produce the header for the given class.

Parameters

- **classname** (*str*) –
- **extends** (*MutableSequence[str]*) –
- **doc** (*str*) –
- **abstract** (*bool*) –

- **field_names** (*MutableSequence*[*str*]) –
- **idfield** (*str*) –
- **optional_fields** (*Set*[*str*]) –

Return type

None

end_class(*classname*, *field_names*)

Signal that we are done with this class.

Parameters

- **classname** (*str*) –
- **field_names** (*List*[*str*]) –

Return type

None

type_loader(*type_declaration*)

Parse the given type declaration and declare its components.

Parameters

type_declaration (*Union*[*List*[*Any*], *Dict*[*str*, *Any*], *str*]) –

Return type

schema_salad.codegen_base.TypeDef

type_loader_enum(*type_declaration*)

Parameters

type_declaration (*Dict*[*str*, *Any*]) –

Return type

schema_salad.codegen_base.TypeDef

declare_field(*name*, *fieldtype*, *doc*, *optional*, *subscope*)

Output the code to load the given field.

Parameters

- **name** (*str*) –
- **fieldtype** (*schema_salad.codegen_base.TypeDef*) –
- **doc** (*Optional*[*str*]) –
- **optional** (*bool*) –
- **subscope** (*str*) –

Return type

None

declare_id_field(*name*, *fieldtype*, *doc*, *optional*)

Output the code to handle the given ID field.

Parameters

- **name** (*str*) –
- **fieldtype** (*schema_salad.codegen_base.TypeDef*) –
- **doc** (*str*) –

- **optional** (*bool*) –

Return type

None

to_dotnet (*val*)

Convert a Python keyword to a DotNet keyword.

Parameters

- **val** (*Any*) –

Return type

Any

uri_loader (*inner, scoped_id, vocab_term, ref_scope*)

Construct the TypeDef for the given URI loader.

Parameters

- **inner** (*schema_salad.codegen_base.TypeDef*) –
- **scoped_id** (*bool*) –
- **vocab_term** (*bool*) –
- **ref_scope** (*Optional[int]*) –

Return type*schema_salad.codegen_base.TypeDef***idmap_loader** (*field, inner, map_subject, map_predicate*)

Construct the TypeDef for the given mapped ID loader.

Parameters

- **field** (*str*) –
- **inner** (*schema_salad.codegen_base.TypeDef*) –
- **map_subject** (*str*) –
- **map_predicate** (*Optional[str]*) –

Return type*schema_salad.codegen_base.TypeDef***typedsl_loader** (*inner, ref_scope*)

Construct the TypeDef for the given DSL loader.

Parameters

- **inner** (*schema_salad.codegen_base.TypeDef*) –
- **ref_scope** (*Optional[int]*) –

Return type*schema_salad.codegen_base.TypeDef***epilogue** (*root_loader*)

Trigger to generate the epilogue code.

Parameters

- **root_loader** (*schema_salad.codegen_base.TypeDef*) –

Return type

None

secondaryfilesdsl_loader(*inner*)

Construct the TypeDef for secondary files.

Parameters

inner (`schema_salad.codegen_base.TypeDef`) –

Return type

`schema_salad.codegen_base.TypeDef`

schema_salad.exceptions

Shared Exception classes.

Module Contents

Functions

`to_one_line_messages`(*exc*)

exception `schema_salad.exceptions.SchemaSaladException`(*msg*, *sl=None*, *children=None*, *bullet_for_children=""*)

Bases: `Exception`

Base class for all schema-salad exceptions.

Parameters

- **msg** (*str*) –
- **sl** (*Optional* [`schema_salad.sourceline.SourceLine`]) –
- **children** (*Optional* [`Sequence` [`SchemaSaladException`]]) –
- **bullet_for_children** (*str*) –

propagate_sourceline()

Return type

`None`

as_warning()

Return type

`SchemaSaladException`

with_sourceline(*sl*)

Parameters

sl (*Optional* [`schema_salad.sourceline.SourceLine`]) –

Return type

`SchemaSaladException`

leaves()

Return type

List[*SchemaSaladException*]

prefix()

Return type

str

summary(*level=0, with_bullet=False*)

Parameters

- **level** (*int*) –
- **with_bullet** (*bool*) –

Return type

str

__str__()

Convert to a string using *pretty_str()*.

Return type

str

pretty_str(*level=0*)

Parameters

level (*int*) –

Return type

str

exception `schema_salad.exceptions.SchemaException`(*msg, sl=None, children=None, bullet_for_children=""*)

Bases: *SchemaSaladException*

Indicates error with the provided schema definition.

Parameters

- **msg** (*str*) –
- **sl** (*Optional[`schema_salad.sourceline.SourceLine`]*) –
- **children** (*Optional[`Sequence[SchemaSaladException]`]*) –
- **bullet_for_children** (*str*) –

exception `schema_salad.exceptions.ValidationException`(*msg, sl=None, children=None, bullet_for_children=""*)

Bases: *SchemaSaladException*

Indicates error with document against the provided schema.

Parameters

- **msg** (*str*) –
- **sl** (*Optional[`schema_salad.sourceline.SourceLine`]*) –

- **children** (*Optional*[*Sequence*[*SchemaSaladException*]]) –
- **bullet_for_children** (*str*) –

exception `schema_salad.exceptions.ClassValidationException`(*msg*, *sl=None*, *children=None*, *bullet_for_children=""*)

Bases: *ValidationException*

Indicates error with document against the provided schema.

Parameters

- **msg** (*str*) –
- **sl** (*Optional*[*schema_salad.sourceline.SourceLine*]) –
- **children** (*Optional*[*Sequence*[*SchemaSaladException*]]) –
- **bullet_for_children** (*str*) –

`schema_salad.exceptions.to_one_line_messages`(*exc*)

Parameters

exc (*SchemaSaladException*) –

Return type

str

`schema_salad.fetcher`

Resource fetching.

Module Contents

Classes

<i>Fetcher</i>	Fetch resources from URIs.
<i>MemoryCachingFetcher</i>	Fetcher that caches resources in memory after retrieval.
<i>DefaultFetcher</i>	The default Fetcher implementation.

class `schema_salad.fetcher.Fetcher`

Bases: `abc.ABC`

Fetch resources from URIs.

schemes = ['file', 'http', 'https', 'mailto']

abstract `fetch_text`(*url*, *content_types=None*)

Retrieve the given resource as a string.

Parameters

- **url** (*str*) –

- **content_types** (*Optional[List[str]*) –

Return type

str

abstract check_exists(*url*)

Check if the given resource exists.

Parameters

- **url** (*str*) –

Return type

bool

abstract urljoin(*base_url, url*)**Parameters**

- **base_url** (*str*) –
- **url** (*str*) –

Return type

str

supported_schemes()

Return the list of supported URI schemes.

Return type

List[str]

class `schema_salad.fetcher.MemoryCachingFetcher`(*cache*)Bases: *Fetcher*

Fetcher that caches resources in memory after retrieval.

Parameters

- **cache** (*schema_salad.utils.CacheType*) –

class `schema_salad.fetcher.DefaultFetcher`(*cache, session*)Bases: *MemoryCachingFetcher*

The default Fetcher implementation.

Parameters

- **cache** (*schema_salad.utils.CacheType*) –
- **session** (*Optional[requests.sessions.Session]*) –

fetch_text(*url, content_types=None*)

Retrieve the given resource as a string.

Parameters

- **url** (*str*) –
- **content_types** (*Optional[List[str]*) –

Return type

str

check_exists(*url*)

Check if the given resource exists.

Parameters

url (*str*) –

Return type

bool

urljoin(*base_url*, *url*)

Parameters

• **base_url** (*str*) –

• **url** (*str*) –

Return type

str

schema_salad.java_codegen

Java code generator for a given schema salad definition.

Module Contents

Classes

JavaCodeGen

Abstract base class for schema salad code generators.

Functions

doc_to_doc_string(doc[, indent_level])

Attributes

USE_ONE_OR_LIST_OF_TYPES

BASIC_JAVA_IDENTIFIER_RE

prims

schema_salad.java_codegen.USE_ONE_OR_LIST_OF_TYPES = False

schema_salad.java_codegen.BASIC_JAVA_IDENTIFIER_RE

`schema_salad.java_codegen.doc_to_doc_string(doc, indent_level=0)`

Parameters

- **doc** (*Optional[str]*) –
- **indent_level** (*int*) –

Return type

str

`schema_salad.java_codegen.prims`

class `schema_salad.java_codegen.JavaCodeGen`(*base, target, examples, package, copyright*)

Bases: *schema_salad.codegen_base.CodeGenBase*

Abstract base class for schema salad code generators.

Parameters

- **base** (*str*) –
- **target** (*Optional[str]*) –
- **examples** (*Optional[str]*) –
- **package** (*str*) –
- **copyright** (*Optional[str]*) –

prologue()

Trigger to generate the prologue code.

Return type

None

static `property_name`(*name*)

Parameters

name (*str*) –

Return type

str

static `safe_name`(*name*)

Generate a safe version of the given name.

Parameters

name (*str*) –

Return type

str

interface_name(*n*)

Parameters

n (*str*) –

Return type

str

begin_class(*classname*, *extends*, *doc*, *abstract*, *field_names*, *idfield*, *optional_fields*)

Produce the header for the given class.

Parameters

- **classname** (*str*) –
- **extends** (*MutableSequence[str]*) –
- **doc** (*str*) –
- **abstract** (*bool*) –
- **field_names** (*MutableSequence[str]*) –
- **idfield** (*str*) –
- **optional_fields** (*Set[str]*) –

Return type

None

end_class(*classname*, *field_names*)

Finish this class.

Parameters

- **classname** (*str*) –
- **field_names** (*List[str]*) –

Return type

None

type_loader(*type_declaration*)

Parse the given type declaration and declare its components.

Parameters

type_declaration (*Union[List[Any], Dict[str, Any], str]*) –

Return type

schema_salad.codegen_base.TypeDef

type_loader_enum(*type_declaration*)

Parameters

type_declaration (*Dict[str, Any]*) –

Return type

schema_salad.codegen_base.TypeDef

declare_field(*name*, *fieldtype*, *doc*, *optional*, *subscope*)

Output the code to load the given field.

Parameters

- **name** (*str*) –
- **fieldtype** (*schema_salad.codegen_base.TypeDef*) –
- **doc** (*Optional[str]*) –
- **optional** (*bool*) –
- **subscope** (*str*) –

Return type

None

declare_id_field(*name*, *fieldtype*, *doc*, *optional*)

Output the code to handle the given ID field.

Parameters

- **name** (*str*) –
- **fieldtype** (`schema_salad.codegen_base.TypeDef`) –
- **doc** (*str*) –
- **optional** (*bool*) –

Return type

None

uri_loader(*inner*, *scoped_id*, *vocab_term*, *ref_scope*)

Construct the TypeDef for the given URI loader.

Parameters

- **inner** (`schema_salad.codegen_base.TypeDef`) –
- **scoped_id** (*bool*) –
- **vocab_term** (*bool*) –
- **ref_scope** (`Optional[int]`) –

Return type`schema_salad.codegen_base.TypeDef`**idmap_loader**(*field*, *inner*, *map_subject*, *map_predicate*)

Construct the TypeDef for the given mapped ID loader.

Parameters

- **field** (*str*) –
- **inner** (`schema_salad.codegen_base.TypeDef`) –
- **map_subject** (*str*) –
- **map_predicate** (`Optional[str]`) –

Return type`schema_salad.codegen_base.TypeDef`**typedsl_loader**(*inner*, *ref_scope*)

Construct the TypeDef for the given DSL loader.

Parameters

- **inner** (`schema_salad.codegen_base.TypeDef`) –
- **ref_scope** (`Union[int, None]`) –

Return type`schema_salad.codegen_base.TypeDef`

`to_java(val)`

Parameters

`val` (*Any*) –

Return type

Any

`epilogue(root_loader)`

Trigger to generate the epilogue code.

Parameters

`root_loader` (`schema_salad.codegen_base.TypeDef`) –

Return type

None

`secondaryfilesdsl_loader(inner)`

Construct the TypeDef for secondary files.

Parameters

`inner` (`schema_salad.codegen_base.TypeDef`) –

Return type

`schema_salad.codegen_base.TypeDef`

`schema_salad.jsonld_context`

Module Contents

Functions

`pred`(datatype, field, name, context, defaultBase, ...)

`process_type`(t, g, context, defaultBase, namespaces, ...)

`salad_to_jsonld_context`(j, schema_ctx)

`fix_jsonld_ids`(obj, ids)

Add missing identity entries.

`makerdf`(workflow, wf, ctx[, graph])

`schema_salad.jsonld_context.pred`(*datatype, field, name, context, defaultBase, namespaces*)

Parameters

- `datatype` (`MutableMapping[str, Union[Dict[str, str], str]]`) –
- `field` (`Optional[Dict[str, Any]]`) –
- `name` (`str`) –
- `context` (`schema_salad.utils.ContextType`) –
- `defaultBase` (`str`) –
- `namespaces` (`Dict[str, rdflib.namespace.Namespace]`) –

Return type

Union[Dict[str, Union[str, None]], str]

schema_salad.jsonld_context.**process_type**(t, g, context, defaultBase, namespaces, defaultPrefix)**Parameters**

- **t** (*MutableMapping*[str, Any]) –
- **g** (*rdflib.Graph*) –
- **context** (*schema_salad.utils.ContextType*) –
- **defaultBase** (str) –
- **namespaces** (*Dict*[str, *rdflib.namespace.Namespace*]) –
- **defaultPrefix** (str) –

Return type

None

schema_salad.jsonld_context.**salad_to_jsonld_context**(j, schema_ctx)**Parameters**

- **j** (*Iterable*[*MutableMapping*[str, Any]]) –
- **schema_ctx** (*MutableMapping*[str, Any]) –

Return typeTuple[schema_salad.utils.ContextType, *rdflib.Graph*]schema_salad.jsonld_context.**fix_jsonld_ids**(obj, ids)

Add missing identity entries.

Parameters

- **obj** (*Union*[*ruamel.yaml.comments.CommentedMap*, float, str, *ruamel.yaml.comments.CommentedSeq*]) –
- **ids** (*List*[str]) –

Return type

None

schema_salad.jsonld_context.**makerdf**(workflow, wf, ctx, graph=None)**Parameters**

- **workflow** (*Optional*[str]) –
- **wf** (*Union*[*ruamel.yaml.comments.CommentedMap*, float, str, *ruamel.yaml.comments.CommentedSeq*]) –
- **ctx** (*schema_salad.utils.ContextType*) –
- **graph** (*Optional*[*rdflib.Graph*]) –

Return type*rdflib.Graph*

schema_salad.main

Command line interface to schema-salad.

Module Contents

Functions

printrdf(workflow, wf, ctx, sr)

arg_parser()

Build the argument parser.

main([argsl])

schema_salad.main.**printrdf**(*workflow*, *wf*, *ctx*, *sr*)

Parameters

- **workflow** (*str*) –
- **wf** (*Union[ruamel.yaml.comments.CommentedMap, ruamel.yaml.comments.CommentedSeq]*) –
- **ctx** (*Dict[str, Any]*) –
- **sr** (*str*) –

Return type

None

schema_salad.main.**arg_parser**()

Build the argument parser.

Return type

argparse.ArgumentParser

schema_salad.main.**main**(*argsl=None*)

Parameters

argsl (*Optional[List[str]]*) –

Return type

int

schema_salad.makedoc

Module Contents

Classes

<i>MyRenderer</i>	Custom renderer with different representations of selected HTML tags.
<i>ToC</i>	
<i>RenderType</i>	

Functions

<i>vocab_type_name</i> (url)	Remove the avro namespace, if any.
<i>has_types</i> (items)	
<i>linkto</i> (item)	
<i>markdown_list_hook</i> (markdown, text, state)	Patches problematic Markdown lists for later HTML generation.
<i>patch_fenced_code</i> (original_markdown_text, ...)	Reverts fenced code fragments found in the modified contents back to their original definition.
<i>to_id</i> (text)	
<i>number_headings</i> (toc, maindoc)	
<i>fix_doc</i> (doc)	
<i>avrold_doc</i> (j, outdoc, renderlist, redirects, brand, ...)	
<i>arg_parser</i> ()	Build the argument parser.
<i>main</i> ()	Shortcut entrypoint.
<i>makedoc</i> (stdout, schema[, redirects, only, brand, ...])	Emit HTML representation of a given schema.

Attributes

<i>basicTypes</i>

`schema_salad.makedoc.vocab_type_name(url)`

Remove the avro namespace, if any.

Parameters

url (*str*) –

Return type

str

`schema_salad.makedoc.has_types(items)`

Parameters

items (*Any*) –

Return type

List[str]

schema_salad.makedoc.**linkto**(*item*)

Parameters

item (*str*) –

Return type

str

class schema_salad.makedoc.**MyRenderer**(*escape=True, allow_harmful_protocols=None*)

Bases: mistune.renderers.HTMLRenderer

Custom renderer with different representations of selected HTML tags.

heading(*text, level*)

Override HTML heading creation with text IDs.

Parameters

- **text** (*str*) –
- **level** (*int*) –

Return type

str

text(*text*)

Don't escape quotation marks.

Parameters

text (*str*) –

Return type

str

inline_html(*html*)

Don't escape characters in predefined HTML within paragraph tags.

Parameters

html (*str*) –

Return type

str

block_html(*html*)

Don't escape characters nor wrap predefined HTML within paragraph tags.

Parameters

html (*str*) –

Return type

str

block_code(*code, info=None*)

Don't escape quotation marks.

Parameters

- **code** (*str*) –
- **info** (*Optional[str]*) –

Return type

str

schema_salad.makedoc.**markdown_list_hook**(*markdown, text, state*)

Patches problematic Markdown lists for later HTML generation.

When a Markdown list with paragraphs not indented with the list markers (no spaces before following lines), `mistune v2` does not handle them correctly. This is however permitted as per <https://daringfireball.net/projects/markdown/syntax#list>

For example:

```
`markdown * some list * item with paragraph * other item`
```

Similarly, lists that are completely indented or that contains nested lists produce incorrect HTML `<p>/` tag combinations.

Because list parsing is deeply nested within `mistune.block_parser.BlockParser` and that there is no easy way to override utility functions it employs to adjust patterns of list items without reimplementing it or a lot of monkey patching, instead catch the problem cases before rendering and adjust them with a hook.

See <https://github.com/lepture/mistune/issues/296> and https://github.com/common-workflow-language/schema_salad/pull/619

Parameters

- **markdown** (*Markdown[str, Any]*) –
- **text** (*str*) –
- **state** (*mistune.State*) –

Return type

Tuple[str, mistune.State]

schema_salad.makedoc.**patch_fenced_code**(*original_markdown_text, modified_markdown_text*)

Reverts fenced code fragments found in the modified contents back to their original definition.

Parameters

- **original_markdown_text** (*str*) –
- **modified_markdown_text** (*str*) –

Return type

str

schema_salad.makedoc.**to_id**(*text*)

Parameters

text (*str*) –

Return type

str

class schema_salad.makedoc.**ToC**

add_entry(*thisdepth, title*)

Add an entry to the table of contents.

Parameters

- **thisdepth** (*int*) –
- **title** (*str*) –

Return type

str

contents (*idn*)

Parameters

idn (*str*) –

Return type

str

```
schema_salad.makedoc.basicTypes = ('https://w3id.org/cwl/salad#null',
'http://www.w3.org/2001/XMLSchema#boolean',...
```

```
schema_salad.makedoc.number_headings(toc, maindoc)
```

Parameters

- **toc** (*ToC*) –
- **maindoc** (*str*) –

Return type

str

```
schema_salad.makedoc.fix_doc(doc)
```

Parameters

doc (*Union[List[str], str]*) –

Return type

str

```
class schema_salad.makedoc.RenderType(toc, j, renderlist, redirects, primitiveType)
```

Parameters

- **toc** (*ToC*) –
- **j** (*List[Dict[str, Any]]*) –
- **renderlist** (*List[str]*) –
- **redirects** (*Dict[str, str]*) –
- **primitiveType** (*str*) –

```
typefmt(tp, redirects, nbsp=False, jsonldPredicate=None)
```

Parameters

- **tp** (*Any*) –
- **redirects** (*Dict[str, str]*) –
- **nbsp** (*bool*) –
- **jsonldPredicate** (*Optional[Union[Dict[str, str], str]]*) –

Return type

str

render_type(*f*, *depth*)

Parameters

- **f** (*Dict*[*str*, *Any*]) –
- **depth** (*int*) –

Return type

None

`schema_salad.makedoc.avrold_doc`(*j*, *outdoc*, *renderlist*, *redirects*, *brand*, *brandlink*, *primetype*, *brandstyle=None*, *brandinverse=False*)

Parameters

- **j** (*List*[*Dict*[*str*, *Any*]]) –
- **outdoc** (*IO*[*Any*]) –
- **renderlist** (*List*[*str*]) –
- **redirects** (*Dict*[*str*, *str*]) –
- **brand** (*str*) –
- **brandlink** (*str*) –
- **primetype** (*str*) –
- **brandstyle** (*Optional*[*str*]) –
- **brandinverse** (*Optional*[*bool*]) –

Return type

None

`schema_salad.makedoc.arg_parser`()

Build the argument parser.

Return type

`argparse.ArgumentParser`

`schema_salad.makedoc.main`()

Shortcut entrypoint.

Return type

None

`schema_salad.makedoc.makedoc`(*stdout*, *schema*, *redirects=None*, *only=None*, *brand=None*, *brandlink=None*, *primetype=None*, *brandstyle=None*, *brandinverse=False*)

Emit HTML representation of a given schema.

Parameters

- **stdout** (*IO*[*Any*]) –
- **schema** (*str*) –
- **redirects** (*Optional*[*List*[*str*]]) –
- **only** (*Optional*[*List*[*str*]]) –
- **brand** (*Optional*[*str*]) –
- **brandlink** (*Optional*[*str*]) –

- **primetype** (*Optional[str]*) –
- **brandstyle** (*Optional[str]*) –
- **brandinverse** (*Optional[bool]*) –

Return type

None

`schema_salad.metaschema`

Module Contents

Classes

<i>LoadingOptions</i>	
<i>Saveable</i>	Mark classes than have a <code>save()</code> and <code>fromDoc()</code> function.
<i>Documented</i>	Mark classes than have a <code>save()</code> and <code>fromDoc()</code> function.
<i>RecordField</i>	A field of a record.
<i>RecordSchema</i>	Mark classes than have a <code>save()</code> and <code>fromDoc()</code> function.
<i>EnumSchema</i>	Define an enumerated type.
<i>ArraySchema</i>	Mark classes than have a <code>save()</code> and <code>fromDoc()</code> function.
<i>JsonldPredicate</i>	Attached to a record field to define how the parent record field is handled for
<i>SpecializeDef</i>	Mark classes than have a <code>save()</code> and <code>fromDoc()</code> function.
<i>NamedType</i>	Mark classes than have a <code>save()</code> and <code>fromDoc()</code> function.
<i>DocType</i>	Mark classes than have a <code>save()</code> and <code>fromDoc()</code> function.
<i>SchemaDefinedType</i>	Abstract base for schema-defined types.
<i>SaladRecordField</i>	A field of a record.
<i>SaladRecordSchema</i>	Mark classes than have a <code>save()</code> and <code>fromDoc()</code> function.
<i>SaladEnumSchema</i>	Define an enumerated type.
<i>Documentation</i>	A documentation section. This type exists to facilitate self-documenting

Functions

<code>load_field(val, fieldtype, baseuri, loadingOptions)</code>	
<code>save(val[, top, base_url, relative_uris])</code>	
<code>save_with_metadata(val, valLoadingOpts[, top, ...])</code>	Save and set \$namespaces, \$schemas, \$base and any other metadata fields at the top level.
<code>expand_url(url, base_url, loadingOptions[, scoped_id, ...])</code>	
<code>file_uri(path[, split_frag])</code>	
<code>prefix_url(url, namespaces)</code>	Expand short forms into full URLs using the given namespace dictionary.
<code>save_relative_uri(uri, base_url, scoped_id, ref_scope, ...)</code>	Convert any URI to a relative one, obeying the scoping rules.
<code>shortname(inputid)</code>	Compute the shortname of a fully qualified identifier.
<code>parser_info()</code>	
<code>load_document(doc[, baseuri, loadingOptions])</code>	
<code>load_document_with_metadata(doc[, baseuri, ...])</code>	
<code>load_document_by_string(string, uri[, loadingOptions])</code>	
<code>load_document_by_yaml(yaml, uri[, loadingOptions])</code>	Shortcut to load via a YAML object.

Attributes

<code>IdxType</code>	
<code>save_type</code>	
<code>strtype</code>	
<code>inttype</code>	
<code>floattype</code>	
<code>booltype</code>	
<code>None_type</code>	
<code>Any_type</code>	
<code>PrimitiveTypeLoader</code>	Names of salad data types (based on Avro schema declarations).
<code>AnyLoader</code>	The Any type validates for any non-null value.

continues on next page

Table 2 – continued from previous page

<i>RecordFieldLoader</i>	
<i>RecordSchemaLoader</i>	
<i>EnumSchemaLoader</i>	
<i>ArraySchemaLoader</i>	
<i>JsonldPredicateLoader</i>	
<i>SpecializeDefLoader</i>	
<i>SaladRecordFieldLoader</i>	
<i>SaladRecordSchemaLoader</i>	
<i>SaladEnumSchemaLoader</i>	
<i>DocumentationLoader</i>	
<i>array_of_strtype</i>	
<i>union_of_None_type_or_strtype_or_array_of_si</i>	
<i>uri_strtype_True_False_None</i>	
<i>union_of_PrimitiveTypeLoader_or_RecordSchem</i>	<i>der_or_strtype</i>
<i>array_of_union_of_PrimitiveTypeLoader_or_Rec</i>	<i>SchemaLoader_o</i>
<i>union_of_PrimitiveTypeLoader_or_RecordSchem</i>	<i>der_or_strtype</i>
<i>typedsl_union_of_PrimitiveTypeLoader_or_Rec</i>	<i>chemaLoader_or</i>
<i>array_of_RecordFieldLoader</i>	
<i>union_of_None_type_or_array_of_RecordFieldL</i>	
<i>idmap_fields_union_of_None_type_or_array_of</i>	
<i>Record_nameLoader</i>	
<i>typedsl_Record_nameLoader_2</i>	
<i>union_of_None_type_or_strtype</i>	
<i>uri_union_of_None_type_or_strtype_True_False</i>	
<i>uri_array_of_strtype_True_False_None</i>	
<i>Enum_nameLoader</i>	

continues on next page

Table 2 – continued from previous page

<code>typedsl_Enum_nameLoader_2</code>	
<code>uri_union_of_PrimitiveTypeLoader_or_RecordSc</code>	<code>aLoader_or_str</code>
<code>Array_nameLoader</code>	
<code>typedsl_Array_nameLoader_2</code>	
<code>union_of_None_type_or_booltype</code>	
<code>union_of_None_type_or_inttype</code>	
<code>uri_strtype_False_False_1</code>	
<code>uri_union_of_None_type_or_strtype_False_Fals</code>	
<code>uri_union_of_None_type_or_strtype_or_array_</code>	
<code>union_of_None_type_or_strtype_or_JsonldPred:</code>	
<code>union_of_None_type_or_Any_type</code>	
<code>array_of_SaladRecordFieldLoader</code>	
<code>union_of_None_type_or_array_of_SaladRecordFi</code>	
<code>idmap_fields_union_of_None_type_or_array_of_</code>	
<code>uri_union_of_None_type_or_strtype_or_array_</code>	
<code>array_of_SpecializeDefLoader</code>	
<code>union_of_None_type_or_array_of_SpecializeDe:</code>	
<code>idmap_specialize_union_of_None_type_or_array</code>	
<code>Documentation_nameLoader</code>	
<code>typedsl_Documentation_nameLoader_2</code>	
<code>union_of_SaladRecordSchemaLoader_or_SaladEn</code>	
<code>array_of_union_of_SaladRecordSchemaLoader_or</code>	<code>r</code>
<code>union_of_SaladRecordSchemaLoader_or_SaladEn</code>	<code>y_of_union_of_</code>

`schema_salad.metaschema.IdxType`

class `schema_salad.metaschema.LoadingOptions`(*fetcher=None, namespaces=None, schemas=None, fileuri=None, copyfrom=None, original_doc=None, addl_metadata=None, baseuri=None, idx=None, imports=None, includes=None*)

Parameters

- **fetcher** (*Optional*[*schema_salad.fetcher.Fetcher*]) –
- **namespaces** (*Optional*[*Dict*[*str*, *str*]]) –
- **schemas** (*Optional*[*List*[*str*]]) –
- **fileuri** (*Optional*[*str*]) –
- **copyfrom** (*Optional*[*LoadingOptions*]) –
- **original_doc** (*Optional*[*Any*]) –
- **addl_metadata** (*Optional*[*Dict*[*str*, *str*]]) –
- **baseuri** (*Optional*[*str*]) –
- **idx** (*Optional*[*IdxType*]) –
- **imports** (*Optional*[*List*[*str*]]) –
- **includes** (*Optional*[*List*[*str*]]) –

property graph: `rdflib.Graph`

Generate a merged rdflib.Graph from all entries in self.schemas.

Return type

`rdflib.Graph`

idx: `IdxType`

fileuri: `str | None`

baseuri: `str`

namespaces: `MutableMapping[str, str]`

schemas: `MutableSequence[str]`

original_doc: `Any | None`

addl_metadata: `MutableMapping[str, Any]`

fetcher: `schema_salad.fetcher.Fetcher`

vocab: `Dict[str, str]`

rvocab: `Dict[str, str]`

cache: `schema_salad.utils.CacheType`

imports: `List[str]`

includes: `List[str]`

class `schema_salad.metaschema.Saveable`

Bases: `abc.ABC`

Mark classes than have a `save()` and `fromDoc()` function.

abstract classmethod fromDoc(*_doc*, *baseuri*, *loadingOptions*, *docRoot=None*)

Construct this object from the result of `yaml.load()`.

Parameters

- **_doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **docRoot** (*Optional[str]*) –

Return type

Saveable

abstract save(*top=False*, *base_url=""*, *relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

Dict[str, Any]

`schema_salad.metaschema.load_field`(*val*, *fieldtype*, *baseuri*, *loadingOptions*)

Parameters

- **val** (*Union[str, Dict[str, str]]*) –
- **fieldtype** (*_Loader*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –

Return type

Any

`schema_salad.metaschema.save_type`

`schema_salad.metaschema.save`(*val*, *top=True*, *base_url=""*, *relative_uris=True*)

Parameters

- **val** (*Any*) –
- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

save_type

`schema_salad.metaschema.save_with_metadata`(*val*, *valLoadingOpts*, *top=True*, *base_url=""*, *relative_uris=True*)

Save and set \$namespaces, \$schemas, \$base and any other metadata fields at the top level.

Parameters

- **val** (*Any*) –
- **valLoadingOpts** (*LoadingOptions*) –
- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

save_type

`schema_salad.metaschema.expand_url(url, base_url, loadingOptions, scoped_id=False, vocab_term=False, scoped_ref=None)`

Parameters

- **url** (*str*) –
- **base_url** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **scoped_id** (*bool*) –
- **vocab_term** (*bool*) –
- **scoped_ref** (*Optional[int]*) –

Return type

str

`schema_salad.metaschema.file_uri(path, split_frag=False)`

Parameters

- **path** (*str*) –
- **split_frag** (*bool*) –

Return type

str

`schema_salad.metaschema.prefix_url(url, namespaces)`

Expand short forms into full URLs using the given namespace dictionary.

Parameters

- **url** (*str*) –
- **namespaces** (*Dict[str, str]*) –

Return type

str

`schema_salad.metaschema.save_relative_uri(uri, base_url, scoped_id, ref_scope, relative_uris)`

Convert any URI to a relative one, obeying the scoping rules.

Parameters

- **uri** (*Any*) –
- **base_url** (*str*) –
- **scoped_id** (*bool*) –
- **ref_scope** (*Optional[int]*) –

- **relative_uris** (*bool*) –

Return type

Any

`schema_salad.metaschema.shortname(inputid)`

Compute the shortname of a fully qualified identifier.

See https://w3id.org/cwl/v1.2/SchemaSalad.html#Short_names.

Parameters

inputid (*str*) –

Return type

str

`schema_salad.metaschema.parser_info()`

Return type

str

class `schema_salad.metaschema.Documented`

Bases: *Saveable*

Mark classes than have a `save()` and `fromDoc()` function.

class `schema_salad.metaschema.RecordField`(*name, type, doc=None, extension_fields=None, loadingOptions=None*)

Bases: *Documented*

A field of a record.

Parameters

- **name** (*Any*) –
- **type** (*Any*) –
- **doc** (*Optional[Any]*) –
- **extension_fields** (*Optional[Dict[str, Any]]*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –

attrs

`__eq__(other)`

Return `self==value`.

Parameters

other (*Any*) –

Return type

bool

`__hash__()`

Return `hash(self)`.

Return type

int

classmethod `fromDoc(doc, baseuri, loadingOptions, docRoot=None)`

Construct this object from the result of `yaml.load()`.

Parameters

- **doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **docRoot** (*Optional[str]*) –

Return type

RecordField

save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

Dict[str, Any]

class `schema_salad.metaschema.RecordSchema`(*type, fields=None, extension_fields=None, loadingOptions=None*)

Bases: *Saveable*

Mark classes that have a `save()` and `fromDoc()` function.

Parameters

- **type** (*Any*) –
- **fields** (*Optional[Any]*) –
- **extension_fields** (*Optional[Dict[str, Any]]*) –
- **loadingOptions** (*Optional>LoadingOptions*) –

attrs

`__eq__`(*other*)

Return `self==value`.

Parameters

other (*Any*) –

Return type

bool

`__hash__`()

Return `hash(self)`.

Return type

int

classmethod `fromDoc(doc, baseuri, loadingOptions, docRoot=None)`

Construct this object from the result of `yaml.load()`.

Parameters

- **doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **docRoot** (*Optional[str]*) –

Return type

RecordSchema

save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

Dict[str, Any]

class `schema_salad.metaschema.EnumSchema(symbols, type, name=None, extension_fields=None, loadingOptions=None)`

Bases: *Saveable*

Define an enumerated type.

Parameters

- **symbols** (*Any*) –
- **type** (*Any*) –
- **name** (*Optional[Any]*) –
- **extension_fields** (*Optional[Dict[str, Any]]*) –
- **loadingOptions** (*Optional>LoadingOptions*) –

attrs

`__eq__(other)`

Return `self==value`.

Parameters

other (*Any*) –

Return type

bool

`__hash__()`

Return `hash(self)`.

Return type

int

classmethod `fromDoc(doc, baseuri, loadingOptions, docRoot=None)`

Construct this object from the result of `yaml.load()`.

Parameters

- `doc` (*Any*) –
- `baseuri` (*str*) –
- `loadingOptions` (*LoadingOptions*) –
- `docRoot` (*Optional[str]*) –

Return type

EnumSchema

save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- `top` (*bool*) –
- `base_url` (*str*) –
- `relative_uris` (*bool*) –

Return type

Dict[str, Any]

class `schema_salad.metaschema.ArraySchema(items, type, extension_fields=None, loadingOptions=None)`

Bases: *Saveable*

Mark classes that have a `save()` and `fromDoc()` function.

Parameters

- `items` (*Any*) –
- `type` (*Any*) –
- `extension_fields` (*Optional[Dict[str, Any]]*) –
- `loadingOptions` (*Optional>LoadingOptions*) –

attrs

`__eq__`(*other*)

Return `self==value`.

Parameters

`other` (*Any*) –

Return type

bool

`__hash__`()

Return `hash(self)`.

Return type

int

classmethod `fromDoc(doc, baseuri, loadingOptions, docRoot=None)`

Construct this object from the result of `yaml.load()`.

Parameters

- **doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **docRoot** (*Optional[str]*) –

Return type

ArraySchema

save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

Dict[str, Any]

class `schema_salad.metaschema.JsonldPredicate`(*_id=None, _type=None, _container=None, identity=None, noLinkCheck=None, mapSubject=None, mapPredicate=None, refScope=None, typeDSL=None, secondaryFilesDSL=None, subscope=None, extension_fields=None, loadingOptions=None*)

Bases: *Saveable*

Attached to a record field to define how the parent record field is handled for URI resolution and JSON-LD context generation.

Parameters

- **_id** (*Optional[Any]*) –
- **_type** (*Optional[Any]*) –
- **_container** (*Optional[Any]*) –
- **identity** (*Optional[Any]*) –
- **noLinkCheck** (*Optional[Any]*) –
- **mapSubject** (*Optional[Any]*) –
- **mapPredicate** (*Optional[Any]*) –
- **refScope** (*Optional[Any]*) –
- **typeDSL** (*Optional[Any]*) –
- **secondaryFilesDSL** (*Optional[Any]*) –
- **subscope** (*Optional[Any]*) –
- **extension_fields** (*Optional[Dict[str, Any]]*) –

- **loadingOptions** (*Optional[LoadingOptions]*) –

attrs

__eq__(*other*)

Return self==value.

Parameters

- **other** (*Any*) –

Return type

bool

__hash__()

Return hash(self).

Return type

int

classmethod fromDoc(*doc, baseuri, loadingOptions, docRoot=None*)

Construct this object from the result of `yaml.load()`.

Parameters

- **doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **docRoot** (*Optional[str]*) –

Return type

JsonldPredicate

save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

Dict[str, Any]

class `schema_salad.metaschema.SpecializeDef`(*specializeFrom, specializeTo, extension_fields=None, loadingOptions=None*)

Bases: *Saveable*

Mark classes than have a `save()` and `fromDoc()` function.

Parameters

- **specializeFrom** (*Any*) –
- **specializeTo** (*Any*) –
- **extension_fields** (*Optional[Dict[str, Any]]*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –

attrs**__eq__**(*other*)

Return self==value.

Parameters**other** (*Any*) –**Return type**`bool`**__hash__**()

Return hash(self).

Return type`int`**classmethod** **fromDoc**(*doc*, *baseuri*, *loadingOptions*, *docRoot=None*)Construct this object from the result of `yaml.load()`.**Parameters**

- **doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (`LoadingOptions`) –
- **docRoot** (`Optional[str]`) –

Return type`SpecializeDef`**save**(*top=False*, *base_url=""*, *relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (`bool`) –
- **base_url** (*str*) –
- **relative_uris** (`bool`) –

Return type`Dict[str, Any]`**class** `schema_salad.metaschema.NamedType`Bases: `Saveable`Mark classes than have a `save()` and `fromDoc()` function.**class** `schema_salad.metaschema.DocType`Bases: `Documented`Mark classes than have a `save()` and `fromDoc()` function.

class `schema_salad.metaschema.SchemaDefinedType`

Bases: *DocType*

Abstract base for schema-defined types.

class `schema_salad.metaschema.SaladRecordField`(*name, type, doc=None, jsonldPredicate=None, default=None, extension_fields=None, loadingOptions=None*)

Bases: *RecordField*

A field of a record.

Parameters

- **name** (*Any*) –
- **type** (*Any*) –
- **doc** (*Optional[Any]*) –
- **jsonldPredicate** (*Optional[Any]*) –
- **default** (*Optional[Any]*) –
- **extension_fields** (*Optional[Dict[str, Any]]*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –

attrs

__eq__(*other*)

Return self==value.

Parameters

other (*Any*) –

Return type

bool

__hash__()

Return hash(self).

Return type

int

classmethod **fromDoc**(*doc, baseuri, loadingOptions, docRoot=None*)

Construct this object from the result of `yaml.load()`.

Parameters

- **doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **docRoot** (*Optional[str]*) –

Return type

SaladRecordField

save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

Dict[str, Any]

class `schema_salad.metaschema.SaladRecordSchema`(*name, type, inVocab=None, fields=None, doc=None, docParent=None, docChild=None, docAfter=None, jsonldPredicate=None, documentRoot=None, abstract=None, extends=None, specialize=None, extension_fields=None, loadingOptions=None*)

Bases: *NamedType, RecordSchema, SchemaDefinedType*

Mark classes that have a `save()` and `fromDoc()` function.

Parameters

- **name** (*Any*) –
- **type** (*Any*) –
- **inVocab** (*Optional[Any]*) –
- **fields** (*Optional[Any]*) –
- **doc** (*Optional[Any]*) –
- **docParent** (*Optional[Any]*) –
- **docChild** (*Optional[Any]*) –
- **docAfter** (*Optional[Any]*) –
- **jsonldPredicate** (*Optional[Any]*) –
- **documentRoot** (*Optional[Any]*) –
- **abstract** (*Optional[Any]*) –
- **extends** (*Optional[Any]*) –
- **specialize** (*Optional[Any]*) –
- **extension_fields** (*Optional[Dict[str, Any]]*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –

attrs

__eq__(*other*)

Return self==value.

Parameters

- **other** (*Any*) –

Return type

bool

`__hash__()`

Return hash(self).

Return type

`int`

classmethod `fromDoc(doc, baseuri, loadingOptions, docRoot=None)`

Construct this object from the result of `yaml.load()`.

Parameters

- `doc` (*Any*) –
- `baseuri` (*str*) –
- `loadingOptions` (*LoadingOptions*) –
- `docRoot` (*Optional[str]*) –

Return type

SaladRecordSchema

save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- `top` (*bool*) –
- `base_url` (*str*) –
- `relative_uris` (*bool*) –

Return type

`Dict[str, Any]`

class `schema_salad.metaschema.SaladEnumSchema`(*symbols, type, name=None, inVocab=None, doc=None, docParent=None, docChild=None, docAfter=None, jsonldPredicate=None, documentRoot=None, extends=None, extension_fields=None, loadingOptions=None*)

Bases: *NamedType, EnumSchema, SchemaDefinedType*

Define an enumerated type.

Parameters

- `symbols` (*Any*) –
- `type` (*Any*) –
- `name` (*Optional[Any]*) –
- `inVocab` (*Optional[Any]*) –
- `doc` (*Optional[Any]*) –
- `docParent` (*Optional[Any]*) –
- `docChild` (*Optional[Any]*) –
- `docAfter` (*Optional[Any]*) –
- `jsonldPredicate` (*Optional[Any]*) –

- **documentRoot** (*Optional[Any]*) –
- **extends** (*Optional[Any]*) –
- **extension_fields** (*Optional[Dict[str, Any]]*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –

attrs

__eq__(*other*)

Return self==value.

Parameters

other (*Any*) –

Return type

bool

__hash__()

Return hash(self).

Return type

int

classmethod fromDoc(*doc, baseuri, loadingOptions, docRoot=None*)

Construct this object from the result of `yaml.load()`.

Parameters

- **doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **docRoot** (*Optional[str]*) –

Return type

SaladEnumSchema

save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

Dict[str, Any]

class `schema_salad.metaschema.Documentation`(*name, type, inVocab=None, doc=None, docParent=None, docChild=None, docAfter=None, extension_fields=None, loadingOptions=None*)

Bases: *NamedType, DocType*

A documentation section. This type exists to facilitate self-documenting schemas but has no role in formal validation.

Parameters

- **name** (*Any*) –
- **type** (*Any*) –
- **inVocab** (*Optional[Any]*) –
- **doc** (*Optional[Any]*) –
- **docParent** (*Optional[Any]*) –
- **docChild** (*Optional[Any]*) –
- **docAfter** (*Optional[Any]*) –
- **extension_fields** (*Optional[Dict[str, Any]]*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –

attrs

__eq__(*other*)

Return self==value.

Parameters

- **other** (*Any*) –

Return type

bool

__hash__()

Return hash(self).

Return type

int

classmethod fromDoc(*doc, baseuri, loadingOptions, docRoot=None*)

Construct this object from the result of `yaml.load()`.

Parameters

- **doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **docRoot** (*Optional[str]*) –

Return type

Documentation

save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

Dict[str, Any]

`schema_salad.metaschema.strptime`

`schema_salad.metaschema.inttype`

`schema_salad.metaschema.floattype`

`schema_salad.metaschema.booltype`

`schema_salad.metaschema.None_type`

`schema_salad.metaschema.Any_type`

`schema_salad.metaschema.PrimitiveTypeLoader`

Names of salad data types (based on Avro schema declarations).

Refer to the [Avro schema declaration documentation](<https://avro.apache.org/docs/current/spec.html#schemas>) for detailed information.

null: no value boolean: a binary value int: 32-bit signed integer long: 64-bit signed integer float: single precision (32-bit) IEEE 754 floating-point number double: double precision (64-bit) IEEE 754 floating-point number string: Unicode character sequence

`schema_salad.metaschema.AnyLoader`

The **Any** type validates for any non-null value.

`schema_salad.metaschema.RecordFieldLoader`

`schema_salad.metaschema.RecordSchemaLoader`

`schema_salad.metaschema.EnumSchemaLoader`

`schema_salad.metaschema.ArraySchemaLoader`

`schema_salad.metaschema.JsonldPredicateLoader`

`schema_salad.metaschema.SpecializeDefLoader`

`schema_salad.metaschema.SaladRecordFieldLoader`

`schema_salad.metaschema.SaladRecordSchemaLoader`

`schema_salad.metaschema.SaladEnumSchemaLoader`

`schema_salad.metaschema.DocumentationLoader`

`schema_salad.metaschema.array_of_strtype`

`schema_salad.metaschema.union_of_None_type_or_strtype_or_array_of_strtype`

`schema_salad.metaschema.uri_strtype_True_False_None`

`schema_salad.metaschema.`

`union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader_or_ArraySchemaLoader_or_strtype`

`schema_salad.metaschema.`

`array_of_union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader_or_ArraySchemaLoader_or_strtype`

`schema_salad.metaschema.`

`union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader_or_ArraySchemaLoader_or_strtype`

schema_salad.metaschema.
typedsl_union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader_or_ArraySchemaLoader_or_

schema_salad.metaschema.**array_of_RecordFieldLoader**

schema_salad.metaschema.**union_of_None_type_or_array_of_RecordFieldLoader**

schema_salad.metaschema.**idmap_fields_union_of_None_type_or_array_of_RecordFieldLoader**

schema_salad.metaschema.**Record_nameLoader**

schema_salad.metaschema.**typedsl_Record_nameLoader_2**

schema_salad.metaschema.**union_of_None_type_or_strtype**

schema_salad.metaschema.**uri_union_of_None_type_or_strtype_True_False_None**

schema_salad.metaschema.**uri_array_of_strtype_True_False_None**

schema_salad.metaschema.**Enum_nameLoader**

schema_salad.metaschema.**typedsl_Enum_nameLoader_2**

schema_salad.metaschema.
uri_union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader_or_ArraySchemaLoader_or_strtype

schema_salad.metaschema.**Array_nameLoader**

schema_salad.metaschema.**typedsl_Array_nameLoader_2**

schema_salad.metaschema.**union_of_None_type_or_booltype**

schema_salad.metaschema.**union_of_None_type_or_inttype**

schema_salad.metaschema.**uri_strtype_False_False_1**

schema_salad.metaschema.**uri_union_of_None_type_or_strtype_False_False_None**

schema_salad.metaschema.
uri_union_of_None_type_or_strtype_or_array_of_strtype_False_False_None

schema_salad.metaschema.**union_of_None_type_or_strtype_or_JsonldPredicateLoader**

schema_salad.metaschema.**union_of_None_type_or_Any_type**

schema_salad.metaschema.**array_of_SaladRecordFieldLoader**

schema_salad.metaschema.**union_of_None_type_or_array_of_SaladRecordFieldLoader**

schema_salad.metaschema.
idmap_fields_union_of_None_type_or_array_of_SaladRecordFieldLoader

schema_salad.metaschema.
uri_union_of_None_type_or_strtype_or_array_of_strtype_False_False_1

schema_salad.metaschema.**array_of_SpecializeDefLoader**

schema_salad.metaschema.**union_of_None_type_or_array_of_SpecializeDefLoader**

schema_salad.metaschema.

idmap_specialize_union_of_None_type_or_array_of_SpecializeDefLoader

schema_salad.metaschema.**Documentation_nameLoader**

schema_salad.metaschema.**typedsl_Documentation_nameLoader_2**

schema_salad.metaschema.

union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader_or_DocumentationLoader

schema_salad.metaschema.

array_of_union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader_or_DocumentationLoader

schema_salad.metaschema.

union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader_or_DocumentationLoader_or_array_of_union_of_S

schema_salad.metaschema.**load_document**(*doc*, *baseuri=None*, *loadingOptions=None*)

Parameters

- **doc** (*Any*) –
- **baseuri** (*Optional[str]*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –

Return type

Any

schema_salad.metaschema.**load_document_with_metadata**(*doc*, *baseuri=None*, *loadingOptions=None*, *addl_metadata_fields=None*)

Parameters

- **doc** (*Any*) –
- **baseuri** (*Optional[str]*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –
- **addl_metadata_fields** (*Optional[MutableSequence[str]]*) –

Return type

Any

schema_salad.metaschema.**load_document_by_string**(*string*, *uri*, *loadingOptions=None*)

Parameters

- **string** (*Any*) –
- **uri** (*str*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –

Return type

Any

schema_salad.metaschema.**load_document_by_yaml**(*yaml*, *uri*, *loadingOptions=None*)

Shortcut to load via a YAML object. `yaml`: must be from `ruamel.yaml.main.YAML.load` with `preserve_quotes=True`

Parameters

- **yaml** (*Any*) –

- **uri** (*str*) –
- **loadingOptions** (*Optional[LoadingOptions]*) –

Return type

Any

schema_salad.python_codegen

Python code generator for a given schema salad definition.

Module Contents

Classes

PythonCodeGen

Generation of Python code for a given Schema Salad definition.

Functions

fmt(text, indent)

Use black to format this snippet.

Attributes

black

prims

schema_salad.python_codegen.**black**

schema_salad.python_codegen.**prims**

schema_salad.python_codegen.**fmt**(text, indent)

Use black to format this snippet.

:param indent the indent level for the current context

Parameters

- **text** (*str*) –
- **indent** (*int*) –

Return type

str

class `schema_salad.python_codegen.PythonCodeGen`(*out, copyright, parser_info, salad_version*)

Bases: `schema_salad.codegen_base.CodeGenBase`

Generation of Python code for a given Schema Salad definition.

Parameters

- **out** (*IO[str]*) –
- **copyright** (*Optional[str]*) –
- **parser_info** (*str*) –
- **salad_version** (*str*) –

static safe_name(*name*)

Generate a safe version of the given name.

Parameters

- name** (*str*) –

Return type

str

prologue()

Trigger to generate the prologue code.

Return type

None

begin_class(*classname, extends, doc, abstract, field_names, idfield, optional_fields*)

Produce the header for the given class.

Parameters

- **classname** (*str*) –
- **extends** (*MutableSequence[str]*) –
- **doc** (*str*) –
- **abstract** (*bool*) –
- **field_names** (*MutableSequence[str]*) –
- **idfield** (*str*) –
- **optional_fields** (*Set[str]*) –

Return type

None

end_class(*classname, field_names*)

Signal that we are done with this class.

Parameters

- **classname** (*str*) –
- **field_names** (*List[str]*) –

Return type

None

type_loader(*type_declaration*)

Parse the given type declaration and declare its components.

Parameters

type_declaration (*Union[List[Any], Dict[str, Any], str]*) –

Return type

schema_salad.codegen_base.TypeDef

declare_id_field(*name, fieldtype, doc, optional*)

Output the code to handle the given ID field.

Parameters

- **name** (*str*) –
- **fieldtype** (*schema_salad.codegen_base.TypeDef*) –
- **doc** (*str*) –
- **optional** (*bool*) –

Return type

None

declare_field(*name, fieldtype, doc, optional, subscope*)

Output the code to load the given field.

Parameters

- **name** (*str*) –
- **fieldtype** (*schema_salad.codegen_base.TypeDef*) –
- **doc** (*Optional[str]*) –
- **optional** (*bool*) –
- **subscope** (*str*) –

Return type

None

uri_loader(*inner, scoped_id, vocab_term, ref_scope*)

Construct the TypeDef for the given URI loader.

Parameters

- **inner** (*schema_salad.codegen_base.TypeDef*) –
- **scoped_id** (*bool*) –
- **vocab_term** (*bool*) –
- **ref_scope** (*Optional[int]*) –

Return type

schema_salad.codegen_base.TypeDef

idmap_loader(*field, inner, map_subject, map_predicate*)

Construct the TypeDef for the given mapped ID loader.

Parameters

- **field** (*str*) –
- **inner** (*schema_salad.codegen_base.TypeDef*) –

- **map_subject** (*str*) –
- **map_predicate** (*Optional[str]*) –

Return type*schema_salad.codegen_base.TypeDef***typedsl_loader**(*inner, ref_scope*)

Construct the TypeDef for the given DSL loader.

Parameters

- **inner** (*schema_salad.codegen_base.TypeDef*) –
- **ref_scope** (*Optional[int]*) –

Return type*schema_salad.codegen_base.TypeDef***secondaryfilesdsl_loader**(*inner*)

Construct the TypeDef for secondary files.

Parameters**inner** (*schema_salad.codegen_base.TypeDef*) –**Return type***schema_salad.codegen_base.TypeDef***epilogue**(*root_loader*)

Trigger to generate the epilogue code.

Parameters**root_loader** (*schema_salad.codegen_base.TypeDef*) –**Return type**

None

schema_salad.python_codegen_support

Template code used by python_codegen.py.

Module Contents**Classes***LoadingOptions**Saveable*

Mark classes than have a save() and fromDoc() function.

Functions

<code>load_field(val, fieldtype, baseuri, loadingOptions)</code>	
<code>save(val[, top, base_url, relative_uris])</code>	
<code>save_with_metadata(val, valLoadingOpts[, top, ...])</code>	Save and set \$namespaces, \$schemas, \$base and any other metadata fields at the top level.
<code>expand_url(url, base_url, loadingOptions[, scoped_id, ...])</code>	
<code>file_uri(path[, split_frag])</code>	
<code>prefix_url(url, namespaces)</code>	Expand short forms into full URLs using the given namespace dictionary.
<code>save_relative_uri(uri, base_url, scoped_id, ref_scope, ...)</code>	Convert any URI to a relative one, obeying the scoping rules.
<code>shortname(inputid)</code>	Compute the shortname of a fully qualified identifier.

Attributes

<code>IdxType</code>
<code>save_type</code>

`schema_salad.python_codegen_support.IdxType`

```
class schema_salad.python_codegen_support.LoadingOptions(fetcher=None, namespaces=None,
                                                         schemas=None, fileuri=None,
                                                         copyfrom=None, original_doc=None,
                                                         addl_metadata=None, baseuri=None,
                                                         idx=None, imports=None,
                                                         includes=None)
```

Parameters

- **fetcher** (*Optional*[`schema_salad.fetcher.Fetcher`]) –
- **namespaces** (*Optional*[`Dict[str, str]`]) –
- **schemas** (*Optional*[`List[str]`]) –
- **fileuri** (*Optional*[`str`]) –
- **copyfrom** (*Optional*[`LoadingOptions`]) –
- **original_doc** (*Optional*[`Any`]) –
- **addl_metadata** (*Optional*[`Dict[str, str]`]) –
- **baseuri** (*Optional*[`str`]) –
- **idx** (*Optional*[`IdxType`]) –
- **imports** (*Optional*[`List[str]`]) –

- **includes** (*Optional[List[str]*) –

property graph: `rdflib.Graph`

Generate a merged `rdflib.Graph` from all entries in `self.schemas`.

Return type

`rdflib.Graph`

idx: `IdxType`

fileuri: `str | None`

baseuri: `str`

namespaces: `MutableMapping[str, str]`

schemas: `MutableSequence[str]`

original_doc: `Any | None`

addl_metadata: `MutableMapping[str, Any]`

fetcher: `schema_salad.fetcher.Fetcher`

vocab: `Dict[str, str]`

rvocab: `Dict[str, str]`

cache: `schema_salad.utils.CacheType`

imports: `List[str]`

includes: `List[str]`

class `schema_salad.python_codegen_support.Saveable`

Bases: `abc.ABC`

Mark classes than have a `save()` and `fromDoc()` function.

abstract classmethod `fromDoc(_doc, baseuri, loadingOptions, docRoot=None)`

Construct this object from the result of `yaml.load()`.

Parameters

- **_doc** (*Any*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **docRoot** (*Optional[str]*) –

Return type

Saveable

abstract save(*top=False, base_url="", relative_uris=True*)

Convert this object to a JSON/YAML friendly dictionary.

Parameters

- **top** (*bool*) –

- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

Dict[*str*, Any]

schema_salad.python_codegen_support.**load_field**(*val*, *fieldtype*, *baseuri*, *loadingOptions*)

Parameters

- **val** (*Union[[str](#), Dict[[str](#), [str](#)]]*) –
- **fieldtype** (*_Loader*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –

Return type

Any

schema_salad.python_codegen_support.**save_type**

schema_salad.python_codegen_support.**save**(*val*, *top=True*, *base_url=""*, *relative_uris=True*)

Parameters

- **val** (*Any*) –
- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

save_type

schema_salad.python_codegen_support.**save_with_metadata**(*val*, *valLoadingOpts*, *top=True*, *base_url=""*, *relative_uris=True*)

Save and set \$namespaces, \$schemas, \$base and any other metadata fields at the top level.

Parameters

- **val** (*Any*) –
- **valLoadingOpts** (*LoadingOptions*) –
- **top** (*bool*) –
- **base_url** (*str*) –
- **relative_uris** (*bool*) –

Return type

save_type

schema_salad.python_codegen_support.**expand_url**(*url*, *base_url*, *loadingOptions*, *scoped_id=False*, *vocab_term=False*, *scoped_ref=None*)

Parameters

- **url** (*str*) –
- **base_url** (*str*) –
- **loadingOptions** (*LoadingOptions*) –

- **scoped_id** (*bool*) –
- **vocab_term** (*bool*) –
- **scoped_ref** (*Optional[int]*) –

Return type

str

schema_salad.python_codegen_support.**file_uri**(*path, split_frag=False*)

Parameters

- **path** (*str*) –
- **split_frag** (*bool*) –

Return type

str

schema_salad.python_codegen_support.**prefix_url**(*url, namespaces*)

Expand short forms into full URLs using the given namespace dictionary.

Parameters

- **url** (*str*) –
- **namespaces** (*Dict[str, str]*) –

Return type

str

schema_salad.python_codegen_support.**save_relative_uri**(*uri, base_url, scoped_id, ref_scope, relative_uris*)

Convert any URI to a relative one, obeying the scoping rules.

Parameters

- **uri** (*Any*) –
- **base_url** (*str*) –
- **scoped_id** (*bool*) –
- **ref_scope** (*Optional[int]*) –
- **relative_uris** (*bool*) –

Return type

Any

schema_salad.python_codegen_support.**shortname**(*inputid*)

Compute the shortname of a fully qualified identifier.

See https://w3id.org/cwl/v1.2/SchemaSalad.html#Short_names.

Parameters

inputid (*str*) –

Return type

str

`schema_salad.ref_resolver`

Module Contents

Classes

<code>NormDict</code>	A Dict where all keys are normalized using the provided function.
<code>Loader</code>	

Functions

<code>file_uri(path[, split_frag])</code>	
<code>uri_file_path(url)</code>	
<code>to_validation_exception(e)</code>	Convert ruamel.yaml exception to our type.
<code>SubLoader(loader)</code>	

`schema_salad.ref_resolver.file_uri(path, split_frag=False)`

Parameters

- `path (str)` –
- `split_frag (bool)` –

Return type

`str`

`schema_salad.ref_resolver.uri_file_path(url)`

Parameters

`url (str)` –

Return type

`str`

`schema_salad.ref_resolver.to_validation_exception(e)`

Convert ruamel.yaml exception to our type.

Parameters

`e (ruamel.yaml.error.MarkedYAMLError)` –

Return type

`schema_salad.exceptions.ValidationException`

`class schema_salad.ref_resolver.NormDict(normalize=str)`

Bases: `Dict[str, Union[ruamel.yaml.comments.CommentMap, ruamel.yaml.comments.CommentSeq, str, None]]`

A Dict where all keys are normalized using the provided function.

Parameters

normalize (*Callable*[[*str*], *str*]) –

__eq__ (*other*)

Return self==value.

Parameters

other (*Any*) –

Return type

bool

__getitem__ (*key*)

x.**__getitem__**(y) <==> x[y]

Parameters

key (*Any*) –

Return type

Any

__setitem__ (*key*, *value*)

Set self[key] to value.

Parameters

- **key** (*Any*) –
- **value** (*Any*) –

Return type

Any

__delitem__ (*key*)

Delete self[key].

Parameters

key (*Any*) –

Return type

Any

__contains__ (*key*)

True if the dictionary has the specified key, else False.

Parameters

key (*Any*) –

Return type

bool

__del__ ()

Return type

None

schema_salad.ref_resolver.**SubLoader** (*loader*)

Parameters

loader (*Loader*) –

Return type

Loader

```
class schema_salad.ref_resolver.Loader(ctx, schemagraph=None, foreign_properties=None, idx=None,
cache=None, session=None, fetcher_constructor=None,
skip_schemas=None, url_fields=None, allow_attachments=None,
doc_cache=True, salad_version=None)
```

Parameters

- **ctx** (*schema_salad.utils.ContextType*) –
- **schemagraph** (*Optional[rdflib.graph.Graph]*) –
- **foreign_properties** (*Optional[Set[str]]*) –
- **idx** (*Optional[schema_salad.utils.IdxType]*) –
- **cache** (*Optional[schema_salad.utils.CacheType]*) –
- **session** (*Optional[requests.sessions.Session]*) –
- **fetcher_constructor** (*Optional[schema_salad.utils.FetcherCallableType]*) –
- **skip_schemas** (*Optional[bool]*) –
- **url_fields** (*Optional[Set[str]]*) –
- **allow_attachments** (*Optional[schema_salad.utils.AttachmentsType]*) –
- **doc_cache** (*Union[str, bool]*) –
- **salad_version** (*Optional[str]*) –

```
expand_url(url, base_url, scoped_id=False, vocab_term=False, scoped_ref=None)
```

Parameters

- **url** (*str*) –
- **base_url** (*str*) –
- **scoped_id** (*bool*) –
- **vocab_term** (*bool*) –
- **scoped_ref** (*Optional[int]*) –

Return type

str

```
add_namespaces(ns)
```

Parameters

- **ns** (*Dict[str, str]*) –

Return type

None

```
add_schemas(ns, base_url)
```

Parameters

- **ns** (*Union[List[str], str]*) –
- **base_url** (*str*) –

Return type

None

add_context(*newcontext*)**Parameters****newcontext** (*schema_salad.utils.ContextType*) –**Return type**

None

resolve_ref(*ref, base_url=None, checklinks=True, strict_foreign_properties=False, content_types=None*)**Parameters**

- **ref** (*schema_salad.utils.ResolveType*) –
- **base_url** (*Optional[str]*) –
- **checklinks** (*bool*) –
- **strict_foreign_properties** (*bool*) –
- **content_types** (*Optional[List[str]]*) –

Return type*schema_salad.utils.ResolvedRefType***resolve_all**(*document, base_url, file_base=None, checklinks=True, strict_foreign_properties=False*)**Parameters**

- **document** (*schema_salad.utils.ResolveType*) –
- **base_url** (*str*) –
- **file_base** (*Optional[str]*) –
- **checklinks** (*bool*) –
- **strict_foreign_properties** (*bool*) –

Return type*schema_salad.utils.ResolvedRefType***fetch**(*url, inject_ids=True, content_types=None*)**Parameters**

- **url** (*str*) –
- **inject_ids** (*bool*) –
- **content_types** (*Optional[List[str]]*) –

Return type*schema_salad.utils.IdxResultType***validate_scoped**(*field, link, docid*)**Parameters**

- **field** (*str*) –
- **link** (*str*) –
- **docid** (*str*) –

Return type

`str`

`validate_link`(*field*, *link*, *docid*, *all_doc_ids*)

Parameters

- **field** (`str`) –
- **link** (`Union[str, ruamel.yaml.comments.CommentedList, ruamel.yaml.comments.CommentedMap]`) –
- **docid** (`str`) –
- **all_doc_ids** (`Dict[str, str]`) –

Return type

`Union[str, ruamel.yaml.comments.CommentedList, ruamel.yaml.comments.CommentedMap]`

`getid`(*d*)

Parameters

d (`Any`) –

Return type

`Optional[str]`

`validate_links`(*document*, *base_url*, *all_doc_ids*, *strict_foreign_properties=False*)

Parameters

- **document** (`schema_salad.utils.ResolveType`) –
- **base_url** (`str`) –
- **all_doc_ids** (`Dict[str, str]`) –
- **strict_foreign_properties** (`bool`) –

Return type

`None`

`schema_salad.schema`

Functions to process Schema Salad schemas.

Module Contents

Functions

<code>get_metaschema()</code>	Instantiate the metaschema.
<code>add_namespaces(metadata, namespaces)</code>	Collect the provided namespaces, checking for conflicts.
<code>collect_namespaces(metadata)</code>	Walk through the metadata object, collecting namespace declarations.
<code>load_schema(schema_ref[, cache])</code>	Load a schema that can be used to validate documents using <code>load_and_validate</code> .
<code>load_and_validate(document_loader, avsc_names, ...[, ...])</code>	Load a document and validate it with the provided schema.
<code>validate_doc(schema_names, doc, loader, strict[, ...])</code>	Validate a document using the provided schema.
<code>get_anon_name(rec)</code>	Calculate a reproducible name for anonymous types.
<code>replace_type(items, spec, loader, found[, ...])</code>	Go through and replace types in the 'spec' mapping.
<code>avro_field_name(url)</code>	Turn a URL into an Avro-safe name.
<code>make_valid_avro(items, alltypes, found[, union, ...])</code>	Convert our schema to be more avro like.
<code>deepcopy_strip(item)</code>	Make a deep copy of list and dict objects.
<code>extend_and_specialize(items, loader)</code>	Apply 'extend' and 'specialize' to fully materialize derived record types.
<code>make_avro(i, loader[, metaschema_vocab])</code>	
<code>make_avro_schema(i, loader[, metaschema_vocab])</code>	All in one convenience function.
<code>make_avro_schema_from_avro(avro)</code>	
<code>shortname(inputid)</code>	Return the last segment of the provided fragment or path.
<code>print_inheritance(doc, stream)</code>	Write a Grapviz inheritance graph for the supplied document.
<code>print_fieldrefs(doc, loader, stream)</code>	Write a GraphViz graph of the relationships between the fields.

Attributes

<code>SALAD_FILES</code>
<code>saladp</code>
<code>cached_metaschema</code>
<code>schema_type</code>
<code>Avro</code>

```
schema_salad.schema.SALAD_FILES = ('metaschema.yml', 'metaschema_base.yml', 'salad.md',
'field_name.yml', 'import_include.md',...
```

```
schema_salad.schema.saladp = 'https://w3id.org/cwl/salad#'
```

```
schema_salad.schema.cached_metaschema: Tuple[schema_salad.avro.schema.Names,
List[Dict[str, str]], schema_salad.ref_resolver.Loader] | None
```

`schema_salad.schema.get_metaschema()`

Instantiate the metaschema.

Return type

`Tuple[schema_salad.avro.schema.Names, List[Dict[str, str]], schema_salad.ref_resolver.Loader]`

`schema_salad.schema.add_namespaces(metadata, namespaces)`

Collect the provided namespaces, checking for conflicts.

Parameters

- **metadata** (*Mapping[str, Any]*) –
- **namespaces** (*MutableMapping[str, str]*) –

Return type

None

`schema_salad.schema.collect_namespaces(metadata)`

Walk through the metadata object, collecting namespace declarations.

Parameters

metadata (*Mapping[str, Any]*) –

Return type

`Dict[str, str]`

`schema_salad.schema.schema_type`

`schema_salad.schema.load_schema(schema_ref, cache=None)`

Load a schema that can be used to validate documents using `load_and_validate`.

Returns

`document_loader, avsc_names, schema_metadata, metaschema_loader`

Parameters

- **schema_ref** (*schema_salad.utils.ResolveType*) –
- **cache** (*Optional[schema_salad.utils.CacheType]*) –

Return type

`schema_type`

`schema_salad.schema.load_and_validate(document_loader, avsc_names, document, strict, strict_foreign_properties=False)`

Load a document and validate it with the provided schema.

return data, metadata

Parameters

- **document_loader** (*schema_salad.ref_resolver.Loader*) –
- **avsc_names** (*schema_salad.avro.schema.Names*) –
- **document** (*Union[rueaml.yaml.comments.CommentMap, str]*) –
- **strict** (*bool*) –
- **strict_foreign_properties** (*bool*) –

Return type

`Tuple[Any, Dict[str, Any]]`

`schema_salad.schema.validate_doc(schema_names, doc, loader, strict, strict_foreign_properties=False)`

Validate a document using the provided schema.

Parameters

- **schema_names** (`schema_salad.avro.schema.Names`) –
- **doc** (`schema_salad.utils.ResolveType`) –
- **loader** (`schema_salad.ref_resolver.Loader`) –
- **strict** (`bool`) –
- **strict_foreign_properties** (`bool`) –

Return type

None

`schema_salad.schema.get_anon_name(rec)`

Calculate a reproducible name for anonymous types.

Parameters

rec (`MutableMapping[str, Union[str, Dict[str, str], List[str]]]`) –

Return type

`str`

`schema_salad.schema.replace_type(items, spec, loader, found, find_embeds=True, deepen=True)`

Go through and replace types in the ‘spec’ mapping.

Parameters

- **items** (`Any`) –
- **spec** (`Dict[str, Any]`) –
- **loader** (`schema_salad.ref_resolver.Loader`) –
- **found** (`Set[str]`) –
- **find_embeds** (`bool`) –
- **deepen** (`bool`) –

Return type

`Any`

`schema_salad.schema.avro_field_name(url)`

Turn a URL into an Avro-safe name.

If the URL has no fragment, return this plain URL.

Extract either the last part of the URL fragment past the slash, otherwise the whole fragment.

Parameters

url (`str`) –

Return type

`str`

`schema_salad.schema.Avro`

`schema_salad.schema.make_valid_avro(items, alltypes, found, union=False, fielddef=False, vocab=None)`

Convert our schema to be more avro like.

Parameters

- **items** (*Avro*) –
- **alltypes** (*Dict[str, Dict[str, Any]]*) –
- **found** (*Set[str]*) –
- **union** (*bool*) –
- **fielddef** (*bool*) –
- **vocab** (*Optional[Dict[str, str]]*) –

Return type

Union[Avro, MutableMapping[str, str], str, List[Union[Any, MutableMapping[str, str], str]]]

`schema_salad.schema.deepcopy_strip(item)`

Make a deep copy of list and dict objects.

Intentionally do not copy attributes. This is to discard CommentedMap and CommentedSeq metadata which is very expensive with regular copy.deepcopy.

Parameters

item (*Any*) –

Return type

Any

`schema_salad.schema.extend_and_specialize(items, loader)`

Apply ‘extend’ and ‘specialize’ to fully materialize derived record types.

Parameters

- **items** (*List[Dict[str, Any]]*) –
- **loader** (`schema_salad.ref_resolver.Loader`) –

Return type

List[Dict[str, Any]]

`schema_salad.schema.make_avro(i, loader, metaschema_vocab=None)`

Parameters

- **i** (*List[Dict[str, Any]]*) –
- **loader** (`schema_salad.ref_resolver.Loader`) –
- **metaschema_vocab** (*Optional[Dict[str, str]]*) –

Return type

List[Any]

`schema_salad.schema.make_avro_schema(i, loader, metaschema_vocab=None)`

All in one convenience function.

Call `make_avro()` and `make_avro_schema_from_avro()` separately if you need the intermediate result for diagnostic output.

Parameters

- **i** (*List[Any]*) –
- **loader** (`schema_salad.ref_resolver.Loader`) –
- **metaschema_vocab** (*Optional[Dict[str, str]]*) –

Return type*schema_salad.avro.schema.Names*`schema_salad.schema.make_avro_schema_from_avro(avro)`**Parameters****avro** (*List[Union[Avro, Dict[str, str], str]]*) –**Return type***schema_salad.avro.schema.Names*`schema_salad.schema.shortname(inputid)`

Return the last segment of the provided fragment or path.

Parameters**inputid** (*str*) –**Return type***str*`schema_salad.schema.print_inheritance(doc, stream)`

Write a Grapviz inheritance graph for the supplied document.

Parameters

- **doc** (*List[Dict[str, Any]]*) –
- **stream** (*IO[Any]*) –

Return type

None

`schema_salad.schema.print_fieldrefs(doc, loader, stream)`

Write a GraphViz graph of the relationships between the fields.

Parameters

- **doc** (*List[Dict[str, Any]]*) –
- **loader** (`schema_salad.ref_resolver.Loader`) –
- **stream** (*IO[Any]*) –

Return type

None

`schema_salad.sourceline`**Module Contents****Classes***SourceLine*

Functions

rename(source)

add_lc_filename(r, source)

reflow_all(text[, maxline])

reflow(text, maxline[, shift])

indent(v[, nolead, shift, bullet])

bullets(textlist, bul)

strip_duplicated_lineno(text)

Strip duplicated line numbers.

strip_dup_lineno(text[, maxline])

cmap(d[, lc, fn])

Attributes

lineno_re

schema_salad.sourceline.lineno_re

schema_salad.sourceline.rename(*source*)

Parameters

source (*str*) –

Return type

str

schema_salad.sourceline.add_lc_filename(*r*, *source*)

Parameters

- **r** (*ruamel.yaml.comments.CommentedException*) –
- **source** (*str*) –

Return type

None

schema_salad.sourceline.reflow_all(*text*, *maxline*=None)

Parameters

- **text** (*str*) –
- **maxline** (*Optional[int]*) –

Return type

str

schema_salad.sourceline.**reflow**(*text*, *maxline*, *shift=""*)

Parameters

- **text** (*str*) –
- **maxline** (*int*) –
- **shift** (*Optional[str]*) –

Return type

str

schema_salad.sourceline.**indent**(*v*, *nolead=False*, *shift=' '*, *bullet=' '*)

Parameters

- **v** (*str*) –
- **nolead** (*bool*) –
- **shift** (*str*) –
- **bullet** (*str*) –

Return type

str

schema_salad.sourceline.**bullets**(*textlist*, *bul*)

Parameters

- **textlist** (*List[str]*) –
- **bul** (*str*) –

Return type

str

schema_salad.sourceline.**strip_duplicated_lineno**(*text*)

Strip duplicated line numbers.

Same as `strip_dup_lineno()` but without reflow.

Parameters

text (*str*) –

Return type

str

schema_salad.sourceline.**strip_dup_lineno**(*text*, *maxline=None*)

Parameters

- **text** (*str*) –
- **maxline** (*Optional[int]*) –

Return type

str

`schema_salad.sourceline.cmap(d, lc=None, fn=None)`

Parameters

- **d** (`Union[int, float, str, MutableMapping[str, Any], MutableSequence[Any], None]`) –
- **lc** (`Optional[List[int]]`) –
- **fn** (`Optional[str]`) –

Return type

`Union[int, float, str, ruamel.yaml.comments.CommentMap, ruamel.yaml.comments.CommentSeq, None]`

class `schema_salad.sourceline.SourceLine`(*item*, *key=None*, *raise_type=str*, *include_traceback=False*)

Parameters

- **item** (*Any*) –
- **key** (`Optional[Any]`) –
- **raise_type** (`Callable[[str], Any]`) –
- **include_traceback** (*bool*) –

`__enter__()`

Return type

`SourceLine`

`__exit__(exc_type, exc_value, tb)`

Parameters

- **exc_type** (*Any*) –
- **exc_value** (*Any*) –
- **tb** (*Any*) –

Return type

`None`

`file()`

Return type

`Optional[str]`

`start()`

Return type

`Optional[Tuple[int, int]]`

`end()`

Return type

`Optional[Tuple[int, int]]`

`makeLead()`

Return type

`str`

makeError(*msg*)

Parameters

msg (*str*) –

Return type

Any

`schema_salad.typescript_codegen`

TypeScript code generator for a given schema salad definition.

Module Contents

Classes

<code>TypeScriptCodeGen</code>	Generation of TypeScript code for a given Schema Salad definition.
--------------------------------	--

Functions

<code>doc_to_doc_string</code> (<i>doc</i> [, <i>indent_level</i>])	Generate a documentation string from a schema salad doc field.
---	--

Attributes

<code>prims</code>

`schema_salad.typescript_codegen.doc_to_doc_string`(*doc*, *indent_level*=0)

Generate a documentation string from a schema salad doc field.

Parameters

- **doc** (*Optional*[*str*]) –
- **indent_level** (*int*) –

Return type

str

`schema_salad.typescript_codegen.prims`

class `schema_salad.typescript_codegen.TypeScriptCodeGen`(*base*, *examples*, *target*, *package*)

Bases: `schema_salad.codegen_base.CodeGenBase`

Generation of TypeScript code for a given Schema Salad definition.

Parameters

- **base** (*str*) –
- **examples** (*Optional[str]*) –
- **target** (*Optional[str]*) –
- **package** (*str*) –

prologue()

Trigger to generate the prologue code.

Return type

None

static safe_name(name)

Generate a safe version of the given name.

Parameters

name (*str*) –

Return type

str

begin_class(classname, extends, doc, abstract, field_names, idfield, optional_fields)

Produce the header for the given class.

Parameters

- **classname** (*str*) –
- **extends** (*MutableSequence[str]*) –
- **doc** (*str*) –
- **abstract** (*bool*) –
- **field_names** (*MutableSequence[str]*) –
- **idfield** (*str*) –
- **optional_fields** (*Set[str]*) –

Return type

None

end_class(classname, field_names)

Signal that we are done with this class.

Parameters

- **classname** (*str*) –
- **field_names** (*List[str]*) –

Return type

None

type_loader(type_declaration)

Parse the given type declaration and declare its components.

Parameters

type_declaration (*Union[List[Any], Dict[str, Any], str]*) –

Return type*schema_salad.codegen_base.TypeDef***type_loader_enum**(*type_declaration*)**Parameters****type_declaration** (*Dict[str, Any]*) –**Return type***schema_salad.codegen_base.TypeDef***declare_field**(*name, fieldtype, doc, optional, subscope*)

Output the code to load the given field.

Parameters

- **name** (*str*) –
- **fieldtype** (*schema_salad.codegen_base.TypeDef*) –
- **doc** (*Optional[str]*) –
- **optional** (*bool*) –
- **subscope** (*str*) –

Return type

None

declare_id_field(*name, fieldtype, doc, optional*)

Output the code to handle the given ID field.

Parameters

- **name** (*str*) –
- **fieldtype** (*schema_salad.codegen_base.TypeDef*) –
- **doc** (*str*) –
- **optional** (*bool*) –

Return type

None

to_typescript(*val*)

Convert a Python keyword to a TypeScript keyword.

Parameters**val** (*Any*) –**Return type**

Any

uri_loader(*inner, scoped_id, vocab_term, ref_scope*)

Construct the TypeDef for the given URI loader.

Parameters

- **inner** (*schema_salad.codegen_base.TypeDef*) –
- **scoped_id** (*bool*) –
- **vocab_term** (*bool*) –
- **ref_scope** (*Optional[int]*) –

Return type

schema_salad.codegen_base.TypeDef

idmap_loader(*field*, *inner*, *map_subject*, *map_predicate*)

Construct the TypeDef for the given mapped ID loader.

Parameters

- **field** (*str*) –
- **inner** (*schema_salad.codegen_base.TypeDef*) –
- **map_subject** (*str*) –
- **map_predicate** (*Optional[str]*) –

Return type

schema_salad.codegen_base.TypeDef

typedsl_loader(*inner*, *ref_scope*)

Construct the TypeDef for the given DSL loader.

Parameters

- **inner** (*schema_salad.codegen_base.TypeDef*) –
- **ref_scope** (*Optional[int]*) –

Return type

schema_salad.codegen_base.TypeDef

epilogue(*root_loader*)

Trigger to generate the epilogue code.

Parameters

root_loader (*schema_salad.codegen_base.TypeDef*) –

Return type

None

secondaryfilesdsl_loader(*inner*)

Construct the TypeDef for secondary files.

Parameters

inner (*schema_salad.codegen_base.TypeDef*) –

Return type

schema_salad.codegen_base.TypeDef

`schema_salad.utils`

Module Contents

Functions

<code>add_dictlist(di, key, val)</code>	
<code>aslist(thing)</code>	Wrap single items and lists.
<code>flatten(thing[, ltypes])</code>	
<code>onWindows()</code>	
<code>convert_to_dict(j4)</code>	
<code>json_dump(obj, fp, **kwargs)</code>	Force use of unicode.
<code>json_dumps(obj, **kwargs)</code>	Force use of unicode.
<code>stdout()</code>	Build a replacement for <code>sys.stdout</code> that allow for writing binary data.
<code>yaml_no_ts()</code>	Get a YAML loader that won't parse timestamps into datetime objects.

Attributes

<code>ContextType</code>
<code>DocumentType</code>
<code>DocumentOrStrType</code>
<code>FieldType</code>
<code>MandatoryResolveType</code>
<code>ResolveType</code>
<code>ResolvedRefType</code>
<code>IdxResultType</code>
<code>IdxType</code>
<code>CacheType</code>
<code>FetcherCallableType</code>
<code>AttachmentsType</code>

`schema_salad.utils.ContextType`

`schema_salad.utils.DocumentType`

`schema_salad.utils.DocumentOrStrType`

`schema_salad.utils.FieldType`

`schema_salad.utils.MandatoryResolveType`

`schema_salad.utils.ResolveType`

`schema_salad.utils.ResolvedRefType`

`schema_salad.utils.IdxResultType`

`schema_salad.utils.IdxType`

`schema_salad.utils.CacheType`

`schema_salad.utils.FetcherCallableType`

`schema_salad.utils.AttachmentsType`

`schema_salad.utils.add_dictlist`(*di*, *key*, *val*)

Parameters

- **di** (*Dict*[*Any*, *Any*]) –
- **key** (*Any*) –
- **val** (*Any*) –

Return type

None

`schema_salad.utils.asList`(*thing*)

Wrap single items and lists.

Return lists unchanged.

Parameters

thing (*Any*) –

Return type

MutableSequence[*Any*]

`schema_salad.utils.flatten`(*thing*, *ltypes*=(*list*, *tuple*))

Parameters

- **thing** (*Any*) –
- **ltypes** (*Any*) –

Return type

Any

`schema_salad.utils.onWindows`()

Return type

`bool`

`schema_salad.utils.convert_to_dict(j4)`

Parameters

`j4` (*Any*) –

Return type

Any

`schema_salad.utils.json_dump(obj, fp, **kwargs)`

Force use of unicode.

Parameters

- `obj` (*Any*) –
- `fp` (*IO[str]*) –
- `kwargs` (*Any*) –

Return type

None

`schema_salad.utils.json_dumps(obj, **kwargs)`

Force use of unicode.

Parameters

`obj` (*Any*) –

Return type

str

`schema_salad.utils.stdout()`

Build a replacement for `sys.stdout` that allow for writing binary data.

Return type

io.BufferedWriter

`schema_salad.utils.yaml_no_ts()`

Get a YAML loader that won't parse timestamps into datetime objects.

Such datetime objects can't be easily dumped into JSON.

Return type

ruamel.yaml.main.YAML

`schema_salad.validate`

Module Contents

Functions

`validate`(expected_schema, datum[, identifiers, ...])

`avro_shortcode`(name)

Produce an avro friendly short name.

`avro_type_name`(url)

Turn a URL into an Avro-safe name.

`friendly`(v)

`vpformat`(datum)

`validate_ex`(expected_schema, datum[, identifiers, ...]) Determine if a python datum is an instance of a schema.

Attributes

`INT_MIN_VALUE`

`INT_MAX_VALUE`

`LONG_MIN_VALUE`

`LONG_MAX_VALUE`

`saladp`

`primitives`

`schema_salad.validate.validate`(expected_schema, datum, identifiers=None, strict=False, foreign_properties=None, vocab=None)

Parameters

- **expected_schema** (`schema_salad.avro.schema.Schema`) –
- **datum** (*Any*) –
- **identifiers** (*Optional*[`List[str]`]) –
- **strict** (*bool*) –
- **foreign_properties** (*Optional*[`Set[str]`]) –
- **vocab** (*Optional*[`Mapping[str, str]`]) –

Return type

`bool`

`schema_salad.validate.INT_MIN_VALUE`

`schema_salad.validate.INT_MAX_VALUE`

`schema_salad.validate.LONG_MIN_VALUE`

`schema_salad.validate.LONG_MAX_VALUE`

`schema_salad.validate.avro_shortcode(name)`

Produce an avro friendly short name.

Parameters

name (*str*) –

Return type

str

`schema_salad.validate.saladp = 'https://w3id.org/cwl/salad#'`

`schema_salad.validate.primitives`

`schema_salad.validate.avro_type_name(url)`

Turn a URL into an Avro-safe name.

If the URL has no fragment, return this plain URL.

Extract either the last part of the URL fragment past the slash, otherwise the whole fragment.

Parameters

url (*str*) –

Return type

str

`schema_salad.validate.friendly(v)`

Parameters

v (*Any*) –

Return type

Any

`schema_salad.validate.vpformat(datum)`

Parameters

datum (*Any*) –

Return type

str

`schema_salad.validate.validate_ex(expected_schema, datum, identifiers=None, strict=False,
foreign_properties=None, raise_ex=True,
strict_foreign_properties=False, logger=_logger,
skip_foreign_properties=False, vocab=None)`

Determine if a python datum is an instance of a schema.

Parameters

- **expected_schema** (`schema_salad.avro.schema.Schema`) –
- **datum** (*Any*) –
- **identifiers** (*Optional[List[str]*) –
- **strict** (*bool*) –
- **foreign_properties** (*Optional[Set[str]*) –
- **raise_ex** (*bool*) –

- `strict_foreign_properties` (*bool*) –
- `logger` (*logging.Logger*) –
- `skip_foreign_properties` (*bool*) –
- `vocab` (*Optional[Mapping[str, str]]*) –

Return type
bool

Package Contents

```
schema_salad.__author__ = 'peter.amstutz@curoverse.com'
```

6.4 Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)

PYTHON MODULE INDEX

S

- schema_salad, 17
- schema_salad.__main__, 59
- schema_salad.avro, 17
- schema_salad.avro.schema, 17
- schema_salad.codegen, 59
- schema_salad.codegen_base, 60
- schema_salad.cpp_codegen, 63
- schema_salad.dlang_codegen, 68
- schema_salad.dotnet_codegen, 70
- schema_salad.exceptions, 74
- schema_salad.fetcher, 76
- schema_salad.java_codegen, 78
- schema_salad.jsonld_context, 82
- schema_salad.main, 84
- schema_salad.makedoc, 84
- schema_salad.metaschema, 90
- schema_salad.python_codegen, 112
- schema_salad.python_codegen_support, 115
- schema_salad.ref_resolver, 120
- schema_salad.schema, 124
- schema_salad.sourceline, 129
- schema_salad.tests, 24
- schema_salad.tests.confstest, 24
- schema_salad.tests.matcher, 24
- schema_salad.tests.test_avro_names, 25
- schema_salad.tests.test_cg, 26
- schema_salad.tests.test_cli_args, 28
- schema_salad.tests.test_cpp_codegen, 28
- schema_salad.tests.test_cwl11, 29
- schema_salad.tests.test_dlang_codegen, 30
- schema_salad.tests.test_dotnet_codegen, 31
- schema_salad.tests.test_errors, 33
- schema_salad.tests.test_examples, 36
- schema_salad.tests.test_fetch, 40
- schema_salad.tests.test_fp, 42
- schema_salad.tests.test_java_codegen, 42
- schema_salad.tests.test_makedoc, 43
- schema_salad.tests.test_misc, 46
- schema_salad.tests.test_pickling, 46
- schema_salad.tests.test_print_online, 47
- schema_salad.tests.test_python_codegen, 48
- schema_salad.tests.test_real_cwl, 50
- schema_salad.tests.test_ref_resolver, 51
- schema_salad.tests.test_schema, 54
- schema_salad.tests.test_schemas_directive, 54
- schema_salad.tests.test_subtypes, 55
- schema_salad.tests.test_typescript_codegen, 56
- schema_salad.tests.util, 58
- schema_salad.typescript_codegen, 133
- schema_salad.utils, 136
- schema_salad.validate, 139

Symbols

- `__author__` (in module `schema_salad`), 142
- `__contains__`() (`schema_salad.ref_resolver.NormDict` method), 121
- `__del__`() (`schema_salad.ref_resolver.NormDict` method), 121
- `__delitem__`() (`schema_salad.ref_resolver.NormDict` method), 121
- `__enter__`() (`schema_salad.sourceline.SourceLine` method), 132
- `__eq__`() (`schema_salad.metaschema.ArraySchema` method), 100
- `__eq__`() (`schema_salad.metaschema.Documentation` method), 108
- `__eq__`() (`schema_salad.metaschema.EnumSchema` method), 99
- `__eq__`() (`schema_salad.metaschema.JsonldPredicate` method), 102
- `__eq__`() (`schema_salad.metaschema.RecordField` method), 97
- `__eq__`() (`schema_salad.metaschema.RecordSchema` method), 98
- `__eq__`() (`schema_salad.metaschema.SaladEnumSchema` method), 107
- `__eq__`() (`schema_salad.metaschema.SaladRecordField` method), 104
- `__eq__`() (`schema_salad.metaschema.SaladRecordSchema` method), 105
- `__eq__`() (`schema_salad.metaschema.SpecializeDef` method), 103
- `__eq__`() (`schema_salad.ref_resolver.NormDict` method), 121
- `__eq__`() (`schema_salad.tests.matcher.JsonDiffMatcher` method), 25
- `__exit__`() (`schema_salad.sourceline.SourceLine` method), 132
- `__getitem__`() (`schema_salad.ref_resolver.NormDict` method), 121
- `__hash__`() (`schema_salad.metaschema.ArraySchema` method), 100
- `__hash__`() (`schema_salad.metaschema.Documentation` method), 108
- `__hash__`() (`schema_salad.metaschema.EnumSchema` method), 99
- `__hash__`() (`schema_salad.metaschema.JsonldPredicate` method), 102
- `__hash__`() (`schema_salad.metaschema.RecordField` method), 97
- `__hash__`() (`schema_salad.metaschema.RecordSchema` method), 98
- `__hash__`() (`schema_salad.metaschema.SaladEnumSchema` method), 107
- `__hash__`() (`schema_salad.metaschema.SaladRecordField` method), 104
- `__hash__`() (`schema_salad.metaschema.SaladRecordSchema` method), 105
- `__hash__`() (`schema_salad.metaschema.SpecializeDef` method), 103
- `__setitem__`() (`schema_salad.ref_resolver.NormDict` method), 121
- `__slots__` (`schema_salad.codegen_base.TypeDef` attribute), 60
- `__str__`() (`schema_salad.exceptions.SchemaSaladException` method), 75
- `--brand`
 - `schema-salad-doc` command line option, 16
 - `schema-salad-tool` command line option, 15
- `--brandinverse`
 - `schema-salad-doc` command line option, 16
 - `schema-salad-tool` command line option, 15
- `--brandlink`
 - `schema-salad-doc` command line option, 16
 - `schema-salad-tool` command line option, 15
- `--brandstyle`
 - `schema-salad-doc` command line option, 16
 - `schema-salad-tool` command line option, 15
- `--codegen`
 - `schema-salad-tool` command line option, 14
- `--codegen-copyright`
 - `schema-salad-tool` command line option, 15
- `--codegen-examples`
 - `schema-salad-tool` command line option, 14
- `--codegen-package`
 - `schema-salad-tool` command line option, 15

--codegen-parser-info
 schema-salad-tool command line option, 15
 --codegen-target
 schema-salad-tool command line option, 14
 --debug
 schema-salad-doc command line option, 16
 schema-salad-tool command line option, 15
 --help
 schema-salad-doc command line option, 16
 schema-salad-tool command line option, 14
 --non-strict
 schema-salad-tool command line option, 15
 --only
 schema-salad-doc command line option, 16
 schema-salad-tool command line option, 15
 --primetype
 schema-salad-doc command line option, 16
 schema-salad-tool command line option, 15
 --print-avro
 schema-salad-tool command line option, 14
 --print-doc
 schema-salad-tool command line option, 15
 --print-fieldrefs-dot
 schema-salad-tool command line option, 14
 --print-index
 schema-salad-tool command line option, 14
 --print-inheritance-dot
 schema-salad-tool command line option, 14
 --print-jsonld-context
 schema-salad-tool command line option, 14
 --print-metadata
 schema-salad-tool command line option, 14
 --print-oneline
 schema-salad-tool command line option, 15
 --print-pre
 schema-salad-tool command line option, 14
 --print-rdf
 schema-salad-tool command line option, 14
 --print-rdfs
 schema-salad-tool command line option, 14
 --quiet
 schema-salad-tool command line option, 15
 --rdf-serializer
 schema-salad-tool command line option, 14
 --redirect
 schema-salad-doc command line option, 16
 schema-salad-tool command line option, 15
 --skip-schemas
 schema-salad-tool command line option, 14
 --strict
 schema-salad-tool command line option, 15
 --strict-foreign-properties
 schema-salad-tool command line option, 14
 --verbose
 schema-salad-tool command line option, 15
 --version
 schema-salad-tool command line option, 15
 -h
 schema-salad-doc command line option, 16
 schema-salad-tool command line option, 14
 -v
 schema-salad-tool command line option, 15

A

add_context() (*schema_salad.ref_resolver.Loader* method), 123
 add_dictlist() (in module *schema_salad.utils*), 138
 add_entry() (*schema_salad.makedoc.ToC* method), 87
 add_lc_filename() (in module *schema_salad.sourceline*), 130
 add_name() (*schema_salad.avro.schema.Names* method), 20
 add_namespaces() (in module *schema_salad.schema*), 126
 add_namespaces() (*schema_salad.ref_resolver.Loader* method), 122
 add_schemas() (*schema_salad.ref_resolver.Loader* method), 122
 add_vocab() (*schema_salad.codegen_base.CodeGenBase* method), 60
 addl_metadata (*schema_salad.metaschema.LoadingOptions* attribute), 94
 addl_metadata (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117
 Any_type (in module *schema_salad.metaschema*), 109
 AnyLoader (in module *schema_salad.metaschema*), 109
 arg_parser() (in module *schema_salad.main*), 84
 arg_parser() (in module *schema_salad.makedoc*), 89
 Array_nameLoader (in module *schema_salad.metaschema*), 110
 array_of_RecordFieldLoader (in module *schema_salad.metaschema*), 110
 array_of_SaladRecordFieldLoader (in module *schema_salad.metaschema*), 110
 array_of_SpecializeDefLoader (in module *schema_salad.metaschema*), 110
 array_of_strtype (in module *schema_salad.metaschema*), 109
 array_of_union_of_PrimitiveTypeLoader_or_RecordSchemaLoader (in module *schema_salad.metaschema*), 109
 array_of_union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader (in module *schema_salad.metaschema*), 111
 ArraySchema (class in *schema_salad.avro.schema*), 22
 ArraySchema (class in *schema_salad.metaschema*), 100
 ArraySchemaLoader (in module *schema_salad.metaschema*), 109
 as_warning() (*schema_salad.exceptions.SchemaSaladException* method), 74

- aslist() (in module *schema_salad.utils*), 138
- AtomicPropType (in module *schema_salad.avro.schema*), 18
- AttachmentsType (in module *schema_salad.utils*), 138
- attrs (in module *schema_salad.metaschema.ArraySchema* attribute), 100
- attrs (in module *schema_salad.metaschema.Documentation* attribute), 108
- attrs (in module *schema_salad.metaschema.EnumSchema* attribute), 99
- attrs (in module *schema_salad.metaschema.JsonldPredicate* attribute), 102
- attrs (in module *schema_salad.metaschema.RecordField* attribute), 97
- attrs (in module *schema_salad.metaschema.RecordSchema* attribute), 98
- attrs (in module *schema_salad.metaschema.SaladEnumSchema* attribute), 107
- attrs (in module *schema_salad.metaschema.SaladRecordField* attribute), 104
- attrs (in module *schema_salad.metaschema.SaladRecordSchema* attribute), 105
- attrs (in module *schema_salad.metaschema.SpecializeDef* attribute), 102
- Avro (in module *schema_salad.schema*), 127
- avro_field_name() (in module *schema_salad.schema*), 127
- avro_shortcode() (in module *schema_salad.validate*), 141
- avro_type_name() (in module *schema_salad.validate*), 141
- AvroException, 18
- avrold_doc() (in module *schema_salad.makedoc*), 89
- avsc_names (in module *schema_salad.tests.test_real_cwl.TestRealWorldCWL* attribute), 50
- avsc_names (in module *schema_salad.tests.test_schemas_directive.TestSchemasDirective* attribute), 55
- ## B
- baseuri (in module *schema_salad.metaschema.LoadingOptions* attribute), 94
- baseuri (in module *schema_salad.python_codegen_support.LoadingOptions* attribute), 117
- BASIC_JAVA_IDENTIFIER_RE (in module *schema_salad.java_codegen*), 78
- basicTypes (in module *schema_salad.makedoc*), 88
- basket_file_uri (in module *schema_salad.tests.util*), 58
- begin_class() (in module *schema_salad.codegen_base.CodeGenBase* method), 61
- begin_class() (in module *schema_salad.dotnet_codegen.DotNetCodeGen* method), 71
- begin_class() (in module *schema_salad.java_codegen.JavaCodeGen* method), 79
- begin_class() (in module *schema_salad.python_codegen.PythonCodeGen* method), 113
- begin_class() (in module *schema_salad.typescript_codegen.TypeScriptCodeGen* method), 134
- black (in module *schema_salad.python_codegen*), 112
- block_code() (in module *schema_salad.makedoc.MyRenderer* method), 86
- block_html() (in module *schema_salad.makedoc.MyRenderer* method), 86
- booltype (in module *schema_salad.metaschema*), 109
- bullets() (in module *schema_salad.sourceline*), 131
- ## C
- cache (in module *schema_salad.metaschema.LoadingOptions* attribute), 94
- cache (in module *schema_salad.python_codegen_support.LoadingOptions* attribute), 117
- cached_metaschema (in module *schema_salad.schema*), 125
- CacheType (in module *schema_salad.utils*), 138
- captured_output() (in module *schema_salad.tests.test_cli_args*), 28
- check_exists() (in module *schema_salad.fetcher.DefaultFetcher* method), 77
- check_exists() (in module *schema_salad.fetcher.Fetcher* method), 77
- check_exists() (in module *schema_salad.tests.test_fetch.CWLTestFetcher* method), 41
- check_exists() (in module *schema_salad.tests.test_fetch.testFetcher* method), 41
- ClassDefinition (class in module *schema_salad.cpp_codegen*), 65
- ClassValidationException, 76
- cmap() (in module *schema_salad.sourceline*), 131
- codegen() (in module *schema_salad.codegen*), 59
- CodeGenBase (class in module *schema_salad.codegen_base*), 60
- collect_namespaces() (in module *schema_salad.schema*), 126
- contents() (in module *schema_salad.makedoc.ToC* method), 88
- ContextType (in module *schema_salad.utils*), 137
- convert_to_dict() (in module *schema_salad.utils*), 138
- convertTypeToCpp() (in module *schema_salad.cpp_codegen.CppCodeGen* method), 67
- cpp_codegen() (in module *schema_salad.tests.test_cpp_codegen*), 29
- CppCodeGen (class in module *schema_salad.cpp_codegen*), 67
- cwl_file_uri (in module *schema_salad.tests.test_dotnet_codegen*), 32
- cwl_file_uri (in module *schema_salad.tests.test_schema*), 54
- cwl_file_uri (in module *schema_salad.tests.test_typescript_codegen*), 54

57
 cwl_file_uri (in module *schema_salad.tests.util*), 58
 cwl_v1_2_schema() (in module *schema_salad.tests.test_cwl11*), 29
 CWLTestFetcher (class in *schema_salad.tests.test_fetch*), 41

D
 declare_field() (*schema_salad.codegen_base.CodeGenBase* method), 61
 declare_field() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 72
 declare_field() (*schema_salad.java_codegen.JavaCodeGen* method), 80
 declare_field() (*schema_salad.python_codegen.PythonCodeGen* method), 114
 declare_field() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 135
 declare_id_field() (*schema_salad.codegen_base.CodeGenBase* method), 61
 declare_id_field() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 72
 declare_id_field() (*schema_salad.java_codegen.JavaCodeGen* method), 80
 declare_id_field() (*schema_salad.python_codegen.PythonCodeGen* method), 114
 declare_id_field() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 135
 declare_type() (*schema_salad.codegen_base.CodeGenBase* method), 60
 deepcopy_strip() (in module *schema_salad.schema*), 128
 default (*schema_salad.avro.schema.Field* property), 21
 DefaultFetcher (class in *schema_salad.fetcher*), 77
 dlang_codegen() (in module *schema_salad.tests.test_dlang_codegen*), 31
 DlangCodeGen (class in *schema_salad.dlang_codegen*), 68
 doc_to_doc_string() (in module *schema_salad.dotnet_codegen*), 71
 doc_to_doc_string() (in module *schema_salad.java_codegen*), 78
 doc_to_doc_string() (in module *schema_salad.typescript_codegen*), 133
 DocType (class in *schema_salad.metaschema*), 103
 document
 schema-salad-tool command line option, 14
 document_loader (*schema_salad.tests.test_real_cwl.TestRealWorldCWL* attribute), 50
 document_loader (*schema_salad.tests.test_schemas_directive.TestSchemasDirective* attribute), 55
 Documentation (class in *schema_salad.metaschema*), 107
 Documentation_nameLoader (in module *schema_salad.metaschema*), 111
 DocumentationLoader (in module *schema_salad.metaschema*), 109
 Documented (class in *schema_salad.metaschema*), 97
 DocumentOrStrType (in module *schema_salad.utils*), 137
 DocumentType (in module *schema_salad.utils*), 137
 dotnet_codegen() (in module *schema_salad.tests.test_dotnet_codegen*), 32
 DotNetCodeGen (class in *schema_salad.dotnet_codegen*), 71

E
 end() (*schema_salad.sourceline.SourceLine* method), 112
 end_class() (*schema_salad.codegen_base.CodeGenBase* method), 61
 end_class() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 72
 end_class() (*schema_salad.java_codegen.JavaCodeGen* method), 80
 end_class() (*schema_salad.python_codegen.PythonCodeGen* method), 113
 end_class() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 134
 Enum_nameLoader (in module *schema_salad.metaschema*), 110
 EnumDefinition (class in *schema_salad.cpp_codegen*), 66
 EnumSchema (class in *schema_salad.avro.schema*), 22
 EnumSchema (class in *schema_salad.metaschema*), 99
 EnumSchemaLoader (in module *schema_salad.metaschema*), 109
 epilogue() (*schema_salad.codegen_base.CodeGenBase* method), 63
 epilogue() (*schema_salad.cpp_codegen.CppCodeGen* method), 67
 epilogue() (*schema_salad.dlang_codegen.DlangCodeGen* method), 69
 epilogue() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 73
 epilogue() (*schema_salad.java_codegen.JavaCodeGen* method), 82
 epilogue() (*schema_salad.python_codegen.PythonCodeGen* method), 115
 epilogue() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 136
 expand_url() (in module *schema_salad.metaschema*), 118
 expand_url() (in module *schema_salad.python_codegen_support*), 118

- `expand_url()` (*schema_salad.ref_resolver.Loader* method), 122
- `extend_and_specialize()` (in module *schema_salad.schema*), 128
- ## F
- `fetch()` (*schema_salad.ref_resolver.Loader* method), 123
- `fetch_text()` (*schema_salad.fetcher.DefaultFetcher* method), 77
- `fetch_text()` (*schema_salad.fetcher.Fetcher* method), 76
- `fetch_text()` (*schema_salad.tests.test_fetch.CWLTTestFetcher* method), 41
- `fetch_text()` (*schema_salad.tests.test_fetch.testFetcher* method), 40
- `Fetcher` (class in *schema_salad.fetcher*), 76
- `fetcher` (*schema_salad.metaschema.LoadingOptions* attribute), 94
- `fetcher` (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117
- `FetcherCallableType` (in module *schema_salad.utils*), 138
- `Field` (class in *schema_salad.avro.schema*), 21
- `FIELD_RESERVED_PROPS` (in module *schema_salad.avro.schema*), 18
- `FIELD_SORT_ORDER` (in module *schema_salad.codegen*), 59
- `FieldDefinition` (class in *schema_salad.cpp_codegen*), 65
- `fields` (*schema_salad.avro.schema.RecordSchema* property), 23
- `FieldType` (in module *schema_salad.utils*), 138
- `file()` (*schema_salad.sourceline.SourceLine* method), 132
- `file_uri()` (in module *schema_salad.metaschema*), 96
- `file_uri()` (in module *schema_salad.python_codegen_support*), 119
- `file_uri()` (in module *schema_salad.ref_resolver*), 120
- `fileuri` (*schema_salad.metaschema.LoadingOptions* attribute), 94
- `fileuri` (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117
- `fix_doc()` (in module *schema_salad.makedoc*), 88
- `fix_jsonld_ids()` (in module *schema_salad.jsonld_context*), 83
- `fixture_metaschema_doc()` (in module *schema_salad.tests.test_makedoc*), 44
- `flatten()` (in module *schema_salad.utils*), 138
- `floattype` (in module *schema_salad.metaschema*), 109
- `fmt()` (in module *schema_salad.python_codegen*), 112
- `friendly()` (in module *schema_salad.validate*), 141
- `fromDoc()` (*schema_salad.metaschema.ArraySchema* class method), 100
- `fromDoc()` (*schema_salad.metaschema.Documentation* class method), 108
- `fromDoc()` (*schema_salad.metaschema.EnumSchema* class method), 99
- `fromDoc()` (*schema_salad.metaschema.JsonldPredicate* class method), 102
- `fromDoc()` (*schema_salad.metaschema.RecordField* class method), 97
- `fromDoc()` (*schema_salad.metaschema.RecordSchema* class method), 98
- `fromDoc()` (*schema_salad.metaschema.SaladEnumSchema* class method), 107
- `fromDoc()` (*schema_salad.metaschema.SaladRecordField* class method), 104
- `fromDoc()` (*schema_salad.metaschema.SaladRecordSchema* class method), 106
- `fromDoc()` (*schema_salad.metaschema.Saveable* class method), 94
- `fromDoc()` (*schema_salad.metaschema.SpecializeDef* class method), 103
- `fromDoc()` (*schema_salad.python_codegen_support.Saveable* class method), 117
- `fullname` (*schema_salad.avro.schema.Name* property), 20
- ## G
- `generate_doc()` (in module *schema_salad.tests.test_makedoc*), 44
- `get_anon_name()` (in module *schema_salad.schema*), 127
- `get_data()` (in module *schema_salad.tests.util*), 58
- `get_data_uri()` (in module *schema_salad.tests.test_dotnet_codegen*), 32
- `get_data_uri()` (in module *schema_salad.tests.test_typescript_codegen*), 57
- `get_data_uri()` (in module *schema_salad.tests.util*), 58
- `get_metaschema()` (in module *schema_salad.schema*), 125
- `get_name()` (*schema_salad.avro.schema.Names* method), 20
- `get_other_props()` (in module *schema_salad.avro.schema*), 23
- `get_prop()` (*schema_salad.avro.schema.Field* method), 21
- `get_prop()` (*schema_salad.avro.schema.Schema* method), 19
- `get_space()` (*schema_salad.avro.schema.Name* method), 20
- `getid()` (*schema_salad.ref_resolver.Loader* method), 124

graph (*schema_salad.metaschema.LoadingOptions* property), 94

graph (*schema_salad.python_codegen_support.LoadingOptions* property), 117

H

has_name() (*schema_salad.avro.schema.Names* method), 20

has_types() (in module *schema_salad.makedoc*), 85

hasFieldValue() (in module *schema_salad.cpp_codegen*), 66

heading() (*schema_salad.makedoc.MyRenderer* method), 86

I

idmap_fields_union_of_None_type_or_array_of_RecordFieldLoader (in module *schema_salad.metaschema*), 110

idmap_fields_union_of_None_type_or_array_of_SaladRecordFieldLoader (in module *schema_salad.metaschema*), 110

idmap_loader() (*schema_salad.codegen_base.CodeGenBase* method), 62

idmap_loader() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 73

idmap_loader() (*schema_salad.java_codegen.JavaCodeGen* method), 81

idmap_loader() (*schema_salad.python_codegen.PythonCodeGen* method), 114

idmap_loader() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 136

idmap_specialize_union_of_None_type_or_array_of_SpecializeLoader (in module *schema_salad.metaschema*), 110

idx (*schema_salad.metaschema.LoadingOptions* attribute), 94

idx (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117

IdxResultType (in module *schema_salad.utils*), 138

IdxType (in module *schema_salad.metaschema*), 93

IdxType (in module *schema_salad.python_codegen_support*), 116

IdxType (in module *schema_salad.utils*), 138

imports (*schema_salad.metaschema.LoadingOptions* attribute), 94

imports (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117

includes (*schema_salad.metaschema.LoadingOptions* attribute), 94

includes (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117

indent() (in module *schema_salad.sourceline*), 131

inline_html() (*schema_salad.makedoc.MyRenderer* method), 86

INT_MAX_VALUE (in module *schema_salad.validate*), 140

INT_MIN_VALUE (in module *schema_salad.validate*), 140

interface_name() (*schema_salad.java_codegen.JavaCodeGen* method), 79

is_primitive_type() (in module *schema_salad.metaschema*), 109

is_fs_case_sensitive() (in module *schema_salad.tests.test_ref_resolver*), 52

is_subtype() (in module *schema_salad.avro.schema*), 24

isArray() (in module *schema_salad.cpp_codegen*), 67

isArraySchema() (in module *schema_salad.cpp_codegen*), 67

isEnumSchema() (in module *schema_salad.cpp_codegen*), 66

isolated_cache() (in module *schema_salad.tests.conftest*), 24

isPrimitiveType() (in module *schema_salad.cpp_codegen*), 66

isRecordSchema() (in module *schema_salad.cpp_codegen*), 66

items (*schema_salad.avro.schema.ArraySchema* property), 22

J

java_codegen() (in module *schema_salad.tests.test_java_codegen*), 43

JavaCodeGen (class in *schema_salad.java_codegen*), 79

json_dump() (in module *schema_salad.utils*), 139

json_dumps() (in module *schema_salad.utils*), 139

JsonDataType (in module *schema_salad.avro.schema*), 18

JsonDiffMatcherLoader (class in *schema_salad.tests.matcher*), 25

JsonldPredicate (class in *schema_salad.metaschema*), 101

JsonldPredicateLoader (in module *schema_salad.metaschema*), 109

L

leaves() (*schema_salad.exceptions.SchemaSaladException* method), 74

lineno_re (in module *schema_salad.sourceline*), 130

linkto() (in module *schema_salad.makedoc*), 86

load_and_validate() (in module *schema_salad.schema*), 126

load_cwl() (in module *schema_salad.tests.test_cwl11*), 30

load_cwl() (*schema_salad.tests.test_real_cwl.TestRealWorldCWL* method), 50

load_cwl() (*schema_salad.tests.test_schemas_directive.TestSchemasDirective* method), 55

load_document() (in module *schema_salad.metaschema*), 111

load_document_by_string() (in module *schema_salad.metaschema*), 111

- load_document_by_yaml() (in module *schema_salad.metaschema*), 111
- load_document_with_metadata() (in module *schema_salad.metaschema*), 111
- load_field() (in module *schema_salad.metaschema*), 95
- load_field() (in module *schema_salad.python_codegen_support*), 118
- load_schema() (in module *schema_salad.schema*), 126
- Loader (class in *schema_salad.ref_resolver*), 122
- LoadingOptions (class in *schema_salad.metaschema*), 93
- LoadingOptions (class in *schema_salad.python_codegen_support*), 116
- LONG_MAX_VALUE (in module *schema_salad.validate*), 140
- LONG_MIN_VALUE (in module *schema_salad.validate*), 140
- ## M
- main() (in module *schema_salad.main*), 84
- main() (in module *schema_salad.makedoc*), 89
- make_avro() (in module *schema_salad.schema*), 128
- make_avro_schema() (in module *schema_salad.schema*), 128
- make_avro_schema_from_avro() (in module *schema_salad.schema*), 129
- make_avsc_object() (in module *schema_salad.avro.schema*), 23
- make_field_objects() (*schema_salad.avro.schema.RecordSchema* static method), 23
- make_valid_avro() (in module *schema_salad.schema*), 127
- makedoc() (in module *schema_salad.makedoc*), 89
- makeError() (*schema_salad.sourceline.SourceLine* method), 132
- makeLead() (*schema_salad.sourceline.SourceLine* method), 132
- makerdf() (in module *schema_salad.jsonld_context*), 83
- MandatoryResolveType (in module *schema_salad.utils*), 138
- markdown_list_hook() (in module *schema_salad.makedoc*), 87
- maxDiff (in module *schema_salad.tests.test_cg*), 27
- MemoryCachingFetcher (class in *schema_salad.fetcher*), 77
- metaschema_file_uri (in module *schema_salad.tests.test_dotnet_codegen*), 32
- metaschema_file_uri (in module *schema_salad.tests.test_typescript_codegen*), 57
- metaschema_file_uri (in module *schema_salad.tests.util*), 58
- metaschema_loader (*schema_salad.tests.test_real_cwl.TestRealWorldCWL* attribute), 50
- metaschema_loader (*schema_salad.tests.test_schemas_directive.TestSchemasDirective* attribute), 55
- metaschema_pre() (in module *schema_salad.tests.test_cg*), 27
- module
- schema_salad*, 17
 - schema_salad.__main__*, 59
 - schema_salad.avro*, 17
 - schema_salad.avro.schema*, 17
 - schema_salad.codegen*, 59
 - schema_salad.codegen_base*, 60
 - schema_salad.cpp_codegen*, 63
 - schema_salad.dlang_codegen*, 68
 - schema_salad.dotnet_codegen*, 70
 - schema_salad.exceptions*, 74
 - schema_salad.fetcher*, 76
 - schema_salad.java_codegen*, 78
 - schema_salad.jsonld_context*, 82
 - schema_salad.main*, 84
 - schema_salad.makedoc*, 84
 - schema_salad.metaschema*, 90
 - schema_salad.python_codegen*, 112
 - schema_salad.python_codegen_support*, 115
 - schema_salad.ref_resolver*, 120
 - schema_salad.schema*, 124
 - schema_salad.sourceline*, 129
 - schema_salad.tests*, 24
 - schema_salad.tests.confstest*, 24
 - schema_salad.tests.matcher*, 24
 - schema_salad.tests.test_avro_names*, 25
 - schema_salad.tests.test_cg*, 26
 - schema_salad.tests.test_cli_args*, 28
 - schema_salad.tests.test_cpp_codegen*, 28
 - schema_salad.tests.test_cwl11*, 29
 - schema_salad.tests.test_dlang_codegen*, 30
 - schema_salad.tests.test_dotnet_codegen*, 31
 - schema_salad.tests.test_errors*, 33
 - schema_salad.tests.test_examples*, 36
 - schema_salad.tests.test_fetch*, 40
 - schema_salad.tests.test_fp*, 42
 - schema_salad.tests.test_java_codegen*, 42
 - schema_salad.tests.test_makedoc*, 43
 - schema_salad.tests.test_misc*, 46
 - schema_salad.tests.test_pickling*, 46
 - schema_salad.tests.test_print_online*, 47
 - schema_salad.tests.test_python_codegen*, 48
 - schema_salad.tests.test_real_cwl*, 50

- schema_salad.tests.test_ref_resolver, 51
 - schema_salad.tests.test_schema, 54
 - schema_salad.tests.test_schemas_directive, 54
 - schema_salad.tests.test_subtypes, 55
 - schema_salad.tests.test_typescript_codegen, 56
 - schema_salad.tests.util, 58
 - schema_salad.typescript_codegen, 133
 - schema_salad.utils, 136
 - schema_salad.validate, 139
 - MyRenderer (class in *schema_salad.makedoc*), 86
- ## N
- Name (class in *schema_salad.avro.schema*), 19
 - name (*schema_salad.avro.schema.NamedSchema* property), 21
 - NAMED_TYPES (in module *schema_salad.avro.schema*), 18
 - NamedSchema (class in *schema_salad.avro.schema*), 20
 - NamedType (class in *schema_salad.metaschema*), 103
 - Names (class in *schema_salad.avro.schema*), 20
 - namespaces (*schema_salad.metaschema.LoadingOptions* attribute), 94
 - namespaces (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117
 - None_type (in module *schema_salad.metaschema*), 109
 - NormDict (class in *schema_salad.ref_resolver*), 120
 - number_headings() (in module *schema_salad.makedoc*), 88
- ## O
- onWindows() (in module *schema_salad.utils*), 138
 - original_doc (*schema_salad.metaschema.LoadingOptions* attribute), 94
 - original_doc (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117
- ## P
- parse() (*schema_salad.cpp_codegen.CppCodeGen* method), 68
 - parse() (*schema_salad.dlang_codegen.DlangCodeGen* method), 70
 - parse_enum() (*schema_salad.dlang_codegen.DlangCodeGen* method), 70
 - parse_record_field() (*schema_salad.dlang_codegen.DlangCodeGen* method), 69
 - parse_record_field_type() (*schema_salad.dlang_codegen.DlangCodeGen* method), 69
 - parse_record_schema() (*schema_salad.dlang_codegen.DlangCodeGen* method), 70
 - parseEnum() (*schema_salad.cpp_codegen.CppCodeGen* method), 68
 - parser_info() (in module *schema_salad.metaschema*), 97
 - parseRecordField() (*schema_salad.cpp_codegen.CppCodeGen* method), 68
 - parseRecordSchema() (*schema_salad.cpp_codegen.CppCodeGen* method), 68
 - patch_fenced_code() (in module *schema_salad.makedoc*), 87
 - pred() (in module *schema_salad.cpp_codegen*), 67
 - pred() (in module *schema_salad.jsonld_context*), 82
 - prefix() (*schema_salad.exceptions.SchemaSaladException* method), 75
 - prefix_url() (in module *schema_salad.metaschema*), 96
 - prefix_url() (in module *schema_salad.python_codegen_support*), 119
 - pretty_str() (*schema_salad.exceptions.SchemaSaladException* method), 75
 - PRIMITIVE_TYPES (in module *schema_salad.avro.schema*), 18
 - primitives (in module *schema_salad.validate*), 141
 - PrimitiveSchema (class in *schema_salad.avro.schema*), 21
 - PrimitiveTypeLoader (in module *schema_salad.metaschema*), 109
 - prims (in module *schema_salad.dotnet_codegen*), 71
 - prims (in module *schema_salad.java_codegen*), 79
 - prims (in module *schema_salad.python_codegen*), 112
 - prims (in module *schema_salad.typescript_codegen*), 133
 - print_fieldrefs() (in module *schema_salad.schema*), 129
 - print_inheritance() (in module *schema_salad.schema*), 129
 - printrdf() (in module *schema_salad.main*), 84
 - process_type() (in module *schema_salad.jsonld_context*), 83
 - prologue() (*schema_salad.codegen_base.CodeGenBase* method), 60
 - prologue() (*schema_salad.dlang_codegen.DlangCodeGen* method), 69
 - prologue() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 71
 - prologue() (*schema_salad.java_codegen.JavaCodeGen* method), 79
 - prologue() (*schema_salad.python_codegen.PythonCodeGen* method), 113
 - prologue() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 134
 - propagate_sourceline()

- (*schema_salad.exceptions.SchemaSaladExceptions*.safe_name() (*schema_salad.python_codegen.PythonCodeGen* method), 74
- property_name() (*schema_salad.java_codegen.JavaCodeGen*.safe_name() (*schema_salad.typescript_codegen.TypeScriptCodeGen* static method), 79
- props (*schema_salad.avro.schema.Schema* property), 19
- PropsType (in module *schema_salad.avro.schema*), 18
- PropType (in module *schema_salad.avro.schema*), 18
- python_codegen() (in module *schema_salad.tests.test_python_codegen*), 49
- PythonCodeGen (class in *schema_salad.python_codegen*), 112
- ## R
- Record_nameLoader (in module *schema_salad.metaschema*), 110
- RecordField (class in *schema_salad.metaschema*), 97
- RecordFieldLoader (in module *schema_salad.metaschema*), 109
- RecordSchema (class in *schema_salad.avro.schema*), 23
- RecordSchema (class in *schema_salad.metaschema*), 98
- RecordSchemaLoader (in module *schema_salad.metaschema*), 109
- reflow() (in module *schema_salad.sourceline*), 131
- reflow_all() (in module *schema_salad.sourceline*), 130
- rename() (in module *schema_salad.sourceline*), 130
- render_type() (*schema_salad.makedoc.RenderType* method), 88
- RenderType (class in *schema_salad.makedoc*), 88
- replace_type() (in module *schema_salad.schema*), 127
- replaceKeywords() (in module *schema_salad.cpp_codegen*), 64
- resolve_all() (*schema_salad.ref_resolver.Loader* method), 123
- resolve_ref() (*schema_salad.ref_resolver.Loader* method), 123
- ResolvedRefType (in module *schema_salad.utils*), 138
- ResolveType (in module *schema_salad.utils*), 138
- rvocab (*schema_salad.metaschema.LoadingOptions* attribute), 94
- rvocab (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117
- ## S
- safe_name() (*schema_salad.codegen_base.CodeGenBase* static method), 61
- safe_name() (*schema_salad.dlang_codegen.DlangCodeGen* static method), 69
- safe_name() (*schema_salad.dotnet_codegen.DotNetCodeGen* static method), 71
- safe_name() (*schema_salad.java_codegen.JavaCodeGen* static method), 79
- safe_name() (*schema_salad.python_codegen.PythonCodeGen* static method), 113
- safe_name() (*schema_salad.typescript_codegen.TypeScriptCodeGen* static method), 134
- safename() (in module *schema_salad.cpp_codegen*), 64
- safename2() (in module *schema_salad.cpp_codegen*), 64
- SALAD_FILES (in module *schema_salad.schema*), 125
- salad_to_jsonld_context() (in module *schema_salad.jsonld_context*), 83
- SaladEnumSchema (class in *schema_salad.metaschema*), 106
- SaladEnumSchemaLoader (in module *schema_salad.metaschema*), 109
- saladp (in module *schema_salad.schema*), 125
- saladp (in module *schema_salad.validate*), 141
- SaladRecordField (class in *schema_salad.metaschema*), 104
- SaladRecordFieldLoader (in module *schema_salad.metaschema*), 109
- SaladRecordSchema (class in *schema_salad.metaschema*), 105
- SaladRecordSchemaLoader (in module *schema_salad.metaschema*), 109
- save() (in module *schema_salad.metaschema*), 95
- save() (in module *schema_salad.python_codegen_support*), 118
- save() (*schema_salad.metaschema.ArraySchema* method), 101
- save() (*schema_salad.metaschema.Documentation* method), 108
- save() (*schema_salad.metaschema.EnumSchema* method), 100
- save() (*schema_salad.metaschema.JsonldPredicate* method), 102
- save() (*schema_salad.metaschema.RecordField* method), 98
- save() (*schema_salad.metaschema.RecordSchema* method), 99
- save() (*schema_salad.metaschema.SaladEnumSchema* method), 107
- save() (*schema_salad.metaschema.SaladRecordField* method), 104
- save() (*schema_salad.metaschema.SaladRecordSchema* method), 106
- save() (*schema_salad.metaschema.Saveable* method), 95
- save() (*schema_salad.metaschema.SpecializeDef* method), 103
- save() (*schema_salad.python_codegen_support.Saveable* method), 117
- save_relative_uri() (in module *schema_salad.metaschema*), 96
- save_relative_uri() (in module *schema_salad.metaschema*), 96

schema_salad.python_codegen_support),
 119
 save_type (in module *schema_salad.metaschema*), 95
 save_type (in module *schema_salad.python_codegen_support*),
 118
 save_with_metadata() (in module *schema_salad.metaschema*), 95
 save_with_metadata() (in module *schema_salad.python_codegen_support*),
 118
 Saveable (class in *schema_salad.metaschema*), 94
 Saveable (class in *schema_salad.python_codegen_support*), 117
 schema
 schema-salad-doc command line option, 16
 schema-salad-tool command line option, 14
 Schema (class in *schema_salad.avro.schema*), 19
 schema_metadata (*schema_salad.tests.test_real_cwl.TestRealCWL* command line
 attribute), 50
 schema_metadata (*schema_salad.tests.test_schemas_directive.SchemaSaladDirective*
 attribute), 55
 SCHEMA_RESERVED_PROPS (in module *schema_salad.avro.schema*), 18
 schema_salad
 module, 17
 schema_salad.__main__
 module, 59
 schema_salad.avro
 module, 17
 schema_salad.avro.schema
 module, 17
 schema_salad.codegen
 module, 59
 schema_salad.codegen_base
 module, 60
 schema_salad.cpp_codegen
 module, 63
 schema_salad.dlang_codegen
 module, 68
 schema_salad.dotnet_codegen
 module, 70
 schema_salad.exceptions
 module, 74
 schema_salad.fetcher
 module, 76
 schema_salad.java_codegen
 module, 78
 schema_salad.jsonld_context
 module, 82
 schema_salad.main
 module, 84
 schema_salad.makedoc
 module, 84
 schema_salad.metaschema
 module, 90
 schema_salad.python_codegen
 module, 112
 schema_salad.python_codegen_support
 module, 115
 schema_salad.ref_resolver
 module, 120
 schema_salad.schema
 module, 124
 schema_salad.sourceline
 module, 129
 schema_salad.tests
 module, 24
 schema_salad.tests.confstest
 module, 24
 schema_salad.tests.matcher
 module, 24
 schema_salad.tests.test_avro_names
 module, 25
 schema_salad.tests.test_cg
 module, 26
 schema_salad.tests.test_cli_args
 module, 28
 schema_salad.tests.test_cpp_codegen
 module, 28
 schema_salad.tests.test_cwl11
 module, 29
 schema_salad.tests.test_dlang_codegen
 module, 30
 schema_salad.tests.test_dotnet_codegen
 module, 31
 schema_salad.tests.test_errors
 module, 33
 schema_salad.tests.test_examples
 module, 36
 schema_salad.tests.test_fetch
 module, 40
 schema_salad.tests.test_fp
 module, 42
 schema_salad.tests.test_java_codegen
 module, 42
 schema_salad.tests.test_makedoc
 module, 43
 schema_salad.tests.test_misc
 module, 46
 schema_salad.tests.test_pickling
 module, 46
 schema_salad.tests.test_print_online
 module, 47
 schema_salad.tests.test_python_codegen
 module, 48
 schema_salad.tests.test_real_cwl
 module, 50

schema_salad.tests.test_ref_resolver
 module, 51
 schema_salad.tests.test_schema
 module, 54
 schema_salad.tests.test_schemas_directive
 module, 54
 schema_salad.tests.test_subtypes
 module, 55
 schema_salad.tests.test_typescript_codegen
 module, 56
 schema_salad.tests.util
 module, 58
 schema_salad.typescript_codegen
 module, 133
 schema_salad.utils
 module, 136
 schema_salad.validate
 module, 139
 schema_type (in module *schema_salad.schema*), 126
 schema-salad-doc command line option
 --brand, 16
 --brandinverse, 16
 --brandlink, 16
 --brandstyle, 16
 --debug, 16
 --help, 16
 --only, 16
 --printtype, 16
 --redirect, 16
 -h, 16
 schema, 16
 schema-salad-tool command line option
 --brand, 15
 --brandinverse, 15
 --brandlink, 15
 --brandstyle, 15
 --codegen, 14
 --codegen-copyright, 15
 --codegen-examples, 14
 --codegen-package, 15
 --codegen-parser-info, 15
 --codegen-target, 14
 --debug, 15
 --help, 14
 --non-strict, 15
 --only, 15
 --printtype, 15
 --print-avro, 14
 --print-doc, 15
 --print-fieldrefs-dot, 14
 --print-index, 14
 --print-inheritance-dot, 14
 --print-jsonld-context, 14
 --print-metadata, 14
 --print-oneline, 15
 --print-pre, 14
 --print-rdf, 14
 --print-rdfs, 14
 --quiet, 15
 --rdf-serializer, 14
 --redirect, 15
 --skip-schemas, 14
 --strict, 15
 --strict-foreign-properties, 14
 --verbose, 15
 --version, 15
 -h, 14
 -v, 15
 document, 14
 schema, 14
 SchemaDefinedType (class in *schema_salad.metaschema*), 103
 SchemaException, 75
 SchemaParseException, 19
 schemas (*schema_salad.avro.schema.UnionSchema* property), 22
 schemas (*schema_salad.metaschema.LoadingOptions* attribute), 94
 schemas (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117
 SchemaSaladException, 74
 SchemaType (in module *schema_salad.tests.test_cwl11*), 29
 schemes (*schema_salad.fetcher.Fetcher* attribute), 76
 secondaryfilesdsl_loader() (*schema_salad.codegen_base.CodeGenBase* method), 63
 secondaryfilesdsl_loader() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 73
 secondaryfilesdsl_loader() (*schema_salad.java_codegen.JavaCodeGen* method), 82
 secondaryfilesdsl_loader() (*schema_salad.python_codegen.PythonCodeGen* method), 115
 secondaryfilesdsl_loader() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 136
 set_prop() (*schema_salad.avro.schema.Field* method), 21
 set_prop() (*schema_salad.avro.schema.Schema* method), 19
 setup_class() (*schema_salad.tests.test_real_cwl.TestRealWorldCWL* class method), 50
 setup_class() (*schema_salad.tests.test_schemas_directive.TestSchemasD* class method), 55
 shortname() (in module *schema_salad.metaschema*), 97

shortname() (in module *schema_salad.python_codegen_support*), 119
 shortname() (in module *schema_salad.schema*), 129
 SourceLine (class in *schema_salad.sourceline*), 132
 SpecializeDef (class in *schema_salad.metaschema*), 102
 SpecializeDefLoader (in module *schema_salad.metaschema*), 109
 split_field() (in module *schema_salad.cpp_codegen*), 65
 split_name() (in module *schema_salad.cpp_codegen*), 64
 start() (*schema_salad.sourceline.SourceLine* method), 132
 stdout() (in module *schema_salad.utils*), 139
 strip_dup_lineno() (in module *schema_salad.sourceline*), 131
 strip_duplicated_lineno() (in module *schema_salad.sourceline*), 131
 StripYAMLComments() (in module *schema_salad.tests.matcher*), 25
 strtype (in module *schema_salad.metaschema*), 108
 SubLoader() (in module *schema_salad.ref_resolver*), 121
 summary() (*schema_salad.exceptions.SchemaSaladException* method), 75
 supported_schemes() (*schema_salad.fetcher.Fetcher* method), 77
 symbols (*schema_salad.avro.schema.EnumSchema* property), 22

T

test_attachments() (in module *schema_salad.tests.test_ref_resolver*), 53
 test_avro_loading() (in module *schema_salad.tests.test_avro_names*), 25
 test_avro_loading_subtype() (in module *schema_salad.tests.test_subtypes*), 56
 test_avro_loading_subtype_bad() (in module *schema_salad.tests.test_subtypes*), 56
 test_avro_regression() (in module *schema_salad.tests.test_examples*), 38
 test_bad_schema() (in module *schema_salad.tests.test_errors*), 34
 test_bad_schema2() (in module *schema_salad.tests.test_errors*), 35
 test_bad_schemas() (in module *schema_salad.tests.test_examples*), 37
 test_blank_node_id() (in module *schema_salad.tests.test_examples*), 40
 test_cache() (in module *schema_salad.tests.test_fetch*), 42
 test_can_use_Any() (in module *schema_salad.tests.test_examples*), 40
 test_check_exists_follows_redirects() (in module *schema_salad.tests.test_ref_resolver*), 53
 test_class_field() (in module *schema_salad.tests.test_dotnet_codegen*), 32
 test_class_field() (in module *schema_salad.tests.test_typescript_codegen*), 57
 test_cmap() (in module *schema_salad.tests.test_examples*), 40
 test_cwl_cpp_gen() (in module *schema_salad.tests.test_cpp_codegen*), 28
 test_cwl_dlang_gen() (in module *schema_salad.tests.test_dlang_codegen*), 31
 test_cwl_gen() (in module *schema_salad.tests.test_dotnet_codegen*), 32
 test_cwl_gen() (in module *schema_salad.tests.test_java_codegen*), 42
 test_cwl_gen() (in module *schema_salad.tests.test_python_codegen*), 48
 test_cwl_gen() (in module *schema_salad.tests.test_typescript_codegen*), 57
 test_default_parser_info() (in module *schema_salad.tests.test_python_codegen*), 49
 test_DefaultFetcher_urljoin_linux() (in module *schema_salad.tests.test_ref_resolver*), 53
 test_DefaultFetcher_urljoin_win32() (in module *schema_salad.tests.test_ref_resolver*), 53
 test_detect_changes_in_html() (in module *schema_salad.tests.test_makedoc*), 45
 test_dir_name (in module *schema_salad.tests.test_cwl11*), 29
 test_dir_name (in module *schema_salad.tests.test_real_cwl*), 50
 test_dir_name (in module *schema_salad.tests.test_schemas_directive*), 55
 test_doc_fenced_code_contents_preserved() (in module *schema_salad.tests.test_makedoc*), 44
 test_doc_headings_target_anchor() (in module *schema_salad.tests.test_makedoc*), 44
 test_doc_render_table_of_contents() (in module *schema_salad.tests.test_makedoc*), 45
 test_dollarsign_schema() (*schema_salad.tests.test_schemas_directive.TestSchemasDirective* method), 55
 test_embedded_html_unescaped() (in module *schema_salad.tests.test_makedoc*), 45

test_empty_input() (in module *schema_salad.tests.test_cli_args*), 28
 test_err() (in module *schema_salad.tests.test_cg*), 26
 test_err2() (in module *schema_salad.tests.test_cg*), 27
 test_error_message1() (in module *schema_salad.tests.test_errors*), 34
 test_error_message10() (in module *schema_salad.tests.test_errors*), 34
 test_error_message11() (in module *schema_salad.tests.test_errors*), 34
 test_error_message15() (in module *schema_salad.tests.test_errors*), 34
 test_error_message2() (in module *schema_salad.tests.test_errors*), 34
 test_error_message3() (in module *schema_salad.tests.test_errors*), 34
 test_error_message4() (in module *schema_salad.tests.test_errors*), 34
 test_error_message5() (in module *schema_salad.tests.test_errors*), 34
 test_error_message7() (in module *schema_salad.tests.test_errors*), 34
 test_error_message8() (in module *schema_salad.tests.test_errors*), 34
 test_error_message9() (in module *schema_salad.tests.test_errors*), 34
 test_errors() (in module *schema_salad.tests.test_errors*), 33
 test_errors_previously_defined_dict_key() (in module *schema_salad.tests.test_errors*), 34
 test_examples() (in module *schema_salad.tests.test_examples*), 38
 test_extend_and_specialize_enums() (in module *schema_salad.tests.test_pickling*), 46
 test_extend_and_specialize_enums() (in module *schema_salad.tests.test_schema*), 54
 test_fetch_inject_id() (in module *schema_salad.tests.test_ref_resolver*), 53
 test_fetcher() (in module *schema_salad.tests.test_fetch*), 42
 test_file_uri() (in module *schema_salad.tests.test_examples*), 39
 test_for_invalid_yaml1() (in module *schema_salad.tests.test_print_oneline*), 47
 test_for_invalid_yaml2() (in module *schema_salad.tests.test_print_oneline*), 47
 test_fp() (in module *schema_salad.tests.test_fp*), 42
 test_fragment() (in module *schema_salad.tests.test_examples*), 39
 test_graph_property() (in module *schema_salad.tests.test_python_codegen*), 49
 test_graph_property_cache() (in module *schema_salad.tests.test_python_codegen*), 49
 test_graph_property_empty_schema() (in module *schema_salad.tests.test_python_codegen*), 50
 test_h3agatk_SNP() (in module *schema_salad.tests.test_real_cwl.TestRealWorldCWL* method), 51
 test_h3agatk_WES() (in module *schema_salad.tests.test_real_cwl.TestRealWorldCWL* method), 51
 test_icgc_pancan() (in module *schema_salad.tests.test_real_cwl.TestRealWorldCWL* method), 51
 test_idmap() (in module *schema_salad.tests.test_cg*), 27
 test_idmap() (in module *schema_salad.tests.test_examples*), 38
 test_idmap2() (in module *schema_salad.tests.test_cg*), 27
 test_import() (in module *schema_salad.tests.test_cg*), 27
 test_import2() (in module *schema_salad.tests.test_cg*), 27
 test_import_list() (in module *schema_salad.tests.test_ref_resolver*), 53
 test_include() (in module *schema_salad.tests.test_cg*), 26
 test_jsonld_ctx() (in module *schema_salad.tests.test_examples*), 38
 test_load() (in module *schema_salad.tests.test_cg*), 26
 test_load_by_yaml metaschema() (in module *schema_salad.tests.test_cg*), 27
 test_load_cwlschema() (in module *schema_salad.tests.test_cg*), 27
 test_load_metaschema() (in module *schema_salad.tests.test_cg*), 27
 test_load_pt() (in module *schema_salad.tests.test_cg*), 27
 test_load_schema_cache() (in module *schema_salad.tests.test_misc*), 46
 test Loader_initialisation_disable_doc_cache() (in module *schema_salad.tests.test_ref_resolver*), 53
 test Loader_initialisation_for_HOME_env_var() (in module *schema_salad.tests.test_ref_resolver*), 52
 test Loader_initialisation_for_TMP_env_var() (in module *schema_salad.tests.test_ref_resolver*), 52
 test Loader_initialisation_with_neither_TMP_HOME_set() (in module *schema_salad.tests.test_ref_resolver*), 53
 test_meta_schema_gen() (in module *schema_salad.tests.test_dotnet_codegen*), 32
 test_meta_schema_gen() (in module *schema_salad.tests.test_java_codegen*), 43
 test_meta_schema_gen() (in module

schema_salad.tests.test_python_codegen), 48

`test_meta_schema_gen()` (in module *schema_salad.tests.test_typescript_codegen*), 57

`test_meta_schema_gen_no_base()` (in module *schema_salad.tests.test_python_codegen*), 49

`test_meta_schema_gen_up_to_date()` (in module *schema_salad.tests.test_python_codegen*), 48

`test_misc()` (in module *schema_salad.tests.test_misc*), 46

`test_mixin()` (in module *schema_salad.tests.test_examples*), 39

`test_multiline_list_entries_without_indentation()` (in module *schema_salad.tests.test_makedoc*), 45

`test_multiline_list_entries_word_spacing()` (in module *schema_salad.tests.test_makedoc*), 45

`test_namespaces_type()` (in module *schema_salad.tests.test_errors*), 35

`test_namespaces_undeclared()` (in module *schema_salad.tests.test_errors*), 35

`test_nested_typedsl_ref()` (in module *schema_salad.tests.test_examples*), 39

`test_not_a_namespace1()` (in module *schema_salad.tests.test_errors*), 35

`test_not_a_namespace2()` (in module *schema_salad.tests.test_errors*), 35

`test_not_a_namespace3()` (in module *schema_salad.tests.test_errors*), 35

`test_nullable_links()` (in module *schema_salad.tests.test_examples*), 40

`test_outputBinding()` (in module *schema_salad.tests.test_cwl11*), 30

`test_parser_info()` (in module *schema_salad.tests.test_python_codegen*), 49

`test_plain_links_autolinked()` (in module *schema_salad.tests.test_makedoc*), 45

`test_print_index()` (in module *schema_salad.tests.test_examples*), 38

`test_print_metadata()` (in module *schema_salad.tests.test_examples*), 38

`test_print_online()` (in module *schema_salad.tests.test_print_online*), 47

`test_print_online_for_errors_in_resolve_ref()` (in module *schema_salad.tests.test_print_online*), 47

`test_print_online_for_errors_in_the_same_line()` (in module *schema_salad.tests.test_print_online*), 47

`test_print_online_for_invalid_yaml()` (in module *schema_salad.tests.test_print_online*), 47

`test_print_pre()` (in module *schema_salad.tests.test_examples*), 38

`test_print_pre_schema()` (in module *schema_salad.tests.test_examples*), 37

`test_print_rdf()` (in module *schema_salad.tests.test_examples*), 37

`test_print_rdf_invalid_external_ref()` (in module *schema_salad.tests.test_examples*), 37

`test_print_schema_index()` (in module *schema_salad.tests.test_examples*), 38

`test_print_schema_metadata()` (in module *schema_salad.tests.test_examples*), 38

`test_rdf_datetime()` (in module *schema_salad.tests.test_examples*), 39

`test_recordschema_pickle()` (in module *schema_salad.tests.test_pickling*), 46

`test_resolve_missing_step_id()` (in module *schema_salad.tests.test_ref_resolver*), 53

`test_safe_identifiers()` (in module *schema_salad.tests.test_python_codegen*), 48

`test_schema_salad_doc_online_doc()` (in module *schema_salad.tests.test_examples*), 38

`test_schema_salad_inherit_docs()` (in module *schema_salad.tests.test_makedoc*), 44

`test_schemas()` (in module *schema_salad.tests.test_examples*), 37

`test_schemas_type()` (in module *schema_salad.tests.test_errors*), 35

`test_scoped_id()` (in module *schema_salad.tests.test_examples*), 39

`test_scoped_ref()` (in module *schema_salad.tests.test_examples*), 38

`test_secondaryFile_dsl_ref()` (in module *schema_salad.tests.test_examples*), 39

`test_secondaryFiles()` (in module *schema_salad.tests.test_cwl11*), 30

`test_self_validate()` (in module *schema_salad.tests.test_examples*), 37

`test_shortcode()` (in module *schema_salad.tests.test_cg*), 27

`test_skip_bad_schemas()` (in module *schema_salad.tests.test_examples*), 37

`test_sourceline()` (in module *schema_salad.tests.test_examples*), 39

`test_subscoped_id()` (in module *schema_salad.tests.test_examples*), 39

`test_subtypes()` (in module *schema_salad.tests.test_subtypes*), 56

`test_topmed_single_doc()` (*schema_salad.tests.test_real_cwl.TestRealWorldCWL* method), 50

`test_typedsl_ref()` (in module *schema_salad.tests.test_examples*), 39

- test_use_of_package_for_parser_info() (in module *schema_salad.tests.test_python_codegen*), 49
- test_version() (in module *schema_salad.tests.test_cli_args*), 28
- test_yaml_datetime() (in module *schema_salad.tests.test_examples*), 39
- test_yaml_float_test() (in module *schema_salad.tests.test_examples*), 39
- test_yaml_tab_error() (in module *schema_salad.tests.test_cwl11*), 30
- testFetcher (class in *schema_salad.tests.test_fetch*), 40
- TestRealWorldCWL (class in *schema_salad.tests.test_real_cwl*), 50
- TestSchemasDirective (class in *schema_salad.tests.test_schemas_directive*), 55
- text() (*schema_salad.makedoc.MyRenderer* method), 86
- tmp_dir_fixture() (in module *schema_salad.tests.test_ref_resolver*), 52
- to_doc_comment() (*schema_salad.dlang_codegen.DlangCodeGen* method), 69
- to_dotnet() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 73
- to_id() (in module *schema_salad.makedoc*), 87
- to_java() (*schema_salad.java_codegen.JavaCodeGen* method), 81
- to_one_line_messages() (in module *schema_salad.exceptions*), 76
- to_typescript() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 135
- to_validation_exception() (in module *schema_salad.ref_resolver*), 120
- ToC (class in *schema_salad.makedoc*), 87
- type_loader() (*schema_salad.codegen_base.CodeGenBase* method), 61
- type_loader() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 72
- type_loader() (*schema_salad.java_codegen.JavaCodeGen* method), 80
- type_loader() (*schema_salad.python_codegen.PythonCodeGen* method), 113
- type_loader() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 134
- type_loader_enum() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 72
- type_loader_enum() (*schema_salad.java_codegen.JavaCodeGen* method), 80
- type_loader_enum() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 135
- TypeDef (class in *schema_salad.codegen_base*), 60
- typedsl_Array_nameLoader_2 (in module *schema_salad.metaschema*), 110
- typedsl_Documentation_nameLoader_2 (in module *schema_salad.metaschema*), 111
- typedsl_Enum_nameLoader_2 (in module *schema_salad.metaschema*), 110
- typedsl_loader() (*schema_salad.codegen_base.CodeGenBase* method), 62
- typedsl_loader() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 73
- typedsl_loader() (*schema_salad.java_codegen.JavaCodeGen* method), 81
- typedsl_loader() (*schema_salad.python_codegen.PythonCodeGen* method), 115
- typedsl_loader() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 136
- typedsl_Record_nameLoader_2 (in module *schema_salad.metaschema*), 110
- typedsl_union_of_PrimitiveTypeLoader_or_RecordSchemaLoader (in module *schema_salad.metaschema*), 109
- typefmt() (*schema_salad.makedoc.RenderType* method), 88
- types (in module *schema_salad.tests.test_subtypes*), 56
- typescript_codegen() (in module *schema_salad.tests.test_typescript_codegen*), 57
- TypeScriptCodeGen (class in *schema_salad.typescript_codegen*), 133
- ## U
- union_of_None_type_or_Any_type (in module *schema_salad.metaschema*), 110
- union_of_None_type_or_array_of_RecordFieldLoader (in module *schema_salad.metaschema*), 110
- union_of_None_type_or_array_of_SaladRecordFieldLoader (in module *schema_salad.metaschema*), 110
- union_of_None_type_or_array_of_SpecializeDefLoader (in module *schema_salad.metaschema*), 110
- union_of_None_type_or_booltype (in module *schema_salad.metaschema*), 110
- union_of_None_type_or_inttype (in module *schema_salad.metaschema*), 110
- union_of_None_type_or_strtype (in module *schema_salad.metaschema*), 110
- union_of_None_type_or_strtype_or_array_of_strtype (in module *schema_salad.metaschema*), 109
- union_of_None_type_or_strtype_or_JsonldPredicateLoader (in module *schema_salad.metaschema*), 110
- union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader (in module *schema_salad.metaschema*), 109
- union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader_or_JsonldPredicateLoader (in module *schema_salad.metaschema*), 109
- union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader_or_JsonldPredicateLoader (in module *schema_salad.metaschema*), 111
- union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader_or_UnionSchemaLoader (in module *schema_salad.metaschema*), 111
- UnionSchema (class in *schema_salad.avro.schema*), 22

uri_array_of_strtype_True_False_None (in module *schema_salad.metaschema*), 110

uri_file_path() (in module *schema_salad.ref_resolver*), 120

uri_loader() (*schema_salad.codegen_base.CodeGenBase* method), 62

uri_loader() (*schema_salad.dotnet_codegen.DotNetCodeGen* method), 73

uri_loader() (*schema_salad.java_codegen.JavaCodeGen* method), 81

uri_loader() (*schema_salad.python_codegen.PythonCodeGen* method), 114

uri_loader() (*schema_salad.typescript_codegen.TypeScriptCodeGen* method), 135

uri_strtype_False_False_1 (in module *schema_salad.metaschema*), 110

uri_strtype_True_False_None (in module *schema_salad.metaschema*), 109

uri_union_of_None_type_or_strtype_False_False_None (in module *schema_salad.metaschema*), 110

uri_union_of_None_type_or_strtype_or_array_of_strtype_False_False_1 (in module *schema_salad.metaschema*), 110

uri_union_of_None_type_or_strtype_or_array_of_strtype_False_False_None (in module *schema_salad.metaschema*), 110

uri_union_of_None_type_or_strtype_True_False_None (in module *schema_salad.metaschema*), 110

uri_union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader_or_ArraySchemaLoader_or_strtype (in module *schema_salad.metaschema*), 110

urljoin() (*schema_salad.fetcher.DefaultFetcher* method), 78

urljoin() (*schema_salad.fetcher.Fetcher* method), 77

urljoin() (*schema_salad.tests.test_fetch.CWLTTestFetcher* method), 41

urljoin() (*schema_salad.tests.test_fetch.testFetcher* method), 41

USE_ONE_OR_LIST_OF_TYPES (in module *schema_salad.java_codegen*), 78

V

VALID_FIELD_SORT_ORDERS (in module *schema_salad.avro.schema*), 18

VALID_TYPES (in module *schema_salad.avro.schema*), 18

validate() (in module *schema_salad.validate*), 140

validate_doc() (in module *schema_salad.schema*), 126

validate_ex() (in module *schema_salad.validate*), 141

validate_link() (*schema_salad.ref_resolver.Loader* method), 124

validate_links() (*schema_salad.ref_resolver.Loader* method), 124

validate_scoped() (*schema_salad.ref_resolver.Loader* method), 123

ValidationException, 75

vocab (*schema_salad.metaschema.LoadingOptions* attribute), 94

vocab (*schema_salad.python_codegen_support.LoadingOptions* attribute), 117

vocab_type_name() (in module *schema_salad.makedoc*), 85

wrap_format() (in module *schema_salad.validate*), 141

with_source_line() (*schema_salad.exceptions.SchemaSaladException* method), 74

writeDefinition() (*schema_salad.cpp_codegen.ClassDefinition* method), 65

writeDefinition() (*schema_salad.cpp_codegen.EnumDefinition* method), 66

writeDefinition() (*schema_salad.cpp_codegen.FieldDefinition* method), 65

writeFwdDeclaration() (*schema_salad.cpp_codegen.ClassDefinition* method), 65

writeTmpDefinition() (*schema_salad.cpp_codegen.ClassDefinition* method), 65

W

Y

yaml_no_ts() (in module *schema_salad.utils*), 139