

---

# **Schema Salad**

**Peter Amstutz and Common Workflow Language Project contributors**

**Apr 10, 2024**



# CONTENTS

<b>1 Installation</b>	<b>3</b>
<b>2 Commands</b>	<b>5</b>
<b>3 Codegen Examples</b>	<b>7</b>
<b>4 Quick Start</b>	<b>9</b>
<b>5 Documentation</b>	<b>11</b>
<b>6 Rationale</b>	<b>13</b>
6.1 Command Line Options . . . . .	13
6.2 How to add new types to the local Typeshed . . . . .	16
6.3 API Reference . . . . .	17
6.4 Indices and tables . . . . .	155
<b>Python Module Index</b>	<b>157</b>
<b>Index</b>	<b>159</b>



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 3.8+ only.



---

**CHAPTER  
ONE**

---

## **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
```



---

## CHAPTER TWO

---

## 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] [--primtype 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
```

---

CHAPTER  
THREE

---

## CODEGEN EXAMPLES

The examples in the tables below 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.

First set of examples is using the CWL v1.2 schema:

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

Second set of examples is for the Galaxy Workflow Format 2 schema:

Language	Path
Python	<a href="https://github.com/galaxyproject/gxformat2/blob/master/gxformat2/schema/v19_09.py">https://github.com/galaxyproject/gxformat2/blob/master/gxformat2/schema/v19_09.py</a>
Java	<a href="https://github.com/galaxyproject/gxformat2/tree/master/java">https://github.com/galaxyproject/gxformat2/tree/master/java</a>
TypeScript	<a href="https://github.com/galaxyproject/gxformat2/tree/master/typescript">https://github.com/galaxyproject/gxformat2/tree/master/typescript</a>



---

CHAPTER  
FOUR

---

## 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)

(continued from previous 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
```

---

**CHAPTER  
FIVE**

---

**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 an popular way to represent structured data. It is attractive because of it's 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-spdx-copyright-text spdx_copyright_text [spdx_]
```

(continues on next page)

(continued from previous page)

```
↳copyright_text ...]]  
    [--codegen-spdx-license-identifier spdx_license_identifier]  
    [--codegen-parser-info parser_info] [--print-oneline]  
    [--print-doc] [--strict | --non-strict]  
    [--verbose | --quiet | --debug] [--only ONLY]  
    [--redirect REDIRECT] [--brand BRAND]  
    [--brandlink BRANDLINK] [--brandstyle BRANDSTYLE]  
    [--brandinverse] [--primtype 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-spdx-copyright-text <spdx_copyright_text>
    List of copyright text. Each entry will show up as 'SPDX-FileCopyrightText: ...' (Currently c++ only)

--codegen-spdx-license-identifier <spdx_license_identifier>
    Optional SPDX license identifier, e.g.: GPL-3.0-only (Currently c++ only)

--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

**--primtype <primtype>**

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]
                         [--primtype PRIMTYPE] [--debug]
                         schema
```

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

show this help message and exit

**--only <only>****--redirect <redirect>****--brand <brand>****--brandlink <brandlink>****--brandstyle <brandstyle>****--brandinverse****--primtype <primtype>****--debug**

## 6.2 How to add new types to the local Typeshed

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 documentation<sup>1</sup>.

### 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 map 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

<a href="#">Schema</a>	Base class for all Schema classes.
<a href="#">Name</a>	Class to describe Avro name.
<a href="#">Names</a>	Track name set and default namespace during parsing.
<a href="#">NamedSchema</a>	Named Schemas specified in NAMED_TYPES.
<a href="#">Field</a>	
<a href="#">PrimitiveSchema</a>	Valid primitive types are in PRIMITIVE_TYPES.
<a href="#">EnumSchema</a>	Named Schemas specified in NAMED_TYPES.
<a href="#">ArraySchema</a>	Avro array schema class.
<a href="#">MapSchema</a>	Avro map schema class.
<a href="#">NamedMapSchema</a>	Avro named map schema class.
<a href="#">UnionSchema</a>	Avro union schema class.
<a href="#">NamedUnionSchema</a>	Avro named union schema class.
<a href="#">RecordSchema</a>	Named Schemas specified in NAMED_TYPES.

---

<sup>1</sup> Created with sphinx-autoapi

### Functions

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

### Attributes

<code>PRIMITIVE_TYPES</code>
<code>NAMED_TYPES</code>
<code>VALID_TYPES</code>
<code>SCHEMA_RESERVED_PROPS</code>
<code>JsonDataType</code>
<code>AtomicPropType</code>
<code>PropType</code>
<code>PropsType</code>
<code>FIELD_RESERVED_PROPS</code>
<code>VALID_FIELD_SORT_ORDERS</code>

---

```
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', 'names', 'symbols', 'values', '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`

Avro array schema class.

**Parameters**

- **items** (`JsonDataType`) –
- **names** (`Names`) –
- **other\_props** (*Optional*[`PropsType`]) –

**property items:** *Schema*

Avro schema describing the array items' type.

**Return type***Schema***class** schema\_salad.avro.schema.**MapSchema**(*values*, *names*, *other\_props*=None)

Bases: *Schema*

Avro map schema class.

**Parameters**

- **values** (*JsonDataType*) –
- **names** (*Names*) –
- **other\_props** (*Optional[PropsType]*) –

**property values:** *Schema*

Avro schema describing the map values' type.

**Return type***Schema***class** schema\_salad.avro.schema.**NamedMapSchema**(*values*, *names*, *name*, *namespace*=None, *doc*=None, *other\_props*=None)

Bases: *NamedSchema*

Avro named map schema class.

**Parameters**

- **values** (*JsonDataType*) –
- **names** (*Names*) –
- **name** (*str*) –
- **namespace** (*Optional[str]*) –
- **doc** (*Optional[Union[str, List[str]]]*) –
- **other\_props** (*Optional[PropsType]*) –

**property values:** *Schema*

Avro schema describing the map values' type.

**Return type***Schema***class** schema\_salad.avro.schema.**UnionSchema**(*schemas*, *names*)

Bases: *Schema*

Avro union schema class.

**Parameters**

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

- **names** ([Names](#)) –

**property schemas:** `List[Schema]`

Avro schemas composing the Union type.

**Return type**

`List[Schema]`

**class** `schema_salad.avro.schema.NamedUnionSchema(schemas, names, name, namespace=None, doc=None)`

Bases: `NamedSchema`

Avro named union schema class.

**Parameters**

- **schemas** (`List[JsonDataType]`) –
- **names** ([Names](#)) –
- **name** ([str](#)) –
- **namespace** (`Optional[str]`) –
- **doc** (`Optional[Union[str, List[str]]]`) –

**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(types, existing, new)`

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

**Parameters**

- **types** (*Dict[str, Any]*) –
- **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**

<code>test_avro_loading()</code>	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_shortname()</code>	Test shortname() 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_cwl_schema()</code>
-------------------------------------

### Attributes

*maxDiff*

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\_shortname()**

Test shortname() function.

**Return type**

None

```
schema_salad.tests.test_cg.metaschema_pre()
```

Pre-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\_codegen\_errors

Tests of helpful error messages.

#### Module Contents

##### Functions

---

```
test_error_message1(tmp_path)
test_error_message2(tmp_path)
test_error_message4(tmp_path)
test_error_message5(tmp_path)
test_error_message6(tmp_path)
test_error_message7(tmp_path)
test_error_message8(tmp_path)
test_error_message9(tmp_path)
test_error_message10(tmp_path)
test_error_message11(tmp_path)
test_error_message15(tmp_path)
load_document_by_uri(tmp_path, path)
python_codegen(file_uri, target[, parser_info, package])
```

---

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message1**(tmp\_path)

##### Parameters

**tmp\_path** (*pathlib.Path*) –

##### Return type

None

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message2**(tmp\_path)

##### Parameters

**tmp\_path** (*pathlib.Path*) –

##### Return type

None

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message4**(tmp\_path)

**Parameters****tmp\_path** (*pathlib.Path*) –**Return type**

None

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message5**(*tmp\_path*)**Parameters****tmp\_path** (*pathlib.Path*) –**Return type**

None

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message6**(*tmp\_path*)**Parameters****tmp\_path** (*pathlib.Path*) –**Return type**

None

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message7**(*tmp\_path*)**Parameters****tmp\_path** (*pathlib.Path*) –**Return type**

None

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message8**(*tmp\_path*)**Parameters****tmp\_path** (*pathlib.Path*) –**Return type**

None

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message9**(*tmp\_path*)**Parameters****tmp\_path** (*pathlib.Path*) –**Return type**

None

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message10**(*tmp\_path*)**Parameters****tmp\_path** (*pathlib.Path*) –**Return type**

None

schema\_salad.tests.test\_codegen\_errors.**test\_error\_message11**(*tmp\_path*)**Parameters****tmp\_path** (*pathlib.Path*) –**Return type**

None

## Schema Salad

---

`schema_salad.tests.test_codegen_errors.test_error_message15(tmp_path)`

### Parameters

`tmp_path (pathlib.Path) –`

### Return type

None

`schema_salad.tests.test_codegen_errors.load_document_by_uri(tmp_path, path)`

### Parameters

- `tmp_path (pathlib.Path) –`
- `path (Union[str, pathlib.Path]) –`

### Return type

Any

`schema_salad.tests.test_codegen_errors.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_cpp_codegen`

Test C++ code generation.

## Module Contents

### Functions

<code>test_cwl_cpp_gen(tmp_path)</code>	End to end test of C++ generator using the CWL v1.0 schema.
<code>test_cwl_cpp_generations(tmp_path, filename)</code>	End to end test of C++ generator using small scenarios.
<code>test_cwl_cpp_generations_with_spdx(tmp_path)</code>	End to end test of C++ generator checking for SPDX headers
<code>cpp_codegen(file_uri, target[, spdx_copyright_text, ...])</code>	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.test_cwl_cpp_generations(tmp_path, filename)`

End to end test of C++ generator using small scenarios.

**Parameters**

- `tmp_path` (`pathlib.Path`) –
- `filename` (`str`) –

**Return type**

None

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

End to end test of C++ generator checking for SPDX headers

**Parameters**

- `tmp_path` (`pathlib.Path`) –

**Return type**

None

`schema_salad.tests.test_cpp_codegen.cpp_codegen(file_uri, target, spdx_copyright_text=None, spdx_license_identifier=None)`

Help using the C++ code generation function.

**Parameters**

- `file_uri` (`str`) –
- `target` (`pathlib.Path`) –
- `spdx_copyright_text` (`Optional[List[str]]`) –
- `spdx_license_identifier` (`Optional[str]`) –

**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

---

`cwl_v1_2_schema(tmp_path_factory)``load_cwl(cwl_v1_2_schema, src)``test_secondaryFiles(cwl_v1_2_schema)`

secondaryFiles

`test_outputBinding(cwl_v1_2_schema)`

secondaryFiles

`test_yaml_tab_error(cwl_v1_2_schema)`Tabs in the file.

---

### Attributes

```
test_dir_name  
SchemaType
```

---

```
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_oneline_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 makeRDF().
<code>test_yaml_datetime()</code>	Affirm that yaml_no_ts 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_oneline_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)
```

Construct a full (“absolute”) URL by combining a “base URL” with another URL.

**Parameters**

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

**Return type**

*str*

```
class schema_salad.tests.test_fetch.CWLTestFetcher(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*

## Schema Salad

---

`urljoin(base, url)`

Construct a full (“absolute”) URL by combining a “base URL” with another 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>	Plan 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(met</code>	Hanging indents in Markdown lists don't lead to wordsmushing.
<code>test_multiline_list_entries_without_indentation()</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)
```

Plain 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_indentation(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://mypy.readthedocs.io/en/latest/differences\\_from\\_python.html#pickling-and-copying-objects](https://mypy.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_specializeEnums(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_specializeEnums(tmp_path)`

#### Parameters

`tmp_path (pathlib.Path) –`

#### Return type

None

---

```
schema_salad.tests.test_print_oneline
```

## Module Contents

### Functions

---

```
test_print_oneline()  
test_print_oneline_for_invalid_yaml()  
test_print_oneline_for_errors_in_the_same_line()  
test_print_oneline_for_errors_in_resolve_ref()  
test_for_invalid_yaml1()  
test_for_invalid_yaml2()
```

---

```
schema_salad.tests.test_print_oneline.test_print_oneline()
```

#### Return type

None

```
schema_salad.tests.test_print_oneline.test_print_oneline_for_invalid_yaml()
```

#### Return type

None

```
schema_salad.tests.test_print_oneline.test_print_oneline_for_errors_in_the_same_line()
```

#### Return type

None

```
schema_salad.tests.test_print_oneline.test_print_oneline_for_errors_in_resolve_ref()
```

#### Return type

None

```
schema_salad.tests.test_print_oneline.test_for_invalid_yaml1()
```

#### Return type

None

```
schema_salad.tests.test_print_oneline.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_nor_HOME_env_var(tmp_dir_fixture)</code>	
<code>test_Loader_initialisation_disable_doc_cache()</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.

### Parameters

`caplog (Any) –`

### Return type

None

`schema_salad.tests.test_schema`

## Module Contents

### Functions

---

`test_extend_and_specializeEnums(tmp_path)`

---

### Attributes

---

`cwl_file_uri`

---

`schema_salad.tests.test_schema.cwl_file_uri`

`schema_salad.tests.test_schema.test_extend_and_specializeEnums(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

<code>test_subtypes(old, new, result)</code>	Test is_subtype() function.
<code>test_avro_loading_subtype()</code>	Confirm conversion of SALAD style names to avro when overriding.
<code>test_avro_loading_subtype_bad()</code>	Confirm subtype error when overriding incorrectly.
<code>test_subtypes_nested()</code>	Confirm correct subtype handling on a nested type definition.
<code>test_subtypes_nested_bad()</code>	Confirm subtype error when overriding incorrectly in nested types.
<code>test_subtypes_recursive()</code>	Confirm correct subtype handling on a recursive type definition.
<code>test_subtypes_union()</code>	Confirm correct subtype handling on an union type definition.
<code>test_subtypes_union_bad()</code>	Confirm subtype error when overriding incorrectly in array types.

### 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.PropTypes) –`
- `new (schema_salad.avro.schema.PropTypes) –`
- `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_subtypes.test_subtypes_nested()`

Confirm correct subtype handling on a nested type definition.

**Return type**

None

`schema_salad.tests.test_subtypes.test_subtypes_nested_bad()`

Confirm subtype error when overriding incorrectly in nested types.

**Return type**

None

`schema_salad.tests.test_subtypes.test_subtypes_recursive()`

Confirm correct subtype handling on a recursive type definition.

**Return type**

None

`schema_salad.tests.test_subtypes.test_subtypes_union()`

Confirm correct subtype handling on an union type definition.

**Return type**

None

`schema_salad.tests.test_subtypes.test_subtypes_union_bad()`

Confirm subtype error when overriding incorrectly in array types.

**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

Shared test functions and attributes.

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

---

#### Attributes

---

<code>FIELD_SORT_ORDER</code>	
-------------------------------	--

---

`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, spdx_copyright_text=None, spdx_license_identifier=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]) –`
- `spdx_copyright_text (Optional[List[str]]) –`
- `spdx_license_identifier (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>LazyInitDef</code>	Lazy initialization logic.
<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.LazyInitDef(name, init)
```

Lazy initialization logic.

**Parameters**

- `name (str)` –
- `init (str)` –

```
__slots__ = ('name', 'init')
```

```
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\_lazy\_init(*lazy\_init*)**

Add lazy initialization logic for a given type.

**Parameters**

**lazy\_init** ([LazyInitDef](#)) –

**Return type**

None

**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, container=None, no\_link\_check=None)**

Parse the given type declaration and declare its components.

**Parameters**

- **type\_declaration** (*Union[List[Any], Dict[str, Any]]*) –
- **container** (*Optional[str]*) –
- **no\_link\_check** (*Optional[bool]*) –

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

**Return type**

None

**abstract uri\_loader(inner, scoped\_id, vocab\_term, ref\_scope, no\_link\_check=None)**

Construct the TypeDef for the given URI loader.

**Parameters**

- **inner** (`TypeDef`) –
- **scoped\_id** (`bool`) –
- **vocab\_term** (`bool`) –
- **ref\_scope** (`Optional[int]`) –
- **no\_link\_check** (`Optional[bool]`) –

**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>	Prototype of a class.
<i>FieldDefinition</i>	Prototype of a single field from a class definition.
<i>MapDefinition</i>	Prototype of a map.
<i>UnionDefinition</i>	Prototype of a union.
<i>EnumDefinition</i>	Prototype of a enum.
<i>CppCodeGen</i>	Generation of C++ code for a given Schema Salad definition.

### Functions

<i>q(s)</i>	Put quotes around a string.
<i>replaceKeywords(s)</i>	Rename keywords that are reserved in C++.
<i>safename(name)</i>	Create a C++ safe name.
<i>safename2(name)</i>	Create a namespaced safename.
<i>split_name(s)</i>	Split url name into its components.
<i>split_field(s)</i>	Split field into its components.
<i>isPrimitiveType(v)</i>	Check if v is a primitive type.
<i>hasFieldValue(e, f, v)</i>	Check if e has a field f value.
<i>isRecordSchema(v)</i>	Check if v is of type record schema.
<i>isEnumSchema(v)</i>	Check if v is of type enum schema.
<i>isArray(v)</i>	Check if v is of type array.
<i>pred(i)</i>	Check if v is any of the simple types.
<i>isArraySchema(v)</i>	Check if v is of type array schema.
<i>isMapSchema(v)</i>	Check if v is of type map schema.
<i>isUnionSchema(v)</i>	Check if v is of type union schema.

## Schema Salad

---

`schema_salad.cpp_codegen.q(s)`

Put quotes around a string.

**Parameters**

`s (str) –`

**Return type**

`str`

`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)`

Split url name into its components.

Splits names like `https://xyz.xyz/blub#cwl/class` into its class path and non class path

**Parameters**

`s (str) –`

**Return type**

`Tuple[str, str]`

`schema_salad.cpp_codegen.split_field(s)`

Split field into its components.

similar to `split_name` but for field names

**Parameters**

`s (str) –`

**Return type**

`Tuple[str, str, str]`

`class schema_salad.cpp_codegen.ClassDefinition(name)`

Prototype of a class.

**Parameters**

`name (str) –`

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

Write forward declaration.

**Parameters**

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

**Return type**

None

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

Write definition of the class.

**Parameters**

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

**Return type**

None

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

Write definition with implementation.

**Parameters**

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

**Return type**

None

**class schema\_salad.cpp\_codegen.FieldDefinition**(*name*, *typeStr*, *optional*, *remap*)

Prototype of a single field from a class definition.

**Parameters**

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

**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.MapDefinition(name, values)
```

Prototype of a map.

### Parameters

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

```
writeFwdDeclaration(target, fullInd, ind)
```

Write forward declaration.

### Parameters

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

### Return type

None

```
writeDefinition(target, ind)
```

Write map definition to output.

### Parameters

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

### Return type

None

```
writeImplDefinition(target, fullInd, ind)
```

Write definition with implementation.

### Parameters

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

### Return type

None

```
class schema_salad.cpp_codegen.UnionDefinition(name, types)
```

Prototype of a union.

### Parameters

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

```
writeFwdDeclaration(target, fullInd, ind)
```

Write forward declaration.

### Parameters

- **target** (*IO[str]*) –

- **fullInd** (*str*) –
- **ind** (*str*) –

**Return type**

None

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

Write union definition to output.

**Parameters**

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

**Return type**

None

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

Write definition with implementation.

**Parameters**

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

**Return type**

None

**class** schema\_salad.cpp\_codegen.**EnumDefinition**(*name, values*)

Prototype of a enum.

**Parameters**

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

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

Write enum definition to output.

**Parameters**

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

**Return type**

None

schema\_salad.cpp\_codegen.**isPrimitiveType**(*v*)

Check if v is a primitive type.

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

bool

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

Check if e has a field f value.

**Parameters**

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

**Return type**

`bool`

`schema_salad.cpp_codegen.isRecordSchema(v)`

Check if v is of type record schema.

**Parameters**

- **v** (*Any*) –

**Return type**

`bool`

`schema_salad.cpp_codegen.isEnumSchema(v)`

Check if v is of type enum schema.

**Parameters**

- **v** (*Any*) –

**Return type**

`bool`

`schema_salad.cpp_codegen.isArray(v)`

Check if v is of type array.

**Parameters**

- **v** (*Any*) –

**Return type**

`bool`

`schema_salad.cpp_codegen.pred(i)`

Check if v is any of the simple types.

**Parameters**

- **i** (*Any*) –

**Return type**

`bool`

`schema_salad.cpp_codegen.isArraySchema(v)`

Check if v is of type array schema.

**Parameters**

- **v** (*Any*) –

**Return type**

`bool`

`schema_salad.cpp_codegen.isMapSchema(v)`

Check if v is of type map schema.

**Parameters****v** (*Any*) –**Return type****bool****schema\_salad.cpp\_codegen.isUnionSchema(v)**

Check if v is of type union schema.

**Parameters****v** (*Any*) –**Return type****bool****class schema\_salad.cpp\_codegen.CppCodeGen(base, target, examples, package, copyright, spdx\_copyright\_text, spdx\_license\_identifier)**Bases: *schema\_salad.codegen\_base.CodeGenBase*

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

**Parameters**

- **base** (*str*) –
- **target** (*I0[str]*) –
- **examples** (*Optional[List[str]]*) –
- **package** (*str*) –
- **copyright** (*Optional[str]*) –
- **spdx\_copyright\_text** (*Optional[List[str]]*) –
- **spdx\_license\_identifier** (*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)**

Generate final part of our cpp file.

**Parameters****root\_loader** (*Optional[schema\_salad.codegen\_base.TypeDef]*) –**Return type****None****parseRecordField(field)**

Parse a record field.

**Parameters****field** (*Dict[str, Any]*) –**Return type***FieldDefinition*

**parseRecordSchema**(*stype*)

Parse a record schema.

**Parameters**

**stype** (*Dict*[*str*, *Any*]) –

**Return type**

None

**parseMapSchema**(*stype*)

Parse a map schema.

**Parameters**

**stype** (*Dict*[*str*, *Any*]) –

**Return type**

*str*

**parseUnionSchema**(*stype*)

Parse a union schema.

**Parameters**

**stype** (*Dict*[*str*, *Any*]) –

**Return type**

*str*

**parseEnum**(*stype*)

Parse a schema salad enum.

**Parameters**

**stype** (*Dict*[*str*, *Any*]) –

**Return type**

*str*

**parse**(*items*)

Parse schema salad items.

This function is being called from the outside and drives the whole code generation.

**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

<code>DlangCodeGen</code>	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

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

#### Functions

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

#### Attributes

<code>prims</code>
--------------------

`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, container=None, no\_link\_check=None)**

Parse the given type declaration and declare its components.

**Parameters**

- **type\_declaration** (*Union*[*List*[*Any*], *Dict*[*str*, *Any*], *str*]) –
- **container** (*Optional*[*str*]) –
- **no\_link\_check** (*Optional*[*bool*]) –

**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** (*Optional[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** (*Optional[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, no\_link\_check=None*)

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]*) –
- **no\_link\_check** (*Optional[bool]*) –

**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)**

Construct a full (“absolute”) URL by combining a “base URL” with another 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*)

Construct a full (“absolute”) URL by combining a “base URL” with another 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*, *container*=`None`, *no\_link\_check*=`None`)

Parse the given type declaration and declare its components.

**Parameters**

- **type\_declaration** (`Union[List[Any], Dict[str, Any], str]`) –
- **container** (`Optional[str]`) –
- **no\_link\_check** (`Optional[bool]`) –

**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** (`Optional[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** (`Optional[str]`) –
- **optional** (`bool`) –

**Return type**

None

**uri\_loader**(*inner*, *scoped\_id*, *vocab\_term*, *ref\_scope*, *no\_link\_check*=*None*)

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]`) –
- **no\_link\_check** (`Optional[bool]`) –

**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**

<b><i>pred</i></b> (datatype, field, name, context, defaultBase, ...)
---

<b><i>process_type</i></b> (t, g, context, defaultBase, namespaces, ...)
--

<b><i>salad_to_jsonld_context</i></b> (j, schema_ctx)
---

<b><i>fix_jsonld_ids</i></b> (obj, ids)	Add missing identity entries.
---	-------------------------------

<b><i>makerdf</i></b> (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, Optional[str]], 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 type**

*Tuple[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)`

<code>arg_parser()</code> <code>main([argsl])</code>	Build the argument parser.
---	----------------------------

`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

`MyRenderer`

Custom renderer with different representations of selected HTML tags.

`ToC`

`RenderType`

## Schema Salad

---

### Functions

<code>escape_html(s)</code>	Escape HTML but otherwise preserve single quotes.
<code>vocab_type_name(url)</code>	Remove the avro namespace, if any.
<code>has_types(items)</code>	
<code>linkto(item)</code>	
<code>patch_fenced_code(original_markdown_text, ...)</code>	Reverts fenced code fragments found in the modified contents back to their original definition.
<code>to_id(text)</code>	
<code>number_headings(toc, maindoc)</code>	
<code>fix_doc(doc)</code>	
<code>avrold_doc(j, outdoc, renderlist, redirects, brand, ...)</code>	
<code>arg_parser()</code>	Build the argument parser.
<code>main()</code>	Shortcut entrypoint.
<code>makedoc(stdout, schema[, redirects, only, brand, ...])</code>	Emit HTML representation of a given schema.

### Attributes

<code>PluginName</code>
<code>basicTypes</code>

`schema_salad.makedoc.PluginName`

`schema_salad.makedoc.escape_html(s)`

Escape HTML but otherwise preserve single quotes.

#### Parameters

`s (str)` –

#### Return type

`str`

`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.html.HTMLRenderer

Custom renderer with different representations of selected HTML tags.

**heading(text, level, \*\*attrs)**  
Override HTML heading creation with text IDs.

**Parameters**

- `text (str) –`
- `level (int) –`
- `attrs (Any) –`

**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.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]) –
    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, primtype,
                                 brandstyle=None, brandinverse=False)

Parameters
  • j (List[Dict[str, Any]]) –
  • outdoc (IO[Any]) –
  • renderlist (List[str]) –
  • redirects (Dict[str, str]) –
  • brand (str) –
  • brandlink (str) –
  • primtype (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, primtype=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]*) –
- **primtype** (*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 save() and fromDoc() function.
<i>Documented</i>	Mark classes than have a save() and fromDoc() function.
<i>RecordField</i>	A field of a record.
<i>RecordSchema</i>	Mark classes than have a save() and fromDoc() function.
<i>EnumSchema</i>	Define an enumerated type.
<i>ArraySchema</i>	Mark classes than have a save() and fromDoc() function.
<i>MapSchema</i>	Mark classes than have a save() and fromDoc() function.
<i>UnionSchema</i>	Mark classes than have a save() and fromDoc() function.
<i>JsonIdPredicate</i>	Attached to a record field to define how the parent record field is handled for
<i>SpecializeDef</i>	Mark classes than have a save() and fromDoc() function.
<i>NamedType</i>	Mark classes than have a save() and fromDoc() function.
<i>DocType</i>	Mark classes than have a save() and fromDoc() 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 save() and fromDoc() function.
<i>SaladEnumSchema</i>	Define an enumerated type.
<i>SaladMapSchema</i>	Define a map type.
<i>SaladUnionSchema</i>	Define a union type.
<i>Documentation</i>	A documentation section. This type exists to facilitate self-documenting

### Functions

<code>load_field(val, fieldtype, baseuri, loadingOptions[, lc])</code>	Load field.
<code>extract_type(val_type)</code>	Take a type of value, and extracts the value as a string.
<code>convert_typing(val_type)</code>	Normalize type names to schema-salad types.
<code>parse_errors(error_message)</code>	Parse error messages from several loaders into one error message.
<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>	Transform a file path into a URL with file scheme.
<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>

---

continues on next page

Table 2 – continued from previous page

<i>PrimitiveTypeLoader</i>	Names of salad data types (based on Avro schema declarations).
<i>AnyLoader</i>	The <b>Any</b> type validates for any non-null value.
<i>RecordFieldLoader</i>	
<i>RecordSchemaLoader</i>	
<i>EnumSchemaLoader</i>	
<i>ArraySchemaLoader</i>	
<i>MapSchemaLoader</i>	
<i>UnionSchemaLoader</i>	
<i>JsonIdPredicateLoader</i>	
<i>SpecializeDefLoader</i>	
<i>SaladRecordFieldLoader</i>	
<i>SaladRecordSchemaLoader</i>	
<i>SaladEnumSchemaLoader</i>	
<i>SaladMapSchemaLoader</i>	
<i>SaladUnionSchemaLoader</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_None</i>	
<i>union_of_PrimitiveTypeLoader_or_RecordSchema</i>	<i>der_or_MapSchema</i>
<i>array_of_union_of_PrimitiveTypeLoader_or_Re</i>	<i>SchemaLoader_o</i>
<i>union_of_PrimitiveTypeLoader_or_RecordSchem</i>	<i>der_or_MapSche</i>
<i>typedsl_union_of_PrimitiveTypeLoader_or_Re</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>	

continues on next page

# Schema Salad

Table 2 – continued from previous page

```
Record_nameLoader
typedsl_Record_nameLoader_2
union_of_None_type_or_strtype
uri_union_of_None_type_or_strtype_True_False
uri_array_of_strtype_True_False_None_None
Enum_nameLoader
typedsl_Enum_nameLoader_2
uri_union_of_PrimitiveTypeLoader_or_RecordSaLoader_or_Map
Array_nameLoader
typedsl_Array_nameLoader_2
Map_nameLoader
typedsl_Map_nameLoader_2
Union_nameLoader
typedsl_Union_nameLoader_2
union_of_None_type_or_booltype
union_of_None_type_or_inttype
uri_strtype_False_False_1_None
uri_union_of_None_type_or_strtype_False_Fals
uri_union_of_None_type_or_strtype_or_array_c
union_of_None_type_or_strtype_or_JsonIdPred;
union_of_None_type_or_Any_type
array_of_SaladRecordFieldLoader
union_of_None_type_or_array_of_SaladRecordF
idmap_fields_union_of_None_type_or_array_of_
uri_union_of_None_type_or_strtype_or_array_c
array_of_SpecializeDefLoader
```

continues on next page

Table 2 – continued from previous page

<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>adUnionSchemaL</code>
<code>array_of_union_of_SaladRecordSchemaLoader_o1</code>	<code>er_or_SaladUni</code>
<code>union_of_SaladRecordSchemaLoader_or_SaladEn</code>	<code>adUnionSchemaL</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, no_link_check=None,
                                              container=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]]`) –
- `no_link_check` (`Optional[bool]`) –
- `container` (`Optional[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]
no_link_check: bool | None
container: str | None

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, lc=None)`

Load field.

**Parameters**

- **val** (`Union[str, Dict[str, str]]`) –
- **fieldtype** (`_Loader`) –
- **baseuri** (`str`) –
- **loadingOptions** (`LoadingOptions`) –
- **lc** (`Optional[List[Any]]`) –

**Return type**

`Any`

`schema_salad.metaschema.save_type`

`schema_salad.metaschema.extract_type(val_type)`

Take a type of value, and extracts the value as a string.

**Parameters**

`val_type (Type[Any])` –

**Return type**

`str`

`schema_salad.metaschema.convert_typing(val_type)`

Normalize type names to schema-salad types.

**Parameters**

`val_type (str)` –

**Return type**

`str`

`schema_salad.metaschema.parse_errors(error_message)`

Parse error messages from several loaders into one error message.

**Parameters**

`error_message (str)` –

**Return type**

`Tuple[str, str, str]`

`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)
```

Transform a file path into a URL with file scheme.

### 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](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 than 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 than 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.MapSchema(type_, values, extension_fields=None, loadingOptions=None)`**

Bases: *Saveable*

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

**Parameters**

- **type\_** (*Any*) –
- **values** (*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

`MapSchema`

### `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.UnionSchema(names, type_, extension_fields=None, loadingOptions=None)`

Bases: `Saveable`

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

#### Parameters

- `names` (`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**

- *UnionSchema*

**`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 than 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*]) –
- **jsonIdPredicate** (*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.SaladMapSchema(name, type_, values, inVocab=None, doc=None,
                                              docParent=None, docChild=None, docAfter=None,
                                              jsonldPredicate=None, documentRoot=None,
                                              extension_fields=None, loadingOptions=None)
```

Bases: *NamedType*, *MapSchema*, *SchemaDefinedType*

Define a map type.

### Parameters

- **name** (Any) –
- **type\_** (Any) –
- **values** (Any) –
- **inVocab** (Optional[Any]) –
- **doc** (Optional[Any]) –
- **docParent** (Optional[Any]) –
- **docChild** (Optional[Any]) –
- **docAfter** (Optional[Any]) –
- **jsonldPredicate** (Optional[Any]) –
- **documentRoot** (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***SaladMapSchema***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.SaladUnionSchema(name, names, type_, inVocab=None, doc=None,
                                                docParent=None, docChild=None, docAfter=None,
                                                documentRoot=None, extension_fields=None,
                                                loadingOptions=None)
```

Bases: *NamedType*, *UnionSchema*, *DocType*

Define a union type.

**Parameters**

- **name** (*Any*) –
- **names** (*Any*) –
- **type\_** (*Any*) –
- **inVocab** (*Optional[Any]*) –
- **doc** (*Optional[Any]*) –
- **docParent** (*Optional[Any]*) –
- **docChild** (*Optional[Any]*) –
- **docAfter** (*Optional[Any]*) –
- **documentRoot** (*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

`SaladUnionSchema`

### `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.strtype``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.MapSchemaLoader

### schema\_salad.metaschema.UnionSchemaLoader

### schema\_salad.metaschema.JsonldPredicateLoader

### schema\_salad.metaschema.SpecializeDefLoader

### schema\_salad.metaschema.SaladRecordFieldLoader

### schema\_salad.metaschema.SaladRecordSchemaLoader

### schema\_salad.metaschema.SaladEnumSchemaLoader

### schema\_salad.metaschema.SaladMapSchemaLoader

### schema\_salad.metaschema.SaladUnionSchemaLoader

### 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\_Non

### schema\_salad.metaschema.

### union\_of\_PrimitiveTypeLoader\_or\_RecordSchemaLoader\_or\_EnumSchemaLoader\_or\_ArraySchemaLoader\_or\_MapSchema

### schema\_salad.metaschema.

### array\_of\_union\_of\_PrimitiveTypeLoader\_or\_RecordSchemaLoader\_or\_EnumSchemaLoader\_or\_ArraySchemaLoader\_or\_M

### schema\_salad.metaschema.

### union\_of\_PrimitiveTypeLoader\_or\_RecordSchemaLoader\_or\_EnumSchemaLoader\_or\_ArraySchemaLoader\_or\_MapSchema

### 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_None
schema_salad.metaschema.uri_array_of_strtype_True_False_None_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_MapSchemaLoader
schema_salad.metaschema.Array_nameLoader
schema_salad.metaschema.typedsl_Array_nameLoader_2
schema_salad.metaschema.Map_nameLoader
schema_salad.metaschema.typedsl_Map_nameLoader_2
schema_salad.metaschema.Union_nameLoader
schema_salad.metaschema.typedsl_Union_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_None
schema_salad.metaschema.uri_union_of_None_type_or_strtype_False_False_None_None
schema_salad.metaschema.
uri_union_of_None_type_or_strtype_or_array_of_strtype_False_False_None_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_None
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_SaladMapSchemaLoader_or_SaladUnionSchemaLo
schema_salad.metaschema.
array_of_union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader_or_SaladMapSchemaLoader_or_SaladUnion
schema_salad.metaschema.
union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader_or_SaladMapSchemaLoader_or_SaladUnionSchemaLo
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_saladcodegen_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*, *container*=None, *no\_link\_check*=None)

Parse the given type declaration and declare its components.

**Parameters**

- **type\_declaration** (*Union[List[Any], Dict[str, Any], str]*) –
- **container** (*Optional[str]*) –
- **no\_link\_check** (*Optional[bool]*) –

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

**Return type**

None

**uri\_loader**(*inner*, *scoped\_id*, *vocab\_term*, *ref\_scope*, *no\_link\_check*=None)

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]*) –
- **no\_link\_check** (*Optional[bool]*) –

**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[, lc])</code>	Load field.
<code>extract_type(val_type)</code>	Take a type of value, and extracts the value as a string.
<code>convert_typing(val_type)</code>	Normalize type names to schema-salad types.
<code>parse_errors(error_message)</code>	Parse error messages from several loaders into one error message.
<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>	Transform a file path into a URL with file scheme.
<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

`IdxType`

`save_type`

`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, no_link_check=None,
                                                       container=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]]*) –
- **no\_link\_check** (*Optional[bool]*) –
- **container** (*Optional[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]`

**no\_link\_check:** `bool | None`

**container:** `str | None`

**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, lc=None)`

Load field.

**Parameters**

- **val** (*Union[str, Dict[str, str]]*) –
- **fieldtype** (*\_Loader*) –
- **baseuri** (*str*) –
- **loadingOptions** (*LoadingOptions*) –
- **lc** (*Optional[List[Any]]*) –

**Return type**

*Any*

`schema_salad.python_codegen_support.save_type`

`schema_salad.python_codegen_support.extract_type(val_type)`

Take a type of value, and extracts the value as a string.

**Parameters**

**val\_type** (*Type[Any]*) –

**Return type**

*str*

`schema_salad.python_codegen_support.convert_typing(val_type)`

Normalize type names to schema-salad types.

**Parameters**

**val\_type** (*str*) –

**Return type**

*str*

`schema_salad.python_codegen_support.parse_errors(error_message)`

Parse error messages from several loaders into one error message.

**Parameters**

**error\_message** (*str*) –

**Return type**

*Tuple[str, str, str]*

`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)`

Transform a file path into a URL with file scheme.

### 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](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

```
file_uri(path[, split_frag])
```

```
uri_file_path(url)
```

```
to_validation_exception(e)
```

Convert ruamel.yaml exception to our type.

```
SubLoader(loader)
```

---

```
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

*schem\_salad.exceptions.ValidationException*

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

Bases: Dict[*str*, Union[ruamel.yaml.comments.CommentedMap, ruamel.yaml.comments.CommentedSeq, *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) –
        Any

Return type
        Any

__setitem__(key, value)
    Set self[key] to value.

Parameters
    • key (Any) –
    • value (Any) –
        Any

Return type
        Any

__delitem__(key)
    Delete self[key].

Parameters
    key (Any) –
        Any

Return type
        Any

__contains__(key)
    True if the dictionary has the specified key, else False.

Parameters
    key (Any) –
        Any

Return type
        bool

__del__()

Return type
        None

schema_salad.ref_resolver.SubLoader(loader)

Parameters
    loader (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.CommentedSeq, ruamel.yaml.comments.CommentedMap]*) –
- **docid** (*str*) –
- **all\_doc\_ids** (*Dict[str, str]*) –

**Return type**

Union[str, ruamel.yaml.comments.CommentedSeq, 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 Graphviz 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_salad.schema.get_metaschema()`

Instantiate the metaschema.

**Return type**

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

`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[ruamel.yaml.comments.CommentedMap, 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 Graphviz 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

<code>relname(source)</code>	
<code>add_lc_filename(r, source)</code>	
<code>reflow_all(text[, maxlen])</code>	
<code>reflow(text, maxlen[, shift])</code>	
<code>indent(v[, nolead, shift, bullet])</code>	
<code>bullets(textlist, bul)</code>	
<code>strip_duplicated_lineno(text)</code>	Strip duplicated line numbers.
<code>strip_dup_lineno(text[, maxlen])</code>	
<code>cmap(d[, lc, fn])</code>	

---

### Attributes

<code>lineno_re</code>
------------------------

`schema_salad.sourceline.lineno_re`

`schema_salad.sourceline.relname(source)`

**Parameters**

`source (str) –`

**Return type**

`str`

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

**Parameters**

- `r (ruamel.yaml.comments.CommentedBase) –`
- `source (str) –`

**Return type**

`None`

`schema_salad.sourceline.reflow_all(text, maxlen=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

---

`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.CommentedMap, ruamel.yaml.comments.CommentedSeq, 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

---

`TypeScriptCodeGen`

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.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, container=None, no\_link\_check=None)**

Parse the given type declaration and declare its components.

#### Parameters

- **type\_declaration** (*Union*[*List*[*Any*], *Dict*[*str*, *Any*], *str*]) –
- **container** (*Optional*[*str*]) –

- **no\_link\_check** (*Optional[bool]*) –
- 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** (*Optional[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** (*Optional[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, no\_link\_check=None*)
- 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]*) –
- **no\_link\_check** (*Optional[bool]*) –

**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``schema_salad.validate`

## Module Contents

### Functions

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

Produce an avro friendly short name.

`avro_type_name(url)`

Turn a URL into an Avro-safe name.

`friendly(v)`

Format an Avro schema into a pretty-printed representation.

`vpformat(datum)`

Truncate a pretty-printed representation of a Python object to 160 characters.

`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_shortname(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)
```

Format an Avro schema into a pretty-printed representation.

**Parameters**  
`v (Any) –`

**Return type**  
Any

```
schema_salad.validate.vpformat(datum)
```

Truncate a pretty-printed representation of a Python object to 160 characters.

**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\_\_, 64  
schema\_salad.avro, 17  
schema\_salad.avro.schema, 17  
schema\_salad.codegen, 64  
schema\_salad\_codegen\_base, 65  
schema\_salad\_cpp\_codegen, 69  
schema\_salad\_dlang\_codegen, 76  
schema\_salad\_dotnet\_codegen, 79  
schema\_salad.exceptions, 82  
schema\_salad.fetcher, 85  
schema\_salad\_java\_codegen, 87  
schema\_salad\_jsonld\_context, 91  
schema\_salad\_main, 92  
schema\_salad\_makedoc, 93  
schema\_salad\_metaschema, 98  
schema\_salad\_python\_codegen, 127  
schema\_salad\_python\_codegen\_support, 130  
schema\_salad\_ref\_resolver, 135  
schema\_salad\_schema, 140  
schema\_salad\_sourceline, 145  
schema\_salad\_tests, 25  
schema\_salad\_tests\_conftest, 25  
schema\_salad\_tests\_matcher, 26  
schema\_salad\_tests\_test\_avro\_names, 27  
schema\_salad\_tests\_test\_cg, 27  
schema\_salad\_tests\_test\_cli\_args, 29  
schema\_salad\_tests\_test\_codegen\_errors, 30  
schema\_salad\_tests\_test\_cpp\_codegen, 32  
schema\_salad\_tests\_test\_cwl11, 33  
schema\_salad\_tests\_test\_dlang\_codegen, 35  
schema\_salad\_tests\_test\_dotnet\_codegen, 35  
schema\_salad\_tests\_test\_errors, 37  
schema\_salad\_tests\_test\_examples, 40  
schema\_salad\_tests\_test\_fetch, 44  
schema\_salad\_tests\_test\_fp, 46  
schema\_salad\_tests\_test\_java\_codegen, 46  
schema\_salad\_tests\_test\_makedoc, 47  
schema\_salad\_tests\_test\_misc, 50  
schema\_salad\_tests\_test\_pickling, 50  
schema\_salad\_tests\_test\_print\_oneline, 51

schema\_salad.tests.test\_python\_codegen, 52  
schema\_salad.tests.test\_real\_cwl, 54  
schema\_salad.tests.test\_ref\_resolver, 55  
schema\_salad.tests.test\_schema, 58  
schema\_salad.tests.test\_schemas\_directive, 58  
schema\_salad.tests.test\_subtypes, 59  
schema\_salad.tests.test\_typescript\_codegen, 61  
schema\_salad.tests.util, 63  
schema\_salad\_typescript\_codegen, 149  
schema\_salad\_utils, 153  
schema\_salad\_validate, 153



# INDEX

## Symbols

`__author__` (in module `schema_salad`), 155  
`__contains__()` (`schema_salad.ref_resolver.NormDict` method), 137  
`__del__()` (`schema_salad.ref_resolver.NormDict` method), 137  
`__delitem__()` (`schema_salad.ref_resolver.NormDict` method), 137  
`__enter__()` (`schema_salad.sourceline.SourceLine` method), 148  
`__eq__()` (`schema_salad.metaschema.ArraySchema` method), 110  
`__eq__()` (`schema_salad.metaschema.Documentation` method), 123  
`__eq__()` (`schema_salad.metaschema.EnumSchema` method), 110  
`__eq__()` (`schema_salad.metaschema.JsonldPredicate` method), 114  
`__eq__()` (`schema_salad.metaschema.MapSchema` method), 111  
`__eq__()` (`schema_salad.metaschema.RecordField` method), 107  
`__eq__()` (`schema_salad.metaschema.RecordSchema` method), 108  
`__eq__()` (`schema_salad.metaschema.SaladEnumSchema` method), 119  
`__eq__()` (`schema_salad.metaschema.SaladMapSchema` method), 120  
`__eq__()` (`schema_salad.metaschema.SaladRecordField` method), 116  
`__eq__()` (`schema_salad.metaschema.SaladRecordSchema` method), 118  
`__eq__()` (`schema_salad.metaschema.SaladUnionSchema` method), 121  
`__eq__()` (`schema_salad.metaschema.SpecializeDef` method), 115  
`__eq__()` (`schema_salad.metaschema.UnionSchema` method), 112  
`__eq__()` (`schema_salad.ref_resolver.NormDict` method), 136  
`__eq__()` (`schema_salad.tests.matcher.JsonDiffMatcher` method), 26  
`__exit__()` (`schema_salad.sourceline.SourceLine` method), 148  
`__getitem__()` (`schema_salad.ref_resolver.NormDict` method), 136  
`__hash__()` (`schema_salad.metaschema.ArraySchema` method), 111  
`__hash__()` (`schema_salad.metaschema.Documentation` method), 123  
`__hash__()` (`schema_salad.metaschema.EnumSchema` method), 110  
`__hash__()` (`schema_salad.metaschema.JsonldPredicate` method), 114  
`__hash__()` (`schema_salad.metaschema.MapSchema` method), 112  
`__hash__()` (`schema_salad.metaschema.RecordField` method), 108  
`__hash__()` (`schema_salad.metaschema.RecordSchema` method), 109  
`__hash__()` (`schema_salad.metaschema.SaladEnumSchema` method), 119  
`__hash__()` (`schema_salad.metaschema.SaladMapSchema` method), 120  
`__hash__()` (`schema_salad.metaschema.SaladRecordField` method), 116  
`__hash__()` (`schema_salad.metaschema.SaladRecordSchema` method), 118  
`__hash__()` (`schema_salad.metaschema.SaladUnionSchema` method), 121  
`__hash__()` (`schema_salad.metaschema.SpecializeDef` method), 115  
`__hash__()` (`schema_salad.metaschema.UnionSchema` method), 113  
`__setitem__()` (`schema_salad.ref_resolver.NormDict` method), 137  
`__slots__` (`schema_salad_codegen_base.LazyInitDefAttribute`), 65  
`__slots__` (`schema_salad_codegen_base.TypeDefAttribute`), 65  
`__str__()` (`schema_salad.exceptions.SchemaSaladException` method), 84  
--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, 15
--codegen-package
    schema-salad-tool command line option, 15
--codegen-parser-info
    schema-salad-tool command line option, 15
--codegen-spdx-copyright-text
    schema-salad-tool command line option, 15
--codegen-spdx-license-identifier
    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
--primtype
    schema-salad-doc command line option, 16
    schema-salad-tool command line option, 16
--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, 16
-h
    schema-salad-doc command line option, 16
    schema-salad-tool command line option, 14
-v
    schema-salad-tool command line option, 16
```

**A**

```
add_context() (schema_salad.ref_resolver.Loader
    method), 138
add_entry() (schema_salad.makedoc.ToC method), 96
add_lazy_init() (schema_salad.codegen_base.CodeGenBase
    method), 66
add_lc_filename() (in module schema_salad.sourceline), 146
add_name() (schema_salad.avro.schema.Names
    method), 20
add_namespaces() (in module schema_salad.schema), 142
add_namespaces() (schema_salad.ref_resolver.Loader
    method), 138
add_schemas() (schema_salad.ref_resolver.Loader
    method), 138
add_vocab() (schema_salad.codegen_base.CodeGenBase
    method), 66
addl_metadata(schema_salad.metatschema>LoadingOptions
    attribute), 104
addl_metadata(schema_salad.python_codegen_support>LoadingOptions
    attribute), 132
Any_type (in module schema_salad.metatschema), 123
```

**A**

- AnyLoader (in module schema\_salad.metaschema), 124
- arg\_parser() (in module schema\_salad.main), 93
- arg\_parser() (in module schema\_salad.makedoc), 98
- Array\_nameLoader (in module schema\_salad.metaschema), 125
- array\_of\_RecordFieldLoader (in module schema\_salad.metaschema), 124
- array\_of\_SaladRecordFieldLoader (in module schema\_salad.metaschema), 125
- array\_of\_SpecializeDefLoader (in module schema\_salad.metaschema), 125
- array\_of\_strtype (in module schema\_salad.metaschema), 124
- array\_of\_union\_of\_PrimitiveTypeLoader\_or\_RecordSchemaLoader\_or\_ArraySchemaLoader\_or\_EnumSchemaLoader\_or\_ArraySchemaLoader\_or\_UnionSchemaLoader (in module schema\_salad.metaschema), 124
- array\_of\_union\_of\_SaladRecordSchemaLoader\_or\_UnionSchemaLoader (in module schema\_salad.metaschema), 126
- ArraySchema (class in schema\_salad.avro.schema), 22
- ArraySchema (class in schema\_salad.metaschema), 110
- ArraySchemaLoader (in module schema\_salad.metaschema), 124
- as\_warning() (schema\_salad.exceptions.SchemaSaladException method), 83
- AtomicPropType (in module schema\_salad.avro.schema), 18
- attrs (schema\_salad.metaschema.ArraySchema attribute), 110
- attrs (schema\_salad.metaschema.Documentation attribute), 122
- attrs (schema\_salad.metaschema.EnumSchema attribute), 109
- attrs (schema\_salad.metaschema.JsonIdPredicate attribute), 114
- attrs (schema\_salad.metaschema.MapSchema attribute), 111
- attrs (schema\_salad.metaschema.RecordField attribute), 107
- attrs (schema\_salad.metaschema.RecordSchema attribute), 108
- attrs (schema\_salad.metaschema.SaladEnumSchema attribute), 119
- attrs (schema\_salad.metaschema.SaladMapSchema attribute), 120
- attrs (schema\_salad.metaschema.SaladRecordField attribute), 116
- attrs (schema\_salad.metaschema.SaladRecordSchema attribute), 118
- attrs (schema\_salad.metaschema.SaladUnionSchema attribute), 121
- attrs (schema\_salad.metaschema.SpecializeDef attribute), 115
- attrs (schema\_salad.metaschema.UnionSchema attribute), 112
- Avro (in module schema\_salad.schema), 143
- avro\_field\_name() (in module schema\_salad.schema), 143
- avro\_shortname() (in module schema\_salad.validate), 154
- avro\_type\_name() (in module schema\_salad.validate), 154
- AvroException, 19
- avrold\_doc() (in module schema\_salad.makedoc), 97
- avsc\_names (schema\_salad.tests.test\_real\_cwl.TestRealWorldCWL attribute), 54
- avsc\_names (schema\_salad.tests.test\_schemas\_directive.TestSchemasDirective attribute), 59

**B**

- baseuri (schema\_salad.metaschema>LoadingOptions attribute), 103
- baseuri (schema\_salad.python\_codegen\_support>LoadingOptions attribute), 132
- BASIC\_JAVA\_IDENTIFIER\_REGEX (in module schema\_salad.java\_codegen), 87
- basicTypes (in module schema\_salad.makedoc), 96
- basket\_file\_uri (in module schema\_salad.tests.util), 63
- begin\_class() (schema\_salad.codegen\_base.CodeGenBase method), 66
- begin\_class() (schema\_salad.dotnet\_codegen.DotNetCodeGen method), 80
- begin\_class() (schema\_salad.java\_codegen.JavaCodeGen method), 88
- begin\_class() (schema\_salad.python\_codegen.PythonCodeGen method), 128
- begin\_class() (schema\_salad.typescript\_codegen.TypeScriptCodeGen method), 150
- black (in module schema\_salad.python\_codegen), 127
- block\_code() (schema\_salad.makedoc.MyRenderer method), 95
- block\_html() (schema\_salad.makedoc.MyRenderer method), 95
- booltype (in module schema\_salad.metaschema), 123
- bullets() (in module schema\_salad.sourceline), 147

**C**

- cache (schema\_salad.metaschema>LoadingOptions attribute), 104
- cache (schema\_salad.python\_codegen\_support>LoadingOptions attribute), 132
- cached\_metaschema (in module schema\_salad.schema), 141
- captured\_output() (in module schema\_salad.tests.test\_cli\_args), 29
- check\_exists() (schema\_salad.fetcher.DefaultFetcher method), 86
- check\_exists() (schema\_salad.fetcher.Fetcher method), 85

check\_exists() (*schema\_salad.tests.test\_fetch.CWLTestFetcher*).  
declare\_id\_field() (*schema\_salad.dotnet\_codegen.DotNetCodeGen*).  
method), 45  
method), 81  
check\_exists() (*schema\_salad.tests.test\_fetch.testFetcher*).  
declare\_id\_field() (*schema\_salad.java\_codegen.JavaCodeGen*).  
method), 45  
method), 89  
ClassDefinition (class in *schema\_salad.cpp\_codegen*), 70  
in declare\_id\_field() (*schema\_salad.python\_codegen.PythonCodeGen*).  
method), 129  
ClassValidationException, 84  
cmap() (in module *schema\_salad.sourceline*), 147  
codegen() (in module *schema\_salad.codegen*), 64  
CodeGenBase (class in *schema\_salad.codegen\_base*), 65  
collect\_namespaces() (in module *schema\_salad.schema*), 142  
144  
container (*schema\_salad.metaschema.LoadingOptions*).  
attribute), 104  
attribute), 104  
container (*schema\_salad.python\_codegen\_support.LoadingOptions*).  
attribute), 132  
DLangCodegen() (in module *schema\_salad.tests.test\_dlang\_codegen*),  
35  
contents() (*schema\_salad.makedoc*.ToC method), 96  
convert\_typing() (in module *schema\_salad.metaschema*), 105  
convert\_typing() (in module *schema\_salad.python\_codegen\_support*),  
133  
convertTypeToCppType() (*schema\_salad.cpp\_codegen.CppCodeGen*).  
method), 75  
CppMethodGen() (in module *schema\_salad.tests.test\_cpp\_codegen*), 33  
CppMethodGen (class in *schema\_salad.cpp\_codegen*), 75  
cwl\_file\_uri (in module *schema\_salad.tests.test\_dotnet\_codegen*),  
36  
cwl\_file\_uri (in module *schema\_salad.tests.test\_schema*), 58  
cwl\_file\_uri (in module *schema\_salad.tests.test\_typescript\_codegen*),  
62  
cwl\_file\_uri (in module *schema\_salad.tests.util*), 63  
cwl\_v1\_2\_schema() (in module *schema\_salad.tests.test\_cwl11*), 34  
CWLTestFetcher (class in *schema\_salad.tests.test\_fetch*), 45  
in declare\_id\_field() (*schema\_salad.codegen\_base.CodeGenBase*).  
method), 67  
DotNetCodeGen (class in *schema\_salad.dotnet\_codegen*), 79  
declare\_field() (*schema\_salad.dotnet\_codegen.DotNetCodeGen*).  
method), 81  
E  
declare\_field() (*schema\_salad.java\_codegen.JavaCodeGen*).  
method), 89  
declare\_field() (*schema\_salad.python\_codegen.PythonCodeGen*).  
method), 129  
declare\_field() (*schema\_salad.typescript\_codegen.TypeScriptCodeGen*).  
method), 151  
declare\_id\_field() (*schema\_salad.codegen\_base.CodeGenBase*).  
method), 67  
EndClass() (*schema\_salad.sourceline.SourceLine*).  
method), 148  
EndClassOrTypeClass() (*schema\_salad.codegen\_base.CodeGenBase*).  
method), 67  
EndTypeClassOrType() (*schema\_salad.dotnet\_codegen.DotNetCodeGen*).  
method), 80  
EndValueClass() (*schema\_salad.java\_codegen.JavaCodeGen*).  
method), 89

## D

declare\_field() (*schema\_salad.codegen\_base.CodeGenBase*).  
method), 67  
declare\_field() (*schema\_salad.dotnet\_codegen.DotNetCodeGen*).  
method), 81  
declare\_field() (*schema\_salad.java\_codegen.JavaCodeGen*).  
method), 89  
declare\_field() (*schema\_salad.python\_codegen.PythonCodeGen*).  
method), 129  
declare\_field() (*schema\_salad.typescript\_codegen.TypeScriptCodeGen*).  
method), 151  
declare\_id\_field() (*schema\_salad.codegen\_base.CodeGenBase*).  
method), 67  
EndClass() (*schema\_salad.sourceline.SourceLine*).  
method), 148  
EndClassOrTypeClass() (*schema\_salad.codegen\_base.CodeGenBase*).  
method), 67  
EndTypeClassOrType() (*schema\_salad.dotnet\_codegen.DotNetCodeGen*).  
method), 80  
EndValueClass() (*schema\_salad.java\_codegen.JavaCodeGen*).  
method), 89

`end_class()` (*schema\_salad.python\_codegen.PythonCodeFetcher* (*schema\_salad.metaschema.LoadingOptions* attribute), 128  
`end_class()` (*schema\_salad.typescript\_codegen.TypeScriptFetcher* (*schema\_salad.python\_codegen\_support.LoadingOptions* attribute), 150  
`Enum_nameLoader` (in module *schema\_salad.metaschema*), 125  
`EnumDefinition` (class in *schema\_salad.cpp\_codegen*), 73  
`EnumSchema` (class in *schema\_salad.avro.schema*), 22  
`EnumSchema` (class in *schema\_salad.metaschema*), 109  
`EnumSchemaLoader` (in module *schema\_salad.metaschema*), 124  
`epilogue()` (*schema\_salad.codegen\_base.CodeGenBase* method), 68  
`epilogue()` (*schema\_salad.cpp\_codegen.CppCodeGen* method), 75  
`epilogue()` (*schema\_salad.dlang\_codegen.DlangCodeGen* method), 77  
`epilogue()` (*schema\_salad.dotnet\_codegen.DotNetCodeGen* method), 82  
`epilogue()` (*schema\_salad.java\_codegen.JavaCodeGen* method), 90  
`epilogue()` (*schema\_salad.python\_codegen.PythonCodeFetcher* (*schema\_salad.python\_codegen\_support.LoadingOptions* attribute), 130  
`epilogue()` (*schema\_salad.typescript\_codegen.TypeScriptFetcher* (*schema\_salad.makedoc*), 96  
`escape_html()` (in module *schema\_salad.makedoc*), 94  
`expand_url()` (in module *schema\_salad.metaschema*), 106  
`expand_url()` (in module *schema\_salad.python\_codegen\_support*), 134  
`expand_url()` (*schema\_salad.ref\_resolver.Loader* method), 138  
`extend_and_specialize()` (in module *schema\_salad.schema*), 144  
`extract_type()` (in module *schema\_salad.metaschema*), 105  
`extract_type()` (in module *schema\_salad.python\_codegen\_support*), 133

**F**

`fetch()` (*schema\_salad.ref\_resolver.Loader* method), 139  
`fetch_text()` (*schema\_salad.fetcher.DefaultFetcher* method), 86  
`fetch_text()` (*schema\_salad.fetcher.Fetcher* method), 85  
`fetch_text()` (*schema\_salad.tests.test\_fetch.CWLTestFetcher* method), 45  
`fetch_text()` (*schema\_salad.tests.test\_fetch.testFetcher* method), 44  
`Fetcher` (class in *schema\_salad.fetcher*), 85  
`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*), 64  
`FieldDefinition` (class in *schema\_salad.cpp\_codegen*), 71  
`fields` (*schema\_salad.avro.schema.RecordSchema* property), 24  
`file()` (*schema\_salad.sourceline.SourceLine* method), 148  
`file_uri()` (in module *schema\_salad.metaschema*), 106  
`file_uri()` (in module *schema\_salad.python\_codegen\_support*), 134  
`file_uri()` (in module *schema\_salad.ref\_resolver*), 136  
`fileuri` (*schema\_salad.metaschema.LoadingOptions* attribute), 103  
`fix_jsonld_ids()` (in module *schema\_salad.jsonld\_context*), 92  
`fixture_metaschema_doc()` (in module *schema\_salad.tests.test\_makedoc*), 48  
`floattype` (in module *schema\_salad.metaschema*), 123  
`fmt()` (in module *schema\_salad.python\_codegen*), 127  
`friendly()` (in module *schema\_salad.validate*), 154  
`fromDoc()` (*schema\_salad.metaschema.ArraySchema* class method), 111  
`fromDoc()` (*schema\_salad.metaschema.Documentation* class method), 123  
`fromDoc()` (*schema\_salad.metaschema.EnumSchema* class method), 110  
`fromDoc()` (*schema\_salad.metaschema.JsonldPredicate* class method), 114  
`fromDoc()` (*schema\_salad.metaschema.MapSchema* class method), 112  
`fromDoc()` (*schema\_salad.metaschema.RecordField* class method), 108  
`fromDoc()` (*schema\_salad.metaschema.RecordSchema* class method), 109  
`fromDoc()` (*schema\_salad.metaschema.SaladEnumSchema* class method), 119  
`fromDoc()` (*schema\_salad.metaschema.SaladMapSchema* class method), 120  
`fromDoc()` (*schema\_salad.metaschema.SaladRecordField* class method), 116  
`fromDoc()` (*schema\_salad.metaschema.SaladRecordSchema* class method), 118

```
fromDoc() (schema_salad.metaschema.SaladUnionSchema
    class method), 122
fromDoc() (schema_salad.metaschema.Saveable class
    method), 104
fromDoc() (schema_salad.metaschema.SpecializeDef
    class method), 115
fromDoc() (schema_salad.metaschema.UnionSchema
    class method), 113
fromDoc() (schema_salad.python_codegen_support.Saveable
    class method), 132
fullname (schema_salad.avro.schema.Name property),
    20
idmap_fields_union_of_None_type_or_array_of_RecordFieldLoad
    (in module schema_salad.metaschema), 125
idmap_fields_union_of_None_type_or_array_of_SaladRecordFie
    (in module schema_salad.metaschema), 125
idmap_loader() (schema_salad.codegen_base.CodeGenBase
    method), 68
idmap_loader() (schema_salad.dotnet_codegen.DotNetCodeGen
    method), 82
idmap_loader() (schema_salad.java_codegen.JavaCodeGen
    method), 90
idmap_loader() (schema_salad.python_codegen.PythonCodeGen
    method), 129
idmap_loader() (schema_salad.typescript_codegen.TypeScriptCodeGen
    method), 152
idmap_specialize_union_of_None_type_or_array_of_Specialize
    (in module schema_salad.metaschema), 126
idx (schema_salad.metaschema>LoadingOptions attribute),
    103
idx (schema_salad.python_codegen_support>LoadingOptions attribute),
    132
IdxType (in module schema_salad.metaschema), 103
IdxType (in module schema_salad.python_codegen_support),
    131
imports (schema_salad.metaschema>LoadingOptions attribute),
    104
imports (schema_salad.python_codegen_support>LoadingOptions attribute),
    132
includes (schema_salad.metaschema>LoadingOptions attribute),
    104
includes (schema_salad.python_codegen_support>LoadingOptions attribute),
    132
indent() (in module schema_salad.sourceline), 147
inline_html() (schema_salad.makedoc.MyRenderer
    method), 95
INT_MAX_VALUE (in module schema_salad.validate), 154
INT_MIN_VALUE (in module schema_salad.validate), 154
interface_name() (schema_salad.java_codegen.JavaCodeGen
    method), 88
inttype (in module schema_salad.metaschema), 123
is_fs_case_sensitive() (in module
    schema_salad.tests.test_ref_resolver), 56
is_subtype() (in module schema_salad.avro.schema),
    25
isArray() (in module schema_salad.cpp_codegen), 74
isArraySchema() (in module
    schema_salad.cpp_codegen), 74
isEnumSchema() (in module
    schema_salad.cpp_codegen), 74
isMapSchema() (in module
    schema_salad.cpp_codegen), 74
isolated_cache() (in module
    schema_salad.tests.conftest), 25
isPrimitiveType() (in module
```

## H

```
has_name() (schema_salad.avro.schema.Names
    method), 20
has_types() (in module schema_salad.makedoc), 94
hasFieldValue() (in module
    schema_salad.cpp_codegen), 73
heading() (schema_salad.makedoc.MyRenderer
    method), 95
```

```
    schema_salad.cpp_codegen), 74
isPrimitiveType() (in module
```

<i>schema_salad.cpp_codegen), 73</i>	<i>module</i>	<i>LoadingOptions (class in schema_salad.metaschema), 103</i>
<i>isRecordSchema() (in schema_salad.cpp_codegen), 74</i>	<i>module</i>	<i>LoadingOptions (class in schema_salad.python_codegen_support), 131</i>
<i>isUnionSchema() (in schema_salad.cpp_codegen), 75</i>	<i>module</i>	<i>LONG_MAX_VALUE (in module schema_salad.validate), 154</i>
<i>items (schema_salad.avro.schema.ArraySchema property), 22</i>		<i>LONG_MIN_VALUE (in module schema_salad.validate), 154</i>
<b>J</b>		
<i>java_codegen() (in module schema_salad.tests.test_java_codegen), 47</i>		
<i>JavaCodeGen (class in schema_salad.java_codegen), 88</i>		
<i>JsonDataType (in module schema_salad.avro.schema), 18</i>		
<i>JsonDiffMatcher (class in schema_salad.tests.matcher), 26</i>		
<i>JsonIdPredicate (class in schema_salad.metaschema), 113</i>		
<i>JsonIdPredicateLoader (in module schema_salad.metaschema), 124</i>		
<b>L</b>		
<i>LazyInitDef (class in schema_salad.codegen_base), 65</i>		
<i>leaves() (schema_salad.exceptions.SchemaSaladException method), 83</i>		
<i>lineno_re (in module schema_salad.sourceline), 146</i>		
<i>linkto() (in module schema_salad.makedoc), 95</i>		
<i>load_and_validate() (in module schema_salad.schema), 142</i>		
<i>load_cwl() (in module schema_salad.tests.test_cwl11), 34</i>		
<i>load_cwl() (schema_salad.tests.test_real_cwl.TestRealWorldCWL method), 54</i>		
<i>load_cwl() (schema_salad.tests.test_schemas_directive.TestSchemasDirective method), 59</i>		
<i>load_document() (in module schema_salad.metaschema), 126</i>		
<i>load_document_by_string() (in module schema_salad.metaschema), 126</i>		
<i>load_document_by_uri() (in module schema_salad.tests.test_codegen_errors), 32</i>		
<i>load_document_by_yaml() (in module schema_salad.metaschema), 126</i>		
<i>load_document_with_metadata() (in module schema_salad.metaschema), 126</i>		
<i>load_field() (in module schema_salad.metaschema), 104</i>		
<i>load_field() (in module schema_salad.python_codegen_support), 133</i>		
<i>load_schema() (in module schema_salad.schema), 142</i>		
<i>Loader (class in schema_salad.ref_resolver), 137</i>		
		<b>M</b>
		<i>main() (in module schema_salad.main), 93</i>
		<i>main() (in module schema_salad.makedoc), 98</i>
		<i>make_avro() (in module schema_salad.schema), 144</i>
		<i>make_avro_schema() (in module schema_salad.schema), 144</i>
		<i>make_avro_schema_from_avro() (in module schema_salad.schema), 145</i>
		<i>make_avsc_object() (in module schema_salad.avro.schema), 25</i>
		<i>make_field_objects() (schema_salad.avro.schema.RecordSchema static method), 24</i>
		<i>make_valid_avro() (in module schema_salad.schema), 143</i>
		<i>makedoc() (in module schema_salad.makedoc), 98</i>
		<i>makeError() (schema_salad.sourceline.SourceLine method), 148</i>
		<i>makeLead() (schema_salad.sourceline.SourceLine method), 148</i>
		<i>makerdf() (in module schema_salad.jsonld_context), 92</i>
		<i>Map_nameLoader (in module schema_salad.metaschema), 125</i>
		<i>MapDefinition (class in schema_salad.cpp_codegen), 72</i>
		<i>MapSchema (class in schema_salad.avro.schema), 23</i>
		<i>MapSchema (class in schema_salad.metaschema), 111</i>
		<i>MapSchemaLoader (in module schema_salad.metaschema), 124</i>
		<i>maxDiff (in module schema_salad.tests.test_cg), 28</i>
		<i>MemoryCachingFetcher (class in schema_salad.fetcher), 86</i>
		<i>metaschema_file_uri (in module schema_salad.tests.test_dotnet_codegen), 36</i>
		<i>metaschema_file_uri (in module schema_salad.tests.test_typescript_codegen), 62</i>
		<i>metaschema_file_uri (in module schema_salad.tests.util), 63</i>
		<i>metaschema_loader (schema_salad.tests.test_real_cwl.TestRealWorldCWL attribute), 54</i>
		<i>metaschema_loader (schema_salad.tests.test_schemas_directive.TestSchemasDirective attribute), 59</i>

metaschema\_pre() (in module schema\_salad.tests.test\_cg), 28  
module  
    schema\_salad, 17  
    schema\_salad.\_\_main\_\_, 64  
    schema\_salad.avro, 17  
    schema\_salad.avro.schema, 17  
    schema\_salad.codegen, 64  
    schema\_salad.codegen\_base, 65  
    schema\_salad.cpp\_codegen, 69  
    schema\_salad.dlang\_codegen, 76  
    schema\_salad.dotnet\_codegen, 79  
    schema\_salad.exceptions, 82  
    schema\_salad.fetcher, 85  
    schema\_salad.java\_codegen, 87  
    schema\_salad.jsonld\_context, 91  
    schema\_salad.main, 92  
    schema\_salad.makedoc, 93  
    schema\_salad.metaschema, 98  
    schema\_salad.python\_codegen, 127  
    schema\_salad.python\_codegen\_support, 130  
    schema\_salad.ref\_resolver, 135  
    schema\_salad.schema, 140  
    schema\_salad.sourceline, 145  
    schema\_salad.tests, 25  
    schema\_salad.tests.conftest, 25  
    schema\_salad.tests.matcher, 26  
    schema\_salad.tests.test\_avro\_names, 27  
    schema\_salad.tests.test\_cg, 27  
    schema\_salad.tests.test\_cli\_args, 29  
    schema\_salad.tests.test\_codegen\_errors, 30  
    schema\_salad.tests.test\_cpp\_codegen, 32  
    schema\_salad.tests.test\_cwl11, 33  
    schema\_salad.tests.test\_dlang\_codegen, 35  
    schema\_salad.tests.test\_dotnet\_codegen, 35  
    schema\_salad.tests.test\_errors, 37  
    schema\_salad.tests.test\_examples, 40  
    schema\_salad.tests.test\_fetch, 44  
    schema\_salad.tests.test\_fp, 46  
    schema\_salad.tests.test\_java\_codegen, 46  
    schema\_salad.tests.test\_makedoc, 47  
    schema\_salad.tests.test\_misc, 50  
    schema\_salad.tests.test\_pickling, 50  
    schema\_salad.tests.test\_print\_oneline, 51  
    schema\_salad.tests.test\_python\_codegen, 52  
    schema\_salad.tests.test\_real\_cwl, 54  
    schema\_salad.tests.test\_ref\_resolver, 55  
    schema\_salad.tests.test\_schema, 58  
    schema\_salad.tests.test\_schemas\_directive, 58  
    schema\_salad.tests.test\_subtypes, 59

schema\_salad.tests.test\_typescript\_codegen, 61  
schema\_salad.tests.util, 63  
schema\_salad.typescript\_codegen, 149  
schema\_salad.utils, 153  
schema\_salad.validate, 153  
MyRenderer (class in schema\_salad.makedoc), 95

## N

Name (class in schema\_salad.avro.schema), 20  
name (schema\_salad.avro.schema.NamedSchema property), 21  
NAMED\_TYPES (in module schema\_salad.avro.schema), 18  
NamedMapSchema (class in schema\_salad.avro.schema), 23  
NamedSchema (class in schema\_salad.avro.schema), 21  
NamedType (class in schema\_salad.metaschema), 115  
NamedUnionSchema (class in schema\_salad.avro.schema), 24  
Names (class in schema\_salad.avro.schema), 20  
namespaces (schema\_salad.metaschema>LoadingOptions attribute), 104  
namespaces (schema\_salad.python\_codegen\_support>LoadingOptions attribute), 132  
no\_link\_check (schema\_salad.metaschema>LoadingOptions attribute), 104  
no\_link\_check (schema\_salad.python\_codegen\_support>LoadingOptions attribute), 132  
None\_type (in module schema\_salad.metaschema), 123  
NormDict (class in schema\_salad.ref\_resolver), 136  
number\_headings() (in module schema\_salad.makedoc), 96

## O

original\_doc (schema\_salad.metaschema>LoadingOptions attribute), 104  
original\_doc (schema\_salad.python\_codegen\_support>LoadingOptions attribute), 132

## P

parse() (schema\_salad.cpp\_codegen.CppCodeGen method), 76  
parse() (schema\_salad.dlang\_codegen.DlangCodeGen method), 78  
parse\_enum() (schema\_salad.dlang\_codegen.DlangCodeGen method), 78  
parse\_errors() (in module schema\_salad.metaschema), 105  
parse\_errors() (in module schema\_salad.python\_codegen\_support), 133

**parse\_record\_field()** (*schema\_salad.dlang\_codegen.DlangCodeGen method*), 78  
**parse\_record\_field\_type()** (*schema\_salad.dlang\_codegen.DlangCodeGen method*), 78  
**parse\_record\_schema()** (*schema\_salad.dlang\_codegen.DlangCodeGen method*), 78  
**parseEnum()** (*schema\_salad.cpp\_codegen.CppCodeGen method*), 76  
**parseMapSchema()** (*schema\_salad.cpp\_codegen.CppCodeGen method*), 76  
**parser\_info()** (in module *schema\_salad.metaschema*), 107  
**parseRecordField()** (*schema\_salad.cpp\_codegen.CppCodeGen method*), 75  
**parseRecordSchema()** (*schema\_salad.cpp\_codegen.CppCodeGen method*), 76  
**parseUnionSchema()** (*schema\_salad.cpp\_codegen.CppCodeGen method*), 76  
**patch\_fenced\_code()** (in module *schema\_salad.makedoc*), 96  
**PluginName** (in module *schema\_salad.makedoc*), 94  
**pred()** (in module *schema\_salad.cpp\_codegen*), 74  
**pred()** (in module *schema\_salad.jsonld\_context*), 91  
**prefix()** (*schema\_salad.exceptions.SchemaSaladException method*), 83  
**prefix\_url()** (in module *schema\_salad.metaschema*), 106  
**prefix\_url()** (in module *schema\_salad.python\_codegen\_support*), 134  
**pretty\_str()** (*schema\_salad.exceptions.SchemaSaladException method*), 84  
**PRIMITIVE\_TYPES** (in module *schema\_salad.avro.schema*), 18  
**primitives** (in module *schema\_salad.validate*), 154  
**PrimitiveSchema** (class in *schema\_salad.avro.schema*), 22  
**PrimitiveTypeLoader** (in module *schema\_salad.metaschema*), 123  
**prims** (in module *schema\_salad.dotnet\_codegen*), 79  
**prims** (in module *schema\_salad.java\_codegen*), 88  
**prims** (in module *schema\_salad.python\_codegen*), 127  
**prims** (in module *schema\_salad.typescript\_codegen*), 149  
**print\_fieldrefs()** (in module *schema\_salad.schema*), 145  
**print\_inheritance()** (in module *schema\_salad.schema*), 145  
**printrdf()** (in module *schema\_salad.main*), 93  
**process\_type()** (in module *schema\_salad.cpp\_codegen*), 91  
**schema\_salad.jsonld\_context**, 91  
**prologue()** (*schema\_salad\_codegen\_base.CodeGenBase method*), 66  
**prologue()** (*schema\_salad.dlang\_codegen.DlangCodeGen method*), 77  
**prologue()** (*schema\_salad.dotnet\_codegen.DotNetCodeGen method*), 79  
**prologue()** (*schema\_salad.java\_codegen.JavaCodeGen method*), 88  
**prologue()** (*schema\_salad.python\_codegen.PythonCodeGen method*), 128  
**prologue()** (*schema\_salad.typescript\_codegen.TypeScriptCodeGen method*), 150  
**propagate\_sourceline()** (*schema\_salad.exceptions.SchemaSaladException static method*), 88  
**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\_codegen\_errors*), 32  
**python\_codegen()** (in module *schema\_salad.tests.test\_python\_codegen*), 53  
**PythonCodeGen** (class in *schema\_salad.python\_codegen*), 127

**Q**

**q()** (in module *schema\_salad.cpp\_codegen*), 69

**R**

**Record\_nameLoader** (in module *schema\_salad.metaschema*), 125  
**RecordField** (class in *schema\_salad.metaschema*), 107  
**RecordFieldLoader** (in module *schema\_salad.metaschema*), 124  
**RecordSchema** (class in *schema\_salad.avro.schema*), 24  
**RecordSchema** (class in *schema\_salad.metaschema*), 108  
**RecordSchemaLoader** (in module *schema\_salad.metaschema*), 124  
**reflow()** (in module *schema\_salad.sourceline*), 147  
**reflow\_all()** (in module *schema\_salad.sourceline*), 146  
**relname()** (in module *schema\_salad.sourceline*), 146  
**render\_type()** (*schema\_salad.makedoc.RenderType method*), 97  
**RenderType** (class in *schema\_salad.makedoc*), 97  
**replace\_type()** (in module *schema\_salad.schema*), 143  
**replaceKeywords()** (in module *schema\_salad.cpp\_codegen*), 70

resolve\_all() (schema\_salad.ref\_resolver.Loader method), 139  
resolve\_ref() (schema\_salad.ref\_resolver.Loader method), 138  
rvocab (schema\_salad.metaschema>LoadingOptions attribute), 104  
rvocab (schema\_salad.python\_codegen\_support>LoadingOptions attribute), 132

S

safe\_name() (schema\_salad\_codegen\_base.CodeGenBase static method), 66  
safe\_name() (schema\_salad\_dlang\_codegen.DlangCodeGen static method), 77  
safe\_name() (schema\_salad\_dotnet\_codegen.DotNetCodeGen static method), 80  
safe\_name() (schema\_salad\_java\_codegen.JavaCodeGen static method), 88  
safe\_name() (schema\_salad\_python\_codegen.PythonCodeGen static method), 128  
safe\_name() (schema\_salad\_typescript\_codegen.TypeScriptCodeGen static method), 150  
safename() (in module schema\_salad.cpp\_codegen), 70  
safename2() (in module schema\_salad.cpp\_codegen), 70  
SALAD\_FILES (in module schema\_salad.schema), 141  
salad\_to\_jsonld\_context() (in module schema\_salad.jsonld\_context), 92  
SaladEnumSchema (class in schema\_salad.metaschema), 118  
SaladEnumSchemaLoader (in module schema\_salad.metaschema), 124  
SaladMapSchema (class in schema\_salad.metaschema), 120  
SaladMapSchemaLoader (in module schema\_salad.metaschema), 124  
saladp (in module schema\_salad.schema), 141  
saladp (in module schema\_salad.validate), 154  
SaladRecordField (class in schema\_salad.metaschema), 116  
SaladRecordFieldLoader (in module schema\_salad.metaschema), 124  
SaladRecordSchema (class in schema\_salad.metaschema), 117  
SaladRecordSchemaLoader (in module schema\_salad.metaschema), 124  
SaladUnionSchema (class in schema\_salad.metaschema), 121  
SaladUnionSchemaLoader (in module schema\_salad.metaschema), 124  
save() (in module schema\_salad.metaschema), 105  
save() (in module schema\_salad.python\_codegen\_support), 133

save() (schema\_salad.metaschema.ArraySchema method), 111  
save() (schema\_salad.metaschema.Documentation method), 123  
save() (schema\_salad.metaschema.EnumSchema method), 110  
save() (schema\_salad.metaschema.JsonldPredicate method), 114  
save() (schema\_salad.metaschema.MapSchema method), 112  
save() (schema\_salad.metaschema.RecordField method), 108  
save() (schema\_salad.metaschema.RecordSchema method), 109  
(schema\_salad.metaschema.SaladEnumSchema method), 119  
(schema\_salad.metaschema.SaladMapSchema method), 121  
(schema\_salad.metaschema.SaladRecordField method), 117  
save() (schema\_salad.metaschema.SaladRecordSchema method), 118  
save() (schema\_salad.metaschema.SaladUnionSchema method), 122  
save() (schema\_salad.metaschema.Saveable method), 104  
save() (schema\_salad.metaschema.SpecializeDef method), 115  
save() (schema\_salad.metaschema.UnionSchema method), 113  
save() (schema\_salad.python\_codegen\_support.Saveable method), 133  
save\_relative\_uri() (in module schema\_salad.metaschema), 106  
save\_relative\_uri() (in module schema\_salad.python\_codegen\_support), 135  
save\_type (in module schema\_salad.metaschema), 105  
save\_type (in module schema\_salad.python\_codegen\_support), 133  
save\_with\_metadata() (in module schema\_salad.metaschema), 105  
save\_with\_metadata() (in module schema\_salad.python\_codegen\_support), 134  
Saveable (class in schema\_salad.metaschema), 104  
Saveable (class in schema\_salad.python\_codegen\_support), 132  
schema schema-salad-doc command line option, 16  
schema schema-salad-tool command line option, 14  
Schema (class in schema\_salad.avro.schema), 19  
schema\_metadata (schema\_salad.tests.test\_real\_cwl.TestRealWorldCWL

```

    attribute), 54
schema_metadata(schema_salad.tests.test_schemas_directive
    attribute), 59
SCHEMA_RESERVED_PROPS (in
    schema_salad.avro.schema), 18
schema_salad
    module, 17
schema_salad.__main__
    module, 64
schema_salad.avro
    module, 17
schema_salad.avro.schema
    module, 17
schema_salad_codegen
    module, 64
schema_salad_codegen_base
    module, 65
schema_salad_cpp_codegen
    module, 69
schema_salad_dlang_codegen
    module, 76
schema_salad_dotnet_codegen
    module, 79
schema_salad_exceptions
    module, 82
schema_salad_fetcher
    module, 85
schema_salad_java_codegen
    module, 87
schema_salad_jsonld_context
    module, 91
schema_salad_main
    module, 92
schema_salad_makedoc
    module, 93
schema_salad_metaschema
    module, 98
schema_salad_python_codegen
    module, 127
schema_salad_python_codegen_support
    module, 130
schema_salad_ref_resolver
    module, 135
schema_salad_schema
    module, 140
schema_salad_sourceline
    module, 145
schema_salad_tests
    module, 25
schema_salad_tests_conftest
    module, 25
schema_salad_tests_matcher
    module, 26
schema_salad_tests_test_avro_names
    module, 27
schema_salad_tests_test_cg
    module, 27
schema_salad_tests_test_cli_args
    module, 29
schema_salad_tests_test_codegen_errors
    module, 30
schema_salad_tests_test_cpp_codegen
    module, 32
schema_salad_tests_test_cwl11
    module, 33
schema_salad_tests_test_dlang_codegen
    module, 35
schema_salad_tests_test_dotnet_codegen
    module, 35
schema_salad_tests_test_errors
    module, 37
schema_salad_tests_test_examples
    module, 40
schema_salad_tests_test_fetch
    module, 44
schema_salad_tests_test_fp
    module, 46
schema_salad_tests_test_java_codegen
    module, 46
schema_salad_tests_test_makedoc
    module, 47
schema_salad_tests_test_misc
    module, 50
schema_salad_tests_test_pickling
    module, 50
schema_salad_tests_test_print_oneline
    module, 51
schema_salad_tests_test_python_codegen
    module, 52
schema_salad_tests_test_real_cwl
    module, 54
schema_salad_tests_test_ref_resolver
    module, 55
schema_salad_tests_test_schema
    module, 58
schema_salad_tests_test_schemas_directive
    module, 58
schema_salad_tests_test_subtypes
    module, 59
schema_salad_tests_test_typescript_codegen
    module, 61
schema_salad_tests_util
    module, 63
schema_salad_typescript_codegen
    module, 149
schema_salad_utils
    module, 153
schema_salad_validate

```

```
    module, 153
schema_type (in module schema_salad.schema), 142
schema-salad-doc command line option
    --brand, 16
    --brandinverse, 16
    --brandlink, 16
    --brandstyle, 16
    --debug, 16
    --help, 16
    --only, 16
    --primtype, 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, 15
    --codegen-package, 15
    --codegen-parser-info, 15
    --codegen-spdx-copyright-text, 15
    --codegen-spdx-license-identifier, 15
    --codegen-target, 14
    --debug, 15
    --help, 14
    --non-strict, 15
    --only, 15
    --primtype, 16
    --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, 16
    -h, 14
    -v, 16
    document, 14
                                                schema, 14
SchemaDefinedType           (class      in
                                schema_salad.metaschema), 116
SchemaException, 84
SchemaParseException, 19
schemas (schema_salad.avro.schema.NamedUnionSchema
          property), 24
schemas (schema_salad.avro.schema.UnionSchema
          property), 24
schemas (schema_salad.metaschema>LoadingOptions at-
          tribute), 104
schemas (schema_salad.python_codegen_support>LoadingOptions
          attribute), 132
SchemaSaladException, 83
SchemaType (in module schema_salad.tests.test_cwl11),
            34
schemes (schema_salad.fetcher.Fetcher attribute), 85
secondaryfilesdsl_loader()
    (schema_salad_codegen_baseCodeGen
     method), 68
secondaryfilesdsl_loader()
    (schema_salad_dotnet_codegen.DotNetCodeGen
     method), 82
secondaryfilesdsl_loader()
    (schema_salad_java_codegen.JavaCodeGen
     method), 91
secondaryfilesdsl_loader()
    (schema_salad_python_codegen.PythonCodeGen
     method), 130
secondaryfilesdsl_loader()
    (schema_salad_typescript_codegen.TypeScriptCodeGen
     method), 152
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), 54
setup_class() (schema_salad.tests.test_schemas_directive.TestSchemasD
               class method), 59
shortname() (in module schema_salad.metaschema),
            107
shortname()           (in      module
                        schema_salad.python_codegen_support),
            135
shortname() (in module schema_salad.schema), 145
SourceLine (class in schema_salad.sourceline), 148
SpecializeDef (class in schema_salad.metaschema),
            114
SpecializeDefLoader       (in      module
                            schema_salad.metaschema), 124
split_field()           (in      module
                            schema_salad_cpp_codegen), 70
split_name() (in module schema_salad_cpp_codegen),
```

<b>T</b>	
test_attachments() (in module <code>schema_salad.tests.test_ref_resolver</code> ), 57	
test_avro_loading() (in module <code>schema_salad.tests.test_avro_names</code> ), 27	
test_avro_loading_subtype() (in module <code>schema_salad.tests.test_subtypes</code> ), 60	
test_avro_loading_subtype_bad() (in module <code>schema_salad.tests.test_subtypes</code> ), 60	
test_avro_regression() (in module <code>schema_salad.tests.test_examples</code> ), 42	
test_bad_schema() (in module <code>schema_salad.tests.test_errors</code> ), 38	
test_bad_schema2() (in module <code>schema_salad.tests.test_errors</code> ), 39	
test_bad_schemas() (in module <code>schema_salad.tests.test_examples</code> ), 41	
test_blank_node_id() (in module <code>schema_salad.tests.test_examples</code> ), 44	
test_cache() (in module <code>schema_salad.tests.test_fetch</code> ), 46	
test_can_use_Any() (in module <code>schema_salad.tests.test_examples</code> ), 44	
test_check_exists_follows_redirects() (in module <code>schema_salad.tests.test_ref_resolver</code> ), 57	
test_class_field() (in module <code>schema_salad.tests.test_dotnet_codegen</code> ), 36	
test_class_field() (in module <code>schema_salad.tests.test_typescript_codegen</code> ), 62	
test_cmap() (in module <code>schema_salad.tests.test_examples</code> ), 44	
test_cwl_cpp_gen() (in module <code>schema_salad.tests.test_cpp_codegen</code> ), 32	
test_cwl_cpp_generations() (in module <code>schema_salad.tests.test_cpp_codegen</code> ), 33	
test_cwl_cpp_generations_with_spdx() (in module <code>schema_salad.tests.test_cpp_codegen</code> ), 33	
test_cwl_dlang_gen() (in module <code>schema_salad.tests.test_dlang_codegen</code> ), 35	
test_cwl_gen() (in module <code>schema_salad.tests.test_dotnet_codegen</code> ), 36	
test_cwl_gen() (in module <code>schema_salad.tests.test_java_codegen</code> ), 46	
test_cwl_gen() (in module <code>schema_salad.tests.test_python_codegen</code> ), 52	
test_cwl_gen() (in module <code>schema_salad.tests.test_typescript_codegen</code> ), 62	
test_default_parser_info() (in module <code>schema_salad.tests.test_python_codegen</code> ), 53	
test_DefaultFetcher_urljoin_linux() (in module <code>schema_salad.tests.test_ref_resolver</code> ), 57	
test_DefaultFetcher_urljoin_win32() (in module <code>schema_salad.tests.test_ref_resolver</code> ), 57	
test_detect_changes_in_html() (in module <code>schema_salad.tests.test_makedoc</code> ), 49	
test_dir_name (in module <code>schema_salad.tests.test_cwl11</code> ), 34	
test_dir_name (in module <code>schema_salad.tests.test_real_cwl</code> ), 54	
test_dir_name (in module <code>schema_salad.tests.test_schemas_directive</code> ), 59	
test_doc_fenced_code_contents_preserved() (in module <code>schema_salad.tests.test_makedoc</code> ), 48	
test_doc_headings_target_anchor() (in module <code>schema_salad.tests.test_makedoc</code> ), 48	
test_doc_render_table_of_contents() (in module <code>schema_salad.tests.test_makedoc</code> ), 49	
test_dollarsign_schema() (schema_salad.tests.test_schemas_directive.TestSchemasDirective method), 59	
test_embedded_html_unescaped() (in module <code>schema_salad.tests.test_makedoc</code> ), 49	
test_empty_input() (in module <code>schema_salad.tests.test_cli_args</code> ), 29	
test_err() (in module <code>schema_salad.tests.test_cg</code> ), 28	
test_err2() (in module <code>schema_salad.tests.test_cg</code> ), 28	
test_error_message1() (in module <code>schema_salad.tests.test_codegen_errors</code> ), 30	
test_error_message1() (in module <code>schema_salad.tests.test_errors</code> ), 38	
test_error_message10() (in module <code>schema_salad.tests.test_errors</code> ), 38	

schema_salad.tests.test_codegen_errors),			schema_salad.tests.test_examples), 42
31			test_extend_and_specializeEnums() (in module schema_salad.tests.test_pickling), 50
test_error_message10() (in module schema_salad.tests.test_errors), 38			test_extend_and_specializeEnums() (in module schema_salad.tests.test_schema), 58
test_error_message11() (in module schema_salad.tests.test_codegen_errors),			test_fetch_inject_id() (in module schema_salad.tests.test_ref_resolver), 57
31			test_fetcher() (in module schema_salad.tests.test_fetch), 46
test_error_message11() (in module schema_salad.tests.test_errors), 38			test_file_uri() (in module schema_salad.tests.test_examples), 43
test_error_message15() (in module schema_salad.tests.test_codegen_errors),			test_for_invalid_yaml1() (in module schema_salad.tests.test_print_oneline), 51
31			test_for_invalid_yaml2() (in module schema_salad.tests.test_print_oneline), 51
test_error_message15() (in module schema_salad.tests.test_errors), 38			test_fp() (in module schema_salad.tests.test_fp), 46
test_error_message2() (in module schema_salad.tests.test_codegen_errors),			test_fragment() (in module schema_salad.tests.test_examples), 43
30			test_graph_property() (in module schema_salad.tests.test_python_codegen), 53
test_error_message2() (in module schema_salad.tests.test_errors), 38			test_graph_property_cache() (in module schema_salad.tests.test_python_codegen), 53
test_error_message3() (in module schema_salad.tests.test_errors), 38			test_graph_property_empty_schema() (in module schema_salad.tests.test_python_codegen), 54
test_error_message4() (in module schema_salad.tests.test_codegen_errors),			test_h3agatk_SNP() (schema_salad.tests.test_real_cwl.TestRealWorldCV method), 55
30			test_h3agatk_WES() (schema_salad.tests.test_real_cwl.TestRealWorldCV method), 55
test_error_message4() (in module schema_salad.tests.test_errors), 38			test_icgc_pancan() (schema_salad.tests.test_real_cwl.TestRealWorldCV method), 55
test_error_message5() (in module schema_salad.tests.test_codegen_errors),			test_idmap() (in module schema_salad.tests.test_cg), 28
31			test_idmap() (in module schema_salad.tests.test_examples), 42
test_error_message5() (in module schema_salad.tests.test_errors), 38			test_idmap2() (in module schema_salad.tests.test_cg), 28
test_error_message6() (in module schema_salad.tests.test_codegen_errors),			test_import() (in module schema_salad.tests.test_cg), 28
31			test_import2() (in module schema_salad.tests.test_cg), 28
test_error_message7() (in module schema_salad.tests.test_codegen_errors),			test_import_list() (in module schema_salad.tests.test_ref_resolver), 57
31			test_include() (in module schema_salad.tests.test_cg), 28
test_error_message7() (in module schema_salad.tests.test_errors), 38			test_jsonld_ctx() (in module schema_salad.tests.test_examples), 42
test_error_message8() (in module schema_salad.tests.test_codegen_errors),			test_load() (in module schema_salad.tests.test_cg), 28
31			test_load_by_yaml_metaschema() (in module schema_salad.tests.test_cg), 29
test_error_message8() (in module schema_salad.tests.test_errors), 38			test_load_cwlSchema() (in module schema_salad.tests.test_cg), 29
test_error_message9() (in module schema_salad.tests.test_codegen_errors),			test_load_metaschema() (in module
31			
test_error_message9() (in module schema_salad.tests.test_errors), 38			
test_errors() (in module schema_salad.tests.test_errors), 37			
test_errors_previously_defined_dict_key() (in module schema_salad.tests.test_errors), 38			
test_examples() (in module			

schema_salad.tests.test_cg), 29		
test_load_pt() (in schema_salad.tests.test_cg), 28	module	test_nullable_links() (in schema_salad.tests.test_examples), 44
test_load_schema_cache() (in schema_salad.tests.test_misc), 50	module	test_outputBinding() (in schema_salad.tests.test_cwlII), 34
test_loader_initialisation_disable_doc_cache() (in module schema_salad.tests.test_ref_resolver), 57		test_parser_info() (in schema_salad.tests.test_python_codegen), 53
test_loader_initialisation_for_HOME_env_var() (in module schema_salad.tests.test_ref_resolver), 56		test_plain_links_autolinked() (in schema_salad.tests.test_makedoc), 49
test_loader_initialisation_for_TMP_env_var() (in module schema_salad.tests.test_ref_resolver), 56		test_print_index() (in schema_salad.tests.test_examples), 42
test_loader_initialisation_with_neither_TMP_HOME_set() (in module schema_salad.tests.test_ref_resolver), 57		test_print_metadata() (in schema_salad.tests.test_examples), 42
test_meta_schema_gen() (in module schema_salad.tests.test_dotnet_codegen), 36		test_print_oneline() (in schema_salad.tests.test_examples), 42
test_meta_schema_gen() (in module schema_salad.tests.test_java_codegen), 47		test_print_oneline_for_errors_in_resolve_ref() (in module schema_salad.tests.test_print_oneline), 51
test_meta_schema_gen() (in module schema_salad.tests.test_python_codegen), 52		test_print_oneline_for_errors_in_the_same_line() (in module schema_salad.tests.test_print_oneline), 51
test_meta_schema_gen() (in module schema_salad.tests.test_typescript_codegen), 62		test_print_oneline_for_invalid_yaml() (in module schema_salad.tests.test_print_oneline), 51
test_meta_schema_gen_no_base() (in module schema_salad.tests.test_python_codegen), 53		test_print_pre() (in schema_salad.tests.test_examples), 42
test_meta_schema_gen_up_to_date() (in module schema_salad.tests.test_python_codegen), 52		test_print_pre_schema() (in schema_salad.tests.test_examples), 41
test_misc() (in module schema_salad.tests.test_misc), 50		test_print_rdf() (in schema_salad.tests.test_examples), 41
test_mixin() (in module schema_salad.tests.test_examples), 43		test_print_rdf_invalid_external_ref() (in module schema_salad.tests.test_examples), 41
test_multiline_list_entries_without_indention() (in module schema_salad.tests.test_makedoc), 49		test_print_schema_index() (in schema_salad.tests.test_examples), 42
test_multiline_list_entries_word_spacing() (in module schema_salad.tests.test_makedoc), 49		test_print_schema_metadata() (in schema_salad.tests.test_examples), 42
test_namespaces_type() (in module schema_salad.tests.test_errors), 39		test_rdf_datetime() (in schema_salad.tests.test_examples), 43
test_namespaces_undeclared() (in module schema_salad.tests.test_errors), 39		test_recordschema_pickle() (in schema_salad.tests.test_pickling), 50
test_nested_typedsl_ref() (in module schema_salad.tests.test_examples), 43		test_resolve_missing_step_id() (in module schema_salad.tests.test_ref_resolver), 57
test_not_a_namespace1() (in module schema_salad.tests.test_errors), 39		test_safe_identifiers() (in module schema_salad.tests.test_python_codegen), 52
test_not_a_namespace2() (in module schema_salad.tests.test_errors), 39		test_schema_salad_doc_oneline_doc() (in module schema_salad.tests.test_examples), 42
test_not_a_namespace3() (in module schema_salad.tests.test_errors), 39		test_schema_salad_inherit_docs() (in module schema_salad.tests.test_makedoc), 48
		test_schemas() (in module schema_salad.tests.test_examples), 41
		test_schemas_type() (in module schema_salad.tests.test_errors), 39
		test_scoped_id() (in module schema_salad.tests.test_examples), 43

<code>test_scoped_ref()</code>	(in <code>schema_salad.tests.test_examples</code> ), 42	<code>module</code>	<code>method)</code> , 77
<code>test_secondaryFile_dsl_ref()</code>	(in <code>schema_salad.tests.test_examples</code> ), 43	<code>module</code>	<code>to_dotnet()</code> ( <code>schema_salad.dotnet_codegen.DotNetCodeGen</code> method), 81
<code>test_secondaryFiles()</code>	(in <code>schema_salad.tests.test_cwl11</code> ), 34	<code>module</code>	<code>to_id()</code> (in module <code>schema_salad.makedoc</code> ), 96
<code>test_self_validate()</code>	(in <code>schema_salad.tests.test_examples</code> ), 41	<code>module</code>	<code>to_java()</code> ( <code>schema_salad.java_codegen.JavaCodeGen</code> method), 90
<code>test_shortname()</code>	(in <code>schema_salad.tests.test_cg</code> ), 28	<code>module</code>	<code>to_one_line_messages()</code> (in module <code>schema_salad.exceptions</code> ), 85
<code>test_skip_bad_schemas()</code>	(in <code>schema_salad.tests.test_examples</code> ), 41	<code>module</code>	<code>to_typescript()</code> ( <code>schema_salad.typescript_codegen.TypeScriptCodeGen</code> method), 151
<code>test_sourceline()</code>	(in <code>schema_salad.tests.test_examples</code> ), 43	<code>module</code>	<code>to_validation_exception()</code> (in module <code>schema_salad.ref_resolver</code> ), 136
<code>test_subscoped_id()</code>	(in <code>schema_salad.tests.test_examples</code> ), 43	<code>module</code>	ToC (class in <code>schema_salad.makedoc</code> ), 96
<code>test_subtypes()</code>	(in <code>schema_salad.tests.test_subtypes</code> ), 60	<code>module</code>	<code>type_loader()</code> ( <code>schema_salad.codegen_base.CodeGenBase</code> method), 67
<code>test_subtypes_nested()</code>	(in <code>schema_salad.tests.test_subtypes</code> ), 60	<code>module</code>	<code>type_loader()</code> ( <code>schema_salad.dotnet_codegen.DotNetCodeGen</code> method), 80
<code>test_subtypes_nested_bad()</code>	(in <code>schema_salad.tests.test_subtypes</code> ), 61	<code>module</code>	<code>type_loader()</code> ( <code>schema_salad.java_codegen.JavaCodeGen</code> method), 89
<code>test_subtypes_recursive()</code>	(in <code>schema_salad.tests.test_subtypes</code> ), 61	<code>module</code>	<code>type_loader()</code> ( <code>schema_salad.python_codegen.PythonCodeGen</code> method), 128
<code>test_subtypes_union()</code>	(in <code>schema_salad.tests.test_subtypes</code> ), 61	<code>module</code>	<code>type_loader()</code> ( <code>schema_salad.typescript_codegen.TypeScriptCodeGen</code> method), 150
<code>test_subtypes_union_bad()</code>	(in <code>schema_salad.tests.test_subtypes</code> ), 61	<code>module</code>	<code>type_loader_enum()</code> ( <code>schema_salad.dotnet_codegen.DotNetCodeGen</code> method), 80
<code>test_topmed_single_doc()</code>	( <code>schema_salad.tests.test_real_cwl.TestRealWorldCWL</code> method), 54	<code>TypeDef</code> (class in <code>schema_salad.codegen_base</code> ), 65	<code>type_loader_enum()</code> ( <code>schema_salad.java_codegen.JavaCodeGen</code> method), 89
<code>test_typeddsl_ref()</code>	(in <code>schema_salad.tests.test_examples</code> ), 43	<code>module</code>	<code>type_loader_enum()</code> ( <code>schema_salad.typescript_codegen.TypeScriptCodeGen</code> method), 151
<code>test_use_of_package_for_parser_info()</code>	(in module <code>schema_salad.tests.test_python_codegen</code> ), 53	<code>typedsl_Array_nameLoader_2</code> (in module <code>schema_salad.metaschema</code> ), 125	
<code>test_version()</code>	(in <code>schema_salad.tests.test_cli_args</code> ), 29	<code>typedsl_Documentation_nameLoader_2</code> (in module <code>schema_salad.metaschema</code> ), 126	
<code>test_yaml_datetime()</code>	(in <code>schema_salad.tests.test_examples</code> ), 43	<code>typedsl_Enum_nameLoader_2</code> (in module <code>schema_salad.metaschema</code> ), 125	
<code>test_yaml_float_test()</code>	(in <code>schema_salad.tests.test_examples</code> ), 43	<code>typedsl_loader()</code> ( <code>schema_salad.codegen_base.CodeGenBase</code> method), 68	
<code>test_yaml_tab_error()</code>	(in <code>schema_salad.tests.test_cwl11</code> ), 34	<code>typedsl_loader()</code> ( <code>schema_salad.dotnet_codegen.DotNetCodeGen</code> method), 82	
<code>testFetcher</code> (class in <code>schema_salad.tests.test_fetch</code> ), 44		<code>typedsl_loader()</code> ( <code>schema_salad.java_codegen.JavaCodeGen</code> method), 90	
<code>TestRealWorldCWL</code>	(class in <code>schema_salad.tests.test_real_cwl</code> ), 54	<code>typedsl_loader()</code> ( <code>schema_salad.python_codegen.PythonCodeGen</code> method), 130	
<code>TestSchemasDirective</code>	(class in <code>schema_salad.tests.test_schemas_directive</code> ), 59	<code>typedsl_loader()</code> ( <code>schema_salad.typescript_codegen.TypeScriptCodeGen</code> method), 152	
<code>text()</code>	( <code>schema_salad.makedoc.MyRenderer</code> method), 95	<code>typedsl_Map_nameLoader_2</code> (in module <code>schema_salad.metaschema</code> ), 125	
<code>tmp_dir_fixture()</code>	(in <code>schema_salad.tests.test_ref_resolver</code> ), 56	<code>typedsl_Record_nameLoader_2</code> (in module <code>schema_salad.metaschema</code> ), 125	
<code>to_doc_comment()</code>	( <code>schema_salad.dlang_codegen.DlangCodeGen</code> (in module <code>schema_salad.metaschema</code> ), 124	<code>typedsl_Union_nameLoader_2</code> (in module <code>schema_salad.metaschema</code> ), 125	
		<code>typedsl_union_of_PrimitiveTypeLoader_or_RecordSchemaLoader</code>	

`typefmt()` (*schema\_salad.makedoc.RenderType method*), 97

`types` (*in module schema\_salad.tests.test\_subtypes*), 60

`typescript_codegen()` (*in module schema\_salad.tests.test\_typescript\_codegen*), 62

`TypeScriptCodeGen` (*class in schema\_salad.typescript\_codegen*), 149

**U**

`Union_nameLoader` (*in module schema\_salad.metashema*), 125

`union_of_None_type_or_Any_type` (*in module schema\_salad.metashema*), 125

`union_of_None_type_or_array_of_RecordFieldLoader` (*in module schema\_salad.metashema*), 124

`union_of_None_type_or_array_of_SaladRecordFieldLoader` (*in module schema\_salad.metashema*), 125

`union_of_None_type_or_array_of_SpecializeDefLoader` (*in module schema\_salad.metashema*), 125

`union_of_None_type_or_booltype` (*in module schema\_salad.metashema*), 125

`union_of_None_type_or_inttype` (*in module schema\_salad.metashema*), 125

`union_of_None_type_or_strtype` (*in module schema\_salad.metashema*), 125

`union_of_None_type_or_strtype_or_array_of_strtype` (*in module schema\_salad.metashema*), 124

`union_of_None_type_or_strtype_or_JsonldPredicateLoader` (*in module schema\_salad.metashema*), 125

`union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader_or_ArraySchemaLoader` (*in module schema\_salad.metashema*), 124

`union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_EnumSchemaLoader_or_ArraySchemaLoader` (*in module schema\_salad.metashema*), 124

`union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader` (*in module schema\_salad.metashema*), 126

`union_of_SaladRecordSchemaLoader_or_SaladEnumSchemaLoader` (*in module schema\_salad.metashema*), 126

`UnionDefinition` (*class in schema\_salad.cpp\_codegen*), 72

`UnionSchema` (*class in schema\_salad.avro.schema*), 23

`UnionSchema` (*class in schema\_salad.metashema*), 112

`UnionSchemaLoader` (*in module schema\_salad.metashema*), 124

`uri_array_of_strtype_True_False_None_None` (*in module schema\_salad.metashema*), 125

`uri_file_path()` (*in module schema\_salad.ref\_resolver*), 136

`uri_loader()` (*schema\_salad\_codegen\_base.CodeGenBase.vocab* (*method*)), 67

`uri_loader()` (*schema\_salad\_dotnet\_codegen.DotNetCodeGen.vocab* (*method*)), 81

`uri_loader()` (*schema\_salad.java\_codegen.JavaCodeGen.vocab* (*type\_name*)), 90

`uri_loader()` (*schema\_salad.python\_codegen.PythonCodeGen method*), 129

`uri_loader()` (*schema\_salad.typescript\_codegen.TypeScriptCodeGen method*), 151

`uri_strtype_False_False_1_None` (*in module schema\_salad.metashema*), 125

`uri_strtype_True_False_None_None` (*in module schema\_salad.metashema*), 124

`uri_union_of_None_type_or_strtype_False_False_None_None` (*in module schema\_salad.metashema*), 125

`uri_union_of_None_type_or_strtype_or_array_of_strtype_Fals` (*in module schema\_salad.metashema*), 125

`uri_union_of_None_type_or_strtype_or_array_of_strtype_Fals` (*in module schema\_salad.metashema*), 125

`uri_union_of_None_type_or_strtype_True_False_None_None` (*in module schema\_salad.metashema*), 125

`uri_union_of_PrimitiveTypeLoader_or_RecordSchemaLoader_or_` (*in module schema\_salad.metashema*), 125

`urljoin()` (*schema\_salad.fetcher.DefaultFetcher method*), 86

`urljoin()` (*schema\_salad.fetcher.Fetcher method*), 85

`urljoin()` (*schema\_salad.tests.test\_fetch.CWLTestFetcher method*), 45

`urljoin()` (*schema\_salad.tests.test\_fetch.testFetcher method*), 45

`USE_ONE_OR_LIST_OF_TYPES` (*in module schema\_salad.java\_codegen*), 87

**V**

`VALID_FIELD_SORT_ORDERS` (*in module schema\_salad.avro.schema*), 142

`VALID_TYPES` (*in module schema\_salad.avro.schema*), 142

`validate()` (*in module schema\_salad.validate*), 153

`validate()` (*schema\_salad.ref\_resolver.Loader method*), 139

`validate_link()` (*schema\_salad.ref\_resolver.Loader method*), 140

`validate_links()` (*schema\_salad.ref\_resolver.Loader method*), 140

`validate_scoped()` (*schema\_salad.ref\_resolver.Loader method*), 139

`ValidationException`, 84

`values` (*schema\_salad.avro.schema.MapSchema property*), 23

`values` (*schema\_salad.avro.schema.NamedMapSchema property*), 23

`vocab` (*schema\_salad.python\_codegen\_support>LoadingOptions attribute*), 104

`vocab` (*schema\_salad.python\_codegen\_support>LoadingOptions attribute*), 132

`vocab_type_name()` (*in module schema\_salad.makedoc*), 94

`vpformat()` (*in module schema\_salad.validate*), 154

## W

`with_sourceline()` (*schema\_salad.exceptions.SchemaSaladException method*), 83  
`writeDefinition()` (*schema\_salad.cpp\_codegen.ClassDefinition method*), 71  
`writeDefinition()` (*schema\_salad.cpp\_codegen.EnumDefinition method*), 73  
`writeDefinition()` (*schema\_salad.cpp\_codegen.FieldDefinition method*), 71  
`writeDefinition()` (*schema\_salad.cpp\_codegen.MapDefinition method*), 72  
`writeDefinition()` (*schema\_salad.cpp\_codegen.UnionDefinition method*), 73  
`writeFwdDeclaration()`  
    (*schema\_salad.cpp\_codegen.ClassDefinition method*), 70  
`writeFwdDeclaration()`  
    (*schema\_salad.cpp\_codegen.MapDefinition method*), 72  
`writeFwdDeclaration()`  
    (*schema\_salad.cpp\_codegen.UnionDefinition method*), 72  
`writeImplDefinition()`  
    (*schema\_salad.cpp\_codegen.ClassDefinition method*), 71  
`writeImplDefinition()`  
    (*schema\_salad.cpp\_codegen.MapDefinition method*), 72  
`writeImplDefinition()`  
    (*schema\_salad.cpp\_codegen.UnionDefinition method*), 73