sphobjinv

Release 2.0.1

Brian Skinn

Jan 27, 2020

CONTENTS

A toolkit for inspection/manipulation of Sphinx objects inventories

When documentation is built using, e.g., Sphinx's `StandaloneHTMLBuilder`, an inventory of the named objects in the documentation set is dumped to a file called objects.inv in the html build directory. (One common location is, doc/build/html, though the exact location will vary depending on the details of how Sphinx is configured.) This file is read by `intersphinx` when generating links in other documentation.

Since version 1.0 of Sphinx (~July 2010), the data in these objects.inv inventories is compressed by `zlib` (presumably to reduce storage requirements and improve download speeds; "version 2"), whereas prior to that date the data was left uncompressed ("version 1"). This compression renders the files non-human-readable. **It is the purpose of this package to enable quick and simple compression/decompression and inspection of these "version 2" inventory files.**

In particular, was developed to satisfy two primary use cases:

1. Searching and inspection of objects.inv contents in order to identify how to properly construct `intersphinx` references.
2. Assembly of new objects.inv files in order to allow `intersphinx` cross-referencing of other documentation sets that were not created by Sphinx.

Install via pip:

```
$ pip install sphobjinv
```

The package is configured for use both as a *command-line script* and as a *Python package*.

Installing the optional dependency `python-Levenshtein` substantially accelerates the the "suggest" functionality; see *here* for more information.

The project source repository is on GitHub: bskinn/sphobjinv.

CHAPTER

ONE

COMMAND-LINE USAGE

The CLI for is implemented using two subparsers, one each for the *convert* and *suggest* sub-functions. More information about the implementation of these features can be found *here* and in the documentation for the *Inventory* object, in particular the *data_file()* and *suggest()* methods.

Some notes on these CLI docs:

- CLI examples are executed in a sandboxed directory pre-loaded with objects_attrs.inv (from, e.g., here).
- Path (from pathlib) is imported into the namespace before all tests.
- cli_run is a helper function that enables doctesting of CLI examples by mimicking execution of a shell command. It is described in more detail here.
- file_head is a helper function that retrieves the head of a specified file.

The options for the parent command are:

-h, --help
Show help message and exit

```
>>> cli_run('sphobjinv --help')
usage: sphobjinv [-h] [-v] {convert,suggest} ...

Format conversion for and introspection of intersphinx 'objects.inv' files.

optional arguments:
  -h, --help         show this help message and exit
  -v, --version      Print package version & other info

Subcommands:
  {convert,suggest}  Execution mode. Type 'sphobjinv [mode] -h' for more
                     information on available options. Mode names can be
                     abbreviated to their first two letters.
    convert (co)     Convert intersphinx inventory to zlib-compressed,
                     plaintext, or JSON formats.
    suggest (su)     Fuzzy-search intersphinx inventory for desired object(s).
```

-v, --version
Print package version & other info

```
>>> cli_run('sphobjinv --version')

sphobjinv v2.0.1

Copyright (c) Brian Skinn 2016-2020
License: The MIT License
```

(continues on next page)

(continued from previous page)

```
Bug reports & feature requests: https://github.com/bskinn/sphobjinv
Documentation: http://sphobjinv.readthedocs.io
```

1.1 Command-Line Usage: "convert" Mode

The convert subparser is used for all conversions of "version 2" Sphinx inventory files among plaintext, zlib-compressed, and (unique to) JSON formats. Currently, the CLI can read inventory data from local files in any of these three formats, as well as the standard zlib-compressed format from files in remote locations (see *--url*). At the moment, the only output method supported is writing to a local file, again in all three formats.

Note: If reading from stdin or writing to stdout would be useful to you, please leave a note at #74 so I can gauge interest.

Basic file conversion to the default output filename is straightforward:

```
>>> Path('objects_attrs.txt').is_file()
False
>>> cli_run('sphobjinv convert plain objects_attrs.inv')

Conversion completed.
'...objects_attrs.inv' converted to '...objects_attrs.txt' (plain).

>>> print(file_head('objects_attrs.txt', head=6))
# Sphinx inventory version 2
# Project: attrs
# Version: 17.2
# The remainder of this file is compressed using zlib.
attr.Attribute py:class 1 api.html#$ -
attr.Factory py:class 1 api.html#$ -
```

A different target filename can be specified, to avoid overwriting an existing file:

```
>>> cli_run('sphobjinv convert plain objects_attrs.inv', inp='n\n')

File exists. Overwrite (Y/N)? n

Exiting...

>>> cli_run('sphobjinv convert plain objects_attrs.inv objects_attrs_foo.txt')

Conversion completed.
'...objects_attrs.inv' converted to '...objects_attrs_foo.txt' (plain).
```

If you don't provide an output file extension, the defaults (*.inv*/*.txt*/*.json*) will be used.

If you want to pull an input file directly from the Web, use *--url* (note that the base filename is **not** inferred from the indicated URL):

```
>>> cli_run('sphobjinv convert plain -u https://github.com/bskinn/sphobjinv/raw/
↪master/tests/resource/objects_attrs.inv')
```

(continues on next page)

(continued from previous page)

```
Remote inventory found.

Conversion completed.
'https://github.com/b[...]ce/objects_attrs.inv' converted to '...objects.txt' (plain).

>>> print(file_head('objects.txt', head=6))
# Sphinx inventory version 2
# Project: attrs
# Version: 17.2
# The remainder of this file is compressed using zlib.
attr.Attribute py:class 1 api.html#$ -
attr.Factory py:class 1 api.html#$ -
```

The URL provided **MUST** have the leading protocol specified (here, https://).

It is not necessary to locate the objects.inv file before running ; for most Sphinx documentation sets, if you provide a URL to any page in the docs, it will automatically find and use the correct objects.inv:

```
>>> cli_run('sphobjinv convert plain -ou https://docs.python.org/3/library/urllib.
↪error.html#urllib.error.URLError')

No inventory at provided URL.
Attempting "https://docs.python.org/3/library/urllib.error.html/objects.inv" ...
Attempting "https://docs.python.org/3/library/objects.inv" ...
Attempting "https://docs.python.org/3/objects.inv" ...
Remote inventory found.

Conversion completed.
'...objects.inv' converted to '...objects.txt' (plain).
```

only supports download of zlib-compressed objects.inv files by URL. Plaintext download by URL is unreliable, presumably due to encoding problems. If download of JSON files by URL is desirable, please submit an issue.

Usage

```
>>> cli_run('sphobjinv convert --help', head=4)
usage: sphobjinv convert [-h] [-e | -c] [-o] [-q] [-u]
                         {zlib,plain,json} infile [outfile]

Convert intersphinx inventory to zlib-compressed, plaintext, or JSON formats.
```

Positional Arguments

mode
: Conversion output format.

  Must be one of *plain*, *zlib*, or *json*

infile
: Path (or URL, if *--url* is specified) to file to be converted.

outfile
: *(Optional)* Path to desired output file. Defaults to same directory and main file name as input file but with extension .inv/.txt/.json, as appropriate for the output format. A bare path is accepted here, using the default output file name/extension.

Flags

-h, --help
Display *convert* help message and exit.

-o, --overwrite
If the output file already exists, overwrite without prompting for confirmation.

-q, --quiet
Suppress all output to *stdout*, regardless of success or failure. Useful for scripting/automation. Implies *--overwrite*.

-u, --url
Treat *infile* as a URL for download.

-e, --expand
Expand any abbreviations in *uri* or *dispname* fields before writing to output; see *here*. Cannot be specified with *--contract*.

-c, --contract
Contract *uri* and *dispname* fields, if possible, before writing to output; see *here*. Cannot be specified with *--expand*.

1.2 Command-Line Usage: "suggest" Mode

The suggest subparser is used to query an inventory for objects fuzzy-matching a given search string. Fuzzy-matching is carried out via the `fuzzywuzzy` library, against the Restructured Text-like representation of each object exposed by *SuperDataObj.as_rst*:

```
>>> cli_run('sphobjinv suggest objects_attrs.inv instance')

:py:exception:`attr.exceptions.FrozenInstanceError`
:py:function:`attr.validators.instance_of`
```

The `fuzzywuzzy` match score and the index of the object within the inventory can be printed by passing the *--score* and *--index* options, respectively:

```
>>> cli_run('sphobjinv suggest objects_attrs.inv instance -s -i')

  Name                                                  Score    Index
------------------------------------------------------  -------  -------
:py:exception:`attr.exceptions.FrozenInstanceError`       90         9
:py:function:`attr.validators.instance_of`                90        23
```

If too few or too many matches are returned, the reporting threshold can be changed via *--thresh*:

```
>>> cli_run('sphobjinv suggest objects_attrs.inv instance -s -i -t 48')

  Name                                                  Score    Index
------------------------------------------------------  -------  -------
:py:exception:`attr.exceptions.FrozenInstanceError`       90         9
:py:function:`attr.validators.instance_of`                90        23
:std:doc:`license`                                        51        47
:py:function:`attr.filters.include`                       48        13
```

Remote objects.inv files can be retrieved for inspection by passing the *--url* flag:

```
>>> cli_run('sphobjinv suggest https://github.com/bskinn/sphobjinv/raw/master/tests/
↪resource/objects_attrs.inv instance -u -t 48')

Remote inventory found.

:py:exception:`attr.exceptions.FrozenInstanceError`
:py:function:`attr.validators.instance_of`
:std:doc:`license`
:py:function:`attr.filters.include`
```

The URL provided **MUST** have the leading protocol specified (here, https://).

It is not necessary to locate the objects.inv file before running ; for most Sphinx documentation sets, if you provide a URL to any page in the docs, it will automatically find and use the correct objects.inv:

```
>>> cli_run('sphobjinv suggest -u https://sphobjinv.readthedocs.io/en/v2.0rc1/cmdline.
↪html compress')

No inventory at provided URL.
Attempting "https://sphobjinv.readthedocs.io/en/v2.0rc1/cmdline.html/objects.inv" ...
Attempting "https://sphobjinv.readthedocs.io/en/v2.0rc1/objects.inv" ...
Remote inventory found.

:py:function:`sphobjinv.zlib.compress`
:py:function:`sphobjinv.zlib.decompress`
```

only supports download of zlib-compressed objects.inv files by URL. Plaintext download by URL is unreliable, presumably due to encoding problems. If download of JSON files by URL is desirable, please submit an issue.

Usage

```
>>> cli_run('sphobjinv suggest --help', head=3)
usage: sphobjinv suggest [-h] [-a] [-i] [-s] [-t {0-100}] [-u] infile search

Fuzzy-search intersphinx inventory for desired object(s).
```

Positional Arguments

infile
: Path (or URL, if *--url* is specified) to file to be converted.

search
: Search term for `fuzzywuzzy` matching

Flags

-h, --help
: Display *suggest* help message and exit.

-a, --all
: Display all search results without prompting, regardless of the number of hits. Otherwise, prompt if number of results exceeds *sphobjinv.cmdline.SUGGEST_CONFIRM_LENGTH*.

-i, --index
: Display the index position within the *Inventory.objects* list for each search result returned.

-s, --score
: Display the `fuzzywuzzy` match score for each search result returned.

-t, --thresh <#>
Change the `fuzzywuzzy` match quality threshold (0-100; higher values yield fewer results). Default is specified in *sphobjinv.cmdline.DEF_THRESH*.

-u, --url
Treat *infile* as a URL for download.

CHAPTER

TWO

API USAGE

In all of the below, the package has been imported as soi, and the working temp directory has been populated with the objects_attrs.inv inventory.

2.1 Inspecting an Inventory

Inspecting the contents of an existing inventory is handled entirely by the *Inventory* class:

```
>>> inv = soi.Inventory('objects_attrs.inv')
>>> print(inv)
<Inventory (fname_zlib): attrs v17.2, 56 objects>
>>> inv.version
'17.2'
>>> inv.count
56
```

The individual objects contained in the inventory are represented by instances of the *DataObjStr* class, which are stored in a `list` in the *objects* attribute:

```
>>> len(inv.objects)
56
>>> dobj = inv.objects[0]
>>> dobj
DataObjStr(name='attr.Attribute', domain='py', role='class', priority='1', uri='api.
↪html#$', dispname='-')
>>> dobj.name
'attr.Attribute'
>>> dobj.domain
'py'
>>> [d.name for d in inv.objects if 'validator' in d.uri]
['api_validators', 'examples_validators']
```

Inventory objects can also import from plaintext or zlib-compressed inventories, as `bytes`:

```
>>> inv2 = soi.Inventory(inv.data_file())
>>> print(inv2)
<Inventory (bytes_plain): attrs v17.2, 56 objects>
>>> inv3 = soi.Inventory(soi.compress(inv.data_file()))
>>> print(inv3)
<Inventory (bytes_zlib): attrs v17.2, 56 objects>
```

Remote objects.inv files can also be retrieved via URL, with the *url* keyword argument:

```
>>> inv4 = soi.Inventory(url='https://github.com/bskinn/sphobjinv/raw/master/tests/
↪resource/objects_attrs.inv')
>>> print(inv4)
<Inventory (url): attrs v17.2, 56 objects>
```

2.2 Modifying an Inventory

The *DataObjStr* instances can be edited in place:

```
>>> inv = soi.Inventory('objects_attrs.inv')
>>> inv.objects[0]
DataObjStr(name='attr.Attribute', domain='py', role='class', priority='1', uri='api.
↪html#$', dispname='-')
>>> inv.objects[0].uri = 'attribute.html'
>>> inv.objects[0]
DataObjStr(name='attr.Attribute', domain='py', role='class', priority='1', uri=
↪'attribute.html', dispname='-')
```

New instances can be easily created either by direct instantiation, or by *evolve()*:

```
>>> inv.objects.append(inv.objects[0].evolve(name='attr.Generator', uri='generator.
↪html'))
>>> inv.count
57
>>> inv.objects[-1]
DataObjStr(name='attr.Generator', domain='py', role='class', priority='1', uri=
↪'generator.html', dispname='-')
```

The other attributes of the *Inventory* instance can also be freely modified:

```
>>> inv.project = 'not_attrs'
>>> inv.version = '0.1'
>>> print(inv)
<Inventory (fname_zlib): not_attrs v0.1, 57 objects>
```

2.3 Formatting Inventory Contents

The contents of the *Inventory* can be converted to the plaintext objects.inv format **as** bytes via *data_file()*:

```
>>> inv = soi.Inventory('objects_attrs.inv')
>>> print(*inv.data_file().splitlines()[:6], sep='\n')
b'# Sphinx inventory version 2'
b'# Project: attrs'
b'# Version: 17.2'
b'# The remainder of this file is compressed using zlib.'
b'attr.Attribute py:class 1 api.html#$ -'
b'attr.Factory py:class 1 api.html#$ -'
```

This method makes use of the *DataObjStr.data_line* method to format each of the object information lines.

If desired, the *shorthand* used for the *uri* and *dispname* fields can be expanded:

```
>>> print(*inv.data_file(expand=True).splitlines()[4:6], sep='\n')
b'attr.Attribute py:class 1 api.html#attr.Attribute attr.Attribute'
b'attr.Factory py:class 1 api.html#attr.Factory attr.Factory'
>>> do = inv.objects[0]
>>> do.data_line(expand=True)
'attr.Attribute py:class 1 api.html#attr.Attribute attr.Attribute'
```

2.4 Exporting an Inventory

Inventory instances can be written to disk in three formats: zlib-compressed objects.inv, plaintext objects.txt, and JSON. The API does not provide single-function means to do this, however.

To start, load the source objects.inv:

```
>>> from pathlib import Path
>>> inv = soi.Inventory('objects_attrs.inv')
```

To export plaintext:

```
>>> df = inv.data_file()
>>> soi.writebytes('objects_attrs.txt', df)
>>> print(*Path('objects_attrs.txt').read_text().splitlines()[:6], sep='\n')
# Sphinx inventory version 2
# Project: attrs
# Version: 17.2
# The remainder of this file is compressed using zlib.
attr.Attribute py:class 1 api.html#$ -
attr.Factory py:class 1 api.html#$ -
```

For zlib-compressed:

```
>>> dfc = soi.compress(df)
>>> soi.writebytes('objects_attrs_new.inv', dfc)
>>> print(*Path('objects_attrs_new.inv').read_bytes().splitlines()[:4], sep='\n')
b'# Sphinx inventory version 2'
b'# Project: attrs'
b'# Version: 17.2'
b'# The remainder of this file is compressed using zlib.'
>>> print(Path('objects_attrs_new.inv').read_bytes().splitlines()[6][:10])
b'5\xcb0\xd7\x9f>\xf3\x84\x89'
```

For JSON:

```
>>> jd = inv.json_dict()
>>> soi.writejson('objects_attrs.json', jd)
>>> print(Path('objects_attrs.json').read_text()[:51])
{"project": "attrs", "version": "17.2", "count": 56
```

CHAPTER

THREE

CREATING AND USING A CUSTOM OBJECTS.INV

The workflow presented here is introduced in the context of manually assembling an objects inventory, but the functionality is mainly intended for use downstream of a web-scraping or other automated content-extraction tool.

A (possibly obsolete) representative example of such a custom objects.inv can be found at the GitHub repo here.

Note: These instructions are for v2.0; the prior instructions for v1.0 can be found here.

1. Identify the head of the URI to the documentation.
2. Construct an *Inventory* containing all of the objects of interest. The *uri* and *dispname* values can be entered with or without the *standard abbreviations*.
   - Create an empty *Inventory*:

```
>>> import sphobjinv as soi
>>> inv = soi.Inventory()
>>> print(inv)
<Inventory (manual): <no project> <no version>, 0 objects>
```

   - Define the *project* and *version* attributes:

```
>>> inv.project = 'foobar'
>>> inv.version = '1.5'
>>> print(inv)
<Inventory (manual): foobar v1.5, 0 objects>
```

   - Append new *DataObjStr* instances to *objects* as needed to populate the inventory:

```
>>> o = soi.DataObjStr(name='baz', domain='py', role='class',
... priority='1', uri='api.html#$', dispname='-')
>>> print(o)
<DataObjStr:: :py:class:`baz`>
>>> inv.objects.append(o)
>>> print(inv)
<Inventory (manual): foobar v1.5, 1 objects>
>>> inv.objects.append(soi.DataObjStr(name='baz.quux', domain='py',
... role='method', priority='1', uri='api.html#$', dispname='-'))
>>> inv.objects.append(soi.DataObjStr(name='quuux', domain='py',
... role='function', priority='1', uri='api.html#$', dispname='-'))
>>> print(inv)
<Inventory (manual): foobar v1.5, 3 objects>
```

Note: The *role* values here must be the **full** role names ("*block directives*"), described as the

"directives" in the Sphinx documentation for domains, and not the abbreviated forms ("*inline directives*") used when constructing cross-references.

Thus, for example, a *DataObjStr* corresponding to a method on a class should be constructed with role='method', not role='meth'.

3. Export the *Inventory* in compressed form.
   - Generate the text of the inventory file with *data_file()*, optionally *contracting* the *uri* and *dispname* fields:

     ```
     >>> text = inv.data_file(contract=True)
     ```

   - Compress the file text:

     ```
     >>> ztext = soi.compress(text)
     ```

   - Save to disk:

     ```
     >>> soi.writebytes('objects_foobar.inv', ztext)
     ```

4. Transfer the compressed file to its distribution location.
   - If only local access is needed, it can be kept local.
   - If external access needed, upload to a suitable host.
5. Add an element to the `intersphinx_mapping` parameter in `conf.py`.
   - The key of the element is an arbitrary name, which can be used to specify the desired documentation set to be searched for the target object, in the event of a *name* collision between one or more documentation projects; e.g.:

     ```
     :meth:`python:str.join`
     ```

   - The value of the element is a `tuple` of length two:
     - The first element of the value tuple is the head URI for the documentation repository, identified in step (1), to which the *uri* of given object is appended when constructing an `intersphinx` cross-reference.
     - The second element of the value tuple can be `None`, in which case the objects.inv file is assumed to be at the repository head URI. Otherwise, this element is the complete address of the distribution location of the compressed inventory file, from step (4), whether a local path or a remote URL.

   Examples:

   ```
   intersphinx_mapping = {
       # Standard reference to web docs, with web objects.inv
       'python': ('https://docs.python.org/3.5', None),

       # Django puts its objects.inv file in a non-standard location
       'django': ('http://docs.djangoproject.com/en/dev/', 'https://
   ↪docs.djangoproject.com/en/dev/_objects/'),

       # Drawing the Sphinx objects.inv from a local copy, but␣
   ↪referring to the 1.7 web docs
   ```

(continues on next page)

(continued from previous page)

```
    'sphinx': ('http://www.sphinx-doc.org/en/1.7/', '/path/to/
↪local/objects.inv',
}
```

CHAPTER

FOUR

SPEEDING UP "SUGGEST" WITH PYTHON-LEVENSHTEIN

uses `fuzzywuzzy` for fuzzy-match searching of object names/domains/roles as part of the *`Inventory.suggest()`* functionality, also implemented as the CLI *suggest* subcommand.

By default, `fuzzywuzzy` uses `difflib.SequenceMatcher` from the Python standard library for its fuzzy searching. However, it also can use `python-Levenshtein`, a Python C extension providing similar functionality. Thus, has been implemented with `python-Levenshtein` as an *optional* dependency: it will be used if it is available.

4.1 Installation

On Linux, simple installation via pip install python-Levenshtein should work properly in the vast majority of cases. This may work relatively smoothly for MacOS as well, but first-hand confirmation was not (yet) available as of this writing.

On Windows, again as of this writing, no binary wheel is available on PyPI. So, the easiest way to install is to download a wheel from Christoph Gohlke's repository and run *pip install C:\path\to\wheel*. Just make sure to match Python version and CPU type (win32 vs win_amd64) when you download.

4.2 Performance Benchmark

The chart below presents one dataset illustrating the performance enhancement that can be obtained by installing `python-Levenshtein`. The timings plotted here are from execution of `timeit.repeat()` in the Python standard library around a *`suggest()`* call, searching for the term "function", for a number of objects.inv files from different projects (see here).

The timings were collected using the following code:

```
import sphobjinv as soi

durations = {}
obj_counts = {}

for fn in os.listdir():
    if fn.endswith('.inv'):
        inv = soi.Inventory(fn)

        # Execute the 'suggest' operation 20 times for each file
        timings = timeit.repeat("inv.suggest('function')", repeat=20, number=1,␣
↪globals=globals())
```

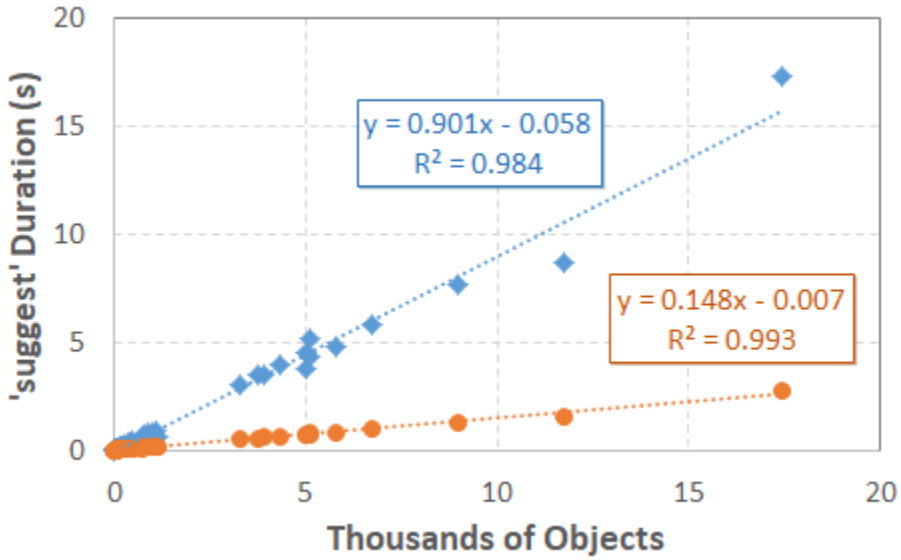
(continues on next page)

(continued from previous page)

```
        # Store the average timing for each file
        durations.update({fn: sum(timings) / len(timings)})

        # Store the number of objects
        obj_counts.update({fn: inv.count})
```

As can be seen, the fifty-two objects.inv files in this dataset contain widely varying numbers of objects n:



Unsurprisingly, larger inventories require more time to search. Also relatively unsurprisingly, the time required appears to be roughly $O(n)$, since the fuzzy search must be performed once on the `as_rst` representation of each object.

For this specific search, using `python-Levenshtein` instead of `difflib` decreases the time required from 0.90 seconds per thousand objects down to 0.15 seconds per thousand objects, representing a performance improvement of almost exactly six-fold. Other searches will likely exhibit somewhat better or worse improvement from the use of `python-Levenshtein`, depending on the average length of the reST-like representations of the objects in an objects.inv and the length of the search term.

4.3 Variations in Matching Behavior

Note that the matching scores calculated by `difflib` and `python-Levenshtein` can often differ appreciably. (This is illustrated in issue #128 of the `fuzzywuzzy` GitHub repo.) This difference in behavior doesn't have much practical significance, save for the potential of causing some confusion between users with/without `python-Levenshtein` installed.

As an example, the following shows an excerpt of the results of a representative CLI *suggest* call **without** `python-Levenshtein`:

```
$ sphobjinv suggest objects_scipy.inv surface -asit 40

  Name                                                  Score    Index
------------------------------------------------------  -------  -------
:py:function:`scipy.misc.face`                            64      1018
:py:function:`scipy.misc.source`                          64      1032
```

(continues on next page)

(continued from previous page)

```
:std:doc:`generated/scipy.misc.face`                          64      4042
:std:doc:`generated/scipy.misc.source`                        64      4056
:py:data:`scipy.stats.rice`                                   56      2688
:std:label:`continuous-rice`                                  56      2896
:py:method:`scipy.integrate.complex_ode.successful`           51       156
:py:method:`scipy.integrate.ode.successful`                   51       171
:py:function:`scipy.linalg.lu_factor`                         51       967

... more with score 51 ...

:py:attribute:`scipy.LowLevelCallable.signature`              50         5
:py:function:`scipy.constants.convert_temperature`            50        53
:py:function:`scipy.integrate.quadrature`                     50       176

... more with score 50 and below ...
```

This is a similar excerpt **with** `python-Levenshtein`:

```
  Name                                                  Score    Index
-------------------------------------------------------  -------  -------
:py:function:`scipy.misc.face`                           64      1018
:py:function:`scipy.misc.source`                         64      1032
:std:doc:`generated/scipy.misc.face`                     64      4042
:std:doc:`generated/scipy.misc.source`                   64      4056
:py:method:`scipy.integrate.ode.successful`              51       171
:py:function:`scipy.linalg.lu_factor`                    51       967
:py:function:`scipy.linalg.subspace_angles`              51      1003

... more with score 51 ...

:py:function:`scipy.cluster.hierarchy.fcluster`          49        23
:py:function:`scipy.cluster.hierarchy.fclusterdata`      49        24
:py:method:`scipy.integrate.complex_ode.successful`      49       156

... more with score 49 and below ...
```

CHAPTER

FIVE

SPHINX OBJECTS.INV V2 SYNTAX

After decompression, "version 2" Sphinx objects.inv files follow a syntax that, to the best of this author's ability to determine, is completely undocumented. The below syntax is believed to be accurate as of Jun 2018 (Sphinx v1.7.4). Based upon a quick `git diff` of the Sphinx repository, it is thought to be accurate for all Sphinx versions >=1.0b1 that make use of this "version 2" objects.inv format.

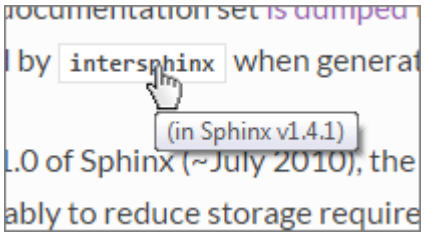
The first line must be exactly:

```
# Sphinx inventory version 2
```

The second and third lines must obey the template:

```
# Project: <project name>
# Version: <full version number>
```

The above project name and version are used to populate mouseovers for the `intersphinx` cross-references:



The fourth line must contain the string 'zlib' somewhere in it, but for the purposes of consistency it should be exactly:

```
# The remainder of this file is compressed using zlib.
```

All remaining lines of the file are the objects data, each laid out in the following syntax:

```
{name} {domain}:{role} {priority} {uri} {dispname}
```

`{name}` The object name used when cross-referencing the object (falls between the backticks)

`{domain}` The Sphinx domain used when cross-referencing the object (falls between the first and second colons; omitted if using the default domain)

`{role}` The Sphinx role used when cross-referencing the object (falls between the second and third colons; or, between the first and second colons if using the default domain)

`{priority}` Flag for placement in search results. Most will be 1 (standard priority) or -1 (omit from results)

`{uri}` Relative URI for the location to which cross-references will point. The base URI is taken from the relevant element of the `intersphinx_mapping` configuration parameter of `conf.py`.

`{dispname}` Default cross-reference text to be displayed in compiled documentation.

Note: The above fields MUST NOT contain spaces, except for `{dispname}` which MAY contain them.

For illustration, the following is the entry for the `join()` method of the `str` class in the Python 3.5 objects.inv, broken out field-by-field:

```
str.join py:method 1 library/stdtypes.html#$ -

{name}      = str.join
{domain}    = py
{role}      = method
{priority}  = 1
{uri}       = library/stdtypes.html#$
{dispname}  = -
```

The above illustrates two shorthand notations that were introduced to shrink the size of the inventory file:

1. If `{uri}` has an anchor (technically a "fragment identifier," the portion following the `#` symbol) and the tail of the anchor is identical to `{name}`, that tail is replaced with `$`.
2. If `{dispname}` is identical to `{name}`, it is stored as `-`.

Thus, a standard `intersphinx` reference to this method would take the form (the leading `:py` could be omitted if `py` is the default domain):

```
:py:meth:`str.join`
```

The cross-reference would show as `str.join()` and link to the relative URI:

```
library/stdtypes.html#str.join
```

Other intersphinx Syntax Examples

To show as only `join()`:

```
:py:meth:`~str.join`
```

To suppress the hyperlink as in `str.join()`:

```
:py:meth:`!str.join`
```

To change the cross-reference text and omit the trailing parentheses as in `This is join!`:

```
:py:obj:`This is join! <str.join>`
```

CHAPTER

SIX

API

Most (all?) of the objects documented in the below submodules are also exposed at the package root. For example, both of the following will work to import the *Inventory* class:

```
>>> from sphobjinv import Inventory
>>> from sphobjinv.inventory import Inventory
```

6.1 sphobjinv.data

sphobjinv *data classes for individual objects*.

sphobjinv is a toolkit for manipulation and inspection of Sphinx objects.inv files.

Author Brian Skinn (bskinn@alum.mit.edu)

File Created 7 Nov 2017

Copyright (c) Brian Skinn 2016-2020

Source Repository http://www.github.com/bskinn/sphobjinv

Documentation http://sphobjinv.readthedocs.io

License The MIT License; see LICENSE.txt for full license terms

Members

class DataFields

Enum for the fields of objects.inv data objects.

DispName = 'dispname'

Default display name for the object in rendered documentation when referenced as :domain:role:`name`

Domain = 'domain'

Sphinx domain housing the object

Name = 'name'

Object name, as recognized internally by Sphinx

Priority = 'priority'

Object search priority

Role = 'role'

Full name of Sphinx role to be used when referencing the object

URI = 'uri'

URI to the location of the object's documentation, relative to the documentation root

class DataObjBytes(*name*, *domain*, *role*, *priority*, *uri*, *dispname*, *as_str=NOTHING*, *as_bytes=NOTHING*)
`SuperDataObj` subclass generating `bytes` object data.

class DataObjStr(*name*, *domain*, *role*, *priority*, *uri*, *dispname*, *as_bytes=NOTHING*, *as_str=NOTHING*)
`SuperDataObj` subclass generating `str` object data.

class SuperDataObj
Abstract base superclass defining common methods &c. for data objects.

Intended only to be subclassed by `DataObjBytes` and `DataObjStr`, to allow definition of common methods, properties, etc. all in one place.

Where marked with †, `DataObjBytes` instances will return `bytes` values, whereas `DataObjStr` instances will return `str` values.

abstract property as_bytes
`DataObjBytes` version of instance.

property as_rst
`str` reST reference-like object representation.

Typically will NOT function as a proper reST reference in Sphinx source (e.g., a *role* of function must be referenced using :func: for the py domain).

abstract property as_str
`DataObjStr` version of instance.

data_line(*, *expand=False*, *contract=False*)
Compose plaintext objects.inv data line from instance contents.

The format of the resulting data line is given by `data_line_fmt`. `DataObjBytes` and `DataObjStr` instances generate data lines as `bytes` and `str`, respectively.

Calling with both *expand* and *contract* as `True` is invalid.

Parameters

- **expand** – `bool` *(optional)* – Return data line with any `uri` or `dispname` abbreviations expanded
- **contract** – `bool` *(optional)* – Return data line with abbreviated `uri` and `dispname`

Returns *dl* – `bytes` (for `DataObjBytes`) or `str` (for `DataObjStr`) – Object data line

Raises **ValueError** – If both *expand* and *contract* are `True`

data_line_fmt = '{name} {domain}:{role} {priority} {uri} {dispname}'
Helper `str` for generating plaintext objects.inv data lines. The field names MUST match the `str` values of the `DataFields` members.

abstract property dispname
Object default name in rendered documentation†.

Possibly abbreviated; see *here*.

abstract property dispname_abbrev
Abbreviation character(s) for display name†.

`'-'` or `b'-'` for *version 2* objects.inv files.

property dispname_contracted
Object display name, contracted with *dispname_abbrev*.

property dispname_expanded
Object display name, with *dispname_abbrev* expanded.

abstract property domain
Sphinx domain containing the object†.

evolve(**kwargs)
Create a new instance with changes applied.

This helper method provides a concise means for creating new instances with only a subset of changed data fields.

The names of any *kwargs* MUST be keys of the `dict`s generated by *`json_dict()`*.

Parameters **kwargs** – `str` or `bytes` – Revised value(s) to use in the new instance for the passed keyword argument(s).

Returns *dobj* – *`DataObjBytes`* or *`DataObjStr`* – New instance with updated data

json_dict(*, *expand=False*, *contract=False*)
Return the object data formatted as a flat `dict`.

The returned `dict` is constructed such that it matches the relevant subschema of *`sphobjinv.schema.json_schema`*, to facilitate implementation of *`Inventory.json_dict()`*.

The `dict`s returned by *`DataObjBytes`* and *`DataObjStr`* both have `str` keys, but they have `bytes` and `str` values, respectively. The `dict` keys are identical to the `str` values of the *`DataFields`* `Enum` members.

Calling with both *expand* and *contract* as `True` is invalid.

Parameters

- **expand** – `bool` *(optional)* – Return `dict` with any *`uri`* or *`dispname`* abbreviations expanded
- **contract** – `bool` *(optional)* – Return `dict` with abbreviated *`uri`* and *`dispname`*

Returns *d* – `dict` – Object data

Raises **ValueError** – If both *expand* and *contract* are `True`

abstract property name
Object name, as recognized internally by Sphinx†.

abstract property priority
Object search priority†.

abstract property role
Sphinx role to be used when referencing the object†.

rst_fmt = ':{domain}:{role}:`{name}`'
`str.format()` template for generating reST-like representations of object data for *`as_rst`* (used with *`Inventory.suggest()`*).

abstract property uri
Object URI relative to documentation root†.

Possibly abbreviated; see *here*.

abstract property uri_abbrev
Abbreviation character(s) for URI tail†.

`'$'` or `b'$'` for *version 2* objects.inv files.

property uri_contracted
Object relative URI, contracted with *uri_abbrev*.

property uri_expanded
Object relative URI, with *uri_abbrev* expanded.

6.2 sphobjinv.enum

Helper enums for `sphobjinv`.

`sphobjinv` is a toolkit for manipulation and inspection of Sphinx objects.inv files.

Author Brian Skinn (bskinn@alum.mit.edu)

File Created 4 May 2019

Copyright (c) Brian Skinn 2016-2020

Source Repository http://www.github.com/bskinn/sphobjinv

Documentation http://sphobjinv.readthedocs.io

License The MIT License; see LICENSE.txt for full license terms

Members

class HeaderFields
`Enum` for various inventory-level data items.

A subset of these `Enum` values is used in various Regex, JSON, and string formatting contexts within class:*~sphobjinv.inventory.Inventory* and *schema.json_schema*.

Count = 'count'
Number of objects contained in the inventory

Metadata = 'metadata'
The `str` value of this `Enum` member is accepted as a root-level key in a `dict` to be imported into an *Inventory*. The corresponding value in the `dict` may contain any arbitrary data. Its possible presence is accounted for in *schema.json_schema*.

The data associated with this key are **ignored** during import into an *Inventory*.

Project = 'project'
Project name associated with an inventory

Version = 'version'
Project version associated with an inventory

class SourceTypes
`Enum` for the import mode used in instantiating an *Inventory*.

Since `Enum` keys iterate in definition order, the definition order here defines the order in which *Inventory* objects attempt to parse a source object passed to *Inventory.__init__()* either as a positional argument or via the generic *source* keyword argument.

This order **DIFFERS** from the documentation order, which is alphabetical.

BytesPlaintext = 'bytes_plain'
Instantiation from a plaintext objects.inv `bytes`.

BytesZlib = 'bytes_zlib'
Instantiation from a zlib-compressed objects.inv `bytes`.

DictJSON = 'dict_json'
Instantiation from a `dict` validated against *schema.json_schema*.

FnamePlaintext = 'fname_plain'
Instantiation from a plaintext objects.inv file on disk.

FnameZlib = 'fname_zlib'
Instantiation from a zlib-compressed objects.inv file on disk.

Manual = 'manual'
No source; *Inventory* was instantiated with *project* and *version* as empty strings and *objects* as an empty `list`.

URL = 'url'
Instantiation from a zlib-compressed objects.inv file downloaded from a URL.

6.3 sphobjinv.error

Custom errors for `sphobjinv`.

`sphobjinv` is a toolkit for manipulation and inspection of Sphinx objects.inv files.

Author Brian Skinn (bskinn@alum.mit.edu)

File Created 5 Nov 2017

Copyright (c) Brian Skinn 2016-2020

Source Repository http://www.github.com/bskinn/sphobjinv

Documentation http://sphobjinv.readthedocs.io

License The MIT License; see LICENSE.txt for full license terms

Members

exception SphobjinvError
Custom `sphobjinv` error superclass.

exception VersionError
Raised when attempting an operation on an unsupported version.

The current version of `sphobjinv` only supports 'version 2' objects.inv files (see *here*).

6.4 sphobjinv.fileops

File I/O helpers for `sphobjinv`.

`sphobjinv` is a toolkit for manipulation and inspection of Sphinx objects.inv files.

Author Brian Skinn (bskinn@alum.mit.edu)

File Created 5 Nov 2017

Copyright (c) Brian Skinn 2016-2020

Source Repository http://www.github.com/bskinn/sphobjinv

Documentation http://sphobjinv.readthedocs.io

License The MIT License; see LICENSE.txt for full license terms

Members

readbytes(*path*)
Read file contents and return as bytes.

Parameters **path** – str – Path to file to be opened.

Returns *b* – bytes – Contents of the indicated file.

readjson(*path*)
Create dict from JSON file.

No data or schema validation is performed.

Parameters **path** – str – Path to JSON file to be read.

Returns *d* – dict – Deserialized JSON.

urlwalk(*url*)
Generate a series of candidate objects.inv URLs.

URLs are based on the seed *url* passed in. Ensure that the path separator in *url* is the standard **forward** slash ('/').

Parameters **url** – str – Seed URL defining directory structure to walk through.

Yields *inv_url* – str – Candidate URL for objects.inv location.

writebytes(*path*, *contents*)
Write indicated file contents.

Any existing file at *path* will be overwritten.

Parameters

- **path** – str – Path to file to be written.
- **contents** – bytes – Content to be written to file.

writejson(*path*, *d*)
Create JSON file from dict.

No data or schema validation is performed. Any existing file at *path* will be overwritten.

Parameters

- **path** – str – Path to output JSON file.
- **d** – dict – Data structure to serialize.

6.5 sphobjinv.inventory

sphobjinv *data class for full inventories*.

sphobjinv is a toolkit for manipulation and inspection of Sphinx objects.inv files.

Author Brian Skinn (bskinn@alum.mit.edu)

File Created 7 Dec 2017

Copyright (c) Brian Skinn 2016-2020

Source Repository http://www.github.com/bskinn/sphobjinv

Documentation http://sphobjinv.readthedocs.io

License The MIT License; see LICENSE.txt for full license terms

Members

class Inventory(*source=None*, *plaintext=None*, *zlib=None*, *fname_plain=None*, *fname_zlib=None*, *dict_json=None*, *url=None*, *count_error=True*)

Entire contents of an objects.inv inventory.

All information is stored internally as `str`, even if imported from a `bytes` source.

All arguments except *count_error* are used to specify the source from which the *Inventory* contents are to be populated. **At most ONE** of these source arguments may be other than `None`.

The *count_error* argument is only relevant to the *dict_json* source type.

source

The *Inventory* will attempt to parse the indicated source object as each of the below types in sequence, **except** for *url*.

This argument is included mainly as a convenience feature for use in interactive sessions, as invocations of the following form implicitly populate *source*, as the first positional argument:

```
>>> inv = Inventory(src_obj)
```

In most cases, for clarity it is recommended that programmatic instantiation of *Inventory* objects utilize the below format-specific arguments.

plaintext

Object is to be parsed as the `bytes` plaintext contents of an objects.inv inventory.

zlib

Object is to be parsed as the `bytes` zlib-compressed contents of an objects.inv inventory.

fname_plain

Object is the `str` path to a file containing the plaintext contents of an objects.inv inventory.

fname_zlib

Object is the `str` path to a file containing the zlib-compressed contents of an objects.inv inventory.

dict_json

Object is a `dict` containing the contents of an objects.inv inventory, conforming to the JSON schema of *schema.json_schema*.

If *count_error* is passed as `True`, then a `ValueError` is raised if the number of objects found in the `dict` does not match the value associated with its *count* key. If *count_error* is passed as `False`, an object count mismatch is ignored.

url

Object is a `str` URL to a zlib-compressed objects.inv file. Any URL type supported by `urllib.request` SHOULD work; only http: and file: have been directly tested, however.

No authentication is supported at this time.

Members

property count

Count of objects currently in inventory.

data_file(*, *expand=False*, *contract=False*)
Generate a plaintext objects.inv as `bytes`.

`bytes` is used here as the output type since the most common use cases are anticipated to be either (1) dumping to file via *sphobjinv.fileops.writebytes()* or (2) compressing via *sphobjinv.zlib.compress()*, both of which take `bytes` input.

Calling with both *expand* and *contract* as `True` is invalid.

Parameters

- **expand** – `bool` *(optional)* – Generate `bytes` with any *uri* or *dispname* abbreviations expanded
- **contract** – `bool` *(optional)* – Generate `bytes` with abbreviated *uri* and *dispname* values

Returns *b* – `bytes` – Inventory in plaintext objects.inv format

Raises **ValueError** – If both *expand* and *contract* are `True`

header_preamble = '# Sphinx inventory version 2'
Preamble line for v2 objects.inv header

header_project = '# Project: {project}'
Project line `str.format()` template for objects.inv header

header_version = '# Version: {version}'
Version line `str.format()` template for objects.inv header

header_zlib = '# The remainder of this file is compressed using zlib.'
zlib compression line for v2 objects.inv header

json_dict(*expand=False*, *contract=False*)
Generate a flat `dict` representation of the inventory.

The returned `dict` matches the schema of *sphobjinv.schema.json_schema*.

Calling with both *expand* and *contract* as `True` is invalid.

Parameters

- **expand** – `bool` *(optional)* – Return `dict` with any *uri* or *dispname* abbreviations expanded
- **contract** – `bool` *(optional)* – Return `dict` with abbreviated *uri* and *dispname* values

Returns *d* – `dict` – Inventory data; keys and values are all `str`

Raises **ValueError** – If both *expand* and *contract* are `True`

objects
`list` of *DataObjStr* representing the data objects of the inventory. Can be edited directly to change the inventory contents. Undefined/random behavior/errors will result if the type of the elements is anything other than *DataObjStr*.

property objects_rst
`list` of objects formatted in a `str` reST-like representation.

The format of each `str` in the `list` is given by *data.SuperDataObj.rst_fmt*.

Calling with both *expand* and *contract* as `True` is invalid.

Parameters

- **expand** – bool *(optional)* – Return strs with any *uri* or *dispname* abbreviations expanded
- **contract** – bool *(optional)* – Return strs with abbreviated *uri* and *dispname* values

Returns *obj_l* – list of str – Inventory object data in reST-like format

Raises **ValueError** – If both *expand* and *contract* are True

project
str project display name for the inventory (see *here*).

source_type
SourceTypes Enum value indicating the type of source from which the instance was generated.

suggest(*name*, *, *thresh=50*, *with_index=False*, *with_score=False*)
Suggest objects in the inventory to match a name.

suggest() makes use of the powerful pattern-matching library fuzzywuzzy to identify potential matches to the given *name* within the inventory. The search is performed over the list of str generated by *objects_rst()*.

thresh defines the minimum fuzzywuzzy match quality (an integer ranging from 0 to 100) required for a given object to be included in the results list. Can be any float value, but best results are generally obtained with values between 50 and 80, depending on the number of objects in the inventory, the confidence of the user in the match between *name* and the object(s) of interest, and the desired fidelity of the search results to *name*.

This functionality is provided by the *'suggest' subparser* of the command-line interface.

Parameters

- **name** – str – Object name for fuzzywuzzy pattern matching
- **thresh** – float – fuzzywuzzy match quality threshold
- **with_index** – bool – Include with each matched name its index within *Inventory.objects*
- **with_score** – bool – Include with each matched name its fuzzywuzzy match quality score

Returns

res_l – list

If both *with_index* and *with_score* are False, members are the str *SuperDataObj.as_rst()* representations of matching objects.

If either is True, members are tuples of the indicated match results:

with_index == *True*: (as_rst, index)

with_score == *True*: (as_rst, score)

with_index == *with_score* == *True*: (as_rst, score, index)

version
str project display version for the inventory (see *here*).

6.6 sphobjinv.re

Helper regexes for `sphobjinv`.

`sphobjinv` is a toolkit for manipulation and inspection of Sphinx objects.inv files.

Author Brian Skinn (bskinn@alum.mit.edu)

File Created 5 Nov 2017

Copyright (c) Brian Skinn 2016-2020

Source Repository http://www.github.com/bskinn/sphobjinv

Documentation http://sphobjinv.readthedocs.io

License The MIT License; see LICENSE.txt for full license terms

Members

p_data = re.compile(' ^ # Start of line\n (?P<name>[^#]\\S+) # --> Name\n \\s+ # Dividing s
Compiled re `str` regex pattern for data lines in `str` decompressed inventory files

pb_comments = re.compile(b'^#.*$', re.MULTILINE)
Compiled re `bytes` pattern for comment lines in decompressed inventory files

pb_data = re.compile(b' ^ # Start of line\n (?P<name>[^#]\\S+) # --> Name\n \\s+ # Dividing
Compiled re `bytes` regex pattern for data lines in `bytes` decompressed inventory files

pb_project = re.compile(b'\n ^ # Start of line\n [#][ ]Project:[ ] # Preamble\n (?P<project
Compiled re `bytes` pattern for project line

pb_version = re.compile(b'\n ^ # Start of line\n [#][ ]Version:[ ] # Preamble\n (?P<versio
Compiled re `bytes` pattern for version line

ptn_data = ' ^ # Start of line\n (?P<name>[^#]\\S+) # --> Name\n \\s+ # Dividing space\n ('
Regex pattern string used to compile *p_data* and *pb_data*

6.7 sphobjinv.schema

JSON schema to validate inventory dictionaries.

This module is part of `sphobjinv`, a toolkit for manipulation and inspection of Sphinx objects.inv files.

Author Brian Skinn (bskinn@alum.mit.edu)

File Created 7 Dec 2017

Copyright (c) Brian Skinn 2016-2020

Source Repository http://www.github.com/bskinn/sphobjinv

Documentation http://sphobjinv.readthedocs.io

License The MIT License; see LICENSE.txt for full license terms

Members

json_schema = {'$schema': 'http://json-schema.org/schema#', 'additionalProperties': False
JSON schema for validating the `dict` forms of Sphinx objects.inv inventories as generated from or expected by *Inventory* classes.

6.8 sphobjinv.zlib

zlib (de)compression helpers for `sphobjinv`.

`sphobjinv` is a toolkit for manipulation and inspection of Sphinx objects.inv files.

Author Brian Skinn (bskinn@alum.mit.edu)

File Created 5 Nov 2017

Source Repository http://www.github.com/bskinn/sphobjinv

Documentation http://sphobjinv.readthedocs.io

Members

compress(*bstr*)

Compress a version 2 `intersphinx` objects.inv bytestring.

The #-prefixed comment lines are left unchanged, whereas the plaintext data lines are compressed with `zlib`.

Parameters **bstr** – `bytes` – Binary string containing the decompressed contents of an objects.inv file.

Returns *out_b* – `bytes` – Binary string containing the compressed objects.inv content.

decompress(*bstr*)

Decompress a version 2 `intersphinx` objects.inv bytestring.

The #-prefixed comment lines are left unchanged, whereas the `zlib`-compressed data lines are decompressed to plaintext.

Parameters **bstr** – `bytes` – Binary string containing a compressed objects.inv file.

Returns *out_b* – `bytes` – Decompressed binary string containing the plaintext objects.inv content.

CHAPTER

SEVEN

SPHOBJINV.CMDLINE (NON-API)

CLI module for `sphobjinv`.

`sphobjinv` is a toolkit for manipulation and inspection of Sphinx objects.inv files.

Note: This module is NOT part of the public API for `sphobjinv`. Its entire contents should be considered implementation detail.

Author Brian Skinn (bskinn@alum.mit.edu)

File Created 17 May 2016

Source Repository http://www.github.com/bskinn/sphobjinv

Documentation http://sphobjinv.readthedocs.io

Members

do_convert(*inv*, *in_path*, *params*)
Carry out the conversion operation, including writing output.

If *OVERWRITE* is passed and the output file (the default location, or as passed to *OUTFILE*) exists, it will be overwritten without a prompt. Otherwise, the user will be queried if it is desired to overwrite the existing file.

If *QUIET* is passed, nothing will be printed to stdout (potentially useful for scripting), and any existing output file will be overwritten without prompting.

Parameters

- **inv** – *Inventory* – Inventory object to be output in the format indicated by *MODE*.
- **in_path** – str – For a local input file, its absolute path. For a URL, the (possibly truncated) URL text.
- **params** – dict – Parameters/values mapping from the active subparser

do_suggest(*inv*, *params*)
Perform the suggest call and output the results.

Results are printed one per line.

If neither *INDEX* nor *SCORE* is specified, the results are output without a header. If either or both are specified, the results are output in a lightweight tabular format.

If the number of results exceeds *SUGGEST_CONFIRM_LENGTH*, the user will be queried whether to display all of the returned results unless *ALL* is specified.

No --quiet option is available here, since a silent mode for suggestion output is nonsensical.

Parameters

- **inv** – *Inventory* – Inventory object to be output in the format indicated by *MODE*.
- **params** – dict – Parameters/values mapping from the active subparser

err_format(*exc*)
Pretty-format an exception.

Parameters **exc** – Exception – Exception instance to pretty-format

Returns *pretty_exc* – str – Exception type and message formatted as '{type}: {message}'

getparser()
Generate argument parser.

Returns *prs* – ArgumentParser – Parser for commandline usage of sphobjinv

import_infile(*in_path*)
Attempt import of indicated file.

Convenience function wrapping attempts to load an *Inventory* from a local path.

Parameters **in_path** – str – Path to input file

Returns *inv* – *Inventory* or None – If instantiation with the file at *in_path* succeeds, the resulting *Inventory* instance; otherwise, None

inv_local(*params*)
Create *Inventory* from local source.

Uses *resolve_inpath()* to sanity-check and/or convert *INFILE*.

Calls sys.exit() internally in error-exit situations.

Parameters **params** – dict – Parameters/values mapping from the active subparser

Returns

- *inv* – *Inventory* – Object representation of the inventory at *INFILE*
- *in_path* – str – Input file path as resolved/checked by *resolve_inpath()*

inv_url(*params*)
Create *Inventory* from file downloaded from URL.

Initially, treats *INFILE* as a download URL to be passed to the *url* initialization argument of *Inventory*.

If an inventory is not found at that exact URL, progressively searches the directory tree of the URL for objects.inv.

Calls sys.exit() internally in error-exit situations.

Parameters **params** – dict – Parameters/values mapping from the active subparser

Returns

- *inv* – *Inventory* – Object representation of the inventory at *INFILE*
- *ret_path* – str – URL from *INFILE* used to construct *inv*. If URL is longer than 45 characters, the central portion is elided.

main()
Handle command line invocation.

Parses command line arguments, handling the no-arguments and *VERSION* cases.

Creates the *Inventory* from the indicated source and method.

Invokes *do_convert()* or *do_suggest()* per the subparser name stored in *SUBPARSER_NAME*.

resolve_inpath(*in_path*)
Resolve the input file, handling invalid values.

Currently, only checks for existence and not-directory.

Parameters **in_path** – str – Path to desired input file

Returns *abs_path* – str – Absolute path to indicated file

Raises **FileNotFoundError** – If a file is not found at the given path

resolve_outpath(*out_path*, *in_path*, *params*)
Resolve the output location, handling mode-specific defaults.

If the output path or basename are not specified, they are taken as the same as the input file. If the extension is unspecified, it is taken as the appropriate mode-specific value from *DEF_OUT_EXT*.

If *URL* is passed, the input directory is taken to be os.getcwd() and the input basename is taken as *DEF_BASENAME*.

Parameters

- **out_path** – str or None – Output location provided by the user, or None if omitted
- **in_path** – str – For a local input file, its absolute path. For a URL, the (possibly truncated) URL text.
- **params** – dict – Parameters/values mapping from the active subparser

Returns *out_path* – str – Absolute path to the target output file

selective_print(*thing*, *params*)
Print *thing* if not in quiet mode.

Quiet mode is indicated by the value at the *QUIET* key within *params*.

Quiet mode is not implemented for the "*suggest*" CLI mode.

Parameters

- **thing** – *any* – Object to be printed
- **params** – dict – Parameters/values mapping from the active subparser

write_json(*inv*, *path*, *, *expand=False*, *contract=False*)
Write an *Inventory* to JSON.

Writes output via *fileops.writejson()*.

Calling with both *expand* and *contract* as True is invalid.

Parameters

- **inv** – *Inventory* – Objects inventory to be written zlib-compressed
- **path** – str – Path to output file
- **expand** – bool *(optional)* – Generate output with any *uri* or *dispname* abbreviations expanded
- **contract** – bool *(optional)* – Generate output with abbreviated *uri* and *dispname* values

Raises **ValueError** – If both *expand* and *contract* are True

write_plaintext(*inv*, *path*, *, *expand=False*, *contract=False*)
Write an *Inventory* to plaintext.

Newlines are inserted in an OS-aware manner, based on the value of `os.linesep`.

Calling with both *expand* and *contract* as `True` is invalid.

Parameters

- **inv** – *Inventory* – Objects inventory to be written as plaintext
- **path** – `str` – Path to output file
- **expand** – `bool` *(optional)* – Generate output with any *uri* or *dispname* abbreviations expanded
- **contract** – `bool` *(optional)* – Generate output with abbreviated *uri* and *dispname* values

Raises **ValueError** – If both *expand* and *contract* are `True`

write_zlib(*inv*, *path*, *, *expand=False*, *contract=False*)
Write an *Inventory* to zlib-compressed format.

Calling with both *expand* and *contract* as `True` is invalid.

Parameters

- **inv** – *Inventory* – Objects inventory to be written zlib-compressed
- **path** – `str` – Path to output file
- **expand** – `bool` *(optional)* – Generate output with any *uri* or *dispname* abbreviations expanded
- **contract** – `bool` *(optional)* – Generate output with abbreviated *uri* and *dispname* values

Raises **ValueError** – If both *expand* and *contract* are `True`

yesno_prompt(*prompt*)
Query user at *stdin* for yes/no confirmation.

Uses `input()`, so will hang if used programmatically unless *stdin* is suitably mocked.

The value returned from `input()` must satisfy either resp.lower() == 'n' or resp.lower() == 'y', or else the query will be repeated *ad infinitum*. This function does **NOT** augment *prompt* to indicate the constraints on the accepted values.

Parameters **prompt** – `str` – Prompt to display to user that requests a 'Y' or 'N' response

Returns *resp* – `str` – User response

ALL = 'all'
Optional argument name for use with the *SUGGEST* subparser, indicating to print all returned objects, regardless of the number returned, without asking for confirmation

CONTRACT = 'contract'
Optional argument name for use with the *CONVERT* subparser, indicating to contract URIs and display names to abbreviated forms in the generated output file

CONVERT = 'convert'
Subparser name for inventory file conversions; stored in *SUBPARSER_NAME* when selected

DEF_BASENAME = 'objects'
Default base name for an unspecified *OUTFILE*

DEF_OUT_EXT = {'json': '.json', 'plain': '.txt', 'zlib': '.inv'}
Default extensions for an unspecified *OUTFILE*

DEF_THRESH = 75
Default match threshold for *sphobjinv suggest --thresh*

EXPAND = 'expand'
Optional argument name for use with the *CONVERT* subparser, indicating to expand URI and display name abbreviations in the generated output file

HELP_CONV_EXTS = "'.inv/.txt/.json'"
Help text for default extensions for the various conversion types

HELP_CO_PARSER = 'Convert intersphinx inventory to zlib-compressed, plaintext, or JSON forr
Help text for the *CONVERT* subparser

HELP_SU_PARSER = 'Fuzzy-search intersphinx inventory for desired object(s).'
Help text for the *SUGGEST* subparser

INDEX = 'index'
Optional argument name for use with the *SUGGEST* subparser, indicating to print the location index of each returned object within *INFILE* along with the object domain/role/name (may be specified with *SCORE*)

INFILE = 'infile'
Required positional argument name for use with both *CONVERT* and *SUGGEST* subparsers, holding the path (or URL, if *URL* is specified) to the input file

JSON = 'json'
Argument value for *CONVERT MODE*, to output an inventory as JSON

MODE = 'mode'
Positional argument name for use with *CONVERT* subparser, indicating output file format (*ZLIB*, *PLAIN* or *JSON*)

OUTFILE = 'outfile'
Optional positional argument name for use with the *CONVERT* subparser, holding the path to the output file (*DEF_BASENAME* and the appropriate item from *DEF_OUT_EXT* are used if this argument is not provided)

OVERWRITE = 'overwrite'
Optional argument name for use with the *CONVERT* subparser, indicating to overwrite any existing output file without prompting

PLAIN = 'plain'
Argument value for *CONVERT MODE*, to output a plaintext inventory

QUIET = 'quiet'
Optional argument name for use with the *CONVERT* subparser, indicating to suppress console output

SCORE = 'score'
Optional argument name for use with the *SUGGEST* subparser, indicating to print the `fuzzywuzzy` score of each returned object within *INFILE* along with the object domain/role/name (may be specified with *INDEX*)

SEARCH = 'search'
Positional argument name for use with the *SUGGEST* subparser, holding the search term for `fuzzywuzzy` text matching

SUBPARSER_NAME = 'sprs_name'
Param for storing subparser name (*CONVERT* or *SUGGEST*)

SUGGEST = 'suggest'
Subparser name for inventory object suggestions; stored in *SUBPARSER_NAME* when selected

SUGGEST_CONFIRM_LENGTH = 30
Number of returned objects from a *SUGGEST* subparser invocation above which user will be prompted for confirmation to print the results (unless *ALL* is specified)

THRESH = 'thresh'
Optional argument name for use with the *SUGGEST* subparser, taking the minimum desired fuzzywuzzy match quality as one required argument

URL = 'url'
Optional argument name for use with both *CONVERT* and *SUGGEST* subparsers, indicating that *INFILE* is to be treated as a URL rather than a local file path

VERSION = 'version'
Optional argument name for use with the base argument parser, to show version &c. info, and exit

VER_TXT = '\nsphobjinv v2.0.1\n\nCopyright (c) Brian Skinn 2016-2020\nLicense: The MIT Li
Version &c. output blurb

ZLIB = 'zlib'
Argument value for *CONVERT MODE*, to output a zlib-compressed inventory

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

Symbols

A

B

Y

Z