wxflow

Release 0.1.0

NOAA/NWS/NCEP/EMC

May 03, 2024

CONTENTS

CHAPTER

ONE

OVERVIEW

wxflow is a Python library of common tools used in weather workflows. It is designed to be used in NWP applications such as GFS, GEFS, and RRFS workflows. Some of the tools included in wxflow are:

- **logger**: A generic program-wide logging tool.
- **yamltools**: A YAML parser that allows loading of nested yaml files and resolves environment variables.
- **timetools**: A collection of datetime tools that allows for easy transformation between the various formats of datetime used in the applications.
- **executable**: A tool that allows for running command line programs within the python environment. executables.
- **jinja**: A jinja tool that allows for the use of Jinja2 templates with filters.
- **task**: A task tool that allows for the use of a YAML file to configure tasks.

1.1 Installation

```
pip install wxflow
```

1.2 Disclaimer

The United States Department of Commerce (DOC) GitHub project code is provided on an "as is" basis and the user assumes responsibility for its use. DOC has relinquished control of the information and no longer has responsibility to protect the integrity, confidentiality, or availability of the information. Any claims against the Department of Commerce stemming from the use of its GitHub project will be governed by all applicable Federal law. Any reference to specific commercial products, processes, or services by service mark, trademark, manufacturer, or otherwise, does not constitute or imply their endorsement, recommendation or favoring by the Department of Commerce. The Department of Commerce seal and logo, or the seal and logo of a DOC bureau, shall not be used in any manner to imply endorsement of any commercial product or activity by DOC or the United States Government.

1.2.1 Contributing

We love contributions here in `wxflow`! If you're looking for something to work on then check out our issue tracker for open issues.

If you want to make a contribution to `wxflow` then please raise a Pull Request on GitHub.

To help speed up the review process please ensure the following:

- The PR addresses an open issue.
- All tests are passing locally with `pytest`.
- The project passes linting with `isort` and `pycodestyle`.
- If adding a new feature you also add documentation.

Developing

To check out a local copy of the project you can fork the project on GitHub and then clone it locally.

```
$ git clone https://github.com/yourusername/wxflow
$ cd wxflow
```

This project uses `isort` to sort Python import definitions alphabetically and `pycodestyle` as the Python style checker against conventions in PEP8. We also support `pre-commit` to ensure these have been run. To configure your local environment please install these development dependencies and set up the commit hooks.

```
$ pip install isort pycodestyle pre-commit
$ pre-commit install
```

You can check that things are working correctly by calling pre-commit directly.

```
$ pre-commit run --all-files
isort....................................Passed
pycodestyle..............................Passed
```

These checks will be run automatically when you make a commit (if `pre-commit` has been installed).

Testing

This project uses `pytest` to run tests and also to test docstring examples.

Install the test dependencies.

```
$ pip install .dev
```

Run the tests.

```
$ pytest
=== 3 passed in 0.13 seconds ===
```

If you are working on a new feature please add tests to ensure the feature works as expected. If you are working on a bug fix then please add a test to ensure there is no regression.

Tests are stored in `wxflow/tests` and follow the pytest format.

```python
from datetime import datetime
from wxflow import to_datetime

def test_to_datetime():
    assert to_datetime('20220314') == datetime(2022, 3, 14)
```

Making a Pull Request

Once you've made your changes and are ready to make a Pull Request please ensure tests and linting pass locally before pushing to GitHub. When making your Pull Request please include a short description of the changes, but more importantly why they are important. Perhaps by writing a before and after paragraph with user examples.

Also consider how your title look when it appears in a changelog. Does it full describe the change to an outside user? For example `Add support for checking supported datetime` is a much better title than `Fixes #56`.

```
# Add support for validating a string can be transformed into a datetime object

Closes #56

**Changes**

This PR allows the inspection of strings to check if it can be transformed into a␣
↪datetime object.

**Before**

If a user passed a random string to `is_supported_datetime` it would return `False`.

```python
>>> from wxflow import is_supported_datetime
>>> is_supported_datetime('2012 Jun 15, 12:23')
False
```

After

If a user passes a valid, supported datetime string, it will return true


```python
>>> from wxflow import is_supported_datetime
>>> is_supported_datetime('20120615T1223z')
True
```
```

1.2.2 Maintaining

Reviewing Pull Requests

This project generally accepts any Pull Request which improves the project.

Any small issues which fix typos or improve documentation can be merged straight in.

Any bug fix or enhancement PRs should reference an open issue which describes the problem. This gives us and other users the opportunity to discuss the change before anyone invests time in implementing it.

Typically we have a "yes and" policy when it comes to reviewing where we generally accept whatever is contributed even if it's not quite right. If you have a small amount of feedback during review, such as the user forgot to run `black` or you want to reword something in a docstring it's preferable to just push extra commits to the PR, or just merge and raise a follow up PR to tweak things.

For larger design or implementation feedback then feel free to push this back on to the contributor.

Releasing

We follow Semantic Versioning and releases are published automatically when a tag is pushed to GitHub. Ensure that the *__version__* in src/wxflow/__init__.py is updated to the updated version before tagging.

```
# Set next version number
export RELEASE=x.x.x

# Edit src/wxflow/__init__.py
__version__ = $RELEASE

# Create tags
git add src/wxflow/__init__.py
git commit -m "Release $RELEASE"
git tag -a $RELEASE -m "Version $RELEASE"

# Push
git push upstream --tags
```

1.2.3 API

class wxflow.**YAMLFile**(*path=None*, *data=None*)

Bases: `AttrDict`

Reads a YAML file as an AttrDict and recursively converts nested dictionaries into AttrDict. This is the entry point for all YAML files.

wxflow.**parse_yaml**(*path=None*, *data=None*, *encoding='utf-8'*, *loader=<class 'yaml.loader.SafeLoader'>*)

Load a yaml configuration file and resolve any environment variables The environment variables must have !ENV before them and be in this format to be parsed: ${VAR_NAME}. E.g.: database:

host: !ENV ${HOST} port: !ENV ${PORT}

app:

log_path: !ENV '/var/${LOG_PATH}' something_else: !ENV '${AWESOME_ENV_VAR}/var/${A_SECOND_AWESOME_VAR}'

Parameters

- **path** (*str*) – the path to the yaml file
- **data** (*str*) – the yaml data itself as a stream
- **loader** (*Type[yaml.loader]*) – Specify which loader to use. Defaults to yaml.SafeLoader
- **encoding** (*str*) – the encoding of the data if a path is specified, defaults to utf-8

Returns
the dict configuration

Return type
Dict[str, Any]

Adopted from: https://dev.to/mkaranasou/python-yaml-configuration-with-environment-variables-parsing-2ha6

wxflow.**parse_j2yaml**(*path: str*, *data: Dict*, *searchpath: str | List = '/'*) → Dict[str, Any]

Description

Load a compound jinja2-templated yaml file and resolve any templated variables. The jinja2 templates are first resolved and then the rendered template is parsed as a yaml.

Parameters

- **path** (*str*) – the path to the jinja2 templated yaml file
- **data** (*Dict[str, Any], optional*) – the context for jinja2 templating
- **searchpath** (*str | List*) – additional search paths for included jinja2 templates

returns
the dict configuration

rtype
Dict[str, Any]

wxflow.**save_as_yaml**(*data*, *target*)

wxflow.**dump_as_yaml**(*data*)

wxflow.**vanilla_yaml**(*ctx*)
Transform an input object of complex type as a plain type

wxflow.**to_datetime**(*dtstr: str*) → datetime

Description

Translate a string into a datetime object in a generic way. The string can also support ISO 8601 representation.

Formats accepted (T, Z, -, :) are optional: YYYY-mm-dd YYYY-mm-ddTHHZ YYYY-mm-ddTHH:MMZ YYYY-mm-ddTHH:MM:SSZ

Parameters
dtstr (*str*) – String to be translated into a datetime object

returns
Datetime object

rtype
datetime.datetime

wxflow.**to_timedelta**(*tdstr: str*) → timedelta

Description

Translate a string into a timedelta object in a generic way

Formats accepted (<sign>, T, Z) are optional: <sign><dd>dT<hh>H<mm>M<ss>SZ <sign><dd>day(s), hh:mm:ss

<sign> can be +/-, default is + <dd> can be any integer, default is 0 <hh> can be any integer, default is 0 <mm> can be any integer, default is 0 <ss> can be any integer, default is 0

Parameters
tdstr (*str*) – String to be translated into a timedelta object

returns
Timedelta object

rtype
datetime.timedelta

wxflow.**datetime_to_YMDH**(*dt: datetime*) → str

Description

Translate a datetime object to 'YYYYmmddHH' format.

Parameters
dt (*datetime.datetime*) – Datetime object to translate.

returns
str – Formatted string in 'YYYYmmddHH' format.

rtype
str

wxflow.**datetime_to_YMD**(*dt: datetime*) → str

Description

Translate a datetime object to 'YYYYmmdd' format.

Parameters
dt (*datetime.datetime*) – Datetime object to translate.

returns
str – Formatted string in 'YYYYmmdd' format.

rtype
str

wxflow.**datetime_to_JDAY**(*dt: datetime*) → str

Description

Translate a datetime object to 'YYYYDOY' format.

Parameters
dt (*datetime.datetime*) – Datetime object to translate

returns
str – Formatted string in 'YYYYDOY' format.

rtype
str

wxflow.**timedelta_to_HMS**(*td: timedelta*) → str

Description

Translate a timedelta object to 'HH:MM:SS' format.

Parameters
td (*datetime.timedelta*) – Timedelta object to translate.

returns
str – Formatted string in 'HH:MM:SS' format.

rtype
str

wxflow.**strftime**(*dt: datetime*, *fmt: str*) → str

Return a formatted string from a datetime object.

wxflow.**strptime**(*dtstr: str*, *fmt: str*) → datetime

Description

Translate a formatted string into datetime object.

Parameters

- **dtstr** (*str*) – Datetime string to translate.
- **fmt** (*str*) – Datetime string format.

returns
datetime.datetime – Datetime object.

rtype
datetime.datetime

wxflow.**to_YMDH**(*dt: datetime*) → str

Description

Translate a datetime object to 'YYYYmmddHH' format.

Parameters
dt (*datetime.datetime*) – Datetime object to translate.

returns
str – Formatted string in 'YYYYmmddHH' format.

rtype
str

wxflow.**to_YMD**(*dt: datetime*) → str

Description

Translate a datetime object to 'YYYYmmdd' format.

Parameters
dt (*datetime.datetime*) – Datetime object to translate.

returns
str – Formatted string in 'YYYYmmdd' format.

rtype
str

wxflow.**to_JDAY**(*dt: datetime*) → str

Description

Translate a datetime object to 'YYYYDOY' format.

Parameters
dt (*datetime.datetime*) – Datetime object to translate

returns
str – Formatted string in 'YYYYDOY' format.

rtype
str

wxflow.**to_julian**(*dt: datetime*) → str

Description

Translate a datetime object to 'YYYYDOY' format.

Parameters
dt (*datetime.datetime*) – Datetime object to translate

returns
str – Formatted string in 'YYYYDOY' format.

rtype
str

wxflow.**to_isotime**(*dt: datetime*) → str

Description

Return a ISO formatted '%Y-%m-%dT%H:%M:%SZ' string from a datetime object.

Parameters
dt (*datetime.datetime*) – Datetime object to format.

returns
str – Formatted string in ISO format.

rtype
str

wxflow.**to_fv3time**(*dt: datetime*) → str

Description

Return a FV3 formatted string from a datetime object.

Parameters
dt (*datetime.datetime*) – Datetime object to format.

returns
str – Formatted string in FV3 format.

rtype
str

wxflow.**add_to_datetime**(*dt: datetime*, *td: timedelta*) → datetime

Description

Adds a timedelta to a datetime object.

Parameters

- **dt** (*datetime.datetime*) – Datetime object to add to.
- **td** (*datetime.timedelta*) – Timedelta object to add.

rtype
datetime.datetime

wxflow.**add_to_timedelta**(*td1*, *td2*)

Description

Adds two timedelta objects.

Parameters

- **td1** (*datetime.timedelta*) – First timedelta object to add.
- **td2** (*datetime.timedelta*) – Second timedelta object to add.

rtype
datetime.timedelta

class wxflow.Jinja(*template_path_or_string: str*, *data: Dict*, *allow_missing: bool = True*, *searchpath: str | List = '/'*)

Bases: object

Description

A wrapper around jinja2 to render templates

Description

Given a path to a (jinja2) template and a data object, substitute the template file with data. Allow for retaining missing or undefined variables. Also provide additional search paths for templates that may be included in the main template :Parameters: * **template_path_or_string** (*str*) – Path to the template file or a templated string

- **data** (*dict*) – Data to be substituted into the template TODO: make "data" optional so that the user can render the same template with different data
- **allow_missing** (*bool*) – If True, allow for missing or undefined variables
- **searchpath** (*str | list*) – Additional search paths for templates (default '/')

static add_filter_to_env(*env: Environment*, *filter_name: str*, *filter_func: callable*) → Environment

Description

Add a custom filter to the jinja2 environment :Parameters: * **env** (*jinja2.Environment*) – Active jinja2 environment

- **filter_name** (*str*) – name of the filter
- **filter_func** (*callable*) – function that will be called

returns
env – Active jinja2 environment with the new filter added

rtype
jinja2.Environment

property dump: None

Description

Render and dump the output to stdout :rtype: None

get_set_env(*loader: BaseLoader*, *filters: Dict[str, callable] = None*) → Environment

Description

Define the environment for the jinja2 template Any number of filters can be added here. Optionally, a dictionary of external filters can be passed in

Currently, the following filters are defined: strftime: convert a datetime object to a string with a user defined format to_isotime: convert a datetime object to an ISO 8601 string to_fv3time: convert a datetime object to a FV3 time string to_YMDH: convert a datetime object to a YYYYMMDDHH string to_YMD: convert a datetime object to a YYYYMMDD string to_julian: convert a datetime object to a julian day to_f90bool: convert a boolean to a fortran boolean relpath: convert a full path to a relative path based on an input root_path getenv: read variable from environment if defined, else UNDEFINED to_timedelta: convert a string to a timedelta object add_to_datetime: add time to a datetime, return new datetime object

The Expression Statement extension "jinja2.ext.do", which enables
{% do ... %} statements. These are useful for appending to lists. e.g. {{ bar.append(foo) }} would print "None" to the parsed jinja template, but {% do bar.append(foo) %} would not.

Parameters

- **loader** (*jinja2.BaseLoader*) – An instance of class jinja2.BaseLoader
- **filters** (*Dict[str, callable] (optional)*) – A dictionary of filters to be added to the environment

returns
env

rtype
jinja2.Environment

property render: str

Description

Render the Jinja2 template with the data :Parameters: **None**

returns

- **rendered** (*str*)
- *Rendered template into text*

save(*output_file: str*) → None

Description

Render and save the output to a file :Parameters: * **output_file** (*str*)

- **Path to the output file**

rtype
None

class wxflow.Logger(*name: str = None, level: str = None, _format: str = '%(asctime)s - %(levelname)-8s - %(name)-12s: %(message)s', colored_log: bool = False, logfile_path: str | Path = None*)

Bases: object

Improved logging

Initialize Logger

Parameters

- **name** (*str*) – Name of the Logger object default : None
- **level** (*str*) – Desired Logging level default : 'INFO'
- **_format** (*str*) – Desired Logging Format default : '%(asctime)s - %(levelname)-8s - %(name)-12s: %(message)s'
- **colored_log** (*bool*) – Use colored logging for stdout default: False
- **logfile_path** (*str or Path*) – Path for logging to a file default : None

classmethod add_file_handler(*logfile_path: str | Path, level: str = 'INFO', _format: str = '%(asctime)s - %(levelname)-8s - %(name)-12s: %(message)s'*)

Create file handler. This classmethod will allow setting custom file handler on children Create stream handler This classmethod will allow setting a custom stream handler on children

Parameters

- **logfile_path** (*str or Path*) – Path for writing out logfiles from logging default : False
- **level** (*str*) – logging level default : 'INFO'
- **_format** (*str*) – logging format default : '%(asctime)s - %(levelname)-8s - %(name)-12s: %(message)s'

Returns
handler – file handler of a logging object

Return type
logging.Handler

classmethod add_handlers(*logger: Logger, handlers: List[Handler]*)

Add a list of handlers to a logger

Parameters

- **logger** (*logging.Logger*) – Logger object to add a new handler to
- **handlers** (*list*) – A list of handlers to be added to the logger object

Returns
logger

Return type
Logger object

classmethod add_stream_handler(*level: str = 'INFO', _format: str = '%(asctime)s - %(levelname)-8s - %(name)-12s: %(message)s', colored_log: bool = False*)

Create stream handler This classmethod will allow setting a custom stream handler on children

Parameters

- **level** (*str*) – logging level default : 'INFO'

- **_format** (*str*) – logging format default : '%(asctime)s - %(levelname)-8s - %(name)-12s: %(message)s'
- **colored_log** (*bool*) – enable colored output for stdout default : False

Returns
handler – stream handler of a logging object

Return type
logging.Handler

`get_logger()`

Return the logging object

Returns
logger

Return type
Logger object

`wxflow.logit(logger, name=None, message=None)`

Logger decorator to add logging to a function. Simply add: @logit(logger) before any function :Parameters: * **logger** (*Logger*) – Logger object

- **name** (*str*) – Name of the module to be logged default: __module__
- **message** (*str*) – Name of the function to be logged default: __name__

`class wxflow.Hsi(quiet: bool = True, echo_commands: bool = True, opts: str | List = [])`

Class offering an interface to HPSS via the hsi utility.

Examples:

```
>>> from wxflow import Hsi
>>> hsi = Hsi()  # Generates an Executable object of "hsi"
>>> output = hsi.put("some_local_file", "/HPSS/path/to/some_file") # Put a file␣
↪onto HPSS
>>> output = hsi.ls("/HPSS/path/to/some_file") # List the file
>>> output = hsi.chgrp("rstprod", "/HPSS/pth/to/some_file") # Change the group to␣
↪rstprod
```

Instantiate the hsi command

Parameters:

quiet
[bool] Run hsi in quiet mode (suppress login information, transfer info, etc)

echo_commands
[bool] Echo each command. Some commands will still not echo (e.g. chmod).

opts
[str | list] Additional arguments to send to each hsi command.

`chgrp(group_name: str, target: str, hsi_opts: str = '', chgrp_opts: str = '') → str`

Method to change the group of a file or directory on HPSS

Parameters

- **group_name** (*str*) – The group to which ownership of the file/directory is to be set.
- **target** (*str*) – Full path of the target location of the file on HPSS.
- **hsi_opts** (*str*) – String of options to send to hsi.
- **chgrp_opts** (*str*) – Options to send to chgrp. See "hsi chgrp -?" for more details.

Returns
output – Concatenated output and error of the hsi chgrp command.

Return type
str

`chmod`(*mod: str*, *target: str*, *hsi_opts: List | str = ''*, *chmod_opts: List | str = ''*) → str

Method to change the permissions of a file or directory on HPSS

Parameters

- **mod** (*str*) – Permissions to set for the file or directory, e.g. "640", "o+r", etc.
- **target** (*str*) – Full path of the target location of the file on HPSS.
- **hsi_opts** (*list | str*) – Options to send to hsi.
- **chmod_opts** (*list | str*) – Options to send to chmod. See "hsi chmod -?" for more details.

Returns
output – Concatenated output and error of the hsi chmod command.

Return type
str

`exists`(*target: str*) → bool

Method to test the existence of a file/directory/glob on HPSS

Parameters
target (*str*) – Full path of the target location on HPSS.

Returns
pattern_exists – True if the target exists on HPSS.

Return type
bool

`get`(*source: str*, *target=None*, *opts: List | str = []*) → str

Method to get a file from HPSS via hsi

Parameters

- **source** (*str*) – Full path location on HPSS of the file
- **target** (*str*) – Location on the local machine to place the file. If not specified, then the file will be placed in the current directory.
- **opts** (*str | list*) – List or string of additional options to send to hsi command.

Returns
output – Concatenated output and error of the hsi get command.

Return type
str

ls(*target: str*, *hsi_opts: str = ''*, *ls_opts: str = ''*, *ignore_missing: bool = False*) → str

Method to list files/directories on HPSS via hsi

Parameters

- **target** (*str*) – Full path of the target location on HPSS.
- **hsi_opts** (*str*) – String of options to send to hsi.
- **ls_opts** (*str*) – Options to send to ls. See "hsi ls -?" for more details.
- **ignore_missing** (*bool*) – Flag to ignore missing files

Returns

output – Concatenated output and error of the hsi ls command.

Return type

str

mkdir(*target: str*, *hsi_opts: str = ''*, *mkdir_opts: str = ''*) → str

Method to delete a file or directory on HPSS via hsi

Parameters

- **target** (*str*) – Full path of the target location of the file on HPSS.
- **hsi_opts** (*str*) – String of options to send to hsi.

Returns

output – Concatenated output and error of the hsi mkdir command.

Return type

str

put(*source: str*, *target: str*, *opts: List | str = []*, *listing_file: str = None*) → str

Method to put a file onto HPSS via hsi

Parameters

- **source** (*str*) – Location on the local machine of the source file to send to HPSS.
- **target** (*str*) – Full path of the target location of the file on HPSS.
- **opts** (*str | List*) – List or string of additional options to send to hsi.

Returns

output – Concatenated output and error of the hsi put command.

Return type

str

rm(*target: str*, *recursive: bool = False*, *hsi_opts: str = ''*, *rm_opts: str = ''*) → str

Method to delete a file or directory on HPSS via hsi

Parameters

- **target** (*str*) – Full path of the target location of the file on HPSS.
- **hsi_opts** (*str*) – String of options to send to hsi.
- **rm_opts** (*str*) – Options to send to rm. See "hsi rm -?" for more details.
- **recursive** (*bool*) – Flag to indicate a call to rmdir.

Returns

output – Concatenated output and error of the hsi rm command.

Return type
str

rmdir(*target: str*, *hsi_opts: str = ''*, *rmdir_opts: str = ''*) → str

Method to delete an empty directory on HPSS via hsi

Parameters

- **target** (*str*) – Full path of the target location of the file on HPSS.
- **hsi_opts** (*str*) – String of options to send to hsi.
- **rmdir_opts** (*str*) – Options to send to rmdir. See "hsi rmdir -?" for more details.

Returns
output – Concatenated output and error of the hsi rmdir command.

Return type
str

class wxflow.**Htar**

Class offering an interface to HPSS via the htar utility.

Examples:

```
>>> from wxflow import Htar
>>> htar = Htar()  # Generates an Executable object of "htar"
>>> output = htar.cvf("/HPSS/path/to/archive.tar", "file1 file2") # Create an HPSS␣
↪archive from two local files
>>> output = htar.tell("/HPSS/path/to/archive.tar") # List the contents of an␣
↪archive
```

create(*tarball: str*, *fileset: List | str*, *dereference: bool = False*, *opts: List | str = '-P'*) → str

Method to write an archive to HPSS

Parameters

- **opts** (*str | list*) – Options to send to htar. By default, "-P" (create parent directories).
- **tarball** (*str*) – Full path location on HPSS to create the archive.
- **fileset** (*list | str*) – List containing filenames, patterns, or directories to archive
- **dereference** (*bool*) – Whether to dereference symbolic links (archive the pointed-to files instead).

Returns
output – Concatenated output and error of the htar command.

Return type
str

cvf(*tarball: str*, *fileset: List | str*, *dereference: bool = False*) → str

Method to write an archive to HPSS verbosely (without options).

Parameters

- **tarball** (*str*) – Full path location on HPSS to create the archive.
- **fileset** (*list | str*) – List containing filenames, patterns, or directories to archive

- **dereference** (*bool*) – Whether to dereference symbolic links (archive the pointed-to files instead).

Returns
output – Concatenated output and error from the htar command

Return type
str

extract(*tarball: str*, *fileset: List | str = []*, *opts: List | str = ''*) → str

Method to extract an archive from HPSS via htar

Parameters

- **opts** (*str*) – String of options to send to htar.
- **tarball** (*str*) – Full path location of an archive on HPSS to extract from.
- **fileset** (*list | str*) – Filenames, patterns, or directories to extract from the archive. If empty, then all files will be extracted.

Returns
output – Concatenated output and error from the htar command

Return type
str

tell(*tarball: str*, *opts: List | str = ''*, *fileset: List | str = []*) → str

Method to list the contents of an archive on HPSS

Parameters

- **opts** (*str*) – String of options to send to htar.
- **tarball** (*str*) – Full path location on HPSS to list the contents of.
- **fileset** (*list | str*) – Filenames, patterns, or directories to list from the archive. If empty, then all files will be listed.

Returns
output – Concatenated output and error from the htar command

Return type
str

xvf(*tarball: str = ''*, *fileset: list = []*) → str

Method to extract an archive from HPSS verbosely (without options).

Parameters

- **tarball** (*str*) – Full path location of an archive on HPSS to extract from.
- **fileset** (*list*) – List containing filenames, patterns, or directories to extract from the archive. If empty, then all files will be extracted.

Returns
output – Concatenated output and error from the htar command

Return type
str

PYTHON MODULE INDEX

W

INDEX

X

Y