pytil Documentation

Release 7.0.1.dev

Tim Diels

Jan 08, 2018

Contents

pytil (formerly known as Chicken Turtle Util) is a Python utility library.

The API reference starts with an overview of all the features and then gets down to the nitty gritty details of each of them. Most of the reference provides examples. For a full overview of features see the module contents overview of the API reference and the table of contents of the user guide (in the sidebar) as they are complementary.

Dependencies are grouped by module. For example, when using `pytil.data_frame`, you should `pip install 'pytil[data_frame]'`. To install dependencies of all modules, use `pip install 'pytil[all]'`. If you are not familiar with pip, see pip's quickstart guide.

While all features are documented and tested, the API is changed frequently. When doing so, the major version is bumped and a changelog is kept to help upgrade. Fixes will not be backported. It is recommended to pin the major version in your setup.py, e.g. for 2.x.y:

```
install_requires = ['pytil.*', ...]
```

Contents:

CHAPTER 1

API reference

See modules for a short description of each modules. For a full listing of the contents of all modules, see the module contents overview.

1.1 Modules

algorithms	
click	
configuration	
data_frame	
debug	
dict	
difflib	Module difflib – helpers for computing deltas between objects.
exceptions	Exception classes and utilities.
hashlib	hashlib extensions.
iterable	
logging	Logging utilities.
multi_dict	Multi-dict class, a dict which maps keys to one or more values.
numpy	numpy extensions.
observable	Observable collections.
parse	File parsers
path	pathlib extensions.
pkg_resources	pkg_resources extensions.
series	
set	Set utilities.
sqlalchemy	
test	

Continued on next page

Table 1.1 – continued from previous page

various	Various utilities.
write	File writers

1.2 Module content overview

algorithms

multi_way_partitioning
toset_from_tosets

click

option
password_option

configuration

ConfigurationLoader

data_frame

assert_equals
equals
replace_na_with_none
split_array_like

debug

pretty_memory_info

dict

DefaultDict
pretty_print_head
invert

difflib

line_diff

exceptions

InvalidOperationError	Operation is illegal/invalid in the current state.
UserException	Exceptional user input/action.
exc_info	Get exc_info tuple from exception.

hashlib

base85_digest	Get base 85 encoded digest of hash.

iterable

is_sorted	

logging

configure	Configure root logger to log INFO to stderr and DEBUG to log file.
set_level	Temporarily change log level of logger.

multi_dict

MultiDict	A multi-dict, a dict which maps keys to one or more values.

numpy

ArrayLike	Type representing any value that can be passed to numpy.array to create an array.

observable

Set	Observable set

parse

csv	Parse CSV file.
tsv	

path

TemporaryDirectory	An extension to `tempfile.TemporaryDirectory`.
chmod	Change file mode bits.
hash	Hash file or directory.
is_descendant	Get whether path is descendant of other path.
is_descendant_or_self	Get whether path is descendant of other path or is equivalent to it.
remove	Remove file or directory (recursively), if it exists.
sorted_lines	Lines of file, sorted.

pkg_resources

resource_copy	Copy file/dir resource to destination.
resource_path	Like `resource_filename` but return a `Path` instead.

series

assert_equals
equals
invert
split

set

merge_by_overlap	Of a list of sets, merge those that overlap, in place.

sqlalchemy

log_sql
pretty_sql

test

assert_dir_equals
assert_dir_unchanged
assert_file_equals
assert_file_mode
assert_lines_equal
assert_matches
assert_search_matches
assert_text_contains
assert_text_equals
assert_xlsx_equals
assert_xml_equals
reset_loggers
temp_dir_cwd

various

join_multiline	Join multiline text into a single line.

write

csv	Write CSV file.
tsv	

1.3 pytil.algorithms

1.4 pytil.click

1.5 pytil.configuration

1.6 pytil.data_frame

1.7 pytil.debug

1.8 pytil.dict

1.9 pytil.difflib

1.10 pytil.exceptions

Exception classes and utilities.

exception pytil.exceptions.**InvalidOperationError**
Bases: Exception

Operation is illegal/invalid in the current state.

If an invalid argument was given, use ValueError instead. An operation can be a method/function call or getting/setting an attribute.

exception pytil.exceptions.**UserException**
Bases: Exception

Exceptional user input/action.

pytil.exceptions.**exc_info**(*exception*)
Get exc_info tuple from exception.

See also:

sys.exc_info()

1.11 pytil.hashlib

`hashlib` extensions.

`pytil.hashlib.`**`base85_digest`**(*hash_*)
Get base 85 encoded digest of hash.

Parameters **`hash`** (*`hash object`*) – E.g. as returned by `hashlib.sha512()`.

Returns Base 85 encoded digest.

Return type str

1.12 pytil.iterable

1.13 pytil.logging

Logging utilities.

`pytil.logging.`**`configure`**(*log_file*)
Configure root logger to log INFO to stderr and DEBUG to log file.

The log file is appended to. Stderr uses a terse format, while the log file uses a verbose unambiguous format.

Root level is set to INFO.

Parameters **`log_file`** (*`Path`*) – File to log to.

Returns Stderr and file handler respectively.

Return type Tuple[StreamHandler, FileHandler]

`pytil.logging.`**`set_level`**(*logger*, *level*)
Temporarily change log level of logger.

Parameters

- **`logger`** (*`str or Logger`*) – Logger name or logger whose log level to change.
- **`level`** (*`int`*) – Log level to set.

Examples

```
>>> with set_level('sqlalchemy.engine', logging.INFO):
...     pass  # sqlalchemy log level is set to INFO in this block
```

1.14 pytil.multi_dict

Multi-dict class, a dict which maps keys to one or more values.

`class` `pytil.multi_dict.`**`MultiDict`**(*dict_*)
Bases: `object`

A multi-dict, a dict which maps keys to one or more values.

> **Warning:** This is very much a work in progress.

Parameters **dict** (*Dict[Hashable, Set[Hashable]]*) – Dict to create a multi-dict view of. No copy is made. Editing the multi-dict, edits the underlying dict. Changes to the underlying dict, affect the multi-dict.

Notes

A multi-dict (or multi map) is a dict that maps each key to one or more values.

Multi-dicts provided by other libraries tend to be more feature rich, while this interface is far more conservative. Instead of wrapping, they provide an interface that mixes regular and multi-dict access. Additionally, other multi-dicts map keys to lists of values, allowing a key to map to the same value multiple times.

dict
Get the underlying dict.

Returns The underlying dict.

Return type Dict[Hashable, Set[Hashable]]

invert()
Invert by swapping each value with its key.

Returns Inverted multi-dict.

Return type *MultiDict*

Examples

```
>>> MultiDict({1: {1}, 2: {1,2,3}}, 4: {}).invert()
MultiDict({1: {1,2}, 2: {2}, 3: {2}})
```

1.15 pytil.numpy

`numpy` extensions.

class pytil.numpy.**ArrayLike**
Bases: `typing.Generic`

Type representing any value that can be passed to `numpy.array` to create an array.

`ArrayLike[T]` is an array-like with data type `T`.

1.16 pytil.observable

Observable collections.

class pytil.observable.**Set**(**args*, ***kwargs*)
Bases: `set`

Observable set

change_listeners
Get change listeners.

Each change listener is called immediately after a mutating operation that actually changed the set. E.g. redundant additions are ignored.

Returns List of change listeners. Each change listener takes 2 arguments: the items that were added, and the items that were removed. Note: Items can be added and removed from a set in a single operation. When a listener raises, the change is rolled back without further notification.

Return type List[Callable[[FrozenSet, FrozenSet], None]]

1.17 pytil.parse

File parsers

class pytil.parse.**csv**(*file*, **args*, ***kwargs*)
Bases: object

Parse CSV file.

Parameters

- **file** (*Path*) –
- ***args** – csv.DictReader args (except the f arg)
- ****kwargs** – csv.DictReader args

Examples

for row in parse.csv(file): print(row['column'])

with parse.csv(file) as reader:

for row in reader: pass

1.18 pytil.path

pathlib extensions.

pytil.path.**TemporaryDirectory**(*suffix=None*, *prefix=None*, *dir=None*, *on_error='ignore'*)
An extension to tempfile.TemporaryDirectory.

Unlike with tempfile, a Path is yielded on __enter__, not a str.

Parameters

- **suffix** (*str*) – See tempfile.TemporaryDirectory.
- **prefix** (*str*) – See tempfile.TemporaryDirectory.
- **dir** (*Path*) – See tempfile.TemporaryDirectory, but pass a Path instead.
- **on_error** (*str*) – Handling of failure to delete directory (happens frequently on NFS), one of:

raise Raise exception on failure.

ignore Fail silently.

pytil.path.**chmod**(*path*, *mode*, *operator='='*, *recursive=False*)

Change file mode bits.

When recursively chmodding a directory, executable bits in `mode` are ignored when applying to a regular file. E.g. `chmod(path, mode=0o777, recursive=True)` would apply `mode=0o666` to regular files.

Symlinks are ignored.

Parameters

- **path** (*Path*) – Path to chmod.
- **mode** (*int*) – Mode bits to apply, e.g. `0o777`.
- **operator** (*str*) – How to apply the mode bits to the file, one of:

  '=' Replace mode with given mode.

  '+' Add to current mode.

  '-' Subtract from current mode.
- **recursive** (*bool*) – Whether to chmod recursively.

pytil.path.**hash**(*path*, *hash_function=<built-in function openssl_sha512>*)

Hash file or directory.

Parameters

- **path** (*Path*) – File or directory to hash.
- **hash_function** (*Callable[[], hash object]*) – Function which creates a hashlib hash object when called. Defaults to `hashlib.sha512`.

Returns hashlib hash object of file/directory contents. File/directory stat data is ignored. The directory digest covers file/directory contents and their location relative to the directory being digested. The directory name itself is ignored.

Return type hash object

pytil.path.**is_descendant**(*descendant*, *ancestor*)

Get whether path is descendant of other path.

Uses the absolute path, so symlinks, ... do not affect this.

Parameters

- **descendant** (*Path*) – Supposed descendant.
- **ancestor** (*Path*) – Supposed ancestor.

Returns Whether `descendant` is indeed a descendant of `ancestor`.

Return type bool

See also:

is_descendant_or_self() Get whether path is descendant of other path or is equivalent to it

Examples

```
>>> is_descendant(Path('a'), Path('a'))
False
>>> is_descendant(Path('a/b'), Path('a'))
True
>>> is_descendant(Path('a'), Path('a/b'))
False
>>> is_descendant(Path('a'), Path('a/..'))
False
```

pytil.path.**is_descendant_or_self**(*descendant*, *ancestor*)

Get whether path is descendant of other path or is equivalent to it.

Uses the absolute path, so symlinks, ... do not affect this.

Parameters

- **descendant** (*Path*) – Supposed descendant.
- **ancestor** (*Path*) – Supposed ancestor.

Returns Whether `descendant` is indeed a descendant of `ancestor` or they are equivalent (equal after path normalisation).

Return type bool

See also:

is_descendant() Get whether path is descendant of other path

Examples

```
>>> is_descendant_or_self(Path('a'), Path('a'))
True
>>> is_descendant_or_self(Path('a/b'), Path('a'))
True
>>> is_descendant_or_self(Path('a'), Path('a/b'))
False
>>> is_descendant_or_self(Path('a'), Path('a/..'))
False
```

pytil.path.**remove**(*path*, *force=False*)

Remove file or directory (recursively), if it exists.

On NFS file systems, if a directory contains `.nfs*` temporary files (sometimes created when deleting a file), it waits for them to go away.

Parameters

- **path** (*Path*) – Path to remove.
- **force** (*bool*) – If True, will remove files and directories even if they are read-only (as if first doing `chmod -R +w`).

pytil.path.**sorted_lines**(*file*)

Lines of file, sorted.

Parameters **file** (*Path*) – Path to file whose lines to read.

Returns Sorted lines of file.

Return type List[str]

1.19 pytil.pkg_resources

`pkg_resources` extensions.

`pytil.pkg_resources.`**`resource_copy`**(*package_or_requirement*, *resource_name*, *destination*)
Copy file/dir resource to destination.

Parameters

- **`package_or_requirement`** (*`str`*) –
- **`resource_name`** (*`str`*) –
- **`destination`** (*`Path`*) – Path to copy to, it must not exist.

`pytil.pkg_resources.`**`resource_path`**(*package_or_requirement*, *resource_name*)
Like `resource_filename` but return a `Path` instead.

Parameters

- **`package_or_requirement`** (*`str`*) –
- **`resource_name`** (*`str`*) –

Returns Path to resource.

Return type Path

1.20 pytil.series

1.21 pytil.set

Set utilities.

`pytil.set.`**`merge_by_overlap`**(*sets*)
Of a list of sets, merge those that overlap, in place.

The result isn't necessarily a subsequence of the original `sets`.

Parameters **`sets`** (*`Sequence[Set[Any]]`*) – Sets of which to merge those that overlap. Empty sets are ignored.

Notes

Implementation is based on this StackOverflow answer. It outperforms all other algorithms in the thread (visited at dec 2015) on python3.4 using a wide range of inputs.

Examples

```
>>> merge_by_overlap([{1,2}, set(), {2,3}, {4,5,6}, {6,7}])
[{1,2,3}, {4,5,6,7}]
```

1.22 pytil.sqlalchemy

1.23 pytil.test

1.24 pytil.various

Various utilities.

pytil.various.**join_multiline**(*text*)
Join multiline text into a single line.

1.25 pytil.write

File writers

pytil.write.**csv**(*file*, **args*, ***kwargs*)
Write CSV file.

Parameters

- **file** (*Path*) –
- ***args** – csv.DictWriter args (except the f arg)
- ****kwargs** – csv.DictWriter args

Examples

with write.csv(file) as writer: writer.writerow((1,2,3))

CHAPTER 2

Developer documentation

Documentation for developers/contributors of pytil.

The project follows a simple project structure and associated workflow. Please read its documentation.

2.1 Project decisions

2.1.1 API design

If it's a path, expect a `Path`, not a `str`.

If extending a module from another project, e.g. `pandas`, use the same name as the module.

If a module is a collection of instances of something, give it a plural name, else make it singular. E.g. `exceptions` for a collection of `Exception` classes, but `function` for a set of related functions operating on functions.

2.1.2 API implementation

When importing things, they also are exported, we do not attempt to remedy this. E.g. `import numpy as np` in `module.py` can be accessed with `module.np`, but `help` or Sphinx documentation will not include them. We assume a user will not use anything not mentioned in the documentation.

CHAPTER 3

Changelog

Semantic versioning is used (starting with v3.0.0).

3.1 7.0.0

- Backwards incompatible changes:
  - Remove `path.tsv_lines`: use `parse.tsv` instead.
  - Remove `algorithms.spread_points_in_hypercube`: it simply returned opints on a grid, which can be achieved with numpy.meshgrid, e.g. 3D grid:

```
import numpy as np
side = np.linspace(0, 1, ceil(n**(1/3)))
points = np.array(np.meshgrid(side, side, side)).reshape(3,-1).T
```

  - Rename `path.assert_mode` to `test.assert_file_mode`
  - Rename `path.assert_equals` to `test.assert_file_equals`
- Enhancements/additions:
  - Add `test.assert_dir_equals`
  - Add `various.join_multiline`
  - Add `test.assert_xlsx_equals`
  - Add `parse.csv`
  - Add `parse.tsv`
  - Add `write.csv`
  - Add `write.tsv`

3.2 6.0.0

- Backwards incompatible changes:
  - Remove `asyncio.stubborn_gather`: More often than not, when you need this, you should look into a full blown pipeline framework such as Nextflow instead.
  - Remove `click.assert_runs`: It is usually simpler to use pytest's isolation and output capturing than to use this function
  - Remove `click.argument`: Click's arguments are required by default, you can simply use the real click.argument directly.
  - Remove `dict.assign`: Esoteric and easily replaced by: `destination.clear(); destination.update(source)`.
  - Remove `function.compose`: Compose can be found in other PyPI packages, e.g. in Toolz: `toolz.functoolz.compose`.
  - Remove `http.download`: `urllib.request.urlretrieve` can be used instead, though the filename suggested by the server is not used. Only the extension of the downloaded file will match that of the server.
  - Remove `inspect.call_args`: Esoteric, can achieve something very similar with:

```
args = inspect.signature(f).bind_partial(*args, **kwargs)
args.apply_defaults()
dict(args.arguments)
```

  - Remove `iterable.sliding_window`: Use `more_itertools.windowed` instead. Drop in replacement.
  - Remove `iterable.partition`: If your data is sorted by key, you can just use `itertools.groupby` as drop in replacement. Else, you can use `toolz.itertoolz.groupby`, but arg order is swapped and element order may not be preserved.
  - Remove `iterable.flatten`: Use `more_itertools.collapse` instead. `flatten(a, b)` becomes `collapse(a, levels=b)`.
  - Remove `path.write`: Use `pathlib.Path.write_text` instead. However, you are now responsible for creating any missing ancestor directories (`os.makedirs`). The use of mode can be replaced by `p.touch(); p.chmod(mode); p.write_text()` or a variation depending on your use case.
  - Remove `path.read`: Use `pathlib.Path.read_text` instead.
  - Remove `pymysql.patch`: Instead of globally patching it, use the `conv` argument when creating a pymysql Connection.
  - `algorithms.multi_way_partitioning` now returns a frozenbag instead of a bag.
  - *`multi_dict.MultiDict.invert`* now returns a MultiDict instead of a dict.
- Enhancements/additions:
  - Add `difflib.line_diff`
  - Add *`numpy.ArrayLike`*
  - Add *`path.TemporaryDirectory`*
  - Add *`path.is_descendant`*
  - Add *`path.is_descendant_or_self`*

  - Add *path.sorted_lines*
  - Add path.tsv_lines
  - Add *pkg_resources.resource_copy*
  - Add test.assert_dir_unchanged
  - Add test.assert_lines_equal
  - Add test.assert_xml_equals
  - Add test.reset_loggers
  - test.assert_text_equals: Show diff when not equal
- Fixes:
  - Fix package: Add missing data files and dependencies
  - Fix formatting of test.assert_matches, test.assert_search_matches: forgot newline after Actual:

3.3 5.0.0

Major backwards incompatible change: Renamed root package, pypi name and project to pytil.

3.4 4.1.2

Announce rename to pytil.

3.5 4.1.1

- Fixes:
  - add missing keys to extras_require: hashlib, multi_dict, test

3.6 4.1.0

- Backwards incompatible changes: None
- Enhancements/additions:
  - click.assert_runs: pass on extra args to click's invoke()
  - path.chmod, path.remove: ignore disappearing children instead of raising
  - Add exceptions.exc_info: exc_info tuple as seen in function parameters in the traceback standard module
  - Add extras_require['all'] to setup.py: union of all extra dependencies
- Fixes:
  - path.chmod: do not follow symlinks
  - iterable.flatten: removed debug prints: +, -

- Internal / implementation details:
  - use simple project structure instead of Chicken Turtle Project
  - `pytest-catchlog` instead of `pytest-capturelog`
  - `extras_require['dev']`: test dependencies were missing
  - `test_http` created `existing_file` in working dir instead of in test dir

3.7 v4.0.1

- Fixed: README formatting error

3.8 v4.0.0

- Major:
  - `path.digest` renamed to `path.hash` (and added `hash_function` parameter)
  - renamed `cli` to `click`
  - require Python 3.5 or newer
  - Changed: `asyncio.stubborn_gather`:
    - raise `CancelledError` if all its awaitables raised `CancelledError`.
    - raise summary exception if any awaitable raises exception other than `CancelledError`
    - log exceptions, as soon as they are raised
- Minor:
  - Added:
    - `click.assert_runs`
    - `hashlib.base85_digest`
    - `logging.configure`
    - `path.assert_equals`
    - `path.assert_mode`
    - `test.assert_matches`
    - `test.assert_search_matches`
    - `test.assert_text_contains`
    - `test.assert_text_equals`
- Fixes:
  - `path.remove`: raised when `path.is_symlink()` or contains a symlink
  - `path.digest/hash`: directory hash collisions were more likely than necessary
  - `pymysql.patch`: change was not picked up in recent pymysql versions

3.9 v3.0.1

- Fixed: README formatting error

3.10 v3.0.0

- Removed:
  - `cli.Context`, `cli.BasicsMixin`, `cli.DatabaseMixin`, `cli.OutputDirectoryMixin`
  - `pyqt` module
  - `URL_MAX_LENGTH`
  - `various` module: `Object`, `PATH_MAX_LENGTH`
- Enhanced:
  - `data_frame.split_array_like`: `columns` defaults to `df.columns`
  - `sqlalchemy.pretty_sql`: much better formatting
- Added:
  - `algorithms.toset_from_tosets`: Create totally ordered set (toset) from tosets
  - `configuration.ConfigurationLoader`: loads a single configuration from one or more files directory according to XDG standards
  - `data_frame.assert_equals`: Assert 2 data frames are equal
  - `data_frame.equals`: Get whether 2 data frames are equal
  - `dict.assign`: assign one dict to the other through mutations
  - `exceptions.InvalidOperationError`: raise when an operation is illegal/invalid, regardless of the arguments you throw at it (in the current state).
  - `inspect.call_args`: Get function call arguments as a single dict
  - `observable.Set`: set which can be observed for changes
  - `path.chmod`: change file or directory mode bits (optionally recursively)
  - `path.digest`: Get SHA512 checksum of file or directory
  - `path.read`: get file contents
  - `path.remove`: remove file or directory (recursively), unless it's missing
  - `path.write`: create or overwrite file with contents
  - `series.assert_equals`: Assert 2 series are equal
  - `series.equals`: Get whether 2 series are equal
  - `series.split`: Split values
  - `test.temp_dir_cwd`: pytest fixture that sets current working directory to a temporary directory

3.11 v2.0.4

No changelog

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index