cawdrey

Release 0.1.5

Dominic Davis-Foster

Jun 03, 2020

CONTENTS

Several useful custom dictionaries

Docs	
Tests	
PyPI	
Anaconda	
Other	

CHAPTER

ONE

INSTALLATION

from PyPI

```
$ pip install cawdrey
```

from Anaconda

First add the required channels

```
$ conda config --add channels http://conda.anaconda.org/domdfcoding
$ conda config --add channels http://conda.anaconda.org/conda-forge
```

Then install

```
$ conda install cawdrey
```

from GitHub

```
$ pip install git+https://github.com//cawdrey@master
```

1.1 AlphaDict

1.1.1 About

1.1.2 Usage

1.1.3 API Reference

Provides AlphaDict, a frozen OrderedDict where the keys are stored alphabetically.

class `cawdrey.alphadict.`**`AlphaDict`**`(`*seq=None*, ***kwargs*`)`

`cawdrey.alphadict.`**`alphabetical_dict`**`(`***kwargs*`)`

1.2 bdict

1.2.1 About

1.2.2 Usage

1.2.3 API Reference

class `cawdrey.bdict.`**`bdict`**(*seq=None*, ***kwargs*)

Returns a new dictionary initialized from an optional positional argument and a possibly empty set of keyword arguments.

Each key:value pair is entered into the dictionary in both directions, so you can perform lookups with either the key or the value.

If no positional argument is given, an empty dictionary is created. If a positional argument is given and it is a mapping object, a dictionary is created with the same key-value pairs as the mapping object. Otherwise, the positional argument must be an iterable object. Each item in the iterable must itself be an iterable with exactly two objects. The first object of each item becomes a key in the new dictionary, and the second object the corresponding value.

If keyword arguments are given, the keyword arguments and their values are added to the dictionary created from the positional argument.

If an attempt is made to add a key or value that already exists in the dictionary a ValueError will be raised

Keys or values of `None`, `True` and `False` will be stored internally as `"_None"`, `"_True"` and `"_False"` respectively

Based on https://stackoverflow.com/a/1063393 by https://stackoverflow.com/users/9493/brian

Improved May 2020 suggestions from https://treyhunner.com/2019/04/why-you-shouldnt-inherit-from-list-and-dict-in-python/

1.3 frozendict

1.3.1 About

`frozendict` is an immutable wrapper around dictionaries that implements the complete mapping interface. It can be used as a drop-in replacement for dictionaries where immutability is desired.

Of course, this is `python`, and you can still poke around the object's internals if you want.

The *`frozendict`* constructor mimics `dict`, and all of the expected interfaces (`iter`, `len`, `repr`, `hash`, `getitem`) are provided. Note that a *`frozendict`* does not guarantee the immutability of its values, so the utility of `hash` method is restricted by usage.

The only difference is that the `copy()` method of *`frozendict`* takes variable keyword arguments, which will be present as key/value pairs in the new, immutable copy.

1.3.2 Usage

```
>>> from frozendict import frozendict
>>>
>>> fd = frozendict({ 'hello': 'World' })
>>>
>>> print fd
<frozendict {'hello': 'World'}>
>>>
>>> print fd['hello']
'World'
>>>
>>> print fd.copy(another='key/value')
<frozendict {'hello': 'World', 'another': 'key/value'}>
>>>
```

In addition, `frozendict` supports the + and - operands. If you add a *dict*-like object, a new `frozendict` will be returned, equal to the old `frozendict` updated with the other object. Example:

```
>>> frozendict({"Sulla": "Marco", 2: 3}) + {"Sulla": "Marò", 4: 7}
<frozendict {'Sulla': 'Marò', 2: 3, 4: 7}>
>>>
```

You can also subtract an iterable from a `frozendict`. A new `frozendict` will be returned, without the keys that are in the iterable. Examples:

```
>>> frozendict({"Sulla": "Marco", 2: 3}) - {"Sulla": "Marò", 4: 7}
<frozendict {'Sulla': 'Marco', 2: 3}>
>>> frozendict({"Sulla": "Marco", 2: 3}) - [2, 4]
<frozendict {'Sulla': 'Marco'}>
>>>
```

Some other examples:

```
>>> from frozendict import frozendict
>>> fd = frozendict({"Sulla": "Marco", "Hicks": "Bill"})
>>> print(fd)
<frozendict {'Sulla': 'Marco', 'Hicks': 'Bill'}>
>>> print(fd["Sulla"])
Marco
>>> fd["Bim"]
KeyError: 'Bim'
>>> len(fd)
2
>>> "Sulla" in fd
True
>>> "Sulla" not in fd
False
>>> "Bim" in fd
False
>>> hash(fd)
835910019049608535
>>> fd_unhashable = frozendict({1: []})
>>> hash(fd_unhashable)
TypeError: unhashable type: 'list'
>>> fd2 = frozendict({"Hicks": "Bill", "Sulla": "Marco"})
>>> print(fd2)
```

(continues on next page)

(continued from previous page)

```
<frozendict {'Hicks': 'Bill', 'Sulla': 'Marco'}>
>>> fd2 is fd
False
>>> fd2 == fd
True
>>> frozendict()
<frozendict {}>
>>> frozendict(Sulla="Marco", Hicks="Bill")
<frozendict {'Sulla': 'Marco', 'Hicks': 'Bill'}>
>>> frozendict((("Sulla", "Marco"), ("Hicks", "Bill")))
<frozendict {'Sulla': 'Marco', 'Hicks': 'Bill'}>
>>> fd.get("Sulla")
'Marco'
>>> print(fd.get("God"))
None
>>> tuple(fd.keys())
('Sulla', 'Hicks')
>>> tuple(fd.values())
('Marco', 'Bill')
>>> tuple(fd.items())
(('Sulla', 'Marco'), ('Hicks', 'Bill'))
>>> iter(fd)
<dict_keyiterator object at 0x7feb75c49188>
>>> frozendict.fromkeys(["Marco", "Giulia"], "Sulla")
<frozendict {'Marco': 'Sulla', 'Giulia': 'Sulla'}>
>>> fd["Sulla"] = "Silla"
TypeError: 'frozendict' object does not support item assignment
>>> del fd["Sulla"]
TypeError: 'frozendict' object does not support item deletion
>>> fd.clear()
AttributeError: 'frozendict' object has no attribute 'clear'
>>> fd.pop("Sulla")
AttributeError: 'frozendict' object has no attribute 'pop'
>>> fd.popitem()
AttributeError: 'frozendict' object has no attribute 'popitem'
>>> fd.setdefault("Sulla")
AttributeError: 'frozendict' object has no attribute 'setdefault'
>>> fd.update({"Bim": "James May"})
AttributeError: 'frozendict' object has no attribute 'update'
```

1.3.3 API Reference

class `cawdrey.frozendict.`**`frozendict`**(**args*, ***kwargs*)

An immutable wrapper around dictionaries that implements the complete `collections.Mapping` interface. It can be used as a drop-in replacement for dictionaries where immutability is desired.

`copy`(***add_or_replace*)

`dict_cls`

alias of `builtins.dict`

`sorted`(**args*, *by='keys'*, ***kwargs*)

Return a new `frozendict`, with the element insertion sorted. The signature is the same as the builtin `sorted` function, except for the additional parameter `by`, that is `"keys"` by default and can also be `"values"` and `"items"`. So the resulting *frozendict* can be sorted by keys, values or items.

If you want more complicated sorts read the documentation of `sorted`.

The the parameters passed to the `key` function are the keys of the `frozendict` if `by = "keys"`, and are the items otherwise.

Note: Sorting by keys and items achieves the same effect. The only difference is when you want to customize the sorting passing a custom `key` function. You *could* achieve the same result using `by = "values"`, since also sorting by values passes the items to the key function. But this is an implementation detail and you should not rely on it.

1.3.4 Copyright

Based on https://github.com/slezica/python-frozendict and https://github.com/mredolatti/python-frozendict .

Also based on https://github.com/Marco-Sulla/python-frozendict

1.4 FrozenOrderedDict

1.4.1 About

`FrozenOrderedDict` is a immutable wrapper around an OrderedDict.

`FrozenOrderedDict` is similar to `frozendict`, and with regards to immutability it solves the same problems:

- Because dictionaries are mutable, they are not hashable and cannot be used in sets or as dictionary keys.
- Nasty bugs can and do occur when mutable data structures are passed around.

It can be initialized just like a `dict` or `OrderedDict`. Once instantiated, an instance of *`FrozenOrderedDict`* cannot be altered, since it does not implement the `MutableMapping` interface.

It does implement the `Mapping` interface, so can be used just like a normal dictionary in most cases.

In order to modify the contents of a *`FrozenOrderedDict`*, a new instance must be created. The easiest way to do that is by calling the *.copy()* method. It will return a new instance of *`FrozenOrderedDict`* initialized using the following steps:

1. A copy of the wrapped OrderedDict instance will be created.
2. If any arguments or keyword arguments are passed to the *.copy()* method, they will be used to create another OrderedDict instance, which will then be used to update the copy made in step #1.
3. Finally, *self.__class__()* will be called, passing the copy as the only argument.

1.4.2 API Reference

class `cawdrey.frozenordereddict.`**`FrozenOrderedDict`**(**args*, ***kwargs*)

An immutable OrderedDict. It can be used as a drop-in replacement for dictionaries where immutability is desired.

`copy`(**args*, ***kwargs*)

`dict_cls`

alias of `collections.OrderedDict`

1.4.3 Copyright

Based on https://github.com/slezica/python-frozendict and https://github.com/mredolatti/python-frozendict .

Also based on https://github.com/wsmith323/frozenordereddict

1.5 NonelessDict

1.5.1 About

`NonelessDict` is a wrapper around dict that will check if a value is `None`/empty/`False`, and not add the key in that case.

The class has a method *`set_with_strict_none_check()`* that can be used to set a value and check only for `None` values.

`NonelessOrderedDict` is based *`NonelessDict`* and `OrderedDict`, so the order of key insertion is preserved.

1.5.2 Usage

1.5.3 API Reference

Provides frozendict, a simple immutable dictionary.

class `cawdrey.nonelessdict.`**`NonelessDict`**(**args*, ***kwargs*)

A wrapper around dict that will check if a value is None/empty/False, and not add the key in that case. Use the set_with_strict_none_check function to check only for None

`copy`(***add_or_replace*)

`dict_cls`

alias of `builtins.dict`

set_with_strict_none_check(*key*, *value*)

Return type `None`

class `cawdrey.nonelessdict.`**NonelessOrderedDict**(**args*, ***kwargs*)

A wrapper around OrderedDict that will check if a value is None/empty/False, and not add the key in that case. Use the set_with_strict_none_check function to check only for None

copy(**args*, ***kwargs*)

dict_cls

alias of `collections.OrderedDict`

set_with_strict_none_check(*key*, *value*)

Return type `None`

1.5.4 Copyright

Based on https://github.com/slezica/python-frozendict and https://github.com/jerr0328/python-helpfuldicts .

1.6 Base Class

1.6.1 About

`FrozenBase` is the base class for *`frozendict`* and *`FrozenOrderedDict`*. If you wish to construct your own frozen dictionary classes, you may wish to inherit from this class.

1.6.2 Usage

1.6.3 API Reference

class cawdrey.base.**FrozenBase**(**args*, ***kwargs*)

Abstract Base Class for Frozen dictionaries

Used by frozendict and FrozenOrderedDict.

Custom subclasses must implement at a minimum `__init__`, `copy`, `fromkeys`.

`__abstractmethods__ = frozenset({'__init__', 'copy'})`

`__annotations__ = {'dict_cls': typing.Union[typing.Type, NoneType]}`

`__args__ = None`

`__contains__`(*key*)

Return type `Any`

`__copy__`(**args*, ***kwargs*)

`__dict__ = mappingproxy({'__module__': 'cawdrey.base', '__annotations__': {'dict_cls`

`__eq__`(*other*)

Return self==value.

`__extra__ = None`

`__getitem__`(*key*)

Return type `Any`

`__hash__ = None`

abstract `__init__`(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

`__iter__`()

`__len__`()

Return type `int`

`__module__ = 'cawdrey.base'`

static `__new__`(*cls*, **args*, ***kwds*)

Create and return a new object. See help(type) for accurate signature.

`__next_in_mro__`

alias of `builtins.object`

`__orig_bases__ = (cawdrey.base.DictWrapper[~KT, ~VT],)`

`__origin__ = None`

`__parameters__ = (~KT, ~VT)`

`__repr__`()

Return repr(self).

Return type `str`

`__reversed__ = None`

`__slots__ = ()`

__subclasshook__()

__tree_hash__ = -9223366135646212879

__weakref__
list of weak references to the object (if defined)

_abc_cache = <_weakrefset.WeakSet object>

_abc_generic_negative_cache = <_weakrefset.WeakSet object>

_abc_generic_negative_cache_version = 42

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 42

_abc_registry = <_weakrefset.WeakSet object>

_gorg
alias of *FrozenBase*

abstract copy(**args*, ***kwargs*)

dict_cls: Optional[Type] = None

classmethod fromkeys(**args*, ***kwargs*)
Returns a new dict with keys from iterable and values equal to value.

get(*k*[, *d*]) → D[k] if k in D, else d. d defaults to None.

items() → a set-like object providing a view on D's items

keys() → a set-like object providing a view on D's keys

values() → an object providing a view on D's values

1.7 Downloading source code

`cawdrey` source code resides on publicly accessible GitHub servers, and can be accessed from the following URL: https://github.com/domdfcoding/cawdrey

If you have `git` installed, you can clone the repository with the following command:

```
$ git clone https://github.com/domdfcoding/cawdrey
> Cloning into 'cawdrey'...
> remote: Enumerating objects: 47, done.
> remote: Counting objects: 100% (47/47), done.
> remote: Compressing objects: 100% (41/41), done.
> remote: Total 173 (delta 16), reused 17 (delta 6), pack-reused 126
> Receiving objects: 100% (173/173), 126.56 KiB | 678.00 KiB/s, done.
> Resolving deltas: 100% (66/66), done.
```

Alternatively, the code can be downloaded in a 'zip' file by clicking:
Clone or download –> Download Zip

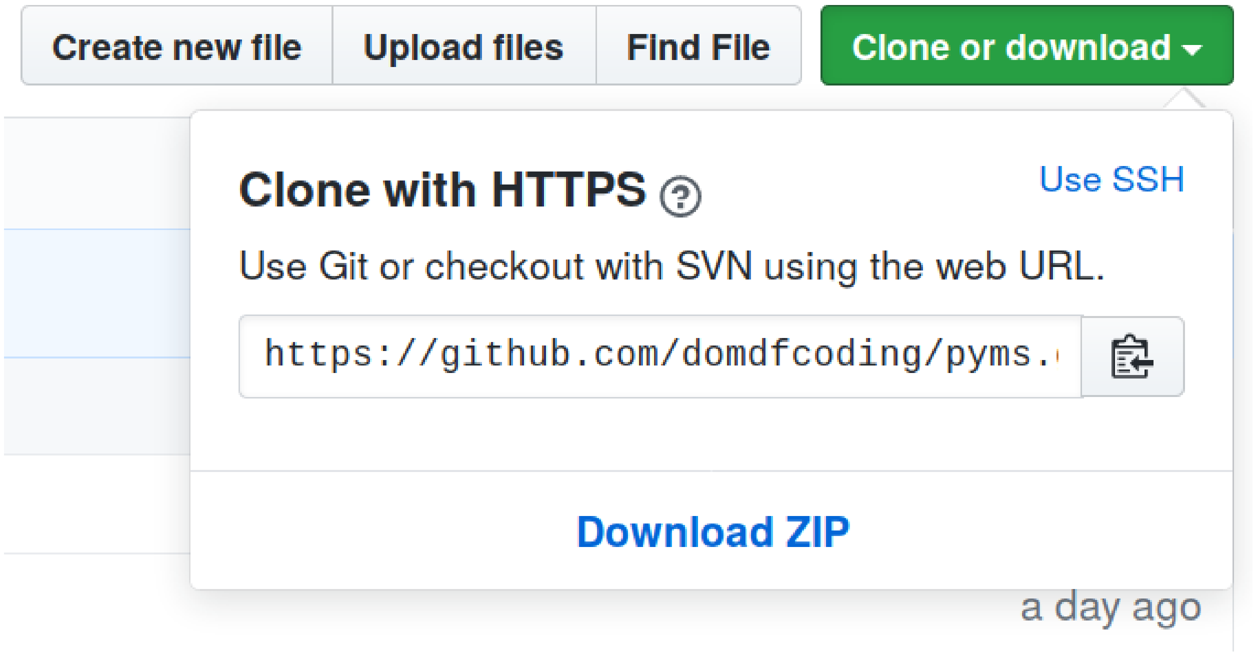

Fig. 1: Downloading a 'zip' file of the source code

1.8 Building from source

To build the `cawdrey` package from source using `setuptools`, run the following command:

```
$ python3 setup.py sdist bdist_wheel
```

`setuptools` is configured using the file `setup.py`.

Different formats are available for built distributions

Format	Description	Notes
`gztar`	gzipped tar file (`.tar.gz`)	default on Unix
`bztar`	bzipped tar file (`.tar.bz2`)	
`xztar`	bzipped tar file (`.tar.bz2`)	
`tar`	tar file (`.tar`)	
`zip`	zip file (`.zip`)	default on Windows
`wininst`	self-extracting ZIP file for Windows	
`msi`	Microsoft Installer	

setup.py

```python
#!/usr/bin/env python
# This file is managed by `git_helper`. Don't edit it directly
"""Setup script"""
```

(continues on next page)

(continued from previous page)

```

# 3rd party
from setuptools import find_packages, setup

from __pkginfo__ import *  # pylint: disable=wildcard-import

setup(
                author=author,
                author_email=author_email,
                classifiers=classifiers,
                description=short_desc,
                entry_points=entry_points,
                extras_require=extras_require,
                include_package_data=True,
                install_requires=install_requires,
                license=__license__,
                long_description=long_description,
                name=pypi_name,
                packages=find_packages(exclude=("tests", "doc-source")),
                project_urls=project_urls,
                py_modules=py_modules,
                python_requires=">=3.6",
                url=web,
                version=__version__,
                keywords=keywords,
                zip_safe=False,

                )
```

__pkginfo__.py

```
#  This file is managed by `git_helper`. Don't edit it directly
#  Copyright (C) 2020 Dominic Davis-Foster <dominic@davis-foster.co.uk>
#
#  This file is distributed under the same license terms as the program it came with.
#  There will probably be a file called LICEN[S/C]E in the same directory as this␣
↪file.
#
#  In any case, this program is distributed in the hope that it will be useful,
#  but WITHOUT ANY WARRANTY; without even the implied warranty of
#  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
#
# This script based on https://github.com/rocky/python-uncompyle6/blob/master/__
↪pkginfo__.py
#

# stdlib
import pathlib

__all__ = [
                "__copyright__",
                "__version__",
                "modname",
                "pypi_name",
                "py_modules",
                "entry_points",
                "__license__",
```

(continues on next page)

(continued from previous page)

```
                "short_desc",
                "author",
                "author_email",
                "github_username",
                "web",
                "github_url",
                "project_urls",
                "repo_root",
                "long_description",
                "install_requires",
                "extras_require",
                "classifiers",
                "keywords",
                "import_name",
                ]

__copyright__ = """
2019-2020 Dominic Davis-Foster <dominic@davis-foster.co.uk>
"""

__version__ = "0.1.5"

modname = "cawdrey"
pypi_name = "cawdrey"
import_name = "cawdrey"
py_modules = []
entry_points = {
                "console_scripts": []
                }

__license__ = "GNU Lesser General Public License v3 or later (LGPLv3+)"

short_desc = "Several useful custom dictionaries"

__author__ = author = "Dominic Davis-Foster"
author_email = "dominic@davis-foster.co.uk"
github_username = "domdfcoding"
web = github_url = f"https://github.com/domdfcoding/cawdrey"
project_urls = {
                "Documentation": f"https://cawdrey.readthedocs.io",
                "Issue Tracker": f"{github_url}/issues",
                "Source Code": github_url,
                }

repo_root = pathlib.Path(__file__).parent

# Get info from files; set: long_description
long_description = (repo_root / "README.rst").read_text().replace("0.1.5", __version__
↪) + '\n'
conda_description = """Several useful custom dictionaries


Before installing please ensure you have added the following channels: domdfcoding,␣
↪conda-forge"""
__all__.append("conda_description")

install_requires = (repo_root / "requirements.txt").read_text().split('\n')
```

(continues on next page)

(continued from previous page)

```
extras_require = {'all': []}

classifiers = [
                'Development Status :: 3 - Alpha',
                'Intended Audience :: Developers',
                'License :: OSI Approved :: GNU Lesser General Public License v3 or␣
↪later (LGPLv3+)',
                'Operating System :: OS Independent',
                'Programming Language :: Python',
                'Programming Language :: Python :: 3.6',
                'Programming Language :: Python :: 3.7',
                'Programming Language :: Python :: 3.8',
                'Programming Language :: Python :: 3 :: Only',
                'Programming Language :: Python :: Implementation :: CPython',
                'Programming Language :: Python :: Implementation :: PyPy',
                'Topic :: Software Development :: Libraries :: Python Modules',
                'Topic :: Utilities',

                ]

keywords = "ordereddict frozenordereddict orderedfrozendict ordered frozen immutable␣
↪frozendict dict dictionary map Mapping MappingProxyType developers"
```

View the Function Index or browse the Source Code.

Browse the GitHub Repository

PYTHON MODULE INDEX

INDEX