

---

# **pywebdoc**

**Quentin Deschamps**

**Jan 30, 2024**



**CONTENTS:**

<b>1</b>	<b>Presentation</b>	<b>1</b>
<b>2</b>	<b>Indices and tables</b>	<b>11</b>
	<b>Python Module Index</b>	<b>13</b>
	<b>Index</b>	<b>15</b>



## PRESENTATION

**pywebdoc** is a CLI application to quickly open web documentation about Python. This tool works for standard libraries and PyPI packages. It opens web pages on your default web browser.

### 1.1 Installation

You can install latest release of *pywebdoc* using `pip3`:

```
pip3 install pywebdoc
```

### 1.2 CLI Reference

The command line interface of *pywebdoc* is available for Linux, macOS and Windows. It uses *click*. To check if you successfully installed the library, you can entry in a command prompt:

```
pywebdoc --help
```

You will see the following content:

```
Usage: pywebdoc [OPTIONS] COMMAND [ARGS]...

Open Python packages url.

Options:
  --version      Show the version and exit.
  -v, --verbose  Give more output
  --help         Show this message and exit.

Commands:
  home          Open the home-page of a PyPI package.
  list-packages Make HTML file with the list of installed PyPI packages.
  list-std      List standard libraries documentation urls.
  py            Open the Python official documentation.
  pypi          Open the PyPI web page of a package.
  rtd           Open the documentation page of a package on ReadTheDocs.
  std           Open the documentation page of a standard Python library.
```

It is the help menu of the CLI. You can see all the commands you can use.

## 1.2.1 Python official documentation

To open the Python official documentation, use the `py` command:

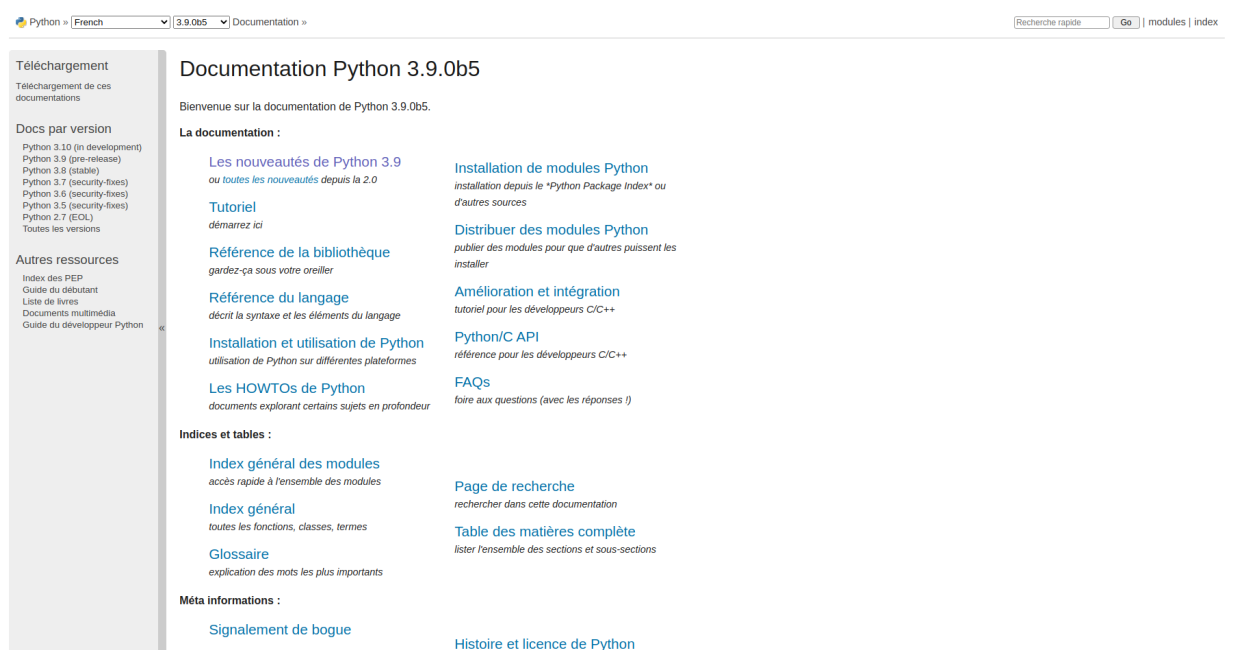
```
pywebdoc py [OPTIONS]
```

You can set the Python version with the `VERSION` option (default: 3) and the language with the `LANG` option (default: english).

For example, to open Python 3.9 documentation in french:

```
pywebdoc py -v 3.9 -l fr
```

This command will open your default web browser on this page:



## 1.2.2 Standard library

To open the documentation page of a standard Python library, use the `std` command:

```
pywebdoc std [OPTIONS] LIBRARY
```

The `LIBRARY` argument is the name of the library or module. You can also set the Python version and the language using the same options than the `py` command.

For example, to open the documentation of `time`:

```
pywebdoc std time
```

This is the web page opened:

Python » English » 3.8.5 » Documentation » The Python Standard Library » Generic Operating System Services »

Quick search   | [previous](#) | [next](#) | [modules](#) | [index](#)

## time — Time access and conversions

This module provides various time-related functions. For related functionality, see also the [datetime](#) and [calendar](#) modules.

Although this module is always available, not all functions are available on all platforms. Most of the functions defined in this module call platform C library functions with the same name. It may sometimes be helpful to consult the platform documentation, because the semantics of these functions varies among platforms.

An explanation of some terminology and conventions is in order.

- The epoch is the point where the time starts, and is platform dependent. For Unix, the epoch is January 1, 1970, 00:00:00 (UTC). To find out what the epoch is on a given platform, look at `time.gmtime(0)`.
- The term *seconds since the epoch* refers to the total number of elapsed seconds since the epoch, typically excluding *leap seconds*. Leap seconds are excluded from this total on all POSIX-compliant platforms.
- The functions in this module may not handle dates and times before the epoch or far in the future. The cut-off point in the future is determined by the C library; for 32-bit systems, it is typically in 2038.
- Function `strptime()` can parse 2-digit years when given `%y` format code. When 2-digit years are parsed, they are converted according to the POSIX and ISO C standards: values 69–99 are mapped to 1969–1999, and values 0–68 are mapped to 2000–2068.
- UTC is Coordinated Universal Time (formerly known as Greenwich Mean Time, or GMT). The acronym UTC is not a mistake but a compromise between English and French.
- DST is Daylight Saving Time, an adjustment of the timezone by (usually) one hour during part of the year. DST rules are magic (determined by local law) and can change from year to year. The C library has a table containing the local rules (often it is read from a system file for flexibility) and is the only source of True Wisdom in this respect.
- The precision of the various real-time functions may be less than suggested by the units in which their value or argument is expressed. E.g. on most Unix systems, the clock “ticks” only 50 or 100 times a second.
- On the other hand, the precision of `time()` and `sleep()` is better than their Unix equivalents: times are expressed as floating point numbers, `time()` returns the most accurate time available (using Unix `gettimeofday()` where available), and `sleep()` will accept a time with a nonzero fraction (Unix `select()` is used to implement this, where available).
- The time value as returned by `gmtime()`, `localtime()`, and `strptime()`, and accepted by `asctime()`, `mktime()` and `strftime()`, is a sequence of 9 integers. The return values of `gmtime()`, `localtime()` and `strptime()` also offer attribute names for individual fields.

You can also open modules like *os.path*:

pywebdoc std os.path

Python » English » 3.8.5 » Documentation » The Python Standard Library » File and Directory Access »

Quick search   | [previous](#) | [next](#) | [modules](#) | [index](#)

## os.path — Common pathname manipulations

**Source code:** [Lib\posixpath.py](#) (for POSIX) and [Lib\ntpath.py](#) (for Windows NT).

This module implements some useful functions on pathnames. To read or write files see [open\(\)](#), and for accessing the filesystem see the [os](#) module. The path parameters can be passed as either strings, or bytes. Applications are encouraged to represent file names as (Unicode) character strings. Unfortunately, some file names may not be representable as strings on Unix, so applications that need to support arbitrary file names on Unix should use bytes objects to represent path names. Vice versa, using bytes objects cannot represent all file names on Windows (in the standard mbcs encoding), hence Windows applications should use string objects to access all files.

Unlike a unix shell, Python does not do any *automatic* path expansions. Functions such as [expanduser\(\)](#) and [expandvars\(\)](#) can be invoked explicitly when an application desires shell-like path expansion. (See also the [glob](#) module.)

**See also:** The [pathlib](#) module offers high-level path objects.

**Note:** All of these functions accept either only bytes or only string objects as their parameters. The result is an object of the same type, if a path or file name is returned.

**Note:** Since different operating systems have different path name conventions, there are several versions of this module in the standard library. The [os.path](#) module is always the path module suitable for the operating system Python is running on, and therefore usable for local paths. However, you can also import and use the individual modules if you want to manipulate a path that is always in one of the different formats. They all have the same interface:

- [posixpath](#) for UNIX-style paths
- [ntpath](#) for Windows paths

*Changed in version 3.8:* [exists\(\)](#), [lexists\(\)](#), [isdir\(\)](#), [isfile\(\)](#), [islink\(\)](#), and [ismount\(\)](#) now return `False` instead of raising an exception for paths that contain characters or bytes unrepresentable at the OS level.

**os.path.abspath(path)**

Return a normalized absolutized version of the pathname *path*. On most platforms, this is equivalent to calling the function [normpath\(\)](#) as follows: `normpath(join(os.getcwd(), path))`.

*Changed in version 3.6:* Accepts a [path-like object](#).

**Note:** If the URL does not exist, you see a “404 Not Found” error in the command prompt.

### 1.2.3 PyPI package

*Pywebdoc* can also open URLs about **PyPI packages**.

#### PyPI web page

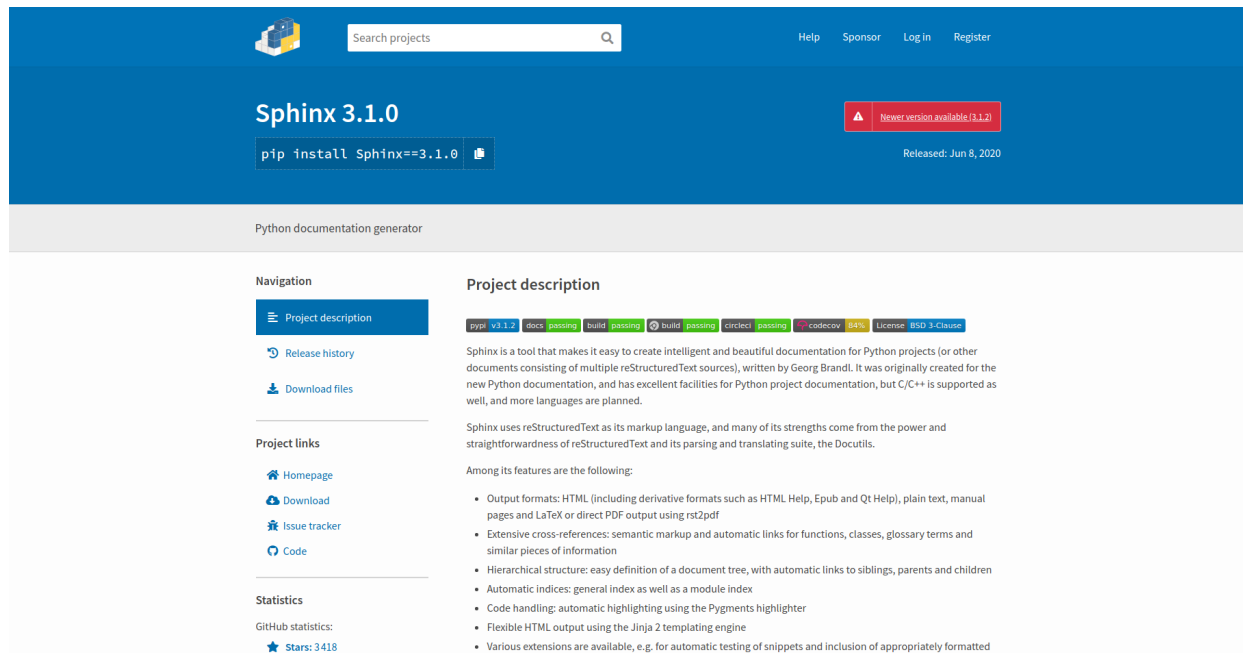
First, you can **open the PyPI web page of a package** using the `pypi` command:

```
pywebdoc pypi [OPTIONS] PACKAGE
```

The `PACKAGE` argument is the name of the package. You can choose the release version with the `VERSION` option. For example, to open the PyPI web page of *Sphinx* 3.1.0:

```
pywebdoc pypi sphinx -v 3.1.0
```

This command will open this web page:



#### Home-page

You can also **open the home-page of a PyPI package** using the `home` command:

```
pywebdoc home [OPTIONS] PACKAGE
```

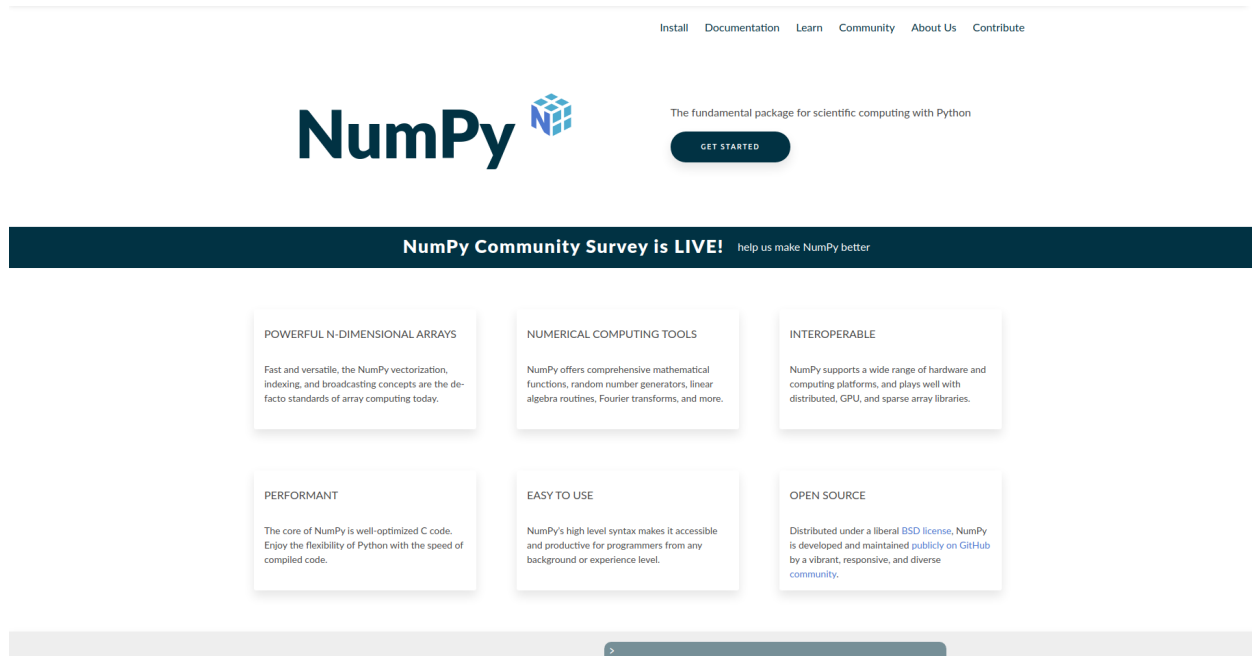
**Warning:** This command will call the `pip` command to get the URL of the home-page. So, **the package must be installed**.

For example, to open the home-page of *numpy*, use:

```
pywebdoc home numpy
```



This command will open this URL:



## ReadTheDocs documentation

Finally, some PyPI packages host documentation on ReadTheDocs. To **open the documentation page of a package on ReadTheDocs**, use the command:

```
pywebdoc rtd [OPTIONS] PACKAGE
```

You can choose the documentation version with the `VERSION` option (default: latest) and the language with the `LANG` option (default: en). For example, you can open the documentation of *numpy* using:

```
pywebdoc rtd numpy
```

None

Resources

- Scipy.org website

## NumPy v1.11 Manual

Welcome! This is the documentation for NumPy 1.11.0, last updated Mar 15, 2017.

Parts of the documentation:

- [NumPy User Guide](#)  
*start here*
- [NumPy Reference](#)  
*reference documentation*
- [F2Py Guide](#)  
*f2py documentation*
- [NumPy Developer Guide](#)  
*contributing to NumPy*

Indices and tables:

- [General Index](#)  
*all functions, classes, terms*
- [Glossary](#)  
*the most important terms explained*
- [Search page](#)  
*search this documentation*
- [Complete Table of Contents](#)  
*lists all sections and subsections*

Meta information:

- [Reporting bugs](#)
- [About NumPy](#)
- [NumPy Enhancement Proposals](#)
- [Release Notes](#)
- [License of NumPy](#)

### Acknowledgements

Large parts of this manual originate from Travis E. Oliphant's book "Guide to NumPy" (which generously entered Public Domain in August 2008). The reference documentation for many of the functions are written by numerous contributors and developers of NumPy, both prior to and during the [NumPy Documentation Marathon](#).

The preferred way to update the documentation is by submitting a pull request on GitHub (see the [Developer Guide](#)). The [NumPy Documentation Wiki](#) can also still be used to submit documentation fixes. Please help us to further improve the NumPy documentation!

© Copyright 2008-2009, The Scipy community. Last updated on Mar 15, 2017. Created using Sphinx 1.5.3.

## 1.2.4 Listing

With *pywebdoc*, you can also see the list of your packages and libraries and their documentation.

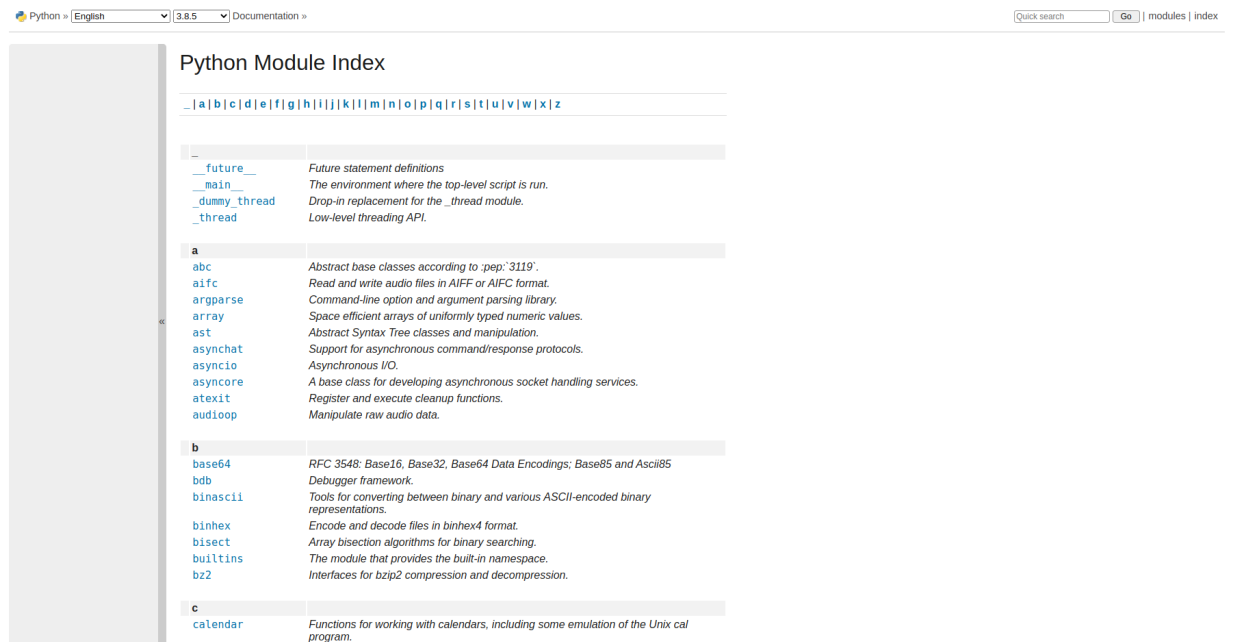
- To get the list of **standard Python libraries**, use the `list-std` command:

```
pywebdoc list-std [OPTIONS]
```

You can use the same options as the `py` command. For example, the command

```
pywebdoc list-std
```

will open this URL:



- To get the list of **installed PyPI packages**, use the `list-packages` command:

```
pywebdoc list-packages [OPTIONS]
```

This command will create a HTML file with the list of installed PyPI packages. The creation of this page may take several minutes. On this page, you will see the following informations about packages:

- name
- version
- summary
- home-page
- license

Once the file is created, it will be opened immediately. If you need to update the HTML file, use the `UPDATE` option. For example, you can see below an example of this file:

## PyPI installed packages

Package	Version	Summary	Home-page	License
<a href="#">affapv</a>	0.1	A Python library for multiprecision Affine Arithmetic	<a href="https://github.com/lip6.fr/hilaire/affapv">https://github.com/lip6.fr/hilaire/affapv</a>	UNKNOWN
<a href="#">alabaster</a>	0.7.12	A configurable sidebar-enabled Sphinx theme	<a href="https://alabaster.readthedocs.io">https://alabaster.readthedocs.io</a>	UNKNOWN
<a href="#">appdirs</a>	1.4.3	A small Python module for determining appropriate platform-specific dirs, e.g. a "user data dir".	<a href="http://github.com/ActiveState/appdirs">http://github.com/ActiveState/appdirs</a>	MIT
<a href="#">apturl</a>	0.5.2	UNKNOWN		UNKNOWN
<a href="#">astroid</a>	2.3.3	An abstract syntax tree for Python with inference support.	<a href="https://github.com/PyCQA/astroid">https://github.com/PyCQA/astroid</a>	LGPL
<a href="#">attrs</a>	19.3.0	Classes Without Boilerplate	<a href="https://www.attrs.org/">https://www.attrs.org/</a>	MIT
<a href="#">Babel</a>	2.8.0	Internationalization utilities	<a href="http://babel.pocoo.org/">http://babel.pocoo.org/</a>	BSD
<a href="#">backcall</a>	0.1.0	Specifications for callback functions passed in to an API	<a href="https://github.com/takuyver/backcall">https://github.com/takuyver/backcall</a>	BSD
<a href="#">bcrypt</a>	3.1.7	Modern password hashing for your software and your servers	<a href="https://github.com/pyca/bcrypt">https://github.com/pyca/bcrypt</a>	Apache License, Version 2.0
<a href="#">beautifulsoup4</a>	4.9.1	Screen-scraping library	<a href="http://www.crummy.com/software/BeautifulSoup/bs4/">http://www.crummy.com/software/BeautifulSoup/bs4/</a>	MIT
<a href="#">binary</a>	1.0.0	UNKNOWN	<a href="https://github.com/0fek/binary">https://github.com/0fek/binary</a>	MIT/Apache-2.0
<a href="#">bleach</a>	3.1.4	An easy safelist-based HTML-sanitizing tool.	<a href="https://github.com/mozilla/bleach">https://github.com/mozilla/bleach</a>	Apache Software License
<a href="#">blinker</a>	1.4	Fast, simple object-to-object and broadcast signaling	<a href="http://pythonhosted.org/blinker/">http://pythonhosted.org/blinker/</a>	MIT License
<a href="#">Brlapi</a>	0.7.0	BrlAPI Client Bindings for Python	<a href="http://brlury.app/">http://brlury.app/</a>	LGPL 2.1
<a href="#">cachetools</a>	4.1.0	Extensible memoizing collections and decorators	<a href="https://github.com/tkem/cachetools/">https://github.com/tkem/cachetools/</a>	MIT
<a href="#">certifi</a>	2019.11.28	Python package for providing Mozilla's CA Bundle.	<a href="https://certifi.io/">https://certifi.io/</a>	MPL-2.0
<a href="#">cffi</a>	1.14.0	Foreign Function Interface for Python calling C code.	<a href="http://cffi.readthedocs.org">http://cffi.readthedocs.org</a>	MIT
<a href="#">chardet</a>	3.0.4	Universal encoding detector for Python 2 and 3	<a href="https://github.com/chardet/chardet">https://github.com/chardet/chardet</a>	LGPL
<a href="#">click</a>	7.1.1	Composable command line interface toolkit	<a href="https://palletsprojects.com/p/click/">https://palletsprojects.com/p/click/</a>	BSD-3-Clause
<a href="#">colorama</a>	0.4.3	Cross-platform colored terminal text.	<a href="https://github.com/tartley/colorama">https://github.com/tartley/colorama</a>	BSD
<a href="#">colored</a>	1.4.2	Simple library for color and formatting to terminal	<a href="https://github.com/dslackw/colored">https://github.com/dslackw/colored</a>	UNKNOWN
<a href="#">colorlog</a>	4.1.0	Log formatting with colors!	<a href="https://github.com/hornyping/python-colorlog">https://github.com/hornyping/python-colorlog</a>	MIT License
<a href="#">command-not-found</a>	0.3	UNKNOWN		UNKNOWN
<a href="#">coverage</a>	5.1	Code coverage measurement for Python	<a href="https://github.com/nedbat/coveragepy">https://github.com/nedbat/coveragepy</a>	Apache 2.0
<a href="#">cryptography</a>	2.8	cryptography is a package which provides cryptographic recipes and primitives to Python developers.	<a href="https://github.com/pyca/cryptography">https://github.com/pyca/cryptography</a>	BSD or Apache License, Version 2.0
<a href="#">cupshelpers</a>	1.0	Helper functions and classes for using CUPS		UNKNOWN
<a href="#">cycler</a>	0.10.0	Composable style cycles	<a href="http://github.com/matplotlib/cycler">http://github.com/matplotlib/cycler</a>	BSD
<a href="#">Cython</a>	0.29.20	The Cython compiler for writing C extensions for the Python language.	<a href="http://cython.org/">http://cython.org/</a>	Apache
<a href="#">dbus-python</a>	1.2.16	Python bindings for libdbus	<a href="http://www.freedesktop.org/wiki/Software/DBusBindings/#python">http://www.freedesktop.org/wiki/Software/DBusBindings/#python</a>	Expat (MIT/X11)

## 1.3 API Reference

You can see below the modules of the *pywebdoc* package.

### 1.3.1 url

The **url** module manages the URLs. The URLs are checked with *requests*.

**url.check\_url(url, log=True)**

Return True if no 404 error on url, else False.

**url.open\_url(url)**

Open url if no 404 error is caught.

**class url.UrlTemplate(template)**

Bases: object

Manage an url template.

**property template**

Get url template.

**render(\*\*kwargs)**

Render url template.

**open\_url(\*\*kwargs)**

Render the url template and open it.

### 1.3.2 packages

The *packages* module manages installed PyPI packages.

**class** `packages.Package(name)`

Bases: object

Manage an installed PyPI package.

**property name**

Get name.

**property url**

Get PyPI url.

**property version**

Get version.

**property summary**

Get summary.

**property home\_page**

Get home-page.

**static get\_list()**

Return the list of installed packages.

**static make\_html\_list(filename)**

Make html file with packages list.

### 1.3.3 pip

The **pip** module can call the **pip** command to get informations about installed PyPI packages.

**pip.call(\*args)**

Call pip command and get result.

**pip.get\_installed\_packages()**

Return the list of installed packages.

**pip.get\_url(package)**

Get the home-page url of a PyPI package. The package must be installed.

### 1.3.4 template

The **template** module manages templates. It uses *jinja2*.

**class** `template.Templates(directory)`

Bases: object

Manage the templates.

**property env**

Get environment.

**get\_template(filename)**

Return a template.

### 1.3.5 logger

The **logger** module manages the configuration of the logger. It uses *colorlog* to get colored logging messages.

`logger.logger_config(level, name=None)`

Setup the logging environment.

### 1.3.6 error

The **error** module manages errors of *pywebdoc*.

**exception** `error.PywebdocError`

Bases: `Exception`

Custom class to raise *pywebdoc* exceptions.

GitHub repository: <https://github.com/Quentin18/pywebdoc/>

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`





## PYTHON MODULE INDEX

### e

`error`, 10

### l

`logger`, 10

### p

`packages`, 9

`pip`, 9

### t

`template`, 9

### u

`url`, 8



## C

`call()` (in module *pip*), 9  
`check_url()` (in module *url*), 8

## E

`env` (*template.Templates* property), 9  
`error`  
     module, 10

## G

`get_installed_packages()` (in module *pip*), 9  
`get_list()` (*packages.Package* static method), 9  
`get_template()` (*template.Templates* method), 9  
`get_url()` (in module *pip*), 9

## H

`home_page` (*packages.Package* property), 9

## L

`logger`  
     module, 10  
`logger_config()` (in module *logger*), 10

## M

`make_html_list()` (*packages.Package* static method), 9  
`module`  
     `error`, 10  
     `logger`, 10  
     `packages`, 9  
     `pip`, 9  
     `template`, 9  
     `url`, 8

## N

`name` (*packages.Package* property), 9

## O

`open_url()` (in module *url*), 8  
`open_url()` (*url.UrlTemplate* method), 8

## P

`Package` (class in *packages*), 9  
`packages`  
     module, 9  
`pip`  
     module, 9  
`PywebdocError`, 10

## R

`render()` (*url.UrlTemplate* method), 8

## S

`summary` (*packages.Package* property), 9

## T

`template`  
     module, 9  
`template` (*url.UrlTemplate* property), 8  
`Templates` (class in *template*), 9

## U

`url`  
     module, 8  
`url` (*packages.Package* property), 9  
`UrlTemplate` (class in *url*), 8

## V

`version` (*packages.Package* property), 9