utools Documentation

Release 1.2.1

Julien Chaumont

May 11, 2016

Contents

A set of useful functions to use in any Python project.

You can go straight to the API documentation here: *Reference/API*.

CHAPTER 1

Description

`utools` is a Python 3 library that gather multiple useful functions for various domains. Developers often have their own private tool library; `utools` is just one of them being publicly released. It does not claim to be exhaustive, always optimized and clearly implemented, but it certainly does the job.

Even if `utools` is at first a personal collection of snippets, every developer is invited to use it and contribute to its code.

CHAPTER 2

Compatibility

Currently supported versions and implementations of Python are:

- python 3.3
- python 3.4
- python 3.5
- pypy3

Others might be compatible but are not officially supported.

CHAPTER 3

Requirements

Required libraries are automatically installed when installing `utools` (see :ref:install to learn more). In its current version, `utools` only requires additional libraries to run the tests and build the documentation:

- pytest 2.9.1
- pytest-cov-2.2.1
- Sphinx 1.4.1
- sphinx-autobuild 0.6.0

CHAPTER 4

Documentation

Refer to other sections of this documentation to learn more about `utools`.

4.1 Installation

4.1.1 Using pip

The `utools` library is hosted on pypi and can be installed with `pip`:

```
$ pip install utools
```

4.1.2 From the Sources

First download the code from the GitHub repository:

```
$ cd /tmp/
$ git clone https://github.com/julienc91/utools
$ cd utools/
$ python setup.py install
```

The repository contains two main branches: `develop` and `master`. The first one is the development branch and might contain work in progress. `master` however is the release branch and is supposed to contain only tested and approved functionalities.

4.2 Tests

Each release of `utools` is garantied to be delivered with a complete set of unit tests that cover the entirety of the code. Of course, it cannot be the assurance of a completely devoid of bug source code; but that's something at least, right?

Tu run the tests, clone the GitHub repository (see *From the Sources*), then run:

```
$ python setup.py test
```

The test suite requires pytest to run but will automatically be installed by running the previous command. It is still possible to run the tests directly from pytest with:

```
$ py.test
```

Code coverage can also be tested with the *pytest-cov <https://pypi.python.org/pypi/pytest-cov>* package this way:

```
$ py.test --cov utools
```

4.3 Reference/API

4.3.1 utools.dates module

Useful functions to work with dates and durations

class utools.dates.**timer**
Bases: object

Get the execution time of a block of code.

Example

The easiest way to use the timer is inside a 'with' statement:

```
>>> import time
>>> t = timer()
>>> with t:
...     time.sleep(1)
>>> t.get()
1.001263
```

The timer class also provides methods to start and stop the timer when you want:

```
>>> t = timer()
>>> t.get()
0.
>>> t.start()
>>> t.get()
1.425219
>>> t.stop()
>>> t.get()
2.636786
```

get()
Get the current timer value in seconds.

Returns: the elapsed time in seconds since the timer started or until the timer was stopped

ongoing()
Check if the timer is running.

Returns: True if the timer is currently running, False otherwise

reset()
Reset the timer.

start()
Start or restart the timer.

stop()
Stop the timer.

4.3.2 utools.dicts module

Useful functions to work with dictionaries.

utools.dicts.**deep_get**(*d*, **keys*, *default=None*)

Recursive safe search in a dictionary of dictionaries.

Parameters

- **d** – the dictionary to work with
- ***keys** – the list of keys to work with
- **default** – the default value to return if the recursive search did not succeed

Returns The value wich was found recursively in d, or default if the search did not succeed

Example

```
>>> d = {"user": {"id": 1, "login": "foo"}, "date": "2016-04-27"}
>>> deep_get(d, "user", "login")
"foo"
>>> deep_get(d, "user")
{"id": 1, "login": "foo"}
>>> deep_get(d, "user", "name")
None
>>> deep_get(d, "user", "name", default="bar")
"bar"
```

4.3.3 utools.files module

Useful functions to manipulate files.

utools.files.**read_item**(*f*, *item_type*)

Extract a single item from the current line of a file-like object.

Parameters

- **f** (*file*) – the file-like object to read from
- **item_type** (*type*) – type of the element to extract

Returns The extracted element

Example

The file "a.input" contains three lines and three with a single digit on each:

```
>>> with open("a.input") as f:
...     print(utools.files.read_item(f, int))
...     print(utools.files.read_item(f, str))
...     print(utools.files.read_item(f, float))
...
1
"2"
3.0
```

utools.files.**read_mutiple_items**(*f*, *container_type*, *item_type*, *separator=' '*)
Extract an iterable from the current line of a file-like object.

Parameters

- **f** (*file*) – the file-like object to read from
- **container_type** (*type*) – type of the iterable that will be returned
- **item_type** (*type*) – type of the values that will be elements of the returned iterable
- **separator** (*str*) – the separator between two consecutive items

Returns The extracted iterable

Example

The file "a.input" contains three lines and three comma-separated digits on each:

```
>>> with open("a.input") as f:
...     print(utools.files.read_multiple_items(f, list, int, separator=","))
...     print(utools.files.read_multiple_items(f, set, str, separator=","))
...     print(utools.files.read_multiple_items(f, tuple, float, separator=","))
...
[1, 2, 3]
{"4", "5", "6"}
(7.0, 8.0, 9.0)
```

4.3.4 utools.math module

Useful mathematical functions.

utools.math.**binomial_coefficient**(*n*, *k*)
Calculate the binomial coefficient indexed by n and k.

Parameters

- **n** (*int*) – positive integer
- **k** (*int*) – positive integer

Returns The binomial coefficient indexed by n and k

Raises

- TypeError – If either n or k is not an integer
- ValueError – If either n or k is negative, or if k is strictly greater than n

utools.math.**find_divisors**(*n*)
Find all the positive divisors of the given integer n.

Parameters **n** (*int*) – strictly positive integer

Returns A generator of all the positive divisors of n

Raises

- TypeError – if n is not an integer
- ValueError – if n is negative

utools.math.**is_prime**(*n*)

Miller-Rabin primality test. Keep in mind that this is not a deterministic algorithm: if it return True, it means that n is probably a prime.

Parameters **n** (*int*) – the integer to check

Returns True if n is probably a prime number, False if it is not

Raises TypeError – if n is not an integer

Note: Adapted from https://rosettacode.org/wiki/Miller%E2%80%93Rabin_primality_test#Python

---

utools.math.**prime_generator**(*p_min=2*, *p_max=None*)

Generator of prime numbers using the sieve of Eratosthenes.

Parameters

- **p_min** (*int*) – prime numbers lower than p_min will not be in the resulting primes
- **p_max** (*int*) – the generator will stop when this value is reached, it means that there will be no prime bigger than this number in the resulting primes. If p_max is None, there will not be any upper limit

Returns A generator of all the consecutive primes between p_min and p_max

Raises TypeError – if p_min or p_max is not an integer

utools.math.**sieve_of_eratosthenes**(*p_min=2*, *p_max=None*)

Generator of prime numbers using the sieve of Eratosthenes.

Note: Adapted from http://code.activestate.com/recipes/117119/

Parameters

- **p_min** (*int*) – prime numbers lower than p_min will not be in the resulting primes
- **p_max** (*int*) – the generator will stop when this value is reached, it means that there will be no prime bigger than this number in the resulting primes. If p_max is None, there will not be any upper limit

Returns A generator of all the consecutive primes between p_min and p_max

Raises TypeError – if p_min or p_max is not an integer

4.4 About

utools is open-source and published under the MIT License. It is a permissive license and therefore let people do whatever they want with the library as long as the attribution is made cleared. The author of the library cannot be held responsible for any use which may be made of utools.

Author Julien Chaumont - https://julienc.io

License MIT - https://opensource.org/licenses/MIT

Repository GitHub - https://github.com/julienc91/utools

4.4.1 Report a Bug

Please use the GitHub bugtracker to report any problem you might have with `utools`:

https://github.com/julienc91/utools/issues

4.4.2 Contribute

Contributions are welcome. Learn how to help on GitHub:

https://guides.github.com/activities/contributing-to-open-source/

4.4.3 Contact

Feel free to contact me by email for any question related to `utools` or to my work in general at `utools[at]julienc.io`.

u

B

D

F

G

I

O

P

R

S

T

U