

---

# **b3j0f.utils Documentation**

*Release 1.4.4*

**b3j0f**

**Sep 27, 2017**



---

## Table of Contents

---

<b>1</b>	<b>ChangeLog</b>	<b>1</b>
<b>2</b>	<b>Indices and tables</b>	<b>5</b>
<b>3</b>	<b>Description</b>	<b>7</b>
<b>4</b>	<b>Links</b>	<b>9</b>
<b>5</b>	<b>Installation</b>	<b>11</b>
<b>6</b>	<b>Features</b>	<b>13</b>
<b>7</b>	<b>Examples</b>	<b>15</b>
<b>8</b>	<b>Perspectives</b>	<b>19</b>
<b>9</b>	<b>Donation</b>	<b>21</b>



### 1.4.4 (2016/10/07)

- fix error while looking up a library which is named like a keyword.

### 1.4.3 (2016/06/05)

- add the function `reflect.isoldstyle`.

### 1.4.2 (2015/02/22)

- add `requirements.txt` and `changelog.rst` in building packages.

### 1.4.1 (2015/02/21)

- add the function `iterable.hashiter` in order to hash not hashable iterables.
- add `runtime.getcodeobj` function in order to produce a code object from old code object.

### 1.4.0 (2015/12/22)

- add the parameter `scope` in the function `path.lookup`.
- add `safe` parameter in the function `path.lookup`.
- use static safe scope in the module `runtime`.

### 1.3.0 (2015/11/18)

- add the decorator `b3j0f.path.alias` function for quick registering python objects.
- change contact address.

### 1.2.0 (2015/11/8)

- add dependency to the modules `six` and `future` and remove version functions.
- use relative imports in UTs.

### 1.1.0 (2015/11/8)

- add functions `last`, `itemat` and `slice` in the `iterable` module.

### 1.0.1 (2015/10/29)

- **add in the module runtime:**
  - the function `_safe_builtins`
  - the variable `SAFE_BUILTINS`
  - the functions `safe_eval` and `safe_exec` used to `eval/exec` python code without IO functions.
- add the `builtins` module in the module version.
- add description of `tu`, `version` and `runtime` in the `readme`.

### 1.0.0 (2015/10/20)

- set stable version.

### 0.10.4 (2015/10/02)

- update the module version with `python3` and `python2` common modules with different names.
- add support of `pyhton3.5`.

### 0.10.3 (15/09/28)

- add `range` and `raw_input` definition in the module version in order to use `xrange` whatever `python2/3` execution environment, and `raw_input` in `python3`.

## 0.10.2 (15/09/28)

- auto generate documentation in the addproperties decorator.

## 0.10.1 (15/09/28)

- improve addproperties execution time and unittests.

## 0.10.0 (15/09/28)

- add addproperties decorator in the module property.

## 0.9.5 (15/07/14)

- add `__version_info__` in the base package.
- fix method proxy generation in the function `b3j0f.utils.proxy.proxify_elt`.
- add public parameter in `b3j0f.utils.proxy.proxify_elt`.

## 0.9.4 (15/06/27)

- add `__getproxy__` instance method name in order to specialize the generation of a proxy from the elt to proxify.

## 0.9.3 (15/06/14)

- add docs directory in order to be hosted by readthedocs.

## 0.9.2 (15/06/13)

- add dependency to ordereddict.

## 0.9.1 (15/06/13)

- add shields.io badges.

## 0.9.0 (15/05/20)

- add wheel distribution package.

## **0.8.7 (15/05/20)**

- Fix UTs.

## **0.8.6 (20/05/15)**

- Add definition of getcallargs and OrderedDict in b3j0f.utils.version module.
- Move changelog from README to a separate documentation page.

## **0.8.5 (16/02/15)**

- Add proxy module.

## CHAPTER 2

---

### Indices and tables

---

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



## CHAPTER 3

---

Description

---

Utilities for Python.



## CHAPTER 4

---

### Links

---

- [Homepage](#)
- [PyPI](#)
- [Documentation](#)



## CHAPTER 5

---

### Installation

---

```
pip install b3j0f.utils
```



This library provides a set of generic tools in order to ease development of projects in python  $\geq 2.6$ .

Provided tools are:

- chaining: chain object methods calls in a dedicated Chaining object. Such method calls return the Chaining object itself, allowing multiple calls to object methods to be invoked in a concise statement.
- iterable: tools in order to manage iterable elements.
- path: python object path resolver, from object to absolute/relative path or the inverse.
- property: (un)bind/find properties in reflective and oop concerns.
- reflect: tools which ease development with reflective concerns.
- runtime: ease runtime execution (transform dynamic variable to static variable in function, provide safe eval/exec functions).
- proxy: create proxy (from design pattern) objects from a routine or an object which respects the signature and description of the proxified element.
- ut: improve unit tests.
- version: ease compatibility between python version (from 2.x to 3.x).



## Chaining

```
>>> # add characters to a string in one line
>>> from b3j0f.utils.chaining import Chaining, ListChaining
>>> c = Chaining("te").__iadd__("s").__iadd__("t")
>>> # display content of Chaining
>>> c._
test
>>> # call several strings operations on several strings and get operation results in
↳one line
>>> ListChaining("Test", "Example").upper().lower()[:]
[["TEST", "EXAMPLE"], ["test", "example"]]
```

## Iterable

```
>>> from b3j0f.utils.iterable import is_iterable, first, last, itemat, sliceit,
↳hashiter
>>> is_iterable(1)
False
>>> is_iterable('aze')
True
>>> is_iterable('aze', exclude=str)
False
```

```
>>> from b3j0f.utils.version import OrderedDict
>>> od = OrderedDict(((1, 2), (3, 4), (5, 6)))
>>> first(od)
'1'
>>> first({}, default='test')
'test'
```

```
>>> last(od)
'5'
>>> last('', default='test')
'test'
```

```
>>> itemat(od, -1)
'5'
>>> itemat(od, 1)
'3'
```

```
>>> sliceit(od, -2, -1)
['3']
```

```
>>> hashiter([1, 2])
8
```

## Path

```
>>> from b3j0f.utils.path import lookup, getpath
>>> getpath(lookup)
'b3j0f.utils.path.lookup'
>>> getpath(lookup("b3j0f.utils.path.getpath"))
'b3j0f.utils.path.getpath'
```

## Property

```
>>> from b3j0f.utils.property import put_properties, get_properties, del_properties
>>> put_properties(min, {'test': True})
>>> assert get_properties(min) == {'test': True}
>>> del_properties(min)
>>> assert get_properties(min) is None
```

```
>>> from b3j0f.utils.property import addproperties
>>> def before(self, value, name): # define a before setter
>>>     self.before = value if hasattr(self, 'after') else None
>>> def after(self, value, name):
>>>     self.after = value + 2 # define a after setter
>>> @addproperties(['test'], bfset=before, afset=after) # add python properties
>>> class Test(object):
>>>     pass
>>> assert isinstance(Test.test, property) # assert property is bound
>>> test = Test()
>>> test.test = 2
>>> assert test.update is None # assert before setter
>>> assert test.test == test._test == 2 # assert default setter
>>> assert test.after == 4
```

## Reflect

```
>>> from b3j0f.utils.reflect import base_elts, is_inherited
>>> class BaseTest(object):
>>>     def test(self): pass
>>> class Test(BaseTest): pass
>>> class FinalTest(Test): pass
>>> base_elts(FinalTest().test, depth=1)[-1].im_class.__name__
Test
>>> base_elts(FinalTest().test)[-1].im_class.__name__
BaseTest
```

```
>>> is_inherited(FinalTest.test)
True
>>> is_inherited(BaseTest.test)
False
```

## Proxy

```
>>> from b3j0f.utils.proxy import get_proxy, proxified_elt
>>> l = lambda: 2
>>> proxy = get_proxy(l, lambda: 3)
>>> proxy()
3
>>> assert proxified_elt(proxy) is l
True
>>> proxified_elt(proxy)()
2
>>> proxy = get_proxy(l)
>>> proxy()
2
>>> assert proxy is not l
>>> assert proxified_elt(proxy) is l
```

## Runtime

```
>>> from b3j0f.utils.runtime import safe_eval
>>> try:
>>>     safe_eval('open')
>>> except NameError:
>>>     print('open does not exist')
open does not exist
```

## Version

```
>>> from b3j0f.utils.version import getcallargs
>>> # getcallargs is same function from python>2.7 for python2.6
>>> from b3j0f.utils.version import PY3, PY2, PY26, PY27
>>> # PY3 is True if python version is 3, etc.
```

## UT

```
>>> from b3j0f.utils.ut import UTCase # class which inherits from unittest.TestCase
>>> UTCase.assertIs and True # all methods of python2/3 TestCase are implemented in
↳ the UTCase for python v>2.
True
```

## CHAPTER 8

---

### Perspectives

---

- Cython implementation.



## CHAPTER 9

---

Donation

---