
methodtools

Release 0.1.2

May 17, 2020

Contents

1	methodtools — functools for methods	3
2	Indices and tables	5
	Python Module Index	7
	Index	9

For now, `methodtools.lru_cache()` is the single content of this package.

CHAPTER 1

methodtools — functools for methods

Expand functools features to methods, classmethods, staticmethods and even for (unofficial) hybrid methods.

For now, methodtools only provides `methodtools.lru_cache()`.

Use methodtools module instead of functools module. Then it will work as you expected - cache for each bound method.

```
from methodtools import lru_cache

class A(object):

    # cached method. the storage lifetime follows `self` object
    @lru_cache()
    def cached_method(self, args):
        ...

    # cached classmethod. the storage lifetime follows `A` class
    @lru_cache() # the order is important!
    @classmethod # always lru_cache on top of classmethod
    def cached_classmethod(self, args):
        ...

    # cached staticmethod. the storage lifetime follows `A` class
    @lru_cache() # the order is important!
    @staticmethod # always lru_cache on top of staticmethod
    def cached_staticmethod(self, args):
        ...

    @lru_cache() # just same as functools.lru_cache
    def cached_function():
        ...
```

`methodtools.lru_cache(maxsize=128, typed=False)`

Least-recently-used cache decorator.

If `maxsize` is set to None, the LRU features are disabled and the cache can grow without bound.

If *typed* is True, arguments of different types will be cached separately. For example, `f(3.0)` and `f(3)` will be treated as distinct calls with distinct results.

Arguments to the cached function must be hashable.

View the cache statistics named tuple (hits, misses, maxsize, currsize) with `f.cache_info()`. Clear the cache and statistics with `f.cache_clear()`. Access the underlying function with `f.__wrapped__`.

See: http://en.wikipedia.org/wiki/Cache_algorithms#Least_Recently_Used

CHAPTER 2

Indices and tables

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

m

methodtools, [1](#)

L

`lru_cache()` (*in module `methodtools`*), 3

M

`methodtools` (*module*), 1