



Модуль functools



Онлайн-тренажер Python 3 для начинающих

Теория без воды. Задачи с автоматической проверкой. Подсказки на русском языке. Работает в любом современном браузере.

НАЧАТЬ БЕСПЛАТНО

Модуль functools - сборник функций высокого уровня: взаимодействующих с другими функциями или возвращающие другие функции.

Модуль functools определяет следующие функции:

functools.cmp_to_key(func) - превращает функцию сравнения в key-функцию. Используется с инструментами, принимающие key-функции (sorted(), min(), max(), heapq.nlargest(), heapq.nsmallest(), itertools.groupby()). Эта функция в основном используется в качестве переходного инструмента для программ, преобразованных из Python 2, которые поддерживали использование функций сравнения.

Функция сравнения - функция, принимающая два аргумента, сравнивающая их и возвращающая отрицательное число, если первый аргумент меньше, ноль, если равен и положительное число, если больше. Key-функция - функция, принимающая один аргумент и возвращающая другое значение, определяющее положение аргумента при сортировке.

@functools.lru_cache(maxsize=128, typed=False) - декоратор, который сохраняет результаты maxsize последних вызовов. Это может сэкономить время при дорогих вычислениях, если функция периодически вызывается с теми же аргументами.

Поскольку в качестве кэша используется словарь, все аргументы должны быть хешируемыми.

Если maxsize установлен в None, кэш может возрасти бесконечно. Также функция наиболее эффективна, если maxsize это степень двойки.

Если typed - True, аргументы функции с разными типами будут кэшироваться отдельно. Например, f(3) и f(3.0) будут считаться разными вызовами, возвращающие, возможно, различный результат.

Чтобы помочь измерить эффективность кэширования и отрегулировать размер кэша, обернутая функция дополняется функцией cache_info(), возвращающая namedtuple, показывающий попадания в кэш, промахи, максимальный размер и текущий размер. В многопоточном окружении, количество попаданий и промахов приблизительно.

Также имеется функция cache_clear() для очистки кэша.

Оригинальная функция доступна через атрибут __wrapped__.

```
>>> from functools import lru_cache
>>> # Числа Фибоначчи (попробуйте убрать lru_cache) :)
...
>>> @lru_cache(maxsize=None)
... def fib(n):
```

Поиск ...

Свежее

- Модуль csv - чтение и запись CSV файлов
- Создаём сайт на Django, используя хорошие практики. Часть 1: создаём проект
- Онлайн-обучение Python: сравнение популярных программ

Категории

- Книги о Python
- GUI (графический интерфейс пользователя)
- Курсы Python
- Модули
- Новости мира Python
- NumPy
- Обработка данных
- Основы программирования
- Примеры программ
- Типы данных в Python
- Видео
- Python для Web
- Работа для Python-программистов

Полезные материалы

- Сделай свой вклад в развитие сайта!
- Самоучитель Python
- Карта сайта
- Отзывы на книги по Python
- Реклама на сайте

Мы в соцсетях

```

...     if n < 2:
...         return n
...     return fib(n-1) + fib(n-2)
...
>>>
>>> print([fib(n) for n in range(100)])
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610,
>>> print(fib.cache_info())
CacheInfo(hits=196, misses=100, maxsize=None, currsize=100)

```

@functools.total_ordering - декоратор класса, в котором задан один или более методов сравнения. Этот декоратор автоматически добавляет все остальные методы. Класс должен определять один из методов `__lt__()`, `__le__()`, `__gt__()`, или `__ge__()`. Кроме того, он должен определять метод

Например:

```

@total_ordering
class Student:
    def __eq__(self, other):
        return ((self.lastname.lower(), self.firstname.lower())
                (other.lastname.lower(), other.firstname.lower()))
    def __lt__(self, other):
        return ((self.lastname.lower(), self.firstname.lower())
                (other.lastname.lower(), other.firstname.lower()))

```

functools.partial(func, *args, **keywords) - возвращает partial-объект (по сути, функцию), который при вызове вызывается как функция func, но дополнительно передают туда позиционные аргументы args, и именованные аргументы kwargs. Если другие аргументы передаются при вызове функции, то позиционные добавляются в конец, а именованные расширяют и перезаписывают.

Например:

```

>>> from functools import partial
>>> basetwo = partial(int, base=2)
>>> basetwo.__doc__ = 'Convert base 2 string to an int.'
>>> print(basetwo('10010'))
18

```

functools.reduce(function, iterable[, initializer]) - берёт два первых элемента, применяет к ним функцию, берёт значение и третий элемент, и таким образом сворачивает iterable в одно значение. Например, `reduce(lambda x, y: x+y, [1, 2, 3, 4, 5])` эквивалентно `((((1+2)+3)+4)+5)`. Если задан `initializer`, он помещается в начале последовательности.

functools.update_wrapper(wrapper, wrapped, assigned=WRAPPER_ASSIGNMENTS, updated=WRAPPER_UPDATES) - обновляет функцию-оболочку, чтобы она стала похожей на обернутую функцию. `assigned` - кортеж, указывающий, какие атрибуты исходной функции копируются в функцию-оболочку (по умолчанию это `WRAPPER_ASSIGNMENTS` (`__name__`, `__module__`, `__annotations__` и `__doc__`)). `updated` - кортеж, указывающий, какие атрибуты обновляются (по умолчанию это `WRAPPER_UPDATES` (обновляется `__dict__` функции-оболочки)).

@functools.wraps(wrapped, assigned=WRAPPER_ASSIGNMENTS, updated=WRAPPER_UPDATES) - удобная функция для вызова `partial(update_wrapper, wrapped=wrapped, assigned=assigned, updated=updated)` как **декоратора** при определении функции-оболочки.

Например:

```
>>>
```

Подпишись на обновления [по RSS](#) или по почте!

Ваш e-mail...

Подписаться!

```
>>> from functools import wraps
>>> def my_decorator(f):
...     @wraps(f)
...     def wrapper(*args, **kwds):
...         print('Calling decorated function')
...         return f(*args, **kwds)
...     return wrapper
...
>>> @my_decorator
... def example():
...     """Docstring"""
...     print('Called example function')
...
>>> example()
Calling decorated function
Called example function
>>> print(example.__name__)
example
>>> print(example.__doc__)
Docstring
```

Ошибка в тексте?
Выделите ее мышкой!
И нажмите:



Для вставки кода на Python в комментарий заключайте его в теги `<pre><code class="python3">Ваш код</code></pre>`



Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?

Имя

♡ 10

Поделиться

Лучшие Новые Старые



Павел Петров

6 лет назад

admin@kamenka.su fibo bench one thread version

```
import time
```

```
targetNum = 1000000
```

```
def fibo(x):
```

```
    if x==0:
```

```
        return(0)
```

```
    if x<=2:
```

```
        return(1)
```

```
    prev=0
```

```
    cur=1
```

```
    while x>1:
```

```
        next=cur+prev
```

```
        prev=cur
```

```
        cur=next
```

```
    x-=1
```

```
    return(cur)
```

```
start = time.perf_counter()
```

```
print(fibo(targetNum))
```

```
stop = time.perf_counter()
```

```
print ('%.2f seconds to done' % (stop-start))
```

3

1

Ответить



Подписаться

О защите персональных данных