Декоратори. Функциальное программирование.

План

0. Источники

1. Декораторы в Python. (Интервью)(Code)

2. Мультидекоратор. (Code)

## 3. Передача декоратором аргументов в функцию (Code)

## 4. Декораторы с аргументами (Code)

5. Функциональном Программировании

6. lambda функция (Code)

#### 7. map функция (Code)

#### 8. reduce функция (Code)

#### 9. Почему map и reduce лучше?

#### 10. filter функция (Code)

0. Источники.

Декоратор

* <https://pythonworld.ru/osnovy/dekoratory.html>
* <https://tproger.ru/translations/demystifying-decorators-in-python/>
* https://habr.com/ru/post/141411/

Функциональном Программировании

* <https://docs.python.org/3/library/functools.html>
* https://docs.python.org/3/library/functions.html
* [https://habr.com/ru/post/507642/#:~:text=%D0%A7%D0%B8%D1%81%D1%82%D0%B0%D1%8F%20%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F%20%E2%80%94%20%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D0%B0%D1%8F%20%D1%8F%D0%B2%D0%BB%D1%8F%D0%B5%D1%82%D1%81%D1%8F,%D0%BD%D0%B0%D0%B1%D0%BE%D1%80%D0%B5%20%D0%B0%D1%80%D0%B3%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2%20%D0%B2%D1%8B%D0%B4%D0%B0%D0%B2%D0%B0%D1%82%D1%8C%20%D0%BE%D0%B4%D0%B8%D0%BD%D0%B0%D0%BA%D0%BE%D0%B2%D1%8B%D0%B9%20%D1%80%D0%B5%D0%B7%D1%83%D0%BB%D1%8C%D1%82%D0%B0%D1%82](https://habr.com/ru/post/507642/" \l ":~:text=Чистая Функция — Функция которая является,наборе аргументов выдавать одинаковый результат).
* <https://habr.com/ru/company/typeable/blog/542440/>
* <https://devpractice.ru/fp-python-part1-general/>
* <https://habr.com/ru/post/257903/>
* <http://pythonicway.com/python-functinal-programming>
* [https://ru.wikibooks.org/wiki/Python/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5\_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5\_%D0%BD%D0%B0\_Python](https://ru.wikibooks.org/wiki/Python/Функциональное_программирование_на_Python)

1. Декораторы в Python. (Интервью)(Code)

Декораторы — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода.

Декораторы не обязательно должны быть функциями, это может быть любой вызываемый объект.

* Декораторы не обязаны возвращать функции, они могут возвращать что угодно. Но обычно мы хотим, чтобы декоратор вернул объект того же типа, что и декорируемый объект.
* Также декораторы могут принимать в качестве аргументов не только функции.
* Необходимость в декораторах может быть неочевидной до написания библиотеки. Поэтому, если декораторы кажутся вам бесполезными, посмотрите на них с точки зрения разработчика библиотеки. Хорошим примером является декоратор представления в Flask.
* Также стоит обратить внимание на [functools.wraps()](_blank) — функцию, которая помогает сделать декорируемую функцию похожей на исходную, делая такие вещи, как сохранение doctstring исходной функции.

Декораторы несколько замедляют вызов функции, не забывайте об этом.

* Если функция декорирована — это не отменить.
* Декораторы оборачивают функции, что может затруднить отладку.

Декораторы могут быть использованы:

* для расширения возможностей функций из сторонних библиотек (код которых мы не можем изменять);
* для упрощения отладки (мы не хотим изменять код, который ещё не устоялся);
* для расширения различных функций одним и тем же кодом, без повторного его переписывания каждый раз.

2. Мультидекоратор. (Code)

Можно использовать несколько декораторов для одной функции

## 3. Передача декоратором аргументов в функцию (Code)

## 4. Декораторы с аргументами (Code)

5. Функциональном Программировании

Давайте кратко разберемся в этих двух понятиях прежде чем продолжим.

**Чистая Функция** — Функция которая является детерминированной и не обладает никакими побочными эффектами.

То есть чтобы функция являлась чистой она должна быть детерминированной — то есть каждый раз при одинаковом наборе аргументов выдавать одинаковый результат. Чистые функции хороши тем что:

* Они проще читаются
* Они проще поддерживаются
* Они проще тестируются
* Они не зависят от того в каком порядке их вызывать

Функциональное программирование можно охарактеризовать тем что в нем используются только чистые функции.

[**Функции высшего порядка**](https://ru.wikipedia.org/wiki/Функция_высшего_порядка)— в программировании функция, принимающая в качестве аргументов другие функции или возвращающая другую функцию в качестве результата.

ФП — подход к программированию, при использовании которого функции можно передавать другим функциям в качестве параметров и использовать функции в качестве значений, возвращаемых другими функциями…

Любой код, написанный в функциональном стиле, может быть довольно легко переписан в обычном стиле без потери качества, но более примитивно.

Цель ФП заключается в том, чтобы:

* писать код более простой, понятный;
* который легче поддерживать;
* который занимает меньше памяти:
* DRY (Don’t Repeat Yourself — Не повторяйся).

6. lambda функция (Code)

#### 7. map функция (Code)

В Python функция map принимает два аргумента: функцию и аргумент составного типа данных, например, список. map применяет к каждому элементу списка переданную функцию.

Функция map может быть так же применена для нескольких списков, в таком случае функция-аргумент должна принимать количество аргументов, соответствующее количеству списков.

#### 8. reduce функция (Code)

#### 9. Почему map и reduce лучше? (Code)

* Во-первых, они обычно укладываются в одну строку.
* Во-вторых, важные части итерации,– коллекция, операция и возвращаемое значение,– всегда находятся в одном месте map и reduce.
* В-третьих, код в цикле может изменить значение ранее определённых переменных, или влиять на код, находящийся после него. По соглашению, map и reduce – функциональны.
* В-четвёртых, map и reduce – элементарные операции. Вместо построчного чтения циклов читателю проще воспринимать map и reduce, встроенные в сложные алгоритмы.
* В-пятых, у них есть много друзей, позволяющих полезное, слегка изменённое поведение этих функций. Например, filter, all, any и find.

#### 10. filter функция (Code)

Функция filter предлагает элегантный вариант фильтрации элементов последовательности. Принимает в качестве аргументов функцию и последовательность, которую необходимо отфильтровать

Обратите внимание, что функция, передаваемая в filter должна возвращать значение True / False, чтобы элементы корректно отфильтровались.