


Программирование, лекция 9. Функции высшего порядка. Модули



Кафедра ИУ7 МГТУ им. Н. Э. Баумана,
2022 год



Функциональное программирование

Функциональное программирование - парадигма, в которой процесс вычисления трактуется как вычисление значений функций в их математическом понимании.

В императивном программировании работа функций может зависеть не только от аргументов, но и от внешних переменных, то есть вызов функции с одними и теми же параметрами на разных этапах выполнения алгоритма может приводить к различному результату.

В функциональном программировании результат выполнения функции всегда будет одним и тем же. Преимущества: кэширование, распараллеливание выполнения.

Элементы функционального программирования в Python

- списковые включения
- встроенные функции высших порядков
- замыкания
- лямбда-выражения
- генераторы
- и т. д.

Функции высшего порядка

Функция первого порядка - та, которая принимает только значения “простых” (не функциональных) типов и возвращает значения таких же типов в качестве результата.

Функция высшего порядка - та, которая принимает в качестве аргументов или возвращает другие функции.

Замыкания

Замыкание (closure) в программировании — функция первого класса, в теле которой присутствуют ссылки на переменные, объявленные вне тела этой функции в окружающем коде и не являющиеся её параметрами.

```
def outer():  
    x = 1  
  
    def inner():  
        print('x in outer function: ', x)  
  
    return inner
```

Аннотации

Аннотации - способ добавлять произвольные метаданные к аргументам функции и возвращаемому значению

```
def div(a: 'the dividend',  
       b: 'the divisor') -> 'the result of dividing a by b':  
    """Divide a by b"""  
    return a / b
```

Анонимные функции. Оператор `lambda`

Оператор **`lambda`** создаёт и возвращает объект функции, который будет вызываться позднее, не присваивая ему имени

`lambda` аргумент1, аргумент2, ...: выражение, использующее аргументы

Используется для сокращения кода в тех местах, где включение оператора `def` не разрешено синтаксисом.

Функция map

`map(function, iterable, ...)`

Возвращает *итератор*, применяющий функцию к каждому элементу итерируемого объекта

Функция filter

`filter(function, iterable, ...)`

Возвращает *итератор*, с теми объектами последовательности, для которых функция вернула True.

Функция reduce (модуль functools)

```
functools.reduce(function, iterable[, initializer])
```

Применяет функцию к элементам итерируемого объекта *кумулятивно* (накопительно): сначала - к первым двум элементам (либо к отдельно заданному начальному значению и первому элементу), далее - к промежуточному результату и очередному значению

Функция zip

```
zip(*iterables, strict=False)
```

Соединяет элементы итерируемых объектов в кортежи.

Параметр `strict` (добавлен в 3.10) приводит к исключению, если длина объектов отличается.

Модули

Модуль Python - отдельный файл с кодом, который можно повторно использовать в других программах.

Преимущества модулей:

- многократное использование кода
- разбиение пространства имён системы
- реализация разделяемых служб или данных