Занятие 6

Лямбда функции

λ-исчисление

Лямбда исчисление — это формальная система в математической логике для описания вычислений.

Лямбда исчисление позволяет вычислять все, что может быть вычислено.

(полный по Тьюрингу)

λx.t

$$\lambda z.(\lambda x.z\ (x\ x))\ (\lambda x.z\ (x\ x))$$

Функциональное программирование

В функциональном программировании программа полностью состоит из вычисления чистых функций (**pure functions**).

Pure functions - это функция, выходное значение которой следует исключительно из входных значений без каких-либо наблюдаемых побочных эффектов.

```
def f(x):
...     return x * x
```

Вычисления выполняются вложенными или составными вызовами функций без изменений состояния или изменяемых данных.

```
calculate(sum, x, y).>=(100)
```

Функция как объекты первого порядка

Объекты первого порядка в контексте конкретного языка программирования - это элементы, с которыми вы можете делать все так же, как и с любым другим объектом: передавать как параметр, возвращаться из функции и присваивать переменной.

Функции высокого порядка

Функция высшего порядка - это функция, которая выполняет по крайней мере одно из следующих действий:

- принимает одну или несколько функций в качестве аргументов
- в качестве результата возвращает функцию.

```
def f(x):
    return x + 3

def g(function, x):
    return function(x) * function(x)

print (g(f, 7))
```

Функциональное программирование возможно на языке, который поддерживает:

1. Принять в качестве аргумента функции другую функцию

```
def calculate(action, x, y):
    return action(x, y)
```

Функциональное программирование возможно на языке, который поддерживает:

1. Принять в качестве аргумента функции другую функцию

```
def calculate(action, x, y):
    return action(x, y)
```

2. Вернуть другую функцию из функции

```
def get_printer(type):
    if “Cat” == type: return cat_print
    if “Dog” == type: return dog_print
    return placeholder
```

```
def cat_print():
    print(“Hello Cat”)

def dog_print():
    print(“Hello Dog”)

def placeholder():
    print(“Unknown”)

def get_printer(type):
    if “Cat” == type: return cat_print
    if “Dog” == type: return dog_print
    else: return placeholder

action = get_printer(animal)
action()
```

Все объекты

```
x = 5
>>> type(x)
<class 'int'>
```

```
x = 5
>>> type(x)
<class 'int'>
```

```
>>> def f(x):
...     return x * x
...
>>> type(f)
<class 'function'>
```

```
x = 5
>>> type(x)
<class 'int'>
```

```
>>> def f(x):
...     return x * x
...
>>> type(f)
<class 'function'>
```

```
>>> f = lambda x: x * x
>>> type(f)
<class 'function'>
```

Лямбда с несколькими аргументами

```
>>> f = lambda x, y: x * y
>>> f(5, 2)
10
```

Лямбда-функция без аргументов

```
>>> f = lambda: “Hello”
>>> f()
Hello
```

```
>>> def twice(f):
...     def result(a):
...         return f(f(a))
...     return result

>>> plusthree = lambda x: x + 3

>>> g = twice(plusthree)

>>> g(7)
 13
```

Лямбда-функции в Python могут состоять только из одной инструкции

1. Функциональное программирование не использует цикл через стандартный оператор
2. В функциональном программировании цикл организован рекурсией.
3. Рекурсия используется через оператор-выражение if:

```
>>> lambda x: x + 2 if x > 0 else x*2
```

Для рекурсии вам нужна ссылка на хранилище для анонимной функции, которая использует эту ссылку в лямбда-функции.

```
fact = lambda x: 1 if x == 0 else x * fact(x-1)
```

```
def foo(y):
    return x
```

```
foo = lambda y: x
```

```
full_name = lambda first, last: f'Full name: {first} {last}'


full_name('guido', 'van rossum')
```

```
_ = lambda x, y: x + y


_(1, 2)
```

```
(lambda x1, x2, x3: (x1 + x2 + x3) / 3)(9, 6, 6)
```

```
def func(x):
    return x, x ** 2, x ** 3

func(15)
```



```
(lambda x: x, x ** 2, x ** 3)(15)
```

```
print(f"###{(lambda s: s[::-1])('dlroW olleH')}###")
```

list comprehensions

```
[ <expression> for <element> in <iterable> ]

numbers = [x for x in (1, 2, 3, 4)]

squares = [x * x for x in numbers]


[ <expression> for <element> in <iterable> if <condition> ]

evens = [x * x for x in numbers if 0 == x % 2]
```

```
l = [v for x in range(10) for v in (lambda size: [random.randint(1, 100) for i in range(size)])(x) if v > 10]
```

Практика

1. Создать лямбда функцию для создания списка заполненного случайными числами, размером size.
2. Создать функцию pow(list) возведения в квадрат каждого элемента последовательности.
3. Попробуйте сделать такую же лямбда функцию.

4. Вывести исходный и измененный списки на экран

map()

map() — это встроенная функция Python.

Принимает в качестве аргументов **функцию** и **последовательность**.

map применяет переданную ей функцию к каждому элементу последовательности.

```
>>> L = [1, 2, 3, 4]
>>> list(map(lambda x: x**2, L))
[1, 4, 9, 16]
```

Практика

1. Добавьте функцию filterOdd(list) которая принимает список и уберет из него все нечетные элементы.
2. Попробуйте написать такую же лямбда функцию.

filter()

filter() — встроенная функция, которая фильтрует последовательность итерируемого объекта.

Если функция возвращает — True, элемент остается. В противном случае — отклоняется.

```
L = [1, 3, 2, 5, 20, 21]
list(filter(lambda x: not x % 2 == 0, L))
```

Домашнее задание

1. Вам дан код сортировки пузырьком, однако он работает неверно, там допущена ошибка. Используя дебагер, найдите и исправьте ошибку.

   Опишите в комментарии к коду, в чем была ошибка.

   Добавьте минимум 5 тестов для этого кода

```
1  def buble_sort(l):
2      t = l.copy()
3      for n in range(0, len(t)):
4          for i in range(len(t) - n):
5              if t[i] > t[n]:
6                  t[i], t[n] = t[n], t[i]
7      return t
8
9  nums = [4, 1, 6, 3, 2, 7, 8]
10 sorted = buble_sort(nums)
11 print(sorted)
```

2. Представьте себе бухгалтерскую книгу в книжном магазине. В этой книге хранятся записи о номере заказа, названии книги, количестве и стоимости одной книги, как представлено ниже, в таблице.

Order Number	Book Title and Author	Quantity	Price per Item
34587	Learning Python, Mark Lutz	4	40.95
98762	Programming Python, Mark Lutz	5	56.80
77226	Head First Python, Paul Barry	3	32.95
88112	Einfuhrung in Python3, Bernd Klein	3	24.99

Напишите программу на Python, которая на вход получает список списков:

```
[
      [34587, 'Learning Python, Mark Lutz', 4, 40.95],

      [98762, 'Programming Python, Mark Lutz', 5, 56.80],

      [77226, 'Head First Python, Paul Barry', 3, 32.95],

      [88112, 'Einfuhrung in Python3, Bernd Klein', 3, 24.99]

]
```

и возвращает список кортежей. Каждый кортеж состоит из номера заказа и произведения цены на товары и количества. Стоимость товара должна быть увеличена на $10, если стоимость заказа меньше $100.

Программа должна использовать lambda и map.

3. Добавьте к исходному списку еще несколько товаров

```
[
    [24387, ’ н а  р у с с к о м’, 2, 4.59],
    [18762, ’The C Programming Language (2nd Edition)’, 12, 78.80],
    [87236, ’C Programming Absolute Beginner’s Guide’, 1, 23.55],
    [58132, ’Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14’, 9, 42.89]
]
```

4. Отсортируйте получившийся список по стоимости и выведите его на экран.
***Используйте lambda**

5. Используя filter() оставьте только книги, количество которых больше 5ти.