Функции 1

Определение, аргументы, упаковка и распаковка, области видимости, PEP 8

def, return

```
def avg(a, b):
    return (a + b)/2


avg(5, 12)  # 8.5
avg  # <function __main__.avg(a, b)>


def avg(a, b):
    (a + b)/2


avg()  # None
```

Имена функций:

- с маленькой буквы
- допускаются подчеркивания и цифры (но не в начале имени)

return

```
def avg(a, b):
    if a < 0 or b < 0:
        print('give me normal numbers, please')
        return
    return (a + b)/2
    print('never gonna make it')


avg(-1, 6)  # 'give me normal numbers, please'

avg(3, 4)  # 3.5
```

Keyword arguments

```
def avg(a, b, weight_a=1, weight_b=1):
    return (a*weight_a + b*weight_b)/2



avg(4, 5)  # 4.5
avg(4, 5, weight_a=0, weight_b=1)  # 2.5
avg(4, 5, 0, 1)  # 2.5

avg(b=44, a=8, weight_a=1, weight_b=0)  # 4
avg(b=44, a=8, 1, 0)  # SyntaxError: positional argument follows keyword arg...
```

Инициализация значений по умолчанию

```
def func(arg, lst=[]):
    lst.append(arg)
    return lst



func(1)  # [1]
func(2)  # [1, 2]
func(3)  # [1, 2, 3]
func(1, ['a'])  # ['a', 1]
```

Инициализация значений по умолчанию

```
def func(arg, lst=None):
    lst = lst or []
    lst.append(arg)
    return lst


func(1)  # [1]
func(2)  # [2]
func(3)  # [3]
func(1, ['a'])  # ['a', 1]
```

Упаковка

```
def avg(*args):
    return sum(args)/len(args)



avg(1, 2, 3, 4, 5, 100)  # 19.166666666666668

avg()  # ZeroDivisionError: division by zero
```

Упаковка

```
def avg(first, *numbers):
    numbers = (first,) + numbers
    return sum(numbers)/len(numbers)


avg()  # TypeError: avg() missing 1 required positional arg...
```

Упаковка и распаковка

```
first, *middle, last = '1 2 3 4 5 6 7 8'.split(' ')

first  # '1'
middle  # ['2', '3', '4', '5', '6', '7']
last  # '8'

x, y = 'x y'.split(' ')
x, y = y, x

x  # 'y'
y  # 'x'
```

Распаковка

```
a, b = 'ab'
a  # 'a'
b  # 'b'

x, [a, b, c, d], y = 'x', 'abcd', 'y'
x  # 'x'
a  # 'a'

['0', *'abcd']  # ['0', 'a', 'b', 'c', 'd']
```

Для словарей и именованных аргументов

```
d = {'a': 100, 'b': 200}

{'a': 1, **d}  # {'a': 100, 'b': 200}


def function(*args, **kwargs):
    pass
```

Функция как объект

```
def avg(first, *numbers):
    '''return arithmetic mean of all arguements'''
    numbers = (first,) + numbers
    return sum(numbers)/len(number)



avg.__name__  # 'avg'
avg.__doc__  # 'return arithmetic mean of all arguements'

avg.attr = 'attribute'
avg.attr  # 'attribute'
```

Области видимости - LEGB

```
global_var = 0


def func(arg1, arg2):
    var = 'some variable'

    def print_vars():
        inner_var = 1
        print('inner_var', inner_var)  # local
        print('var', var)  # enclosing
        print('global_var', global_var)  # global
        print('list', list)  # built-in
    print_vars()
```

```
# inner_var 1
# var some variable
# global_var 0
# list <class 'list'>
```

Области видимости

```
global_var = 0


def func(arg1, arg2):
    var = 'some variable'

    def print_vars(arg):
        inner_var = 1
        print(locals())
        print(globals())
    print_vars('arguement')

# {'inner_var': 1, 'arg': 'arguement'}
# {'__name__': '__main__', ..., 'global_var': 0}
```

Области видимости

```
global_var = 0


def func():
    global_var = 1


global_var  # 0
func()
global_var  # 0
```

Области видимости

```
global_var = 0


def func():
    global_var = global_var + 1


global_var  # 0
func()  # UnboundLocalError: local variable 'global_var'
# referenced before assignment
```

Области видимости, оператор global

```
global_var = 0


def func():
    global global_var
    global_var = global_var + 1


global_var  # 0
func()
global_var  # 1
```

Области видимости, оператор nonlocal

```
def func():
    var = 0

    def inner():
        nonlocal var
        var += 1

    inner()
    print('var', var)


func()  # var 1
```

Аннотации

```
def is_palindrome(input_str: str) -> bool:
    len_s = len(input_str)
    iter_num = len_s // 2
    r = True
    for i in range(iter_num):
        if input_str[i] != input_str[len_s - i - 1]:
            r = False
    return r
```

PEP 8

- PEP – Python enhancement proposal
- Программные тексты – описывают нововведения/требования/практики написания кода
- Описывают также почему и зачем в языке происходят те или иные изменения

- PEP 8 – про то, как лучше оформлять код

PEP 8

```
some_constant = 'postgres:5432'

def Avg(a, b, weight_a = 1, weight_b = 1):
    m=(a*weight_a+b*weight_b)/2
    return m

Avg(4,5, weight_a = 2)
```

```
SOME_CONSTANT = 'postgres:5432'


def avg(a, b, weight_a=1, weight_b=1):
    m = (a*weight_a + b*weight_b)/2
    return m


avg(4, 5)
```

Naming и другое

- Функция должна делать одну вещь (логически)
- Функции на на несколько экранов – это ужасно
- Имя функции должно максимально ясно и коротко отражать то, что она делает.
- Часто лучше длинное, но корректное и понятное название, чем короткое и туманное.
- Часто лучше, чтобы имя функции строилось от глагола