Функции в python

Основы

- Задаётся с помощью ключеевого слова **def**, затем следует название функции, после чего в круглых скобках входные параметры.
- Тело функции идёт с отступом.
- Функция возвращает параметры с помощью ключевого слова **return**.
- Функция - своего рода набор инструкций, которые выполняются в момент вызова функции в основном коде

Зададим функцию, вычисляющую сумму двух чисел

In [ ]:
```
def sum(x,y):
    return x+y
```

Вызовем функцию и передадим ей два параметра

In [ ]:
```
sum(3,5)
```

Out[ ]:
```
8
```

Функция необязательно должна возвращать какое-либо значение, она может лишь исполнять код в теле

In [ ]:
```
def print_hw():
    print('хеллоу ворлд')
```

In [ ]:
```
print_hw()
```

```
хеллоу ворлд
```

Необязательные параметры

Мы можем указать "значение по умолчанию" для каких-то параметров. Тогда их необязательно передавать при вызове функции

In [ ]:
```
def mult(x,k=3):
    return x*k
```

In [ ]:
```
print(mult('_abc_'))
print(mult('_abc_',2))
```

```
_abc__abc__abc_
_abc__abc_
```

Заглушка

Если какая-то часть кода будет дописана позже, можно использовать временную заглушку - **pass**

In [ ]:
```
def change_num(x):
    if x%2==0:
        pass
    else:
        return x-1
```

In [ ]:
```
print(change_num(4))
print(change_num(5))
```

```
None
4
```

Множественный **return**

У функции может быть несколько точек выхода

In [ ]:
```
def change_num(x):
    if x%2==0:
        return x*2
    return x-1
```

In [ ]:
```
print(change_num(4))
print(change_num(5))
```

```
8
4
```

Вывод нескольких значений

Возможны несколько вариантов:

- кортеж
- список
- словарь

1. Кортеж

In [ ]:
```
def numbers_tuple(x):
    a=x//100
    b=(x-a*100)//10
    c=x%10
    return (a,b,c)
```

In [ ]:
```
numbers_tuple(123)
```

Out[ ]:
```
(1, 2, 3)
```

2. Список

In [ ]:
```
def numbers_list(x):
    a=x//100
    b=(x-a*100)//10
    c=x%10
    return [a,b,c]
```

In [ ]:
```
numbers_list(123)
```

Out[ ]:
```
[1, 2, 3]
```

3. Словарь

In [ ]:
```
def numbers_dict(x):
    a=x//100
    b=(x-a*100)//10
    c=x%10

    d=dict()
    d['a']=a
    d['b']=b
    d['c']=c
    return d
```

In [ ]:
```
numbers_dict(123)
```

Out[ ]:
```
{'a': 1, 'b': 2, 'c': 3}
```

In [ ]:
```
d=numbers_dict(123)
d['a']
```

Out[ ]:
```
1
```

Анонимные функции

In [ ]:
```
func = lambda x, y: x + y
func(1, 2)
```

Out[ ]:
```
3
```

In [ ]:
```
func = lambda x: str(x)
func(1)
```

Out[ ]:
```
'1'
```

In [ ]:
```
import random

mass=[]
for _ in range(10):
    mass.append(random.randint(0,100))

print(mass)
```

```
[42, 94, 78, 9, 56, 37, 93, 33, 70, 20]
```

Функция map()

Функция **map()** принимает два аргумента: функцию и аргумент составного типа данных, например, список.

map() применяет к каждому элементу списка переданную функцию

In [ ]:
```
new_mass = list(map(lambda x: x * 2 , mass))
print(new_mass)
```

```
[84, 188, 156, 18, 112, 74, 186, 66, 140, 40]
```

Функция filter()

Функция **filter()** принимает в качестве аргументов функцию и последовательность, которую необходимо отфильтровать.

Функция, передаваемая в **filter()**, должна возвращать значение *True* / *False*.

In [ ]:
```
new_mass = list(filter(lambda x: (x%2 == 0) , mass))
print(new_mass)
```

```
[42, 94, 78, 56, 70, 20]
```

Функция reduce()

Функция **reduce()** принимает 2 аргумента: функцию и последовательность.

reduce() последовательно применяет функцию-аргумент к элементам списка, возвращает единичное значение.

В Python 2.x функция reduce доступна как встроенная, в Python 3 она была перемещена в модуль functools.

reduce(**lambda** a, b: a + b, [12, 25, 3, 4]) = (((12+25)+3)+4)

In [ ]:
```
from functools import reduce
reduce(lambda a, b: a + b, mass)
```

Out[ ]:
```
532
```