Python

4

Функции

Функция это блок организованного, многократно используемого кода, который используется для выполнения конкретного задания. Функции обеспечивают лучшую модульность приложения и значительно повышают уровень повторного использования кода.

Простая функция

```
def add(x, y):

    return x + y
```

Инструкция **return** говорит, что нужно вернуть значение. В нашем случае функция возвращает сумму x и y.

```
>>> add(1, 10)

11

>>> add('abc', 'def')

 'abcdef'
```

Функция может быть любой сложности и возвращать любые объекты (списки, кортежи, и даже функции!)

```
def my_func(a, b):
    summ = a + b
    print(f'{a} + {b} = {summ}')
    return summ
my_func(4, 5)
```

Пример

Написать функцию, которая получает на вход имя и выводит строку вида: “Hello, {name}”. Создать список из 5 имен. Вызвать функцию для каждого элемента списка в цикле.

Аргументы функции

Функция может принимать произвольное количество аргументов или не принимать их вовсе. Также распространены функции с произвольным числом аргументов, функции с позиционными и именованными аргументами, обязательными и необязательными.

```
>>> def func(a, b, c=2): # c - необязательный аргумент

...     return a + b + c

>>> func(1, 2)  # a = 1, b = 2, c = 2 (по умолчанию)
5
>>> func(1, 2, 3)  # a = 1, b = 2, c = 3
6
>>> func(a=1, b=3)  # a = 1, b = 3, c = 2
6
>>> func(a=3, c=6)  # a = 3, c = 6, b не определен

Traceback (most recent call last):
  File "", line 1, in
    func(a=3, c=6)
TypeError: func() takes at least 2 arguments (2 given)
```

Именованные аргументы

```
def my_pow(number, power):
    result = number ** power + 1
    return result


result = my_pow(power=3, number=5)
print(result)
```

Функция также может принимать переменное количество позиционных аргументов, тогда перед именем ставится *

```
>>> def func(*args):
...     return args
…
>>> func(1, 2, 3, 'abc')
(1, 2, 3, 'abc')
>>> func()
()
>>> func(1)

 (1,)


a,*b=1,2,3,4
```

Функция может принимать и произвольное число именованных аргументов, тогда перед именем ставится **:

```
>>> def func(**kwargs):
...     return kwargs
...
>>> func(a=1, b=2, c=3)
{'a': 1, 'c': 3, 'b': 2}
>>> func()
{}
>>> func(a='python')
{'a': 'python'}
```

Обобщенное определение функции

```
def full_func(*args, **kwargs):
    print(args)
    print(kwargs)
full_func(1,2,3,a=4,b=5,c=6)
```

Анонимные функции, инструкция lambda

```
>>> func = lambda x, y: x + y
>>> func(1, 2)    3
>>> func('a', 'b')
'ab'
>>> (lambda x, y: x + y)(1, 2)
3
>>> (lambda x, y: x + y)('a', 'b')
'ab'
```

Пример

```
>>> func = lambda *args: args

>>> func(1, 2, 3, 4)

 (1, 2, 3, 4)
```