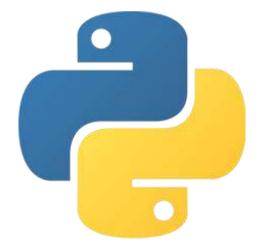
# Язык программирования Python

11 Функции



### Функция

Функция — это блок кода, выполняющий определенную задачу и который можно повторно использовать в коде программы.

До этого мы уже пользовались такими функциями, как print, input, len и так далее.

Мы можем создавать и свои функции. Для этого функцию необходимо определить.

#### Определение функции

Определение функции начинается с ключевого слова def, затем следует имя функции, круглые скобки, внутри которых могут быть параметры, и двоеточие, после которого идет тело функции.

То, что указано в квадратных скобках, является необязательным.

#### Вызов функции

Напишем простейшую функцию.

```
def say_hi():
    print('hi')
```

Функция называется say\_hi, у нее нет параметров. Само по себе определение не запускает функцию. Чтобы заставить ее выполниться, необходимо вызвать функцию.

Для вызова функции нужно указать имя функции, затем указать скобки и аргументы внутри них, если у функции есть параметры.

## Вызов функции

```
def say hi(): # Определение функции
    print('hi')
say hi() # Вызов функции say hi
say hi()
say hi()
Вывод:
hi
hi
hi
```

#### Функция с параметрами

```
def say hi(name): # Определение функции с параметром name
    print('hi, ', name)
say hi('Petya') # Вызов функции say hi с аргументом 'Petya'
say hi('Kolya')
say hi('Masha')
Вывод:
hi, Petya
hi, Kolya
hi, Masha
```

#### Параметры со значениями по умолчанию

Некоторые параметры функции можно сделать необязательными, указав для них значения по умолчанию в определении функции.

```
def say hi(name = 'Ivan'): # Определение функции с
параметром пате со значением по умолчанию
      print('hi, ', name)
  say hi()
  say hi('Petya')
  say hi('Masha')
  Вывод:
  hi, Ivan
  hi, Petya
```

hi, Masha

#### Именованные параметры

При вызове функции с параметрами мы передаем аргументы в порядке следования параметрам.

```
def say_hi(first_name, second_name):
    ...
say_hi('Ivan', 'Ivanov')
```

Первому параметру first\_name передается аргумент 'Ivan', второму — 'Ivanov'.

С помощью именованных параметров можно поменять порядок аргументов.

```
def say_hi(first_name, second_name):
    ...
say hi(second name = 'Ivanov', first name = 'Ivan')
```

#### Возвращение значения

Функция может возвращать значение. Для этого используется оператор return.

```
def sum(a, b):
    return a + b # возвращаемое значение — результат вычисления выражения a + b

result = sum(100, 200)
print(result)

Вывод:
```

300

#### Возвращение нескольких значений

В Python функция может возвращать сразу несколько значений.

```
def get_numerals(): # Получить цифры return 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

my_tuple = get_numerals() x0, x1, x2, x3, x4, x5, x6, x7, x8, x9, = get_numerals() print('Кортеж:', my_tuple) print('Переменные:', x0, x1, x2, x3, x4, x5, x6, x7, x8, x9)
```

#### Вывод:

Кортеж: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) Переменные: 0 1 2 3 4 5 6 7 8 9

#### Итоги

Мы познакомились с такими понятиями, как функция, значение по умолчанию, именованные параметры, оператор return, возвращаемое значение,

а также попрактиковались в работе с интерпретатором в файловом режиме.