# Функции

Функция в программировании — это именованный фрагмент кода, предназначенный для многократного использования.

Функции бывают:

• встроенные — написанные разработчиками Python или авторами модулей, например, str(), print(), type();

• кастомные — те, что разработчик написал самостоятельно.

Чтобы написать функцию нужно:

• придумать ей информативное имя;

• объявить её;

• описать тело функции — код, который будет выполняться при вызове.

Тело функции отбивается четырьмя пробелами от начала строки, в которой объявлена эта функция.

```
def best_function_in_this_code():     # Объявление функции.
    print('Других функций тут нет!')  # Тело функции.

best_function_in_this_code()          # Вызов функции.
best_function_in_this_code()          # Ещё один вызов функции.

# В результате двух вызовов функции будет напечатано:
# Других функций тут нет!
# Других функций тут нет!
```

## Зачем нужны функции

Функции пишут так, чтобы они обрабатывали входящие данные и возвращали значение, результат работы.

Входящие данные функция принимает в параметры — переменные, обрабатываемые в теле функции. Параметры указываются после имени функции в скобках, через запятую.

```
# Параметры функции — a и b
def never_print_result(a, b):
```

Значения, передаваемые в функцию при вызове, называют аргументами функции.

```
def never_print_result(a, b):
    ...

# Вызываем функцию с аргументами 5 и 4
never_print_result(5, 4)
```

Переданные в функцию аргументы можно обрабатывать только в теле функции, получить доступ к параметрам функции извне нельзя.

Функция обрабатывает входящие данные и получает какой-то результат. Этот результат функция может вернуть — передать для дальнейшей обработки вне этой функции.

В теле функции возвращаемое значение указывается после ключевого слова return. После выполнения инструкции return функция завершит выполнение.

```
def print_recommendation(rating):
    if rating > 4.7:
        return 'Фильм крут'
    if rating > 3.5:
        return 'Смотреть можно'
    # Если не сработало ни одно условие -
    # функция вернёт это сообщение:
    return 'Так себе киношечка'


result = print_recommendation(4.1)
```

Функция не обязательно должна возвращать какие-то значения: она может выполнить какую-то работу (например, вывести строку на печать) — и завершиться.

```
def super_print(movies):
    print('Подготовка к печати')
    print(movies)
    print('Готово!')

movies_names = ['Хакеры', 'Сеть', 'Трон', 'Матрица']
# Вызов функции не нужно передавать в переменную:
# функция ничего не вернёт, записывать в переменную нечего.
super_print(movies_names)
```

## Работа с функциями

### Значения параметров по умолчанию

Применяются, если при вызове параметр не передан.

```
# При указании значения по умолчанию
# пробелы вокруг оператора в выражении b=1 не нужны. Так говорит PEP8.
def get_mod_diff(a, b=1):
    """Функция возвращает результат разницы полученных значений по модулю."""
    diff = abs(a - b)
    return diff


x = 3
y = 4
print(get_mod_diff(x))
# Вывод в терминал: 2
```

### Позиционные и именованные параметры

При вызове функции можно передавать аргументы, указывая имена параметров. Порядок именованных аргументов при вызове не важен.

```
# При указании значения по умолчанию
# пробелы вокруг оператора в выражении b=1 не нужны. Так говорит PEP8.
def get_mod_diff(a, b=1):
    """Функция возвращает результат разницы полученных значений по модулю."""
    diff = abs(a - b)
    return diff


x = 3
y = 4
# Именованные аргументы можно передавать в любом порядке.
print(get_mod_diff(b=y, a=x))
```

### Пустая функция

Создаётся с помощью ключевого слова pass...

```
def empty():
    pass
```

...или многоточия Ellipsis:

```
def empty():
    ...
```