**Функции в Python**

Оглавление

[Именные функции, инструкция def 1](#_Toc37966986)

[Аргументы функции 2](#_Toc37966987)

[Анонимные функции, инструкция lambda 4](#_Toc37966988)

# Именные функции, инструкция def

Функция в python - объект, принимающий аргументы и возвращающий значение. Обычно функция определяется с помощью инструкции **def**.

Определим простейшую функцию:

**def** add(x, y):

**return** x + y

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

Теперь мы ее можем вызвать:

**>>>** add(1, 10)

11

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

'abcdef'

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

**>>> def** newfunc(n):

**...**  **def** myfunc(x):

**...**  **return** x + n

**...**  **return** myfunc

**...**

**>>>** new = newfunc(100) *# new - это функция*

**>>>** new(200)

300

Функция может и не заканчиваться инструкцией return, при этом функция вернет значение [None](https://pythonworld.ru/tipy-dannyx-v-python/none.html" \t "_blank):

**>>> def** func():

**...**  **pass**

**...**

**>>>** print(func())

None

# 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)

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,)

Как видно из примера, args - это [кортеж](https://pythonworld.ru/tipy-dannyx-v-python/kortezhi-tuple.html) из всех переданных аргументов функции, и с переменной можно работать также, как и с кортежем.

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'}

В переменной kwargs у нас хранится [словарь](https://pythonworld.ru/tipy-dannyx-v-python/slovari-dict-funkcii-i-metody-slovarej.html), с которым мы, опять-таки, можем делать все, что нам заблагорассудится.

# Анонимные функции, инструкция lambda

Анонимные функции могут содержать лишь одно выражение, но и выполняются они быстрее. Анонимные функции создаются с помощью инструкции **lambda**. Кроме этого, их не обязательно присваивать переменной, как делали мы инструкцией def func():

**>>>** 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'

lambda функции, в отличие от обычной, не требуется инструкция return, а в остальном, ведет себя точно так же:

**>>>** func = **lambda** \*args: args

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

(1, 2, 3, 4)