Python: Параметры функций

Функции могут не только возвращать значения, но и принимать параметры. В этом уроке мы научимся создавать такие функции.

Напомним, что с параметрами функций мы уже сталкивались:

```
# Принимает на вход один параметр любого типа
print('я параметр')
# Принимает на вход два строковых параметра
# первый — что ищем, второй — на что меняем
'google'.replace('go', 'mo')  # moogle
# Принимает на вход два числовых параметра
# первый — округляемое число, второй — число знаков после з
round(10.23456, 3)  # 10.235
```

А теперь представим, что нам нужно реализовать функцию `get_last_char()`, которая возвращает последний символ в строке, переданной ему на вход как параметр.

Вот как будет выглядеть использование этой функции:

```
# Передача параметров напрямую без переменных
get_last_char("Hexlet")  # t
# Передача параметров через переменные
```

```
name1 = 'Hexlet'
get_last_char(name1)  # t
name2 = 'Goo'
get_last_char(name2)  # o
```

Из этого примера можно сделать следующие выводы:

- Нам нужно определить функцию `get_last_char()`
- Функция должна принимать на вход один параметр строкового типа
- Функция должна возвращать значение строкового типа

Определяем функцию:

```
def get_last_char(text):
    return text[-1]
```

В скобках указывается имя переменной `text`, которая служит параметром. Имя параметра может быть любым. Главное, чтобы оно отражало смысл значения, которое содержится внутри. Например:

```
def get_last_char(string):
    return string[-1]
```

Значение параметра будет зависеть от вызова этой функции:

```
# Внутри функции string будет равна 'hexlet'
get_last_char('hexlet')  # t

# Внутри функции string будет равна 'code'
get_last_char('code')  # e

# Внутри функции string будет равна 'Winter is coming'
# Имя переменной снаружи не связанно с именем переменной в
text = 'Winter is coming'
get_last_char(text)  # g
```

https://replit.com/@hexlet/python-basics-define-functions-parameters

Параметр нужно обязательно указывать. Если вызвать функцию без него, то интерпретатор выдаст ошибку:

```
get_last_char()  # У такого кода нет смысла

TypeError: get_last_char() missing 1 required positional ar
```

Многие функции работают одновременно с несколькими параметрами. Например, чтобы округлить числа, нужно указать не только само число, но и количество знаков после запятой:

```
round(10.23456, 3)  # 10.235
```

То же самое относится и к методам. Они могут

требовать на вход любое количество параметров, которое им нужно для работы:

```
# Первый параметр — что ищем
# Второй параметр — на что меняем
'google'.replace('go', 'mo')  # moogle
```

Чтобы создать такие функции и методы, в определении нужно указать необходимое количество параметров через запятую. Еще им нужно дать понятные имена.

Ниже пример определения функции `replace()`, которая заменяет в слове одну часть строки на другую:

```
def replace(text, from, to):
    # Здесь тело функции, но мы его
    # опускаем, чтобы не отвлекаться

replace('google', 'go', 'mo')  # moogle
```

Когда параметров два и более, то практически для всех функций важен порядок передачи этих параметров. Если его поменять, то функция отработает по-другому:

```
# Ничего не заменилось,
# так как внутри google нет mo
```

```
replace('google', 'mo', 'go')  # google
```

Теперь вы знаете, как создавать функции, которые могут принимать на вход параметры.

Задание

Допишите функцию `truncate()`, которая обрезает переданную строку до указанного количества символов, добавляет в конце троеточие и возвращает получившуюся строку. Подобная логика часто используется на сайтах, чтобы отобразить длинный текст в сокращенном виде.

Функция принимает два параметра:

1. Строка, которую нужно обрезать
2. Число символов, которые нужно оставить

Пример того, как должна работать написанная вами функция:

```
# Передаём текст напрямую
# Обрезаем текст, оставляя 2 символа
truncate('hexlet', 2)  # 'he...'

# Через переменную
text = 'it works!'
# Обрезаем текст, оставляя 4 символа
truncate(text, 4)  # 'it w...'
```

Выполнить задание можно различными способами, подскажем лишь один из них. Для решения этим способом вам понадобится взять подстроку из строки, переданной первым параметром в функцию. Используйте для этого срезы строк. Подумайте, исходя из задания, с какого индекса и по какой вам надо извлечь подстроку?

```
word = 'welcome!'
index = 3
word[:index] # wel
```

- ▶ Упражнение не проходит проверку — что делать? 😶
- ▶ В моей среде код работает, а здесь нет 😐
- ▶ Мой код отличается от решения учителя 🤔
- ▶ Прочитал урок — ничего не понятно 🙄

Полезное

- [Параметры функции]https://ru.wikipedia.org/wiki/Параметр_(программирование))