Функции

Функции – это многократно используемые фрагменты программы. Они позволяют дать
имя определённому блоку команд с тем, чтобы впоследствии запускать этот блок по указанному имени в любом месте программы и сколь угодно много раз. Это называется вызовом функции. Мы уже использовали много встроенных функций, как то len и range.

Функции определяются при помощи зарезервированного слова def. После этого слова
указывается имя функции, за которым следует пара скобок, в которых можно указать
имена некоторых переменных, и заключительное двоеточие в конце строки. Далее следует блок команд, составляющих функцию. На примере можно видеть, что на самом деле
это очень просто:

```
def sayHello():
    print('Привет, Мир!')  # блок, принадлежащий функции
# Конец функции


sayHello()  # вызов функции
sayHello()  # ещё один вызов функции
```

Каков механизм работы?

Мы определили функцию с именем sayHello, используя описанный выше синтаксис. Эта функция не принимает параметров, поэтому в скобках не объявлены какие-либо переменные. Параметры функции – это некие входные данные, которые мы можем передать функции, чтобы получить соответствующий им результат.

Обратите внимание, что мы можем вызывать одну и ту же функцию много раз, а значит нет необходимости писать один и тот же код снова и снова.

Параметры функции

Функции могут принимать параметры, т.е. некоторые значения, передаваемые функции
для того, чтобы она что-либо сделала с ними. Эти параметры похожи на переменные, за
исключением того, что значение этих переменных указывается при вызове функции, и
во время работы функции им уже присвоены их значения.
Параметры указываются в скобках при объявлении функции и разделяются запятыми.
Аналогично мы передаём значения, когда вызываем функцию. Обратите внимание на
терминологию: имена, указанные в объявлении функции, называются **параметрами**, тогда как значения, которые вы передаёте в функцию при её вызове, – **аргументами**.

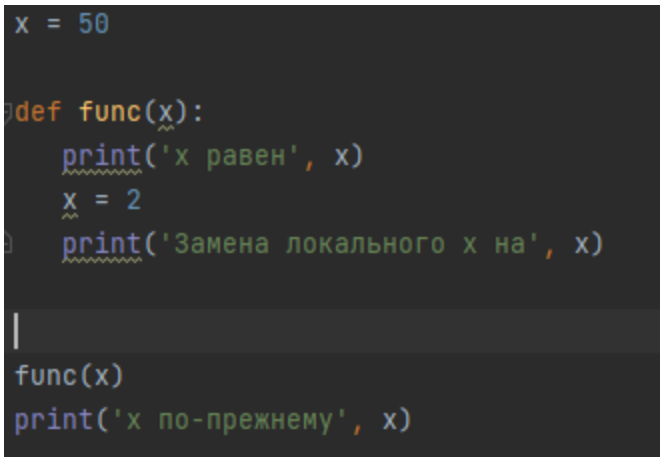
```
def printmax(f_n, s_n):

    if f_n > s_n:
        print(f'{a}, максимально')
    elif f_n == s_n:
        print(f'{f_n}, равно, {s_n}')
    else:
        print(f'{s_n} максимально')


printMax(3, 4)  # прямая передача значений
x = 5
y = 7
printMax(x, y)  # передача переменных в качестве аргументов
```

Каков механизм работы?

Здесь мы определили функцию с именем printmax, которая использует два параметра с именами f_n и s_n. Мы находим наибольшее число с применением простого оператора if..else и выводим это число.
При первом вызове функции printmax мы напрямую передаём числа в качестве аргументов. Во втором случае мы вызываем функцию с переменными в качестве аргументов. printmax(x, y) назначает значение аргумента x параметру a, а значение аргумента y – параметру b. В обоих случаях функция printmax работает одинаково.

Локальные переменные

При объявлении переменных внутри определения функции, они никоим образом не связаны с другими переменными с таким же именем за пределами функции – т.е. имена
переменных являются локальными в функции. Это называется областью видимости переменной. Область видимости всех переменных ограничена блоком, в котором они объявлены, начиная с точки объявления имени.



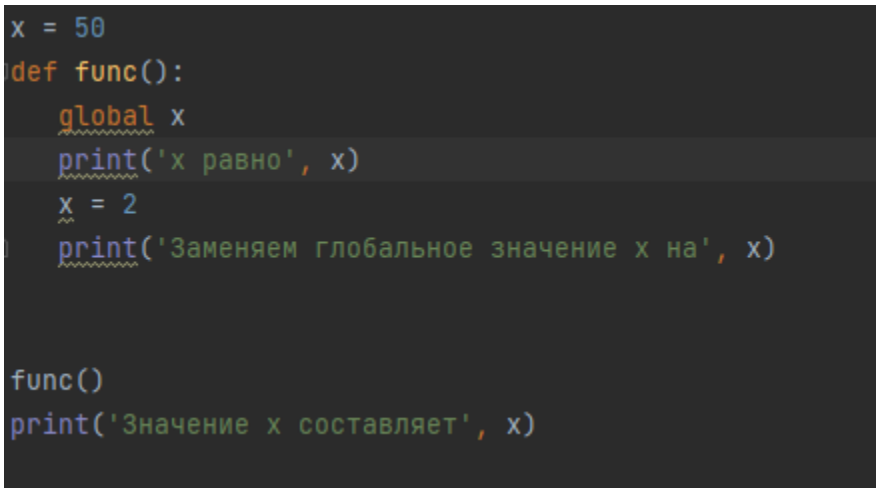
Каков механизм работы?

При первом выводе значения, присвоенного имени x, в первой строке функции Python использует значение параметра, объявленного в основном блоке, выше определения функции.
Далее мы назначаем x значение 2. Имя x локально для нашей функции. Поэтому когда мы заменяем значение x в функции, x, объявленный в основном
блоке, остаётся незатронутым.
Последним вызовом функции print мы выводим значение x, указанное в основном

блоке, подтверждая таким образом, что оно не изменилось при локальном присваивании значения в ранее вызванной функции.

Зарезервированное слово global

Чтобы присвоить некоторое значение переменной, определённой на высшем уровне программы (т.е. не в какой-либо области видимости, как то функции или классы), необходимо
указать Python, что её имя не локально, а глобально (global). Сделаем это при помощи зарезервированного слова global. Без применения зарезервированного слова global невозможно присвоить значение переменной, определённой за пределами функции.
Можно использовать уже существующие значения переменных, определённых за пределами функции (при условии, что внутри функции не было объявлено переменной с таким
же именем). Однако, это не приветствуется, и его следует избегать, поскольку человеку,
читающему текст программы, будет непонятно, где находится объявление переменной.
Использование зарезервированного слова global достаточно ясно показывает, что переменная объявлена в самом внешнем блоке.



Каков механизм работы?

Зарезервированное слово global используется для того, чтобы объявить, что
x – это глобальная переменная, а значит, когда мы присваиваем значение имени x

внутри функции, это изменение отразится на значении переменной x в основном блоке программы.
Используя одно зарезервированное слово global, можно объявить сразу несколько переменных: global x, y, z.

Значения аргументов по умолчанию

Зачастую часть параметров функций могут быть необязательными, и для них будут использоваться некоторые заданные значения по умолчанию, если пользователь не укажет
собственных. Этого можно достичь с помощью значений аргументов по умолчанию. Их
можно указать, добавив к имени параметра в определении функции оператор присваивания (=) с последующим значением. Обратите внимание, что значение по умолчанию должно быть константой. Или точнее говоря, оно должно быть неизменным – это объясняется подробнее в последующих главах. А пока запомните это.

```
def say(message, times=1):
    print(message * times)


say('Привет')
say('Мир', 5)
```

Каков механизм работы?

Функция под именем say используется для вывода на экран строки указанное число раз. Если мы не указываем значения, по умолчанию строка выводится один раз. Мы достигаем этого указанием значения аргумента по умолчанию, равного 1 для параметра times^2
При первом вызове say мы указываем только строку, и функция выводит её один раз. При втором вызове say мы указываем также и аргумент 5, обозначая таким образом, что мы хотим сказать^3 фразу 5 раз.