Функции

ЛЕКЦИЯ 2

Содержание

- Функции и их структура
- Инструкция **return**
- Область видимости: Локальные и глобальные переменные
- Инструкция **global**
- Вложенные функции

Основы функций

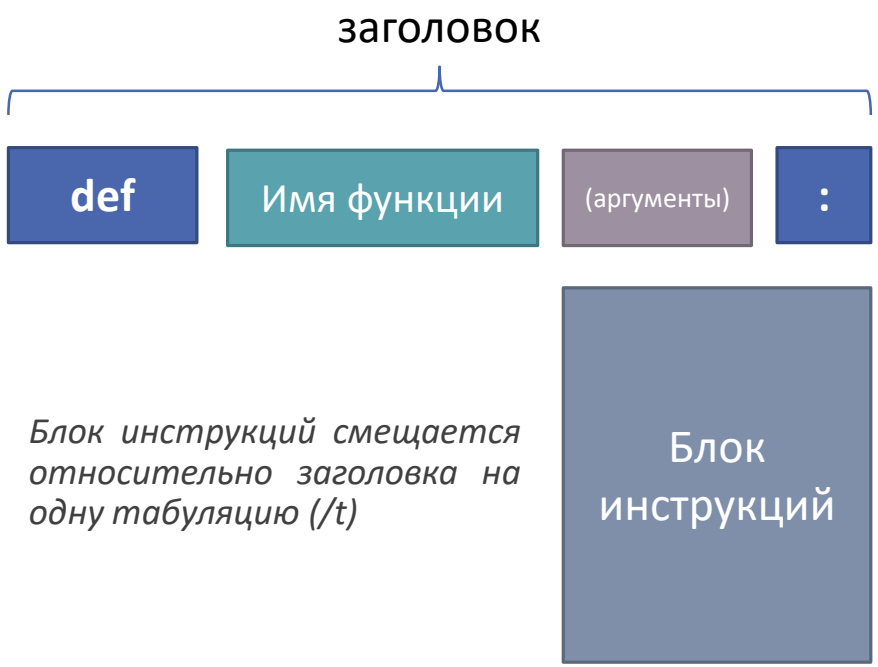
Функция – это средство позволяющее группировать наборы инструкций так, что в программе эти инструкции могут запускаться неоднократно.

- Функции представляют собой средство проектирования, которое позволяет разбить сложную систему на достаточно простые и легко управляемые части.
- Функции максимизируют многократное использование программного кода и минимизируют его избыточность

Создание функций

- Для создания функций используется инструкция **def**
- В отличии от компилируемые языков программирования, функция не существует до того момента пока компилятор не доберется до инструкции **def**. Таким образом в языке Python функции можно реализовывать внутри любых других инструкций (к примеру циклов, условий или других функций)
- Когда интерпретатор выполняет инструкцию **def** он создает новый объект – функцию и связывает его с именем функции. Таким образом имя становится ссылкой на объект функцию.

Структура функций



Функция представляет собой составную инструкцию начинающуюся с **def**. Она состоит из:

- Заголовка
- Блока инструкций, смещенного относительно заголовка на табуляцию

Примечание: вместо блока инструкций может идти единичная инструкция, которую можно разместить сразу после «:»

Заголовок функции

- Инструкция **def** определяет **имя** функции, которое будет связано с объектом функции.
- **Аргументы** функции заключаются в **круглые** скобки. Количество аргументов может быть 0 и более.
- Имена аргументов связываются в строке заголовка с соответствующими интересующими объектами передаваемыми в функцию

return

return

Возвращаемый объект

Блок инструкций функции (или **тело функции**) может содержать инструкцию **return**.

- Инструкция **return** может располагаться в любом месте в теле функции. Данная инструкция завершает работу функции и передает результат вызывающей программе.
- Объект указываемый после ключевого слова **return** представляет собой результат работы функции и возвращается основной программе.
- Инструкция return может использоваться без возвращаемого объекта. В этом случае результатом работы функции будет объект **None**
- Функция может не содержать инструкции **return**. В этом случае она завершит работу выполнив все инструкции в теле функции. В результате работы такой функции так-же будет создан объект **None**

Пример функции № 1

Приведенная ниже функция выводит в консоль «Hello world»

```
1  def foo():
2      print('Hello World!')
3
4  foo()
```

- инициализация функции должна осуществляться до её первого вызова

Пример функции № 2

Приведенная ниже функция выводит значение переданных ей аргументов в консоль

```
1  def foo(a, b):
2      print(a, b)
3
4
5  foo(5, 'hello')
```

Так как тело функции состоит только из одной инструкции, её можно поместить сразу после оператора «:»

```
1  def foo(a, b): print(a, b)
2
3
4  foo(5, 'hello')
```

Пример функции №3

В нижестоящем коду функция возвращает основной программе сумму переданных ей аргументов, которая впоследствии печатается в консоль посредством функции **print**()

```
1    def foo(a, b):
2        return a + b
3
4
5    print(foo(5, 7))
```

Пример функции №4

Данная функция выполняет печать своих аргументов, а инструкция **return** в данном случае создает объект None, который и выводится в консоль.

```
1  def foo(a, b):
2      print(a + b)
3      return
4
5
6  print(foo(5, 7))
```

Результат:

```
12
None
```

Если опустить инструкцию return объект None все равно создается

```
1  def foo(a, b):
2      print(a + b)
3
4
5  print(foo(5, 7))
```

Результат:

```
12
None
```

Области видимости

Локальные переменные – это имя которое доступно только программному коду внутри инструкции, в рамках которой оно было создано

Пространство имен – это область в которой находятся имена

- Аргументы функции являются локальными переменными данной функции

Области видимости

- Если присваивание переменной выполняется внутри инструкции def переменная является локальной для этой функции
- Если присваивание производится за пределами всех инструкций def, она является глобальной для всего файла

Пример использования «Области видимости»

```
1   X = 99  # глобальная переменная X
2
3
4   def foo():
5       X = 88  # локальная переменная X
6       print(X)
7
8
9   foo()
10  print(X)
```

Результат:

```
88
99
```

Инструкция global

Инструкция **global** позволяет использовать и определять переменную как глобальную внутри какой-либо составной инструкции

```
1   X = 99   # глобальная переменная X
2
3
4   def foo():
5       global X
6       X = 88   # глобальная переменная X
7       print(X)
8
9
10  foo()
11  print(X)
```

Результат:

```
88
88
```

Инструкция global

Нельзя использовать переменную как локальную в рамках составной инструкции, после чего переходить к её глобальной версии

```
1  X = 99
2
3
4  def foo():
5
6      X = 88
7      global X
8      X = 77
```

Вложенные функции

Инструкция **def** может быть вызвана в любой области видимости, в том числе и в рамках другой функции. В таком случае полученная функция будет локальной для того места где видна соответствующая инструкция **def**

```
1   X = 99  # глобальная переменная X
2
3
4   def foo():
5       def bar(var):
6           print(var)
7
8       bar(X)
9
10
11  foo()
12  # bar() недоступна
```