Дополнительные возможности в определении функций

Муськин А.В.
09-412

В языке Python можно определить функцию с переменным числом аргументов. Для этого существуют три способа, которые можно комбинировать:

Произвольное количество аргументов

Именованные аргументы

Произвольное количество именованных аргументов

Произвольное количество аргументов

Замечание:

Обозначается звездочкой перед аргументом - *args

Внутри функции выглядит как кортеж, элементы кортежа расположены в том же порядке, что и аргументы функции, указанные при вызове.

Передать список при вызове функции как набор аргументов можно, приписав к обозначению списка спереди звездочку

Пример:

```
1def a(*args):
2print type(args)
3print args
4
5a(1,2,3,4,5)
6
7>>> <type 'tuple'>
8>>> (1, 2, 3, 4, 5)
```

Именованные аргументы

Замечания:

При вызове указывать необязательно. Если не указаны, им присваиваются дефолтные значения.

Пример:

```
1 def a(b=4,c=5):
2 print b,c
3 a()
4 a(12,13)
5 a(b=15,c=16)
6 >>> 4 5
7 >>> 12 13
8>>> 15 16
```

Примечание:

При работе программы значения именованным параметрам присваиваются один раз, в месте определения функции. Если присваемый объект изменяемый, то измененный во время работы функции, он при в следующих вызовах будет иметь не то, значение, которое указано как значение в определении функции, а то, которое было присвоено во время предыдущего вызова.

Пример к примечанию:

```
1 def a(b=[1]):
2     b[0] += 1
3     b.append(1)
4     print b
5
6 a()
7 a()
8 a()
9
10 >>> [2, 1]
11 >>> [3, 1, 1]
12 >>> [4, 1, 1, 1]
```

Произвольное количество именованных аргументов

Замечания:

Обозначается двумя звездочками перед аргументом - **kwargs
Внутри функции выглядит как словарь, с ключами, соответствующими именам аргументов, указанными при вызове функции.
Передать словарь при вызове функции как набор именованных аргументов можно, приписав две звездочки перед обозначением словаря. Напримиер так: **kwargs

Пример:

```
1 def a(**kwargs):
2     print kwargs
3
4 a()
5 a(b=1,c=2)
6
7 >>> {}
8 >>> {'c': 2, 'b': 1}
```

Короткая форма

С помощью ключевого слова lambda Вы можете создать простую функцию без имени.

Например, функция, возвращающая сумму двух своих аргументов, может быть записана как 'lambda a, b: a+b'.

Короткая форма может быть использована везде, где требуется объект-функция. Её синтаксис ограничен одним выражением.

Документирование функции

В языке Python вы можете документировать функции, снабжая их *строками документации*.

```
def buildConnectionString(params):
    """Создает и возвращает строку соединения из словаря параметров."""
```

Утроенные кавычки используются для многострочных строковых литералов. Все, что находится между утроенными кавычками, является одним строковым значением, включая символы перехода на новую строку и другие символы кавычек.

Первая строка в определении функции является строкой документации, в которой поясняется, что функция делает. Python не требует наличия строки документации у функции, но все же ее стоит всегда определять.

Вызов Функции

Помимо описанного (*func(arg ...)*), язык Python предоставляет еще несколько способов вызова функций. Начиная с версии 1.6, Вы можете указать кортеж позиционных и словарь именованных аргументов, например:

```
args = ('Это очень мого', 'Это действительно очень много')
kwds = {'language': 'Python', 'author': 'Guido van Rossum')
example(1000000, *args, **kwds )
```

Такой же эффект можно получить используя встроенную функцию apply():

```
apply(example, (1000000,)+args, kwds)
```

Тест