Introducción a la programación en Python

Tema 5. Funciones

Autor: Francisco Mariño Ruiz

Contenido

1 Funciones

1.1 Definición

Una función, es la forma que tenemos en los lenguajes de programación de agrupar expresiones y sentencias que realizarán determinadas acciones. Las funciones solo se ejecutarán cuando sean llamadas y podrán ser reutilizadas dentro de un programa para ejecutarse todas las veces que se desee.

Las funciones son muy útiles para reutilizar código y no tener que escribir varias veces las mismas instrucciones. Son básicamente una forma de ser más eficientes cuando desarrollamos nuestros códigos.

Para definir funciones en Python utilizaremos la palabra clave "*def*" más un nombre de función lo más descriptivo posible para evitar confusiones, seguido de paréntesis. Además, al igual que en bucles y condicionales, requiere finalizar la instrucción con dos puntos e identación con una tabulación o cuatro espacios en las instrucciones que irán dentro de la función.

```
1  def holaMundo():
2      print ("Hola Mundo!")
```

Si escribimos ese código en nuestro programa, no hará nada, ya que de este modo hemos definido la función, pero no la hemos llamado. Para llamar a la función simplemente escribiremos el nombre de la función más los paréntesis.

```
1  def holaMundo():
2      print ("Hola Mundo!")
3
4  holaMundo()
```

```
Hola Mundo!

Process finished with exit code 0
```

Es importante decir que la llamada a la función nunca estará antes de su definición, por la naturaleza del lenguaje Python, que es un lenguaje interpretado y no compilado, las instrucciones se ejecutan según el intérprete las va leyendo, por lo que al llamar a la función, como aún no la habrá leído no podrá ejecutarla al no encontrarla y nos devolverá un error.

```
1  holaMundo()
2
3  def holaMundo():
4      print ("Hola Mundo!")
```

```
Traceback (most recent call last):
  File "C:/Users/fmari/OneDrive/Datos/CursoPython18/Ejemplos/funciones_basico.py",
line 1, in <module>
    holaMundo()
NameError: name 'holaMundo' is not defined

Process finished with exit code 1
```

La primera función que hemos definido no devuelve nada, simplemente ejecuta una sentencia. Sin embargo, en muchas ocasiones querremos que nuestras funciones devuelvan algún valor. En estos casos, para devolver valores al programa principal, tendremos que usar la palabra clave *"return"* seguida de la variable que queramos devolver como resultado de la función. Podrá tratarse de una variable o de varias.

Por ejemplo, en lugar de hacer que la función imprima la cadena de texto, vamos a decirle que nos devuelva la cadena, de este modo llamaremos a la función asignándola a una variable, que será la que almacene el resultado y finalmente, ya desde el programa principal, imprimiremos por pantalla la cadena de texto.

```
1  def holaMundo():
2      return "Hola Mundo!"
3
4  saludo = holaMundo()
5  print(saludo)
```

```
Hola Mundo!

Process finished with exit code 0
```

Otra opción muy interesante a la hora de utilizar funciones, es hacerlo añadiéndole parámetros de entrada. Estas variables podrán ser usadas dentro de la función como variables y la función podrá operar libremente con sus valores. A una misma función, podemos establecer que se le pasen tantos parámetros como queramos.

```
1  def saludar(nombre, apellido):
2      print ("Hola " + nombre + " " + apellido)
3
4  saludar("Pepe", "García")
```

```
Hola Pepe García

Process finished with exit code 0
```

Es importante remarcar, que las variables que creemos dentro de una función, serán variables de tipo local, es decir, estas variables tan sólo existen dentro de la función. Esto implica que si intentamos llamar a una variable de una función desde el programa principal vamos a obtener un error, ya que para el programa principal dicha variable no existe.

```
1  def saludar(nombre, apellido):
2      print ("Hola " + nombre + " " + apellido)
3
4  saludar("Pepe", "García")
5  print(nombre)
```

```
Hola Pepe García
Traceback (most recent call last):
  File "C:/Users/fmari/OneDrive/Datos/CursoPython18/Ejemplos/funciones_basico.py",
line 5, in <module>
    print(nombre)
NameError: name 'nombre' is not defined
```

```
Process finished with exit code 1
```

Al estar aisladas las variables locales (variables de la función) respecto a las variables globales (variables del programa principal), podemos nombrar variables del mismo modo dentro y fuera de la función, aunque esto a veces puede generar confusión a la hora de leer el código e interpretarlo.

```
1  def saludar(nombre, apellido):
2      print ("Hola " + nombre + " " + apellido)
3
4  nombre = "Pepe"
5  nombre = "García"
6  saludar(nombre, apellido)
7  print(nombre)
```

```
Hola Pepe García
Pepe

Process finished with exit code 0
```

En general, cuando llamemos a una función a la que queremos pasarle parámetros, deberemos de hacerlo escribiendo todos los parámetros en el mismo orden que hemos definido la función, ya que por defecto es como intentará leerla. En caso de que todos los parámetros de la función sean del mismo tipo de dato, la función se ejecutará correctamente, pero se producirá un problema a la hora de interpretarlos que hará que se desordene la asignación de los mismos dentro de la función.

```
1  def saludar(nombre, apellido):
2      print ("Hola " + nombre + " " + apellido)
3
4  saludar("García", "Pepe")
```

```
Hola García Pepe
```

En caso de que los parámetros de nuestra función sean de distinto tipo de dato, la función probablemente no se ejecutará y nos devolverá un error.

```
1  def precioFruta(fruta, precio_kilo, kilos):
2      total = precio_kilo * kilos
3      print("Las " + fruta + " valen " + str(total) + " euros")
4
5  precioFruta(2, "Manzanas", 5.95)
```

```
Traceback (most recent call last):
  File "C:/Users/fmari/OneDrive/Datos/CursoPython18/Ejemplos/funciones_basico.py",
line 5, in <module>
    precioFruta("Manzanas", 5.95, 2)
  File "C:/Users/fmari/OneDrive/Datos/CursoPython18/Ejemplos/funciones_basico.py",
line 3, in precioFruta
    total = precio_kilo * kilos
TypeError: can't multiply sequence by non-int of type 'float'

Process finished with exit code 1
```

Este error solo sucederá en caso de que la función utilice los valores de distinto tipo para hacer operaciones con ellos que se sólo se puedan hacer con ese tipo de dato.

```
def precioFruta(fruta, precio_kilo, kilos):
    total = precio_kilo * kilos
    print("Las " + fruta + " valen " + str(total) + " euros")

precioFruta("a", "Manzanas", 6)
```

```
Las a valen ManzanasManzanasManzanasManzanasManzanasManzanas euros

Process finished with exit code 1
```

Como se ve en el ejemplo anterior, aunque hemos introducido tipos de dato incorrecto en alguno de los parámetros, no se devuelve error, aunque el resultado no tenga sentido, esto se debe a que al introducir una cadena de texto y un número entero en la primera multiplicación, el lenguaje si que nos permite hacer esa multiplicación, aunque no tenga sentido para nuestro resultado. En función de la complejidad de la función y de la importancia de sus resultados habrá que tener estas cosas en cuenta e incluso introducir elementos para comprobar el tipo de datos de cada parámetro antes de ejecutar la función, esto se verá posteriormente, en el tema de excepciones.

A pesar de los problemas que una asignación incorrecta nos puede producir, Python nos permite saltarnos esta necesidad de pasar los parámetros ordenados, utilizando las denominadas *"Keywords"* que son pares de clave y valor con el nombre de los parámetros en las funciones y el valor que se le asigna.

```
def precioFruta(fruta, precio_kilo, kilos):
    total = precio_kilo * kilos
    print("Las " + fruta + " valen " + str(total) + " euros")

precioFruta(kilos=2, fruta="Manzanas", precio_kilo=5.95)
```

```
Las Manzanas valen 11.9 euros

Process finished with exit code 0
```

En ocasiones, también podemos querer definir parámetros opcionales en una función, es decir, parámetros que salvo que cuando se llame a la función se les dé explícitamente un valor, tendrá lo que se denomina un valor por defecto que se habrá establecido en la definición de la función. Para definir los valores de los parámetros por defecto, utilizaremos el operador de asignación dentro de la definición de la función en el parámetro que queremos fijar.

```
def saludar(nombre, saludo = "Hola"):
    print (saludo, nombre)

saludar("Pepe")

saludar("Pepe", "Buenos días")

saludar("Pepe")
```

```
Hola Pepe
Buenos días Pepe
Hola Pepe

Process finished with exit code 0
```

En ocasiones, generalmente en funciones definidas por terceros, nos encontraremos con funciones con muchos parámetros opcionales que tienen establecidos parámetros por defecto. En caso de que queramos utilizar esas funciones en nuestros programas y deseemos modificar sólo alguno de los valores por defecto, será muy útil el uso de las "*keywords*" que hemos visto anteriormente.

Con todo esto podemos concluir que una función podrá tener cualquier tipo de algoritmo en su interior, de mayor o menos complejidad y que se recomienda siempre que cada función realice una sola acción, que sea reutilizable y lo más genérica posible.

1.2 Ejercicios propuestos

A continuación, se propone una serie de ejercicios para aplicar los conceptos básicos de funciones. Las soluciones a dichos ejercicios, se encuentran al final del documento, pero se recomienda al alumno que trate de desarrollarlos.

1.2.1 Ejercicio 1

Escribe un programa que, introducidos dos números, llame a una función que los sume, otra que los reste, otra que los multiplique y otra que los divida. Mostrará por pantalla todos los resultados.

```
Introduce un número: 5

Introduce otro número: 8

La suma de 5.0 y 8.0 es 13.0

La resta de 5.0 menos 8.0 es -3.0

La multiplicación de 5.0 por 8.0 es 40.0

La división de 5.0 y 8.0 entre 0.625


Process finished with exit code 0
```

1.2.2 Ejercicio 2

Escribe un programa que, usando una función, calcule la distancia entre dos puntos, que se introducirán por coordenadas.

```
Introduce la coordenada x del primer punto: 4

Introduce la coordenada y del primer punto: 6

Introduce la coordenada x del segundo punto: 3.1

Introduce la coordenada y del segundo punto: 2.2

La distancia entre los dos puntos es 3.905124837953327


Process finished with exit code 0
```

1.2.3 Ejercicio 3

Escribe un programa que, usando una función, elimine todas las apariciones de la letra “a” en una cadena de texto. Mostrará el resultado antes y después del borrado.

```
Introduce un texto: Las cadenas que perderán vocales

Las cadenas que perderán vocales

Ls cdens que perdern vocles


Process finished with exit code 0
```

2 Soluciones a los ejercicios

Nota: Cómo en todos los problemas de programación, las soluciones no son únicas. En este apartado se plantean posibles soluciones. Al tratarse de ejemplos de carácter didáctico no siempre serán las soluciones óptimas.

2.1 Ejercicio 1

Escribe un programa que, introducidos dos números, llame a una función que los sume, otra que los reste, otra que los multiplique y otra que los divida. Mostrará por pantalla todos los resultados.

```
1  def sumar(a, b):
2      return a + b
3
4  def restar(a, b):
5      return a - b
6
7  def multiplicar(a, b):
8      return a * b
9
10 def dividir(a, b):
11     return a / b
12
13 a = float(input('Introduce un número: '))
14 b = float(input('Introduce otro número: '))
15
16 print('La suma de', str(a), 'y', str(b), 'es', sumar(a, b))
17 print('La resta de', str(a), 'menos', str(b), 'es', restar(a, b))
18 print('La multiplicación de', str(a), 'por', str(b), 'es', multiplicar(a, b))
19 print('La división de', str(a), 'y', str(b), 'entre', dividir(a, b))
```

```
Introduce un número: 5
Introduce otro número: 8
La suma de 5.0 y 8.0 es 13.0
La resta de 5.0 menos 8.0 es -3.0
La multiplicación de 5.0 por 8.0 es 40.0
La división de 5.0 y 8.0 entre 0.625

Process finished with exit code 0
```

La resolución de este ejercicio es sencilla. Entre las líneas 1 y 11 se definen las funciones que van a realizar las operaciones aritméticas. A continuación, en las líneas 13 y 14 se recogen los números que se van a utilizar y finalmente, entre las líneas 16 y 19 se imprimen los resultados.

2.2 Ejercicio 2

Escribe un programa que, usando una función, calcule la distancia entre dos puntos, que se introducirán por coordenadas.

```python
def distancia(x1, y1, x2, y2):
    dx = x2 - x1
    dy = y2 - y1
    dist = (dx**2 + dy**2)**0.5
    return dist

x1 = float(input('Introduce la coordenada x del primer punto: '))
y1 = float(input('Introduce la coordenada y del primer punto: '))
x2 = float(input('Introduce la coordenada x del segundo punto: '))
y2 = float(input('Introduce la coordenada y del segundo punto: '))

print('La distancia entre los dos puntos es', distancia(x1, y1, x2, y2))
```

```
Introduce la coordenada x del primer punto: 4
Introduce la coordenada y del primer punto: 6
Introduce la coordenada x del segundo punto: 3.1
Introduce la coordenada y del segundo punto: 2.2
La distancia entre los dos puntos es 3.905124837953327

Process finished with exit code 0
```

La estructura es semejante a la anterior, primero la función, que calcula la distancia como la raíz cuadrada de la suma de los cuadrados de los incrementos de x e y, a continuación, pide las coordenadas y finalmente, imprime resultados.

2.3 Ejercicio 3

Escribe un programa que, usando una función, elimine todas las apariciones de la letra “a” en una cadena de texto. Mostrará el resultado antes y después del borrado.

```python
def elimina_a(cadena):
    cadena = cadena.replace('a', '')
    cadena = cadena.replace('A', '')
    cadena = cadena.replace('á', '')
    cadena = cadena.replace('Á', '')
    return cadena

cadena = input('Introduce un texto: ')

print (cadena)
cadena = elimina_a(cadena)
print(cadena)
```

```
Introduce un texto: Las cadenas que perderán vocales
Las cadenas que perderán vocales
Ls cdens que perdern vocles

Process finished with exit code 0
```

Misma estructura que en los dos anteriores.

3 Bibliografía

Bahit, E. (s.f.). *Python para principiantes*. Obtenido de https://librosweb.es/libro/python/

Foundation, P. S. (2013). *Python.org*. Obtenido de https://www.python.org/

Suárez Lamadrid, A., & Suárez Jiménez, A. (Marzo de 2017). *Python para impacientes*. Obtenido de http://python-para-impacientes.blogspot.com.es/2017/03/el-modulo-time.html