PROGRAMACIÓN EN PYTHON
YOLANDA GARCÍA RUIZ

Definiendo funciones en Python

Las funciones son el método principal de organización y reutilización de código.

Ya hemos visto el uso de algunas funciones predefinidas de Python, como la función **print**, **abs**, etc. Predefinidas porque nosotros también podemos definir nuestras propias funciones.

- Las funciones tienen un *nombre* y se declaran con la palabra reservada **def** y devuelve un valor usando la palabra reservada **return**.
- También tienen una lista de argumentos:
  - posicionales
  - por clave
  - argumentos agrupados (de estos hablaremos más adelante)

Argumentos posicionales:

Veamos algunos ejemplos de funciones con argumentos posicionales:

```
# función que suma 3 números y devuelve el resultado
# 3 argumentos posicionales. más adelante hablaremos de los argumentos por clave
def suma_tres(x, y, z):
    m1 = x + y
    m2 = m1 + z
    return m2
```

Podemos utilizar la función anterior para sumar los números 4, 2 y 3. Para ello basta con escribir el nombre la función junto con los valores 4, 2 y 3 como argumentos. A esto se le conoce como **llamada a la función** o **invocar a la función**:

```
suma_tres(1,2,8)
```

```
11
```

```
a = 1
b = 8
 # los valores de a y b se sustituyen por 1 y 8 respectivamente en la llamada
resultado = suma_tres (a,2,b)
resultado
```

```
11
```

Los argumentos por clave se usan para indicar valores por defecto y siempre se sitúan después de los argumentos posicionales. De esta forma podemos crear una función que puede ser invocada con menos argumentos que los que permite.

```
# función que suma 3 números y devuelve el resultado
def suma_varios(x, y, z1 = 0, z2 = 0):       # 2 argumentos posicionales y 2 por clave
    m = x + y + z1 + z2
    return m
```

Esta función puede ser invocada pasando sólo los argumentos obligatorios x e y :

```
resultado1 = suma_varios(2, 3)    # z1 y z2 tomarán los valores por defecto
resultado1
```

```
5
```

Esta función puede ser invocada pasando los argumentos obligatorios x e y junto con un argumento por clave:

```
resultado2 = suma_varios(2, 3, z2 = 1)    # z1 toma el valor por defecto 0
resultado2
```

```
6
```

Esta función puede ser invocada pasando los argumentos obligatorios x e y junto con los dos argumento por clave:

```
resultado3 = suma_varios(2, 3, z2 = 1, z1 = 9)   # ningún valor por defecto
resultado3
```

```
15
```

Valores devueltos por una función

- Una función puede tener varias instrucciones **return**.
- Si la ejecución de una función no alcanza ninguna instrucción **return**, se devuelve **None** que es el valor que representa el valor nulo en Python. El tipo de **None** es **NoneType**.

```
def suma_todos(x, y, z, t):
    resultado = x + y + z + t     # calcula el resultado pero no devuelve nada
                                  # No hay instrucción return
```

```
r = suma_todos(2, 3, 4, 6)
r
```

```
type(r)
```

```
NoneType
```

```
# función que pretende escribir cuatro cadenas: Uno, Dos, Tres, Cuatro
# si añadimos una instrucción return, no se ejecutan todas las instrucciones
def cuenta():
    print('Uno')
    print('Dos')
    return ('Fin de la función')  # la función termina tras ejecutar el return
    print('Tres')
    print('Cuatro')
```

```
cuenta()
```

```
Uno
Dos


'Fin de la función'
```

Semántica de Python

A diferencia de otros lenguajes como C++, Java o Perl, Python no utilliza corchetes o llaves para estructurar el código. **Python utiliza espacios en blanco o tabulaciones para dar estructura a su código**.

```
def hola():
    print('Hola Mundo!')    # Tabulación al comienzo de ésta linea
```

```
hola()    # llamada a la dunción 'hola'
```

```
Hola Mundo!
```

```
def producto(a,b):
    res = a * b
     print(res)    # más o menos espacios al comienzo produce un error de sintaxis
    return res
```

```
  File "<ipython-input-23-ef2aedbfdebd>", line 3
    print(res)    # más o menos espacios al comienzo produce un error de sintaxis
    ^
IndentationError: unexpected indent
```

El uso de tabuladores hace el código más legible. En otros lenguajes el código de la función 'hola' ha de escribirse utilizando llaves, como se muestra a continuación:

```
def hola():
    {
    print('Hola Mundo!')
    }
```

Return múltiples valores

Aquí tenemos otra de las características que hace a Python atractivo a los programadores.

A diferencia de Java y C++, las funciones Python pueden devolver múltiples valores:

```
def orden(a,b):
    if a <= b:
        print('entra por el primer bloque')
    else:
        return int(b), a
    print('Esta linea es el final')
```

```
m = orden(7, 1)
m
```

```
(1, 7)
```

La función orden devuelve una **tupla** con dos valores.

References

- Tutorial de Python. Por Guido Van Rossum