Capítulo 6

Definición de Funciones

```
def strPos(s,n):
    b=False
    if s[n]=='1': b=True
    return b


def pantalla(k):
    c=0
    for n in range(100):
        if strPos(strBin(n),-k):
            c+=1
            print(str(n).rjust(3),end=' ')
        if c==10:
            c=0
            print()
```

En el capítulo anterior hemos visto algunos módulos de Python que resultan muy útiles para llevar a cabo determinadas tareas.

Python es de los lenguajes de programación que dispone de un mayor número de librerías (módulos) que amplían con nuevas funcionalidades las características generales del lenguaje, pero a pesar de disponer de muchísimas funciones en los módulos de podamos encontrar, es muy probable que tengamos que crear nuestras propias funciones que realicen tareas específicas y personalizadas para el programa que tengamos que realizar.

¿Cuando es necesario crear una función?

Las funciones nos ayudan a organizar el código de nuestra aplicación. Si nuestro programa comienza a tener una extensión considerable, será recomendable separar nuestro código en funciones que cada una de ellas realice una tarea determinada del programa.

En otras ocasiones, hay una cierta tarea que hay que repetir en múltiples ocasiones dentro del programa, para estos casos, en lugar de repetir el código, se debe crear una función que haga dicha tarea y llamar a la función cada vez que necesitemos esa funcionalidad.

¿Cómo definir nuestras funciones?

Las funciones **debemos definirlas al principio de nuestro archivo**, de esta forma estarán disponibles en el resto del código para poder utilizarlas.

Las funciones se definen como un bloque de código que comienza con la palabra reservada **def**, y tiene la siguiente apariencia:

```
def nombrefuncion(parametro1, parametro2,...):
      instruccion1
      instruccion2
      …
      return valor      # Aunque es opcional, es lo más habitual
```

Cuando definimos una función, es necesario darle un nombre a la función, y generalmente tendremos que pasarle unos valores para que la función realice los cálculos o las tareas necesarias con esos valores. A los valores que se le pasan a la función se le llaman parámetros o argumentos, y una función puede recibir uno, varios o ninguno.

También es habitual que las funciones devuelvan un valor, aunque no es obligatorio. Generalmente el valor devuelto es el resultado de realizar las operaciones que lleva a cabo la función con los valores que le hemos pasado como argumentos.

Veamos un primer ejemplo de un programa en el que definiremos una función que calcule la diagonal de un cuadrado.

La función la llamaremos diagonal y recibirá un parámetro que llamaremos lado.

Dentro de nuestra función tendremos una variable que llamaremos resultado que será el valor que finalmente devolverá nuestra función con las instrucción return.

En el programa calcularemos la diagonal de varios cuadrados, de manera que llamaremos en varias ocasiones a la función diagonal.

Vamos a recordar que la diagonal del cuadrado podemos calcularla aplicando el Teorema de Pitágoras, $\mathbf{d^2=l^2+l^2}$ (en este caso los 2 catetos coinciden con el lado), por lo tanto, la diagonal es la raíz cuadrada de $\mathbf{2*l^2}$.

Programa: funciones1.py

1º Realizaremos la importación del módulo math.

2º El programa mostrará un mensaje indicando su funcionalidad.

3º Definiremos la función diagonal.

4º Mostraremos la diagonal de los cuadrados de lado 1, 5, 10 y 12.

```
1     from math import sqrt
2     def diagonal(lado):
3         resultado=sqrt(lado**2+lado**2)
4         return resultado
5
6     print('Programa que calcula la diagonal de varios cuadrados')
7     print('El cuadrado de lado',1,'tiene de diagonal',diagonal(1))
8     print('El cuadrado de lado',5,'tiene de diagonal',diagonal(5))
9     print('El cuadrado de lado',10,'tiene de diagonal',diagonal(10))
10    print('El cuadrado de lado',12,'tiene de diagonal',diagonal(12))
```

Obtendremos como ejecución del programa:

```
>>> %Run funciones1.py

  Programa que calcula la diagonal de varios cuadrados
  El cuadrado de lado 1 tiene de diagonal 1.4142135623730951
  El cuadrado de lado 5 tiene de diagonal 7.0710678118654755
  El cuadrado de lado 10 tiene de diagonal 14.142135623730951
  El cuadrado de lado 12 tiene de diagonal 16.97056274847714

>>>
```

Ejercicios:

Nota: Recuerda que para realizar los siguientes ejercicios debes importar los módulos que correspondan en cada caso.

Ejercicio 1:

Realizar un programa que obtenga la raíz cuadrada de todos los números impares entre 1 y 100.

Ejercicio 2:

Realizar un programa que obtenga la raíz cuadrada de todos los números pares entre 1 y 100.

Ejercicio 3:

Realizar un programa que obtenga el logaritmo en base 10 de todos los números entre 1 y 100. **Nota:** log10(x) sirve para calcular el logaritmo en base 10.

Ejercicio 4:

Realiza un programa que solicite una palabra y a continuación muestre un carácter aleatorio de los que contenga la palabra introducida.

Ejercicio 5:

Realiza un programa que tenga 10 nombres escritos en una lista, y que elija aleatoriamente uno de los nombres de la lista. **Nota:** usa el módulo random y la función **choice** de este módulo.

Ejercicio 6:

Realiza un programa que tenga 10 nombres escritos en una lista, y que elija aleatoriamente tres de los nombres de la lista. **Nota:** usa el módulo random y la función **sample** de este módulo.

Ejercicio 7:

Realiza un programa que tenga 10 nombres escritos en una lista, y que muestre los nombres desordenados, es decir, desordenarlos aleatoriamente. **Nota:** usa el módulo random y la función **shuffle** de este módulo.

Ejercicio 8:

Realiza un programa que genere 6 números aleatorios entre 1 y 49. Como el juego de la primitiva.

Ejercicio 9:

Realiza un programa que genere 15 resultados 1, x o 2. Como una apuesta de la quiniela.

Ejercicio 10:

Realiza un programa que genere 2 número aleatorios entre 1 y 6 simulando el lanzamiento de 2 dados. Debe mostrar los 2 números obtenidos y la suma de dichos números.

Ejercicio 11:

Realiza un programa que genere 10 fechas aleatorias a partir de la fecha de hoy. Nota: puedes usar la función **timedelta()** que se encuentra en el módulo **timedate**.