**FUNCIONES**

Una *función* es como un mini programa dentro de un programa. Python proporciona varias funciones integradas, como las funciones print(), input() y len() de las clases anteriores, pero también pueden escribir las propias.

### Creación de funciones

Para entender mejor cómo funcionan las funciones, vamos a crear una.

```
def saludo():
    print("Hola a TODOS")

saludo()
```

La primera línea es una declaración **def**, que define una función llamada **saludo()**. El código en el bloque que sigue la declaración def es el **cuerpo de la función**. Este código se ejecuta cuando se llama a la función, no cuando la función se define por primera vez.

La línea saludo() después de la función son **llamadas de función**. En el código, una llamada de función es solo el nombre de la función seguido de paréntesis. Cuando la ejecución del programa llegue a estas llamadas, saltará a la primera línea de la función y comenzará a ejecutar el código allí. Cuando llega al final de la función, la ejecución vuelve a la línea que llamaba a la función y continúa moviéndose a través del código como antes.

Un propósito principal de las funciones es agrupar el código que se ejecuta varias veces. Sin una función definida, tendría que copiar y pegar este código cada vez que quisiera ejecutarlo.

En general, siempre desea evitar duplicar el código, porque si alguna vez decide actualizar el código (por ejemplo, porque encuentra un error que necesita corregir), tendrá que recordar cambiar el código en cada lugar que lo copió.

### Argumentos y parámetros

Cuando llama a la función print() o len(), lo pasa por valores, llama *a argumentos*, introduciéndolos entre los paréntesis. También puede definir sus propias funciones que aceptan argumentos.

```
def saludo(nombre):
    print("Buenas Tardes " + nombre)

saludo("José")
```

La definición de la función **saludo(nombre)** tiene un parámetro llamado nombre . *Los parámetros* son variables que contienen argumentos. Cuando una función se llama con argumentos, los argumentos se almacenan en los parámetros. La primera vez que se llama a la función saludo(), se pasa el argumento 'José'. La ejecución del programa entra en la función, y el nombre del parámetro se establece automáticamente en 'José', que se imprime por la sentencia print().

**Valores de devolución y declaraciones de devolución**

Cuando llama a la función len() y la pasa un argumento como 'Hola', la llamada de función evalúa el valor entero 4, que es la longitud de la cadena que la pasó.

Al crear una función con la sentencia def, puede especificar el valor de retorno con una sentencia **return**, que consiste en lo siguiente:

- La palabra clave return
- El valor o la expresión que la función debería devolver

En el caso de una expresión, el valor de retorno es lo que sea que esta expresión se evalúe.

```
def sumar(x, y):
   return x + y

print (sumar(3, 2))
```

Como vemos esta función tan sencilla no hace otra cosa que sumar los valores pasados como parámetro y devolver el resultado como valor de retorno.

En el cuerpo de una función no solo pueden aparecer sentencias return; también podemos usar estructuras de control: sentencias condicionales, bucles, etc. Lo podemos comprobar diseñando una función que recibe un número y devuelve un booleano. El valor de entrada es la edad de una persona y la función devuelve True si la persona es mayor sino False.

```
def es_mayor_de_edad(edad):
        if edad < 18:
                resultado = False
        else:
                resultado = True

        return(resultado)
```

Pero no el único modo:

```
def es_mayor_de_edad(edad):
        if edad < 18:
                return False
        else:
                return True
```