Funciones

En Python, una función es un grupo de instrucciones que constituyen una unidad lógica dentro del programa. Resuelve un problema específico y permiten la modularidad del código

Los beneficios de la programación funcional son

- Facilita el trabajo en equipo, al modularizar el código permite trabajar en unidades lógicas separadas
- Encapsulamiento, dividir y organizar el código en partes más sencillas que se pueden encapsular en funciones y ser reutilziado a lo largo del proyecto
- Simplifica la lectura, el código estructurado en funciones tiene un cuerpo principal reducido y funciones bien delimitadas
- Reutilización, el código encapsulado en una función puede utilizarse en diferentes proyectos
- Mantenimiento, el software que utiliza funciones es más fácil de mantener

Elementos de una función

1. Primera línea: cabecera o definición de la función.

   def es una palabra reservada que define una función, seguida por el nombre de la función, los parámetros y dos puntos de cierre de la cabecera

2. Cuerpo: se delimita con la indentación y constituye el código que se encapsula en la función
3. Return: es opcional, permite a la función devolver valores al cuerpo principal del programa

```
# Definición de la función
def imprimir_mensaje_cinco_veces():
for i in range(5):
print("Este es el mensaje " + str(i))
# Invocación de la función
imprimir_mensaje_cinco_veces()
```

Paso por valor y por referencia

Dependiendo del tipo de dato que enviemos a la función, se tienen dos comportamientos diferentes

- Paso por valor: se crea una copia del valor de los argumentos en los respectivos parámetros. Las modificaciones en los valores de los parámetros no afectan a las variables externas
- Paso por referencia: se pasa un puntero a la posición de memoria donde se aloja el dato, por lo que cualquier cambio que se haga en su valor dentro de la función afecta el contenido de la variable en el resto del programa

En Python los argumentos de tipos de datos simples se pasan por valor y los tipos de datos compuestos (listas, diccionarios, conjuntos, etc) se pasan por referencia

Parámetros opcionales

En una función Python se pueden indicar una serie de parámetros opcionales con el operador =. Son parámetros que, si no se incluyen al invocar a la función toman ese valor por defecto

```
def sumar(a = 0, b = 0):
  return a + b
print(sumar(2,6))
print(sumar(5))
print(sumar())
```

En una función se pueden especificar tantos parámetros opcionales como se quiera. Sin embargo, una vez que se indica uno, todos los parámetros a su derecha también deben ser opcionales

Parámetros posicionales

Al invocar una función con diferentes argumentos, los valores se asignan a los parámetros en el mismo orden en que se indican. Sin embargo, el orden se puede cambiar si llamamos a la función indicando el nombre de los parámetros

Devolución de valores

Una función puede devolver información para ser utilizada o almacenada en una variable. Se utiliza la palabra clave return, que regresa un valor y finaliza la ejecución de la función. Si existe código después del return, nunca será ejecutado. Puede haber más de un return por función.

```
# La función resta dos valores numéricos.
def restar(num1,num2):
  resta= num1-num2
  return resta # Retorna un valor
# Programa principal
resultado= restar(10,3)
print("El 1er resultado es:", resultado)
print("El 2do resultado es:", restar(10,4))
```

La sentencia return es opcional, y puede devolver, o no, un valor. Es posible que aparezca más de una vez dentro de una misma función

Devolución de varios valores

Es posible devolver más de un valor con una sentencia return. La siguiente función cuadrado_y_cubo() devuelve el cuadrado y el cubo de un número:

```
# La función devuelve dos valores
def cuadrado_y_cubo(numero):
  return numero ** 2, numero ** 3
# Programa principal
cuadrado, cubo = cuadrado_y_cubo(2)
print("Cuadrado:", cuadrado)
print("Cubo:", cubo)
```

Funciones anónimas o lambda

A diferencia de las funciones que se definen con la palabra reservada def, las funciones anónimas se definen con la palabra reservada lambda. Su sintaxis es:

```
lambda parámetros: expresión
```

Dada su sintáxis, estas funciones se suelen llamar funciones lambda

- Pueden tener varios parámetros pero una única expresión
- Se suelen usar en combinación con otras funciones, generalmente como argumentos de otra función

Documentar funciones

Los comentarios delimitados por triples comillas dobles sirven apra documentar funciones o bloques de código. Son la primera sentencia de cada uno de ellos. Todos los módulos que deberían tener docstrings y todas las funciones y clases exportadas por un módulo deberían tenerlos

```
def cuad(x):
"""Dado un número x, calcula x²"""
return x * x # También podríamos haber hecho x ** 2
def modulo_vector(x, y):
"""Calcula el módulo de un vector en 2D.
Argumentos:
x: (float|int) coordenada de las abscisas
y: (float|int) coordenada de las ordenadas
Devuelve: (float) el módulo del vector
"""
return (x ** 2 + y ** 2) ** 0.5
```