Funciones

Una función es un bloque de código con el fin de resolver una tarea específica. De esta forma el código en su totalidad queda más prolijo y ordenado ya que cada bloque se encarga de tareas específicas y cada función resuelve problemas determinados.

Podría decirse que una función tiene dos partes:

- definición
- invocación

Definición

Es donde se escribe el código que resuelve el problema para la que la función fue invocada y tiene la siguiente sintaxis:

```
def nombre_funcion (parametros):
      codigo
      return resultado

#def es una palabra clave que se utiliza para definir una función
#return es una palabra clave que solo se utiliza cuando la función
devuelve un valor
```

Puntos a tener en cuenta:

- Comienza con la palabra clave **def** que significa definir
- Luego va el nombre de la función
- Después sigue un par de paréntesis **()**, en caso de la función requiera parámetros se detallan dentro de estos
- Termina con dos puntos **:**
- En las siguientes se desarrolla el código de la función, respetando la indexación
- Si la función devuelve algún valor se utiliza la palabra clave **return** y por consiguiente la variable
- Para invocar a la función se escribe el nombre de la misma

Invocación

Es el momento en el que se llama a la función, y se le pasan los argumentos, si es que los requiere. Tambien, la función puede retornar algún valor, en ese caso antes

de invocar se debe poner la variable donde se guardará la información que devuelve la función, por ejemplo

```
#función que no retorna ningún valor ni requiere parámetros
funcion()

#función que requiere argumentos y sin retorno
funcion(valor1, valor2)

#en caso de que si retorne un valor
variable = funcion()
```

Flujo de la función

Cuando se llama a una función, Python registra esta línea y se dirige hacia donde está el código de la función y se ejecuta. Cuando finaliza Python vuelve a la línea donde se realizó la invocación. Este proceso se repite siempre que la función sea llamada.

Función con parámetros

Los parámetros en una función es la forma que esta recibe datos de entrada. Un parámetro es una variable pero con algunas diferencias:

- solo existen dentro de la función donde fueron definidos
- la asignación de un valor a un parámetro se hace en el momento en que se llama a la función, especificando el argumento correspondiente

Funciones con xargs parámetros

Tenemos la posibilidad de poder indicar en los argumentos que se van a poder muchos de ellos, sin indicar exactamente la cantidad.
Esto se hace agregando un **asterisco** * delante del parámetro, así vamos a poder iterar sobre dicha variable, y de esta manera recibir la cantidad de número que queramos.

```
def funcion (*numeros):
    suma=0
    for num in numeros:
        suma+=num
```

```
    return suma



print(funcion(1,2))

print(funcion(1,2,3))

print(funcion(1,2,3,4))
```

Uso de **kwargs para argumentos clave-valor

En Python, **kwargs se utiliza para pasar un número variable de argumentos clave-valor a una función. El nombre kwargs es una convención y puede ser cualquier otro nombre precedido por dos asteriscos **. Dentro de la función, kwargs se comporta como un diccionario que contiene todos los pares clave-valor pasados a la función.

```
def  datos_personales(**kwargs):
    for clave,valor in kwargs.items():
        print(f'{clave}: {valor}')


#llamada de la funcion
datos_personales(nombre='Juan', apellido='Kildegaard', edad=37)
```

Puntos a tener en cuenta

- ➔ No se debe invocar a una función antes de que se haya definido
- ➔ Una función y una variable no pueden compartir el mismo nombre
- ➔ Se debe proveer el mismo número de argumentos como parámetros definidos, el no hacerlo provocará un error.
- ➔ Las variables declaradas dentro de una función solo viven dentro de ese bloque de código, en otras palabras no se puede utilizar en otro lugar que no sea en la función donde fueron creadas.
- ➔ Los parámetros son los nombres que aparecen en la definición de la función

y los argumentos son los valores que le pasamos a una función

Por último, voy a tomar dos principios del **Zen de Python** que me parecen tiene mucho que ver con la definición de funciones:

1. **"Lo explícito es mejor que lo implícito"**
   Cuando definimos una función, es importante que su nombre y argumentos sean claros sobre lo que realiza, también ayuda a describir su propósito. Esto ayuda a que el código sea fácil de entender para otros

2. **"Lo simple es mejor que lo complejo"**
   Cuando se definen funciones debemos buscar lo simple. Las funciones deben hacer una sola cosa y hacerlo bien. Si se vuelve muy compleja podría ser una señal de que deba ser dividida en otras funciones.