

Documentación de funciones

Documentar funciones se refiere al proceso de proporcionar información descriptiva sobre cómo funciona una función, qué parámetros acepta, qué valor retorna y cualquier otra información relevante. Esta documentación ayuda a los usuarios (incluido tú mismo en el futuro) a entender cómo usar la función sin tener que leer o interpretar su código fuente.

En Python, la documentación de funciones generalmente se realiza utilizando docstrings (cadenas de documentación) que se colocan justo debajo de la definición de la función. Los docstrings son una forma estándar de documentar funciones, clases y módulos en Python.

La convención más común es utilizar el formato de triple comillas (“ “ “...” “ “ “)

Formato básico de un Docstring:

```
def nombre_funcion(Parámetros):  
  
    """  
    Breve descripción de lo que hace la función.  
  
    Parámetros:  
    param1 (tipo): Descripción de param1.  
    param2 (tipo): Descripción de param2.  
  
    Retorna:  
    tipo: Descripción de lo que devuelve la función.  
    """  
  
    # Cuerpo de la función
```

Ejemplo 1

```
def saludar():
    """
    Imprime un saludo en la consola.

    Esta función no toma ningún parámetro ni retorna ningún valor.
    Simplemente muestra el mensaje "Hola mundo" en la consola.
    """
    print("Hola mundo")
```

Ejemplo 2

```
def factorial_de_5():
    """
    Calcula el factorial del número 5.

    El factorial de 5 (5!) es el producto de todos los números
    enteros positivos desde 1 hasta 5. Esta función realiza la
    multiplicación de 1*2*3*4*5 y retorna el resultado.

    Retorna:
    int: El valor del factorial de 5, que es 120.
    """
    factorial = 1 * 2 * 3 * 4 * 5
    return factorial
```

Ejemplo 3

```
def suma_2_numeros(a, b):
    """
    Esta función toma dos números como parámetros y devuelve su suma.

    Parámetros:
    a (int, float, complex): El primer número a sumar
    b (int, float, complex): El segundo número a sumar

    Retorna:
    int, float, complex: La suma de los dos números. El tipo de retorno
    depende del tipo de los parámetros
```

```
"""  
return a + b
```

Ejemplo 4

```
def mayor_2_numeros(a, b):  
    """  
    Encuentra el mayor de dos números.  
  
    Esta función compara dos números (enteros o flotantes)  
    y devuelve el mayor de ellos.  
  
    Parámetros:  
    a (int, float): El primer número a comparar.  
    b (int, float): El segundo número a comparar.  
  
    Retorna:  
    int, float: El mayor de los dos números proporcionados. El tipo de retorno  
                depende del tipo de los parámetros.  
    """  
    if a > b:  
        return a  
    else:  
        return b
```

Ejemplo 5

```
def potencia(num, pot):  
    """  
    Calcula la potencia de un número.  
  
    Esta función toma un número base y un exponente, y calcula el resultado  
    de elevar la base a la potencia del exponente.  
  
    Parámetros:  
    num (int, float): El número base que se va a elevar a una potencia.  
    pot (int, float): El exponente al que se elevará la base.
```

```

Retorna:
int, float: El resultado de elevar el número base a la potencia del
             exponente. El tipo de retorno depende del tipo de los
             parámetros. Si ambos parámetros son enteros, el retorno es
             un entero; si alguno es flotante, el retorno es un flotante.
"""
return num ** pot

```

Ejemplo 6

```

def hipotenusa(a, b):
    """
    Calcula la hipotenusa de un triángulo rectángulo utilizando el teorema
    de Pitágoras.

    Esta función toma las longitudes de los dos catetos de un triángulo
    rectángulo y calcula la longitud de la hipotenusa usando la fórmula:
         $c = (a^2 + b^2)$ , donde a y b son los catetos.

    Parámetros:
    a (int, float): La longitud del primer cateto del triángulo rectángulo.
    b (int, float): La longitud del segundo cateto del triángulo rectángulo.

    Retorna:
    float: La longitud de la hipotenusa del triángulo rectángulo. El valor
           retornado es un flotante que representa la distancia calculada
           mediante el teorema de Pitágoras.
    """
    c = math.sqrt(potencia(a, 2) + potencia(b, 2))
    return c

```

Ejemplo 7

```

def construir_diccionario(nombre, edad, estatura):
    """
    Crea un diccionario con información personal.

    Esta función toma el nombre, la edad y la estatura de una persona y

```

construye un diccionario que almacena estos datos con las claves correspondientes.

Parámetros:

nombre (str): El nombre de la persona.

edad (int): La edad de la persona.

estatura (float): La estatura de la persona en metros.

Retorna:

dict: Un diccionario con tres claves: "nombre", "edad" y "estatura".

El valor asociado a cada clave es el parámetro correspondiente proporcionado a la función. El diccionario tiene la estructura {'nombre': nombre, 'edad': edad, 'estatura': estatura}.

"""

```
dicc = {"nombre": nombre,  
        "edad": edad,  
        "estatura": estatura}  
return dicc
```

Ejemplo 8

```
def cuadrado(x):
```

"""

Calcula el cuadrado de un número.

Esta función toma un número y devuelve su cuadrado.

Parámetros:

x (int, float): El número del cual se calculará el cuadrado.

Retorna:

int, float: El cuadrado del número proporcionado. El tipo de retorno depende del tipo del parámetro `x`.

"""

```
return x ** 2
```

Ejemplo 9

```
def raiz(x):
    """
    Calcula la raíz cuadrada de un número.

    Esta función toma un número no negativo y devuelve su raíz cuadrada
    utilizando la función `sqrt` del módulo `math`.

    Parámetros:
    x (int, float): El número no negativo del cual se calculará
                    la raíz cuadrada.

    Retorna:
    float: La raíz cuadrada del número proporcionado.
           El resultado es un flotante.
    """
    return math.sqrt(x)
```

Ejemplo 10

```
def aplicar_operacion(funcion, valor):
    """
    Aplica una función a un valor dado.

    Esta función toma una función y un valor, y aplica la función al valor.
    La función proporcionada debe aceptar un solo argumento y devolver un
    resultado.

    Parámetros:
    funcion (function): Una función que acepta un solo argumento y devuelve
                        un resultado.
    valor (int, float): El valor al que se aplicará la función.

    Retorna:
    El resultado de aplicar la función al valor. El tipo de retorno depende
    de la función aplicada.
    """
    resultado = funcion(valor)
    return resultado
```

Formas de extraer información de las funciones

```
help(suma_2_numeros)
```

Help on function suma_2_numeros in module __main__:

```
suma_2_numeros(a, b)
```

Esta función toma dos números como parámetros y devuelve su suma.

Parámetros:

a (int, float, complex): El primer número a sumar

b (int, float, complex): El segundo número a sumar

Retorna:

int, float, complex: La suma de los dos números. El tipo de retorno depende del tipo de los parámetros

```
suma_2_numeros?
```

Signature: suma_2_numeros(a, b)

Docstring:

Esta función toma dos números como parámetros y devuelve su suma.

Parámetros:

a (int, float, complex): El primer número a sumar

b (int, float, complex): El segundo número a sumar

Retorna:

int, float, complex: La suma de los dos números. El tipo de retorno depende del tipo de los parámetros

File: c:\users\jose_\appdata\local\temp\ipykernel_15340\1801936373.py

Type: function