



Clase 1: Funciones y Manejo de Excepciones en Python

1. Principios Fundamentales de las Funciones

Una de las bases del diseño de software es el principio de **Divide y Vencerás**. Este principio se refiere a la práctica de dividir el código en porciones más pequeñas (funciones) para facilitar su legibilidad, mantenimiento y reutilización. Las funciones permiten encapsular porciones de lógica que realizan tareas específicas, evitando la duplicación de código.

1.1 Definición de Funciones


En Python, las funciones se definen utilizando la palabra reservada `def`, seguida del nombre de la función y paréntesis que pueden incluir parámetros. La sintaxis básica es la siguiente:

```
def nombre_funcion(parametros):  
    # Cuerpo de la función  
    pass
```

1.2 Ejemplo Básico

Vamos a crear una función simple que imprime "Hola, Mundo":

```
def saludar():  
    print("Hola, Mundo")  
  
saludar() # Llamada a la función
```

 **Nota:** Asegúrate de llamar a la función después de definirla.

1.3 Funciones con Parámetros

Las funciones pueden aceptar parámetros para trabajar con datos específicos. Por ejemplo, si queremos personalizar el saludo:

```
def saludar(nombre):  
    print(f"Hola, {nombre}")  
  
saludar("Diego") # Salida: Hola, Diego
```

1.4 Parámetros Predeterminados

Puedes definir valores predeterminados para los parámetros:

```
def saludar(nombre="Invitado"):  
    print(f"Hola, {nombre}")  
  
saludar() # Salida: Hola, Invitado  
saludar("Carla") # Salida: Hola, Carla
```

1.5 Argumentos Posicionales y Nombrados

Los parámetros pueden ser pasados de forma posicional o nombrada:

```
def saludar(nombre, apellido=""):  
    print(f"Hola, {nombre} {apellido}")  
  
saludar("Diego", "García") # Salida: Hola, Diego García
```

```
saludar(apellido="Pérez", nombre="Juan") # Salida: Hola, Juan Pérez
```

2. Manejo de Excepciones ⚠

El manejo de excepciones en Python permite gestionar errores y evitar que el programa se detenga inesperadamente. Utilizamos bloques `try` y `except` para manejar excepciones.

2.1 Sintaxis Básica

```
try:
    # Código que puede causar un error
    resultado = 10 / 0
except ZeroDivisionError:
    print("¡Error! No se puede dividir por cero.")
```

🔍 **Nota:** Es importante manejar las excepciones adecuadamente para brindar una mejor experiencia al usuario.

3. Ejemplo de Calculadora 🧮

Vamos a implementar una calculadora simple que utiliza funciones para sumar, restar, multiplicar y dividir.

3.1 Definición de Funciones de Operaciones

```
def sumar(a, b):
    return a + b

def restar(a, b):
    return a - b

def multiplicar(a, b):
    return a * b
```

```
def dividir(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "¡Error! No se puede dividir por cero."
```

3.2 Función Principal de la Calculadora

```
def calculadora():
    while True:
        print("Selecciona una operación:")
        print("1. Sumar")
        print("2. Restar")
        print("3. Multiplicar")
        print("4. Dividir")
        print("5. Salir")

        opcion = input("Ingrese su opción (1-5): ")

        if opcion == '5':
            print("Saliendo de la calculadora.")
            break

        if opcion in ['1', '2', '3', '4']:
            num1 = float(input("Ingrese el primer número: "))
            num2 = float(input("Ingrese el segundo número: "))

            if opcion == '1':
                print(f"La suma es: {sumar(num1, num2)}")
            elif opcion == '2':
                print(f"La resta es: {restar(num1, num2)}")
            elif opcion == '3':
                print(f"La multiplicación es: {multiplicar(num1, num2)}")
            elif opcion == '4':
                print(f"La división es: {dividir(num1, num2)}")
        else:
            print("Opción no válida. Por favor, ingrese una opción entre 1 y 5.")

    return
```

```

m1, num2))}")
        elif opcion == '4':
            print(f"La división es: {dividir(num1, num
2))}")
        else:
            print("Opción no válida. Por favor intenta de nue
vo.")

```

3.3 Ejecución de la Calculadora

Para iniciar la calculadora, simplemente llama a la función:

```
calculadora()
```

4. Ejercicios Prácticos

Ejercicio 1: Saludo Personalizado

Objetivo: Crear una función que reciba un nombre y devuelva un saludo.

Solución:

```

def saludo_personalizado(nombre):
    return f"¡Hola, {nombre}! 🖐️"

print(saludo_personalizado("Carlos")) # Salida: ¡Hola, Carlos! 🖐️

```

Ejercicio 2: Calculadora de IMC

Objetivo: Crear una función que calcule el Índice de Masa Corporal (IMC).

Solución:

```

def calcular_imc(peso, altura):
    return peso / (altura ** 2)

```

```
peso = float(input("Ingrese su peso en kg: "))
altura = float(input("Ingrese su altura en metros: "))
imc = calcular_imc(peso, altura)
print(f"Su IMC es: {imc:.2f} 📏")
```

5. Resumen

- Las **funciones** permiten dividir el código, mejorar su legibilidad y facilitar su mantenimiento.
- Se pueden definir con o sin parámetros, y se pueden usar valores predeterminados.
- El **manejo de excepciones** es crucial para evitar errores en tiempo de ejecución.
- Crear funciones para tareas específicas, como operaciones matemáticas, mejora la organización del código.

¡Con estos conceptos y ejercicios, estás listo para profundizar en el uso de funciones y excepciones en Python! 🚀