Manejo de Excepciones en POO (Python)

Alumno: Puma Huanca Anthony Rusbel
Docente: Ing. Coyla Idme Leonel
Lenguajes de Programación II – FINESI
Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática

Introducción

El **manejo de excepciones** en la Programación Orientada a Objetos (POO) con Python es un mecanismo que permite identificar y gestionar errores en tiempo de ejecución. Utilizando bloques como `try`, `except`, `else` y `finally`, se puede interrumpir el flujo normal del programa y ejecutar código específico para tratar errores de forma controlada, garantizando la estabilidad y confiabilidad del sistema.

Ejercicio 1 División entre cero

```python
class Division:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def dividir(self):
        try:
            resultado = self.a / self.b
            return resultado
        except ZeroDivisionError:
            return "Error: No se puede dividir entre cero."
        except Exception as e:
            return f"Ocurri  un error: {e}"
        finally:
            print("Proceso de divisi n concluido.")

valor_a = 10
valor_b = 0
operacion = Division(valor_a, valor_b)
print(f"Resultado de la operaci n: {operacion.dividir()}")
```

Listing 1: Clase `Division` con manejo de excepciones

Ejecución:

```
Proceso de división concluido.
Resultado de la operación: Error: No se puede dividir entre cero.
```

Ejercicio 2 Cálculo de la hipotenusa

```python
import math

def calcular_hipotenusa(cateto_a, cateto_b):
    return math.sqrt(cateto_a**2 + cateto_b**2)

def main():
    try:
        a = float(input("Ingrese el valor de A:"))
        b = float(input("Ingrese el valor de B:"))
        if a<=0 or b<=0:
            raise ValueError("Los catetos deben ser n meros positivos")

        hipotenusa = calcular_hipotenusa(a,b)
        print(f"La hipotenusa es: {hipotenusa:2f}")

    except ValueError as ve:
        print("Error",ve)

    except Exception as e:
        print("Ocurri  un problema inesperado",e)

if __name__ == "__main__":
    main()
```

Listing 2: Validación de entradas numéricas positivas

Ejecución:

```
Entrada 1:
    Ingrese el valor de A:3
    Ingrese el valor de B:4
    La hipotenusa es: 5.000000
Entrada 2:
    Ingrese el valor de A:cinco
    Error invalid literal for float(): cinco
Entrada 3:
    Ingrese el valor de A:3
    Ingrese el valor de B:-4
    Error Los catetos deben ser números positivos
```

Ejercicio 3 Fibonacci con manejo de excepciones

```python
def generar_fibonacci(n):

    if n <= 0:
        print("La cantidad de t rminos debe ser un n mero positivo mayor que cero.")
        return
```

```
6
7       a = 0
8       b = 1
9
10      print("Serie de Fibonacci:")
11
12      print(a)
13
14      if n > 1:
15          print(b)
16
17      for i in range(3, n + 1):
18          siguiente_termino = a + b
19          print(siguiente_termino)
20
21          a = b
22          b = siguiente_termino
23
24
25  def main():
26      try:
27          n_terminos = float(input("Ingrese la cantidad de
        t rminos de Fibonacci a generar: "))
28
29          if n_terminos != int(n_terminos) or n_terminos <= 0:
30              raise ValueError("Debe ingresar un n mero entero
        positivo (ej: 10).")
31
32          generar_fibonacci(int(n_terminos))
33
34      except ValueError as ve:
35          print("Error:", ve)
36
37      except Exception as e:
38          print("Ocurri  un problema inesperado:", e)
39
40  if __name__ == "__main__":
41      main()
```

Listing 3: Generación controlada de la secuencia de Fibonacci

Ejecución:

```
Entrada 1:
    Ingrese la cantidad de términos de Fibonacci a generar: 5
    Serie de Fibonacci:
    0
    1
    1
    2
    3
Entrada 2:
```

```
    Ingrese la cantidad de términos de Fibonacci a generar: -5
    Error: Debe ingresar un número entero positivo (ej: 10)
Entrada 3:
    Ingrese la cantidad de términos de Fibonacci a generar: letras
    Error: could not convert string to float: 'letras'
```