


MANEJO DE EXCEPCIONES EN PYTHON



Buenas prácticas en el manejo de excepciones y entradas del usuario en Python

Introducción

En esta lección aprenderás a declarar correctamente las variables dentro y fuera de bloques `try-except`, además de cómo solicitar entradas al usuario y convertirlas a tipos numéricos. También verás cómo capturar errores comunes como `ValueError` y `ZeroDivisionError`, para mantener un flujo controlado en tus programas. 

Paso 1: Crear el archivo base del proyecto

 Ruta y nombre del archivo: `manejo-excepciones.py`



Paso 2: Declarar variables fuera del bloque try



Descripción:

Declaramos `resultado` fuera del bloque `try`, ya que se usará también fuera de este bloque, es decir, tiene un ámbito mayor. Esto evita errores de variables no definidas. Las variables `a` y `b` pueden ir dentro del bloque porque solo se usan ahí.



Archivo: `manejo-excepciones.py`

```
resultado = None

try:
    a = int(input('Proporciona el primer número: '))
    b = int(input('Proporciona el segundo número: '))
    resultado = a / b
except ZeroDivisionError as e:
    print(f'Ocurrió un error (ZeroDivisionError): {e}, tipo: {type(e)}')
except TypeError as e:
    print(f'Ocurrió un error (TypeError): {e}, tipo: {type(e)}')
except Exception as e:
    print(f'Ocurrió un error (Exception): {e}, tipo: {type(e)}')

print(f'Resultado: {resultado}')
print('Continuamos...')
```



Explicación:

La variable `resultado` se declara antes del bloque `try` porque luego se imprime. Si se declarara dentro, sería inaccesible desde fuera. Las demás variables pueden estar dentro sin problema si solo se usan ahí.



Recordatorio: Ámbito o Alcance de una variable

El **ámbito** o alcance de una variable se refiere a la sección del programa donde esa variable es **visible y accesible**. En Python, si una variable se declara **dentro de un bloque `try`, `except`, o una función**, solo puede usarse **dentro de ese bloque**. Eso se conoce como **ámbito local**.

Por eso:

- La variable `resultado` se declara **fuera** del bloque `try`, ya que **se utiliza también después** del manejo de excepciones (fuera del bloque). Así evitamos errores como "variable no definida".
- En cambio, las variables `a` y `b` **solo se usan dentro** del bloque `try`, por lo tanto, **pueden declararse allí sin problema**.

Este patrón ayuda a mantener el código limpio, claro y sin errores por variables fuera de su alcance. 🌸 ✨

Paso 3: Probar errores comunes

Descripción:

Realizamos diferentes pruebas para generar errores y observar cómo se capturan.

Casos que puedes probar:

- Ingresar 10 y luego 0 → lanza `ZeroDivisionError`
- Ingresar 'a' y luego 2 → lanza `ValueError`
- Ingresar 10 y luego 'b' → lanza `ValueError`
- Ingresar 10 y 2 → resultado correcto: 5.0

Explicación:

El bloque `except` captura las excepciones según el tipo. Si no se convierte correctamente a número, se lanza `ValueError`. Si se divide entre cero, se lanza `ZeroDivisionError`. Todo es controlado sin terminar abruptamente.

Conclusión

En esta lección aprendiste cómo declarar variables correctamente en relación con los bloques `try-except`, cómo manejar entradas del usuario de forma segura y cómo capturar errores comunes como divisiones por cero o entradas inválidas. Esto mejora la experiencia del usuario y hace tu programa mucho más robusto.



Sigue adelante con tu aprendizaje 🚀, ¡el esfuerzo vale la pena!

¡Saludos! 🙌

Ing. Marcela Gamiño e Ing. Ubaldo Acosta

Fundadores de [GlobalMentoring.com.mx](https://www.globalmentoring.com.mx)