


MANEJO DE EXCEPCIONES EN PYTHON



Procesar excepciones con diferentes tipos de errores en Python

Introducción

En esta lección aprenderás cómo capturar distintas excepciones de forma segura en Python. Definiremos variables para controlar mejor la ejecución, exploraremos cómo afectan los distintos tipos de errores (como `ZeroDivisionError` y `TypeError`), y entenderás por qué conviene utilizar clases de excepción más generales para evitar que el programa termine abruptamente. 

Paso 1: Usar el mismo archivo base del proyecto

 Ruta y nombre del archivo:

<https://www.globalmentoring.com.mx>

manejo_excepciones.py

⚙ Paso 2: Definir variables y estructura básica

🔍 Descripción:

Definimos las variables necesarias, incluyendo una variable `resultado` inicializada como `None`, y luego declaramos dos variables numéricas `a` y `b`. En este caso `b` tiene valor 0, lo que provocará una división entre cero.

📄 Archivo: `manejo-excepciones.py`

```
resultado = None
a = 10
b = 0

try:
    resultado = a / b
except ZeroDivisionError as e:
    print(f'Ocurrió un error: {e}')

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

📖 Explicación:

La división entre `10 / 0` lanza una excepción que es capturada por el bloque `except`. Se imprime el error, pero el programa continúa y la variable `resultado` conserva su valor original (`None`).

🧪 Paso 3: Probar con otro tipo de excepción (TypeError)

🔍 Descripción:

Ahora provocamos un `TypeError` cambiando la variable `a` a una cadena de texto en lugar de un número. Esto genera un nuevo tipo de excepción.

📄 Archivo: `manejo-excepciones.py`

```
resultado = None
a = '10' # Valor tipo cadena
b = 0

try:
    resultado = a / b
```

```
except ZeroDivisionError as e:
    print(f'Ocurrió un error: {e}')

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

📖 Explicación:

Aquí ocurre un `TypeError` ya que no se puede dividir una cadena por un número. Como la excepción específica capturada es `ZeroDivisionError`, este nuevo error no es atrapado, y el programa termina de manera abrupta.

🔄 Paso 4: Usar una clase de excepción genérica

🔍 Descripción:

Actualizamos el `except` para capturar cualquier tipo de error utilizando la clase padre `Exception`.

📄 Archivo: `manejo-excepciones.py`

```
resultado = None
a = '10'
b = 0

try:
    resultado = a / b
except Exception as e:
    print(f'Ocurrió un error: {e}')

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

📖 Explicación:

Con `Exception` como clase de captura, cualquier error derivado, como `ZeroDivisionError` o `TypeError`, es capturado. Esto permite que el programa siga funcionando sin detenerse.

✅ Conclusión

En esta lección aprendiste cómo utilizar variables para capturar resultados de manera segura y cómo manejar distintos tipos de errores en Python. Comprobaste la diferencia entre capturar errores específicos (`ZeroDivisionError`) y errores generales (`Exception`), y entendiste la importancia de usar clases padre para evitar fallos inesperados. 🧠

¡Muy bien hecho! Ahora puedes enfrentar errores con mayor confianza en tus programas Python.  

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)