

MANEJO DE EXCEPCIONES EN PYTHON



🧩 Excepciones personalizadas en Python y uso de `raise`

Introducción

En esta lección aprenderás a **crear tus propias clases de excepción** en Python y a **lanzar errores personalizados** utilizando la palabra clave `raise`. Esta técnica te permite mejorar el control de errores en tu programa y definir tus propias reglas de validación. 🛡️

🧱 Paso 1: Crear el archivo para la clase personalizada

📄 **Ruta y nombre del archivo:** `NumerosIdenticosException.py`

Descripción:

Aquí definiremos una nueva clase para representar un error personalizado cuando se detectan dos números iguales.

Código del archivo `NumerosIdenticosException.py`:

```
class NumerosIdenticosException(Exception):
    def __init__(self, mensaje):
        self.message = mensaje
```

Explicación:

- Esta clase hereda de `Exception`, que es la clase base para manejar errores en Python.
- Al constructor `__init__` se le pasa un mensaje personalizado que se puede mostrar al usuario cuando se lanza esta excepción.

Paso 2: Importar y usar la excepción personalizada

Ruta y nombre del archivo principal:

`manejo_excepciones.py`

Descripción:

Ahora usaremos la clase personalizada para **lanzar una excepción** si los números ingresados por el usuario son idénticos.

Código desde `manejo_excepciones.py`:

```
from NumerosIdenticosException import NumerosIdenticosException

resultado = None
try:
    a = int(input('Primer número: '))
    b = int(input('Segundo número: '))
    if a == b:
        raise NumerosIdenticosException('números idénticos')
    resultado = a/b
except ZeroDivisionError as e:
    print(f'ZeroDivisionError - Ocurrió un error: {e} , {type(e)}')
except TypeError as e:
    print(f'TypeError - Ocurrió un error: {e} , {type(e)}')
except Exception as e:
```

```
print(f'Exception - Ocurrió un error: {e} , {type(e)}')
else:
    print('No se arrojó ninguna excepción')
finally:
    print('Ejecución del bloque finally')

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

Explicación paso a paso:

- Se importó la clase personalizada desde el archivo anterior.
- Se comparan los números `a` y `b`. Si son iguales, se lanza una excepción usando `raise`.
- Esa excepción personalizada será capturada por el bloque `except Exception`, ya que es una clase hija de `Exception`.

Importante: Uso general de `raise`

Observación adicional:

La instrucción `raise` también puede usarse para lanzar otras excepciones estándar de Python, como:

```
raise ValueError('Error de valor')
```



Explicación:

Esto permite crear validaciones a medida en cualquier parte de tu código, incluso sin necesidad de una clase personalizada.

Conclusión

En esta lección aprendiste dos cosas muy poderosas en Python:

- Cómo **crear tus propias clases de excepción** heredando de `Exception`
- Cómo **lanzar errores manualmente** usando la palabra clave `raise`

Esto te da mucho más control sobre la validación de datos y el comportamiento de tus programas.  

¡Sigue explorando el mundo de Python y sus excepciones!  

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)