Python: Manejo de excepciones

Centro de Servicios y Gestión Empresarial
SENA Regional Antioquia

@SENAComunica

www.sena.edu.co

Conceptualización

Manejo de Excepciones

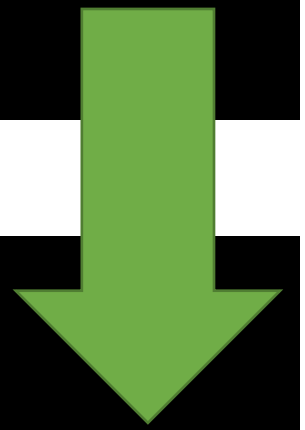
Python utiliza **try-except** para manejar errores y excepciones en **tiempo de ejecución**, evitando que el programa se **detenga abruptamente**.

Sintaxis:
try → Contiene el código que podría generar un error.
except → Captura la excepción específica y ejecuta un código alternativo.

```
try:
    # Código que puede generar un error
except Exception as e:
    # Código que se ejecuta si ocurre la excepción
```

Manejo de Excepciones

```
operacion = 10 / 0
print("El resultado de la operación es:", operacion)
```



```
try:
    # Código que puede generar un error
    operacion = 10 / 0  # División por cero
    print("El resultado de la operación es:", operacion)
except ZeroDivisionError:
    # Código que se ejecuta si ocurre la excepción
    print("Error: No se puede dividir entre cero.")
```

Manejo de Múltiples Excepciones

```
try:
    num1 = int(input("Ingrese el numerador: "))
    num2 = int(input("Ingrese el denominador: "))
    resultado = num1 / num2
    print(f"Resultado: {resultado}")
except ZeroDivisionError:
    print("Error: No se puede dividir por cero.")
except ValueError:
    print("Error: Debe ingresar un número válido.")
```

Uso de else **y** finally

Sintaxis:

```
try:
    # Código que puede generar un error
except Exception as e:
    # Código que se ejecuta si ocurre la excepción
else:
    # Código que se ejecuta si no ocurre la excepción
finally:
    # Código que se ejecuta siempre, haya o no ocurrido la
excepción
```

Uso de else **y** finally

```
try:
    num1 = int(input("Ingrese el numerador: "))
    num2 = int(input("Ingrese el denominador: "))
    resultado = num1 / num2
except ZeroDivisionError:
    print("No se puede dividir por cero")
except ValueError:
    print("Debe ingresar un número")
else:
    print("La división es: ", resultado)
finally:
    print("Fin del programa")
```

Manejo de Excepciones

En Python, se utiliza **raise** para lanzar excepciones personalizadas.

Sintaxis:

```
# Crear excepciones personalizadas
raise NombreDeLaExcepcion("Mensaje de error personalizado")
```

Manejo de Excepciones

```
try:
    num1 = int(input("Ingrese el numerador: "))
    num2 = int(input("Ingrese el denominador: "))
    if num1 < 0 or num2 < 0:
        raise ValueError("Los números deben ser positivos")
    resultado = num1 / num2
except ZeroDivisionError:
    print("No se puede dividir por cero")
except ValueError:
    print("Debe ingresar un número o Los números deben ser positivos")
else:
    print("La división es: ", resultado)
finally:
    print("Fin del programa")
```

Manejo de Excepciones

```
try:
    num1 = int(input("Ingrese el numerador: "))
    num2 = int(input("Ingrese el denominador: "))
    if num1 < 0 or num2 < 0:
        raise Exception ("Los números deben ser positivos")
    resultado = num1 / num2
except ZeroDivisionError:
    print("No se puede dividir por cero")
except ValueError:
    print("Debe ingresar un número")
except Exception as e:
    print(e)
else:
    print("La división es: ", resultado)
finally:
    print("Fin del programa")
```

Manejo de Excepciones

Bloque	Función	Ejemplo
try:	Contiene el código que puede generar un error.	try: resultado = 10 / 0
except TipoError:	Captura errores específicos (ej. ZeroDivisionError).	except ZeroDivisionError:
except Exception as e:	Captura cualquier error y lo almacena en e.	except Exception as e: print(e)
else:	Se ejecuta solo si **no hay errores** en el try.	else: print("Operación exitosa")
finally:	Se ejecuta **siempre**, ocurra o no un error.	finally: print("Ejecución finalizada.")
raise TipoError("Mensaje")	Lanza manualmente una excepción.	raise ValueError("Error personalizado")

Manejo de Excepciones

Excepción	Causa Común
ZeroDivisionError	División entre cero.
ValueError	Conversión de tipo inválida.
TypeError	Operaciones entre tipos incompatibles.
IndexError	Acceso a un índice fuera de rango.
KeyError	Acceso a una clave inexistente en un diccionario.
AttributeError	Uso de un método/atributo inexistente en un objeto.
FileNotFoundError	Archivo no encontrado.
ImportError	Módulo no encontrado.
MemoryError	Uso excesivo de memoria.
RecursionError	Llamada recursiva sin fin.

Ejericicio de clase

Calificaciones Académicas
Descripción: Un docente necesita registrar N calificaciones, calcular su promedio y determinar si un estudiante aprueba o reprueba. Si el promedio es 3.0 o superior, el estudiante aprueba; si es menor, reprueba.

Restricciones:
- Solicitar al usuario la cantidad de calificaciones a ingresar.
- Validar que cada calificación esté entre 0.0 y 5.0.
- Usar **try-except** para evitar errores en la entrada de datos.
- Calcular el promedio y mostrar si aprueba o reprueba.
- Las calificaciones deben ser almacenadas en una lista.

G R A C I A S

Presentó: Alvaro Pérez Niño
Instructor Técnico
Correo: aperezn@sena.edu.co
http://centrodeserviciosygestionempresarial.blogspot.com/