Programación con python

Exception (Excepción)

Exception (Excepción)

Una excepción, en Python, se lanza (raised) cuando ocurre algo inesperado durante la ejecución del programa.
Por ejemplo, si intentas convertir la cadena "hello" en un número entero (integer), tu programa generará un error del tipo ValueError.

Nota:

Usa try/except para manejar errores inevitables, no para ocultarlos.

Manejo de excepciones con try/except

Si anticipas que una parte de tu código podría, en algunos casos, generar una excepción, puedes rodear esa sección con un bloque try/except para evitar que el programa se bloquee por completo.

```
try:
    # Código que podría generar un error
    ...
except TipoDeError:
    # Código que se ejecuta si ocurre ese error
    ...
```

Manejo de excepciones con try/except

También puedes capturar varios tipos de errores o usar una variable para obtener detalles del error:

```
try:
    # Código propenso a error
    ...
except TipoDeError as e:
    print(f"Ocurrió un error: {e}")
```

Manejo de excepciones con try/except

Y si no sabes qué tipo de error puede ocurrir:

```
try:
    ...
except Exception as e:
    print(f"Error inesperado: {e}")
```

ValueError

Ocurre cuando el tipo de dato es correcto pero el valor no tiene sentido.

```
num = int("hola")
# ValueError: invalid literal for int() with base 10: 'hola'
```

ZeroDivisionError

Intentar dividir un número entre cero.

```
resultado = 10 / 0
# ZeroDivisionError: division by zero
```

TypeError

Sucede cuando usas un tipo de dato no compatible.

```
suma = "10" + 5
# TypeError: can only concatenate str (not "int") to str
```

IndexError

Intentar acceder a un índice que no existe en una lista.

```
lista = [1, 2, 3]
print(lista[5])
# IndexError: list index out of range
```

KeyError

Ocurre al intentar acceder a una clave inexistente en un diccionario.

```
datos = {"nombre": "Luis"}
print(datos["edad"])
# KeyError: 'edad'
```

Consejo/Tip

Puedes combinar varios tipos de excepción en un mismo bloque:

```
try:
    ...
except (ValueError, TypeError, ZeroDivisionError) as e:
    print(f"Error detectado: {e}")
```

Bloque finally en Python

El bloque finally se utiliza junto con try y except para asegurar que un bloque de código se ejecute siempre, sin importar si ocurrió una excepción o no.

Es muy útil cuando necesitas cerrar archivos, conexiones o liberar recursos, incluso si el programa falló.

Bloque finally en Python

```
try:
    # Código que puede generar un error
    ...
except TipoDeError:
    # Código que se ejecuta si ocurre el error
    ...
finally:
    # Código que siempre se ejecuta, haya error o no
    ...
```

Bloque finally en Python

```
try:
    x = int(input("Ingresa un número: "))
    print(10 / x)
except ZeroDivisionError:
    print("No puedes dividir entre cero.")
finally:
    print("Fin del programa (bloque finally ejecutado).")
```

Cuando usar y cuando no

Usar Try:Except cuando:

- El error es posible, real y viene del usuario o del entorno.
- El programa debe continuar aunque haya un error.
- Quieres dar mensajes claros al usuario.
- Manejas archivos, internet, BD, hardware, etc.

Ejemplo cuando sí, dar un mensaje claro al usuario

```
try:
    edad = int(input("Ingresa tu edad: "))
except ValueError:
    print("Error: La edad debe ser un número entero.")
```

Cuando usar y cuando no

NO usar Try:Except cuando:

- El error es tuyo como programador y se debe corregir.
- Puedes validar antes de ejecutar.
- Lo usas solo “por si acaso”.
- Ocultas errores que deberías arreglar.

Ejemplo cuando No, Cuando lo correcto es validar antes

```
try:
    print(lista[2])
except:
    print("No existe el índice")
```

En este caso lo correcto es manejarlo asi:

```
if len(lista) > 2:
    print(lista[2])
else:
    print("No existe el índice")
```

Actividad

1. Crea un programa que:

   Pida al usuario que ingrese dos números.

   Intente dividir el primero entre el segundo.

   Maneje los siguientes errores:

   Si el usuario escribe texto en lugar de un número → ValueError

   Si intenta dividir entre cero → ZeroDivisionError

   Cualquier otro error inesperado → Exception

   Sigue pidiendo dos números hasta que la división se realice correctamente sin errores.

NO USE INTELIGENCIA ARTIFICIAL PARA SU TRABAJO
ENTREGUELA EN LA SECCION DE TAREAS

Actividad

Pide una lista fija lst = [10, 20, 30] y un índice del usuario.

Muestra el elemento en ese índice.

Maneja ValueError (si el índice no es número) e IndexError (si está fuera de rango).

Siempre imprime “Fin de la operación” (usa finally).

NO USE INTELIGENCIA ARTIFICIAL PARA SU TRABAJO
ENTREGUELA EN LA SECCION DE TAREAS

Actividad

Dado info = {"nombre": "Luis", "edad": 18}, pide una clave al usuario y muestra su valor.

Si la clave no existe, imprime “Clave no encontrada” (KeyError).

Si el usuario deja vacío, lanza manualmente un ValueError con raise.

NO USE INTELIGENCIA ARTIFICIAL PARA SU TRABAJO
ENTREGUELA EN LA SECCION DE TAREAS