MANEJO DE EXCEPCIONES Y VALIDACIÓN

1. Fundamentación

La validación de datos es un pilar fundamental en el desarrollo de software para garantizar la robustez, la estabilidad y la seguridad de las aplicaciones. En los niveles iniciales de programación, los estudiantes suelen asumir que el usuario final interactuará con el sistema de forma perfecta. Sin embargo, la realidad demuestra que las entradas erróneas o inesperadas son comunes.

Introducir el bloque estructural try / except permite a los alumnos anticiparse a los fallos de ejecución (errores en tiempo de ejecución), brindándoles herramientas para controlar el flujo de sus programas y evitar cierres abruptos (cuelgues). Esto no solo mejora sus habilidades lógicas de codificación, sino que fomenta una actitud orientada a la experiencia de usuario (UX) y a la creación de software confiable

Objetivos

Objetivo General:

- ✔ Controlar y gestionar errores comunes mediante la validación de entradas de datos en la consola de Python.

Objetivos Particulares:

- ✔ Comprender e implementar la estructura de control try / except para interceptar excepciones.
- ✔ Reconocer y diferenciar los errores de conversión (ValueError) de otros fallos lógicos habituales.
- ✔ Combinar estructuras iterativas (while) con el manejo de excepciones para asegurar que un dato sea correcto antes de procesarlo.

La Problemática de los Datos Inválidos

Cuando un programa solicita una entrada por teclado a través de input(), Python recibe una cadena de texto (string). Para realizar operaciones matemáticas, es obligatorio convertir ese texto a un tipo numérico (como int o float). Si el usuario introduce caracteres no numéricos (letras, símbolos o simplemente presiona "Enter" vacío), el intérprete de Python colapsa porque no sabe cómo convertir esas letras en números.

```
# Código vulnerable
edad = int(input("Ingrese su edad: "))
print("El año que viene tendrás:", edad + 1)
```

Si el usuario escribe "veinte", el programa se detiene lanzando un ValueError: invalid literal for int().

La Red de Seguridad — try y except

Concepto: Python provee un mecanismo para "atrapasueños" o interceptar estos fallos antes de que congelen la aplicación.

- ✔ try: Bloque donde colocamos el código "peligroso" que podría llegar a fallar.
- ✔ except: Bloque de contingencia que se ejecuta **únicamente** si el bloque try falló.

La sentencia try funciona de la siguiente manera.

- Primero, se ejecuta la cláusula *try* (la(s) linea(s) entre las palabras reservadas try y la except).
- Si no ocurre ninguna excepción, la cláusula *except* se omite y la ejecución de la cláusula try finaliza.
- Si ocurre una excepción durante la ejecución de la cláusula try, se omite el resto de la cláusula. Luego, si su tipo coincide con la excepción nombrada después de la palabra clave except, se ejecuta la *cláusula except*, y luego la ejecución continúa después del bloque try/except.

```
try:
    edad = int(input("Ingrese su edad: "))
    print("El año que viene tendrás:", edad + 1)

except ValueError:
    print("Error: ¡Debes ingresar un número entero válido!")
```

El Bucle de Validación Infalible

El problema pendiente: El código anterior evita que el programa se rompa, pero si el usuario se equivoca, el programa termina igual sin calcular nada.

Solución combinada: Para obligar al usuario a ingresar el dato correcto, encerramos la estructura en un bucle while True (infinito). El bucle solo se romperá con la instrucción break si la conversión numérica tiene éxito y cumple con nuestras condiciones lógicas.

Código Patrón de Validación:

```
while True:
    try:
        precio = float(input("Ingrese el precio del producto: $"))
        if precio > 0:
            break # Rompe el bucle porque el dato es correcto y lógico
        else:
            print("El precio debe ser un número positivo mayor a 0.")
    except ValueError:
        print("Entrada inválida. Por favor, use números enteros o decimales (con punto).")

print(f"Precio registrado con éxito: ${precio}")
```

El Validador de Divisiones

Actividad 1:

Escribir un programa que solicite al usuario dos números enteros y muestre el resultado de dividir el primero por el segundo. El programa debe prevenir dos tipos de errores comunes:

1. Que el usuario ingrese letras en lugar de números (ValueError).
2. Que el usuario intente dividir por cero (ZeroDivisionError).

Actividad 2: Registro de Alumnos con Validación Estricta

Registro de Alumnos con Validación Estricta

Consigna: Diseñar un sistema de login o registro por consola. El programa debe pedir dos datos clave y validar de forma continua (usando while True) hasta que ambos cumplan las reglas:

1. **Nombre de usuario:** No puede quedar vacío (validar que el texto tenga una longitud mayor a cero).
2. **Edad:** Debe ser un número entero válido y debe estar en el rango de 13 a 18 años inclusive (simulando un sistema escolar secundario).

Actividad 3: Sistema de Caja Registradora Comercial

Consigna: Desarrollar un programa que simule el cobro de una compra en una tienda. El sistema debe realizar los siguientes pasos de forma secuencial y blindada contra errores:

1. Preguntar cuántos productos lleva el cliente (debe ser un entero mayor a 0).
2. Mediante un bucle controlado por esa cantidad, pedir el precio de cada producto (debe ser un número flotante mayor a 0). Si el usuario se equivoca al ingresar un precio, el sistema debe capturar el error, avisar y **volver a pedir el mismo precio** sin perder el progreso de los montos anteriores.
3. Finalmente, mostrar el total acumulado de la compra en la consola.