Clase 11 – Validación de entradas

Fundamentación

La validación de datos es esencial para evitar errores y hacer programas más robustos. Introducir try/except enseña a manejar excepciones y prevenir cierres inesperados.

Objetivo General

Controlar errores comunes mediante validación de entradas.

Objetivos Particulares

- Usar try/except para manejar errores en conversiones de datos.
- Asegurar que los valores ingresados sean válidos antes de usarlos.

Contenidos

- **Conceptuales:**
  - Manejo de excepciones (try, except).
  - Tipos de errores comunes en entrada de datos.
- **Procedimentales:**
  - Validar que un número ingresado sea entero positivo.
  - Evitar fallos en Turtle por valores inválidos.
- **Actitudinales:**
  - Prevenir errores antes de que ocurran.
  - Mejorar la experiencia del usuario.

¿Por qué validar datos?

- Un usuario puede escribir algo que el programa **no espera**.
- Si no controlamos, el programa **se detiene con un error**.
- Validar entradas **evita fallos** y mejora la experiencia de uso.

Ejemplo de problema sin validación:

```
numero = int(input("Ingrese un número: "))
print("El doble es:", numero * 2)
```

Si el usuario escribe "hola", el programa lanza un error ValueError.

```
Ingrese un número: hola
Traceback (most recent call last):
  File "/home/main.py", line 1, in <module>
    numero = int(input("Ingrese un número: "))
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ValueError: invalid literal for int() with base 10: 'hola'
```

Solución: try y except

- try → intenta ejecutar un bloque de código.
- except → define qué hacer si ocurre un error.

```
try:
    # Código que podría fallar
except:
    # Qué hacer si falla
```

Tipos de errores comunes al ingresar datos:

- ValueError → ocurre al intentar convertir letras a número.
- ZeroDivisionError → división por cero.
- TypeError → operación entre tipos incompatibles.

Ejemplo 1 – Validación básica

```
try:
    numero = int(input("Ingrese un número entero: "))
    print("El doble es:", numero * 2)
except ValueError:
    print("Error: Debe ingresar un número entero.")
```

Explicación:

- Si el usuario escribe letras, entra en el except y muestra el mensaje de error sin cerrar el programa.

Ejemplo 2 – Validar hasta que sea correcto

```
while True:
    try:
        numero = int(input("Ingrese un número entero positivo: "))
        if numero > 0:
            break
        else:
            print("Debe ser un número mayor que cero.")
    except ValueError:
        print("Error: Debe ingresar un número entero.")

print("Número válido:", numero)
```

Explicación:

- El while True repite hasta que se cumpla la condición.
- Si hay error o el número no es positivo, vuelve a pedir el dato.

Ejemplo 3 – Validar entradas para Turtle

```
import turtle

while True:
    try:
        lados = int(input("Ingrese número de lados (mínimo 3): "))
        if lados >= 3:
            break
        else:
            print("Debe ser un número mayor o igual a 3.")
    except ValueError:
        print("Error: Ingrese un número entero.")

tamaño = 100

for i in range(lados):
    turtle.forward(tamaño)
    turtle.right(360 / lados)

turtle.done()
```

Explicación:

- Se asegura que el usuario ingrese **un número entero y mayor o igual a 3**.
- Con eso se dibuja un polígono válido en Turtle.

Actividad práctica guiada

Objetivo: validar datos para dibujar una figura.

Instrucciones:

1. Pedir al usuario:
    - Número de lados (mínimo 3).
    - Tamaño de la figura (mínimo 20).
    - Color del trazo (validar que no esté vacío).
2. Dibujar la figura con Turtle usando esos valores.
3. Mostrar un mensaje "Dibujo completado con éxito".