Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Otras herramientas para el manejar errores 2.

Alumno: Bryan De Anda Reyes

Código de alumno: 216195537

Materia: Computación Tolerante a Fallas (D06)

Horario: lunes y miércoles (11:00 – 13:00)

Carrera: INCO

Fecha de entrega: 08 de febrero de 2022

Programa para manejo de errores.

Este código lo realice en el lenguaje de programación Python, en el cual implemente un menú de operaciones (división y raíz cuadrada) para manejar diferentes excepciones.

Lo que hice primero fue crear las funciones para cada operación, primero tenemos la función para la raíz cuadrada, es necesario utilizar la librería “math” para su funcionamiento, utilicé un “if” para comprobar si el numero al que se le va a sacar la raíz es menor a “0”, dentro de este “if” utilicé la función “raise” para crear una excepción propia, esta en este caso no funciona por si solo, es necesaria otra excepción al momento de llamar a la función y sí es número es mayor a cero, realiza la operación.

```
def calculaRaiz(num1):

    if num1<0:
        raise ValueError ("El número no puede ser negativo")
    else:
        return math.sqrt(num1)
```

Ahora tenemos la función de la división, en la cual se utiliza la declaración “try” para que ejecute el primer bloque de código y si la variable “num2” es igual a cero se generará una excepción para que el código siga funcionando y muestra unos mensajes, en cambio, si “num2” es diferente de 0 regresa el resultado de la división.

```
def Division(num1, num2):

    try:
        return num1/num2
    except ZeroDivisionError:
        print("No se puede dividir entre 0")
        return "Operación erronea"
```

Ahora creé un menú para elegir la operación a realizar, aquí también implemente una excepción para en el caso de que se ingrese un dato que no sea un número muestre un mensaje y el resto del código se siga ejecutando sin problema.

Por otro lado, independientemente de la opción que se eligió, existe otra excepción por si a los números ingresados que se desea realizar una operación no sean correctos. En

la opción de a división después de que se ingresaron los números que se les aplicara la operación simplemente manda a llamar la función “Division()”, en el caso de la raíz cuadrada es diferente, aquí se implementó otra excepción en la cual se intenta realizar la operación pero si no se tiene éxito se definió una excepción con nombre de “ErrorDeNumero”, y se manda a imprimir la excepción propia creada dentro de la función “calculaRaiz()”, por ultimo tenemos la función “finally” que mostrará un mensaje independientemente de si se realizó la operación o no.

```
print("-----Menú división y raiz cuadrada-----")
print("1. División")
print("2. Raiz cuadrada")

try:
    op = (int(input("Introduce un numero: ")))
    if op == 1:
        try:
            opc1 = int(input("Ingrese el primer número: "))
            opc2 = int(input("Ingrese el segundo número: "))
            print(Division(opc1,opc2))
        except ValueError:
            print("Valores no validos")

    elif op == 2:
        try:
            opc1 = int(input("Ingrese un número: "))
            try:
                print(calculaRaiz(opc1))

            except ValueError as ErrorDeNumero:

                print(ErrorDeNumero)

            finally:
                print("El programa terminó")
        except ValueError:
            print("Valor no valido")
    else:
        print("Opción no valida")
except ValueError:
    print("Opción no valida")
```

A continuación, se muestra la ejecución del programa:

1. Dato no valido en el menú

```
-----Menú división y raiz cuadrada-----
1. División
2. Raiz cuadrada

Introduce un numero: qe
Opción no valida
```

2. Números incorrectos en la división y la raíz cuadrada

```
-----Menú división y raiz cuadrada-----
1. División
2. Raiz cuadrada

Introduce un numero: 1

Ingrese el primer número: a
Valores no validos
```

```
-----Menú división y raiz cuadrada-----
1. División
2. Raiz cuadrada

Introduce un numero: 2

Ingrese un número: nose
Valor no valido
```

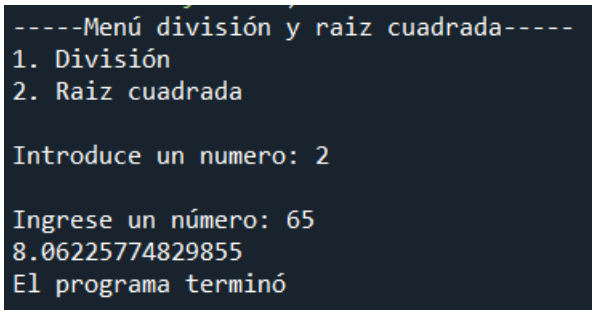
3. Operación valida en la división

```
-----Menú división y raiz cuadrada-----
1. División
2. Raiz cuadrada

Introduce un numero: 1

Ingrese el primer número: 72

Ingrese el segundo número: 4
18.0
```

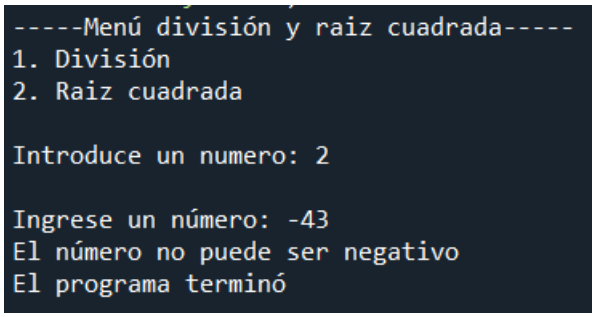
4. Operación no valida en la división

```
-----Menú división y raiz cuadrada-----
1. División
2. Raiz cuadrada

Introduce un numero: 1

Ingrese el primer número: 65

Ingrese el segundo número: 0
No se puede dividir entre 0
Operación erronea
```

5. Operación valida en la raíz cuadrada



6. Operación no valida en la raíz cuadrada



Link al código en el repositorio: https://github.com/BryanDeAnda/Programa1-Tolerante-a-fallas.git

Conclusión

Para finalizar quiero decir que, ahora después de conocer la importancia de utilizar diferentes metodos para hacer nuestros códigos mas robustos o sin errores, en este caso el uso de las excepciones, me ayudo bastante.

Decidí realizar este código por que incluye varias funciones de las excepciones, como por ejemplo la de crear una excepción propia o la de asignarle un nombre a una excepción, puede ser que hacer esto no fuera necesario para que este código funcione correctamente, sin embargo, me ayudo bastante a entender de una mejor manera las diferentes habilidades que nos brinda esta herramienta.