Herramientas para el manejar errores 2

Nombre del alumno: José Ulises Vallejo Sierra

Código: 219747905

Sección: D06

Curso: Computación Tolerante a Fallas

Nombre del profesor: Michel Emanuel López Franco

Herramientas para el manejo de errores

Técnica a utilizar: Try except con cláusula raise

Como vimos en el reporte anterior de las técnicas de manejo de errores, la función try permite ejecutar una línea de código cualquiera que puede ser susceptible a errores, por lo que se realiza la función except, la cuál permite hacer una excepción al error y de esta manera no suspender la ejecución del programa ni cerrar abruptamente el mismo, sino que se le informa al usuario el error que ocasionó y le da la posibilidad de enmendar o corregir el error.

Dicho error puede ser sintáctico, semántico o de ejecución, pero hay ocasiones en que la excepción al error no está ya especificada en el programa, o simplemente el desarrollador quiere personalizar un poco más el error cometido por el usuario, para ello se utiliza la cláusula raise, la cual explicaré de manera detallada con el siguiente programa de ejemplo:

Programa: Adivinar un número generado por la computadora entre el 1 y el 5.

Analizando este programa, he decidido utilizar la clausula raise para especificar el tipo de error que comete el usuario si es que ingresa un número que no se encuentre en el rango establecido, ya sea que este por debajo o por encima de dicho rango; para ello, comenzaré definiendo una clase para estos dos posibles casos en que el usuario no ingresó un valor dentro del rango.

```
1   from random import randint
2
3
4   class ErrorMinimo(Exception):
5       def __init__(self, *args):
6           if args:
7               self.message = args[0]
8           else:
9               self.message = None
10
11      def __str__(self):
12          if self.message:
13              return f"ErrorMinimo: {self.message}"
14          else:
15              "Se generó ErrorMinimo."
```

Figura 1. Clase para la excepción personalizada ErrorMinimo, la cual sirve para cuando el usuario ingresa un valor menor al rango.

```
17  class ErrorMaximo(Exception):
18      def __init__(self, *args):
19          if args:
20              self.message = args[0]
21          else:
22              self.message = None
23
24      def __str__(self):
25          if self.message:
26              return f"ErrorMaximo: {self.message}"
27          else:
28              "Se generó ErrorMaximo."
29
```

Figura 2. Clase para la excepción personalizada ErrorMaximo, la cual sirve para cuando el usuario ingresa un valor mayor al rango.

Ahora definimos los rangos a utilizar

```
30
31  valormin = 1
32  valormax = 5
33
```

Figura 3. Para este caso, el usuario adivinará un número generado entre el número 1 y el número 5.

Después, definimos dentro de un ciclo while infinito, el bloque de código susceptible a errores por parte del usuario.

```
34  while True:
35      try:
36          num = int(input(f"\nEscribe un valor entre {valormin} y {valormax}: "))
37
38          if num < valormin:
39              raise ErrorMinimo("El valor es menor al rango")
40          elif num > valormax:
41              raise ErrorMaximo("El valor es mayor al rango")
42
```

Figura 4. El usuario escribe un número para adivinar el que está pensando la computadora, si este es menor entonces con la cláusula raise creamos la excepción ErrorMinimo indicando que el valor es menor, y si éste último es mayor, creamos con rise la excepción que ErrorMaximo que nos lo indica.

Ahora llamamos dichas excepciones en el caso que el usuario haya indicado un valor fuera de rango, la excepción ValueError indica que es un valor no apropiado, y especificamos con las excepciones creadas por raise para cada caso posible de valor ingresado, ya sea que es menor (ErrorMinimo) o mayor (ErrorMaximo) e imprimimos el contenido del mensaje de error (e).

```
44      except ValueError:
45          print("Debe der un entero dentro del rango")
46      except ErrorMinimo as e:
47          print("ERROR: ",e)
48      except ErrorMaximo as e:
49          print("ERROR: ",e)
```

Figura 5. Si no se ingresó un valor apropiado, creamos la excepción para cada caso, evitando el cierre abrupto del programa.

Por último, como sabemos, el bloque de código dentro de la cláusula finally se ejecuta siempre, por lo que aquí creamos la variable aleatoria que contendrá el número a adivinar, si esta es igual entonces se felicita y se rompe el ciclo while para terminar, de lo contrario te agradece y se volverá a pedir un número ya que continúa en ciclo.

```
51      finally:
52          loteria = randint(valormin, valormax)
53          if num == loteria:
54              print("ADIVINASTE, ERES BRUJO")
55              break
56          else:
57              print("Gracias por participar, no adivinaste el número")
```

Ejecución del programa

Cuando todo sale bien

```
Escribe un valor entre 1 y 5: 2
Gracias por participar, no adivinaste el número

Escribe un valor entre 1 y 5: 2
ADIVINASTE, ERES BRUJO
```

Como observamos, escojo un valor válido en el rango, pero al no ser el que pensó la computadora se vuelve a pedir, elijo de nuevo el 2 y está vez adiviné, así que termina el programa; todo sale bien puesto que respeté el flujo del programa

Cuando se comete un error de rango (Menor al rango)

```
Escribe un valor entre 1 y 5: 2
Gracias por participar, no adivinaste el número

Escribe un valor entre 1 y 5: -5
ERROR:  ErrorMinimo: El valor es menor al rango
Gracias por participar, no adivinaste el número

Escribe un valor entre 1 y 5:
```

Como observamos, primero ingreso un valor correcto, al no adivinar me solicita de nuevo un valor y esta vez ingreso uno por debajo del rango, por lo que se llama la excepción personalizada por el raise ErrorMinimo que indica el tipo de error y justifica el mismo, pero en lugar de cerrar o detener el programa, el error se maneja con la excepción y se le solicita al usuario ingresar otro, así lo obliga a hacer un uso correcto del rango sin necesidad de cerrar o detener la ejecución.

Cuando se comete un error de rango (Mayor al rango)

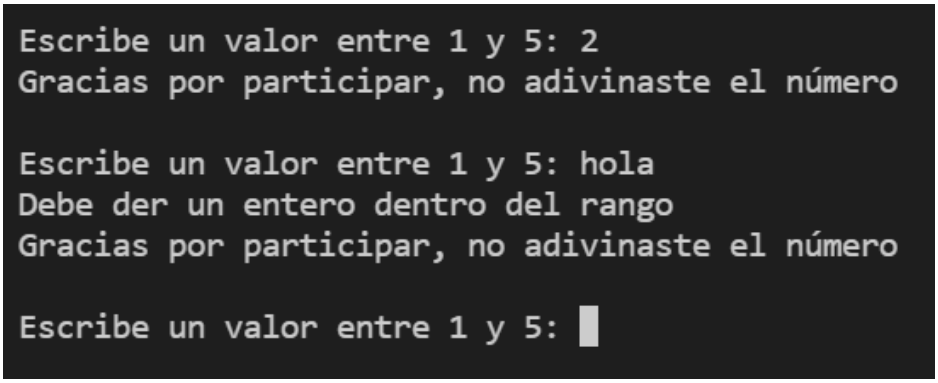
```
Escribe un valor entre 1 y 5: 3
Gracias por participar, no adivinaste el número

Escribe un valor entre 1 y 5: 8
ERROR:  ErrorMaximo: El valor es mayor al rango
Gracias por participar, no adivinaste el número

Escribe un valor entre 1 y 5:
```

Como observamos, primero ingreso un valor correcto, al no adivinar me solicita de nuevo un valor y esta vez ingreso uno por encima del rango, por lo que se llama la excepción personalizada por el raise ErrorMaximo que indica el tipo de error y justifica el mismo, pero en lugar de cerrar o detener el programa, el error se maneja con la excepción y se le solicita al usuario ingresar otro, así lo obliga a hacer un uso correcto del rango sin necesidad de cerrar o detener la ejecución.

Cuando no se ingresa un valor numérico (ValueError)



Como observamos, primero ingreso un valor correcto, al no adivinar me solicita de nuevo un valor y esta vez ingreso una cadena de caracteres, por lo que se llama la excepción ValueError que justifica el error cometido, pero en lugar de cerrar o detener el programa, el error se maneja con la excepción y se le solicita al usuario ingresar otro, así lo obliga a hacer un uso correcto del rango sin necesidad de cerrar o detener la ejecución.

Conclusión

Este primer programa me pareció muy interesante y entretenido, ya que además de poner a prueba nuestros conocimientos adquiridos con la investigación anterior, también aporta a dicho tema acerca del manejo de los errores para hacer los programas más eficientes y menos susceptibles a posibles fallos provocados por el usuario. Sin duda estas dos actividades en conjunto aportan a mis conocimientos sobre el tema y enriquecen mis capacidades para poner en práctica dichos conocimientos, me parecieron excelentes métodos de introducirnos al tema de la computación tolerante a fallos y serán, desde luego, pilares para resolver futuros problemas y retos en el curso.

Bibliografías

- *Manejo de errores con Python—Ayuda*. (s/f). Arcgis.com. Recuperado el 8 de febrero de 2022, de https://desktop.arcgis.com/es/arcmap/10.3/analyze/python/error-handling-with-python.htm
- Ordoñez, J. O. [JohnOrtizOrdoñez]. (2020, marzo 24). *Python Curso V2: 237 Uso de la Cláusula raise para Lanzar Excepciones Personalizadas*. Youtube. https://www.youtube.com/watch?v=EA56BqD-zCU
- *Python Exception Handling Using try, except and finally statement*. (s/f). Programiz.Com. Recuperado el 8 de febrero de 2022, de https://www.programiz.com/python-programming/exception-handling

LINK AL REPOSITORIO: https://github.com/UlisesVallejo/Manejo-de-errores---raise.git