Herramientas para el manejo de errores (par. 2)

Nombre: Diana Fernanda Castillo Rebolledo

Materia: Computación Tolerante a Fallas

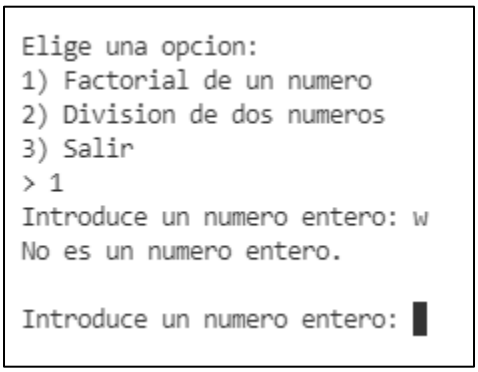
Horario: L-I 11:00 am-1:00 pm

Profesor: Dr. Michel Emanuel López Franco

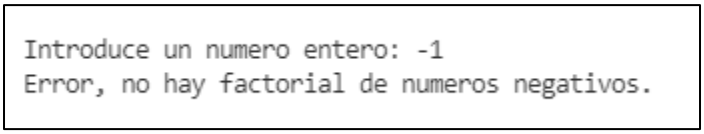
Sección: D06

El ejemplo que desarrollé pone en práctica el try except, la sentencia raise para excepciones y la sentencia assert para realizar comprobaciones, todo esto en el lenguaje de Python. Lo primero que hace es mostrar un menú en el que el usuario tiene dos opciones a escoger, la primera es calcular el factorial de un número, y la otra es la división entre dos números.

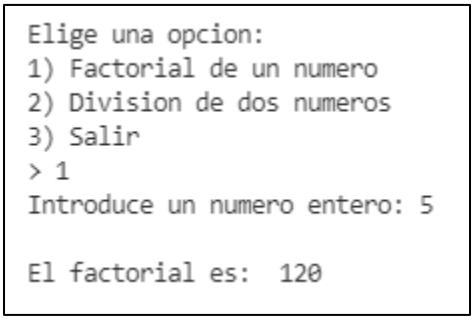
En la primera opción, se le pide un número entero al usuario, y dentro de un try se intenta convertir ese número a un entero, pero si no es posible se muestra un mensaje de error en el except y el usuario tiene que volver a intentarlo.



Si no hay ningún problema, se manda el número a una función que calculará su factorial. Una vez en la función, dentro de un try hay una sentencia assert, la cual comprueba que el número sea mayor o igual a cero, o de lo contrario mostrará un mensaje de error, después se intenta calcular el factorial con un ciclo for, y en el except se trata la excepción AssertionError en caso de que esta se levante, que es una excepción del mismo tipo que la que lanza assert.

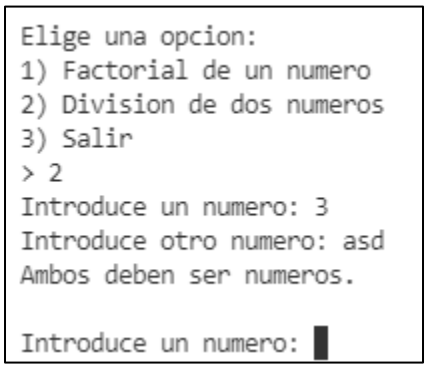


Finalmente, en el else se imprime el resultado del factorial en caso de que no haya habido ningún error.

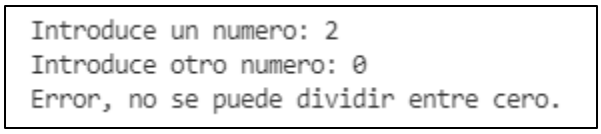


En la segunda opción, se le piden dos números al usuario, en un try se intentan convertir a un número flotante, y en caso de que ambos o alguno de los dos no se pueda convertir, se

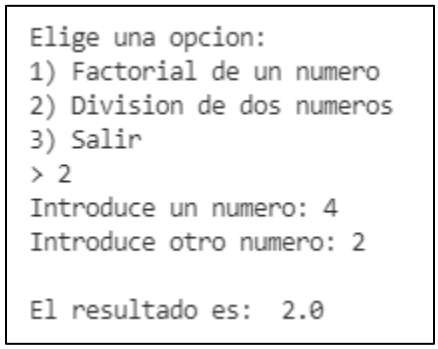
muestra un mensaje de error en el except y el usuario tiene que volver a introducir ambos números.



Si no hubo problemas, se mandan ambos números a una función que dividirá el primero entre el segundo. En la función hay un try en el que hay una condición, si el dividendo es igual a 0, se levanta una excepción de tipo ZeroDivisionError con raise, y se muestra un mensaje de error, después se intenta dividir ambos números, y en el except se trata la excepción ZeroDivisionError en caso de que esta se levante, que es una excepción del mismo tipo que la que lanza raise.



Finalmente, en el else se imprime el resultado de la división en caso de que no haya habido ningún error.



Link al código en GitHub: https://github.com/diana-castillo/Herramientas-para-el-manejo-de-errores.git

Conclusión:

A la hora de programar, se deben tener en cuenta los diversos posibles errores que se pueden presentar en un código, aunque este sea relativamente pequeño, pueden ser errores sencillos como los de sintaxis o semánticos, o los más tardados de manejar como son los errores que se producen en el tiempo de ejecución del programa. Para estos últimos,

en el caso de Python hay herramientas muy útiles, las que utilicé en este ejemplo fueron el try except que existe en la mayoría de los lenguajes, y también encontré otras dos técnicas para lanzar excepciones que son assert y raise, las cuales me parecieron bastante interesantes, pues assert podría funcionar de manera similar a un if, pero assert se utiliza más bien para cazar condiciones que no deberían ocurrir en el programa, pues se considerarían un error de programación. La sentencia raise también es muy interesante, pues ayuda a levantar excepciones que pueden ser personalizadas, lo cual es bastante útil a la hora de controlar diversos errores dependiendo del propósito del programa. Con ayuda de estas sencillas técnicas se puede lograr un programa mucho más robusto.

Referencias:

Castaño Giraldo, S. A. (s. f.). *Manejo de Errores en Python.* Control Automático Educación. Recuperado 4 de febrero de 2022, de https://controlautomaticoeducacion.com/python-desde-cero/manejo-de-errores-en-python/#:%7E:text=Veremos%20que%20para%20manejar%20los,%2DRaise%2DElse%2DFinally