Centro Universitario de Ciencias Exactas e Ingenierías.

Ingeniería en computación.

Alumno: Vargas López David Guadalupe.

Computación tolerante a fallos.

Profesor: López Franco Michel Emanuel.

Sección: D06.

Guadalajara Jal. febrero del 2022.

Vargas López David Guadalupe.

Otras herramientas para el manejar errores

Cómo se ve anteriormente existen muchas técnicas con las que se puede hacer el manejo de errores de una mejor manera, de forma que se eviten a toda costa errores que hagan que el programa deje de funcionar o funcione de manera inesperada; existen diferentes cláusulas como por ejemplo try-except la cual es muy útil al momento de hacer programas y hacer un manejo de errores bastante amplio.

a continuación, se presenta un programa en la que se utilizan algunas de las técnicas que se mencionan en el reporte anterior como lo son la cláusula try-except antes mencionada además de variantes como else, Raise, la cláusula finally, esto con el fin de hacer este programa robusto y mostrar el funcionamiento de cada uno de estas cláusulas para su correcto uso.

```
#programa para graficar datos.

from distutils.log import error
from matplotlib.pyplot import plot, show
from pip import main
import statistics


class error_menor(Exception):
    pass
    def min(minimo):
        return minimo <= 0

def main():
    min=0
    list=[]
    contador=0
```

```
    intentos=int(input("Cuantos datos se van a graficar?  "))
    for intentos in range(intentos):
        while True:
            try:
                numero=float(input("Dame el numero a graficar"))
                if numero > min:
                    list.append(numero)
                    contador += 1
                else:
                    raise error_menor ("El valor ingresado debe ser mayor a 0")
                if contador >= intentos:
                    break
            except error_menor as error:
                print(error)
            finally:
                print("El promedio de los numeros graficados hasta el momento es de:")
                mean = statistics.mean(list)
                print(mean)
    plot(list)
    show()
main()
```

El código anterior consiste en que el usuario introduzca algunos números qué se graficarán posteriormente en una especie de gráfica de picos, en la que estos datos se van guardando en una lista para finalmente una vez que el usuario concluyó la tarea de introducir los datos dentro del programa, este proceder a graficar los y mostrar en pantalla una pequeña ventana con la gráfica creada con los datos introducidos por el usuario.

Cláusula.

A continuación, se verá más a detalle cada una de las partes de que componen el programa, y para qué sirven con el fin de que el usuario pueda entender correctamente el funcionamiento de cada una de las cláusulas antes mencionadas en un ejemplo práctico de su utilización y de esta manera pueda ponerlo en práctica en cualquier proyecto que se le presente.

```
#programa para graficar datos .

from distutils.log import error
from matplotlib.pyplot import plot, show
from pip import main
import statistics


class error_menor(Exception):
    pass
    def min(minimo):
        return minimo <= 0
```

En la parte del código anterior se puede observar las librerías que conforman el programa y que hacen posible la utilización de todos los métodos para que este programa funcione, se utiliza también una librería la cual es la segunda que nos permite realizar la función de gráfica dentro de una ventana en consola, además de que se presenta posteriormente la definición de una clase error, en la cual se define que el usuario no podrá meter valores menores o iguales a cero esto se realizó solo como un ejemplo práctico de cómo realizar un ejemplo personalizado que posteriormente se explicará más a detalle.

```
def main():
    min=0
    list=[]
    contador=0
    intentos=int(input("Cuantos datos se van a graficar?  "))
    for intentos in range(intentos):
```

En esta parte del código lo único que se hace es declaración de variables además de la declaración de una lista vacía, y se le preguntará al usuario cuántos son el total de datos que va AA graficar para de esta manera posteriormente sí claro la entrada de los datos las veces que el usuario lo requiera, además de obtener posteriormente el promedio de todos los datos que introdujo el usuario a lo largo de la corrida del programa.

```
while True:
        try:
            numero=float(input("Dame el numero a graficar"))
            if numero > min:
                list.append(numero)
                contador += 1
            else:
                raise error_menor ("El valor ingresado debe ser mayor a 0")
            if contador >= intentos:
                break
        except error_menor as error:
            print(error)
        finally:
            print("El promedio de los numeros graficados hasta el momento es de:")
            mean = statistics.mean(list)
            print(mean)
    plot(list)
    show()
```

Después se realiza un ciclo infinito el cual sólo se rompe una vez que el usuario ha terminado de introducir todos los datos de su tabla conforme al número de datos que introdujo al principio o en la parte anterior a este código; Para posteriormente realizar el cuerpo del programa, Es decir en donde se realizan todos los procedimientos importantes que éste posee.

```
try:
            numero=float(input("Dame el numero a graficar"))
            if numero > min:
                list.append(numero)
                contador += 1
            else:
                raise error_menor ("El valor ingresado debe ser mayor a 0")
            if contador >= intentos:
                break
        except error_menor as error:
            print(error)
```

En este fragmento de código podemos observar la cláusula tríceps la cual mencionamos al principio del documento que funciona con el fin de hacer robusto el programa y en este caso funciona para evitar que el usuario introduzca letras en el código al momento de la entrada de datos debido a que los datos que se están pidiendo son números, de esta forma el programa volverá a preguntar las veces que sea necesario hasta que el usuario introduzca todos los números conforme al dato recibido anteriormente en la pregunta de cuántos datos se van a graficar.

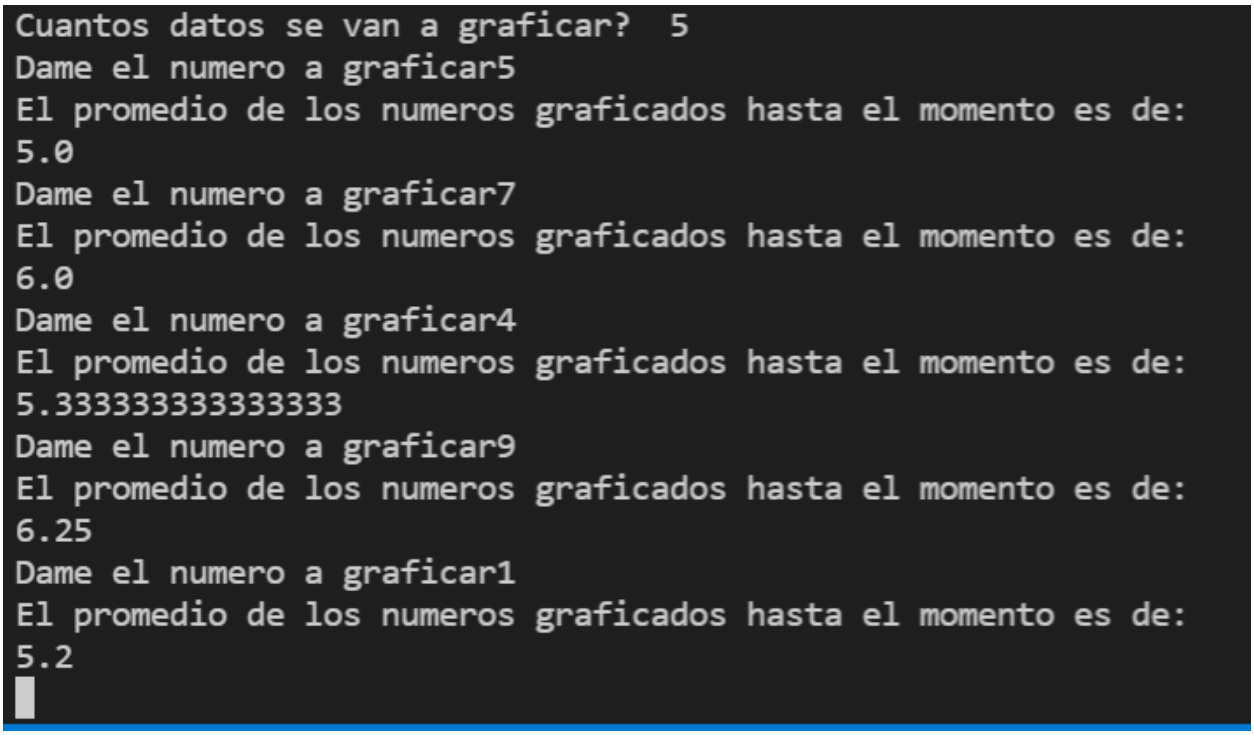
También se muestra el apartado del else, con lo cual su traducción en español sería sí sí no, esto quiere decir que es un condicionante y en caso de que no se realizará una acción anterior en la cláusula y se pasará al apartado ns en donde existirá algún mensaje o acción que se deberá realizar en caso de no cumplirse con el apartado if.

```
 if numero > min:
                list.append(numero)
                contador += 1
            else:
                raise error_menor ("El valor ingresado debe ser mayor a 0")
```

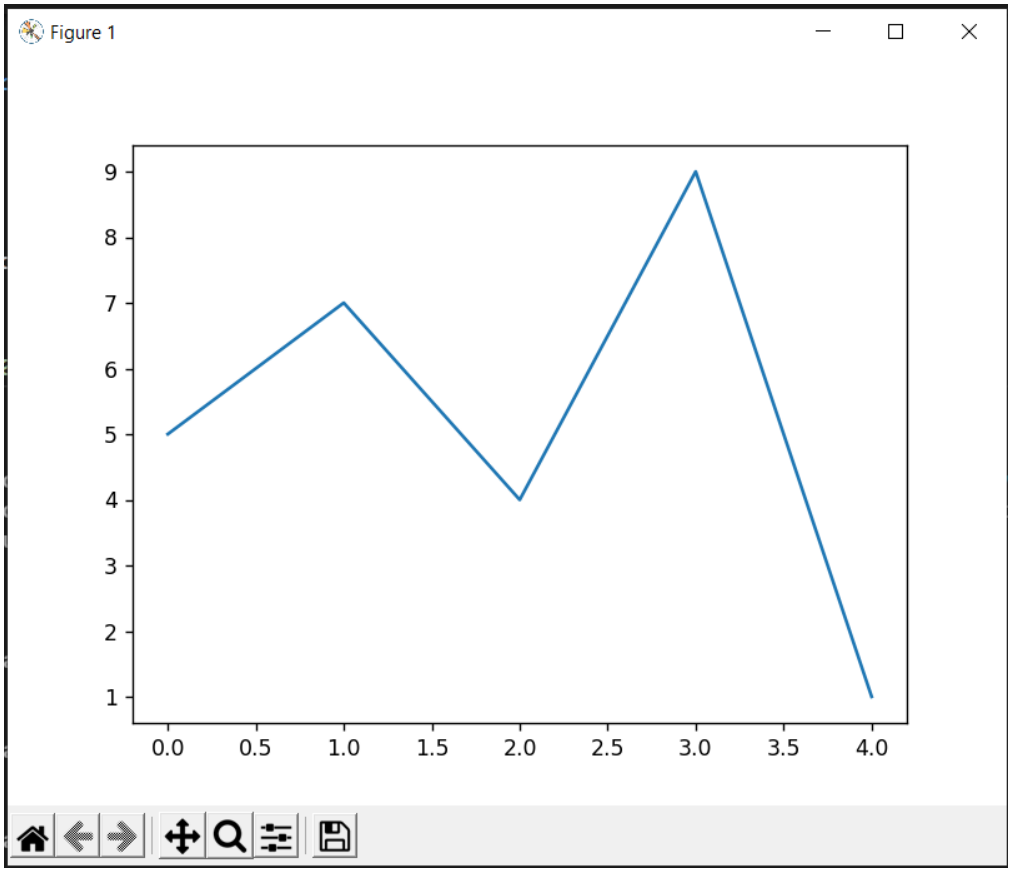
En el apartado de fragmento de código anterior se puede observar la cláusula raise, la cual funciona como un manejo de errores personalizado en el que nosotros podemos observar y gestionar los errores que se vayan dando, en este caso el error que se está presentando aquí es que no puede ingresar valores menores o iguales al valor cero por lo que en caso de que el usuario intente ingresar alguno de estos valores se mostrará el mensaje De El valor ingresado debe ser mayor a cero.

Imágenes de la corrida del programa:

A continuación se presenta una corrida del programa en donde el usuario introduce los datos totalmente correctos; como se ve en la corrida del programa cuando el usuario va ingresando los datos el programa automáticamente le va haciendo un conteo del promedio de todos los números que ha ingresado esto en caso de que el usuario lo requiera, cómo se ve hasta arriba en este caso se introdujeron 5 datos una vez que se terminaron de ingresar los datos el programa automáticamente realiza la gráfica correspondiente conforme a los datos ingresados y de esta manera ésta se muestra a continuación.



Como se menciona anteriormente la gráfica que a continuación se muestra es una gráfica de picos en la que los datos que fueron ingresados por el usuario corresponden al eje de las y y se realizó una gráfica en la que se muestran las coordenadas y se pueden realizar algunas operaciones en este de manera que el usuario podría utilizar esta gráfica para muchas acciones a futuro.



A continuación, se presenta un caso en el que el usuario por accidente o por no saber introduce una letra en este caso la y y el usuario da como resultado un mensaje de dato no válido, posteriormente continúa con la corrida del programa preguntando los datos que se requieren para la gráfica continuando en el dato que se quedó anteriormente.

```
Cuantos datos se van a graficar?  3
Dame el numero a graficar1
El promedio de los numeros graficados hasta el momento es de:
1.0
Dame el numero a graficar6
El promedio de los numeros graficados hasta el momento es de:
3.5
Dame el numero a graficary
dato no valido
El promedio de los numeros graficados hasta el momento es de:
3.5
Dame el numero a graficar4
El promedio de los numeros graficados hasta el momento es de:
3.6666666666666665
```

Y por último se muestra una imagen en donde el usuario intenta ingresar un dato invalido como se dijo anteriormente este programa sólo acepta datos mayores a cero, sólo con fines prácticos debido a que este programa puede utilizarse con datos numéricos reales, pero en este caso el usuario al ingresar el valor de cero el programa automáticamente salta el error y le dice que el valor ingresado debe ser mayor a cero, para posteriormente continuar con la ejecución del programa y al igual que en la cláusula anterior continúa donde se quedó el programa anteriormente.

```
Cuantos datos se van a graficar?  2
Dame el numero a graficar4
El promedio de los numeros graficados hasta el momento es de:
4.0
Dame el numero a graficar0
El valor ingresado debe ser mayor a 0
El promedio de los numeros graficados hasta el momento es de:
4.0
Dame el numero a graficar1
El promedio de los numeros graficados hasta el momento es de:
2.5
```

Conclusiones:

Esta práctica fue bastante interesante hacer porque aunque conocía algunas técnicas de manejo de errores como el tríceps no conocía todas sus cláusulas y qué es lo que tanto podía ser como por ejemplo con la cláusula raise por ejemplo, esta práctica me pareció bastante interesante y muy utilizable en programas futuros a realización; el código fuente de este programa está realizado totalmente en Python ya que es un lenguaje que se presta bastante para realizar cualquier tipo de código o acción que se necesite, en esta ocasión opte por realizar una gráficación de datos pero se pudieron realizar infinidad de acciones.

Cabe destacar que entre más lenguajes de programación se conozcan es más fácil conocer el manejo de errores debido a que entendiendo cómo funciona un manejador de errores nos podemos dar una idea de cómo utilizarlo en diferentes tipos de lenguajes de programación así no esté especificado en dicho lenguaje, por lo que es importante conocerlas todas o la mayoría para tratar de hacer los programas lo más robustos posibles e intolerantes a fallos simples.

Bibliografía:

- *8. Errores y excepciones — documentación de Python - 3.10.2*. (n.d.). BLOG. Retrieved February 8, 2022, from https://docs.python.org/es/3/tutorial/errors.html
- *Manejo de errores con Python—Ayuda | ArcGIS Desktop*. (n.d.). BLOG. Retrieved February 8, 2022, from https://desktop.arcgis.com/es/arcmap/10.3/analyze/python/error-handling-with-python.htm
- C, S. (2021, December 20). *Manejo de Errores en Python*. Control Automático Educación. Retrieved February 8, 2022, from https://controlautomaticoeducacion.com/python-desde-cero/manejo-de-errores-en-python/

Vargas López David Guadalupe.

Link al repositorio de GitHub:

https://github.com/David-1212/actividad1-tolerante-grafica