Actividad 8

Manejo de Errores y Excepciones

TIC

TALENTO TECH

Universidad Tecnológica de Pereira

faceIT

Manejo de Errores y Excepciones

El manejo de excepciones en Python contribuye significativamente a la robustez y estabilidad de un código. razones clave por las cuales el manejo de excepciones puede hacer que tu código sea más robusto:

Contingencia para Entradas Inesperadas:

El manejo de excepciones permite gestionar situaciones inesperadas, como datos de entrada incorrectos o inesperados, sin que el programa se bloquee.

Aumento de la Tolerancia a Errores:

Permite que un programa continúe ejecutándose incluso cuando se encuentre con errores predecibles. Esto puede ser útil en situaciones en las que se prefiere que el programa siga funcionando con una degradación controlada en lugar de detenerse por completo.

Mantenimiento Eficiente:

Facilita la introducción de cambios y actualizaciones en el código, ya que el manejo de excepciones puede aislar áreas específicas que pueden requerir ajustes debido a cambios en los requisitos o en el entorno de ejecución.

Facilita el Diagnóstico y Depuración:

Al capturar excepciones y proporcionar información detallada sobre el error, facilita el proceso de diagnóstico y depuración. Los mensajes de error personalizados pueden indicar claramente qué salió mal y dónde.

Mejora de la Legibilidad:

Al manejar excepciones de manera explícita, el código puede enfocarse en la lógica principal, haciendo que sea más legible y fácil de entender al separar la lógica de manejo de errores del código principal.

Prevención de Interrupciones Inesperadas:

El manejo de excepciones permite anticipar y controlar errores potenciales, evitando que el programa se detenga abruptamente por condiciones inesperadas.

Manejo de Errores y Excepciones

Python maneja varios tipos de errores, que se clasi can comúnmente en tres categorías principales: errores de sintaxis, errores de tiempo de ejecución y errores de lógica. Aquí hay una descripción de cada categoría:

Errores de Sintaxis:
Descripción: Ocurren cuando el intérprete de Python encuentra un código que viola las reglas de sintaxis del lenguaje. Ejemplo:

```
if x == 5
    print("Hola")
```

En este caso, falta el : después de la condición if, lo que generará un error de sintaxis.

Manejo de Errores y Excepciones

Errores de Tiempo de Ejecución:
Descripción: Ocurren durante la ejecución del programa debido a condiciones imprevistas o incorrectas.

Ejemplos:

En estos ejemplos, se intenta dividir por cero, convertir una cadena no numérica a un entero y acceder a un índice fuera de los límites de una lista, lo que generará errores de tiempo de ejecución.

```
# ZeroDivisionError
resultado = 10 / 0

# ValueError
numero = int("abc")

# IndexError
lista = [1, 2, 3]
elemento = lista[5]
```

Manejo de Errores y Excepciones

Errores de Lógica (Excepciones):

Descripción: Ocurren cuando el código tiene una estructura o lógica incorrecta, pero no genera errores de sintaxis ni de tiempo de ejecución. Se manejan mediante excepciones.
Ejemplo:

```
edad = int(input("Ingrese su edad: "))
if edad < 0:
    raise ValueError("La edad no puede ser un número negativo.")
```

En este caso, el programa verifica si la edad ingresada es negativa y lanza una excepción ValueError si lo es.

Manejo de Errores y Excepciones

Errores de Archivo (FileNotFoundError, IOError, etc.):
Descripción: Relacionados con la manipulación de archivos. Pueden ocurrir al intentar abrir, leer o escribir archivos y directorios.
Ejemplo:

```
with open("archivo_no_existente.txt", "r") as archivo:
    contenido = archivo.read()
```

Si el archivo "archivo_no_existente.txt" no existe, se generará un error del tipo FileNotFoundError.

+ info

Estos son solo algunos ejemplos y Python maneja muchos otros tipos de errores. La capacidad de manejar estas excepciones permite que los programas en Python sean más robustos y resistentes a situaciones inesperadas. Puedes utilizar bloques try, except, else y nally para gestionar estas excepciones y proporcionar un comportamiento adecuado en caso de errores.

Manejo de Errores y Excepciones

El manejo de errores y excepciones en Python permite controlar situaciones excepcionales o errores que pueden ocurrir durante la ejecución de un programa. Esto ayuda a que el programa no se detenga abruptamente y a proporcionar información significativa sobre los errores que se producen. La estructura básica incluye los bloques try, except, else, y finally.

Explicación detallada y algunos ejemplos:

Estructura Básica:

```
try:
    # Bloque de código que podría generar una excepción
    # ...
except TipoDeExcepcion as variable:
    # Bloque de código a ejecutar si se produce la excepción
    # ...
else:
    # Bloque de código a ejecutar si no se produce ninguna excepción
    # ...
finally:
    # Bloque de código a ejecutar siempre, independientemente de si se produce una excepción o no
    # ...
```

Manejo de Errores y Excepciones

Ejemplo Básico:

```python
try:
    numero = int(input("Ingrese un número: "))
    resultado = 10 / numero
    print("El resultado es:", resultado)
except ZeroDivisionError as e:
    print("Error: División por cero no permitida.")
except ValueError as e:
    print("Error: Ingrese un número válido.")
else:
    print("Operación exitosa.")
finally:
    print("Fin del programa.")
```

+ info

En este ejemplo, el programa intenta dividir 10 por el número ingresado por el usuario. Si el usuario ingresa un cero, se produce un ZeroDivisionError; si ingresa algo que no es un número, se produce un ValueError. Dependiendo del tipo de error, se ejecuta el bloque except correspondiente. El bloque else se ejecuta si no se produce ninguna excepción, y el bloque finally se ejecuta siempre, independientemente de si se produjo una excepción o no.

Manejo de Errores y Excepciones

Manejo de Excepciones Genéricas:
python

```python
try:
    resultado = 10 / 0
except Exception as e:
    print("Se produjo una excepción:", str(e))
```

+ info

Usar Exception capturará cualquier tipo de excepción, pero se debe tener precaución al hacerlo, ya que puede ocultar errores que deberían ser manejados de manera diferente.

Manejo de Errores y Excepciones

Lanzar Excepciones:

```python
def dividir(a, b):
    if b == 0:
        raise ValueError("No se puede dividir por cero.")
    return a / b

try:
    resultado = dividir(10, 0)
except ValueError as e:
    print("Error:", str(e))
```

En este ejemplo, la función dividir lanza explícitamente una excepción (ValueError) si se intenta dividir por cero. El bloque except maneja esta excepción específica.

El manejo de errores y excepciones en Python es esencial para escribir código robusto y manejar situaciones inesperadas de manera elegante.

Ejercicios

Ejercicios relacionados con el manejo de errores y excepciones en Python. Cada ejercicio presenta un escenario específico que puede generar un error, y se te pide que utilices el manejo de excepciones para manejar esos errores de manera adecuada.

Ejercicios relacionados con el manejo de errores y excepciones en Python. Cada ejercicio presenta un escenario específico que puede generar un error, y se te pide que utilices el manejo de excepciones para manejar esos errores de manera adecuada.

Ejercicio : División Segura
Modifica el siguiente código para manejar el error de división por cero.
Si el usuario ingresa 0 como divisor, imprime un mensaje indicando que la división por cero no está permitida.

```
try:
    numerador = int(input("Ingrese el numerador: "))
    divisor = int(input("Ingrese el divisor: "))

    resultado = numerador / divisor

    print("El resultado de la división es:", resultado)

except ZeroDivisionError as e:
    print("Error: División por cero no permitida.")
except ValueError as e:
    print("Error: Ingrese números válidos.")
```

Ejercicios

Ejercicio: Conversión de Números
Completa el siguiente código para manejar el error de conversión de número.
Si el usuario ingresa algo que no es un número, imprime un mensaje indicando que debe ingresar un número válido.

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

except ValueError as e:
    print("Error: Ingrese un número válido.")
```

Ejercicios

Ejercicio : Índice Fuera de Límites

Modifica el siguiente código para manejar el error de índice fuera de límites.
Si el usuario ingresa un índice que no existe en la lista, imprime un mensaje indicando que el índice es inválido.

```
lista = [10, 20, 30, 40, 50]

try:
    indice = int(input("Ingrese un índice para acceder a la lista: "))
    elemento = lista[indice]
    print("El elemento en el índice {} es: {}".format(indice, elemento))

except IndexError as e:
    print("Error: Índice fuera de límites. Ingrese un índice válido.")
except ValueError as e:
    print("Error: Ingrese un número válido como índice.")
```

Ejercicios

Ejercicio: Manejo de Archivos

Completa el siguiente código para manejar el error de archivo no encontrado.
Si el archivo no existe, imprime un mensaje indicando que el archivo no fue encontrado.

```
try:
    with open("archivo_existente.txt", "r") as archivo:
        contenido = archivo.read()
    print("Contenido del archivo:", contenido)

except FileNotFoundError as e:
    print("Error: El archivo no fue encontrado.")
except IOError as e:
    print("Error: Ocurrió un error al leer el archivo.")
```

+ info

Estos ejercicios te ayudarán a practicar el manejo de errores y excepciones en Python y a desarrollar habilidades para escribir código más robusto y resistente.

TIC

TALENTO TECH

Universidad Tecnológica de Pereira

faceIT