

Guía Teórica N°03

Manejo de Excepciones

Es turno de conocer los nuevos temas a tratar durante el curso, a partir de este bloque en adelante, aprenderemos nuevos términos que nos ayudarán a crear las bases suficientes y prepararnos para el ámbito laboral como programadores en Python. Y para ello, uno de los temas muy importantes a considerar es el manejo de excepciones al escribir nuestro código, así que empecemos esta pequeña aventura generando errores en nuestro código.

Excepciones:

Una excepción es un evento que ocurre durante la ejecución de un programa que interrumpe el flujo normal de las instrucciones del programa. En general, cuando una secuencia de comandos de Python se encuentra con una situación que no puede afrontar, genera una excepción. Una excepción es un objeto de Python que representa un error.

Cuando una secuencia de comandos de Python genera una excepción, debe manejar la excepción inmediatamente, de lo contrario, termina y se cierra.

Manejando Excepciones:

Si tiene algún código sospechoso que pueda generar una excepción, puede defender su programa colocando el código sospechoso en un **try**. Después del bloque **try**, incluya una declaración **except**, seguida de un bloque de código que maneje el problema de la manera más elegante posible.

Veamos un ejemplo manejando excepciones en Python.

```
try:
    # Código Sospechoso
    a = int(input('Ingrese un número: '))
    b = int(input('Ingrese otro número: '))
    division = a / b
    print(f'El resultado es: {division}')

except:
    # Manejo de la excepción
    print('No se puede dividir entre cero')


else:
    # Código a ejecutar si todo resulta bien
    print('El programa finalizó bien :)' )
```

Aquí hay algunos puntos muy importantes a considerar sobre la sintaxis de manejo de excepciones mencionada anteriormente:

- Una sola declaración **try** puede tener varias declaraciones **except**. Esto es útil cuando el bloque **try** contiene declaraciones que pueden generar diferentes tipos de excepciones.
- También puede proporcionar una cláusula **except** genérica, que maneja cualquier excepción.
- Después de la (s) cláusula (s) **except**, puede incluir una cláusula **else**. El código del bloque **else** se ejecuta si el código del bloque **try** no genera una excepción.
- El bloque **else** es un buen lugar para el código que no necesita la protección de **try**.

Múltiples Excepciones:

Podemos manejar diferentes excepciones haciendo uso de múltiples sentencias **except** o podemos definir un único **except** el cual haga un manejo de diversas excepciones que puedan llegar a suceder dentro del bloque código que se encuentra en el **try**.

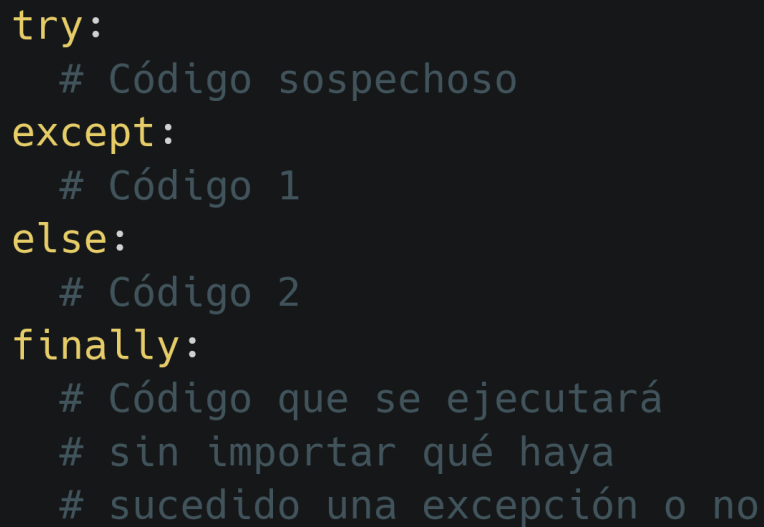


```
# ---Ejemplo #01---
try:
    # Código Sospechoso
except Excepcion1, Excepcion2, Excepcion3:
    # Código para tres (3) excepciones

# ---Ejemplo #02---
try:
    # Código Sospechoso
except Excepcion1:
    # Código 1
except Excepcion2:
    # Código 2
except Excepcion3:
    # Código 3
```

Sentencia finally:

Podemos agregar una última sentencia a nuestro manejador de excepciones el cual ejecutará un bloque de código sin importar que haya sucedido o no una excepción.



```
try:
    # Código sospechoso
except:
    # Código 1
else:
    # Código 2
finally:
    # Código que se ejecutará
    # sin importar qué haya
    # sucedido una excepción o no
```

Excepciones Incorporadas:

Python incluye una gran lista de excepciones a considerar cuando queremos manejar cualquier situación problemática que pueda llegar a suceder durante la ejecución de nuestro código.

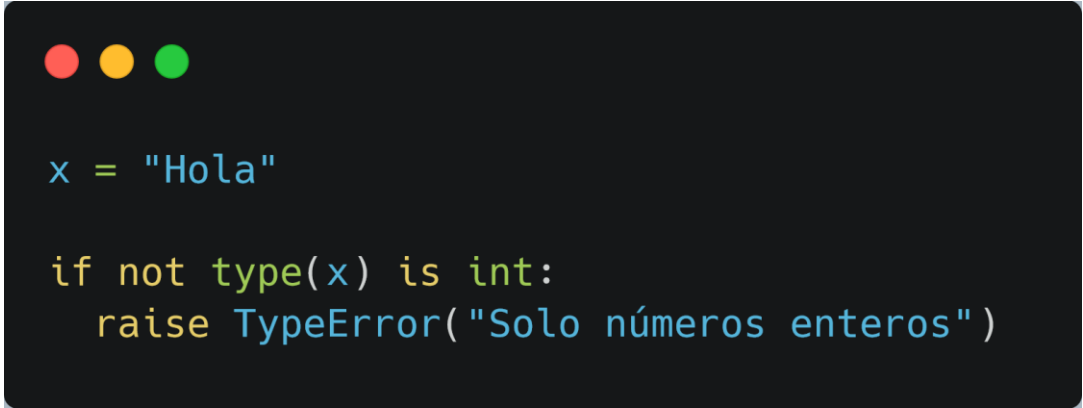
Se invita a que visiten el siguiente enlace para conocer cada una de las excepciones que Python incorpora por defecto.

Enlace: <https://docs.python.org/3/library/exceptions.html>

Plantear una Excepción:

Python nos ofrece la instrucción **raise** la cual tiene el objetivo de provocar una excepción de manera intencional en nuestro código con fines de testing y así conocer el comportamiento de nuestro código en base a un posible error que pueda llegar a suceder durante la ejecución de nuestro programa.

Veamos un caso sencillo donde hagamos uso de esta instrucción.



```
x = "Hola"

if not type(x) is int:
    raise TypeError("Solo números enteros")
```

¡Nos vemos en la próxima clase!

De esta manera finalizamos el bloque de manejo de excepciones en Python el cual nos ha dado las bases para poder controlar la ejecución de nuestro código en dado caso de que ocurra un posible error. Esto nos ayuda a crear un código más robusto en caso de errores y así controlar todos los posibles escenarios al momento de desarrollar nuestros proyectos.