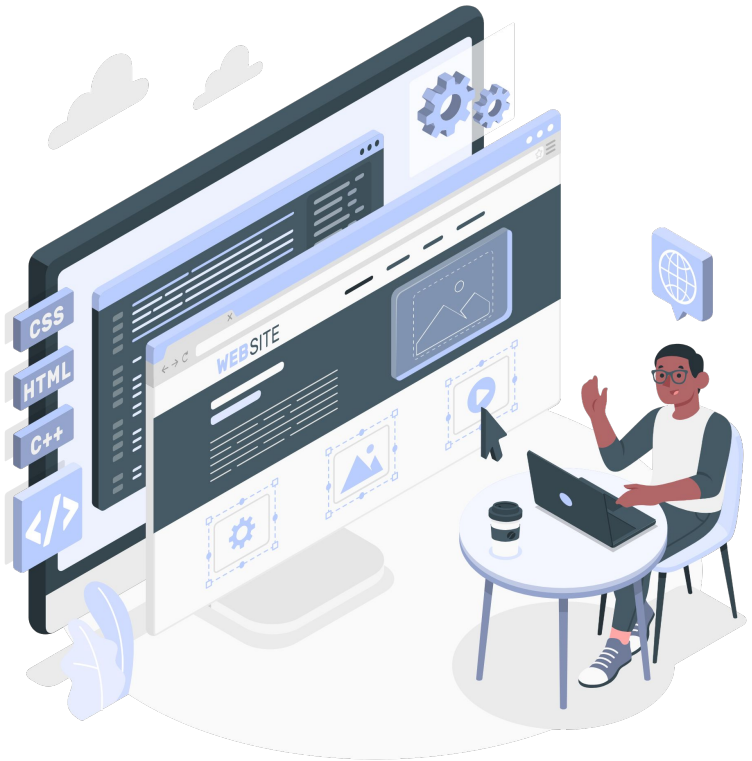

PROTECO

Manejo de Excepciones en Python

Try, except, else, finally, raise

Temario

PROTECO

01 try, except

02 else

03 finally

04 raise

¿Qué es una excepción?

PROTECO

Las excepciones son aquellos errores o problemas que suceden en un código durante tiempo de ejecución.

Estos errores no necesariamente son fatales, y pueden ser tratados por el programador. Tenemos errores de sintaxis, lógicos y las excepciones.

Este concepto no es exclusivo de Python y se presenta en otros lenguajes, como Java.

Error de sintaxis vs Excepción

```
>>> for i in range(2):print(i)

0
1
>>>
```

Error de sintaxis

```
>>> for  in range(2): print(i)
SyntaxError: invalid syntax
>>>
```

Excepción

```
>>> print(255/0)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    print(255/0)
ZeroDivisionError: division by zero
```

Ejemplos

- Siempre que ocurre un error en tiempo de ejecución, se crea una **excepción**, Normalmente el programa se **detiene** y Python presenta un mensaje de error.

- Por ejemplo, la división por cero crea una excepción:
```
>>> print 255/0
ZeroDivisionError: integer division or modulo
```

- Un elemento no existente en una lista:
```
>>> a = []
>>> print a[2]
IndexError: list index out of range
```

- Acceso a una clave no definida en un diccionario:
```
>>> d = {}
>>> print d[´indice´]
KeyError: indice
```

Manejo de excepciones

- En ocasiones es necesario realizar alguna operación que podría provocar una excepción, pero no queremos que se detenga el programa. Para ello, podemos manejar la excepción usando las sentencias **try** y **except**.

```
try:
    #Se ejecuta algún código
except:
    #Ejecuta este código sí hubo
    #un error en el bloque try
```

- NameError
- IndexError
- TypeError
- valueError
- ImportError/ModuleNotFound

Documentación:

https://docs.python.org/3/library/exceptions.html

Manejo de excepciones

- try: Aquí se coloca código que esperamos que pueda lanzar algún error.
- Except: En este bloque se maneja el error, si ocurre un error dentro del bloque de código try, se deja de ejecutar el código del try y se ejecuta lo que se haya definido en el bloque except.

```
try:
    #Se ejecuta algún código
except:
    #Ejecuta este código sí hubo
    #un error en el bloque try
```

1 y 2

Cláusula else

- En Python, también puede usar la cláusula else en el bloque try-except que debe estar presente después de todas las cláusulas except. El código ingresa al bloque else solamente si la cláusula try no generó excepción.

```
try:
    #Se ejecuta algún código
except:
    #Ejecuta este código sí hubo
    #un error en el bloque try
else:
    #Ejecuta este código si
    #no hubo excepción
```

Cláusula finally

- Python proporciona la palabra clave finally, que siempre se ejecuta después de try y except.
- Este código siempre se ejecuta después de los otros bloques, incluso si hubo una excepción no detectada o una declaración de retorno en uno de los otros bloques.

```
try:
    #Se ejecuta algún código
except:
    #Ejecuta este código sí hubo
    #un error en el bloque try
else:
    #Ejecuta este código si
    #no hubo excepción
finally:
    #Algún otro código... (siempre se ejecuta)
```

Cláusula raise

- Podemos usar raise para lanzar una excepción si ocurre una condición. La declaración se puede complementar con una excepción personalizada.

Usa raise para forzar una excepción


```
raise Exception('Mensaje') #Sintaxis

raise ValueError('Mensaje')

raise TypeError('Mensaje')

raise NameError('Mensaje')
```

"Cuando te enfrentes a la ambigüedad,
no caigas en la tentación de adivinar.“

—Guido Van Rossum

¡Muchas Gracias!