# **Excepciones.**

Para entender que es una excepción primero debemos saber que es un evento, un evento es básicamente cualquier cosa que ocurra dentro de nuestro programa, en lenguaje de programación lo que hacemos es capturar estos eventos para actuar en consecuencia, por ejemplo si tenemos esto:

nombre = input("Ingresa tu nombre: ")

print(nombre)

Esto es un evento, cuando el usuario lo teclea el programa lo detecta y muestra lo que introdujo el usuario.

Las excepciones tambien son eventos que ocurren dentro de la ejecución de nuestro programa, pero este evento lo que hace es interrumpir el flujo de ejecución de nuestro programa, es decir, hace que se detenga toda acción en el programa.

Debemos saber actuar en consecuencia contra las excepciones para evitar que nuestro programa se detenga.

Vamos a tener esta función en donde forzaremos una excepción:

*#Creamos funcion suma*

*#Forzaremos una excepcion*

def sumar\_dos():

    a = input("Numero 1: ")

    b = input("Numero 2: ")

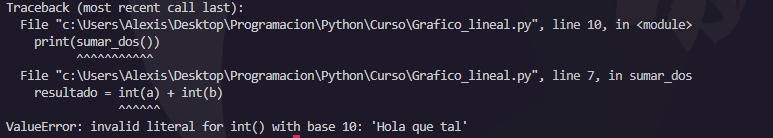
    resultado = int(a) + int(b)

    return resultado

print(sumar\_dos())



Al ejecutar pondremos un str en vez de un int, y al momento de seguir con el programa nos tirara una excepción:



Es nos esta diciendo que es un literal invalido para un entero, aquí lo que pide es un numero de base 10, o sea los números normales que conocemos, 12345678910.etc. Pero le estamos pasando “Hola que tal” y obviamente esto no esta dentro de los números de base 10. EL TIPO DE EXCEPCION ES VALUEERROR.

Empezaremos por usar la sentencia “try:” que es una sentencia que nos dice, que lo está debajo de esta misma inténtalo, o sea si sabe que tirara una excepción inténtalo, pero esta sentencia va acompañada de otra mas que es el “except:”, que en caso de que nos de una excepción hagamos otra cosa.

*#Creamos funcion suma*

*#Forzaremos una excepcion*

def sumar\_dos():

    a = input("Numero 1: ")

    b = input("Numero 2: ")

    try:

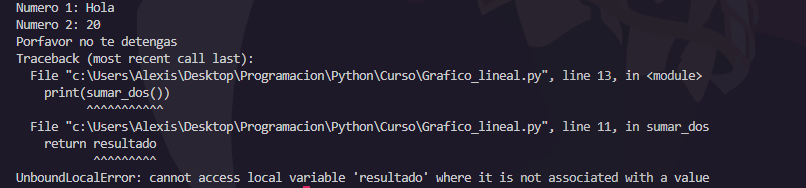
        resultado = int(a) + int(b)

    except:

        print("Porfavor no te detengas")

    return resultado

print(sumar\_dos())



Ahora nos dice que no se puede acceder a la variable local “resultado”, porque no tiene ningún valor asociado, pero lo que haremos es que hasta que noi tengamos el valor de resultado, no lo vamos a retornar. Para esto podemos meter todo dentro de un bucle.

*#Creamos funcion suma*

*#Forzaremos una excepcion*

def sumar\_dos():

    while True:

        a = input("Numero 1: ")

        b = input("Numero 2: ")

        try:

            resultado = int(a) + int(b)

            break

        except:

            print("Ingresa un dato valido")

    return resultado

print(sumar\_dos())

De esta manera lo que hacemos es que dentro del bucle While, le pedimos el numero 1 y 2, intentara sumar esos dos números, y en caso de que de una excepción va a pedir al usuario que ingrese un dato valido, y si se logra sumar va a romper el bucle.



Y aquí nos deshacemos de la excepción y evitamos que nuestro codigo se detenga.

El posicionamiento del break no es el mejor para trabajar, quedaría mejor asi.

def sumar\_dos():

    while True:

        a = input("Numero 1: ")

        b = input("Numero 2: ")

        try:

            resultado = int(a) + int(b)

        except:

            print("Ingresa un dato valido")

        else:

            break

    return resultado

print(sumar\_dos())

Pero despues temenos una clausula extra, que es el “finally” que no importa si hay excepciones o no siempre se va a ejecutar.

def sumar\_dos():

    while True:

        a = input("Numero 1: ")

        b = input("Numero 2: ")

        try:

            resultado = int(a) + int(b)

        except:

            print("Ingresa un dato valido")

        else:

            break

        finally:

            print("Esto se ejecuta siempre")

    return resultado

print(sumar\_dos())



Es una sentencia muy extraña de uso.

def sumar\_dos():

    while True:

        a = input("Numero 1: ")

        b = input("Numero 2: ")

        try:

            resultado = int(a) + int(b)

        except Exception as e:

            print("Ingresa un dato valido")

            print(f"ERROR: {e}")

        else:

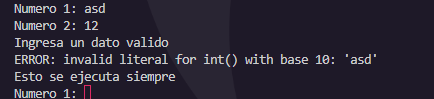
            break

        finally:

            print("Esto se ejecuta siempre")

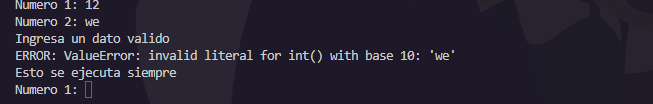
    return resultado

print(sumar\_dos())



Con el “except Exception” podemos acceder a todas las excepciones, es decir, Exception es un objeto de la clase excepción que es el padre de todas las excepciones, porque todas las excepciones son derivados de la clase Exception o son de una subclase de esta misma. Incluso podemos mostrar nuestra misma excepción como vemos en este ejemplo. Y para saber el nombre de la excepción podríamos hacer esto:

            print(f"ERROR: {type(e).**\_\_name\_\_**}: {e}")



Podemos usar varias excepciones.

Inclusive podemos crear nuestras propias excepciones, trabajaremos con P.O.O:

class Miexcepcion(Exception):

    def \_\_init\_\_(self, err):

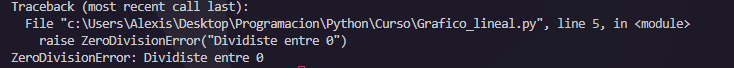
        print(f"El error es {err}")

Aquí creamos ya nuestra clase Miexcepcion

raise ZeroDivisionError

Con la sentencia “raise” podemos llamar a cualquier excepción.

raise ZeroDivisionError("Dividiste entre 0")



Y de esta manera juntando todo podemos ver nuestra excepción

class Miexcepcion(Exception):

    def \_\_init\_\_(self, err):

        print(f"El error es {err}")

try:

    raise Miexcepcion("Mi error")

except:

    print("Esta fue mi excepcion")

raise Miexcepcion("Mi error")

