Manejo de Excepciones

September 6, 2023

[ ]:
```
def mi_funcion():
    # Error de sintaxis
    print("Hola, "mundo!")

mi_funcion()
```

```
  Cell In[1], line 3
    print("Hola, "mundo!")
                       ^
SyntaxError: unterminated string literal (detected at line 3)
```

[ ]:
```
def mi_funcion():
    # Error de ejecución
    1 / 0

mi_funcion()
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[2], line 5
      1 def mi_funcion():
      2     # Error de ejecución
      3     1 / 0
----> 5 mi_funcion()

Cell In[2], line 3, in mi_funcion()
      1 def mi_funcion():
      2     # Error de ejecución
----> 3     1 / 0

ZeroDivisionError: division by zero
```

[ ]:
```
def mi_funcion(numero):
    # Error de entrada
```

```
    print(numero / 0)

mi_funcion(int(input("Introduce un número: ")))
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[2], line 5
      1 def mi_funcion(numero):
      2     # Error de entrada
      3     print(numero / 0)
----> 5 mi_funcion(int(input("Introduce un número: ")))

Cell In[2], line 3, in mi_funcion(numero)
      1 def mi_funcion(numero):
      2     # Error de entrada
----> 3     print(numero / 0)

ZeroDivisionError: division by zero
```

[ ]:
```
def mi_funcion():
    try:
        # Código que podría generar una excepción
        1 / 0
    except ZeroDivisionError:
        # Código para gestionar la excepción
        print("No se puede dividir por cero.")

mi_funcion()
```

```
No se puede dividir por cero.
```

[ ]:
```
def mi_funcion():
    try:
        # Código que podría generar una excepción
        1 / 0
    except ZeroDivisionError:
        # Código para gestionar la excepción
        print("No se puede dividir por cero.")
    finally:
        # Bloque `finally`
        print("Se ha ejecutado el bloque `finally`.")

mi_funcion()
```

```
No se puede dividir por cero.
Se ha ejecutado el bloque `finally`.
```

1 ERRORES

1.1 Errores de Sintaxis:

[ ]:
```
print("Hola"
```

```
  Cell In[6], line 1
    print("Hola"
                ^
SyntaxError: incomplete input
```

1.2 Errores Semánticos:

[ ]:
```
import os # libreria con funciones del sistema operativo.

print("Trabajando con una LIsta:")

lista = [1 ,2, 3]
print(lista)

#os.system("Pause")
print("Vaciando Lista:")
lista.pop()
lista.pop()
lista.pop()
print(lista)

print("Utilizando una vez mas la opción borrar elemento:")
lista.pop()
print(lista)
```

```
Trabajando con una LIsta:
[1, 2, 3]
Vaciando Lista:
[]
Utilizando una vez mas la opción borrar elemento:
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[2], line 16
     13 print(lista)
     15 print("Utilizando una vez mas la opción borrar elemento:")
---> 16 lista.pop()
     17 print(lista)
```

```
IndexError: pop from empty list
```

[ ]:
```python
# Colocar el código propenso de errores en un try:
errores = 0
calculos = 0

while True:
    try:
        n = int(input("Ingresar un número: "))
        print("El número ", n, " elevado al mismo número: ", n**n)

        calculos += 1

    except:
        print("Al parecer no se introdujo correctamente el número.")
        errores += 1
    #except Exception as e:
    #    raise e
    else:
        print("El programa se ejecutó correctamente.")

        #print("¿Desea volver a calcular? S/N", end="")
        res = input("¿Desea volver a calcular? S/N")

        if res[0] == "N" or res[0] == "n":
            break

    finally:
        print(f"\nerrores: {errores}  cálculos: {calculos}.")
```

```
El número  3  elevado al mismo número:  27
El programa se ejecutó correctamente.

errores: 0  cálculos: 1.
Al parecer no se introdujo correctamente el número.

errores: 1  cálculos: 1.
El número  3  elevado al mismo número:  27
El programa se ejecutó correctamente.

errores: 1  cálculos: 2.
El número  2  elevado al mismo número:  4
El programa se ejecutó correctamente.

errores: 1  cálculos: 3.
```

1.3 Excepciones Múltiples:

[ ]:
```python
import random
while True:
    try:
        num1 = random.randint(0, 10)

        #num2 = input("Número: ")
        num2 = float(input("Número: "))

        print(f"{num1} / {num2} = {num1/num2}")

    except TypeError as e:
        print("Error en el tipo de dato. [error: ", e, "]")

    except ValueError as e:
        print("Valor incorrecto para la operación. [error: ", e, "]")

    except ZeroDivisionError as e:
        print("No se puede dividir entre cero. [error: ", e, "]")

    except Exception as e:
        print("Ha ocurrido de tipo: ", type(e).__name__ ,"[error: ", e, "]")

    else:
        print("El programa finalizó correctamente.")
        break
```

```
Error en el tipo de dato. [error:  unsupported operand type(s) for /: 'int' and
'str' ]
Error en el tipo de dato. [error:  unsupported operand type(s) for /: 'int' and
'str' ]
Error en el tipo de dato. [error:  unsupported operand type(s) for /: 'int' and
'str' ]
Error en el tipo de dato. [error:  unsupported operand type(s) for /: 'int' and
'str' ]
Error en el tipo de dato. [error:  unsupported operand type(s) for /: 'int' and
'str' ]
Error en el tipo de dato. [error:  unsupported operand type(s) for /: 'int' and
'str' ]
Error en el tipo de dato. [error:  unsupported operand type(s) for /: 'int' and
'str' ]
Error en el tipo de dato. [error:  unsupported operand type(s) for /: 'int' and
'str' ]
Error en el tipo de dato. [error:  unsupported operand type(s) for /: 'int' and
'str' ]
```

1.4 Invocando tipo de error:

[ ]:
```
try:
    n = None

    if n is None:
        raise ValueError("Error! No se permite un valor nulo.")

except ValueError as e:
    print(e, "(desde la excepción)")
```

```
Error! No se permite un valor nulo. (desde la excepción)
```