Contenido creado por: Isabel Maniega

-1- Excepciones: en la web de Python

In [ ]:
```
# https://docs.python.org/3/library/exceptions.html
```

-2- Ejemplo básico try-except

In [ ]:
```
# Imagina que tenemos 2 variables
```

In [1]:
```
x = 5
# 'y' no la tenemos definida
```

In [2]:
```
print(x)
```

```
5
```

In [ ]:
```
# print(y)
# NameError: name 'y' is not defined
# OBVIAMENTE DA ERROR, AL NO TENERLA DEFINIDA
```

In [3]:
```
try:
    print(x)
except:
    print("No tenemos definida la variable:")
```

```
5
```

In [4]:
```
try:
    print(y)
except:
    print("No tenemos definida la variable:")
```

```
No tenemos definida la variable:
```

In [ ]:
```
# De esta forma podemos conseguir que un código funcione saltando un erro

# Pero OJO! en donde colocamos este Try-Except. Porque si es algo crítico

# Solo sirve cuando es algo que necesitamos saltar,

# (para que el código ejecute en un momento que sabemos que algo no va)
```

-3- Forma básica de crear una excepción

-3.1- En la División por cero

In [5]:
```
a = 5
b = 0
# a/b
# Si descomentamos "a/b" nos sale:
# ZeroDivisionError: division by zero

# No es posible dividir un número por cero. (Daria infinito)
```

Podemos hacer lo siguiente, sin excepciones

In [7]:
```
def funcion_dividir_1(a, b):
    if b != 0:
        print(a / b)
    else:    # b = 0  -> no puede dividir
        print("el denominador es 0, no podemos dividir")
```

In [8]:
```
funcion_dividir_1(2,3)
```

```
0.6666666666666666
```

In [9]:
```
funcion_dividir_1(2,0)
```

```
el denominador es 0, no podemos dividir
```

el mismo ejercicio cambiando el operador

In [10]:
```
def funcion_dividir_1(a, b):
    if b == 0:
        print("el denominador es 0, no podemos dividir")
    else:      # b es distinto de 0
        print(a/b)
```

In [11]:
```
funcion_dividir_1(2,3)
```

```
0.6666666666666666
```

In [12]:
```
funcion_dividir_1(2,0)
```

```
el denominador es 0, no podemos dividir
```

-4- Uso de raise

In [ ]:
```
# Podemos lanzar excepciones, no lo usaremos
```

-5- Try-Except-Else

In [13]:
```
# me creo una función para comprobar más casos.
```

```
def funcion_division(a,b):
    try:
        division = a / b
        print('estamos en try y hemos calculado a/b')
    except ZeroDivisionError:
        print("Un número dividido por 0 sale infinito")
        print("No pongas un 0 en el deniminador!")
    else:
        print('estamos en el else')
        print('valor de la división:', division)
```

In [14]:
```
funcion_division(1,0)
```

```
Un número dividido por 0 sale infinito
No pongas un 0 en el deniminador!
```

In [15]:
```
funcion_division(1,2)
```

```
estamos en try y hemos calculado a/b
estamos en el else
valor de la división: 0.5
```

-6- try-except-else con archivos

Un archivo que no existe

(o no se encuentra en ese lugar)

In [16]:
```
try:
    f = open('archivo_excepciones.txt')  # El fichero no existe
except FileNotFoundError:
    print('¡El fichero no existe!')
else:
    print(f.read())
```

```
¡El fichero no existe!
```

In [ ]:
```
# lo cerramos, si esta abierto
# f.close()
```

Un archivo que SI existe

(y lo encuentra en esa ubicación)

In [17]:
```
try:
    f = open('archivo_excepciones_2.txt')  # El fichero si existe
except FileNotFoundError:
    print('¡El fichero no existe!')
else:
    print(f.read())
    f.close()
```

```
¡ Hola Mundo!
```

-7- Errores cuando sumamos strings en vez de números

In [18]:
```
# print(2+"2")
# TypeError: unsupported operand type(s) for +: 'int' and 'str'

# al hacer esa operación nos devuelve un error
```

In [19]:
```
3
```

Out[19]: 3

In [20]:
```
type(3)
```

Out[20]: int

In [21]:
```
str(3)
```

Out[21]: '3'

In [22]:
```
'3'
```

Out[22]: '3'

In [23]:
```
type('3')
```

Out[23]: str

In [24]:
```
def funcion_formatos_diferentes(a,b):
    try:
        suma = a + b
        print(suma)
    except TypeError:
        print("revisa el formato de los números, porque no es correcto")
```

In [25]:
```
funcion_formatos_diferentes(2,"3")
```

```
revisa el formato de los números, porque no es correcto
```

In [26]:
```
funcion_formatos_diferentes(2,3)
```

```
5
```

-8- except Exception

Otra forma si no sabes que excepción puede saltar, puedes usar la clase genérica Exception. Sirve para cualquier tipo de excepción. De hecho todas las excepciones heredan de Exception

except Exception: Ejemplo 1

In [27]:
```
def funcion_suma_2(a,b):
    try:
```

```
        suma = a + b
        print("la suma es: ", suma)
    except Exception:
        print("Ha habido una excepción")
```

In [28]:
```
funcion_suma_2(2,0)
```

```
la suma es:  2
```

In [29]:
```
funcion_suma_2(2,"2")
```

```
Ha habido una excepción
```

except Exception: Ejemplo 2

In [30]:
```
def funcion_division_3(a,b):
    try:
        division = a / b
        print("la division es: ", division)
    except Exception:
        print("Ha habido una excepción")
```

In [31]:
```
funcion_division_3(1,3)
```

```
la division es:  0.3333333333333333
```

In [32]:
```
funcion_division_3(2,0)
```

```
Ha habido una excepción
```

In [33]:
```
funcion_division_3(2,"2")
```

```
Ha habido una excepción
```

-9- except Exception as e (una de las mejores opciones)

In [34]:
```
def funcion_division_4(a,b):
    try:
        division = a / b
        print("la division es: ", division)
    except Exception as e:
        print("Ha habido una excepción")
        print("tipo del error: ", type(e))
        print('str(e):', str(e))
```

In [35]:
```
funcion_division_4(1,3)
```

```
la division es:  0.3333333333333333
```

In [36]:
```
funcion_division_4(2,0)
```

```
Ha habido una excepción
tipo del error:  <class 'ZeroDivisionError'>
str(e): division by zero
```

In [37]:
```
funcion_division_4(2,"2")
```

```
Ha habido una excepción
tipo del error:  <class 'TypeError'>
str(e): unsupported operand type(s) for /: 'int' and 'str'
```

-10- except Exception as e (otra posibilidad: try-except-else)

In [38]:
```
def funcion_division_5(a,b):
    try:
        division = a / b
        print("la division es: ", division)
    except Exception as e:
        print("Ha habido una excepción")
        print("tipo del error: ", type(e))
    else:
        print("estamos en else, no hubo excepciones")
```

In [39]:
```
funcion_division_5(1,3)
```

```
la division es:  0.3333333333333333
estamos en else, no hubo excepciones
```

In [40]:
```
funcion_division_5(2,0)
```

```
Ha habido una excepción
tipo del error:  <class 'ZeroDivisionError'>
```

In [41]:
```
funcion_division_5(2,"2")
```

```
Ha habido una excepción
tipo del error:  <class 'TypeError'>
```

-11- except Exception as e (otra posibilidad: try-except-finally)

Este bloque se suele usar si queremos ejecutar algún tipo de acción de limpieza.

Si por ejemplo estamos escribiendo datos en un fichero pero ocurre una excepción,

tal vez queramos borrar el contenido que hemos escrito con anterioridad,

para no dejar datos inconsistenes en el fichero.

In [45]:
```
def funcion_division_6(a,b):
    try:
        division = a / b
        print("la division es: ", division)
        print("\n")
    except Exception as e:
        print("Ha habido una excepción")
        print("tipo del error: ", type(e))
```

```
        print("\n")
    finally:
        print("estamos en finally")
        print("esto se ejecuta SIEMPRE haya o no excepciones")
```

In [43]:
```
funcion_division_6(1,3)
```

```
la division es:  0.3333333333333333


estamos en finally
esto se ejecuta SIEMPRE haya o no excepciones
```

In [44]:
```
funcion_division_6(2,0)
```

```
Ha habido una excepción
tipo del error:  <class 'ZeroDivisionError'>


estamos en finally
esto se ejecuta SIEMPRE haya o no excepciones
```

In [46]:
```
funcion_division_6(2,"2")
```

```
Ha habido una excepción
tipo del error:  <class 'TypeError'>


estamos en finally
esto se ejecuta SIEMPRE haya o no excepciones
```

-12- Ejemplo de excepciones con archivos

In [47]:
```
def funcion_lectura(archivo):
    try:
        with open(archivo) as file:
            lectura_archivo = file.read()
    except Exception as e:
        print("no se pudo abrir")
        print("Tipo de error:", type(e))
        print(str(e))
```

In [48]:
```
funcion_lectura('archivo_excepciones_1.txt')  # no lo encuentra
```

```
no se pudo abrir
Tipo de error: <class 'FileNotFoundError'>
[Errno 2] No such file or directory: 'archivo_excepciones_1.txt'
```

In [49]:
```
funcion_lectura('archivo_excepciones_2.txt') # si lo encuentra

# (si lo coloco yo previamente este archivo)
# SE ENCUENTRA EN LA MISMA RUTA
```

Más sobre Excepciones...

Ordenadas las excepciones por orden de preferencia:

In [ ]:

```
"""
BaseException
├── BaseExceptionGroup
├── GeneratorExit
├── KeyboardInterrupt
├── SystemExit
└── Exception
    ├── ArithmeticError
    │    ├── FloatingPointError
    │    ├── OverflowError
    │    └── ZeroDivisionError
    ├── AssertionError
    ├── AttributeError
    ├── BufferError
    ├── EOFError
    ├── ExceptionGroup [BaseExceptionGroup]
    ├── ImportError
    │    └── ModuleNotFoundError
    ├── LookupError
    │    ├── IndexError
    │    └── KeyError
    ├── MemoryError
    ├── NameError
    │    └── UnboundLocalError
    ├── OSError
    │    ├── BlockingIOError
    │    ├── ChildProcessError
    │    ├── ConnectionError
    │    │    ├── BrokenPipeError
    │    │    ├── ConnectionAbortedError
    │    │    ├── ConnectionRefusedError
    │    │    └── ConnectionResetError
    │    ├── FileExistsError
    │    ├── FileNotFoundError
    │    ├── InterruptedError
    │    ├── IsADirectoryError
    │    ├── NotADirectoryError
    │    ├── PermissionError
    │    ├── ProcessLookupError
    │    └── TimeoutError
    ├── ReferenceError
    ├── RuntimeError
    │    ├── NotImplementedError
    │    └── RecursionError
    ├── StopAsyncIteration
    ├── StopIteration
    ├── SyntaxError
    │    └── IndentationError
    │         └── TabError
    ├── SystemError
    ├── TypeError
    ├── ValueError
    │    └── UnicodeError
```

```
   │          ├── UnicodeDecodeError
   │          ├── UnicodeEncodeError
   │          └── UnicodeTranslateError
   └── Warning
        ├── BytesWarning
        ├── DeprecationWarning
        ├── EncodingWarning
        ├── FutureWarning
        ├── ImportWarning
        ├── PendingDeprecationWarning
        ├── ResourceWarning
        ├── RuntimeWarning
        ├── SyntaxWarning
        ├── UnicodeWarning
        └── UserWarning
"""
```

Tipos de excepciones más relevantes

Para capturar cualquier excepción podemos usar Exception o BaseException:

In [1]:
```
try:
    30* (2/0)
except BaseException as e:
    print('Error %s' % str(e))
```

```
Error division by zero
```

In [2]:
```
try:
    30* (2/0)
except Exception as e:
    print('Error %s' % str(e))
```

```
Error division by zero
```

ArithmeticError

- Division entre 0: **ZeroDivisionError**

In [ ]:
```
30 * (2/0)
```

In [3]:
```
try:
    30* (2/0)
except ZeroDivisionError as e:
    print('Error %s' % str(e))
```

```
Error division by zero
```

AttributeError

- Error en el uso: **AttributeError**

In [4]:
```
num= 10
num.append(6)
```

```
print(num)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[4], line 2
      1 num= 10
----> 2 num.append(6)
      3 print(num)

AttributeError: 'int' object has no attribute 'append'
```

In [5]:
```
try:
    num= 10
    num.append(6)
    print(num)
except AttributeError as e:
    print('Error %s' % str(e))
```

```
Error 'int' object has no attribute 'append'
```

ImportError

- Error al importar un módulo: **ImportError**

In [6]:
```
from pandas import hola
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[6], line 1
----> 1 from pandas import hola

ImportError: cannot import name 'hola' from 'pandas' (/home/isabelmaniega/Documentos/PCEP/env/lib/python3.8/site-packages/pandas/__init__.py)
```

In [7]:
```
try:
    from pandas import hola
except ImportError as e:
    print('Error %s' % str(e))
```

```
Error cannot import name 'hola' from 'pandas' (/home/isabelmaniega/Documentos/PCEP/env/lib/python3.8/site-packages/pandas/__init__.py)
```

Error en importar un modulo será: **ModuleNotFoundError**

In [8]:
```
try:
    import hola
except ModuleNotFoundError as e:
    print('Error %s' % str(e))
```

```
Error No module named 'hola'
```

LookupError

- Error de índice: **IndexError**

In [9]:
```
L = [10, 50, 60]
L[3]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[9], line 2
      1 L = [10, 50, 60]
----> 2 L[3]

IndexError: list index out of range
```

In [10]:
```
try:
    L = [10, 50, 60]
    L[3]
except IndexError as e:
    print('Error %s' % str(e))
```

```
Error list index out of range
```

Si en vez de poner un número entero ponemos un string el error sería de tipo:

In [11]:
```
try:
    L = [10, 50, 60]
    L['3']
except IndexError as e:
    print('Error Index %s' % str(e))
except TypeError as e:
    print('Error TypeError %s' % str(e))
```

```
Error TypeError list indices must be integers or slices, not str
```

- Error de clave en un diccionario: **KeyError**

In [12]:
```
ages = {'Juan': 25, 'Luis':36, 'Pedro':41}
ages['Maria']
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Cell In[12], line 2
      1 ages = {'Juan': 25, 'Luis':36, 'Pedro':41}
----> 2 ages['Maria']

KeyError: 'Maria'
```

In [13]:
```
try:
    ages = {'Juan': 25, 'Luis':36, 'Pedro':41}
    ages['Maria']
except KeyError as e:
    print('Error %s' % str(e))
```

```
Error 'Maria'
```

NameError

- Nombre no definido: **NameError**

In [14]:
```
4 + x*3
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 4 + x*3

NameError: name 'x' is not defined
```

In [ ]:
```
try:
    4 + x*3
except NameError as e:
    print('Error %s' % str(e))
```

OSError

- Archivo no encontrado: **FileNotFoundError**

In [15]:
```
try:
    file = open('data.csv')
except FileNotFoundError as e:
    print('Error %s' % str(e))
```

```
Error [Errno 2] No such file or directory: 'data.csv'
```

- Archivo no encontrado: **NotADirectoryError**

In [16]:
```
try:
    # crear una carpeta vacia llamada "solucion" al lado del archivo Exce
    file = open('./solucion')
except IsADirectoryError as e:
    print('Error %s' % str(e))
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[16], line 3
      1 try:
      2     # crear una carpeta vacia llamada "solucion" al lado del archivo Excepciones.ipynb
----> 3     file = open('./solucion')
      4 except IsADirectoryError as e:
      5     print('Error %s' % str(e))

File ~/Documentos/PCEP/env/lib/python3.8/site-packages/IPython/core/interactiveshell.py:284, in _modified_open(file, *args, **kwargs)
    277 if file in {0, 1, 2}:
    278     raise ValueError(
    279         f"IPython won't let you open fd={file} by default "
    280         "as it is likely to crash IPython. If you know what you are doing, "
    281         "you can use builtins' open."
    282     )
--> 284 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: './solucion'
```

SyntaxError

- Error en la identación o sintaxis: **SyntaxError**

IndentationError

In [17]:
```
name = 'Pepe'

if name == 'Pepe':
print('El nombre es Pepe')
```

```
  Cell In[17], line 4
    print('El nombre es Pepe')
    ^
IndentationError: expected an indented block
```

In [18]:
```
name = 'Pepe'

try:
    if name == 'Pepe':
    print('El nombre es Pepe')
except IndentationError as e:
    print('Error %s' % str(e))
```

```
  Cell In[18], line 5
    print('El nombre es Pepe')
    ^
IndentationError: expected an indented block
```

Este no se puede capturar, solo si se realiza con dos scripts y se importa uno en otro, podremos realizar la excepción

In [ ]:
```
# test1.py
try:
    import test2
except IndentationError as ex:
    print(ex)

# test2.py
def f():
    pass
        pass # error
```

SyntaxError

Por ejemplo si definimos mal un string, lista, etc se nos olvida el cierre

In [19]:
```
name = 'Pepe
```

```
  Cell In[19], line 1
    name = 'Pepe
                ^
SyntaxError: EOL while scanning string literal
```

In [20]:
```
try:
    name = 'Pepe
except SyntaxError as e:
    print('Error %s' str(e))
```

```
  Cell In[20], line 2
    name = 'Pepe
                ^
SyntaxError: EOL while scanning string literal
```

Pasa lo mismo que con la identación, no se puede capturar.

TypeError

- Error de tipo de variable: **TypeError**

In [22]:
```
'4' + 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[22], line 1
----> 1 '4' + 2

TypeError: can only concatenate str (not "int") to str
```

In [21]:
```
try:
    '4' + 2
except TypeError as e:
    print('Error %s' % str(e))
```

```
Error can only concatenate str (not "int") to str
```

ValueError

- Error al recibir un error de tipo o de valor inapropiado: **ValueError**

In [23]:
```
import math

x = -3

print(f'Square Root of {x} is {math.sqrt(x)}')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[23], line 5
      1 import math
      3 x = -3
----> 5 print(f'Square Root of {x} is {math.sqrt(x)}')

ValueError: math domain error
```

In [24]:
```
x = -3

try:
    print(f'Square Root of {x} is {math.sqrt(x)}')
except ValueError as ve:
    print(f'You entered {x}, which is not a positive number.')
    print('Error %s' % str(ve))
```

```
You entered -3, which is not a positive number.
Error math domain error
```

Múltiples excepciones

En el caso de declarar multiples excepciones se tomarán por orden de preferencia según la primera tabla al ser detectados, para declararlos se pone except y entre paréntesis las excepciones:

In [25]:
```
try:
    # Error al pulsar enter sin insertar dato o insertar un texto, error
    value = input('Inserte un número: ')
    result = 25/int(value)
    print(f'El resultado es: {result}')
    print(f'Square Root of {x} is {math.sqrt(int(value))}')
    L = [10, 5, 6]
    print(f'El valor en la lista es: {L[value]}')
except (IndexError, TypeError, ValueError) as e:
    print('Error %s' % str(e))
```

```
Error invalid literal for int() with base 10: ''
```

In [28]:
```
try:
    # Error al poner un número. el index es un string
    value = input('Inserte un número: ')
    result = 25/int(value)
    print(f'El resultado es: {result}')
```

```
    print(f'Square Root of {x} is {math.sqrt(int(value))}')
    L = [10, 5, 6]
    print(f'El valor en la lista es: {L[value]}')
except (IndexError, TypeError, ValueError) as e:
    print('Error %s' % str(e))
```

```
El resultado es: 4.166666666666667
Square Root of -3 is 2.449489742783178
Error list indices must be integers or slices, not str
```

In [27]:
```
try:
    # Error al poner un número negativo
    value = input('Inserte un número: ')
    result = 25/int(value)
    print(f'El resultado es: {result}')
    print(f'Square Root of {x} is {math.sqrt(int(value))}')
    L = [10, 5, 6]
    print(f'El valor en la lista es: {L[value]}')
except (IndexError, TypeError, ValueError) as e:
    print('Error %s' % str(e))
```

```
El resultado es: -12.5
Error math domain error
```

Raise

También se puede usar raise directamente con las excepciones:

In [29]:
```
# Error al poner un número un número negativo
value = input('Inserte un número: ')

if not type(value) is int:
    raise TypeError('Error en el index')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call las
t)
Cell In[29], line 5
      2 value = input('Inserte un número: ')
      4 if not type(value) is int:
----> 5     raise TypeError('Error en el index')

TypeError: Error en el index
```

-14- EJERCICIOS

Ejemplo con try except

(el primero es el de examen)

In [ ]:
```
"""
try:
    print(5/0)
    break
```

```
except:
    print("Sorry, something went wrong...")
except (ValueError, ZeroDivisionError):
    print("Too bad...")
"""

# SyntaxError: 'break' outside loop
```

In [ ]:
```
# ejemplo 2 de este tipo (este si funciona)

# se ha intentado que ejecute la parte de ZeroDivisionError
```

In [30]:
```
try:
    print(5/0)
except (ValueError, ZeroDivisionError):
    print("Too bad...")
except:
    print("Sorry, something went wrong...")
```

```
Too bad...
```

In [31]:
```
try:
    print(5/0)
except (ValueError):
    print("Too bad...")
except:
    print("Sorry, something went wrong...")
```

```
Sorry, something went wrong...
```

In [32]:
```
try:
    print(5/0)
except ValueError:
    print("Too bad...")
except:
    print("Sorry, something went wrong...")
```

```
Sorry, something went wrong...
```

UNA POSIBILIDAD

In [33]:
```
try:
    print(5/0)
except Exception as e:
    print(type(e))
    print(str(e))
```

```
<class 'ZeroDivisionError'>
division by zero
```

In [ ]:
```
"""try:
    print(5/0)
except:
    print("Sorry, something went wrong...")
except (ValueError, ZeroDivisionError):
    print("Too bad...")"""

# SyntaxError: 'break' outside loop
```

Gracias por la atención

Isabel Maniega