Unió Europea
Fons social europeu

Tratamiento de Errores

Aprendiendo del error

A programar se aprende programando y cometer errores es la prueba de que estás avanzando, no te preocupes.

En esta unidad veremos:

- La identificación de errores
- Su gestión con excepciones

Errores Sintácticos

- Falta de comillas o paréntesis o : , etc.

```
In [1]: print("Hola"
          File "<ipython-input-1-8bc9f5174855>", line 1
            print("Hola"
                        ^
        SyntaxError: unexpected EOF while parsing
```

- Llamada a comandos o funciones inexistentes

```
pint("Hola")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-155163d628c2> in <module>()
----> 1 pint("Hola")

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

- Índices inexistentes en listas o pilas vacías, etc.

```
In [4]: l.pop()
        l.pop()
        l.pop()

Out[4]: 1

In [5]: l

Out[5]: []

In [7]: l.pop()
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-7-9e6f3717293a> in <module>()
----> 1 l.pop()

IndexError: pop from empty list
```

- Errores en los tipos de datos

```
In [20]: n = float(input("Introduce un número: "))
         m = 4
         print("{}/{}={}".format(n,m,n/m))

Introduce un número: aaa

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-20-c0e7fd4a26a9> in <module>()
----> 1 n = float(input("Introduce un número: "))
      2 m = 4
      3 print("{}/{}={}".format(n,m,n/m))

ValueError: could not convert string to float: 'aaa
```

Excepciones

- La instrucción TRY permite abrir un bloque y capturar las excepciones para mostrar un error controlado.

In [2]:
```
try:
    n = float(input("Introduce un número: "))
    m = 4
    print("{}/{}={}".format(n,m,n/m))
except:
    print("Ha ocurrido un error, introduce bien el número")
```

```
Introduce un número: aaa
Ha ocurrido un error, introduce bien el número
```

Podemos crear un bucle para repetir la petición del dato hasta que se introduzca un valor correcto

In [3]:
```
while(True):
    try:
        n = float(input("Introduce un número: "))
        m = 4
        print("{}/{}={}".format(n,m,n/m))
        break  # Importante romper la iteración si todo ha salido bien
    except:
        print("Ha ocurrido un error, introduce bien el número")
```

```
Introduce un número: aaa
Ha ocurrido un error, introduce bien el número
Introduce un número: sdsdsd
Ha ocurrido un error, introduce bien el número
Introduce un número: sdsdsd
Ha ocurrido un error, introduce bien el número
Introduce un número: sdsd
Ha ocurrido un error, introduce bien el número
Introduce un número: 10
10.0/4=2.5
```

In [5]:
```
while(True):
    try:
        n = float(input("Introduce un número: "))
        m = 4
        print("{}/{}={}".format(n,m,n/m))
    except:
        print("Ha ocurrido un error, introduce bien el número")
    else:
        print("Todo ha funcionado correctamente")
        break  # Importante romper la iteración si todo ha salido bien
    finally:
        print("Fin de la iteración")
```

```
Introduce un número: aaa
Ha ocurrido un error, introduce bien el número
Fin de la iteración
Introduce un número: 10
10.0/4=2.5
Todo ha funcionado correctamente
Fin de la iteración
```

Otro ejemplo

```
#-------------------------------------------
# definicion de funciones
#-------------------------------------------

def area_triangulo (base, altura):
 return  (base * altura / 2)

#-------------------------------------------
# Proceso principal
#-------------------------------------------

print("CALCULO DEL AREA DEL TRIANGULO----------")
salir = False
#-------------------------------------------
while salir == False :
   try:
        a=(input("Base (x= salir):"))
        if a !=  "x" :
          a=float (a)
          b=float (input("Altura:"))
          area_triangulo(a, b)
          print(f"La media de {a} y {b} es :
                   area {area_triangulo(a, b)}")
        else :
          salir = True
   except:
        print("** Error: Cáculo no realizado **")
        print("Revise los datos de entrada")
#
```

Excepciones múltiples

```
try:
    nombref = "dat\\lista-negra.dat"
    f = open(nombref,"r")
    texto = f.read()
    print (texto)

except IOError:
    print("No existe el archivo : ", nombref)

except Exception as e :
    print (e)
else :
   print("Contenido leido correctamente")
   f.close()
```

Es posible realizar acciones diferentes en función del tipo de excepción producida.

[2]:
```
try:
   n= input("Introduce un número: ")
   5/n

except TypeError :
    print("No se puede dividir el número por una cadena.")
except Exception as e:
   print( type(e).__name__)
```

```
Introduce un número: 9
No se puede dividir el número por una cadena.
```

[3]:
```
try:
   n= float(input("Introduce un número: "))
   5/n

except TypeError :
    print("No se puede dividir el número por una cadena.")
except Exception as e:
   print( type(e).__name__)
```

```
Introduce un número: 5
```

[5]:
```
try:
   n= float(input("Introduce un número: "))
   5/n

except TypeError :
    print("No se puede dividir el número por una cadena.")
except ZeroDivisionError :
    print("No se puede dividir por cero, introduzca otro número.")
except Exception as e:
   print( type(e).__name__)
```

```
Introduce un número: 0
No se puede dividir por cero, introduzca otro número.
```

Práctica P01

https://docs.hektorprofe.net/python/errores-y-excepciones/ejercicios/

Ejercicio 1

Localiza el error en el siguiente bloque de código. Crea una excepción para evitar que el programa se bloquee y además explica en un mensaje al usuario la causa y/o solución:

```
resultado = 10/0
```

Ejercicio 2

Localiza el error en el siguiente bloque de código. Crea una excepción para evitar que el programa se bloquee y además explica en un mensaje al usuario la causa y/o solución:

```
lista = [1, 2, 3, 4, 5]
lista[10]
```

Ejercicio 3

Localiza el error en el siguiente bloque de código. Crea una excepción para evitar que el programa se bloquee y además explica en un mensaje al usuario la causa y/o solución:

```
colores = { 'rojo':'red', 'verde':'green', 'negro':'black' }
colores['blanco']
```

Ejercicio 4

Localiza el error en el siguiente bloque de código. Crea una excepción para evitar que el programa se bloquee y además explica en un mensaje al usuario la causa y/o solución:

```
resultado = 15 + "20"
```

Invocación de excepciones raise

[9]:
```
def mi_funcion (cadena=None) :
    if cadena is None :
        print ("Error! : No se permite un valor nulo ")
    else :
        print("Hola ", cadena)
#-------- Main

mi_funcion("rosa")
```

```
Hola  rosa
```

[10]:
```
mi_funcion()
```

```
Error! : No se permite un valor nulo
```

La instrucción Raise, eleva una excepción

[36]:
```
def mi_funcion (cadena=None) :
    try :
        if cadena is None :
            raise ValueError
    except ValueError:
        print ("Exception! : No se permite un valor nulo ")
    except Exception as e:
       print( type(e).__name__)
#-------- Main

mi_funcion()
```

```
Exception! : No se permite un valor nulo
```

Ejercicio 5

Realiza una función llamada **agregar_una_vez(lista, el)** que reciba una lista y un elemento. La función debe añadir el elemento al final de la lista con la condición de no repetir ningún elemento. Además si este elemento ya se encuentra en la lista se debe invocar un error de tipo *ValueError* que debes capturar y mostrar este mensaje en su lugar:

```
Error: Imposible añadir elementos duplicados => [elemento].
```

Cuando tengas la función intenta añadir los siguiente valores a la lista **10**, **-2**, "**Hola**" y luego muestra su contenido.

> **Sugerencia**
>
> Puedes utilizar la sintaxis "elemento in lista"

```
elementos = [1, 5, -2]
```

Práctica P02

- El siguiente programa lee un fichero.

```
try:
    nombref = "dat\\lista-negra.dat"
    f = open(nombref,"r")
    texto = f.read()
    print (texto)
except IOError:
    print("No existe el archivo : ", nombref)
except Exception as e :
    print (e)
else :
   print("Contenido leido correctamente")
   f.close()
```

1. Cuando el fichero no existe, el error está controlado?
2. Modifícalo para que cree el archivo vacío, si no existe.
3. Verifica que los mensajes son claros para un usuario.