**ERRORES Y EXCEPCIONES**

Una excepción o un error de ejecución se produce durante la ejecución del programa. Las excepciones se puede manejar para que no termine el programa. Veamos algunos ejemplos de excepciones:

```
>>> 4/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero

>>> a+4
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined

>>> "2"+2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

Hemos obtenido varias excpciones: ZeroDivisionError, NameError y TypeError. Puedes ver la lista de excepciones y su significado.

# Manejando excepciones. try, except, else, finally

Veamos un ejemplo simple como podemos tratar una excepción:

```
#!/usr/bin/env python
isNumber = False
while not isNumber:
    try:
        x = int(input("Introduce un número:"))
        isNumber = True
    except ValueError:
        print ("Debes introducir un número")

print(x)
```

1. Se ejecuta el bloque de instrucciones de `try`.

2. Si no se produce la excepción, el bloque de `except` no se ejecuta y continúa la ejecución secuencia.

3. Si se produce una excepción, el resto del bloque `try` no se ejecuta, si la excepción que se ha produce corresponde con la indicada en `except` se salta a ejecutar el bloque de instrucciones `except`.

4. Si la excepción producida no se corresponde con las indicadas en `except` se pasa a otra instrucción `try`, si finalmente no hay un manejador nos dará un error y el programa terminará.

Un bloque `except` puede manejar varios tipos de excepciones:

```
... except (RuntimeError, TypeError, NameError):
...     pass
```

Si quiero controlar varios tipos de excepciones puedo poner varios bloques `except`. Teniendo en cuenta que en el último, si quiero no indico el tipo de excepción:

```
>>> try:
...   print (10/int(cad))
... except ValueError:
...   print("No se puede converir a entero")
... except ZeroDivisionError:
...   print("No se puede divir por cero")
... except:
...   print("Otro error")
```

Por último indicar que podemos indicar una clausula `finally` para indicar un bloque de instrucciones que siempre se debe ejecutar, independientemente de la excepción se haya producido o no.

```
>>> try:
...     result = x / y
... except ZeroDivisionError:
...     print("División por cero!")
... else:
...     print("El resultado es", result)
... finally:
...     print("Terminamos el programa")
```

# Obteniendo información de las excepciones

```
>>> cad = "a"
>>> try:
...   i = int(cad)
... except ValueError as error:
...   print(type(error))
...   print(error.args)
...   print(error)
...
<class 'ValueError'>
("invalid literal for int() with base 10: 'a'",)
invalid literal for int() with base 10: 'a'
```

# Propagando excepciones. raise

Si construimos una función donde se maneje una excepción podemos hacer que la excepción se envía a la función desde la que la hemos llamado. Para ello utilizamos la instrucción `raise`. Veamos algunos ejemplos:

```
def dividir(x,y):
    try:
        return x/y
    except ZeroDivisionError:
        raise
```

Con `raise` también podemos propagar una excepción en concreto:

```
def nivel(numero):
    if numero<0:
        raise ValueError("El número debe ser positivo:"+str(numero))
    else:
        return numero
```

**EJERCICIOS PROPUESTOS DE EXCEPCIONES.**

**EJERCICIO 1**

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].
```

**EJERCICIO 2**

Crea una calculadora en Python, en la que solicites al usuario los operandos y el operador por pantalla y realices las operaciones pertinentes.

El menú se debe visualizar hasta que el usuario teclee “S” como operación a realizar.

Debes controlar todas las excepciones posibles.