Excepciones

Las excepciones son bloques de código que nos permiten continuar con la ejecución de un programa pese a que ocurra un error.

Siguiendo con el ejemplo de la lección anterior, teníamos el caso en que leíamos un número por teclado, pero el usuario no introducía un número:

```
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-14-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'
```

Bloques try - except

Para prevenir el fallo debemos poner el código propenso a errores en un bloque **try** y luego encadenar un bloque **except** para tratar la situación excepcional mostrando que ha ocurrido un fallo:

```
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
```

Como vemos esta forma nos permite controlar situaciones excepcionales que generalmente darían error y en su lugar mostrar un mensaje o ejecutar una pieza de código alternativo.

Podemos aprovechar las excepciones para forzar al usuario a introducir un número haciendo uso de un bucle *while*, repitiendo la lectura por teclado hasta que lo haga bien y entonces romper el bucle con un *break*:

```
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
```

Bloque else

Es posible encadenar un bloque *else* después del *except* para comprobar el caso en que **todo funcione correctamente** (no se ejecuta la excepción).

El bloque *else* es un buen momento para romper la iteración con *break* si todo funciona correctamente:

```
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
```

```
Introduce un número: 10
10.0/4 = 2.5
Todo ha funcionado correctamente
```

Bloque finally

Por último es posible utilizar un bloque *finally* que se ejecute al final del código, **ocurra o no ocurra un error**:

```
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") # Siempre se ejecuta
```

```
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
```

Última edición: 26 de Septiembre de 2018