**CONTROL DE ERRORES (EXCEPCIONES)**

**Python implementa excepciones con el propósito de conseguir código robusto.**

**Cuando ocurre un error en un programa, el código que encuentra el error lanza una excepción que se puede capturar desde la aplicación con el propósito de recuperarse de ella.**

Veamos un ejemplo de control de errores:

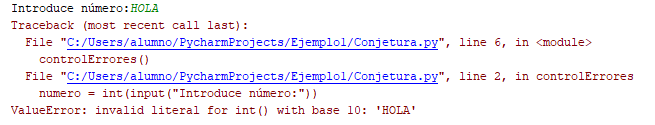
Solicitaremos al usuario que introduzca un número por teclado. Si lo hace correctamente e introduce el número comprobaremos que nuestra aplicación se ejecuta sin problema.

**def** controlErrores():

numero = int(input(**"Introduce número:"**))  
 print(**"Número:"**,numero)  
controlErrores()



No sucederá lo mismo si el usuario introduce texto.



* Nos dice claramente que no puede convertir una cadena (Hola) a entero.
* Tenemos que fijarnos en el nombre de la excepción que se ha producido (ValueError)
* Para que nuestro código sea totalmente correcto tenemos que tratar el error, mostraremos un mensaje al usuario para avisarle de lo sucedido.

**CONTROLANDO EXCEPTION**

En Python el manejo de excepciones se hace mediante los bloques que utilizan las sentencias try y except.

Sintaxis:

**try:**

# Incluimos el código que puede provocar excepciones

**except NOMBRE\_EXCEPTION:**

# Ejecutará las instrucciones que aquí pongamos en caso que se haya producido una excepción

Modificamos el ejemplo anterior para controlar la Exception ValueError.

**def** controlErrores():  
 **try**:  
 numero = int(input(**"Introduce número:"**))  
 print(**"Número:"**,numero)  
 **except** ValueError:  
 print (**"Error, debes introducir un número"**)  
  
controlErrores()

Obtendremos el siguiente resultado en caso de que el usuario introduzca una cadena:



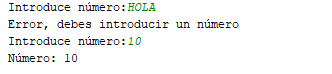
Si el usuario introduce un número, el resultado será correcto:



Otra posibilidad, es avisar del error y que el usuario vuelva a intentarlo.

Vemos en el ejemplo como volvemos a llamar a la función para que el usuario introduzca un número.

**def** controlErrores():  
 **try**:  
 numero = int(input(**"Introduce número:"**))  
 print(**"Número:"**,numero)  
 **except** ValueError:  
 print (**"Error, debes introducir un número"**)  
 controlErrores()  
  
controlErrores()



En la documentación oficial de Python podemos ver la lista de Excepciones integradas y su significado:

<https://docs.python.org/3.8/library/exceptions.html>

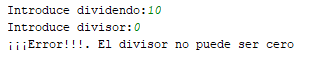
**MANEJANDO VARIOS ERRORES**

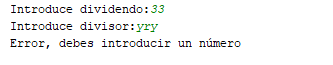
Dentro del bloque “try” se pueden producir varios errores, por ese motivo Python nos permite incluir varios bloques “except”, para capturar diferentes excepciones.

En el siguiente ejemplo realizaremos una división de dos números, controlando que se introduzcan número y que el divisor no sea cero.

**def** controlErrores():  
 **try**:  
 dividendo = int(input(**"Introduce dividendo:"**))  
 divisor = int(input(**"Introduce divisor:"**))  
 resultado = dividendo / divisor  
 print(**f"Resultado división: {resultado}"**)  
 **except** ValueError:  
 print(**"Error, debes introducir un número"**)  
 **except** ZeroDivisionError:  
 print(**"¡¡¡Error!!!. El divisor no puede ser cero"**)  
  
controlErrores()

RESULTADO:





Se puede utilizar la sentencia **except sin especificar ningún nombre de excepción** para captura cualquier tipo de error.

Modificamos el ejemplo anterior para que entre en la excepción ValueError o ZeroDivisionError si se producen sus respectivos errores, pero si se produce cualquier otro entrará en el último except.

**def** controlErrores():  
 **try**:  
 dividendo = int(input(**"Introduce dividendo:"**))  
 divisor = int(input(**"Introduce divisor:"**))  
 resultado = dividendo / divisor  
 print(**f"Resultado división: {resultado}"**)  
 **except** ValueError:  
 print(**"Error, debes introducir un número"**)  
 **except** ZeroDivisionError:  
 print(**"¡¡¡Error!!!. El divisor no puede ser cero"**)  
 **except**:  
 print(**"Error general"**)  
  
controlErrores()

\*\*\* Podemos probarlo eliminando el segundo Exception (ZerodivideError). Comprobaremos que entra en el Except sin nombre de error cuando intentemos dividir por cero.

**sys.exc\_info()**

Con la función exc\_info() podemos mostrar el mensaje de error que se ha producido.

Esa función pertenece al módulo sys, para poder utilizarlo realizaremos un import.

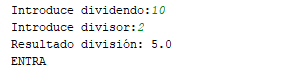
En el ejemplo mostraremos el mensaje de error que se produce si entra en el último except.

**import** sys  
**def** controlErrores():  
 **try**:  
 dividendo = int(input(**"Introduce dividendo:"**))  
 divisor = int(input(**"Introduce divisor:"**))  
 resultado = dividendo / divisor  
 print(**f"Resultado división: {**resultado**}"**)  
 **except** ValueError:  
 print(**"Error, debes introducir un número"**)  
 **except** ZeroDivisionError:  
 print(**"¡¡¡Error!!!. El divisor no puede ser cero"**)  
 **except**:  
 print(**f"Error: {**sys.exc\_info()**}"**)  
  
controlErrores()

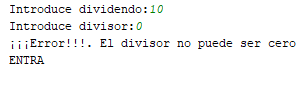
Aunque el bloque **finally** no es obligatorio, es importante conocer que existe por si queremos que se ejecute un bloque de código se produzca o no una excepción.

**def** controlErrores():  
 **try**:  
 dividendo = int(input(**"Introduce dividendo:"**))  
 divisor = int(input(**"Introduce divisor:"**))  
 resultado = dividendo / divisor  
 print(**f"Resultado división: {resultado}"**)  
 **except** ValueError:  
 print(**"Error, debes introducir un número"**)  
 **except** ZeroDivisionError:  
 print(**"¡¡¡Error!!!. El divisor no puede ser cero"**)  
 **finally**:  
 print(**"ENTRA"**)  
  
controlErrores()

RESULTADO SIN ERROR



RESULTADO CON ERROR



Ejemplo claro de utilizar **finally**

Imaginemos que estamos leyendo de una base de datos:

1. Abrir conexión con la base de datos
2. Ejecutar una consulta
3. Leer datos
4. Cerrar conexión

Conexión.abrir()

Consulta.Execute()

Consulta.Read()

Conexión.cerrar()

**try**:  
 Conexión.abrir()

Consulta.Execute()

Consulta.Read()

**except**:  
 print(“Error GORDO”)

**finally**:  
 Conexión.cerrar()

print(“Programa continuando tranquilo por aqui”)