

# Excepciones

Antonio Espín Herranz

# Introducción

- Las excepciones son **errores** detectados por Python durante la **ejecución** del programa. Cuando el intérprete se encuentra con una situación excepcional, situaciones que lanzan excepciones:
  - Intentar dividir un número entre 0
  - Acceder a un archivo que no existe.
- Este genera o lanza una excepción, informando al usuario de que existe algún problema.
- **Si la excepción no se captura el flujo de ejecución se interrumpe** y se muestra la información asociada a la excepción en la consola de forma que el programador pueda solucionar el problema.

# Ejemplo

```
def division(a, b):  
    return a / b
```

```
def calcular():  
    division(1, 0)
```

```
calcular()
```

## **Respuesta del intérprete:**

- \$ python ejemplo.py
- Traceback (most recent call last):
- File "ejemplo.py", line 7, in
- calcular()
- File "ejemplo.py", line 5, in calcular
- division(1, 0)
- File "ejemplo.py", line 2, in division
- a / b
- ZeroDivisionError: integer division or modulo by zero

# Excepciones

- Palabras: **try** y **except**.

**try:**

Acción potencialmente errónea.

Acción potencialmente errónea 2.

**except:**

Acción para tratar el error.

EJEMPLO:

**try:**

```
f = open("archivo.txt")
```

**except:**

```
print "El archivo no existe"
```

# Ejemplo

#Otro ejemplo de captura de excepciones:

```
a = 0
```

```
b = 3
```

```
try:
```

```
    x = -b / a
```

```
    print ('solucion: %d' % x)
```

```
except:
```

```
    if b != 0:
```

```
        print ('la ecuacion no tiene solucion')
```

```
    else:
```

```
        print ('tiene infinitas soluciones')
```

# Excepciones

- **Python permite utilizar varios except para un solo bloque try**, de forma que podamos dar un tratamiento distinto a la excepción dependiendo del tipo de excepción de la que se trate.
- Se indica el nombre del tipo de excepción a continuación de la palabra **except**.

**try:**

```
num = int("3a")  
print (no_existe)
```

**except NameError:**

```
print ("La variable no existe")
```

**except ValueError:**

```
print ("El valor no es un numero")
```

En este caso si se lanza otra  
Excepción distinta se propagaría ...

# Excepciones

- Otra posible estructura del bloque try ... except, indicar todas las excepciones entre paréntesis.

```
try:
```

```
    num = int("3a")
```

```
    print (no_existe)
```

```
except (NameError, ValueError):
```

```
    print "Ocurrio un error"
```

# Excepciones

- Se puede añadir una clausula else para indicar que no hay excepciones:

```
try:
```

```
    num = 33
```

```
except:
```

```
    print ("Hubo un error!")
```

```
else:
```

```
    print("Todo esta bien")
```



# Excepciones

- La clausula finally que se ejecuta siempre, se produzca o no una excepción.
- Esta clausula se suele utilizar, entre otras cosas, para tareas de limpieza.

```
try:
```

```
    z = x / y
```

```
except ZeroDivisionError:
```

```
    print ("Division por cero")
```

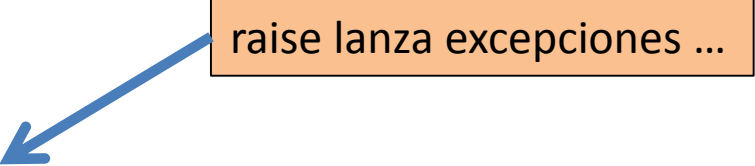
```
finally:
```

```
    print ("Limpiando")
```

# Excepciones personalizadas

```
class MiError(Exception):  
    def __init__(self, valor):  
        self.valor = valor  
  
    def __str__(self):  
        return "Error " + str(self.valor)
```

```
try:  
    if resultado > 20:  
        raise MiError(33)  
except MiError as e: # válido también: except MiError as e:  
    print (e)
```



raise lanza excepciones ...

# Excepciones personalizadas

- Cuando heredamos de Exception si no necesitamos añadir atributos (a parte del mensaje asociado al mensaje de error) podemos hacerlo así:

```
class MiException(Exception):  
    pass
```

- Dispondremos del mensaje asociado al error.
- Al lanzar nuestra excepción haremos:

```
raise MiException("mensaje de error")
```

# Jerarquía de Excepciones

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StandardError
        | +-- BufferError
        | +-- ArithmeticError
        | | +-- FloatingPointError
        | | +-- OverflowError
        | | +-- ZeroDivisionError
        +-- AssertionError
        +-- AttributeError
        +-- EnvironmentError
        | +-- IOError
        | +-- OSError
        | | +-- WindowsError (Windows)
        | | +-- VMSError (VMS)
        +-- EOFError
        +-- ImportError
        +-- LookupError
        | +-- IndexError
        | +-- KeyError
        +-- MemoryError
        +-- NameError
        | +-- UnboundLocalError
        +-- ReferenceError
        +-- RuntimeError
        | +-- NotImplementedError
        +-- SyntaxError
        | +-- IndentationError
        | +-- TabError
        +-- SystemError
        +-- TypeError
        +-- ValueError
        | +-- UnicodeError
        | | +-- UnicodeDecodeError
        | | +-- UnicodeEncodeError
        | | +-- UnicodeTranslateError
    +-- Warning
        +-- DeprecationWarning
        +-- PendingDeprecationWarning
        +-- RuntimeWarning
        +-- SyntaxWarning
        +-- UserWarning
        +-- FutureWarning
        +-- ImportWarning
        +-- UnicodeWarning
        +-- BytesWarning
```

# Lista de excepciones predefinidas

- **BaseException:**
  - Clase de la que heredan todas las excepciones.
- **Exception(BaseException):**
  - Super clase de todas las excepciones que no sean de salida.
- **GeneratorExit(Exception):**
  - Se pide que se salga de un generador.
- **StandardError(Exception):**
  - Clase base para todas las excepciones que no tengan que ver con salir del intérprete.

# Lista de excepciones predefinidas

- **ArithmeticError(StandardError):**
  - Clase base para los errores aritméticos.
- **FloatingPointError(ArithmeticError):**
  - Error en una operación de coma flotante.
- **OverflowError(ArithmeticError):**
  - Resultado demasiado grande para poder representarse.
- **ZeroDivisionError(ArithmeticError):**
  - Lanzada cuando el segundo argumento de una operación de división o módulo era 0.

# Lista de excepciones predefinidas

- **AssertionError(StandardError):**
  - Falló la condición de un estamento assert.
- **AttributeError(StandardError):**
  - No se encontró el atributo
- **EOFError(StandardError):**
  - Se intentó leer más allá del final de fichero.
- **EnvironmentError(StandardError):**
  - Clase padre de los errores relacionados con la entrada/salida.

# Lista de excepciones predefinidas

- **IOError(EnvironmentError):**
  - Error en una operación de entrada/salida.
- **OSError(EnvironmentError):**
  - Error en una llamada a sistema.
- **WindowsError(OSError):**
  - Error en una llamada a sistema en Windows.
- **ImportError(StandardError):**
  - No se encuentra el módulo o el elemento del módulo que se quería importar.



# Lista de excepciones predefinidas

- **LookupError(StandardError):**
  - Clase padre de los errores de acceso.
- **IndexError(LookupError):**
  - El índice de la secuencia está fuera del rango posible.
- **KeyError(LookupError):**
  - La clave no existe.
- **MemoryError(StandardError):**
  - No queda memoria suficiente.
- **NameError(StandardError):**
  - No se encontró ningún elemento con ese nombre.
- **UnboundLocalError(NameError):**
  - El nombre no está asociado a ninguna variable.

# Lista de excepciones predefinidas

- **ReferenceError(StandardError):**
  - El objeto no tiene ninguna referencia fuerte apuntando hacia él.
- **RuntimeError(StandardError):**
  - Error en tiempo de ejecución no especificado.
- **NotImplementedError(RuntimeError):**
  - Ese método o función no está implementado.
- **SyntaxError(StandardError):**
  - Clase padre para los errores sintácticos.

# Lista de excepciones predefinidas

- **IndentationError(SyntaxError):**
  - Error en la indentación del archivo.
- **TabError(IndentationError):**
  - Error debido a la mezcla de espacios y tabuladores.
- **SystemError(StandardError):**
  - Error interno del intérprete.
- **TypeError(StandardError):**
  - Tipo de argumento no apropiado.
- **ValueError(StandardError):**
  - Valor del argumento no apropiado.
- **UnicodeError(ValueError):**
  - Clase padre para los errores relacionados con unicode.

# Lista de excepciones predefinidas

- **UnicodeDecodeError(UnicodeError):**
  - Error de decodificación unicode.
- **UnicodeEncodeError(UnicodeError):**
  - Error de codificación unicode.
- **UnicodeTranslateError(UnicodeError):**
  - Error de traducción unicode.
- **StopIteration(Exception):**
  - Se utiliza para indicar el final del iterador.
- **Warning(Exception):**
  - Clase padre para los avisos.
- **DeprecationWarning(Warning):**
  - Clase padre para avisos sobre características obsoletas.

# Lista de excepciones predefinidas

- **FutureWarning(Warning):** Aviso.
  - La semántica de la construcción cambiará en un futuro.
- **ImportWarning(Warning):**
  - Aviso sobre posibles errores a la hora de importar.
- **PendingDeprecationWarning(Warning):**
  - Aviso sobre características que se marcarán como obsoletas en un futuro próximo.
- **RuntimeWarning(Warning):**
  - Aviso sobre comportamientos dudosos en tiempo de ejecución.

# Lista de excepciones predefinidas

- **SyntaxWarning(Warning):**
  - Aviso sobre sintaxis dudosa.
- **UnicodeWarning(Warning):**
  - Aviso sobre problemas relacionados con Unicode, sobre todo con problemas de conversión.
- **UserWarning(Warning):**
  - Clase padre para avisos creados por el programador.
- **KeyboardInterrupt(BaseException):**
  - El programa fué interrumpido por el usuario.
- **SystemExit(BaseException):**
  - Petición del intérprete para terminar la ejecución.