T- Control de errores

Contenido

Introducción

Los errores en los programas aparecen por diferentes situaciones, se pueden dar por un código mal programado, unos datos mal introducidos, o por otro tipo de causas. Independientemente de por qué se ha generado, es imprescindible realizar una gestión adecuada del fallo para evitar que la aplicación finalice sin ninguna explicación. Para la adecuada gestión de los errores, el sistema nos va a presentar un modelo basado en la gestión de Excepciones.

Una Excepción es un objeto que contiene información sobre un error que se ha producido durante la ejecución. Toda excepción generada tendrá que recogerse y tratarse de alguna manera, si no el propio interprete proporciona un mecanismo por defecto que consiste en mostrar un mensaje por consola y parar la aplicaciónón.

Las Excepciones son objetos explicativos, siendo una de las informaciones más valiosas que aportan el tipo. Cada una tendrá un tipo definido en el lenguaje dándonos datos sobre las causas que han generado el fallo. Así, una Excepción de IOError determinará un error en una operación de entrada - salida, una de MemoryError se generará cuando no quede memoria suficiente, etc.

Captura

Las excepciones se lanzan por el sistema y estamos en la obligación de capturarlas evitando el comportamiento por defecto. La gestión se realiza mediante una construcción conocida de otros lenguajes:

```
try ... except. .. else ... finaly   (try catch de Java).
```

Para capturar un posible error encerraremos dentro del bloque try las sentencias susceptibles de generarlo, determinando con tantas sentencias except como necesitemos los posibles tipos de error producidos, si dos errores se van a tratar de forma similar es posible añadir en una línea except ambos, encerrándolos entre paréntesis y separándolos con comas.

Añadiremos un bloque else si queremos dotar un tratamiento por defecto en caso que no se produzca un error y dentro de finally se situaran aquellas sentencias que se ejecutaran siempre tras el tratamiento de un problema independientemente de este, generalmente para realizar algún tipo de limpieza.

Hemos dicho que el sistema genera las excepciones, pero en algunos casos es recomendable usar nosotros el mismo mecanismo para comunicar un error. Podemos lanzar una excepción en cualquier momento con la palabra clave raise y un objeto que herede de Exception. Dotando de esta manera un mecanismo coherente de gesti6n de los errores independientemente de cómo se produzca.

Las excepciones más comunes se muestran a continuación:

- BaseException. Clase base para todas las demás.
- Exception. Clase base para todas las excepciones que no son de entrada - salida.
- AritmeticError. Clase base para las errores aritméticos, extienden de ella: FloatingPointError, OverflowError, ZeroDivisionError.
- EOFError. Sobrepasado el fin del fichero.
- IOError. Error de entrada - salida.
- ImportEror. Error en la importación, con este error podemos comprobar funcionalidades instaladas en el sistema.
- IndexError. Error de índice en el acceso.
- KeyError. Clave no existente.

Más excepciones integradas:

- https://docs.python.org/3/library/exceptions.html#concrete-exceptions
- http://bioportal.weizmann.ac.il/course/python/PyMOTW/PyMOTW/docs/exceptions/index.html

Ejemplo

- Código:

```
if __name__ == '__main__':
    try:
        numero1=int(input("Introduce un numero dividendo: "))
        numero2=int(input("Introduce un numero divisor: "))
        numero1/numero2

    except NameError:
        print "Has introducido un carácter no numérico"
    except ZeroDivisionError:
        print "No es posible dividir por 0"
    except ValueError:
        print "Posible valor introducido no valido"
    except Exception as Ex:
        print "Error sin definir:",Ex
    finally:
        print "\rNo problemmm, be happy. Continua.....\r\r"

    class miExcepcion (Exception):
        def __init__(self,val):
            self.valor=val
        def __str__(self):
            cadena= "Mi Mensaje:",self.valor
```

```
            return str(cadena)

    try:
        raise miExcepcion ("El programa va a finalizar")
    except miExcepcion as e:
        print e
```

- Ejemplo1 de ejecución:

```
Introduce un número dividendo: 45
Introduce un número divisor: 5
No problemmm, be happy. Continúa.....
('Mi Mensaje:', 'El programa va a finalizar')
```

- Ejemplo2 de ejecución:

```
Introduce un número dividendo: 45
Introduce un número divisor: /
Error sin definir: unexpected EOF while parsing (<string>,
line 1)

No problemmm, be happy. Continúa.....

('Mi Mensaje:', 'El programa va a finalizar')
```

- Ejemplo3 de ejecución:

```
Introduce un número dividendo: 56
Introduce un número divisor: 0
No es posible dividir por 0

No problemmm, be happy. Continúa.....

('Mi Mensaje:', 'El programa va a finalizar')
```

- Ejemplo4 de ejecución:

```
Introduce un número dividendo: 56
Introduce un número divisor: sd
Has introducido un carácter no numérico

No problemmm, be happy. Continúa.....

('Mi Mensaje:', 'El programa va a finalizar')
```

Más información: https://docs.python.org/2/library/exceptions.html