INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

En un programa podemos encontrarnos con distintos tipos de errores pero a grandes rasgos podemos decir que todos los errores pertenecen a una de las siguientes categorías:

- ✓ Errores de sintaxis.
- ✓ Errores semánticos.
- ✓ Errores de ejecución

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

Errores de sintaxis.

Estos errores son seguramente los más simples de resolver, pues son detectados por el **intérprete** (o por el compilador, según el tipo de lenguaje que estemos utilizando) al procesar el código fuente y generalmente son consecuencia de **equivocaciones al escribir** el programa. En el caso de Python estos errores son indicados con un mensaje ***SyntaxError***.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

Errores semánticos.

Se dan cuando un programa, a pesar de no generar mensajes de error, **no produce el resultado esperado**. Esto puede deberse, por ejemplo, a un algoritmo incorrecto o a la omisión de una sentencia.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

Errores de ejecución.

Estos errores aparecen durante la **ejecución del programa** y su origen puede ser diverso. En ocasiones pueden producirse por un uso incorrecto del programa por parte del usuario, por ejemplo si el usuario ingresa una cadena cuando se espera un número. En otras ocasiones pueden deberse a errores de programación.

INTRODUCCIÓN A LA PROGRAMACIÓN

Python - Errores.

Tanto a los errores de **sintaxis** como a los **semánticos** se los puede **detectar y corregir durante la construcción del programa** ayudados por el intérprete y la ejecución de pruebas. Pero no ocurre esto con los errores de **ejecución** ya que no siempre es posible saber cuando ocurrirán y puede resultar muy complejo (o incluso casi imposible) reproducirlos.

INTRODUCCIÓN A LA PROGRAMACIÓN

Python - Errores.

Los **errores de ejecución** son llamados comúnmente **excepciones**. Durante la ejecución de un programa, si dentro de una función surge una excepción y la función no la maneja, la excepción se propaga hacia la función que la invocó, si esta otra tampoco la maneja, la excepción continua propagándose hasta llegar a la función inicial del programa y si esta tampoco la maneja se interrumpe la ejecución del programa.

INTRODUCCIÓN A LA PROGRAMACIÓN

Python - Errores.

Gestionar una excepción recibe el nombre de **capturar una excepción**.

Capturar una excepción no es otra cosa que guardar el estado en el que se encontraba el programa en el momento justo del error e interrumpir el programa para ejecutar un código fuente concreto. En muchos casos, dependiendo del error ocurrido, el control de la excepción implicará que el programa siga ejecutándose después de controlarla, aunque en muchos casos esto no será posible.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

El proceso de controlar excepciones es similar en todos los lenguajes de programación.

En primer lugar, es necesario incluir el código fuente de ejecución normal dentro de un bloque con la sentencia **try**.

Posteriormente, se crea un bloque de código dentro de una sentencia **except** que es la que se ejecutará en caso de error.

El bloque **except** permite especificar el tipo de error que se controla con el bloque de código, es por ello por lo que puedes tener tantos bloques except como errores quieras controlar, aunque también es posible controlar un error genérico que incluya a todos los errores.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

En el control de excepciones existe la posibilidad de crear un bloque de código que se ejecute siempre al final, independientemente de si ocurre error o no. Dicho bloque de código se escribe como parte de la sentencia **finally**.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

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

Dentro del bloque **try** se ubica todo el **código que pueda llegar a generar una excepción**.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

A continuación se ubica el bloque **except**, que se encarga de **capturar la excepción** y nos da la oportunidad **de procesarla** mostrando por ejemplo un mensaje adecuado al usuario.

Dado que dentro de un mismo bloque **try** pueden producirse excepciones de distinto tipo, es posible utilizar **varios** bloques **except**, cada uno para capturar un tipo **distinto** de **excepción**.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

Un **mismo bloque except** puede atrapar **varios** tipos de excepciones, lo cual se hace especificando los nombres de la excepciones separados por comas a continuación de la palabra **except**.

También es posible utilizar una sentencia **except** sin especificar el tipo de excepción a capturar, en cuyo caso se captura cualquier excepción, sin importar su tipo.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

```
try:
    # aquí ponemos el código que puede lanzar excepciones
except IOError:
    # entrará aquí en caso que se haya producido una excepción IOError
except ZeroDivisionError:
    # entrará aquí en caso que se haya producido una excepción ZeroDivisionError
except:
    # entrará aquí en caso que se haya producido
    # una excepción que no corresponda a ninguno
    # de los tipos especificados en los except previos
```

En caso de utilizar una sentencia **except** sin especificar el tipo, la misma debe ser siempre la **última** de las sentencias except.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

En el control de excepciones es posible añadir bloques **else** al **except** si es que necesitamos ejecutar un código si todo va bien.

Si el código realiza su trabajo exitosamente, significa que no hubo nunca una excepción, por lo que pasará directamente al else.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

Finalmente, puede ubicarse un bloque **finally** donde se escriben las **sentencias de finalización**, que son típicamente **acciones de limpieza**. La particularidad del bloque finally es que se **ejecuta siempre**, haya surgido una excepción o no.

Si hay un bloque **except**, no es necesario que esté presente el **finally**, y es posible tener un bloque **try** sólo con **finally**, sin **except**.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

Python comienza a ejecutar las instrucciones que se encuentran dentro de un bloque **try** normalmente. Si durante la ejecución de esas instrucciones se **produce** una **excepción**, Python **interrumpe** la ejecución en el **punto exacto** en que surgió la excepción y pasa a la ejecución del bloque **except** correspondiente.

Para ello, Python **verifica uno a uno** los bloques **except** y si encuentra alguno cuyo tipo haga referencia al tipo de excepción levantada, comienza a ejecutarlo. Sino encuentra ningún bloque del tipo correspondiente pero hay un bloque **except** sin tipo, lo ejecuta. Al terminar de ejecutar el bloque correspondiente, se pasa a la ejecución del bloque **finally**, si se encuentra definido.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

Si, por otra parte, no hay problemas durante la ejecución del bloque **try**, se completa la ejecución del bloque, y luego se pasa directamente a la ejecución del bloque **finally** (si es que está definido).

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

```
print("¡Iniciando programa!")
try:
    print(str(17/0))
except ZeroDivisionError:
    print("ERROR: Division por cero")
except:
   print("ERROR: General")
else:
    print("¡No se han producido errores!")
finally:
    print("¡Programa acabado!")
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

```
try:
    num = float(input("Introduce un número: "))
    b = 10
    print(b/num)
except TypeError:
    print("Error al intentar dividir una cadena")
except ValueError:
    print("Introduce una cadena que sea un número")
except ZeroDivisionError:
    print("Introduce un número mayor a cero")
except Exception as e:
    print(type(e).__name__)
    #En la última línea podemos capturar un error genérico y ver su valor.
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

```
try:
    archivo = open("miarchivo.txt")
    # procesar el archivo
except IOError:
    print ("Error de entrada/salida.“)
    # procesar la excepción IOError
except:
   # procesar cualquier otra excepción
finally:
   # si el archivo no está cerrado hay que cerrarlo
    if not(archivo.closed):
        archivo.close()
```

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

Resumiendo, tenemos lo siguiente:

- **Try**: captura una excepción
- **Except**: ejecuta lo que captura de la excepción. Depende de nosotros como programadores lo que queramos mostrar al usuario.
- **Else**: ejecuta si todo va bien
- **Finally**: ejecuta al finalizar el try.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

En Python existen diferentes tipos de excepciones que pueden controlarse, todas ellas derivan de una serie de excepciones base.
Las excepciones base son las siguientes:

- **Excepción**: tipo de excepción más genérica, de ella derivan todas las excepciones existentes en Python.
- **ArithmeticError**: tipo de excepción genérica para errores aritméticos.
- **BufferError**: tipo de excepción genérica para errores relacionados con buffers.
- **LookupError**: tipo de excepción genérica para errores relacionados con acceso a datos de colecciones.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

El grupo de excepciones base anterior da lugar a un grupo de excepciones concretas:

- ✓ **AssertionError**: excepción que se lanza cuando la instrucción assert falla.
- ✓ **AttributeError**: excepción que se lanza cuando hay un error a la hora de asignar un valor a un atributo o cuando se intenta acceder a él.
- ✓ **EOFError**: excepción que se lanza cuando la instrucción input no devuelve datos leídos.
- ✓ **GeneratorExit**: excepción que se lanza cuando se cierra una función de tipo generator o coroutine.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- ✓ **ImportError**: excepción que se lanza cuando se intenta importar un módulo al programa y falla.
- ✓ **ModuleNotFoundError**: excepción que se lanza cuando se intenta importar un módulo y no se encuentra. Deriva de la anterior.
- ✓ **IndexError**: excepción que se lanza cuando se intenta acceder a una posición de una secuencia y ésta es superior a la posición mayor.
- ✓ **KeyError**: excepción que se lanza cuando se intenta acceder a la clave de un diccionario y no se encuentra.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- ✓ **KeyboardInterrupt**: excepción que se lanza cuando el usuario utiliza el comando de interrupción con el teclado (Control-C o delete).
- ✓ **MemoryError**: excepción que se lanza cuando el programa ejecuta una instrucción y ésta supera el máximo de memoria disponible.
- ✓ **NameError**: excepción que se lanza cuando el nombre local o global no se encuentra.
- ✓ **NotImplementedError**: excepción que se lanza cuando un método de una clase no ha sido implementado todavía y tiene que hacerlo.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- ✓ **OSError**: excepción que se lanza cuando el sistema operativo lanza una excepción al ejecutar una instrucción.

Existen las siguientes excepciones específicas que puede lanzar el sistema operativo:

- **BlockingIOError**: excepción que se lanza cuando una operación se bloquea en un objeto que no debería de bloquearse.
- **ChildProcessError**: excepción que se lanza cuando una operación de un proceso hijo devuelve un error.
- **ConnectionError**: excepción genérica que se lanza para errores relacionados a conexión.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- **BrokenPipeError**: excepción que se lanza cuando se intenta escribir en un socket o pipe (tubería) y ya ha sido cerrado.
- **ConnectionAbortedError**: excepción que se lanza cuando durante un intento de conexión ésta es abortada por el otro extremo.
- **ConnectionRefusedError**: excepción que se lanza cuando durante un intento de conexión ésta es rechazada por el otro extremo.
- **ConnectionResetError**: excepción que se lanza cuando la conexión es reseteada por el otro extremo.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- **FileExistsError**: excepción que se lanza cuando se intenta crear un fichero o directorio y éste ya existe.
- **FileNotFoundError**: excepción que se lanza cuando se intenta acceder a un fichero o directorio y no existe o no se encuentra.
- **IsADirectoryError**: excepción que se lanza cuando se intenta ejecutar una operación relacionada con ficheros sobre un directorio.
- **NotADirectoryError**: excepción que se lanza cuando se intenta ejecutar una operación relacionada con directorios sobre algo que no es un directorio.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- **PermissionError**: excepción que se lanza cuando se intenta ejecutar una operación y no se tienen los permisos suficientes.
- **ProcessLookupError**: excepción que se lanza cuando se ejecuta un proceso que no existe y se ha indicado que sí.
- **TimeoutError**: excepción que se lanza cuando se sobrepasa el tiempo de espera en alguna función del sistema.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- ✓ **OverflowError**: excepción que se lanza cuando el resultado de una operación matemática es demasiado grande para ser representado.
- ✓ **RecursionError**: excepción que se lanza cuando el número de recursividades supera el máximo permitido.
- ✓ **ReferenceError**: excepción que se lanza al intentar acceder a ciertos atributos por parte de la clase proxy y que ya se encuentran en el recolector de basura.
- ✓ **RuntimeError**: excepción que se lanza cuando el error que ocurre no puede ser categorizado en ninguno de los tipos existentes.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- ✓ **StopIteration**: excepción que se lanza cuando se intenta acceder al siguiente elemento de un iterador y ya no tiene más elementos sobre los que iterar.
- ✓ **StopAsyncIteration**: excepción que se lanza cuando se intenta acceder al siguiente elemento de un iterador asíncrono y ya no tiene más elementos sobre los que iterar.
- ✓ **SyntaxError**: excepción que se lanza cuando el analizador sintáctico encuentra un error de sintaxis.
- ✓ **IndentationError**: excepción genérica que se lanza cuando se encuentran errores de indentación del código fuente.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- ✓ **TabError**: excepción que se lanza cuando se encuentran errores de uso en las tabulaciones y espaciados del código fuente. La excepción deriva de la anterior.
- ✓ **SystemError**: excepción que se lanza cuando el intérprete de Python encuentra un error interno mientras ejecuta el programa.
- ✓ **SystemExit**: excepción que se lanza al ejecutar la instrucción sys.exit() y que provoca que se pare la ejecución del programa.
- ✓ **TypeError**: excepción que se lanza cuando una operación o función se usa con un tipo de dato incorrecto.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- ✓ **UnboundLocalError**: excepción que se lanza cuando se utiliza una variable local en una función o método y no ha sido asignado ningún valor previamente.
- ✓ **UnicodeError**: excepción que se lanza cuando se produce un error a la hora de codificar o decodificar Unicode.
- ✓ **UnicodeEncodeError**: excepción que se lanza cuando se produce un error a la hora de realizar una codificación a Unicode.
- ✓ **UnicodeDecodeError**: excepción que se lanza cuando se produce un error a la hora de realizar una decodificación de Unicode.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

- ✓ **UnicodeTranslateError**: excepción que se lanza cuando se produce un error a la hora de traducir Unicode.
- ✓ **ValueError**: excepción que se lanza cuando una operación o función recibe un parámetro del tipo correcto, pero con un valor incorrecto.
- ✓ **ZeroDivisionError**: excepción que se lanza cuando se realiza una división por cero.

INTRODUCCIÓN A LA PROGRAMACIÓN

Errores y Excepciones.

“Fin del tema”