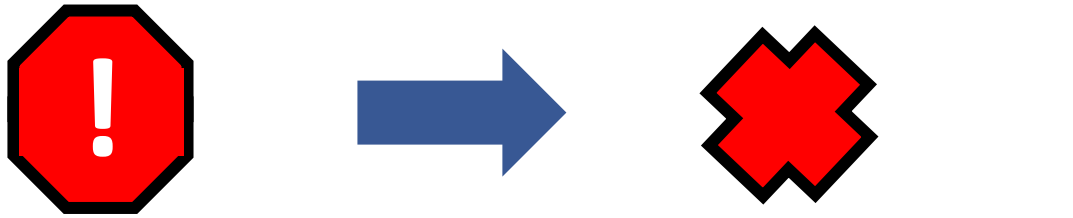
# Errores y Excepciones

Es un problema en un programa de computador o sistema de software que desencadena un resultado indeseado. Estos pueden ser generado por el desarrollador mientras construye el programa, o se generan durante la ejecución del código por situaciones anormales que no se esperaban.



Excepción

Ejecución detenida

# Errores de sintaxis

- Estructura incorrecta del código. Ya que no se reconoce dicha estructura.
- Es responsabilidad del desarrollador.

Nota: Es común cuando se está aprendiendo el lenguaje

# Excepciones

- Se producen durante la ejecución del programa.
- Son situaciones anormales que no se esperaban y que, por tanto, no se sabe como tratarlas.
- Existen diferentes tipos de excepciones.
- Provee información del tipo de la excepción y que la causo.
- Los nombres de las excepciones estándar son identificadores incorporado.

Ejemplo:

```
Traceback (most recent call last):
  File "c:/Users/Computador/Desktop/Proyectos/errores.py", line 2, in <module>
    numero = int("a")
ValueError: invalid literal for int() with base 10: 'a'
```

# Excepciones comunes

- SyntaxError.
- NameError
- TypeError
- IndexError
- ValueError
- AttributeError
- KeyError
- OSError
- RuntimeError
- Etc.

# Gestión de excepciones

- Cláusula **try**: Se ejecuta este bloque de código, el cual contiene uno o varios potenciales excepciones. En caso que ocurra una excepción, se deja de ejecutar esta cláusula y continua en **except**.

- Cláusula **except**: Bloque de código que se ejecuta cuando ocurre un excepción en la cláusula **try**. En caso de no ocurrir, se omitirá.

- Cláusula **else**: Solo se ejecutará el bloque de código cuando no ocurran excepciones. Su uso no es de carácter obligatorio.

- Cláusula **finally**: Siempre se ejecutará el bloque de código. Su uso no es de carácter obligatorio.

Nota: La cláusula **try**, puede gestionar también las excepciones que ocurren dentro de funciones que son llamadas directa o indirectamente en ella.

# Gestión de excepciones

- Se pueden colocar varias cláusulas **except**, para especificar gestores para diferentes excepciones. Se ejecutará un solo gestor.

  ejemplo: **except ValueError**:

  y

  **except ZeroDivisionError:**

  **…**

- La cláusula **except**, se puede continuar colocando uno o varios tipos de excepciones especificas que se desean manejar.

  ejemplo: **except ValueError**:

  ó

  **except (ValueError, ZeroDivisionError, … ):**

Nota: Si ocurre una excepción que no coincidan con las indicada en las clausulas **except** se pasa a los **try** externos, si no se encuentra un gestor, se genera un **unhandled exception**.

# Gestión de excepciones

Sintaxis:

```
try:
    código
except:
    código
```

```
try:
    código
except  NameClassError:
    código
```

```
try:
    código
except  NombreClaseError as error:
    código
```

```
try:
    código
except  NameClassError:
    código
except  NameDosClassError:
    código
except … :
    código
else:
    código
finally:
    código
```

# Gestión de excepciones

- Cuando ocurre una excepción, puede tener un valor asociado, también conocido como el argumento de la excepción. La presencia y el tipo de argumento depende del tipo de excepción.

- La cláusula **except** puede especificar una variable después del nombre de la excepción. La variable está ligada a la instancia de la excepción, que normalmente tiene un atributo **args** que almacena los argumentos.

- Por conveniencia, los tipos de excepción incorporados definen **__str__()** para imprimir todos los argumentos sin acceder explícitamente a **args**.

ejemplo:

```
except ValueError as error:
    print( error.args )
    print( error.__str__() )
```

→

```
("invalid literal for int() with base 10: '',)
invalid literal for int() with base 10: ‘’
```

# Gestión de excepciones

- Una clase en una cláusula **except** es compatible con una excepción si estas clases es **BaseException** o una subclase de esta, por ejemplo: **Exception**.

ejemplo:

```
class NombreClaseError(Exception):
    pass

class NombreClaseDosError(NombreClaseError):
    pass

try:
    pass
except NombreClaseDosError :
    pass
except NombreClaseError:
    pass
```

# Lanzando excepciones

- Para forzar a que ocurra una excepción específica se debe utilizar la declaración **raise**.
- **raise** solo pude lanzar una instancia de la clase **BaseException** o una subclase de esta, por ejemplo: **Exception**.
- Si se pasa una clase de excepción, se instanciará implícitamente llamando a su constructor sin argumentos.
- Si una excepción fue lanzada pero sin intención de gestionarla, se debe colocar la instrucción **raise** en el **except** correspondiente para permite relanzarla.

# Encadenamiento de excepciones

- Si se produce una excepción no gestionada dentro de una sección **except**, se le adjuntará la excepción que se está gestionando y se incluirá en el mensaje de error.
- Para indicar que una excepción es consecuencia directa de otra, la sentencia **raise** permite una cláusula opcional **from** <instancia>.
- Para deshabilitar el encadenamiento automático de excepciones se utiliza **from None.**

# Excepciones definidas por el usuario

- Se pueden crear excepciones personalizadas implantando **clases que hereden de Exception** o de subclases de esta.
- Se recomienda que la excepción solo tenga un número de atributos con información del error que leerán los gestores de la excepción.
- Se recomienda que el nombre de la excepción termine en “Error”, de manera similar a la nomenclatura de las excepciones estándar.

Nota: Los módulos estándar definen sus propias excepciones para reportar errores que puedan ocurrir en funciones propias.

# Cláusula **finally**

- Se ejecutará al final antes de que todo el bloque **try** se complete.
- Se ejecuta independientemente de que la cláusula **try** produzca o no una excepción.
- Si ocurre una excepción durante la ejecución de la cláusula **try** la excepción podría ser gestionada por una cláusula **except**. Si la excepción no es gestionada por una cláusula **except** la excepción es relanzada después de que se ejecute el bloque de la cláusula **finally**.
- Podría aparecer una excepción durante la ejecución de una cláusula **except** o **else**. De nuevo, la excepción será relanzada después de que el bloque de la cláusula **finally** se ejecute.

# Cláusula **finally**

- Si la cláusula **finally** ejecuta una declaración **break**, **continue** o **return**, las excepciones no se vuelven a lanzar.

- Si el bloque **try** llega a una sentencia **break**, **continue** o **return**, la cláusula finally se ejecutará justo antes de la ejecución de dicha sentencia.

- Si una cláusula **finally** incluye una sentencia **return**, el valor retornado será el de la cláusula **finally**, no la del de la sentencia **return** de la cláusula **try**.

# Acciones predefinidas de limpieza

- Se realiza a objetos que ya no son necesarios (independientemente de que las operaciones sobre el objeto hayan sido exitosas o no).
- La declaración **with** permite que los objetos como archivos sean usados de una forma que asegure que siempre se los libera rápido y en forma correcta.
- Una vez que la declaración se ejecuta, el fichero *f* siempre se cierra, incluso si aparece algún error durante el procesado de las líneas.
- Los objetos que, como los ficheros, posean acciones predefinidas de limpieza lo indicarán en su documentación.

# Lanzando y gestionando múltiples excepciones no relacionadas

- ExceptionGroup envuelve una lista (puede estar anidada) de instancias de excepción para que puedan ser lanzadas juntas. Es una excepción en sí misma, por lo que puede capturarse como cualquier otra excepción.

- Utilizando **except*** en lugar de **except**, podemos manejar selectivamente sólo las excepciones del grupo que coincidan con un determinado tipo.

- Cada cláusula **except*** extrae del grupo las excepciones de un tipo determinado, mientras que deja que el resto de excepciones se propaguen a otras cláusulas y, finalmente, se vuelvan a lanzar.

- Las excepciones anidadas en un grupo de excepciones deben ser instancias, no tipos.

# Enriqueciendo excepciones con notas

- Se puede añadir información a una excepción después de que la haya sido capturada.
- El método add_note( note ) acepta una cadena y la añade a la lista de notas de la excepción.
- La representación estándar del rastreo incluye todas las notas, en el orden en que fueron añadidas, después de la excepción.