.UBA fiuba

FACULTAD DE INGENIERÍA

Exceptions

Algoritmos y Programación I

rev1.0

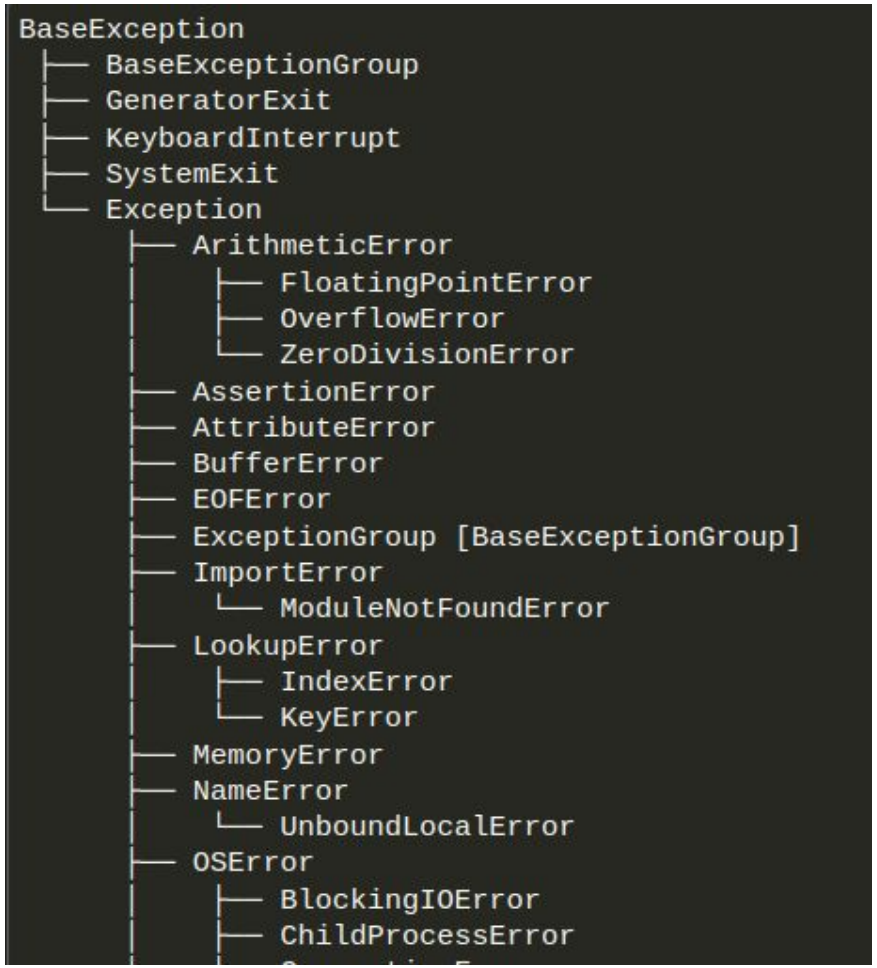
Exceptions en python:

Incluso si una declaración o expresión es sintácticamente correcta, puede generar un error cuando se intenta ejecutar. Los errores detectados por el intérprete durante la ejecución se llaman **excepciones**

Ejemplos:

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'spam' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

Tipos y jerarquías:



Las excepciones creadas por el programador deberán heredar de la clase **Exception**

Jerarquía completa

Gestionando excepciones:

- Forma de capturar errores inesperados.
- Nos permite evitar el cierre abrupto de la ejecución de un programa producido por errores inesperados.
- Debemos ser capaces de anticiparnos a las situaciones en donde puedan producirse excepciones para capturarlas y decidir qué se debe hacer en el programa.

Uso try...except :

```
try:
    ...
except Exception:
    ...
```

Dentro del bloque **try** colocamos la porción de código que podría causar una **Exception**.

Dentro del bloque except colocamos el código que debe ejecutarse ante la captura de la **Exception.**

Uso finally (opcional):

```
try:
    ...
except Exception:
    ...
finally:
    ...
```

El bloque **finally** se ejecutará al final antes de que todo el bloque **try** se complete.

La cláusula **finally** se ejecuta independientemente de que la cláusula **try** produzca o no una **Exception**.

Uso del objeto Exception:

```
try:
    ...
except Exception as error:
    ...
```

Al capturar la **exception**, podemos hacer uso del objeto para obtener información propia de la misma.