Python Intermedio

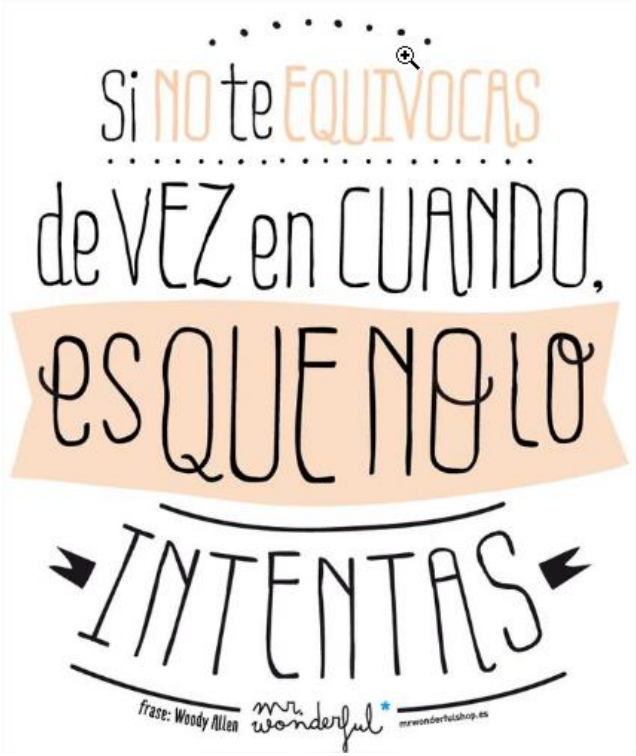
Manejo de Excepciones

Enero 2017

PROTECO

Aprendiendo del error

SE APRENDE A BASE DE EQUIVOCARSE



ERRORES

- Errores de Sintaxis
- Errores semánticos
- Errores de ejecución

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.

Errores de Ejecución

Causa y Consecuencia.

Python muestra excepciones “pre-construidas”.

¿Cómo nos adelantamos al error?

Creando estados de “EXCEPCION” que sigan ejecutándose aunque ocurra un error

Excepciones

Bloque de código que nos permitirá continuar con la ejecución del programa aunque exista un error

Manejar Excepciones

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

```
try:
    #Captura cualquier error dentro de un bloque de instrucciones
except:
    #Definir que hacer cuando el error ocurra o
else:
    #Definir el código que se ejecutará si no ocurre ningun error
finally:
    #Se ejecuta al final, haya o no un error
```

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.

Excepciones Múltiples

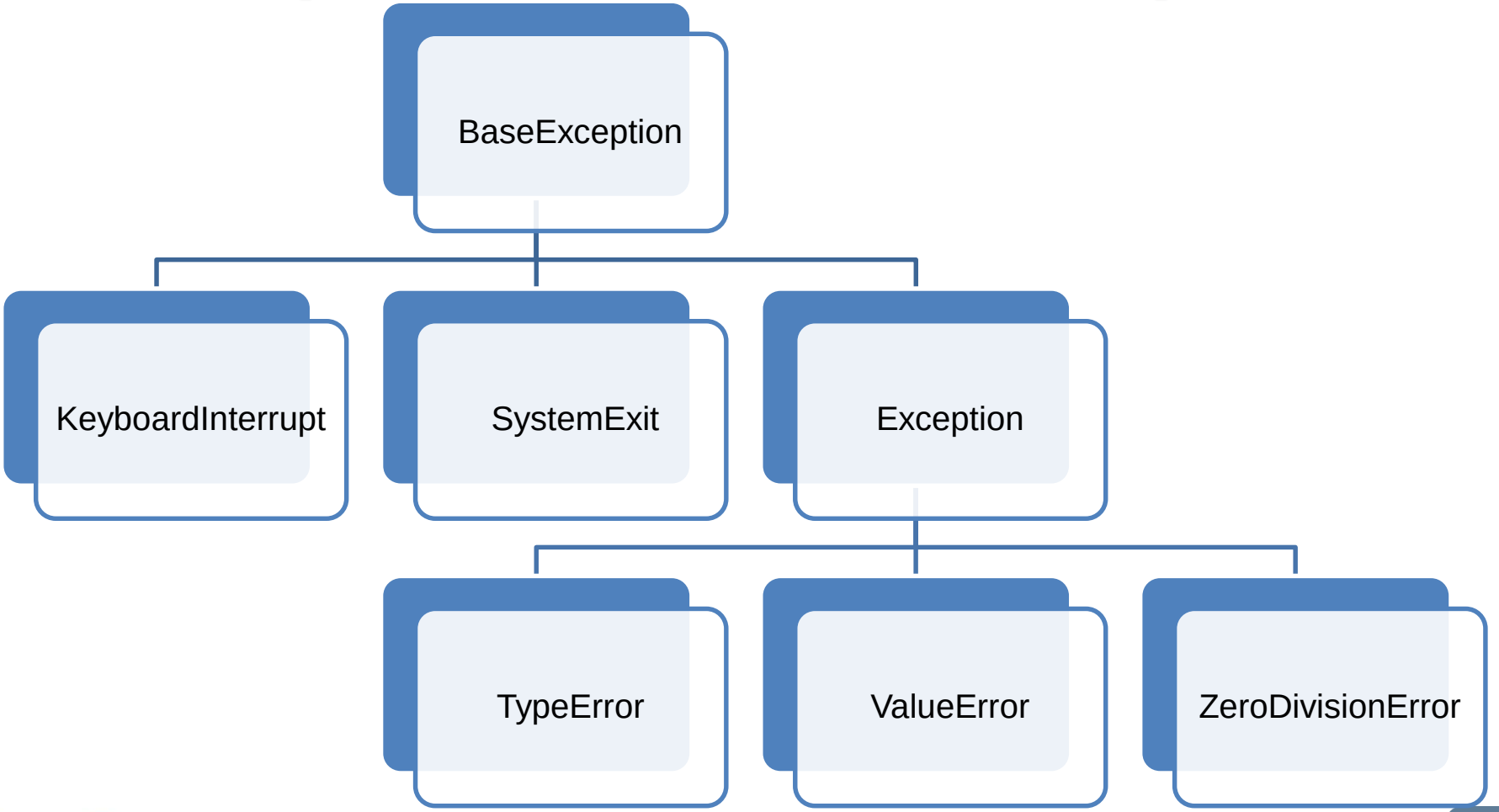
Es posible definir distintas excepciones, gracias a que cuando ocurre un error dentro del try, cada excepción tiene su propio identificador, si lo capturamos y lo guardamos en una variable podríamos manejarlos independientemente.

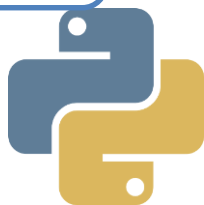
```
try:
    # código que pueda llegar a generar (levantar) una excepción.
except nombreDeLaExcepción:
    #se encarga de capturar la excepción y nos da la oportunidad de
    #procesarla mostrando #por ejemplo un mensaje adecuado al usuario.
except IOError:
    # entrará aquí en caso que se haya producido
    # 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
finally:
    #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.
```

Raise

Es posible, por otra parte, que luego de realizar algún procesamiento particular del caso se quiera que la excepción se propague hacia la función que había invocado a la función actual. Para hacer esto Python nos brinda la instrucción raise.

Jerarquía de las Excepciones





Mis Excepciones

También es interesante comentar que como programadores podemos crear y lanzar nuestras propias excepciones. Basta crear una clase que herede de Exception o cualquiera de sus hijas y lanzarla con raise.