

Agencia de
Aprendizaje
a lo largo
de la vida

DJANGO

Clase 6

Python – Excepciones

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 05

Python – Herencia

- Clases y objetos, constructores, variables de instancia y de clase
- Visibilidad de atributos (público y privado)
- Generalización, herencia simple y múltiple
- Polimorfismo
- Clase abstractas

Clase 06

Python – Excepciones

- Manejo de excepciones
- Árbol de herencia de las excepciones
- Excepciones personalizadas
- Lanzando excepciones
- Buenas prácticas en el manejo de excepciones

¿Qué es una Excepción?

Los errores detectados durante la ejecución del programa se llaman
EXCEPCIONES



Si una excepción ocurre en algún lugar de nuestro programa y no es capturada en ese punto, va subiendo (burbujeando)



Hasta que es capturada en alguna función que ha hecho la llamada. Si en toda la «pila» de llamadas no existe un control de la excepción el programa se parará

Excepciones en Python



try:

Ejecutar este código

except:

Ejecutar este código solo **SI** arriba ocurre alguna **excepción**

else:

Ejecutar este código solo si **NO** ocurre alguna **excepción**

finally:

Ejecutar **SIEMPRE** este código

Excepciones

```
def mostrar_division_entera(dividendo, divisor):  
    try:  
        print("Intentando hacer la división")  
        resultado = dividendo // divisor  
        print(f"El resultado entero de la división es: {resultado}")  
    except TypeError:  
        print('Revisar los operandos hay un dato mal cargado...')  
    except ZeroDivisionError:  
        print('No se puede dividir por cero...')  
    except Exception:  
        print('Algo anduvo mal...')  
    else:  
        print("Este programa nunca falla..")  
    finally:  
        print('El super programa ha finalizado..')
```

Herencia de excepciones

Todas las excepciones en Python deben ser instancias de una clase que derive de BaseException

Si se definen nuevas excepciones, se recomienda a los programadores derivar de Exception

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
    +-- StopIteration
    +-- StopAsyncIteration
    +-- ArithmeticError
        +-- FloatingPointError
        +-- OverflowError
        +-- ZeroDivisionError
    +-- AssertionError
    +-- AttributeError
    +-- BufferError
    +-- EOFError
    +-- ImportError
        +-- ModuleNotFoundError
    +-- LookupError
        +-- IndexError
        +-- KeyError
    +-- MemoryError
    +-- NameError
        +-- UnboundLocalError
    +-- OSError
```



Solo son algunas

Excepciones personalizadas

```
class DivisorNegativoError(Exception):  
    """Excepción lanzada se divide por números negativos"""  
    pass
```

Debe estar definida
antes de ser utilizada


Se suele crear un módulo
específico para las excepciones de
negocio

Palabra reservada para
lanzar una excepción

```
def mostrar_division_entera(dividendo, divisor):  
    try:  
        if divisor < 0:  
            raise DivisorNegativoError("Mandaron un número negativo")  
        print("Intentando hacer la división")  
        resultado = dividendo // divisor  
        print(f"El resultado entero de la división es: {resultado}")  
    except TypeError:  
        print('Revisar los operandos hay un dato mal cargado...')  
    except ZeroDivisionError:  
        print('No se puede dividir por cero...')  
    except Exception as ex:  
        print(f'Algo anduvo mal: {ex}')  
    else:  
        print("Este programa nunca falla..")  
    finally:  
        print('El super programa ha finalizado..')
```

Aserciones

```
def mostrar_division_entera(dividendo, divisor):  
    try:  
        assert divisor >= 0, "Mandaron un número negativo"
```



Si el divisor es menor a 0 lanza la excepción **AssertionError** con el mensaje “Mandaron un número negativo”

Buenas prácticas en el manejo de excepciones

**No te olvides de completar la
asistencia y consultar dudas**

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.

TODO EN EL AULA VIRTUAL