

# Clase 2\_parte2 : Programación

Excepciones

# Temario

- Excepciones
- Control de excepciones.  
Manejadores



Excepciones.

# ¿Qué es una excepción?

Una **excepción** es un acontecimiento, que ocurre durante la ejecución de un programa, que **interrumpe** el flujo normal de las instrucciones de programa.

– Ejemplo de excepciones:

- Abrir un archivo que no existe.
- Acceder en una lista a un valor que no existe.
- Invocar a un método que no existe.
- Referirse a una variable que no existe.
- Mezclar tipos de datos sin convertirlos previamente.
- Etc.

# Ejemplo de excepciones “sin manejar o gestionar”

```
>>> lista = [0,1,2]
>>> lista[2]
2
>>> lista[3]

Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    lista[3]
IndexError: list index out of range
```

Ej. de excepción por tratar de **acceder a un índice no existente** de la lista.

**IndexError**: nombre de la excepción.

```
>>> print m

Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    print m
NameError: name 'm' is not defined
```

Ej. de excepción por **variable no definida.**

**NameError**: nombre de la excepción.

```
>>> open('pepe.txt','r')

Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    open('pepe.txt','r')
IOError: [Errno 2] No such file or directory: 'pepe.txt'
```

Ej. de excepción por **no existir un archivo.**

**IOError**: nombre de la excepción.

# Manejo de excepciones

- Una excepción no controlada finaliza inmediatamente con la ejecución del programa.
- Python, como otros lenguajes, provee un mecanismo para "capturar" excepciones y "hacer algo" cuando estas ocurren.
- El poder "capturar" una excepción le permite al programador "manejar" la situación para que el programa siga ejecutándose con normalidad.

# Manejo de excepciones

```
lista = [0,1,2,3]

for x in range(1,6):
    print(lista[x])

print(lista)
```

```
>>>
1
2
3

Traceback (most recent call last):
  File "C:\Users\Matias\Desktop\app 2012\test.py", line 4, in <module>
    print lista[x]
IndexError: list index out of range
>>> |
```

```
lista = [0,1,2,3]

try:
    for x in range(1,6):
        print(lista[x])
except (IndexError):
    lista.append("NUEVO")

print(lista)
```

**Constructor**  
para manejo  
excepciones

Salida con manejo de excepciones por el programador. **SIN ERROR!**

***“Crea un nuevo elemento en la lista”***

```
>>>
1
2
3
[0, 1, 2, 3, 'NUEVO']
```

# Manejo de excepciones

La estructura que presenta Python para manejo de excepciones es la siguiente:

**try:**

**sentencia 1**

    ....

**sentencia n**

**except** nombre de la excepción 1:

**sentencias**

.....

**except** nombre de la excepción n:

**sentencias**

**else:** **sentencias**

**finally:**

**sentencias**



# Manejo de excepciones

Dentro del try van las sentencias que pueden producir la excepción

**try:**

**sentencia 1**

**....**

**sentencia n**

**except** nombre de la excepción 1:  
**sentencias**

**..**

**except** nombre de la excepción n:  
**sentencias**

**else:**

**sentencias**

**finally:**

**sentencias**

Opcionales



# Manejo de Excepciones

El bloque except permite capturar y manejar las excepciones que ocurren en el bloque try.

**try:**

**sentencia 1**

....

**sentencia n**

**except** nombre de la excepción 1:  
**sentencias**

..  
**except** nombre de la excepción n:  
**sentencias**

**else:**

**sentencias**

**finally:**

**sentencias**

Puede haber tantos bloques except como se necesiten, uno para cada tipo de excepción que se pueda producir pero ejecuta uno solo.

# Manejo de excepciones

Un conjunto de excepciones pueden ser manejadas por un mismo manejador. En ese caso se puede colocar:

**except**  
(exp1,exp2,...):

Puede aparecer un except **SIN nombre** de excepción, pero **SOLO** al final. Actúa como comodín

**except:**

**try:**

sentencia 1

....

sentencia n

**except** nombre de la excepción 1:  
sentencias

.....  
**except :**  
sentencias

**else:**  
sentencias

**finally:**  
sentencias

# Manejo de excepciones

## else:

- El código colocado en la cláusula **else** se ejecuta **solo si** no se levanta o produce una excepción

## finally:


- El código colocado en la cláusula **finally** se ejecuta **siempre,** se haya o no levantado una excepción

# Ejemplo con else

```
1 lista = [0,1,2,3]
2
3 try:
4     for x in range(1,6):
5         print(lista[x])
6     except(IndexError):
7         lista.append("NUEVO")
8 else:
9     print("Se ejecutó el bloque try de forma correcta!")
10
11 print(lista)
```

Se levanta la excepción..

Cláusula opcional, que se ejecuta **SOLO** si **NO** se levanta excepción en el bloque try except




```
>>>
1
2
3
[0, 1, 2, 3, 'NUEVO']
>>>
```

❑ NO se ejecuta el bloque else

```
1 lista = [0,1,2,3]
2
3 try:
4     for x in range(1,4):
5         print(lista[x])
6     except(IndexError):
7         lista.append("NUEVO")
8 else:
9     print("Se ejecutó el bloque try de forma correcta!")
10
11 print(lista)
```

NO se levanta la excepción..



```
>>>
1
2
3
Se ejecutó el bloque try en forma correcta!!
[0, 1, 2, 3]
>>>
```

❑ SI se ejecuta el bloque else

# Ejemplo con finally

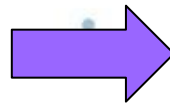
```
def dividir(x,y):  
    try:  
        resultado = x / y  
        print("se terminó la ejecución.")  
    except(ZeroDivisionError):  
        print("ERROR! División por cero.")  
    else:  
        print("El resultado es: ", resultado)  
    finally:  
        print("Ejecutando la clausula finally!!")
```

Cláusula opcional, pero que permite que se ejecute una serie de sentencias **SIEMPRE**, ya sea que el proceso haya o no terminado bien

**NO se levanta** la excepción..

```
12 #invocando función  
13 dividir(8,4)
```

```
se terminó la ejecución.  
El resultado es:  2.0  
Ejecutando la clausula finally!!
```



```
>>>
```

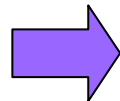
```
Se terminó la ejecución del módulo sin excepción  
el resultado es 2  
ejecutando la clausula finally
```

**SI levanta** la excepción..

**SE ejecuta** el bloque finally

```
1 dividir(8,0)
```

```
ERROR! División por cero.  
Ejecutando la clausula finally!!
```



```
>>>
```

```
;division por cero!  
ejecutando la clausula finally  
>>>
```

**SE ejecuta** el bloque finally

# Algunas excepciones predefinidas

**ZeroDivisionError (ArithmeticError):** Lanzada cuando el segundo argumento de una operación de división o módulo, era 0.

**EOFError (StandardError):** Se intentó leer más allá del final de fichero.

**IOError (EnvironmentError):** Error en una operación de entrada/salida..

**ImportError (StandardError):** No se encuentra el módulo o el elemento del módulo que se quería importar.

**IndexError (LookupError):** El índice de la secuencia está fuera del rango posible.

**KeyError (LookupError):** La clave no existe.

**MemoryError (StandardError):** Memoria disponible insuficiente.

# Algunas de excepciones predefinidas

**NameError (StandardError):** No se encontró ningún elemento con ese nombre. Suele lanzarse cuando se usan variables no declaradas (se puede deber a errores de tipeo).

**RuntimeError (StandardError):** Error en tiempo de ejecución no especificado.

**NotImplementedError (RuntimeError):** Ese método o función no está implementado.

**SyntaxError (StandardError):** Clase padre para los errores sintácticos.

**IndentationError (SyntaxError):** Error en la indentación del archivo.

**TypeError (StandardError):** Tipo de argumento no apropiado.

**ValueError (StandardError):** Valor del argumento no apropiado.

.....