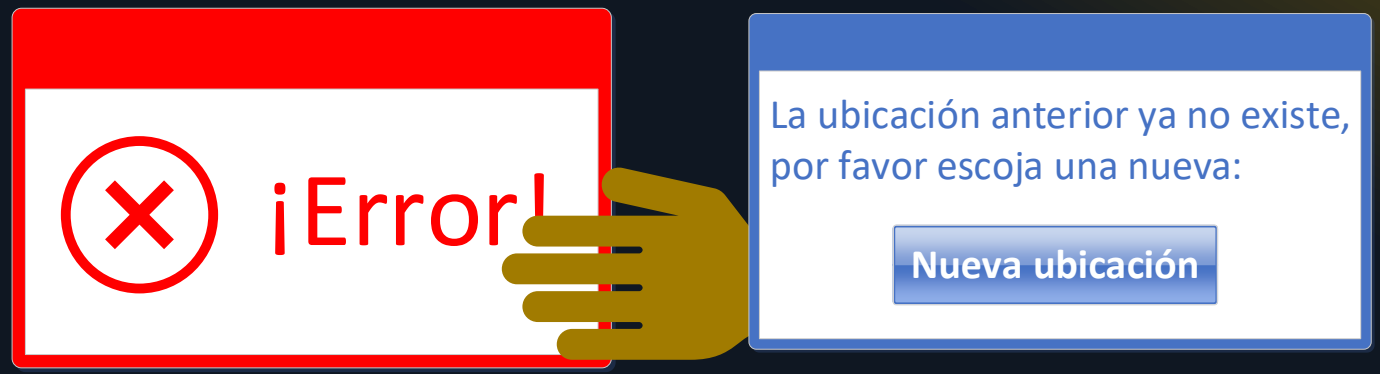
Curso de Python

7 – Excepciones

Ramón Invarato Menéndez

Ricardo Moya García



Exception (Excepción)

Consola

```
Traceback (most recent call last):
  File "C:/directorio/mi_script_de_python.py", line 185, in <module>
    with open("Fichero_que_no_existe.txt") as f:
FileNotFoundError: [Errno 2] No such file or directory: 'Fichero_que_no_existe.txt'
```

- Las excepciones **son errores o problemas en tiempo de ejecución**
- **Cuando ocurre algo inesperado se lanza una excepción** (por ejemplo: cuando queremos acceder a una posición de listado que no existe se lanza un IndexError, ya que el programa no sabe que hacer en ese caso)
- Las excepciones **se pueden capturar** para hacer algo con ellas (por ejemplo: al guardar un documento, si no existe previamente el fichero se lanzará una excepción IOException que podrá ser capturada para -en vez de dar error en el programa- preguntar al usuario que dónde quiere guardar el documento). Entonces, **el programa puede continuar** sin problemas
- Si la **excepción no se captura, entonces el flujo de ejecución se interrumpe**. En este caso, **se mostrará información asociada** a la excepción (el número de línea donde ocurrió el problema, la estructura del programa en ese momento, el nombre del error para consultarlo en alguna documentación, textos puestos por el desarrollador, etc.)

try

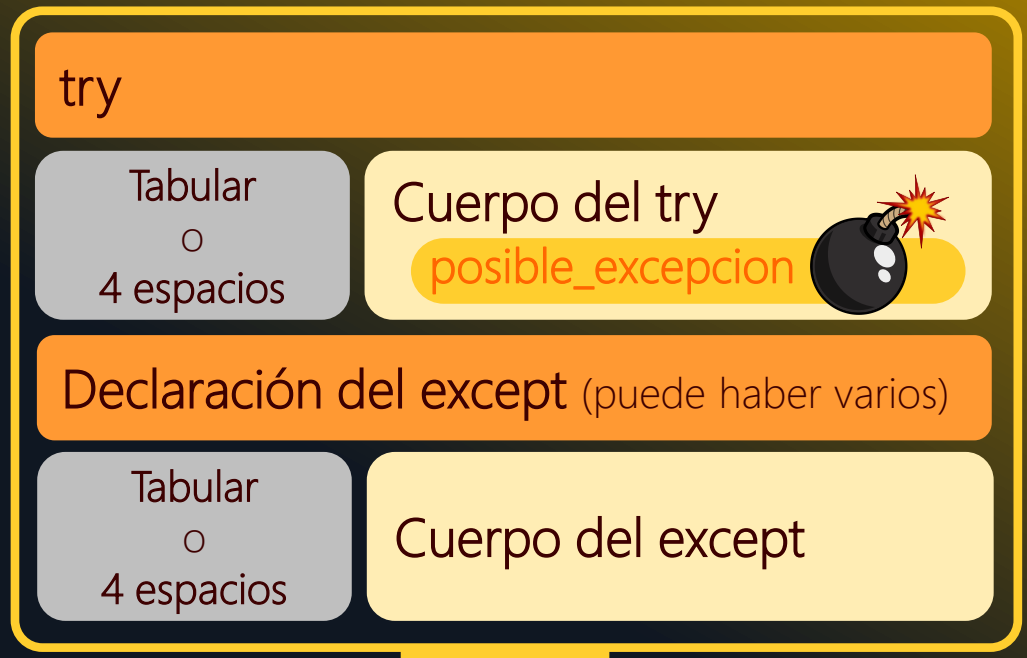
except excepcion

Captu… (try … exc…

- Try: código que se va a i… todo o una parte
- Except: Excepción a capturar, ya que dentro del cuerpo del try pueden lanzarse diferentes excepciones (ejemplo: si leemos un fichero que se guarda en un listado es posible que se lance una excepción IOError por el fichero e IndexError por el listado). La excepción de "except" se puede utilizar si la seguimos de un "as" y el nombre que queramos. Se puede utilizar "Exception" para capturar todas la excepciones (aunque no se recomienda). Se puede capturar y dar continuidad a la excepción con "raise" vacío

Pseudo-código

```
try:
    print("Cuerpo del try")
except Exception as mi_error:
    print(mi_error)
    raise
```



Pseudo-código

```
try:
    print("Cuerpo del try")
except Exception1:
    print("Cuerpo de la Excepción capturada 1")
except Exception2:
    print("Cuerpo de la Excepción capturada 2")
```

Capturar una excepción (Finally)

- **Finally**: Algo que **se va ejecutar siempre**, se capture excepción o no. Es útil si es usan estructuras de control de flujo (return, continue, break), pues finally se ejecuta sí o sí antes de estas.

Pseudo-código

```
try:
    print("Cuerpo del try")
except Exception1:
    print("Cuerpo de la Excepción capturada 1")
except Exception2:
    print("Cuerpo de la Excepción capturada 2")
finally:
    print("Cuerpo del finally")
```

Capturar una excepción (Else)

- Else: Código a ejecutar en caso de no haber capturado ninguna excepción (por ejemplo: no se ha lanzado ninguna). Evita capturar accidentalmente excepciones que no queramos proteger (por ejemplo: si abrimos un fichero podremos querer capturar esa excepción IOError para hacer algo con ella; pero no querremos capturar accidentalmente la excepción IOError que pueda dar el cierre del fichero, para en caso de dar se deje pasar).

Pseudo-código

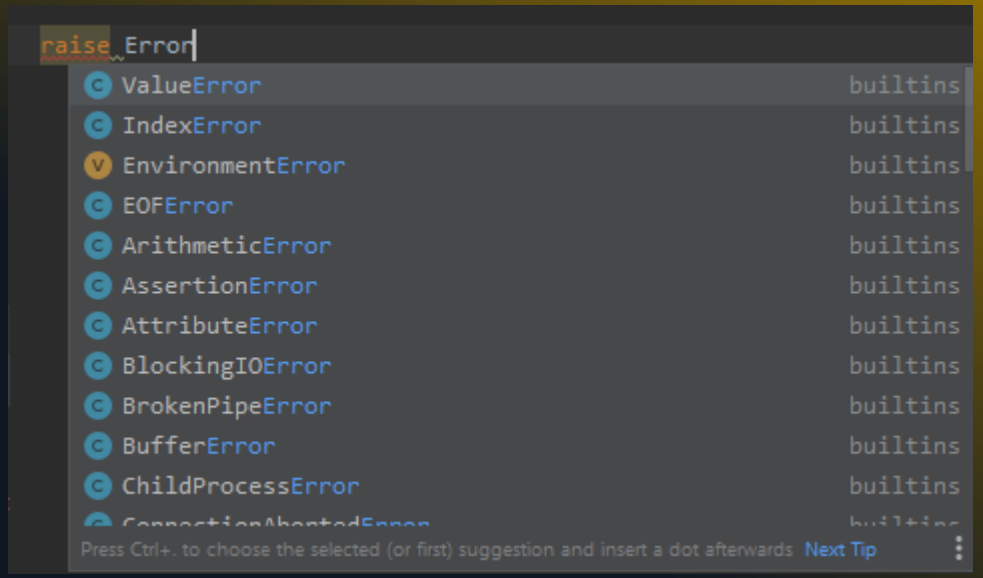
```
try:
    print("Cuerpo del try")
except Exception1:
    print("Cuerpo de la Excepción capturada 1")
except Exception2:
    print("Cuerpo de la Excepción capturada 2")
else:
    print("Cuerpo del else")
finally:
    print("Cuerpo del finally")
```

https://docs.python.org/3/library/exceptions.html#bltin-exceptions

Lanzar una Excepción (raise)

raise Excepcion("Mi mensaje de error")

- Se puede **lanzar una excepción con "raise"**
- Podremos crear nuestras propias excepciones, pero lo más sencillo (y lo más normal) es **utilizar una de las muchas excepciones ya creadas**: ValueError, IndexError, KeyError, ArithmeticError, AttributeError
- **Podremos añadir un mensaje de error personalizado a todas nuestras excepciones**; algunas necesitarán valores extra para crear la excepción
- Si se lanza **dentro de un bucle actuará como break y dentro de una función como return** (vacío)



* Un truco para conocer rápidamente las Excepciones ya creadas en un lenguaje de programación es escribir "Error" o "Exception" en un IDE y que nos muestre el listado

Pseudo-código

```
raise Exception("Mi mensaje de error")
```

Código

```
diccionario = {
    "casa": "house",
    "coche": "car",
    "lápiz": "pencil"
}
```

Truco dict: Usar valor si existe en dict, sino añadir valor y usar

Código

```
clave = "casa"
...
```

print

Consola

```
{
'casa': 'house is big',
'coche': 'car',
'lápiz': 'pencil'
}
```

- En vez de comprobar si tiene la clave, podremos intentar obtenerla, si la obtenemos la modificamos, sino la obtenemos se lanzará la excepción KeyError que trataremos para añadir la nueva clave y el valor

Código

```
try:
    valor = diccionario[clave]
    nuevo_valor = valor + " is big"
except KeyError:
    nuevo_valor = "new value"

diccionario[clave] = nuevo_valor
```

Código

```
clave = "new_value"
...
```

print

Consola

```
{
'casa': 'house is big',
'coche': 'car',
'lápiz': 'pencil' ,
'new_value': 'new value'
}
```

¡GRACIAS!

Web: **https://jarroba.com/**

Ramón Invarato Menéndez

Linked-in
https://www.linkedin.com/in/rinvarato/

Github
https://github.com/Invarato

Ricardo Moya García

Linked-in
https://www.linkedin.com/in/phdricardomoya

Github
https://github.com/RicardoMoya