HINTS

USO DE RAISE

También podemos ser nosotros los que levantemos o lancemos una excepción. Tal como ***ZeroDivisionError*** o ***NameError***, usando ***raise***. La sintaxis es muy fácil:

```
raise ZeroDivisionError("Información de la excepción")
```

O podemos lanzar otra de tipo ***NameError***.

```
raise NameError("Información de la excepción")
```

Se puede ver como el **String** que hemos pasado se imprime a continuación de la excepción. Se puede llamar también sin ningún parámetro, de la siguiente forma:

```
raise ZeroDivisionError
```

Visto esto, ya sabemos cómo una excepción puede ser lanzada. Existen tres maneras:

Si hacemos una operación que no puede ser realizada (como dividir por cero).

- En este caso, Python se encarga de lanzar automáticamente la excepción.
- O también podemos lanzar nosotros una excepción manualmente, usando ***raise***.
- Habría un tercer caso que sería lanzar una excepción que no pertenece a las definidas por defecto en Python. En ese caso, deberíamos definir nuestra propia excepción.

USO DE CLASE EXCEPTION

Si queremos crear una excepción, solamente tenemos que crear una clase que herede de la clase **Exception**. Es tan sencillo como el siguiente ejemplo:

```
# Creamos una excepción personalizada
class MiExcepcionPersonalizada(Exception):
    pass
```

Y ya podríamos lanzarla con ***raise*** cuando quisiéramos:

```
raise MiExcepcionPersonalizada()
```

También se pueden pasar parámetros de entrada al lanzarla. Esto es muy útil para dar información adicional a la excepción. En el siguiente ejemplo se pasan dos parámetros.

Para ello tenemos que modificar la función **__init__()**, de la siguiente manera:

```
# Creamos nuestra propia excepción heredando
# de la clase Exception
class MiExcepcionPersonalizada(Exception):
    def __init__(self, parametro1, parametro2):
        self.parametro1 = parametro1
        self.parametro2 = parametro2
```

Y una vez hemos creado nuestra excepción, podemos lanzarla con ***raise*** como ya hemos visto.

También es posible acceder a los parámetros pasados como argumentos al lanzar la excepción.

```
# Lanzamos con raise la excepción que hemos creado
try:
    raise MiExcepcionPersonalizada("ValorPar1", "ValorPar2")
except MiExcepcionPersonalizada as ex:
    p1, p2 = ex.args
    print(type(ex))
    print("parametro1 =", p1)
    print("parametro2 =", p2)

#<class '__main__.MiExcepcionPersonalizada'>
#parametro1 = ValorPar1
#parametro2 = ValorPar2
## Ejemplos Hay un truco muy interesante que nos permite pasar a la
excepción un argumento en forma de diccionario como se muestra a
continuación.

# Se define una excepción
class MiExcepcion(Exception):
    pass

# Se lanza
try:
    raise MiExcepcion({"mensaje":"Mi Mensaje", "informacion":"Mi
Informacion"})

# Se captura
except MiExcepcion as e:
    detalles = e.args[0]
    print(detalles)
    print(detalles["mensaje"])
    print(detalles["informacion"])

#{'mensaje': 'Este es el mensaje', 'informacion': 'Esto es la
informacion'}
# Mi Mensaje
# Mi Informacion
```

Como se puede ver, los parámetros son accesibles con [] ya que se trata de un diccionario.

Una forma un poco más larga de hacer lo mismo sería de la siguiente manera. En este caso, los

parámetros que se pasan no son accesibles como si fueran un diccionario, sino como de un objeto normal que se tratase con `.mensaje` e `.informacion`

```
class MiExcepcion(Exception):
    def __init__(self, mensaje, informacion):
        self.mensaje = mensaje
        self.informacion = informacion

try:
    raise MiExcepcion("Mi Mensaje", "Mi Informacion")
except MiExcepcion as e:
    print(e.mensaje)
    print(e.informacion)
```

Nótese que para los ejemplos hemos usado mensaje e información, pero estas variables pueden ser las que se te ocurran, y por supuesto tampoco tienen porque ser dos, podrían ser más.

USO DE FINALLY

A los ya vistos bloques ***try***, ***except*** y ***else***, podemos añadir un bloque más: el ***finally***. Dicho bloque se ejecuta siempre, haya o no haya habido excepción.

Este bloque se suele usar si queremos ejecutar algún tipo de acción de limpieza. Si por ejemplo estamos escribiendo datos en un fichero, pero ocurre una excepción, tal vez queramos borrar el contenido que hemos escrito con anterioridad, para no dejar datos inconsistentes en el fichero.

En el siguiente código vemos un ejemplo. Haya o no haya excepción el código que haya dentro de ***finally***, será ejecutado.

```
try:
    # Forzamos excepción
    x = 2/0
except:
    # Se entra ya que ha habido una excepción
    print("Entra en except, ha ocurrido una excepción")
finally:
    # También entra porque finally es ejecutado siempre
    print("Entra en finally, se ejecuta el bloque finally")

#Entra en except, ha ocurrido una excepción
#Entra en finally, se ejecuta el bloque finally
```

EJEMPLO DE EXCEPCIONES CON FICHEROS

Un ejemplo muy típico de excepciones es en el manejo de ficheros. Se intenta abrir, pero se

captura una posible excepción. De hecho, si entras en la documentación de open, se indica que **OSError** es lanzada si el fichero no puede ser abierto.

```
# Se intenta abrir un fichero y se captura una posible excepción
try:
    with open('fichero.txt') as file:
        read_data = file.read()
except:
    # Se entra aquí si no pudo ser abierto
    print('No se pudo abrir')
```

Como ya hemos comentado, en el ***except*** también se puede capturar una excepción concreta. Dependiendo de nuestro programa, tal vez queramos tratar de manera distinta diferentes tipos de excepciones, por lo que es una buena práctica especificar qué tipo de excepción estamos tratando.

```
# Se intenta abrir un fichero y se captura una posible excepción
try:
    with open('fichero.txt') as file:
        read_data = file.read()
# Capturamos una excepción concreta
except OSError:
    print('OSError. No se pudo abrir')
```

En este otro ejemplo vemos el uso de los bloques ***try***, ***except***, ***else*** y ***finally***, todos juntos.

```
try:
    # Se fuerza excepción
    x = 2/0
except:
    print("Entra en except, ha ocurrido una excepción")
else:
    print("Entra en el else, no ha ocurrido ninguna excepción")
finally:
    print("Entra en finally, se ejecuta el bloque finally")
```

También se puede capturar una excepción de tipo **SyntaxError**, que hace referencia a errores de sintaxis. Sin embargo, el código debería estar libre de este tipo de fallos, por lo que tal vez nunca deberías usar esto.

```
try:
    print("Hola"))
except SyntaxError:
    print("Hay un error de sintaxis")
```