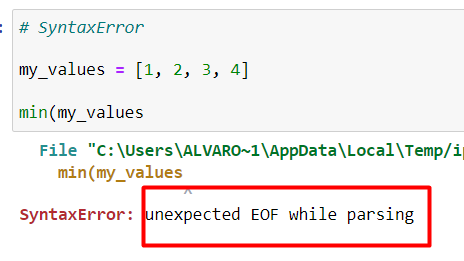
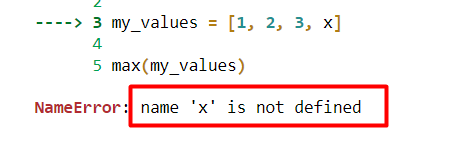
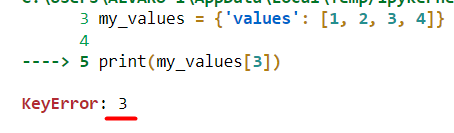
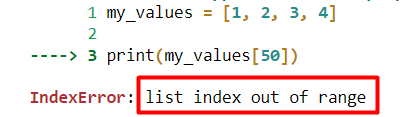
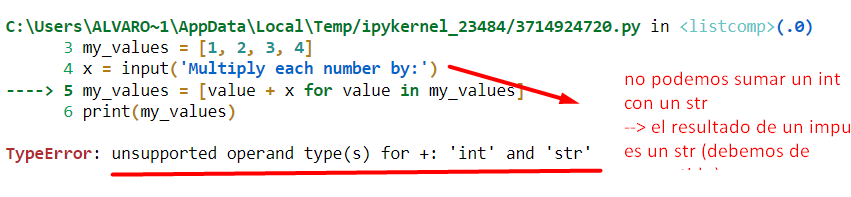
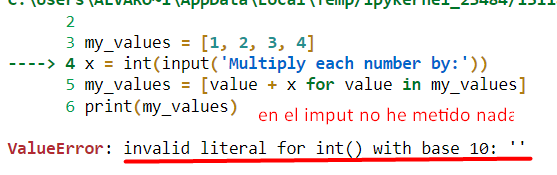
**Error Handling**

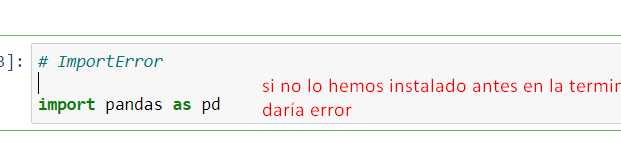
→ PYTHON\_ERROR\_HANDLING

## **Tipos de error de Python**

## Hay muchos tipos de errores en Python, pero a continuación se muestran algunos tipos de errores comunes que es probable que encuentre.

* **SyntaxError:** cuando el código se ha escrito incorrectamente.
* **AttributeError:** cuando intenta acceder a un atributo en un **objeto (variable) que no existe**. 
* **KeyError:** cuando intenta acceder a un key en un diccionario que no existe.
* **IndexError:** cuando intenta acceder a un valor de una lista que no existe.
* **TypeError:** cuando un argumento de una función no es del tipo correcto (por ejemplo, un str en lugar de int).
* **ValueError:** cuando un argumento de una función es del tipo correcto pero no está en el dominio correcto (por ejemplo, una cadena vacía)



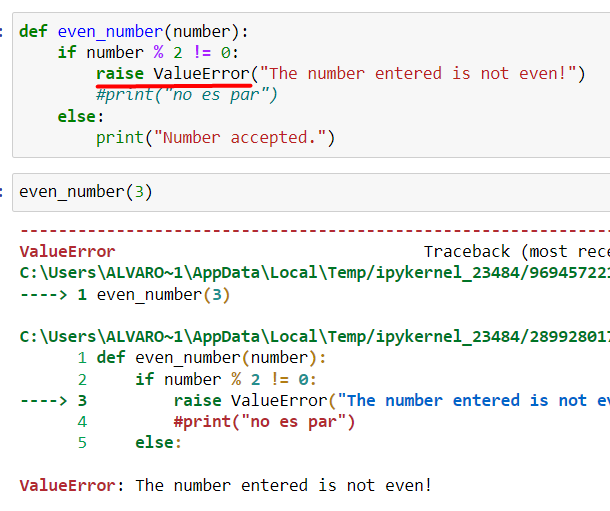
* **ImportError:** cuando falla una importación (ejemplo: import pandas as pd, pero sin haber instalado la librería pandas en su entorno de jupyter)
* **IOError:** cuando Python no puede acceder correctamente a un archivo en el disco.

Además de informarle el tipo de error, Python también le proporcionará un mensaje de error que le indicará específicamente por qué está recibiendo el error.

**Writing Exceptions (raise errors)*→ generar el error para avisar***

Podemos y debemos en muchas ocasiones generar condicionales (if y else) para que, si el usuario introduce un error o nosotros podemos cometer un error, el código siga corriendo. Pero en ocasiones **nos interesa indicar que hay un error** y **comunicárselo al usuario**, para que inserte el código nuevo otra vez.

→ a través de **raise ValueError(“....”)**:para avisar de que hay un error aunque el código no lo diese (ejemplo en un if si no se cumple la condición pasaría al else, pero podemos poner ahí el raise para que SÍ que se pare y avisar)



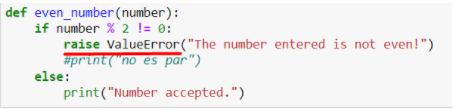
→ **pass**: para continuar o indicar al else “que no haga nada”. Esto se puede utilizar en cualquier condicionante, haya o no haya error.

**Catching Exceptions *→ saltarnos el error para que el código continue***

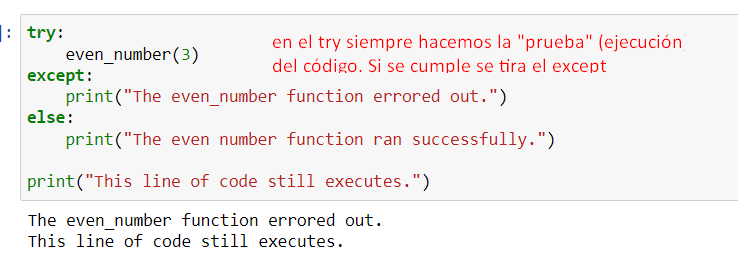
Periódicamente, sabremos que puede ocurrir una excepción y nuestro código va a producir un error, pero **no queremos que el error bloquee todo nuestro código**.



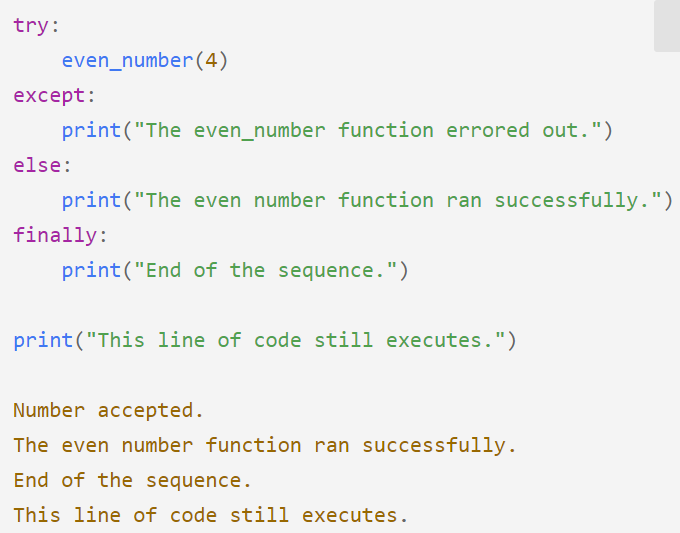
* El try bloque le permite probar un bloque de código en busca de errores.
* El except bloque le permite manejar el error.
* El else bloque te permite ejecutar código cuando no hay ningún error.
* El finally bloque le permite ejecutar código, independientemente del resultado de los bloques de prueba y excepción.

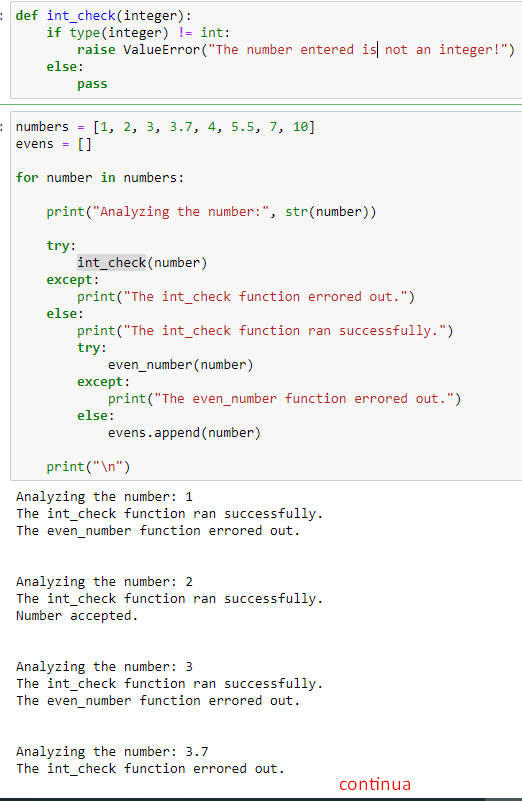


1. se cumple el error, debería de saltar el raise ValueError, pero no salta porque a continuación el try hace su función y ejecuta el except:



1. no se cumple el error. 1º se ejecuta el primer else y el else y finally de acontinuación:



**otro ejemplo:**