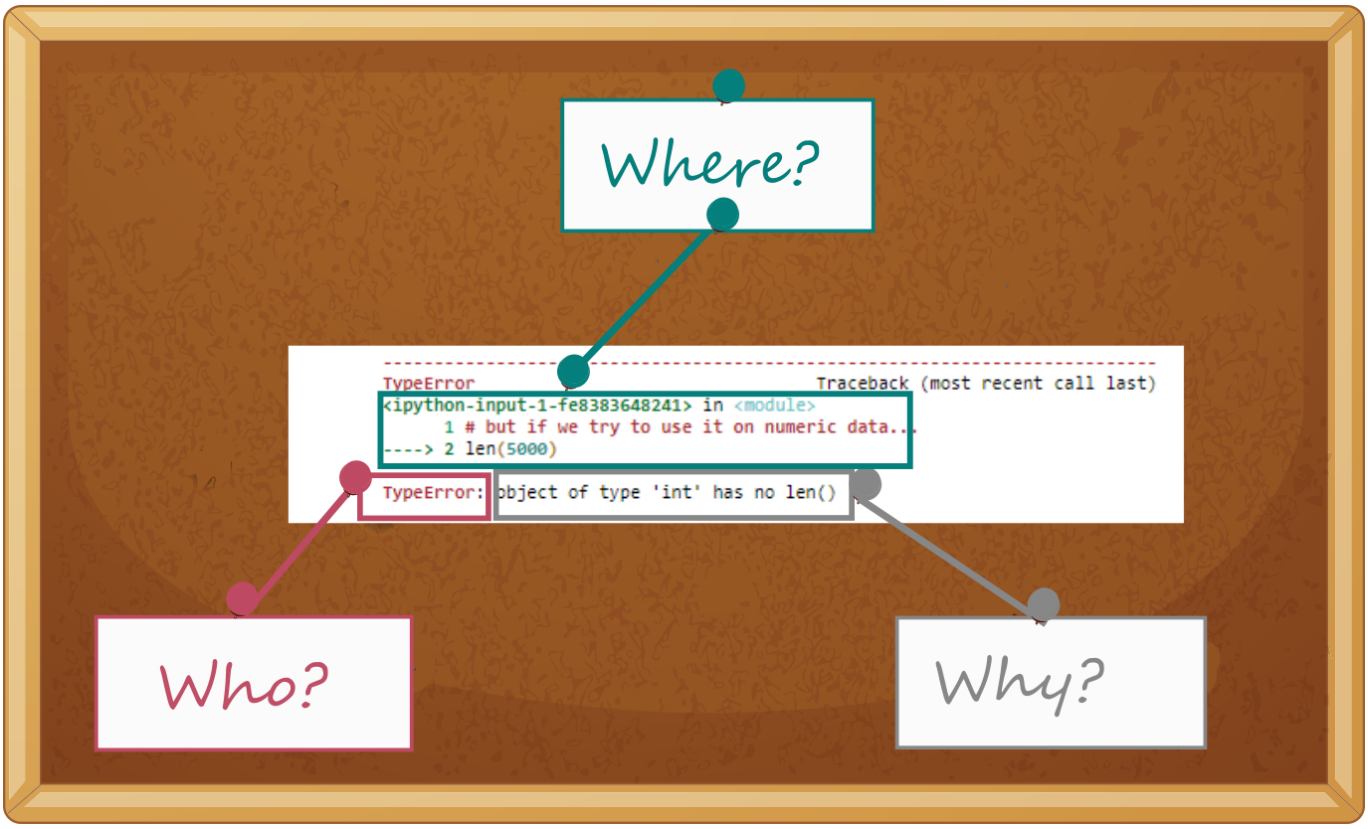
Reading errors like a Detective!

Every time an error occurs in your script, a crime has been committed, and every crime can be solved, by looking at the clues!

We even made a crime drama about it! Watch it here!

Python error messages tell you three key things in order to solve the crime:

- ***Where*** the error occured
- ***Who*** committed the crime (Wat type of error occured)
- and an error message explaining ***why*** the error occured



Background vector created by brgfx - www.freepik.com

Usually, copying and pasting the error message (the why) into google will get you the help you need, and you just need to go back to the scene of the crime (or a few steps before) to fix it. However, the type of error can also help find the culprit.

The usual suspects:

- `TypeError` This error occurs when an object of a certain type, gets used in a function that can't accept that type for whatever reason, this is a good time to use a coercion function before the crime occurs.
- `ValueError` This occurs when an input to a function is of the right type, but is the wrong value of that type. One example would be trying to divide by 0.

- `NameError` This means that you have used a name of a variable or function that Python doesn't know about, meaning you misspelled it(as I often do), tried to use a variable before assigning the data to it, or used a function before importing it. Don't worry its an easy fix.
- `SyntaxError` this means that the rules for using punctuation marks in python have been violated (happens to me all the time, my mortal enemy!) It usually means that a `:` has been forgotten, or there are too many or too few `,`'s, or an operator (like `+`) is used in the wrong place. if the error message says EOF, or EOL, this means an end of function or end of line error, and means that one of these: `[,]`, `{,}`, `(,)`, `',"` is missing it's other half.
- `IndentationError` Usually means there is a space or a tab accidentally at the beginning of a line of code, in advanced python programming, with loops and if statemetns, the code blocks are separated by tabs, so if that is done improperly this error will also occur.
- `IndexError` Usually when getting something out of a list, or a python object that is 'list like' and the number index is higher than the length of the list
- `KeyError` Usually happens when getting something out of a dictionary or something that is like a dictionary (ie a Pandas Dataframe) and the key doesn't exist (usually happens when I spell the key wrong 😛)

Silent/Semantic errors

- These occur when Python has still *worked* - it ran your code, and gave you a valid result
- but it hasn't done what you intended it to do.