SECTION 14: ADVANCED PYTHON MODULES, 2 hours 23 minutes, 13 sections
6/13 Python Debugger

- **-> errors in your code**
  - -> printing out the result to find the error
  - -> the debugger tool in Python
  - -> exploring variables within mid-operation of the Python code

- **-> in the .ipynb file**
  - example error and how you would debug it with print statements vs the Python debugger tool
  - x = [1,2,3]
  - y= 2
  - z = 3
  - result = y + z
  - result2 = x + y
  - -> she's deliberately written code which returns an error (adding an integer to a list)
  - -> the error it returns is a concatenation error
  - -> you can try printing out y e.g -> this returns a type error
  - -> **using print functions to try and understand where the error is**
  - **-> in other words, we have an error message, and we're trying to figure out in the code where it comes from**

  - **using the debugger tool**
    - **import pdb <- Python debugger**
    - **-> this pauses operations mid script and allows us to play with them while the code is executed**
    - **-> type errors report which line the error happened on**
    - **-> so she's setting a trace before the error line**
      - **pdb.set_trace(), then ran the code again**
      - **-> before it hits the line with the error message, you can explore and call variables at this point in time -> the code is asking for an input**
      - -> she is printing out the values of the different variables
      - **-> (pdb) is the Python debugger <- it is to allow developers to see mid-operation where the errors are by checking what the values of the different variables are at that point in time**
      - -> **to quit the debugger you type in q -> there is documentation for the debugger**
        - -> the idea is - you set the trace where the code tells you the error message is