An unfinished OOP School system has two classes: a "Person" base class and a "Student" sub class. You should create two new classes, "Teacher" and "Sixth_Form_Student". "Teacher" will be based on "Person", but will have additional properties such as salary (amount the teacher earns) and the subjects they teach e.g. "Computer Science", "IT" etc. "Sixth_Form_Student" will be based on "Student" but will have additional properties such as a list of the A Levels being studied (see below):

Person Class (base Class)

- name of the person
- person's age
- "M" for male, "F" for female "O" Other

Student_Class (Sub Class Inherits from person +)

- name of school

Sixth_Form_Student_Class (Sub Sub Class Inherits from Student Class +)

- List of subjects

Teacher_Class (Sub Class Inherits from Person +)

- Salary
- List of subjects taught

Write an OOP Python Program to model the above scenario with the following functionality:

(Part A – Minimum standard for all to achieve)

1). Add a teacher (to a list of teacher objects)
2). Add a 6th form Student (to a list of student objects)
3). List all the teachers (with a count of how many – class variable)
4). List all the 6th form students (with a count of how many –class variable)
5). Search for a subject : print all teachers who teach it and all 6th form students taking it
6). Delete a teacher (drop from the list – you'll also need to amend option 3)
7). Delete a 6th form student (drop from the list – you'll also need to amend option 4)
8). Save all current teachers to an external binary file (use pickle example)
9). Save all current students teachers to an external binary file (use pickle example)
10). When the Program Starts, load all teachers and students (from their binary files) and set the Class variables (object counts) to their correct values
11). Add other Sub Classes of Person for additional adults who may work at the school (your choice) – Add the relevant functionality for these new people: (1-10 above – you decide)
12). Extend the Student Class so it also copes with Year Group, Tutor Group and Behaviour (Good, Average, Bad) – Add the relevant functionality for these new attributes (1-10 above – you decide) **Structure your program as you have been shown : separate .py files for all classes and all classes should have docstrings. Import everything into the main program as required. Test each class as you go.**

Improving your program **(Part B)**

- **Before you start the improvements, your program should be 100% finished and do everything exactly as required in points 1-12 above (essential)**
- **You could at this point chose to make the 1st part of your Voiceover Screen Capture Video proving 1-12 above**
- From now on everything should now be run and tested in the Console
- Try to break your program (using test data) everywhere user input occurs; think about :
    - Valid: test data that represents typical input into the system
    - Valid extreme: test data that is valid but at the extreme end of the range of acceptable input
    - Invalid: test data that is out of range and should be rejected
    - Invalid extreme: test data that is invalid, but only just at the limit of being unacceptable
    - Erroneous: test data that should be rejected because it is the wrong type of input
- Add validation to all places user input occurs to cope with the above
- Add validation everywhere else you can think of: Start-up (state of files), Exit (state of files)?
- Lock user input down to known choices (numbered lists are the best)
- Remove all "freestyle" input from the user : It only invites errors and leads to more complex validation coding being required
- Remember GIGO – Garbage in Garbage Out – Far better to prevent poor data being input than allow it into your program trying to deal with it programmatically. Lock it out up front
- Keep users input to the absolute minimum (think speed, simplicity and reducing user error)
- Make sure instructions and feedback (output) from a user's actions are as clear, concise and friendly as possible so they know exactly what's going on and where they are (no blind alleys)
- F1 Key – Help?
- Add any info that will make the program easier to use, current file state, num. records of each person etc.
- Structure your program correctly. See *Vehicles Exercise* as an example
- Add Docstrings to **all** classes and functions
- Add a detailed Comment Box to your main program (you could also duplicate the comment box info in a Docstring). It should include details of exactly what it does and all imported modules it includes + anything else you think will be of use
- "Seed" your bin file/s with a rich mixture of data (use Excel, Internet) – Optionally write modules to do this for you (see next point)
- Put different versions of your bin file/s in a folder so you can swap in/out to main program as required (during runtime)
- Always consider the cosmetics (the look and feel of the program) – Is it consistent?
- Always try to think about your program from an inexperienced user's perspective – Would they say it was quick, efficient and easy to use?

If you finish everything early and want more to do and enhance your projected even further, then add a GUI (use Guizero as it's the simplest quickest way to do this). However adding a GUI will be a **significant** piece of additional work over and above what you've already done. So - do everything including the testing proof video below (first) and save all the "console" programs first. Take copies of everything to work on your GUI, then if it goes wrong you can always abandon it and submit your original "console" programs.

Proving your submission works

Plan / Practice this first and time yourself! Make a **5 minute (maximum) screen capture** were you **quickly, efficiently and clearly** run through all your functionality. **GO THRU ALL THE MARKS IN THE SPREADSHEET IN ORDER!!!** Make sure you **record an audio voice over as well very clearly** explaining what you are doing. 1st Prove your program has incorporated all essential tasks (1-12 above) which work perfectly (the bare minimum). Then showcase any other features (from Part B) which you've added. Submit **this single video (MP4 format)** with your code, see the example submission folder structure.