Practical work 1: student mark management

- Functions
  - Input functions:
    - Input number of students in a class
    - Input student information: id, name, DoB
    - Input number of courses
    - Input course information: id, name
    - Select a course, input marks for student in this course
  - Listing functions:
    - List courses
    - List students
    - Show student marks for a given course
- Push your work to corresponding forked Github repository

Practical work 2: OOP'ed student mark management

- Copy your practical work 1 to 2.student.mark.oop.py
- Make it OOP'ed
- Same functions
  - Proper attributes and methods
  - Proper encapsulation
  - Proper polymorphism
    - e.g. `.input()`, `.list()` methods
- Push your work to corresponding forked Github repository

Practical work 3: some maths and decorations

- Copy your practical work 2 to 3.student.mark.oop.math.py
- Use `math` module to round-down student scores to 1-digit decimal upon input, `floor()`
- Use `numpy` module and its `array` to
  - Add function to calculate average GPA for a given student
    - Weighted sum of credits and marks
  - Sort student list by GPA descending
- Decorate your UI with `curses` module
- Push your work to corresponding forked Github repository

Practical work 4: modularization

- Split your program 3.student.mark.oop.math.py to modules and packages in a new `pw4` directory
  - `input.py`: module for input
  - `output.py`: module for `curses` output
  - `domains`: package for classes
  - `main.py`: main script for coordination
- Push your work to corresponding forked Github repository

Practical work 5: persistent info

- Before closing your program
  - Select a compression method
  - Compress all files aboves into `students.dat`
- Upon starting your program,
  - Check if `students.dat` exists
  - If yes, decompress and load data from it
- Push your work to corresponding forked Github repository

Practical work 6: `pickled` management system

- Copy your `pw5` directory into `pw6` directory
- Upgrade the persistence feature of your system to use `pickle` instead, still with compression
- Push your work to corresponding forked Github repository

Practical work 8: multithreaded management system

- Copy your `pw6` directory into `pw8` directory
- Upgrade the persistence feature of your system to use `pickle` **in background thread**, still with compression
- Push your work to corresponding forked Github repository

Practical work 7: Python shell

- Create a new python program, name it «7.shell.py»
- Make a shell
  - User inputs command
  - Shell executes the command, print output
  - Support IO redirection
    - input from file to process
    - output from process to file
    - e.g. input from one process being output of another

Practical work 7: Python shell

- Run it and test some commands
  - `ls -la`
  - `ls -la > out.txt`
  - `bc < input.txt`
  - `ps aux | grep term`
- Push your work to corresponding forked Github repository

Practical work 9: GUI'ed management system

- Copy your `pw8` directory to `pw9` directory
- Upgrade your user interface to GUI using Tkinter
- Push your work to corresponding forked Github repository