219114 In-class final project

Task 1:

Create a directory called task1 and put all your code for this task in this directory.

Implement a class Polynomial in poly.py so that you can run it with run_poly.py. The sample output is given in the file task1_result.txt. Your output formats do not have to match exactly with the sample output, but the output values do. Do not forget to include docstring for each method in the class. You might need to use the OO_complex module when implementing a method to find the roots of a polynomial object.

More details about the Polynomial class can be found in "Lecture for week 13" post in the course's Google Classroom. You may use poly_template.py as a guide to implement poly.py.

Task 2:

Create a directory called task2 and put all your code for this task in this directory. In addition to the python code, this task requires you to submit a report detailing your design, implementation, final output, and any known bugs encountered.

You are to design and implement a multi-player blackjack game that builds on top of the assignment in Exercise 8. The design and implementation have to be done in OO style. **No procedural-stye code will be accepted.**

In a report file (named it report.pdf), describe your high-level design, i.e., how many classes will be implemented, what are the representations and methods in those classes, preferably in the form of a UML diagram. In addition, explain functionality for each method (what each method does).

Once you are done with your design, start implementing your multi-player blackjack.

- Add an additional scoring feature where if a player wins his score gets incremented by 1, if he looses, the score gets decremented by 1, otherwise no effect on the score. All players start with the score value of zero.

- The dealer (computer) also has his score and it is calculated the number of wins minus the number of losses. For example, if there are 3 players and the dealer wins against 2 and looses against 1, his score for this round is 2 - 1 = 1.

- The dealer must come up with an optimal strategy if he wants to stay (if already above the minimum stay threshold) or draw more cards. If his total score will be positive when he stays, he will do so. Otherwise, he will have to decide if he wants to draw more cards and win/loose more. Describe your dealer strategy and put it in the report and explain the code that implement your strategy.

- A sample session with this multi-player blackjack is given in the file task2_sample_output.txt There is also a file players_info.txt containing information of players. In your output, you can replace Player 0, Player 1, … with names of players from players_info.txt.

Once you are done with your implementation, include a sample output of your interactive session in your report. Explain if you are able to finish this task completely if not describe the difficulties you face. Include any know bugs in your report if there are any.

In addition to all the code files and the report file, you must include a README file explaining the details of each file you put in this directory. Also, explain how to run your program here.

Task 3:

Create a directory called task3 and put all your code for this task in this directory. (You may need to copy task2 directory to task 3 first.)

Extend task2 such that it works with data of players.

File players_info.txt contains name, gender, hometown region, budget, number of wins and number of losses of 100 players. Budget is how much money each player has.

See sample output of task 3 in task3_sample_output.txt.

In task3_sample_output.txt, in main menu, there are 4 menu choices (excluding Quit program).

Choice 1 is to play blackjack game as in task2. **However,** in task3, edit your blackjack code, so it

- Works with budget of each player. Each time a player or a dealer opens a card, he/she must put 100 Baht into the central money. The central money will be accumulated as the players and the dealer opens the card. At the end of each round, the winner will take the central money. If there are multiple winners, the central money will be equally split among the winners. If a player has no money left in budget, he/she can no longer play. Assume that the dealer has infinite money.
- Update number of wins and losses for each player

Choice 2 is to add more budget of each specific player.
Choice 3 is to view information (or profile) of a specific player.
Choice 4 is to view statistics of players. The examples of statistics are: top 5 players with maximum number of wins, top region male players with maximum number of wins. (see more stat choices in sample output.)

Besides what are in sample output, feel free to add more statistics to view. Instead of showing text output, you can design and save more output as graph. For example, percentage of players from each region.

Once you are done with task 3, write a report (named it task3_report.pdf). Include a sample in your report. Explain if you are able to finish this task completely. If not, describe the difficulties you face. Include any know bugs in your report if there are any. If you have more menu choices or more stat choices, besides what are in task3_sample_output.txt, include explanation of the added features as well.

In addition to all the code files and the report file, you must include a README file explaining the details of each file you put in this directory. Also, explain how to run your program here