COMP 1001 (Fall 2024): Practice Problem Set 4

Suggested Completion Date: Oct 27, 2024

Important Notes:

- Practice Problem Sets are not a part of the evaluation for the course.
- After the Suggested Completion Date, a rubric and solution outline will be posted on the course Brightspace shell so you can check your answers.
- Completing the Problem Sets will help you to gain hands-on experience with coding in Python, and will help you apply the concepts we have covered to solve and code solutions for the given problems. This will help you prepare for your tests and exam.
- While coding solutions for the problems given below, keep in mind that on the test/exam you will also be marked on the following:
  - *Efficient solution of the problem.* A given problem can be solved in a number of different ways, but the best solution is the one that is efficient; i.e., the one that uses the right concepts in a very productive way.
  - *Including sufficient descriptive comments in your program.* The person looking at your code should be able to understand how your code is solving the given problem even if the person reading your Python program does not know the Python language. In addition, the reader of your program should be able to understand what each variable represents.
  - *Labelling of input and output.* All input and output should have a descriptive label so that the reader of your program understands what input is expected and what output the program has generated.
  - *Program style* - consistent formatting and indentation of program statements, meaningful variable names (identifiers), and the use of constants (constant identifiers), where appropriate.

  Practicing these rules will build a good foundation for programming.
- This Problem Set is based on Chapter 6 of the textbook, without the graphics components. Please use only concepts from Chapters 1–6 of the textbook.

1. A school has 100 lockers and 100 students. All lockers are closed on the first day of school. As the students enter, the first student, denoted `S1`, opens every locker. then the second student, `S2`, begins with the second locker, denoted `L2`, and closes every second locker. Student `S3` begins with the third locker and changes every third locker (closes it if it was open, and opens it if it was closed). Student `S4` begins with locker `L4` and changes every fourth locker. Student `S5` starts with `L5` and changes every fifth locker, and so on, until student `S100` changes `L100`. Write a Python program, in a file called `lockers.py`, to determine and output the list of lockers that are open after all the students have passed through the building and changed the lockers. (Hint: Use a list of 100 Boolean elements, ie. `True` or `False`)

2. The Department of Computer Science would like to know how the 5 new courses (CS101, CS105, CS110, CS115 and CS120), and the students in those courses, are doing. For each course, the Department Head would like to know:

   - the range of marks achieved for each course (the difference between the highest and lowest marks); and
   - the average marks obtained by each student in all five courses.

   Given two one dimensional lists, `courses` and `students`, and a two dimensional list/table, `marks`, write a Python program, in a file called `newCoursesInfo.py`, to provide the desired information.

   Your program should include the following functions:

   - `initializeMarks` which, given the number of courses (`numC`) and the number of students (`numS`), initializes and returns a two dimensional $numC \times numS$ list of randomly generated integers between `0` and `100`, inclusive (i.e., the minimum and maximum possible mark values for each course).
   - `computeAllRanges` which, given the `courses` and `marks` lists, uses the following functions to compute and print the range of marks for each of the courses:
     - `findMinForRow` which, given the `marks` table and the row number, finds and returns the minimum mark for that row (ie. for a single course).
     - `findMaxForRow` which, given the `marks` table and the row number, finds and returns the maximum mark for that row (ie. for a single course).
     - `printRangeForCourse` which, given the `courses` list, the position of the course to print, and the maximum and minimum values, prints the name of the course and the range of the values in that course.
   - `computeAllAverages` which, given the `students` and `marks` lists, computes and prints each student's name and the average mark for that student.
   - a `main` function that declares and initializes the students and courses lists and tests all of the above functions.

   For example, given the following initial data (marks randomly generated as above):

```
students = ["Andy Pandy","Benny Menny", "Kim Simms","Rolly Polly",
            "Cindy Mindy","Geeta Peeta"]
courses = ["CS101", "CS105", "CS110", "CS115", "CS120"]
marks = [[41, 48, 89, 52, 76, 14],
         [96, 4, 88, 58, 70, 73],
         [84, 78, 99, 18, 57, 26],
         [5, 19, 77, 92, 93, 90],
         [31, 12, 100, 67, 76, 81]]
```

the output should be:

```
Courses    Range of Marks
  CS101          75
  CS105          92
  CS110          81
  CS115          88
  CS120          88

Student Name    Average Marks
  Andy Pandy           51.4
  Benny Menny          32.2
  Kim Simms            90.6
  Rolly Polly          57.4
  Cindy Mindy          74.4
  Geeta Peeta          56.8
```

For testing, you do not necessarily need to use user input to obtain the `courses` and `students`, you may include those lists in your `main` function as literals if you wish. The `marks` list must be generated using the `initialMarks` function, with the sizes of the other two lists passed as parameters.