Fall 2020
Due Nov 16th

Computer Science 1001
Problem Set #7

Section 001/002

Important Notes:

- Even though this, or any other subsequent, Problem Set is only graded based on whether or not a reasonable attempt was made for each problem, it is a good idea to complete each Problem Set by the due date and upload it to the appropriate dropbox in the course Brightspace shell to get credit for completing the Problem Set and see if you have understood the required concepts.
- While working on your Problem Set you can get help by posting your questions in the appropriate folder in the discussion forum. Note that sometimes, even though your code can generate the expected output, it may still not be correct as it may work for a specific data set and not for all valid data sets. Also, it may not use the programming concepts and best practices that we have emphasized in the course.
- Completing the Problem Sets will help you to gain hands-on experience with coding in Python and understand the learnt concepts well enough so that you can apply the concepts to solve and code the solution for the given problems. All of this will help you in doing well on your tests and exam.
- The files that you upload to the dropbox should be your source code (`.py`) files, as practiced in Lab 0, and any other requested solution files.
- 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; ie., 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 10 of the textbook, without the graphics components. Please use only concepts from Chapters 1–6, 9–10, and 12 of the textbook.

- Rubrics/solution outlines for each Problem Set will be provided after the grades for the Problem Set have been released.

Full solutions will not be posted, however you may get help to complete your Problem Set if the rubrics/solution outlines are insufficient.

---

1. In a file called `courses.py`, do the following:

   - Create a superclass called `Course` to represent a university course containing attributes for course title and course ID number. Provide accessor methods for each attribute. Include a `display` method to output the course information in the following format:

     ```
     Title: Introduction to Programming
     ID: 83713
     ```

   - Create a class called `OfferedCourse` that is a subclass of the `Course` class and includes the additional attribute of a list of IDs of students enrolled. Provide a method to return the number of students enrolled in the course. Include a method called `addStudent` to add a student to the course, provided the student is not already registered, as well as a corresponding `dropStudent` method (provided the student is registered). Also include a `display` method that overrides the `Course display` method to output the course information in the following format:

     ```
     Title: Introduction to Programming
     ID: 83713
     Enrolment: 56
     ```

     Note that the `display` method in the `OfferedCourse` class should call the `display` method of its superclass.

   - Create a `StudentCourse` class that is a subclass of the `Course` class and includes the additional attribute of `grade`. Provide an accessor method for `grade`. Include a `display` method that overrides the `Course display` function to output the course information in the following format:

     ```
     Title: Introduction to Programming
     ID: 83713
     Grade: 88
     ```

     Note that the `display` method in the `StudentCourse` class should call the `display` method of its superclass.

   - Write a `main` function to create instances of each class created above and call the `display` method on each object created.