COMP 1001 (Fall 2024): Practice Problem Set 6

Suggested Completion Date: Nov 18, 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 9 and 10, without the graphics components. Please use only concepts from Chapters 1–6, 9, 10, 12 of the textbook.

1. A complex number is a number of the form $a + bi$, where a and b are real numbers and i is $\sqrt{-1}$. The numbers a and b are known as the real part and the imaginary part of the complex number, respectively.

   Addition, subtraction, multiplication, and division for complex numbers are defined as follows:

$$\begin{aligned}(a+bi)+(c+di) &= (a+c)+(b+d)i\\ (a+bi)-(c+di) &= (a-c)+(b-d)i\\ (a+bi)*(c+di) &= (ac-bd)+(bc+ad)i\\ (a+bi)/(c+di) &= \frac{(ac+bd)}{(c^2+d^2)}+\frac{(bc-ad)}{(c^2+d^2)}i\end{aligned}$$

   The absolute value for a complex number is given by:

$$|a+bi| = \sqrt{a^2+b^2}$$

   A complex number can be interpreted as a point on a plane by identifying the (a, b) values as the coordinates of the point. The absolute value of the complex number corresponds to the distance of the point to the origin.

   Python has the `complex` class for performing complex number arithmetic. Here, you will design and implement your own class to represent complex numbers.

   In a file called `complex.py`, create a class named `Complex` for representing complex numbers, including overriding special methods for addition, subtraction, multiplication, division, and absolute value. Also include a `__str__` method that returns a string representation for a complex number as `(a+bi)`. If `b` is `0`, `__str__` should simply return `a`.

   Provide a constructor `Complex(a,b)` to create a complex number $a + bi$ with the default values of `0` for a and b. Also provide `getRealPart()` and `getImaginaryPart()` methods for returning the real and imaginary parts of the complex number, respectively.

   Write a `main` function to test your `Complex` class by prompting the user to enter the real and imaginary parts of two complex numbers and displaying the results of calling each method.

2. 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.