Fall 2020
Due Nov 9th

Computer Science 1001
Problem Set #6

Section 001/002

Important Notes:

- **For this Problem set you should submit a solution to ONE of the TWO questions.**
- 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 9 of the textbook, without the graphics components. Please use only concepts from Chapters 1–6, 9, 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. (a) In a file called `music.py`, create a `Song` class that contains the following:
   - The private data fields `name`, `minutes`, `seconds` for the name of a song and its length in minutes and seconds
   - A constructor with arguments for `name`, `time` (both strings). The `time` string shoud be in the format `mm:ss`.
   - Accessor methods for `name`, `minutes`, and `seconds`.

   (b) Add to your `music.py` file to create an `Album` class that contains the following:
   - The private data fields `title`, `artist`, and `date` (all strings) for the title of the album, the name of the artist, and the release date (year), as well as a list of `Song`s (in data field `songs`).
   - A constructor with arguments for `title`, `artist`, and `date`.
   - Accessor methods for `title`, `artist`, `date`, and `songs`.
   - A method to add a `Song` to the `Album`.
   - A method to calculate the total length (time) of the `Album` in the format `hh:mm:ss`.

   (c) Write a `main` function, in `music.py`, to illustrate the use of your `Album` and `Song` classes.
   Sample input/output:

```
Enter name of album: Album1
Enter the name of the artist: Artist1
Enter the release date: 2018
How many songs? 3
Enter name of song: Song1
Enter the length of the song (mm:ss): 03:43
Enter name of song: Song2
Enter the length of the song (mm:ss): 04:02
Enter name of song: Song3
Enter the length of the song (mm:ss): 02:58
Album length (hh:mm:ss) is 00:10:43
```

2. 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.