Chapter 10 Lab: Classes

Goal: This lab is intended to help you understand classes. You should hand in at least one *.py* file for each of the below problems. Please name and number the problems appropriately (something like *Lab10_1_YourName.py*). Each code should contain extensive comments in the style discussed in class and output should be clearly displayed and labeled.

1. Create a *Student* class consisting of the following data members: first name, last name, site, year (SO,JR,SR), and GPA. You should have a constructor which sets all data (it should be passed into the constructor, not prompted via input). You should have three class functions that perform each operation, correspondingly:

   - Changes the year
   - Changes the GPA
   - Returns the following information for a student nicely in a string (should be a *__str__* method)
     - Full name
     - Site
     - Year
     - GPA

   Test your class by creating two instances of the class with different information. Print both Students to the screen. Something like:

   ```
   Name: Ben Neb
   Site : GSSM
   Year: JR
   GPA: 3.23

   Name: Deb Bed
   Site : Liberty
   Year: SO
   GPA: 2.55
   ```

2. Continue your previous problem, but create a new class, called *Roster*. The roster should have the following data elements:

   - *subject* : A course subject (CS)
   - *number* : The course number
   - *name* : Course name
   - *professor* : Professor's name
   - *students* : A list of type Student that contains all of the students in the class

   And functions:

   - *__init__* : Sets each of the above data elements, in order
   - *removeStudent* : Takes in an instance of type Student. If the first and last name of the inputted student matches one of the elements in the *students* list, then deletes that student from the *students* list.
   - *addStudent* : Takes in an instance of type student, adds them to the end of the *students* list.
   - *__str__* : Prints the course subject, number, name, professor, and the name and site for each student. Something like:

     ```
     CS 110: Intro to Programming
     Professor: Dr. Dostert
     Roster:
      First Last Site
      Ben   Neb  GSSM
      Deb   Bed  Liberty
      Les   Sel  Byrnes
      Liz   Zil  LTC
     ```

In your main code, do the following:

- Create a list of type Student which contains the name and made-up information for four people.
- Using the newly created list, create an instance of the *Roster* class for CS 110.
- Print out the roster
- Remove a student from the class
- Print the roster
- Add a student to the class (a different one than what you just removed)
- Print the roster

3. Create the *RationalNumber* class discussed in the lectures. You should have data members:

   - num: the numerator
   - den: the denominator

   Corresponding to the rational number *self.num/self.denom* . Create function members:

   - __init__: Takes in a numerator and denominator. Sets the values of *self.num* and *self.den*
   - mult: Multiplies the current class number to the inputted number
     I'll give you *just* this one:
     ```
     def mult(self,rationalIn):
         return RationalNumber(self.num*rationalIn.num, self.den*rationalIn.den)
     ```
   - divide: Divides the current class number by the inputted number (classNumber/inputtedNumber)
   - add: Adds the current class number by the inputted number
   - subtract: Subtracts the current class number by the inputted number (classNumber - inputtedNumber)
   - __str__: Prints out the fraction nicely as: self.num/self.denom (eg 5/8)

   Create two rational numbers $\frac{3}{7}$ and $\frac{4}{9}$. Test each of your functions. Your output should be:

```
3/7 + 4/9 = 55/63
3/7 - 4/9 = -1/63
3/7 * 4/9 = 4/21
( 3/7 ) / ( 4/9 ) = 27/28
```

4. Create a *TacoShopItem* class which contains items for your Accelerate Taco Shop. Each item should have a name, description, per unit cost, and total number in inventory (this is not how many is ordered, but how many of each are available in the shop at a given time). Include:

   - A constructor (__init__): Sets each initial value for each item.
   - A function that reduces the number of units by one, when the item is ordered.
   - A function that increases the number of units by any amount specified (for when something gets remade).
   - An *__str__* method which formats a single item, in the format shown below:

     | Taco | Beef, tomato, cheese | $0.89 | 20 |

   Store the inventory as a list of *TacoShopItem*. Create each item as discussed in lab 3 (you may just have 'drink' as an item). Create a function which takes in the list of *TacoShopItem* and prints the store inventory nicely, like this:

Taco Shop Inventory			
Type	Description	Units	Per Unit Cost
Taco	Beef, tomato, cheese in crispy tortilla	$0.89	20
Soft Taco	Beef, tomato, cheese in soft tortilla	$0.99	15
.	.	.	.
Drink	RC, Dr. Pepper, 7UP or tea	$0.99	100

   In the 'main' code:

1. Print the inventory.
2. Create an order (at least 5 items, but anything you want)
3. Print the order to the screen
4. Print out the new inventory with the appropriate ingredients reduced

Notes:

- You do not have to re-code an ordering system. Simply display the specific 5 item (or more) order, then reduce the inventory correspondingly.
- You are more than welcome to 'hard code' locations (it's fine if you use the fact that *myInventory[0]* corresponds to, say, a taco while *myInventory[5]* corresponds to drink or something like that).