ENGR 101 - Introduction to Programming
Study Questions - Week 13

1. Define a class that will represent a circle, having the radius as an attribute. Add two methods: area and perimeter that will compute the area and perimeter respectively.

2. Define a class **Student** that has three attributes: id, name, and a dictionary of courses by semester (e.g. {*'Semester4':{'PHYS101':3, 'CS 100':1*}} Note: the values for course codes are credits.). Add a method that takes an argument like 'SemesterX' as input (X can be 1-8) and prints all the courses student took in that semester along with total number of credits that semester.

```
>> student1 = Student(21899129, "Alper", {"Semester1":{"PHYS 101":3, "MATH
101":3}})

>> student1.semester_history("Semester1")

   PHYS 101

   MATH101

   Total credits in Semester1: 6

>> student1.semester_history("Semester2")

   Error
```

3. Design a Course class that has course code, course name, room, instructor, and students attributes, where students attribute is a dictionary of Student objects (from question 2) with student id as the key, and the corresponding Student object as the value. Add a method, get_attendance_list, that will print the class list (i.e., on each line, it should print a student name and id) sorted by student name. Add another method, students_sharing_course, with input parameter course_name. This new method will print the list of students (name and id) who share a given course. The output should be sorted by student id.

4. Design an Inventory class which has the following attributes and methods:

Attributes: productName and currentCount

Methods: FullfillCustomerOrder and AddProductIntoInventory

> AddProductIntoInventory: Return current count of the product after adding x products to the inventory
>
> FullfillCustomerOrder: Return the current count of the product remaining after fulfilling customer order for x products

Your class should have a __str__ method as well.

5. Write a Club class which has a list (initially empty) of member names, club name, and club web site as its attributes. Include a method in this class, called add_member, which would take a first name and last name of a person as an input. This methold should first concatenate the first and last names, and then add the full name to the member list. You should initialize your list inside the init method.

6. Write a class StudentGPAList which has a dictionary of GPAs (key: student name, value: gpa as a float), department name, and a letter grade conversion dictionary (key: letter grade, value: a tuple with min and max boundaries for that letter) as its attributes. Your class should have init, sum, avg, and convert_to_letter_grade methods which would convert the grade of a student to the corresponding letter grade by using the last attribute in the above list.

7. Add new methods min and max to the above class which will compute and return a list of the students with minimum and maximum GPA , respectively.

8. Have you ever wondered how the vending machines are programmed? Define a class **VendingMachine** that can be initialized with three attributes: number of sales, total sales revenue, and a dictionary of items with structure {itemID:[price, quantity]}. Add a method **fetchItem** that takes an item ID and cash as inputs and returns the corresponding item from the dictionary of items (if cash >= price). This method should also update the number of sales and total sales revenue attributes. Also, print appropriate error messages if cash is less than the price or the vending machine is out of the item requested.

   Note: To keep things simple, don't worry about change. We can assume cash <= price, and if a transaction fails, the user takes back the cash and retries.

```
>>   mymachine   =   VendingMachine({"Item1":[1.50,   10],   "Item2":[2.50,   5],
"Item3":[3.50,4]})
>> mymachine.fetchItem("Item1",1.50)

   Please take your Item1.
```

9. Extend the class in the above question and add a method analytics that takes no input. When called, this method should print item ID, quantity sold and sales generated for all types of items that the machine has sold (each item on a separate line). Note that the fetchItem method results in a sale and you should modify it as well for this question.

```
>> mymachine.analytics()

/   Item: Item1   Quantity Sold:  1  Sales Volume:  1.5
```

10. Imagine a class Critic with two attributes: Critic Name and Movies Reviewed By Critic (e.g. {'Hababam Sinifi':10}). Define a method filter that takes a number n and returns all the movies with rating greater than or equal to n.

```
>>   test   =   Critic('Melih',   {'Hababam   Sinifi':10,   'Hababam   Sinifi
Tatilde':8.5, 'Hababam Sinifi Askerde':5})

>> print test.filter(7)

   ['Hababam Sinifi', 'Hababam Sinifi Tatilde']
```

11. Define a class InstantExtreme that stores integers in its data attribute (a list) using an instant_add method (takes one integer at a time). The class has two additional methods namely instant_max and instant_min that should return the maximum and minimum respectively of the numbers in the data attribute.

```
>> myds = InstantExtreme()

>> myds.instant_add(9)

>> myds.instant_add(2)

>> myds.instant_add(1)

>> print myds.instant_max()
```

```
    9
```

12. Python 2.0 does not have built-in functions for mean, mode or median. Define a class ExtendedList that can be initialized using a list and has three methods that return the above mentioned statistics.

```
>> mylist = ExtendedList([1,2,3,4,5])

>> print mylist.mean()

   3
```

13. Design a Shoe class that has attributes size, color, and price. Your Shoe class should have getSize(), getColor(), and getPrice() methods.

14. Add a adjust_price method for the Shoe class. This new method will take a percentage value, and adjust the price of the shoe by increasing its price by the given percentage. Ex:

C = Shoe(29, “green”, 40)

C.adjust_price(.5)

Print c.getPrice()

60

15.
```
class Pokemon:
       def __init__(self, name, health, attack):
           self.PokemonsName = name
           self.PokemonsHealth = health
           self.PokemonsAttack = attack
```

Create a __str__ method for pokemon class and print out all pokemons attiributes.

16. Create a hpBoost method for pokemon class, which increases the pokemon objects PokemonsHealth by the given amount(as an input).

17. Create an apBoost method for pokemon class, which increases the pokemon objects PokemonsAttack by the given amount(as an input).

18. Define a class Pet that will have “name” as its only attribute. The name’s default value will be None. Add a method called intro to this class, if the name of the pet is provided when created as an instance it should return “Hi, I am <the given name>”, if it is not provided it should say “Hi, I am a pet without a name”.

(check absolute definition from here)

19. What is the output of the following code?

```
class StrCircus:
    def __init__(self, word = 'co-operative', flag = False):
        if not flag:
            self.word = word*3
        self.word = word
```

```python
    def pretty_print(self, flag = True):
        if flag:
            print '-'.join(self.word.split("-"))
        else:
            print '\n'.join(self.word.split("-"))
test = StrCircus('bi-directional')
test.pretty_print(False)
```