Intermediate Python Programming

Description: Lists, Sets, Dictionaries and Comprehensions

Instructions

- Please follow all instructions closely. There is no penalty for going "above and beyond" the assignment requirements. However, failure to meet all required parts can affect your grade. Please consult the rubric for each section.
- Either raw Python (*.py) or Jupyter Notebook (*.ipynb) files are acceptable for this lab.
- A single file may be used for this project.

Part I

Create a Student class which will contain a dictionary and be used in lists. It inherits from object:

```
class Student(object):
```

- __init__() signature:
  ```
  def __init__(self, id, firstName, lastName, courses = None):
  ```
  - The "id", "firstName" and "lastName" parameters are to be directly assigned to member variables (ie: self.id = id)
  - The "courses" parameter is handled differently. If it is None, assign dict() to self.courses, otherwise assign courses directly to the member variable.
  - Note: The "courses" dictionary contains key/value pairs where the key is a string that is the course number (like "CSE-123") and the value is a number from 0-4.0 (represents the grade the student received).

- gpa() signature:
  ```
  def gpa(self):
  ```
  - This method calculates the cumulative grade point average for the student.
  - This can be done by looping through the courses member variable and summing the values in that dictionary then dividing by the count of items in courses.
  - If self.courses is empty, just return 0.

- addCourse() signature:
  ```
  def addCourse(self, course, score):
  ```
  - This method adds a new entry into the self.courses member variable where "course" is the item's key and "score" is its value.
  - Use assertions to ensure "score" is a numeric type and is between 0 and 4

- addCourses() signature:
  ```
  def addCourses(self, courses):
  ```
  - This method appends the given dictionary (called "courses") to the member variable self.courses. Hint: Check the dictionary update method.

- Note: This may bother you since the passed in parameter has the same name as the member variable defined in __init__. However, remember scope plays a part here. The one defined in __init__ can only be accessed using "self.courses" which allows Python to figure out which one to use.
- Use assertions to ensure "courses" is a dictionary (type dict)

- __str__():
  - This method should output a formatted string such that printing multiple Student objects will line up. Example result for 3 objects (fields shown: id, lastName, firstName, GPA, courses):

```
123456    Smith          Johnnie        3.650 CSE-101,CSE-102,CSE-201,CSE-220,CSE-325
234567    Strauss        Jamie          3.550 CSE-101,CSE-103,CSE-202,CSE-220,CSE-401
345678    O'Neill        Jack           3.250 CSE-101,CSE-102,CSE-103,CSE-104
```

- __repr__():
  - We will not directly use this method, but it should have an output similar to __str__. However, the column formatting is not necessary. You can use commas in between fields:

```
123456,Smith,Johnnie,{'CSE-101': 3.5, 'CSE-102': 3, 'CSE-201': 4, 'CSE-220': 3.75, 'CSE-325': 4}
```

- header() signature:

  `def header():`
  - This method returns a string that can used as a header for the results of the __str__ method.
  - Notice that this is a ***class method*** because there is no "self" parameter, so it will be called using Student.header().
  - Example result:

```
ID         Last Name        First Name        GPA Courses
==========================================================================================
```

Extra Credit

- Use reduce() function in the gpa() method when calculating the sum before dividing.

Scoring:

Criteria	**Points**
__init__() implementation/functionality	5
__str__() implementation/functionality	10
__repr__() implementation/functionality	5
gpa() implementation/functionality	10
addCourse() implementation/functionality/assertions	10
addCourses() implementation/functionality/assertions	10
header() implementation/functionality	5
TOTAL	**55**
EXTRA CREDIT	
reduce() used in gpa()	5

Part II

Create a list of Student objects and populate it. Then create a method to print the student list with a header.

- The list should be declared using:

```
students = []
```

- Use the following data when populating students:

id	firstName	lastName	courses
123456	Johnnie	Smith	'CSE-101': 3.50, 'CSE-102': 3.00, 'CSE-201': 4.00, 'CSE-220': 3.75, 'CSE-325': 4.00
234567	Jamie	Strauss	'CSE-101': 3.00, 'CSE-103': 3.50, 'CSE-202': 3.25, 'CSE-220': 4.00, 'CSE-401': 4.00
345678	Jack	O'Neill	'CSE-101': 2.50, 'CSE-102': 3.50, 'CSE-103': 3.00, 'CSE-104': 4.00
456789	Susie	Marks	'CSE-101': 4.00, 'CSE-103': 2.50, 'CSE-301': 3.50, 'CSE-302': 3.00, 'CSE-310': 4.00
567890	Frank	Marks	'CSE-102': 4.00, 'CSE-104': 3.50, 'CSE-201': 2.50, 'CSE-202': 3.50, 'CSE-203': 3.00
654321	Annie	Marks	'CSE-101': 4.00, 'CSE-102': 4.00, 'CSE-103': 3.50, 'CSE-201': 4.00, 'CSE-203': 4.00
456987	John	Smith	'CSE-101': 2.50, 'CSE-103': 3.00, 'CSE-210': 3.50, 'CSE-260': 4.00
987456	Judy	Smith	'CSE-102': 4.00, 'CSE-103': 4.00, 'CSE-201': 3.00, 'CSE-210': 3.50, 'CSE-310': 4.00
111354	Kelly	Williams	'CSE-101': 3.50, 'CSE-102': 3.50, 'CSE-201': 3.00, 'CSE-202': 3.50, 'CSE-203': 3.50
995511	Brad	Williams	'CSE-102': 3.00, 'CSE-110': 3.50, 'CSE-125': 3.50, 'CSE-201': 4.00, 'CSE-203': 3.00

- Create a method called printStudents() that prints the header defined in part I and prints all the students under it.
- Signature:

```
    def printStudents(students):
```

- Sample output:

```
ID        Last Name       First Name        GPA Courses
==========================================================================================
123456    Smith           Johnnie         3.650 CSE-101,CSE-102,CSE-201,CSE-220,CSE-325
234567    Strauss         Jamie           3.550 CSE-101,CSE-103,CSE-202,CSE-220,CSE-401
345678    O'Neill         Jack            3.250 CSE-101,CSE-102,CSE-103,CSE-104
456789    Marks           Susie           3.400 CSE-101,CSE-103,CSE-301,CSE-302,CSE-310
567890    Marks           Frank           3.300 CSE-102,CSE-104,CSE-201,CSE-202,CSE-203
654321    Marks           Annie           3.900 CSE-101,CSE-102,CSE-103,CSE-201,CSE-203
456987    Smith           John            3.250 CSE-101,CSE-103,CSE-210,CSE-260
987456    Smith           Judy            3.700 CSE-102,CSE-103,CSE-201,CSE-210,CSE-310
111354    Williams        Kelly           3.400 CSE-101,CSE-102,CSE-201,CSE-202,CSE-203
995511    Williams        Brad            3.400 CSE-102,CSE-110,CSE-125,CSE-201,CSE-203
```

Requirements

- At least 3 students must be created without passing an argument to __init__'s courses parameter.
  - The courses then must be added using individual calls to the student object's addCourse() method.
- At least 3 students must be created without passing an argument to __init__'s courses parameter.

    - The courses then must be added using a call to the student object’s addCourses() method using a dictionary.
- The remaining 4 student objects are to be created where the courses are passed to __init__’s courses parameter which is a dictionary containing all the courses and scores.

Scoring:

Criteria	**Points**
List population	10
printStudents() implementation/functionality	10
TOTAL	**20**

Part III

Use comprehensions, lambdas and loops to query the student list.

- Query 1:
    - Sort the list by lastName, firstName, both in ascending order, and print the results using printStudents().

- Query 2:
    - Sort the list by GPA in descending order and print the results using printStudents().

- Query 3:
    - Create a set that contains all unique courses taken by all students and print the set.
    - Can use set comprehension (see extra credit).
    - Note: there are 17 unique courses

- Query 4:
    - Get a list of students who have taken ‘CSE-201’ and print the list using printStudents().
    - Must use list comprehension.
    - Note: there should be 6 students.

- Query 5:
    - Get a list of “honor roll” students (GPA >= 3.5) and print the list using printStudents().
    - Must use list comprehension.
    - Note: there should be 4 students.

Extra Credit:

Implement query 3 using only a set comprehension. It can have nested “for” statements, but only a single set comprehension.

Scoring:

Criteria	**Points**
Each query worth 5 points	25
TOTAL	**25**

<u>EXTRA CREDIT</u>	
Query 3 set comprehension	5

Hints and Tips:

- For Student.header(), you can use a “\n” character to inject a new line between the header text and the bar of equal signs.
- Speaking of equal sign bar… Don’t forget you can create a string of consecutive characters using the * operator. Example: 'A'*10
- Remember, you can create a dictionary inline with:

  ```
  {"key1": 1, " key2": 2 , "key3": 3}
  ```