CS256 Lab 3 - We need a bigger class...

Value: 25 points (plus extra credit)
Due: 13 September, 2019 @ 23:50
Submission Format: Two files, the python source code and the output file.

This lab gives you more practice with Python's class definitions and related bits. As before you will be editing files remotely, connecting to tools.cs.earlham.edu, running your Python script from the command line on tools, and processing command line arguments. In this lab you will create a family of classes related to students at a large university.

Setup

genericUniversity has two different types of students, **undergraduate**, and **graduate,** with these attributes we would like to track:

- All students have a **studentType** (e.g. under, grad), **first name, last name, ID number, email**
- Undergraduate students also have a **dormRoom** (e.g. BUNDY201)
- Graduate students also have an **office** (e.g. CST218)

genericUniversity offers **courses** with the following attributes and enrollment rules:

- Courses have a **department** (e.g. CS) and a **number** (e.g. 256)
- **enrolledIDList**, a list of student IDs that have been properly enrolled in the class

Tasks

1. Design a family of classes, one for each type of student plus a base class, that include all of the data elements listed above. They should all inherit the appropriate data and methods from the base class. Be sure to follow the guidelines of encapsulation, modularity, naming conventions, scope (public, protected, private), etc. Your design should include **getters and setters** for each of the attributes.

2. Design a course object that includes all the data elements. It should also provide **enrollStudent(studentID)**, and a **countStudents()** method that returns the number of students in the class (this is an important design criteria). This class does not need to expose any public data members.

   Use paper and pen/cil for the design and bring it to class on Wednesday 11 September.

3. Write a function **loadStudents(inputFileName)** that takes a student input file name an argument and returns an array of populated Student objects (this can be a slightly modified version of your earlier code). This can be done by iterating over the input and for each line instantiating & populating a Student object, storing that object in an array and then returning the array.

4. Write a function **enrolling(inputFileName, studentArray)** that takes a course enrollment input file name as an argument and returns a populated course object for CS256 (the data file can be assumed to have entries only for CS256). This can be done by instantiating the course object and then iterating over the input and for each line checking the validity of the ID and then calling the **courseObject.enrollStudent()** method.

5. Roughly speaking your main() will need to process the command line arguments and then invoke your functions in this order: **loadStudents(), enrolling(), cs256CourseObject.countStudents(),** and some print() statements. Your main() will also have the list of student objects as in Lab 2. From the command line redirect the output into a file named **enrollmentCounts.csv.** Your output file should contain a count of the students enrolled in CS256, like so (your value will be different):

   10

Implementation Notes

- Work-out the complete design on paper before you write any code.
- The student input file name and the course enrollment input file name should be required command line arguments (-s and -e).
- Make sure to use Python3 (e.g. "`$ python3 <script> <arguments>`") when testing your code.
- Check to make sure the appropriate file exists before calling **loadStudents()** and **enrolling().**
- We will provide a CSV formatted data files of students and enrollments for you to use in testing. Note that the parsing is a bit more complicated, <aPlace> can be interpreted to be either a dorm room or office (no validation is required).
    - allStudents.csv
      studentType, studentID, lastName, firstName, email, <aPlace>
      …

    - cs256Enrollments.csv
      studentID
      …

Extra Credit

- Implement dropStudent(studentID) and a dropStudents(inputFileName) in the appropriate places. 3 additional points.