CIS 41A - Lab Assignment 4: OOP

Overview

Write a program that maintains a class list for a student.

A student can take:

* for-credit classes, such as CIS, English, math classes that are a certain number of units and have a grade.
* non-credit classes, or self-interest classes such as photography, jewelry making, exercise classes that have no unit and no grade.

The program lets a student enters data on their current classes, then the program prints out the class list, the supplies that are needed each class day, and the grade record for each for-credit class.

Program description

The program has 2 files:

* *course.py*: contains a superclass called Course and 2 subclasses called CreditCourse and NonCreditCourse
* *lab4.py*: provides the interaction with the user and calls the appropriate methods of the Course classes

Class description (in *course.py*)

The superclass Course contains:

* A constructor that accepts 3 input arguments: a class name, start time, class days

Example of the input arguments: CIS 41A, 9:30, MW

* + The class name is a text string
  + The start time must be in the format: 1 - 12 for hour field, colon, 0 or 30 for minute field
  + The days can only be 1 or more of the letters: M T W R F (for Mon - Fri)

- Check that the input arguments are valid and store them in instance variables. If any of the input is not valid, print an appropriate error message and raise an exception. (Suggestion: put the input checking in a separate private method since it is fairly long)

- To check for valid days, you should not have to use multiple or or and conditions: if day == 'M' or day == 'T' ...

* An additional instance variable called supplies: contains supplies that are need for a class, such as "laptop" for a CIS class or "work out clothes, gym shoes" for a PE class.
* An addSupplies method that lets the user add the supplies to a class. This method accepts 1 input argument: a sequence of supplies.  
  A getSupplies method that returns the supplies instance variable
* Any other method as needed to do the unit test shown below

The subclass NonCreditCourse is for all classes that don't have any unit and are not graded.

The NonCreditCourse contains:

* An additional instance variable called activities that contains the main activities of the class. For example, a drawing class might have activities of sketching, texturing, composition, etc.
* An addActivities method that lets the user add a sequence of activities in the class. This method accepts 1 input argument: a sequence of activities.
* Add any other method as needed to do the unit test shown below.

The subclass CreditCourse is for all classes that have a certain number of units and the student receives a grade.

The CreditCourse class contains:

* Two additional instance variables:
  + units: to contain the number of units for the class. For example, for CIS 41A the units will be 4.5.  
    The units must be a value of: 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0.   
    Check for a valid unit *without* using multiple or or and conditions: if unit == 0.5 or unit == 1.0 or unit == 1.5 ...
  + grades: to contain the graded tasks and the corresponding scores. For example, for CIS 41A, the graded tasks are: labs, quizzes, exams, and each task has an associated score.
* An addGrades method that lets the user add a sequence of graded tasks and a sequence of corresponding scores to the grades instance variable. This method accepts 2 input arguments, which is the 2 sequences.  
  If the 2 input sequences don't contain the same number of items, print an error message and don't add anything.
* Add any other method as needed to do the unit test shown below

Unit test:

In addition to the 3 classes, the *course.py* file also contains a printActivities function and a main function.

The code for printActivities function is as shown:

def printActivities(obj) :

obj.printActivities()

The main function does unit testing of the classes.

Here is the code for main to do the unit testing of the classes. Each line of code is numbered for reference.

You need to fill in the input arguments, as shown in the comments.

def main() :

1 c = CreditCourse( ) # pass in name, time, day, units

2 c.addSupplies( ) # pass in a sequence of supplies

3 c.addGrades( ) # pass in a sequence of tasks, a sequence of scores

4 print("Class:\n" + str(c))

5 print("Supplies:", ) # pass in method to get list of supplies

6 printActivities(c)

7

8 print()

9 nc = NonCreditCourse( ) # pass in name, time, day

10 nc.addSupplies( ) # pass in a sequence of supplies

11 nc.addActivities( ) # pass in a sequence of activities

12 print("Class:\n" + str(nc))

13 print("Supplies:" ) # pass in method to get list of supplies

14 printActivities(nc)

Without adding any other line of code in main, fill in the input arguments to main such that it produces the following output:

Class:

CIS 41A, 9:30, MW, 4.5 units # output of line 4

Supplies: laptop # output of line 5

Assignments: 92.5 # output of line 6

Exams: 86.0 # output of line 6

Quizzes: 88.4 # output of line 6

Class:

Hiking, 9:00, T # output of line 12

Supplies: boots, sunscreen, hat # output of line 13

Big Sur, Coastal Trail, Mt. Diablo # output of line 14

Note that when the printActivities function runs (lines 6 and 14):

* It can accept either a creditCourse object or a nonCreditCourse object.
* Note the difference in printing between a creditCourse object and a nonCreditCourse object:
  + nonCreditCourse: All activities are printed on the same line, with comma separation.
  + creditCourse:
    - Each task is printed with a corresponding score, on one line
    - The tasks and scores line up in column format, and the score has 1 digit after the decimal point

*[3 pts extra credit if you can finish the entire course.py with the main shown above by the due date on Canvas]*

Note also that the sample output above doesn't show user input error. Make sure you handle it properly and test it:

* Any invalid input in the constructor: raise an exception or let Python raise the exception
* Any invalid input in other methods: follow the instruction for the method

Test driver description (lab4.py)

Write a program that uses the Course classes to:

* Let the user add current classes to a class list
* Print the supplies for each class day and print the activities of each class.

When you start working on lab4.py, comment out the function call to main so that the main function of course.py doesn't run any more.

You can implement lab4.py using procedural programming (with functions) or using OOP (classes). It's your choice.

* If you use procedural programming, your main function should be:

def main() :

classList = addClasses() # create classList data structure (point 1 below)

report(classList) # print all info of data structure

* If you use OOP, your main function should be:

def main() :

L = ClassList() # create classList object (point 1 below)

L.report() # object prints all data

In addition to the main function, lab4.py should have the following functions or methods:

1. A function or method that lets the user add all their classes and class information

* Prompt the user for the number of classes. Check that it's a number or loop back and re-prompt.
* Loop until you've added the same number of classes that the user entered. For each class:
  + Ask the user whether it's a credit or non-credit class. Check for invalid input and re-prompt.
  + Call a function / method (point 2 below) to create and fill the appropriate Course object and add it to the class list.   
    - Detect any exception and let the user know with an error message.  
    - Note that if the constructor for a Course object has an exception, then the Course object should not be added to the class list.
  + Keep track of the Course objects in a container.

1. A function or method to create and fill the appropriate CreditCourse or NonCreditCourse object.

* Prompt for all the information to create the appropriate object and fill the object with class information.

1. A report function or method that calls 2 function or method:
2. a function or method to print the supplies of each class day
3. a function or method to print the class names and the grades or activities

See sample output for each print

Program requirement:

* Have a document block at the top of the file with your name, lab number and short description of the file.
* Have a docstring for each function and for each public method.
* There should be 3 classes in course.py and each class should work as described above.
* In addition to the given main, there should be 5 functions in lab4.py and each function should work as described above.

Sample output:

The following output is separated into 3 parts for explanation. When your program runs, it will just be one long output.

# 1. This block is the result of the functions / methods of steps 1 and 2 of the lab4.py requirements

Enter number of class: 3

Enter 'c' for credit, 'nc' for non-credit: c

Enter comma-separated class name, time, days, units: CIS 41A,MW,9:30,4.5

Incorrect data entered # error with input data value

Enter comma-separated class name, time, days, units: CIS 41A,9:30,MW,4.5

Enter comma-separated list of supplies: laptop

Enter comma-separated list of tasks: exams,labs,quizzes

Enter comma-separated list of corresponding scores: 92.5,83,98.2

Enter 'c' for credit, 'nc' for non-credit: a

invalid class type # error with class type

Enter 'c' for credit, 'nc' for non-credit: nc

Enter comma-separated class name, time, days: Hiking

Incorrect number of fields entered # error with number of input data

Enter comma-separated class name, time, days: Hiking,8:00,T

Enter comma-separated list of supplies: hat,sunscreen,water

Enter comma-separated list of activities: Coastal Trail,Big Sur,Mt. Diablo

Enter 'c' for credit, 'nc' for non-credit: c

Enter comma-separated class name, time, days, units: EWRT,1:30,MTW,5

Incorrect data entered # error with input data value

Enter comma-separated class name, time, days, units: EWRT,1:30,MTW,5.0

Enter comma-separated list of supplies: textbook,laptop

Enter comma-separated list of tasks: essays

Enter comma-separated list of corresponding scores: 86.5

# This block is the result of the function / method of step 3a of the lab4.py requirements

Supplies for each day of class:

M: laptop, textbook

T: hat, laptop, sunscreen, textbook, water

W: laptop, textbook

# Note: 2 classes require laptop on MW, but laptop only appears once

# Note: only MTW are shown because those are the only class days

# and the days should be printed in order of M T W R F

# This block is the result of the function / method of step 3b of the lab4.py requirements

Activities for each class

CIS 41A, 9:30, MW, 4.5 units

exams: 92.5

labs: 83

quizzes: 98.2

EWRT, 1:30, MTW, 5.0 units

essays: 86.5

Hiking, 8:00, T

Coastal Trail, Big Sur, Mt. Diablo

*[2 pts extra credit:*

*Print the classes in alphabetical order by name when print activities for each class.*

*To earn the extra credit, your code must not go through the container of Course objects more than once.*

*For example, the code should not go through the Course objects to get all the class names, then sort the class names, and then use the sorted names to go through the Course objects again to print the class information.]*