CSE 231 **Fall 2016**

Computer Project #11

(Update 12/2: clarified Grade __init__ and Student __str__ requirements. Update 11/30: clarified specifications of __init__ and __str__ and added a note about the sample test file).

Assignment Overview

This assignment focuses on the implementation of two Python classes to store grades of students in a class.

It is worth 55 points (5.5% of course grade) and must be completed no later than 11:59 PM on Tuesday, December 6.

Assignment Specifications

You will develop "Student" and "Grade" classes and implement all the methods described in the following section.

You will demonstrate that your implementation of the classes are correct by developing a program which serves as a test bed for that class.

You will develop an application program which uses those classes to read "students.txt" and "grades.txt" files and store their data in instances of Student and Grade classes.

Assignment Deliverables

The deliverables for this assignment are the following files:

classes.py – the source code for your Student and Grade classes
proj11-test.py – the source code for your test program
proj11-app.py – the source code for your application program

Be sure to use the specified file names and submit them for grading via the **handin** system before the project deadline.

Specifications for Class Grade

1. The constructor (method `__init__`) will accept 3 values as parameters, name of the assignment (string), grade and assignment weight (floats). All three parameters must have default values—you choose appropriate ones. Also, the attributes **cannot** be private because you need to access them in the Student class. That is, do **not** use double underscores in your names, i.e. do **not** use an attribute name such as `self.__name`.

2. The string method (`__str__`) and representation method (`__repr__`) should return a string that includes the name, grade and weight of the Grade object in one line.

Specifications for Class Student

1. The constructor (method `__init__`) will accept 4 values as parameters, student id (int), first name (string), last name (string), and a list of all the grade objects for that student. All three parameters must have default values—you choose appropriate ones, but the appropriate default value for a list parameter is `None`, not an empty list (`[]`).

2. **`add_grade`** method should take a Grade object as a parameter and append it to a list of grades for that student.

3. **`calculate_grade`** should multiply all the grade points of the student by their weight and return the final course grade. (Each Grade object in the list has a value and weight that should be multiplied together. The sum of all the calculations results the final weighted grade of the student in the course)

4. The string method (`__str__`) and representation method (`__repr__`) should return a string that has the student's name (with last name first with a comma between the last name and first name) includes all the grades of the student. This method **must** call the `__str__` method of the Grade objects. Note that implicitly calling the `__str__` method is fine such as `str()` or `format()`, but do not access the individual attributes such as `grade.name`.

5. `__gt__`, `__lt__` and `__eq__` methods should compare the final grades of two students together.

   Note that you can use `calculate_grade` method to calculate the final grade of two students. Never use "`==`" to check for equality of two float numbers (because floats are approximations of real numbers). Instead, find the absolute value of the difference of two numbers; if the difference is less than epsilon (a very small number such as `10**-6`), the floats are equal to each other (that is they are close enough to consider them to be equal). Python has an absolute value function: `abs()`

Please note that you may wish to use function "print" to display various items as you are developing your implementation of the class. However, all invocations of function "print" must be removed from the final version of your classes (or at least turned into comments).

Hint: check the "Student" class in page 546 of the book for reference.

Specifications for the Test Program

1. You will develop a program to serve as a test bed for classes. That is, the only purpose of the program is to demonstrate that each method of your "Student" and "Grade" classes is implemented correctly. The source code for your test bed will be contained in the file named "proj11-test.py". That file will import "classes.py". A file named "proj11-test.py" is provided as a starting template.

2. Your test bed will not perform any input operations. Instead, all test cases will be embedded in the program itself.

3. The output produced by your test bed must be appropriately labeled so that the reader can understand the purpose and result of each test case without examining the source code. For example, when

demonstrating `__lt__` you might print "Demonstrating Fred < Mary" followed by `print(Fred<Mary)`

Specifications for the Application Program

1. You will develop an application program which uses both classes to solve the problem described below.

The source code for your application program will be contained in the file named "proj11-app.py". That file will import "classes.py".

2. The program will attempt to access the files named "students.txt" and "grades.txt". If any of the input files cannot be opened, the program will display an appropriate message and halt.

3. The grades.txt file will contain three or more lines. The first line contains the weight of each assignment, the second line contains the labels and the name of the assignments. Line 3 and subsequent lines contain grade information for a student. Each line has a student id and all the grades of a student. The information of each assignment should be stored in an instance of "Grade" class. A list of Grade objects should be passed to the constructor of "Student" class.

4. The students.txt file will contain one or more lines, where each line contains the information of a student. Each line has an id, first name and last name. That information should be stored in an instance of "Student" class along with the list of "Grade" objects explained above.

5. The application should store all "Student" objects in a list and should print them (print(student)) like the sample output provided below.

6. The application should print the class average at the end.

Assignment Notes

1. You would be wise to develop your implementation of "Student" and "Grade" classes and your test program incrementally and in parallel. That is, implement one class method at a time and then test that method by adding statements to your test program.

Clearly, the first class method which must be implemented and tested is the constructor `(__init__)`.

Perhaps the second class method to implement and test is `__str__` so that you have a way to display the value of an object of type "Student" and "Grade" using function "print".

After implementing and testing those methods, you would continue to implement and test one method at a time until you have completed the class.

Be sure to insert and test any necessary error handling at the appropriate time.

2. Approximately 60% of the 55 points available for the project will be allocated to the implementation of "Student" and "Grade" classes, approximately 15% will be allocated to the test program, and approximately 25% will be allocated to the application program.

Sample Output (Application file)

```
Hopper, Grace
proj01     :   75%   0.07
proj02     :   81%   0.08
proj03     :   91%   0.10
proj04     :   87%   0.10
proj05     :   73%   0.15
exam01     :   90%   0.20
exam02     :   86%   0.30
Final grade:   84%

Knuth, Donald
proj01     :   79%   0.07
proj02     :   73%   0.08
proj03     :   83%   0.10
proj04     :   87%   0.10
proj05     :   82%   0.15
exam01     :   97%   0.20
exam02     :   99%   0.30
Final grade:   89%

Goldberg, Adele
proj01     :   92%   0.07
proj02     :   82%   0.08
proj03     :   92%   0.10
proj04     :   83%   0.10
proj05     :   96%   0.15
exam01     :   92%   0.20
exam02     :   93%   0.30
Final grade:   91%

Kernighan, Brian
proj01     :   77%   0.07
proj02     :   95%   0.08
proj03     :   98%   0.10
proj04     :   87%   0.10
proj05     :   87%   0.15
exam01     :   85%   0.20
exam02     :   82%   0.30
Final grade:   86%

Liskov, Barbara
proj01     :   72%   0.07
proj02     :   82%   0.08
proj03     :   98%   0.10
proj04     :   90%   0.10
proj05     :   75%   0.15
exam01     :   98%   0.20
exam02     :   94%   0.30
Final grade:   89%


The class average is 88.17%


========================================
```

Educational Research

When you have completed the project insert the 5-line comment specified below.

For each of the following statements, please respond with how much they apply to your experience completing the programming project, on the following scale:

1 = Strongly disagree / Not true of me at all
2
3
4 = Neither agree nor disagree / Somewhat true of me
5
6
7 = Strongly agree / Extremely true of me

*********Please note that your responses to these questions will not affect your project grade, so please answer as honestly as possible.**********

Q1: Upon completing the project, I felt proud/accomplished

Q2: While working on the project, I often felt frustrated/annoyed

Q3: While working on the project, I felt inadequate/stupid

Q4: Considering the difficulty of this course, the teacher, and my skills, I think I will do well in this course.

Please insert your answers into the bottom of your project program as a comment, formatted exactly as follows (so we can write a program to extract them).

```
# Questions
# Q1: 5
# Q2: 3
# Q3: 4
# Q4: 6
```