VSFX 705 / Exercise 4 / Spring 2013 / Fowler*

Exercise 4 – OOP with Python – Interactive Student Database

DATE DUE: Friday after Class 18
DATE ASSIGNED: Class 17

Goals:
This assignment will focus on the student becoming familiar with python and object oriented programming. The emphasis will be on coding.

Requirements:
Create a grading database that tracks student records and allows queries of those records. You must use object oriented programming in your design and implementation of an interactive student records database.

The program will process a file of student records and interactively answer queries from a user.

Considerations:
As a warm up exercise, start with the student record as show in the notes. Expand on this to achieve the following.

1. Grades entered and their weighting.
2. Average for the student.
3. Letter grade for the student.

This program should be written so that you could use it to compute your grade average in any of the classes you take this quarter, regardless of number of assignments or weighting based on those assignments.

The program should be able to be run for multiple students, not just one.

Once you have the above working, start designing a program based on the following specifications:

Specifications:

1. **File Input Specification:**

   The program should read in a file of student records in the following format:

   StudentSurname StudentGivenName
   YearInProgram
   Assignment1Grade Assignment2Grade Assignment3Grade Assignment4Grade

   Explanation:

StudentSurname:	The student’s surname. Contains no white space.
StudentGivenName:	The student’s given name. Contains no white space.
YearInProgram:	The student’s year of study. A whole number $\geq$ 1.
Assignment1Grade:	The student’s grade on the first assignment. (Whole numbers from 0 to 100)

Assignment2Grade:	The student's grade on the second assignment.
Assignment3Grade:	The student's grade on the third assignment.
Assignment4Grade:	The student's grade on the fourth assignment.

Example:

Here is an example of a data file to be read by the program:

```
Cool Joe
3
99 80 100 70
Brown Sally
1
80 99 70 100
VanPelt Linus
2
65 82 38 40
```

For this exercise you may assume that no two students have the same surname/given name combination.

2. **Program Output and Behavior Specification:**

Once the program has read in the data file, it should request and process interactive queries from the user. It should request queries in the following format:

```
Student data read in.
Query student average[a], maximum grade[m], year[y] or exit session[e]?
```

If the user responds by entering **a**, the program should request a student name:

```
Student data read in.
Query student average[a], maximum grade[m], year[y] or exit session[e]? a
Student Name?
```

After the user enters the student's name, the program should compute and output the student's average, to two decimal points, and then request the next query:

```
Student data read in.
Query student average[a], maximum grade[m], or year[y] or exit session[e]? a
Student Name? Brown Sally
Brown Sally average is 87.25
Query student average[a], maximum grade[m], or year[y] or exit session[e]?
```

The user may then similarly query a student's maximum grade by entering m:

```
Student data read in.
Query student average[a], maximum grade[m], or year[y] or exit session[e]? a
Student Name? Brown Sally
Brown Sally average is 87.25
Query student average[a], maximum grade[m], or year[y] or exit session[e]? m
Student Name? Cool Joe
Cool Joe maximum grade is 82
Query student average[a], maximum grade[m], or year[y] or exit session[e]?
```

The user can also request a student's academic year by entering y, or exit the program by entering e.

3. **Program Structure:**

   The program must use an object oriented approach: specifically it must use an object to store each student's data, and provide that object with methods to return and/or output each student's average, maximum grade, and academic year.

4. **Error Handling:**

   For this exercise, you can assume that the student data input file is properly formatted. However, the program should gracefully handle incorrect interactive input from the user. For example, if the user does not respond with a, m, y or e to the program's query request, or provides the name of a student not in the database, the program should respond sensibly, and definitely not crash or freeze.

Submissions guidelines

Create a directory named **LastnameFirstnameExercise4**

- you should be prepared to have your program run on test data
- LastnameFirstnameExercise4.py files and any additional files named appropriately

Grading:
Functional, clean, efficient code will be the emphasis. Design and implementation are key. Neatness, organization and clear understandable code that achieves the purpose and functionality according to the specifications are important.
Meeting the minimum specifications, 80%. To move your grade above 80% go beyond the specifications, demonstrate exploration and understanding.

Remember, when working on Windows you may use idle. When working on linux, when you save your file with a .py extension it will recognize the proper formatting. Run your program from the command line using python nameofyourfile.