CSCI 1300 Computer Science I: Starting Programming
Homework 7

Objectives:

- Implement simple file I/O commands to read data from a file, work with it, and write data onto a file.
- Apply the concept of modular programming through the design of functions.

Reading Data from File and Generating a Report

You are a counselor working at George Washington High School in Illinois. Over the years you have prided yourself on the quality of education provided to your students, and you have always tried to make sure that your students are met with the best care possible. You are so dedicated, in fact, that you have talked with every one of your students' parents and have determined goals for them to meet during the year.

Every week you check your student's grades to make sure that they are doing well across all of their subjects. This has worked wonderfully over the past few years, and you have seen a remarkable improvement in the performance of your students. However, there is now a problem. Due to budget cuts, the school had to let one of the other councilors go, and the number of students that you are responsible for has nearly doubled! You have tried to keep up with all of their grades, but it has simply been too much to do.

But there is light in the darkness! You remember that you took a programming course in college, and you set out to build a program that will generate a report to tell you whose parents you need to call!

What is given:

Currently, all of the data on your students is stored in the file *student_files.csv* (*.csv means comma-separated-value file). Each line of data contains the student's first and last name, their **current grade** for each of the subjects, as well as their **objective grade** for each of the subjects, as set after the discussion with the parents. The file also contains a **warning grade** for each subject.

The warning grade is higher than the objective grade and it serves as a buffer; if the student's current grade dips below the warning grade, then it's time to call the parents, before the students dip below their objective grade.

In the header of the file (the first row in the table), you will find the following fields:

Last Name,First Name,Math_cur,Math_obj,Math_war,Engl_cur,Engl_obj,Engl_war,
Hist_cur,Hist_obj,Hist_war,Sci_cur,Sci_obj,Sci_war,Lang_cur,Lang_obj,Lang_war

Note: The field names occur on the same line in the file (the 1st line), but they did not fit on the page here, so the list is broken up. Also, you should note that none of these will be acceptable variable names to use since there are either spaces in the name, or they start with a capital letter. If you try and copy these names from this sheet and use them as variable names, they will not work.

Your task:
You need to generate a report that tells you two things about the students that are having difficulties achieving the grades they want:

1. Which courses do you need to be concerned about?
    - A grade will be concerning only when the current grade is at or below the "warning" level, but still above the semester "objective" (i.e. it is the score below which the student is in danger of falling below their desired grade).
2. Which courses where the student is in trouble?
    - A grade will indicate "trouble" when the current grade is at or below the "objective" for the semester.

Note: These statuses are mutually exclusive, so courses that appear in the "warning" status should not appear in the "trouble" status and vice versa.

Here is an example of a student's performance in Math and English:

Math_cur	Math_obj	Math_war	Engl_cur	Engl_obj	Engl_war
77	76	79	70	78	81

This student would be considered to have a "concerning" status for Math, and a "trouble" status for English.

The report should contain a block for each student how has at least one "concerning" or "trouble" subject. The block should include the student's full name on the first line, followed by a list of the "concerning" courses, followed by a list of "trouble" courses. For the "concerning" courses, the report should output the current grade and the warning grade for the subject in case, while for the "trouble" courses, the report should output the current grade and the objective grade. The report you will generate should look like in the sample output below:

Student name: Jane Testcase
    Concerning Courses:
        Current Math grade: 77, Warning Math grade: 79
    Trouble Courses:
        Current English grade: 70, Objective English grade: 78

To get the a consistent formatting, you will want to make use of the tab character '\t', so an example of a line in your code would be:

```
print("\t Trouble Courses:")
```

There are some other things that you want to keep in mind for this. First of all, you do not want redundant data, so items that appear in the trouble section to not appear in the concerning section and vice versa. Second of all, while you do have access to the file, it can change on an almost daily basis (damn budget cuts!), so you want to write a program that will work for a file of any length.

Program Flow

1. Import the correct libraries
2. Open input file with student data
3. Create new output file *grades_report.txt* to write the reports into

(a) You can open files in the directory you are working in by using the *open* function:
`open("<File_name_you_want_to_open>", "<filetype>")`
The <filetype> commands that you might need for this assignment will be:

- "a" (append)
- "w" (write)
- "r" (read)

(b) WARNING: If the file *grades_report.txt* does not exist in the project directory, then the command `open("grades_report.txt", "w")` will create the file. If the file already exists in the project directory, then the command `open("grades_report.txt", "w")` will overwrite the existing file.

4. Ignore the THE HEADER line (the *readline* function is wonderful for this). You will need to remember however, the subjects evaluated and their order:

- Math: columns 3-5
- English: columns 6-8
- History: columns 9-11
- Science: columns 12-14
- Language: columns 15-17

5. Iterate through each line of the input file:
(a) For each line in the file, split the line into a list of 17 strings. Based on the values of the grades if each subject, decide whether or not a student has any grades in the "concerning" or "trouble" region.

Suggestion: since comparing each student's grades is a task you will be doing repeatedly, consider creating functions like
`has_trouble_course(<input_arguments>)` or
`has_concerning_course(<input_arguments>)`.
You may elect to pass the entire line as an argument to one of these functions, or just the 3 scores for one particular subject. Consider the advantages and disadvantages of each approach. It's your choice!

(b) If the student does have grades in the "concerning" or "trouble" region, you must append the student's information to the report file, in the same format as the provided sample output file *grades_report_sample.txt*.

Suggestion: since writing to the report file is a task you will be doing repeatedly, consider creating functions like
`add_trouble_to_report(<input_arguments>)` or
`add_concern_to_report(<input_arguments>).`

6. Once you are done iterating through the input file and have written the entire report to the *grades_report.txt* file, remember to CLOSE ANY FILES THAT ARE STILL OPEN (use the *close* function).

You will be given the *student_files.csv* file to test the report generation on, but you should be warned that this will not be the file that we will be grading on. You can compare your report file with the *grades_report_sample.txt.*

Modular Programming Approach

Modular programming is a software design technique that emphasizes separating the functionality of a program into independent parts, such that each contains everything necessary to execute only one aspect of the desired functionality. In this homework assignment, you will implement this approach by identifying different parts of the program and creating one function for each.

Some functions are already suggested above. Try to identify other parts of the program that can be implemented and tested independently of the rest of the program. Also, every action that is executed repeatedly should also be separated into a function. Your program should have at least **three** meaningful user-defined functions.

The only code in your program besides the function definitions should be a single call to *main():*

```
if __name__ == "__main__":
    main()
```

main() is where variables are created, where the functions are called, where parameters are passed and where values are returned. In addition, *main()* may contain control statements (if/elif/else), and print statements. All functions should be called, either from *main()* or from other functions.

Program Variables

Your program should not have any global variables. Global variables are variables defined above *main(),* outside of any function. Variables should be declared in the *main()* function, preferably at the beginning of the function. Variables should then be passed as arguments to the functions that are being called from *main(),* if necessary.

Here's how your modular components are arranged into your .py file:

```
# 1. import needed libraries

# 2. define functions

# main() should be defined last
# def main():

# 3. if statement that calls the main() function. It also allows the
user-defined functions to be imported and used in another program

if __name__ == "__main__":
    main()
```

Note: all the style/commenting/naming expectations from homework 6 still apply. Please review the **Style and Comments** section from homework 6.