Assignment 2

Due: Thursday, February 27, 2025 (11:59 pm)

Submission via Git only

Overview

Programming environment 1
Individual work 1
Objectives of this assignment 2
This assignment: *spf_analyzer.py* 2
Restrictions 3
Testing your solution 4
What you must submit 4
Input specification 4
Output specification 4

Programming environment

For this assignment, you must ensure that your code executes correctly on the UGLS server, which you learned about as part of Lab 01. This same environment will also be used by the teaching team to evaluate your submitted work. You must ensure that your program compiles, links, and executes perfectly on UGLS. **If your program does not run on UGLS, even if it works fine on your local computer, your submission will receive 0 marks**. All test files and sample code for this assignment are available on your Git repository. After cloning your repository, you can use the following command to obtain the sample code (inside your repository, of course):

```
git pull
```

Individual work

This assignment is to be completed by each individual student (i.e., no group work). You are encouraged to discuss aspects of the problem with your fellow students. **However, sharing of code fragments is strictly forbidden.** Note SENG 265 uses highly effective plagiarism detection tools to discover copied code in your submitted work. Both, using code from others and providing code to others, are considered cheating. If cheating is detected, we'll abide by the strict UVic policies on academic integrity: https://www.uvic.ca/library/help/citation/plagiarism/

Objectives of this assignment

- Understand a problem description, along with the role of the provided sample input and output.
- Learn or review basic features of the Python 3 programming language.
- Use the Python 3 programming language to write an introductory data science project.
- Use Git to manage changes in your source code and annotate the evolution of your solution with messages provided during commits. Remember, the only acceptable way of submitting your work is using Git.
- Test your code against the provided test cases.

This assignment: *spf_analyzer.py*

Similar to A1, you will learn Python by solving a problem involving the CSV file format. The primary objective of this assignment is to process data from a single .csv file containing information about curricular and extracurricular factors influencing student performance.

During each execution, your program must complete one of the specified tasks and generate a corresponding CSV file named output.csv. To assist with implementation, the expected solution for each task (i.e., the CSV file your program should produce) is provided in the tests folder within the a1 directory of your Git repository.

The following table lists the expected tasks to be supported by your implementation and the expected outputs.

TABLE 1.[1]

Task	Expected Output
1. Generate a CSV file containing the Record_ID, Hours_Studied and Exam_Score for students who studied more than 40 hours.	a2/tests/test01.csv
2. Generate a CSV file that contains the top 10 records and the columns Record_ID, Hours_Studied, and Exam_Score, but only for entries where Exam_Score is at least 85. Sort the results primarily by Exam_Score in descending order; if two scores are identical, sort by Record_ID in ascending order.	a2/tests/test02.csv

[1] An outlier is a data point that differs significantly from others. Here, a student scored 101 on the exam, exceeding the normal 0-100 range. This outlier may affect results in Task 4 and Task 5. While removing outliers is common in data cleaning, we chose to retain it to highlight its impact on the analysis. Thus expected results will include this record.

3. Generate a CSV file containing the Record_ID and Exam_Score for students who meet the following conditions: Attendance is exactly 100%, and Extracurricular_Activities is marked as 'Yes'.	a2/tests/test03.csv
4. Generate a CSV file containing two columns: Grade and the corresponding Average_Attendance. The Grade column should classify students based on their exam scores using the following scale: A+ (90-100), A (85-89), A- (80-84), B+ (77-79), B (73-76), B- (70-72), C+ (65-69), C (60-64), D (50-59), and F (0-49). The Average_Attendance for each grade group should be computed and rounded to one decimal place.	a2/tests/test04.csv
5. Compute the grade classification for each student using the following grading scale: A (80-100), B (70-79), C (60-69), D (50-59), and F (0-49). Calculate the average number of Tutoring_Sessions for each grade group, rounded to one decimal place. Generate a CSV file containing the following columns: Record_ID; Tutoring_Sessions; Grade_Average_Tutoring_Sessions: the average tutoring sessions for the student's grade group; Above_Average: a boolean value (True or False) indicating whether the student's Tutoring_Sessions exceed the average for their grade group; Exam_Score; and Grade. Ensure that the records in the CSV are sorted by Exam_Score in descending order, and for students with the same score, by Record_ID in ascending order. Limit the results to 50 records.	a2/tests/test05.csv

Restrictions

- Your program must run as a script.
- Your program must be decomposed into easy-to-understand functions. Good program decomposition is required. Methods or functions must be short and effectively parameterized.
- Unwieldy functions are not accepted. The main function should especially be easy to understand and represent a decomposition of the problem at hand.
- **Typing (i.e., type hints) must be used in variables and functions defined in your program.**
- Do not use global variables.
- **You can only use Python modules and libraries that DO NOT require additional installations on UGLS**. As part of the configuration of UGLS, we included relevant libraries for managing data such as numpy and pandas. The latter might be very useful to answer the assignment

tasks. **Failing to comply with this restriction will result in significant lost marks or even 0 marks for the assignment.**

- Keep all your code in one file (**spf_analyzer.py**) for this assignment. In Assignment 4 we will use the multi-module features of Python.

Testing your solution

As with A1, the tester file will execute your program automatically. You'll only need to pass the number of the task as an argument (e.g., `./tester 1`). If no arguments are passed to the tester file (i.e., `./tester`), it will run the tests for all the questions. Using a specialized library that compares the differences between .csv files, the tester file will describe the differences between the expected output and the one provided by your program.

- Refer to the example commands in the file "**TESTS.md**" for appropriate command line input. Your solution must accommodate all specified command line inputs.
- Make sure to use the test files provided as part of this assignment.
- Use the test files and listed test cases to guide your implementation effort. Develop your program incrementally. Save stages of your incremental development in your Git repository.
- For this assignment you can assume all test inputs are well-formed (i.e., exception handling is not required).
- **DO NOT rely on visual inspection**. You can use the provided **tester** file so you can verify the validity of your outputs in a simpler manner.

What you must submit

A single Python source file named **spf_analyzer.py**, submitted to the a2 folder **in your Git repository**. Git is the **only** acceptable way of submission.

Input specification

1. The input dataset for this assignment is a simplified version of the Student Performance Factors dataset from Kaggle.com.
2. Refer to https://www.kaggle.com/datasets/lainguyn123/student-performance-factors for more information about each column/property for each record (e.g., Hours_Studied).

Output specification

1. All output must be generated in a file called *output.csv*.
2. Refer to the tests folder for more information about the expected output for each test.