SEHH/SEHS2376 Data Science with Programming

Take-home Assignment 2

Deadline: 18 Nov 2024, 6:00PM

Instructions:

- Complete this assignment using the Python Notebook Template provided on Blackboard. You must name your file using your student ID. E.g. if your student ID is 24123456A, then your notebook should be named as 24123456A.ipynb.
- You are expected to use methods taught in our classes. Using methods beyond the class will not have additional bonus.
- You should submit your completed Python Notebooks onto Blackboard on or before the deadline.
- Submissions or modifications after the deadline will be considered as late, regardless of the changes made.
- DO NOT submit a link to your program file. You must submit the actual .ipynb file onto Blackboard.
- Penalties will apply for assignments submitted after the deadline. Five marks will be deducted for each hour after the deadline. (i.e., 5 marks deduction if the assignment is within 1 hour late, 10 marks deduction if the assignment is 1 hour late to within 2 hours late, etc.)
- Assignments will not be accepted 12 hours after the deadline.
- Plagiarism will be penalized severely. Marks will be deducted for assignments that are plagiarized in whole or in part, regardless of the sources.
- Indicate if you have used Generative AI when completing the assignment and explain how it is used.

Assessment Criteria: The full mark of this assignment is 100 marks. The assignment will be assessed based on the following criteria and percentages.

Marking Criteria	Marks
Problem solving and program development	80 marks
Comments for explaining your code	5 marks
Tidiness of the code	5 marks
Generative AI Declaration	10 marks

Marking Rubrics

Criteria	Rating					
	0	**1**	**2**	**3**	**4**	**5**
Problem solving and program development	The program is not being written. No output can be produced.	There are runtime errors, and no output can be produced.	There are no run-time errors, but the program is not functioning correctly. Majority of the outputs are incorrect.	There are no run-time errors, but some logical error errors. Leading to some incorrect outputs.	There are no syntax errors, logical errors, or run-time errors. The program functioned as required but not written in an efficient way, or there are some aesthetic issues.	There are no syntax errors, logical errors, or run-time errors. The program functioned as required and written in an efficient way.
Comments	There are no comments in the code.	Very little to no comments in the code.	Comments are provided for part of the code, and they are not clear enough.	Comments are provided for majority of the code, but they are too brief.	Clear comments are provided for majority of the code.	Clear comments are provided for all sections of the code.
Tidiness	The program is not written using the template provided and the codes are badly organized with redundant codes.	The program is not written using the template provided and the codes are badly organized.	The program is not written using the template provided. However, there is an attempt to format the code.	The program is written using the template provided. The formatting and coding style vary greatly in the code.	The program is written using the template provided. There is a mild inconsistent in formatting and coding style.	The program is written using the template provided. The coding style and formatting are consistent throughout the code.
GenAI Declaration	No declaration	Declared but no explanations given.	Declared with very little explanations.	Declared and there are some explanations on how problems are solved with or without GenAI.	Declared and explained how problems are solved with or without GenAI, but the explanations are not totally clear.	Declared and explained clearly how problems are solved with or without GenAI.

Question 1 (40 marks)

In this question, you are required to write a Python program to calculate summary statistics for categorical data. However, you are NOT ALLOWED to use statistical summary functions from any modules.

Requirements of your program:

1. Use a loop to first read names from the user and store them in a list. Then, use a second loop to read scores corresponding to each name and store the scores in a separate list. Simultaneously, create a dictionary with names as keys and scores as values. The scores should be integers between 0 and 100, inclusive. Round the score to an integer if it is not an integer. If a score is outside the range, discard it and prompt the user to input again until it meets the range constraint. The program should continue prompting the user to enter names until the input is the string "end", which will stop the input process.

2. The teacher needs to convert the scores into grade levels (A, B, C, D, F). We will classify scores between 90-100 as A, 80-89 as B, 70-79 as C, 60-69 as D, and scores under 60 will be considered as F. Create a dictionary to count the number of occurrences in each grade level, with the grade level as the key and the number of students in each grade level as the value.

3. Find the mode(s) of the grade levels in the list, meaning the grade level(s) with the highest frequencies. There can be more than one mode, and all modes found should be stored in a list.

4. Calculate the average score of the students, find the maximum score with the corresponding student names, and the minimum score with corresponding student names. You cannot use any built-in functions like mean(), max(), or min(); instead, write your own code to perform these calculations. There may be more than one student with the maximum or minimum score.

5. Output the following at the end:
   a. The dictionary of names and scores provided by the user.
   b. Number of scores entered.
   c. The dictionary contains the frequencies of the grade levels.
   d. Mode(s) of the grade level.
   e. Frequency of the mode(s).
   f. Average score of the students
   g. Maximum score with the student names (can be more than 1)
   h. Minimum score with the student names (can be more than 1)

6. Your output should follow the sample output provided below.

Sample Output 1:

```
Enter student name (type 'end' to finish): Amy
Enter student name (type 'end' to finish): Jack
Enter student name (type 'end' to finish): John
Enter student name (type 'end' to finish): Elsa
Enter student name (type 'end' to finish): end
Enter score for Amy: 100
Enter score for Jack: 100
Enter score for John: 20
Enter score for Elsa: 20
The names and scores entered are: {'Amy': 100, 'Jack': 100, 'John': 20, 'Elsa': 20}
Number of scores entered: 4
Frequencies for each grade level: {'A': 2, 'B': 0, 'C': 0, 'D': 0, 'F': 2}
Mode(s) of grade level: ['A', 'F']
Frequency of the mode(s): 2
Average score of students: 60.0
Maximum Score(s): 100
Students with the highest scores: ['Amy', 'Jack']
Minimum Score(s): 20
Students with the lowest scores: ['John', 'Elsa']
```

Sample Output 2:

```
Enter student name (type 'end' to finish): a1
Enter student name (type 'end' to finish): a2
Enter student name (type 'end' to finish): a3
Enter student name (type 'end' to finish): a4
Enter student name (type 'end' to finish): a5
Enter student name (type 'end' to finish): end
Enter score for a1: 67
Enter score for a2: 79
Enter score for a3: 83
Enter score for a4: 22
Enter score for a5: 55
The names and scores entered are: {'a1': 67, 'a2': 79, 'a3': 83, 'a4': 22, 'a5': 55}
Number of scores entered: 5
Frequencies for each grade level: {'A': 0, 'B': 1, 'C': 1, 'D': 1, 'F': 2}
Mode(s) of grade level: ['F']
Frequency of the mode(s): 2
Average score of students: 61.2
Maximum Score(s): 83
Students with the highest scores: ['a3']
Maximum Score(s): 22
Students with the lowest scores: ['a4']
```

Question 2 (40 marks)

In this task, you will process student answers for three tests following optical machine scanning. The scanned data is stored in three CSV files as detailed below.

The file **Student_List.csv** contains the personal information of all students. The table below describes the meaning of each column in the file.

Column	Type	Description
ID	Integer	It is the unique identification number of the student.
Full Name	String	It is the English name of the student.
Username	String	It is the email address of the student.

The files **Scanned_Test_1.csv**, **Scanned_Test_2.csv**, and **Scanned_Test_3.csv** contain the direct scanned results of each student's answers for individual questions. The first row displays the correct answers, while all subsequent rows show the scanned answers from students. Each test consists of 15 multiple-choice questions with options A, B, C, D, and E. The columns in all three files are consistent, as described in the table below, which explains the meaning of each column in these files.

Column	Type	Description
respondent_id	String	Except for the first row, this represents the sequence number assigned when scanning each student's answer sheet.
STUDENTNUMBER	String	It is the unique identification number of the student.
P01_P1_Q*	String	It indicates the student's answer choice for this question (options: A, B, C, D, E).
TotalScore	Integer	This is the student's score, calculated by comparing their answers to the correct answers in row 0. Each question is worth 1 point, with a maximum score of 15 points.

Main Goals of the program:

1. Manipulate the DataFrames for each test and calculate each student's TotalScore.
2. Merge all the TotalScore columns with the student list.
3. Calculate and display summary statistics of the test scores.

Requirements of your program:

1. Read all four CSV files into DataFrames.

2. The **TotalScore** column currently contains random numbers. Calculate the correct total scores and update the **TotalScore** column accordingly.
   Hint: One possible solution is to use a for loop to compare the answers in each row with the first row, and then store the calculated values to TotalScore iteratively.

3. Use a for loop to create a dictionary **new_column_names** where the keys are column names P01_P1_Q1 to P01_P1_Q15, and the corresponding values are the new column names Q1 to Q15. Use a for loop and string concatenation to achieve this.
   Hint: If you're unsure about the value, print it in each iteration before assigning it.

4. Use the **new_column_names** dictionary created above to rename the columns in the DataFrame. Display the first two rows of the modified DataFrame.

5. Save the cleaned file as **Scanned_Test_1_Cleaned.csv** (don't need to save index). Repeat this process for the other two files and save them as **Scanned_Test_2_Cleaned.csv** and **Scanned_Test_3_Cleaned.csv**.

6. In the DataFrame loaded from **Student_List.csv**, create three new columns: **"in-class1", "in-class2",** and "**in-class3**" by merging the **TotalScore** column from each Test CSV file respectively. Display the first 5 rows of the merged DataFrame. The merged DataFrame should contain only the original three columns from Student_list.csv and three new columns for in-class1, in-class2, and in-class3. Save the modified DataFrame as **Student_Results.csv** without the index.

7. Calculate the summary statistics for the three tests, including the minimum, maximum, average, and the 25th, 50th, and 75th percentiles. Display the results in a DataFrame.

8. Below is a reference for the intermediate results.

Reference Intermediate Results: (Note that the values shown are not the actual results; they are provided for illustration purposes only.)

First 2 rows of the DataFrame containing calculated TotalScore (Similar for all 3 tests)

	respondent_id	STUDENTNUMBER	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	TotalScore
0	MC(Answer)	0000000000	B	A	C	C	B	C	A	E	C	E	E	D	E	A	B	15
1	0000000001	63820206LO	C	C	B	C	A	B	C	B	A	D	A	A	B	A	B	3

First 5 rows of the DataFrame containing personal information of the students and the scores from the 3 in-class tests.

	Full Name	ID	Username	in-class1	in-class2	in-class3
0	Grace Adams	00313273ZK	9cv10ll3@common.cpce-polyu.edu.hk	0.0	6.0	0.0
1	Mason Roberts	69421036WD	d5tfm00l@common.cpce-polyu.edu.hk	15.0	11.0	15.0
2	Scarlett Green	38543922OC	074qz3j4@common.cpce-polyu.edu.hk	10.0	0.0	10.0
3	Amelia Martin	55952731VK	741blaij@common.cpce-polyu.edu.hk	12.0	11.0	12.0
4	Lily Thompson	69088854YO	dk8xl4mb@common.cpce-polyu.edu.hk	8.0	12.0	8.0

Summary statistics of the 3 in-class tests

	in-class1	in-class2	in-class3
mean	3.365854	8.243902	7.439024
min	1.000000	0.000000	0.000000
25%	2.000000	5.000000	4.000000
50%	4.000000	9.000000	7.000000
75%	4.000000	12.000000	11.000000
max	7.000000	15.000000	15.000000

Tidiness of the Code **(5 marks)**

Here are some basic guidelines about tidiness:

- The number of spaces used indentation should be consistent throughout the assignment.
- Variable names should be meaningful but not excessively long.
- Suitable use of blank lines to separate different sections of your code.
- It is not limited to the above points. The goal is to make your program easy to read.

Comments for explaining your code **(5 marks)**

Your program should have in-code comments.

- In-code comments should be able to explain what your program is doing.
- The comments should be clear and concise.
- You should align your comments consistently in your program.
- It is not limited to the above points. The goal is to make sure another programmer will understand the logic used in your program.

Generative AI Usage Declaration (200 words) **(10 marks)**

At the end of the assignment, you need to write a declaration of the use of generative AI in your assignment. Below are some items that you can cover in your declaration.

- Have you used Generative AI when working on your assignment?
- If you have used Generative AI, how do you use it in your assignment? E.g., debugging, understanding syntax, … You need to provide some examples.
- Which part of your assignment is completed with the use of generative AI?
- If you have not used Generative AI, you should also explain how you resolve your bugs and problems encountered in this assignment. You need to the provide the source of your information.

End of the Assignment