Marking Report

Overview

There are 3 main assessment criteria in this assignment:

Marking criterion 6.1(Outputs From Tasks 1 to 3) focuses on the *correctness of logic*.
Marking criterion 6.2(Quality Of Graph And Its Explanation) focuses on *data analysis skills*.
Marking criteria 6.3(Quality Of Code Structure And Style) and **6.4**(Quality Of Documentation) focus on the *programming style*.

Before applying the late submission penalty, there are more than *35%* of the students getting HD and more than *65%* of the students getting D or above overall.

Marking criterion 6.1

Marking Process

The programming script submitted is run on the testing file, which is another transcript with a highly similar format.

If the programming script **can** be run successfully, the files generated are then evaluated by an auto-marking mechanism. Non-consequential marking has been introduced in the mechanism, i.e., the output of Task 2 is evaluated based on the file generated in Task 1 instead of the gold truth. With this mechanism, the outputs can be marked independently such that the mistakes in the previous tasks would not affect the mark of the current task.

If the programming script **cannot** be run successfully, the teaching team will mark the submission based on the programming script. The score is given based on the correctness of logic in the programming script. However, penalties are applied.

Common Issues

Hardcoding is one of the most serious issues found in the submissions. For example, some students have pre-defined a set of roles(e.g. Chandler, Monica, Ross, Rachel, Joey, Phoebe etc.). However, these characters may not appear in the testing file.

Another example is some submissions try to detect where the dialogue starts based on the whole sentence(e.g. [Scene: Central Perk, Ross, Chandler, and Phoebe are there. Joey is working.]). However, this line may not appear in the testing file as well.

Incorrect output file format is another common issue found.

For Task 1, from the specification, it says “*Save the returned* ***list*** *to …*”. However, some students have saved one tuple per line in the output *txt* files. Some even generate non-parsable outputs. For the Task 1 output format, please refer to the Assignment 2 FAQ post on the ED forum.

For Task 3, many outputs generated do not follow the schema provided in the specification. The column names should be *role, word and freq* in lowercase. Some submissions also do not lower-case the character names.

Marking criteria 6.2 - 6.4

Feedback from Deep

Common mistakes:

1. Class concept is used incorrectly for some students and some students have not used it at all. Some of the students do not even use the "self" or first argument of the class method properly.
2. Some of the students have not provided any comments or if they provided, then they have only a couple of incomplete statements. And they did not use docstring and/or provide header comments.
3. Variable names are not properly provided. used like a,b,f,i,j,c1,.... etc.
4. They are not aware or did not read enough about the inbuilt functions. which lead them to use nested loops at places in their code.
5. Lack of proper justification and observation from the graph.
6. Did not follow the instruction for the output files as well.

Feedback from Jiang

Common mistakes:

1. Use meaningless names for variables, functions and classes
2. Missing the header comments, some inline comments are not detailed enough to explain the logic
3. Missing x-label, y-label or title when plotting.

Suggestions:

1. Practice more to design the classes and functions
2. Always add the header comments for each script file

Feedback from Minfeng

Some common mistakes:

1. The class concept is incorrectly used. Even two classes are defined, but most people don't know how to use them.
2. Hardcoding is a common issue.
3. Most assignments lack header comments.

Suggestions:

1. See more examples using class, try to understand what's the benefits of using class.
2. There is no other way that can help improve your code logic except more practice.
3. Keep a good habit of writing header comments.

Feedback from Shahriar

Common mistakes:

1. Missing header comments
2. Hardcoding character names
3. Insufficient justification of graph/observation of results
4. Inconsistency in naming style
5. Class design not robust enough

Feedback from Xinyu

Common mistakes:

1. Hard code the characters
2. Variables do not have meaningful names
3. Most students understand how to draw the graph separately, but when put everything together, the layout looks messy.
4. For some students, the code structure is not clear. All the code is written together without using any functions or classes.

Feedback from Yidan

Common mistakes:

1. Missing appropriate justification and observation for the graph.
2. Meaningless variable names
3. Missing header comments
4. Fail to use class methods to perform tasks.

Suggestions:

1. Improve the ability to justify the reason for choosing a graph and summarise significant information from the graph.
2. Keep a good habit of writing header comments and using meaningful variable names
3. Practice more to use class and class methods to solve problems.