Assignment 2 (A2)

Due date: Friday, June 23, 2023 (11:59pm)
Submission via Git only

Contents

Programming environment

For this assignment you must ensure your code executes correctly on the *reference platform* (i.e., computers on ELW B238, which can be accessed using ssh) you configured as part of Lab 01. This same environment will also be used by the teaching team when evaluating your submitted work. You will have to make sure that your program executes perfectly on the reference platform. If your programs do not run on reference platform, your submission will receive 0 marks.

All test files and sample code for this assignment are available in the 'a2' folder of your git repository and you must use the git pull command to download a copy of the files:

```
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/

Learning objectives

- Learn or review basic features of the Python 3 programming language.
- 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.

music_manager.py: Descriptive analytics about songs data

a) In A2, music_manager is a small program that uses relevant Python structures and libraries to process songs data to produce descriptive analytics.

b) Based on provided arguments and data files (i.e., datasets), music_manager will help us answer the following questions:
   a. **Q1**: What are the top 10 songs released in 1999 with most 'popularity'?
   b. **Q2**: What are the top 5 songs released in 1999 with most 'energy'?
   c. **Q3**: What are the top 3 songs released in 1999 with most 'danceability'?
   d. **Q4**: What are the top 3 songs released in 2009 with most 'popularity'?
   e. **Q5**: What are the top 5 songs released in 2019 with most 'danceability'?
   f. **Q6**: What are the top 5 songs released in 1999 or 2009 with most 'energy'?
   g. **Q7**: What are the top 10 songs released in 1999, 2009, or 2019 with most 'popularity'?

c) After each execution, your program **must produce a file named output.csv** that represents the answer to the question asked to program (i.e., arguments passed). The data in this file should follow a descending order based on a required column (i.e., 'popularity', 'energy', or danceability'). If two songs share the same value for the sorting column used, the items should follow a lexicographical order based on the 'song' column. This means that the (descending) order of elements (rows) in output.csv is determined by:
   a. 'popularity', 'energy', or 'danceability'
   b. Lexicographical order in 'song' (descending)

d) The most reliable way (and the only one encouraged) to test your program is to use the provided **tester** file which will validate the output produced by your program, given a particular question.

Restrictions

- Your program must run as a script.
- Your program must be decomposed into easy-to-understand components. 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 the reference platform**. As part of the installation and configuration of the reference platform, we included relevant libraries for managing data such as numpy and pandas. **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 (**music_manager.py**) for this assignment. In Assignment 4 we will use the multi-module features of Python.

Testing your solution

- This time the **tester** file will execute your program automatically. You'll only need to pass the number of the question 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 to submit

A single Python source file named "music_manager.py" submitted (i.e., Git push to your remote repository) to the a2 folder in your repository.

Grading

Assignment 2 grading scheme is as follows. In general, straying from the assignment requirements might result in zero marks due to automated grading.

Scheme

- **(50%) # Tests Passed**
- **(50%) Code Qualitative Assessment**
    - **(20%) Functional decomposition, program-scope or file-scope variables:** quality coding requires the good use of functions. Code that relies on few large functions to accomplish its goals is considered poor-quality code. Typically, a good program has a main function that does some basic tasks and calls other functions that do most of the work.

- **(20%) Code structure:** The code style must be consistent, uniform, and facilitate readability throughout the file.
- **(15%) Proper naming conventions:** You must use proper names for functions and variables. Using random or single character variables is considered improper coding and significantly reduces code readability. Single character variables as loop variables is fine.
- **(15%) Typing:** to facilitate readability and maintainability in your solution, typing (i.e., type hints) is required for this assignment when declaring variables and functions.
- **(10%) Debugging/Comment artifacts:** You must submit a clean file with no residual commented lines of code or unintended text.
- **(10%) Documentation and commenting:** the purpose of documentation and commenting is to write information so that anyone other than yourself (with knowledge of coding) can review your program and quickly understand how it works. In terms of marking, documentation is not a large mark, but it will be part of the quality assessment.
- **(10%) Quality of solution:** marker will access the submission for logical and functional quality of the solution. Some examples that would result in a reduction of marks: solutions that read the input files several times, solutions that represent the data in inappropriate data structures, solutions which scale unreasonably with the size of the input.

Scale

A grade		B grade		C grade		D grade		F grade		
grade	marks	grade	marks	grade	marks	grade	marks	grade	marks	description
A+	90-100	B+	77-79							Either no submission given, or submission represents little work or none of the tests pass. No submission, 0 marks. Submissions that do not run, 0 marks. Submissions that fail all tests and show a poor to no effort (as assessed by the marker) are given 0 marks. Submissions that fail all tests, but represent a sincere effort (as assessed by the marker) may be given a few marks
A	85-89	B	73-76	C+	65-69					
A-	80-84	B-	70-72	C	60-64	D	50-59	F	0-49	

Input specification

- All input test files are in CSV format and are based on the “Top Hits Spotify from 2000-2019” dataset in Kaggle.com.
- The metadata (i.e., description of each column) for the input datasets can be found online in the ‘Content’ section of the dataset description.

Program Arguments

Argument	Description	Example (Test 07)
--sortBy	The column to use for sorting the songs	`./music_manager --sortBy=popularity --display=10 --files=top_songs_1999.csv,top_songs_2009.csv,top_songs_2019.csv`
--display	The number or songs that will be displayed	
--files	The files used to answer the question	

Output specification

- After execution, your program must produce\create the following file: **output.csv**. This file will contain four-dimensional data (i.e., a table with four columns) that represents the answer to the question passed as an argument to the program (e.g., **Q1**, **Q2**, **Q3**, **Q4**, **Q5**, **Q6**, **Q7**).
    - The first column refers to the name of the song's artist and must be named 'artist'.
    - The second column refers to the name of the song and must be named 'song'.
    - The third column refers to the release year of the song and must be named 'year'.
    - The fourth column refers to the column used to sort the data and must be named 'popularity', 'energy', or 'danceability'.