CP1404/CP1804/CP5632 2020 SP1 Assignment 1 – Songs to Learn 1.0

Task:

You are to plan and then code a console-based program in Python 3, as described in the following information and sample output. This assignment will help you build skills using selection, repetition, file input/output, exceptions, lists, functions and string formatting. Do not define any of your own classes or use code constructs that haven't been taught in this subject. Assignment 2 will build on this program with more advanced code constructs including dictionaries, classes and a Graphical User Interface (GUI).

Some requirements have in-text help references, like [0], that refer to the resources list near the bottom of this document. Please check these references to find help on that topic. ***Everything*** you need to complete this assignment can be found in the subject materials.

Start your work by clicking this link to create a new repository in GitHub Classroom:

https://classroom.github.com/a/po5jhJiW

Do not use any other repo or a copy of this one... just use this actual repository!
This will give you a new repo containing starter files including a README for your project, all of which you must use. Do not add any other files in this project, and do not rename anything - just use this as your assignment repo. Do not "download" this repo, but rather *checkout* this repo using PyCharm.

Program Overview:

This program is a simple song list that allows a user to track songs that they wish to learn and songs they have completed learning. The program reads and writes a list of songs in a file. Each song has:

- title, artist, year, whether it is learned (l) or unlearned (u)

Users can choose to see the list of songs, which should be sorted by artist then by title. [1]
Did you see that reference? Look at the references list below for reference [1] and you'll find out where we taught you how to sort lists by multiple keys.
Users can add new songs and mark songs as learned.
They cannot change songs from learned to unlearned.

Program Functionality Details:

Ensure that your program has the following features, as demonstrated in the sample output below. Your program should:

- display a welcome message with your name in it
- display a menu for the user to choose from [2, 3]
- return to the menu after each action and loop until the user chooses to quit
- load a CSV (Comma Separated Values) file of songs (just once at the very start); a sample CSV file is provided for you and you must use this format [4] (note: you're not expected to use the csv module, but you're welcome to)
- *when the user chooses* ***list****:* display a neatly formatted (lined up) list of all the songs with their details (unlearned songs have an * next to them) and a count of these songs [5] (note: you are welcome to either guess or calculate the size of the title and artist fields to line them up - either way is fine)
- *when the user chooses* ***add****:* prompt for the song's title, artist and year, error-checking each of these [3, 5], then add the song to the song list in memory (not to the file); new songs are always unlearned
- *when the user chooses to* ***complete a song****:* allow the user to choose one song by number (error-checked), then change that song's status to learned

    - if no songs are unlearned, then display a "No more songs to learn!" message
- *when the user chooses **quit**:* save the songs to the CSV file, overwriting file contents

Coding Requirements and Suggestions:

- Work incrementally on this task: complete small parts of it at a time rather than trying to get it all working at once.
- You are assessed on your use of version control including commits and commit messages, so please commit regularly (each logical chunk or milestone) and use meaningful commit messages in the imperative voice, as taught in class. Your commits should show steady work completed over reasonable time, not all in a short period.
- Edit the module docstring at the very top of your code file to contain your own details.
- Make use of named constants as appropriate (e.g. for the characters that represent the song's learned/unlearned status).
- Use functions appropriately for each significant part of the program: this is the divide-and-conquer problem-solving approach. Follow the principles you've learned about functions, including the single responsibility principle (SRP).
- For efficiency, you should ***only load the songs file once*** – when the program starts.
- **Only save the songs file once** – when the user quits.
- **Store the song data in a list of lists** and pass this to any functions that need access to it. Note: this variable should *not* be global. The only global variables you may have are CONSTANTS. Do not store a song's index – this is just its position in the list.
- The menu should handle both uppercase and lowercase letters as valid choices.
- Use exception handling as appropriate to deal with input errors (including when entering numbers and selecting songs) [3]. You should be able to use generic, customisable functions to perform input with error checking (e.g. getting the title and artist can reuse the same function).

Check the rubric carefully to understand how you will be assessed. There should be no surprises here – this is about following the best practices we have taught in class.

Output Requirements:

Sample output from the program is provided below. **Ensure that your program matches this, including spaces, spelling, and the formatting of the song lists.** Think of this as helpful guidance as well as training you to pay attention to detail. The sample output is intended to show a large (but not exhaustive) range of situations including user input error handling.

Submission:

Submit a zip file (not a rar or anything other than a zip) containing your project, including all code (.py, .csv), your project reflection README, PyCharm project files and the .git directory (just zip up your project/repo directory). Do NOT include a venv. (If you have setup your project correctly as instructed, you will not have a venv folder in it.) Name your zip file like: **FirstnameLastnameA1.zip**. Upload your single zip file on LearnJCU under Assessment.

Due:

Submit your assignment by the date and time specified on LearnJCU. Submissions received after this date will incur late penalties as described in the subject outline.

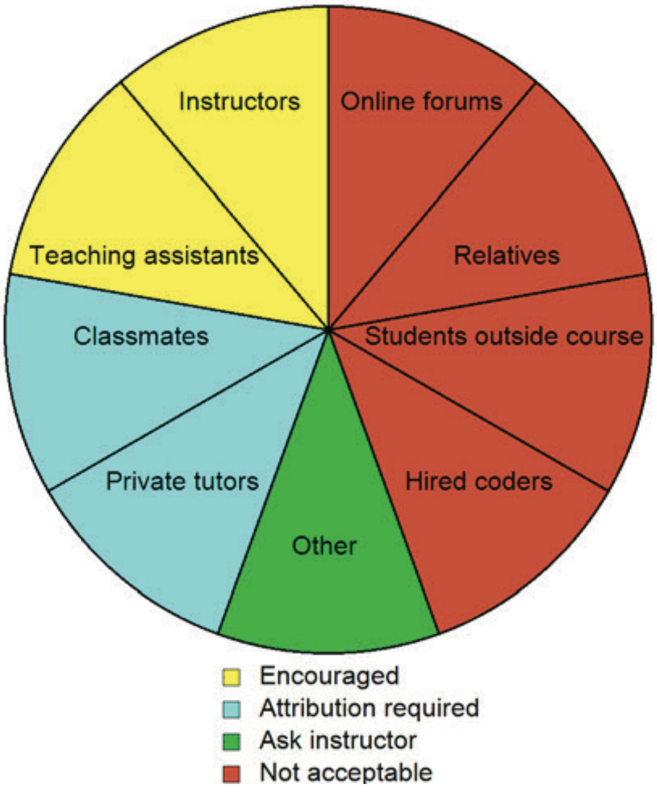
Integrity:

The work you submit for this assignment must be your own. Submissions that are detected to be too similar to that of another student or other work (e.g. code found online) will be dealt with according to the College procedures for handling plagiarism and may result in serious penalties.

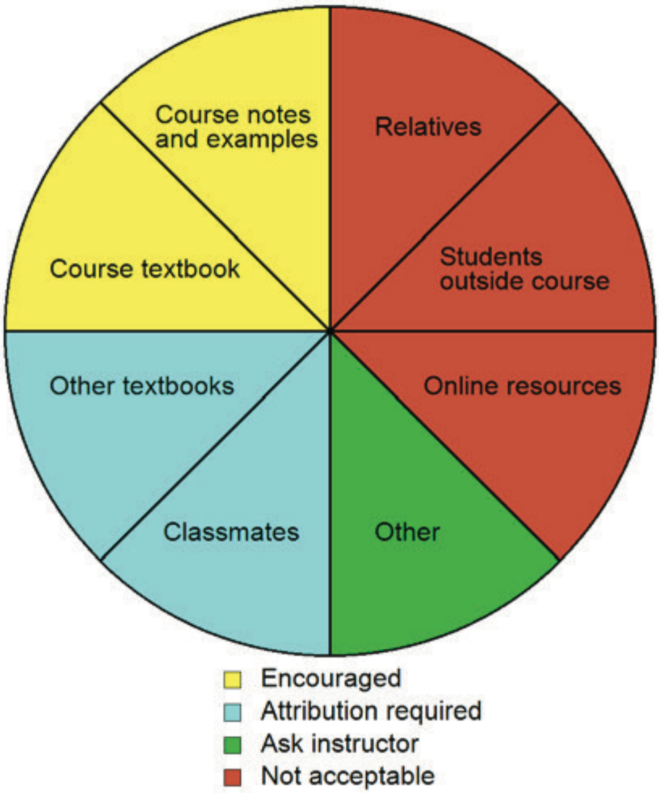
The goals of this assignment include helping you gain understanding of fundamental programming concepts and skills, and future subjects will build on this learning. Therefore, it is important that you develop these skills to a high level by completing the work and gaining the understanding yourself. You may discuss the assignment with other students and get assistance from your peers, but you may not do any part of anyone else's work for them and you may not get anyone else to do any part of your work. Note that this means you should never give a copy of your work to anyone or accept a copy of anyone else's work, including looking at another student's work or having a classmate look at your work. If you require assistance with the assignment, please ask **general** questions on the discussion forum, or get **specific** assistance with your own work by talking with your lecturer or tutor.

The subject materials (lecture notes, practicals, textbook and other guides provided in the subject) contain all of the information you need for this particular assignment. You should not use online resources (e.g. Stack Overflow or other forums) to find resources or assistance as this would limit your learning and would mean that you would not achieve the goals of the assignment - mastering fundamental programming concepts and skills.

Assistance: Who can you get help from? Use this diagram to determine from whom you may seek help with your programs. [6]



Resources: Where can you get code from? Use this diagram to determine where you may find code to use in your programs.



Sample Output:

The following sample run was made using a CSV file that contained:

```
Heartbreak Hotel,Elvis Presley,1956,u
I Want to Hold Your Hand,The Beatles,1964,u
My Sharona,The Knack,1979,l
Macarena,Los Del Rio,1996,l
Boom Boom Pow,The Black Eyed Peas,2009,u
Somebody That I Used to Know,Gotye featuring Kimbra,2012,u
```

Bold green text below shows user input for this sample run.

```
Songs to Learn 1.0 - by Lindsay Ward
6 songs loaded
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> l
 0. * Heartbreak Hotel               - Elvis Presley             (1956)
 1. * Somebody That I Used to Know   - Gotye featuring Kimbra    (2012)
 2.   Macarena                       - Los Del Rio               (1996)
 3. * I Want to Hold Your Hand       - The Beatles               (1964)
 4. * Boom Boom Pow                  - The Black Eyed Peas       (2009)
 5.   My Sharona                     - The Knack                 (1979)
2 songs learned, 4 songs still to learn
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> c
Enter the number of a song to mark as learned
>>> blah
Invalid input; enter a valid number
>>> -1
Number must be >= 0
>>> 6
Invalid song number
>>> 2
You have already learned Macarena
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> C
Enter the number of a song to mark as learned
>>> 0
Heartbreak Hotel by Elvis Presley learned
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> L
 0.   Heartbreak Hotel               - Elvis Presley             (1956)
 1. * Somebody That I Used to Know   - Gotye featuring Kimbra    (2012)
 2.   Macarena                       - Los Del Rio               (1996)
 3. * I Want to Hold Your Hand       - The Beatles               (1964)
 4. * Boom Boom Pow                  - The Black Eyed Peas       (2009)
 5.   My Sharona                     - The Knack                 (1979)
3 songs learned, 3 songs still to learn
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> A
Title:
Input can not be blank
```

```
Title: Let It Be
Artist: 
Input can not be blank
Artist: The Beatles
Year: -12
Number must be >= 0
Year: why?
Invalid input; enter a valid number
Year: 1970
Let It Be by The Beatles (1970) added to song list
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> L
 0.   Heartbreak Hotel               - Elvis Presley             (1956)
 1. * Somebody That I Used to Know   - Gotye featuring Kimbra    (2012)
 2.   Macarena                       - Los Del Rio               (1996)
 3. * I Want to Hold Your Hand       - The Beatles               (1964)
 4. * Let It Be                      - The Beatles               (1970)
 5. * Boom Boom Pow                  - The Black Eyed Peas       (2009)
 6.   My Sharona                     - The Knack                 (1979)
3 songs learned, 4 songs still to learn
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> c
Enter the number of a song to mark as learned
>>> 1
Somebody That I Used to Know by Gotye featuring Kimbra learned
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> c
Enter the number of a song to mark as learned
>>> 3
I Want to Hold Your Hand by The Beatles learned
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> C
Enter the number of a song to mark as learned
>>> 4
Let It Be by The Beatles learned
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> L
 0.   Heartbreak Hotel               - Elvis Presley             (1956)
 1.   Somebody That I Used to Know   - Gotye featuring Kimbra    (2012)
 2.   Macarena                       - Los Del Rio               (1996)
 3.   I Want to Hold Your Hand       - The Beatles               (1964)
 4.   Let It Be                      - The Beatles               (1970)
 5. * Boom Boom Pow                  - The Black Eyed Peas       (2009)
 6.   My Sharona                     - The Knack                 (1979)
6 songs learned, 1 songs still to learn
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> c
Enter the number of a song to mark as learned
>>> 5
Boom Boom Pow by The Black Eyed Peas learned
Menu:
```

```
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> c
No more songs to learn!
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> j
Invalid menu choice
Menu:
L - List songs
A - Add new song
C - Complete a song
Q - Quit
>>> Q
7 songs saved to songs.csv
Have a nice day :)
```

At the end of this run, the saved CSV file contained:

```
Heartbreak Hotel,Elvis Presley,1956,l
Somebody That I Used to Know,Gotye featuring Kimbra,2012,l
Macarena,Los Del Rio,1996,l
I Want to Hold Your Hand,The Beatles,1964,l
Let It Be,The Beatles,1970,l
Boom Boom Pow,The Black Eyed Peas,2009,l
My Sharona,The Knack,1979,l
```

References – Resources from Subject Materials:

Selected subject materials are referenced here to help you find guidance for specific parts of the assignment (e.g. sorting a list by multiple values is covered in [1] and you will find a pattern for exception handling for error-checking in [3]). General references are not listed specifically but should be obvious (e.g. file input/output is covered in the lecture and practical on files).

1. itemgetter from Chapter 7 - Lists and Tuples.
2. Practical 01 - PyCharm, Control.
3. Programming Patterns. https://github.com/CP1404/Starter/wiki/Programming-Patterns
4. Chapter 5 - Files and Exceptions 1.
5. Practical 02 - Strings, Files, Exceptions.
6. Negotiating the Maze of Academic Integrity in Computing Education. https://dl.acm.org/citation.cfm?doid=3024906.3024910

Marking Scheme:

Ensure that you follow the processes and guidelines taught in class in order to produce high quality work. Do not just focus on getting the program working.
This assessment rubric provides you with the characteristics of exemplary to very limited work in relation to task criteria.

Criteria	Exemplary (9, 10)	Good (7, 8)	Satisfactory (5, 6)	Limited (2, 3, 4)	Very Limited (0, 1)
Correctness *Worth double*	Program works correctly for all functionality required.	Exhibits aspects of exemplary (left) and satisfactory (right)	Program mostly works correctly for most functionality, but there is/are some required aspects missing or that have problems.	Exhibits aspects of satisfactory (left) and very limited (right)	Program works incorrectly for all functionality required.
Error checking	Invalid inputs are handled well using exceptions and control logic as instructed, for all user inputs.		Invalid inputs are mostly handled correctly as instructed, but there is/are some problem(s), e.g. exceptions not well used.		Error checking is not done or is very poorly attempted.
Similarity to sample output (including all formatting)	All outputs match sample output perfectly, or only one minor difference, e.g. wording, spacing.		Multiple differences (e.g. typos, spacing, formatting) in program output compared to sample output.		No reasonable attempt made to match sample output. Very many differences.
Identifier naming	All function, variable and constant names are appropriate, meaningful and consistent.		Multiple function, variable or constant names are not appropriate, meaningful or consistent.		Many function, variable or constant names are not appropriate, meaningful or consistent.
Use of code constructs	Appropriate and efficient code use, including good logical choices for data structures and loops, good use of constants, etc.		Mostly appropriate code use but with definite problems, e.g. unnecessary code, poor choice of data structures or loops, no use of constants.		Many significant problems with code use.
Use of functions	Functions and parameters are appropriately used, functions are well reused to avoid code duplication.		Functions used but not well, e.g. incorrect/missing parameters or calls, unnecessary duplication or main code outside main function.		No functions used or functions used very poorly.
Formatting	All formatting is appropriate, including correct indentation, horizontal spacing and consistent vertical line spacing. PyCharm shows no formatting warnings.		Multiple problems with formatting reduces readability of code. PyCharm shows formatting warnings.		Readability is poor due to formatting problems. PyCharm shows many formatting warnings.
Commenting	Helpful block/inline comments and meaningful docstrings for all functions, top docstring contains all program details (name, date, basic description, GitHub URL).		Comments contain some noise (too many/unhelpful comments) or some missing program details in top docstring or some inappropriate or missing block/inline comments.		Commenting is very poor either through having too many comments (noise) or too few comments.
Use of version control	Git/GitHub used effectively and the repository contains a good number of commits with good messages that demonstrate incremental code development.		Aspects of the use of version control are poor, e.g. not many commits, meaningless messages that don't represent valuable incremental development.		Git/GitHub not used at all.