CP1404/CP1804 2020 SP1 – Assignment 2 – Songs to Learn 2.0 – Part 1

JAMES COOK UNIVERSITY AUSTRALIA

This document is part 1 of your assignment (focused on classes and console program); not the complete assignment itself.
Part 2 will come later and will include full requirements, including the GUI details. ***Why?***

One of the most common problems with assignment 1 and with this assignment 2 in past semesters is students not reading and understanding the task before attempting it. Here are some wise words from students who reflected on their assignment 2. What can you learn from them?

- At about 4 hours in i thought i had finished, I went to go in and see if there was anything that i missed and found out that I hadnt used Classes at all
  I then had to go back through all my code and modify it around to fix it so that i used classes instead of lists
- I often found myself having to completely redesign sections of my code due to a requirement in the task sheet that I had not adhered to. ...
  it does indicate that perhaps the first thing I should do with assignments like this is have a very thorough read-through of what is required to the exact specifics. This would save me a great amount of time.

Don't be like the students above… Be like the one below:

- Breaking the problem down into smaller parts and tackling one issue at a time. Creating, then testing, then fixing, testing again, etc... before moving on to the next part. I find the success of the whole program can be hinged on having a good starting foundation, so it's crucial to have the basics correct.

So… Here are some highlights from the complete assignment specification that we want you to read before you get started. Read them again when you get the full details.

Task:

Create both a console program and a Graphical User Interface (GUI) program similar to your first assignment, using Python 3 and the Kivy toolkit, as described in the following information and accompanying screencast video. This assignment will help you build skills using **classes** and **GUIs** as well as giving you more practice using techniques like selection, repetition, exceptions, lists, file I/O and functions.

Some requirements have in-text help references, like [0], that refer to the resources list near the bottom. 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/3Z7-0lVm

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 code files and a README for your project reflection, all of which you must use.
You should clone this using PyCharm (VCS > Get from Version Control…) so that your PyCharm project matches your assignment repo.
Do not add other files, and do not rename anything - just use this as your assignment repo.
Do not "download" this repo, but rather ***checkout*** (get) this repo using PyCharm.

Classes:

The most important learning outcome of this assignment is to be able to use **classes** to create reusable data types that simplify and modularise your programs. The fact that you can use these classes in both console and GUI programs highlights this modularity.

It is important that you ***create these classes first* – before any code that requires them**. This is good coding practice. You should write and then test each method of each class – **one at a time – committing as you go** (e.g. you might commit each time you complete a method and its tests).

The starter code includes two files (test_song.py and test_songcollection.py) with incomplete code for testing your classes. **Complete these files with simple tests**, that you write as you develop your Song and SongCollection classes.

Do not change the existing tests… write code that makes these tests pass.

You may use `assert` as shown in lectures [1], or just very simple tests that print the results of calling the methods you are testing with expected and actual results [2].

Once you have written and tested your classes, you can then use the Song class in your console program.

We will assess your Git commit history to see (and mark) that you do these in an appropriate order, so make sure you write your classes, with tests, then the console program, *before* attempting any functionality for the GUI.

- Complete the **Song** class in song.py. This should be a simple class with the required attributes for a song and the standard methods: __init__ (constructor), __str__ (used when displaying song details in the status message), and:
    - two (not one or zero) methods to mark the song as learned or unlearned.

- Complete the **SongCollection** class in songcollection.py. It should contain a *single* attribute: a list of Song objects, and at least the following methods:
    - add song – add a single Song object to the song list attribute
    - get number of unlearned songs
    - get number of learned songs
    - load songs (from csv file into the list of Song objects)
    - save songs (from song list into csv file)
    - sort (by the key passed in, then by title) [3]

  Notice that you do not store additional attributes like the number of songs, because this information is easily derived from what you do store (just the songs) [4].

Console Program:

After you have written and tested your classes, rewrite your first assignment to make use of your new Song class. Start by copying the code from your first assignment into the existing a1_classes.py file and committing that. In the first assignment, each song was stored as a list. Modify your code so that each song is stored as an object of your new Song class.

You do *not* need to rewrite your first assignment in any other way, but you do need to get it working. We will only evaluate how you use the Song class in the console program.

Project Reflection:

It is important that you start developing good coding and working practices, so you are required to complete a short but thoughtful reflection on this project. When you have finished the assignment, complete the template provided in the README of your repository and reflect on what you learned regarding both coding and your development process.
Note that this is worth significant marks, so allocate significant time to it.
We expect answers that show some **detail and thought**, not just trivial statements. You may find it useful to keep notes as you develop your program.

Git/GitHub:

You must use Git version control with your project stored in the private repository on GitHub that will be created when you accept the GitHub Classroom invitation above. You are assessed on your use of version control including commits and commit messages, using the **imperative voice** (like "Add X" not "Added X"). [5]

Submission:

Submit a single zip file by uploading it on LearnJCU under Assessment (click on the title of the assignment). Your zip file should contain the entire project directory, including the .git directory (just zip up your project directory). Make sure your GitHub URL is included in main.py. Please name the zip file like: **FirstnameLastnameA2.zip.**

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:

There are more details in the full assignment specification, but for now, please ***understand***:

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**.

The subject materials (lecture notes, practicals, textbook, sample code 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. Look for the references, like [0], to know where to find help.

References – Resources from Subject Materials:

1. Lecture Notes for Chapter 15 - Testing.
2. Practical 06 - Classes. https://github.com/CP1404/Practicals/tree/master/prac_06
3. attrgetter from Chapter 11 - Classes
4. Programming Patterns. https://github.com/CP1404/Starter/wiki/Programming-Patterns
5. Lecture Notes for Version Control.

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 down 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)
Project reflection ***14%***	The project reflection is complete and describes development and learning well, shows careful thought, highlights insights made during code development.	Exhibits aspects of exemplary (left) and satisfactory (right)	Project reflection contains some good content but is insufficient in coverage, depth or insight.	Exhibits aspects of satisfactory (left) and very limited (right)	Many aspects of the project reflection are missing or could be improved.
Use of version control **9%**	Git/GitHub has been used effectively and the repository contains a good number of commits with good messages that demonstrate incremental code development **starting with classes and testing then console before GUI**.		Git/GitHub used but several aspects of the use of version control are poor, e.g. not enough commits, or meaningless messages that don't represent valuable incremental development in an appropriate order.		Git/GitHub not used.
Console program **9%**	Class is used correctly in console program.		Class is used in console program but not correctly.		Class is not used in console program.
Error handling **9%**	Errors are handled correctly and robustly as required.		Some errors are handled but not all, or errors are not handled properly.		No reasonable error handling.
Correctness ***14%***	GUI layout is correct and program works correctly for all functionality required.		Aspects of the GUI layout are incomplete or poorly done or there are significant problems with functionality required.		GUI layout is very poor or not done. Program works incorrectly for all functionality required.
Identifier naming **9%**	All function, variable and constant names are appropriate, meaningful and consistent.		Several 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 **9%**	Appropriate and efficient code use, including no unnecessary duplication, good logical choices for control and storage, good use of constants, no global variables, good use of functions in main app, etc.		Several problems, e.g. unnecessary duplication, poor control, no use of constants, improper use of global variables, poor use of functions in main app.		Many problems with code use.
Use of classes and methods **9%**	Classes and methods are used correctly as required. Method inputs and outputs are well designed.		Some aspects of classes and methods are not well used, e.g. methods not used where they should be, problems with method/parameter design, incorrect use of objects.		Classes and methods used very poorly or not used at all.
Commenting **9%**	Code contains helpful # block comments, all classes and methods have meaningful docstrings and main module docstring contains all details (name, date, basic description, GitHub URL).		Comments are reasonable, but some classes and methods have no docstrings, and/or there is some noise (too many comments), and/or missing details in main module docstrings.		Commenting is very poor or not done.
Formatting **9%**	All formatting is appropriate, including indentation, horizontal spacing and vertical line spacing. PyCharm shows no formatting warnings.		Problems with formatting reduces readability of code. PyCharm shows multiple formatting warnings.		Readability is poor due to formatting problems. PyCharm shows many formatting warnings.