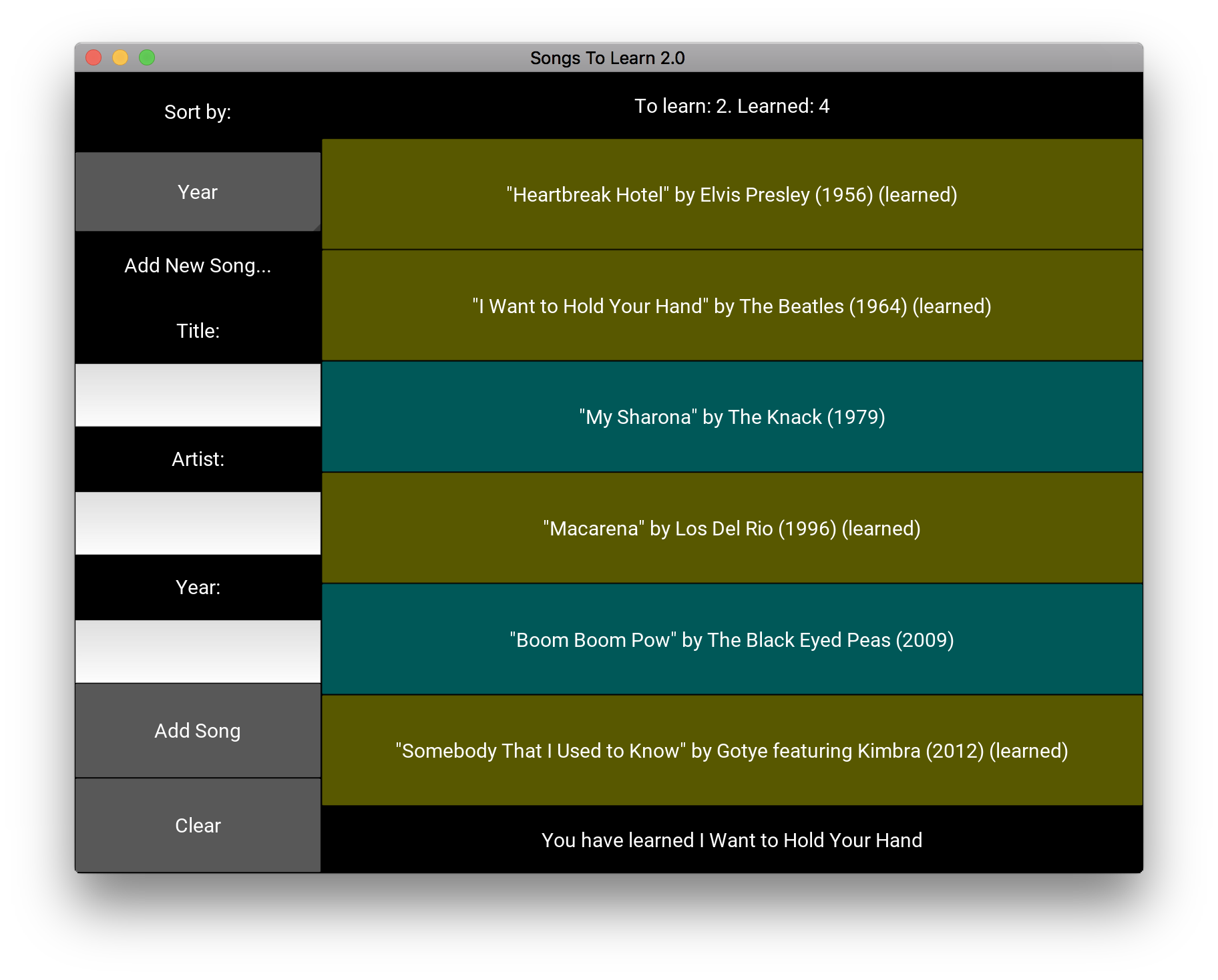
# JCU_Logo_RGBCP1404 SP53-2018 Assignment 2 – Songs To Learn 2.0

Task:

Create 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. 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 know to complete this assignment can be found in the subject materials.



Program Overview:

Ensure that your program GUI has the following features, as demonstrated in the screenshots and accompanying screencast:

* the left side of the screen contains a drop-down "spinner" for the user to choose the song sorting (spinner\_demo from [[1](#_ENREF_1)]), and text entry fields for inputting information for a new song
* the right side contains buttons for the songs, colour-coded based on whether they are learned or not
* the status bar at the top of the right side shows the number of songs learned and still to learn
* the status bar at the bottom of the right side shows messages about the state of the program, including updating when a song is clicked on
* the user can add a new song by entering text in the input fields and clicking “Add Song”
* the exact style (including colours) is up to you, but ensure that all functionality is readily accessible with your chosen GUI style

Program Functionality Details:

* Complete the main program in a Kivy App subclass in main.py. There will be no main() function, but rather your program will run() the Kivy app in the same way as you've seen in our example programs. [[1](#_ENREF_1), [2](#_ENREF_2)]
* The program should start by loading the same CSV file of songs as with your first assignment. This must be done with a method of your main app class and will save the songs as Song instances in a SongList instance (see below for details).
* The songs file must be saved when the program ends, updating any changes made with the app by the user (adding new songs or marking them as learned).

**Adding:**

* All song fields are required. If a field is left blank, the bottom status bar should display “**All fields must be completed**” when “Add Song” is clicked.
* The year field must be a valid integer. If this is invalid, the status bar should display “**Please enter a valid number**”.
* Pressing the Tab key should move between the text fields. (popup\_demo from [[1](#_ENREF_1)])
* When the user successfully adds a song, the fields should be cleared and the song should appear in the songs list on the right. (dynamic\_widgets from [[1](#_ENREF_1)])
* When the user clicks the “Clear” button, all text in the input fields and the status bar should be cleared.

Coding Requirements:

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

<https://classroom.github.com/a/6NYBOh41>

You should clone this using PyCharm so that your PyCharm project matches your repo. This will give you a new repo containing starter code files and a README for your reflection. Do not add any other files in this project, and do not rename anything - just use this as your assignment repo.  
***Do not use any other repo or a copy of this one... just use this actual repository!***

* At the very top of your main.py file, complete the comment containing your details.
* Make use of named constants where appropriate. E.g. colours could be constants.
* Use functions/methods appropriately for each significant part of the program. Remember that functions should follow the Single Responsibility Principle.
* Use exception handling where appropriate to deal with input errors.
* Complete your GUI design using the kv language in the app.kv file. Creating the song buttons should be done from Python (main.py), not in the kv file, since it will be dynamic. (dynamic\_widgets from [[1](#_ENREF_1)])
* Document your classes and methods clearly with docstrings. Include inline/block comments as appropriate. You do not need comments in the kv file (but you can use them if you wish).
* You do not need to submit any pseudocode (but you can write some if you wish).

Classes

One of the most important parts of this assignment is to learn how to use classes to create reusable data types that simplify and modularise your program. In future, you will design your programs by choosing your own classes, but for now we will tell you what classes to create. 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.

The starter code that you get in your repository when you start your work with GitHub includes two files (test\_song.py and test\_songlist.py) with incomplete code for testing your classes. Complete these files with simple tests, that you write as you develop your Song and SongList classes. You may use things like assert and doctest as shown in lectures [[3](#_ENREF_3)], or just very simple tests that print the results of calling the methods you are testing. **You are marked on the process of doing these in an appropriate order, based on your Git commit history, so make sure you write your classes, with tests, before attempting any functionality (the main program) that needs to use the classes.**

* 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).  
  If you like, you may use two methods to mark the song as required/learned.
* Complete the **SongList** class in songlist.py. It should contain a single attribute: a list of Song objects, and at least the following methods:
  + get song by title – take in a title (string) and return the Song object with that title; this will be useful when handling song button clicking
  + add song – add a single Song object to the song list attribute
  + get number of required songs
  + get number of learned songs
  + load songs (from csv file into Song objects in the list)
  + save songs (from song list into csv file)
  + sort (by the key passed in, then by title) [[4](#_ENREF_4)]

GUI Requirements:

The functionality can be achieved with a variety of GUI styles and colour schemes. You are welcome to customise the GUI, but it should do everything required and match any constraints specified.

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. 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.

**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](#_ENREF_5)]

Submission:

Submit a zip file containing the entire project directory, including all code (.py, .kv, .csv), your project reflection README, PyCharm project files and the .git directory (just zip up your project/repo directory). Please name the file like: **FirstnameLastnameA2.zip** e.g. if your name were Miles Davis, the filename would be MilesDavisA2.zip. Submit your single zip file by uploading it on LearnJCU under Assessment (click on the title of the assignment).

Due:

Submit your assignment bythe 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 will be dealt with according to the College procedures for handling plagiarism and may result in serious penalties.

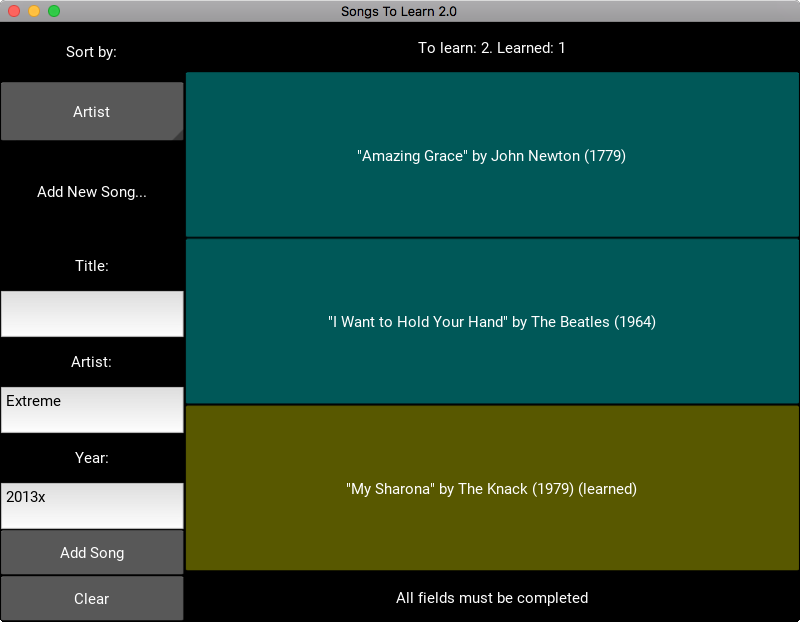
The goals of this assignment include helping you gain understanding of fundamental programming concepts and skills, and future subjects will build on this learning. Therefore, it is important that you develop these skills to a high level by completing the work and gaining the understanding yourself. 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](#_ENREF_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:

Screenshots have been provided here. In addition, you should study the **screencast** provided with this assignment to see how the GUI program should work, including what the messages should be and when they occur.



References – Resources from Subject Materials:

1. KivyDemos. <https://github.com/CP1404/KivyDemos>

2. Kivy Lecture Notes.

3. Chapter 15 - Testing.

4. attrgetter from Chapter 11 - Classes

5. Version Control Lecture Notes.

6. Simon, J. Sheard, M. Morgan, A. Petersen, A. Settle, J. Sinclair, G. Cross and C. Riedesel. Negotiating the Maze of Academic Integrity in Computing Education. 2016

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)** |
| **Thoughtful and useful project reflection**  ***Worth double*** | 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** | 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**. | 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. |
| **GUI layout** | GUI layout is well constructed, and contains all the required widgets. | Multiple aspects of the GUI layout are incomplete or poorly done. | GUI layout is very poor or not done. |
| **Error handling** | 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**  ***Worth double*** | Program works correctly for all other functionality required. | There are some significant problems with other functionality required. | Program works incorrectly for all other functionality required. |
| **Identifier naming** | 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** | Appropriate and efficient code use, including no unnecessary duplication, good logical choices for control and storage, good use of constants, no global variables etc. | Several problems, e.g. unnecessary duplication, poor control, no use of constants, improper use of global variables. | Many problems with code use. |
| **Use of methods in main Kivy app** | Methods and parameters in main app are appropriately used with good design including good reuse. | Some problems with methods, e.g. unsuitable parameters or calls, global code, or not designing methods for reuse. | No methods used or methods used very poorly. |
| **Use of classes and methods in Song and SongList** | 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** | 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** | 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. |