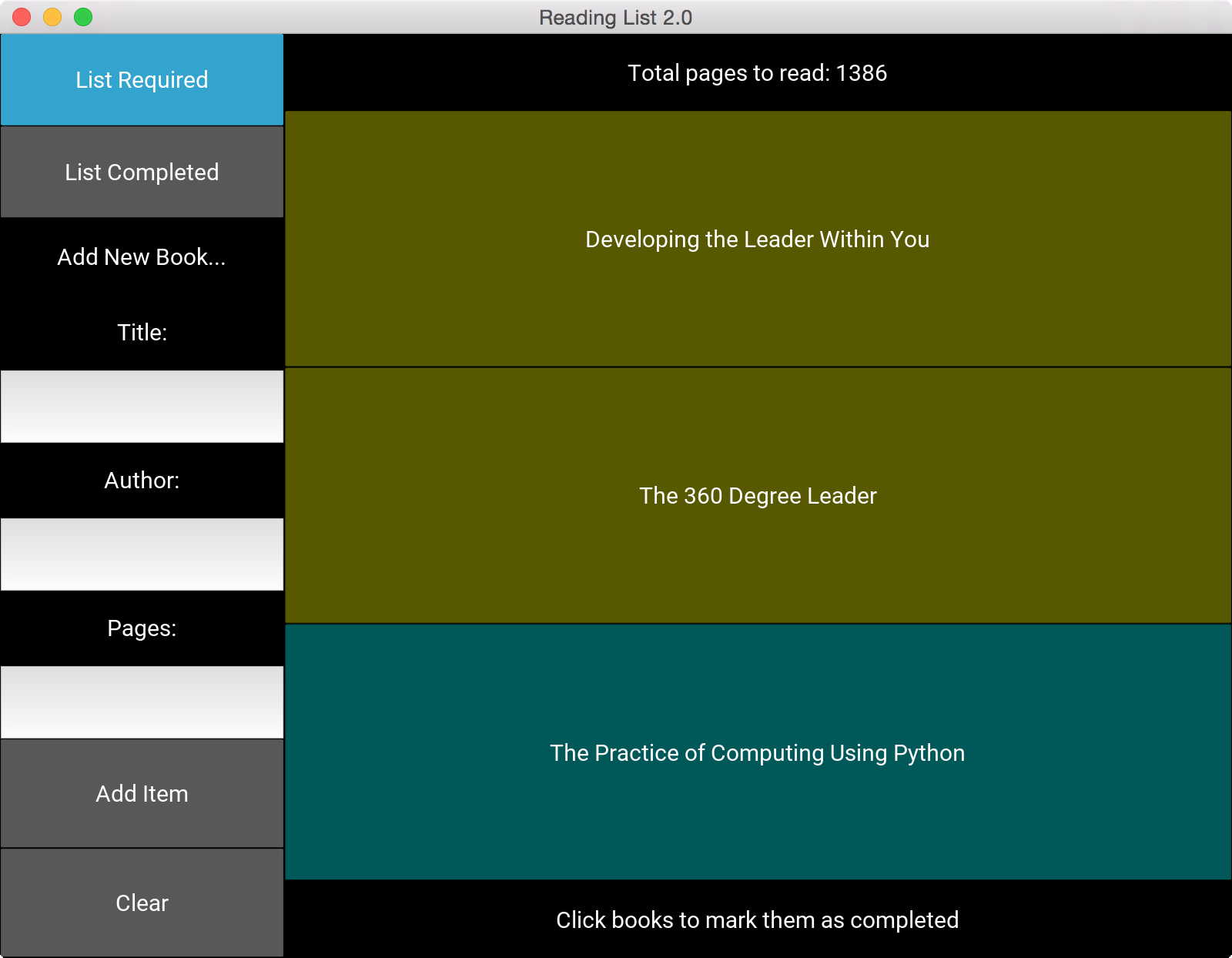
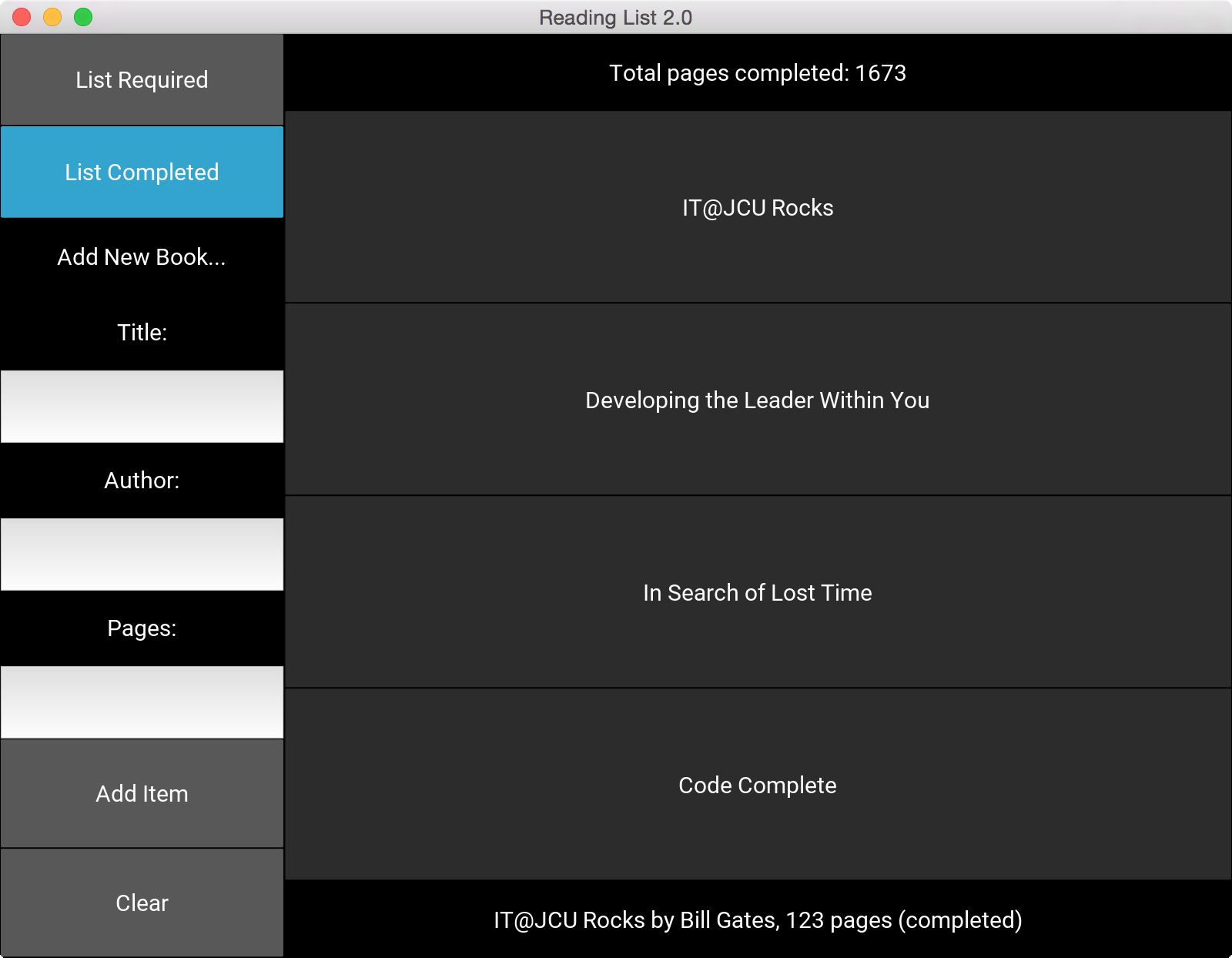
# JCU_Logo_RGBCP1404/CP5632 2016 SP53 Assignment 2 – Reading List 2.0

Task:

Create a Graphical User Interface (GUI) version of the program you made in assignment 1, 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 references like [0] that refer to the resources list near the bottom. 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:

* the left side of the screen contains buttons for the user to choose actions from, and text entry fields for inputting information for a new book
* the right side contains buttons for each of the books, colour-coded based on their length
* the status bar at the top of the right side shows the number of pages to read or completed
* the status bar at the bottom of the right side shows messages about what to do, or when in list completed mode, shows the book's details when a book is clicked on
* “List Required” is the default (starting) state and shows the books that are required; clicking an item in this state marks it as completed (the button will immediately disappear)
* “List Completed” changes the right side view to show books that have been completed (no length-based colouring); clicking an item in this state shows its details in the bottom status bar
* the user can add a new book by entering text in the input fields and clicking “Add Book”
* 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:

* The main program will be done 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 starts by loading the same CSV (Comma Separated Values) file of items as in your first assignment. This should be done with a method of your main app class and will be similar to the code for the first assignment, but will save the books as Book instances in a BookList (see below for more details).
* The books file is saved when the program ends, updating any changes made with the app by the user (adding new books or marking them as completed).

**Adding:**

* All fields are required. If a field is left blank, the bottom status bar should display “**All fields must be completed**” when “Add Book” is clicked.
* The pages 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 book, the fields should be cleared and the book should appear in the required reading list. (dynamic\_widgets from [[1](#_ENREF_1)])
* When the user clicks the “Clear” button, all text in the input fields should be cleared.

Coding Requirements:

* Start your work by clicking cloning this repository:

<https://github.com/yokyen/a2-starter>  
You should clone this using PyCharm. After cloning, you could create the repository in your Github.

From PyCharm, Select VCS -> Import Into Version Control -> Share Project on Github.

\*\* Ensure the repository is private upon creation.

This will give you a new repo containing starter code files and a **README** for your **reflection**.

* 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 “do one thing”.
* Use exception handling where appropriate to deal with input errors.
* Use app.kv file for your GUI design. Creating the book 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 if you wish).
* You do not need any pseudocode (but you can write some if you wish).

Classes

A significant part 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.

Note: some incomplete test code (in test\_book.py and test\_booklist.py) has been provided for you as a starting point. [[3](#_ENREF_3)]

* Use a class for **Book** (in book.py). This should be a simple class with the required attributes for a book and four methods: \_\_init\_\_, \_\_str\_\_ (used when displaying book details in the status bar) and:
  + a method to mark the book as completed
  + a method to determine if the book is long (500 or more pages)
* Use a class for **BookList** (in booklist.py). It should contain a single attribute, a list of Book objects, and at least the following methods:
  + get book by title – take in a title (string) and return the Book object with that title; this will be useful when handling book button clicking
  + add book – add a single Book object to the book list attribute
  + get total pages for required books
  + get total pages for completed books
  + load books (from csv file into Book objects in the list)
  + save books (from book list into csv file)
  + sort (by author then number of pages) [[4](#_ENREF_4)]
* The starter code includes two files with incomplete code for testing your classes. You are not marked on completing these, but it is good practice and very helpful to write tests as you develop your Book and BookList classes.

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 repo reflect on what you learned regarding both coding and your 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.

You are assessed on your use of version control including commits and commit messages, using 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. You are allowed to 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. Programs that are detected to be too similar to another student’s work will be dealt with promptly according to University procedures for handling plagiarism.

If you require assistance with the assignment, please ask usingthe appropriate channels, including talking with your lecturer or tutor.

Sample Output:

Screenshots have been provided above. 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.

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.

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 (4)** | **Good (3)** | **Satisfactory (2)** | **Limited (1)** | **Very Limited (0)** |
| **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 competent (right) | Project reflection contains some good content but is insufficient in coverage, depth or insight. | Exhibits aspects of competent (left) and unacceptable (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 number of commits with good messages that demonstrate incremental code development. | 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. | Git/GitHub not used. |
| **Program Execution**  **GUI layout and behaviour** | GUI layout is well constructed, and contains all of the required widgets, easy to use. All GUI widgets behave as required and are coded appropriately. | There are multiple aspects of the widgets, layout or behaviour that are missing, incomplete or poorly done. | The GUI layout and behaviour is very poor or not done. |
| **Correctness**  ***Worth double*** | Program works correctly for all functionality required. | There are some significant problems with program functionality. | Program works incorrectly for all functionality required. |
| **Quality of Code 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, etc. | Several problems, e.g. unnecessary duplication, poor control, no use of constants. | Many problems with code use. |
| **Use of functions** | Functions and parameters in main program are appropriately used with good design. | Minimal use of functions or other significant problems with functions, e.g. incorrect parameters or function calls or main code outside main function. | No functions used or functions used very poorly. |
| **Use of classes and methods** | 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. |
| **Formatting** | All formatting is appropriate, including correct indentation, horizontal spacing and consistent vertical line spacing. PyCharm shows no formatting warnings. | Significant problems with formatting reduces readability of code. PyCharm shows formatting warnings. | Readability is poor due to formatting problems. PyCharm shows many formatting warnings. |
| **Commenting** | Code contains inline comments where helpful, and meaningful function/method docstrings and top docstring containing program details (name, date, basic description, GitHub URL). | Comments are reasonable, but there is some noise (too many comments) or some missing program details in docstrings or some inappropriate or missing inline comments. | Commenting is very poor or not done. |