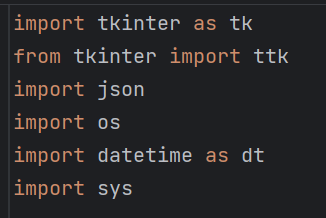
Name: Xuan Liao

Professor Janghoon Yang

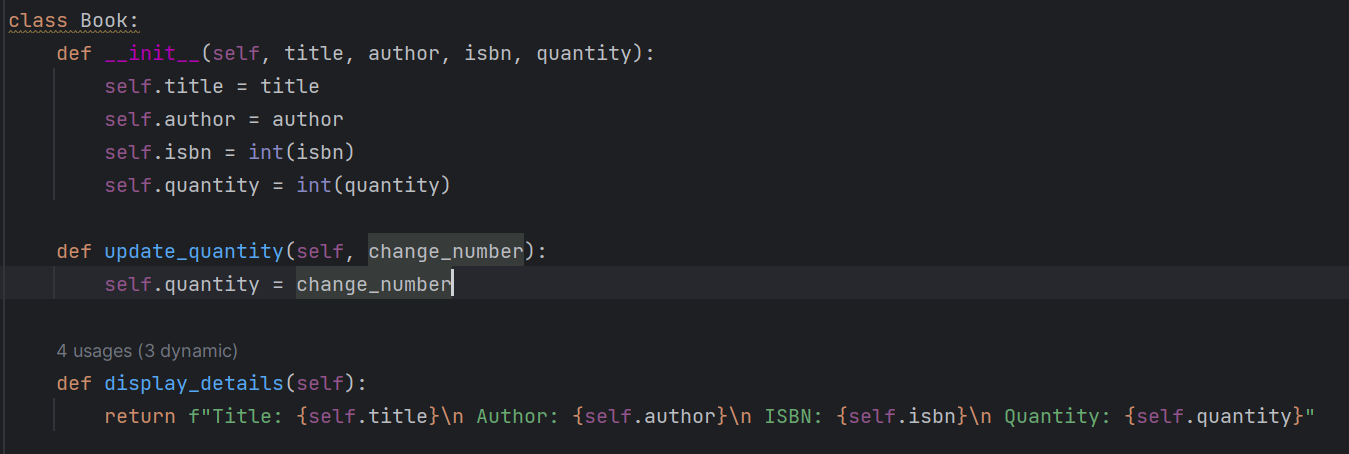
CMPSC 132

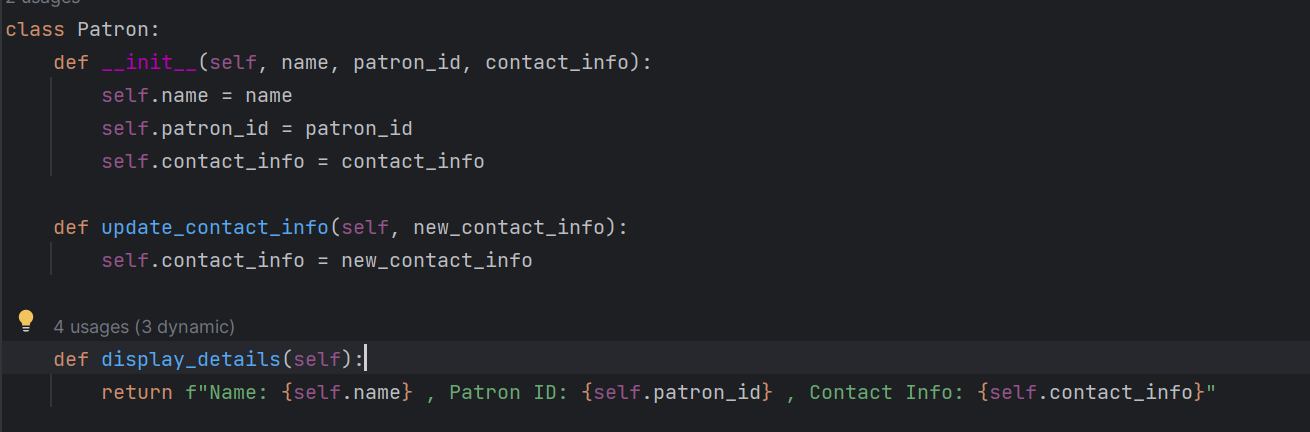
March 6, 2024

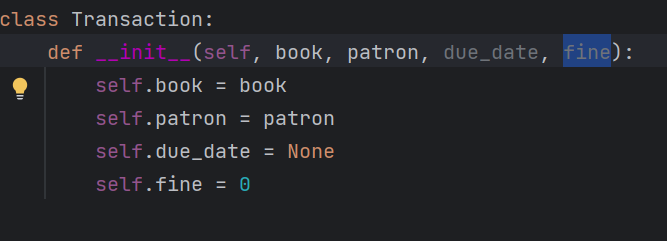
This project was centered around creating a program that acted as a virtual library. I liked the idea of making something like this because it would utilize much of what we covered in class thus far. As far as implementation goes, I was not sure how to use some of the more difficult aspects of object-oriented programming, but I tried my best to make something presentable.



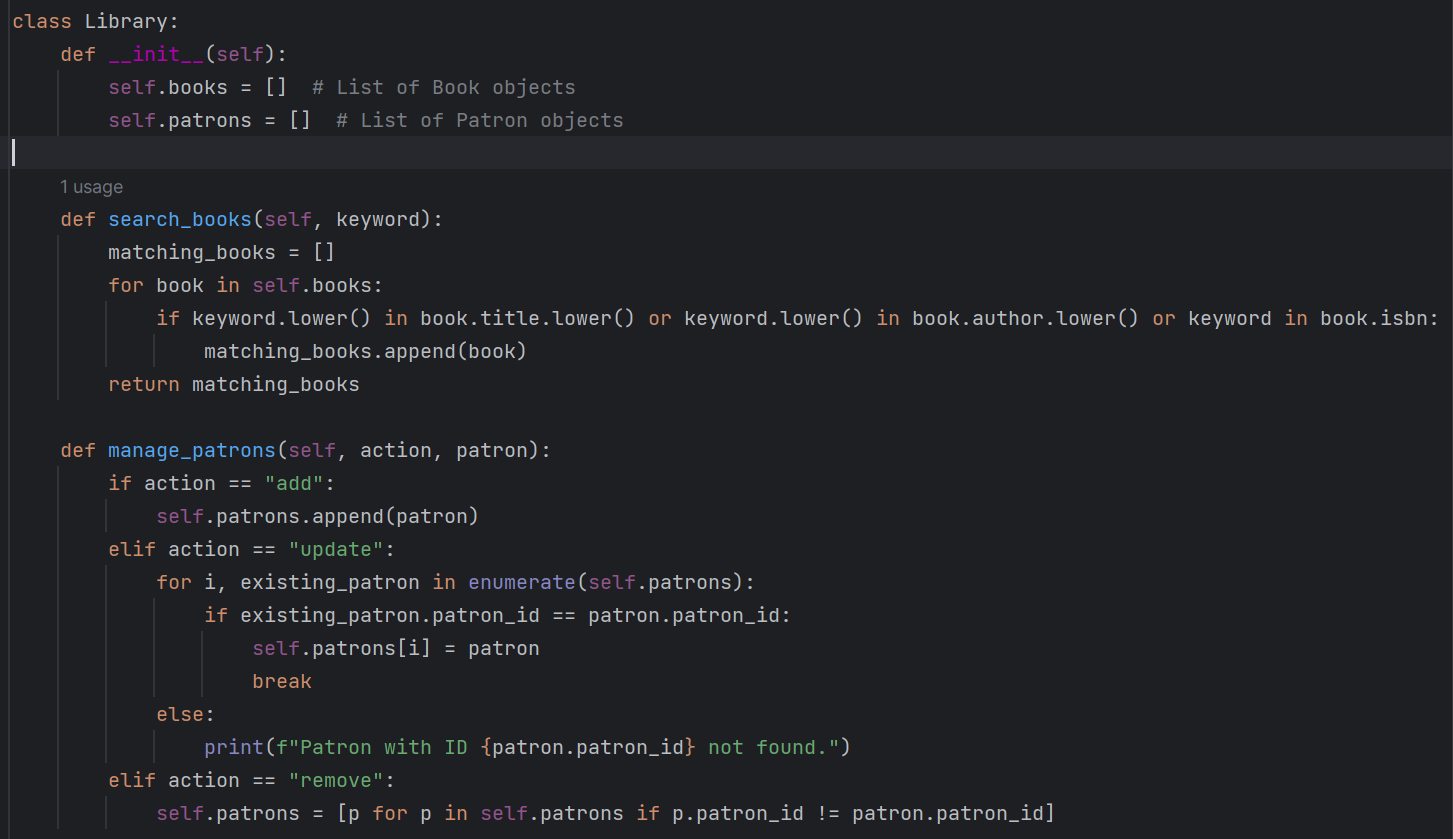
My program starts with imported libraries I used throughout the program to meet some of the requirements. Tkinter is helpful for creating the Gui, while json and os are mostly for data storage. Datetime was imported for the transaction part of the program, where I needed to keep track of the check in date as well as the due date. I imported sys because I wanted to reboot the app without having to press stop main and run main every time. In other words, it was mostly used for debugging.

The first class in my program is a simple book class that initializes a title, author, isbn, and quantity. I also set up a simple display method to print out all the relevant objects in this class. Although the instructions on Canvas wanted a method that could allow users to add, remove, and update books, I decided to keep the initial class simple so that I can implement those more complex methods in the library class.

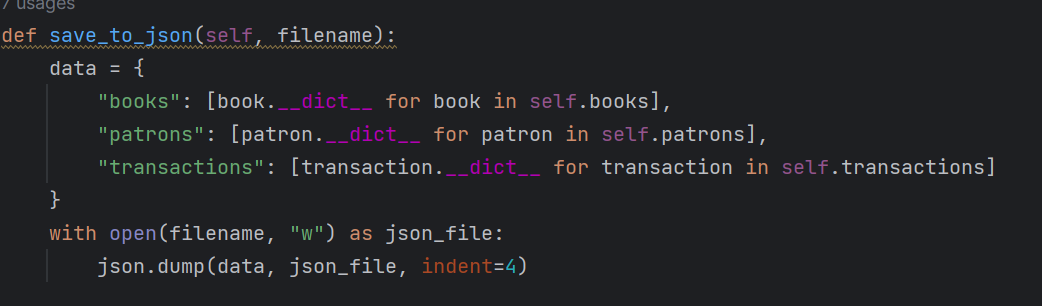
There is nothing interesting going on here in the patron class either. Just a simple innit method to declare some of the parameters, a simple method that allows users to modify a patron‘s information to a small degree, as well as a display method to print out relevant information. Once again, these methods in the beginning are just to set up all the features that are coming up later.

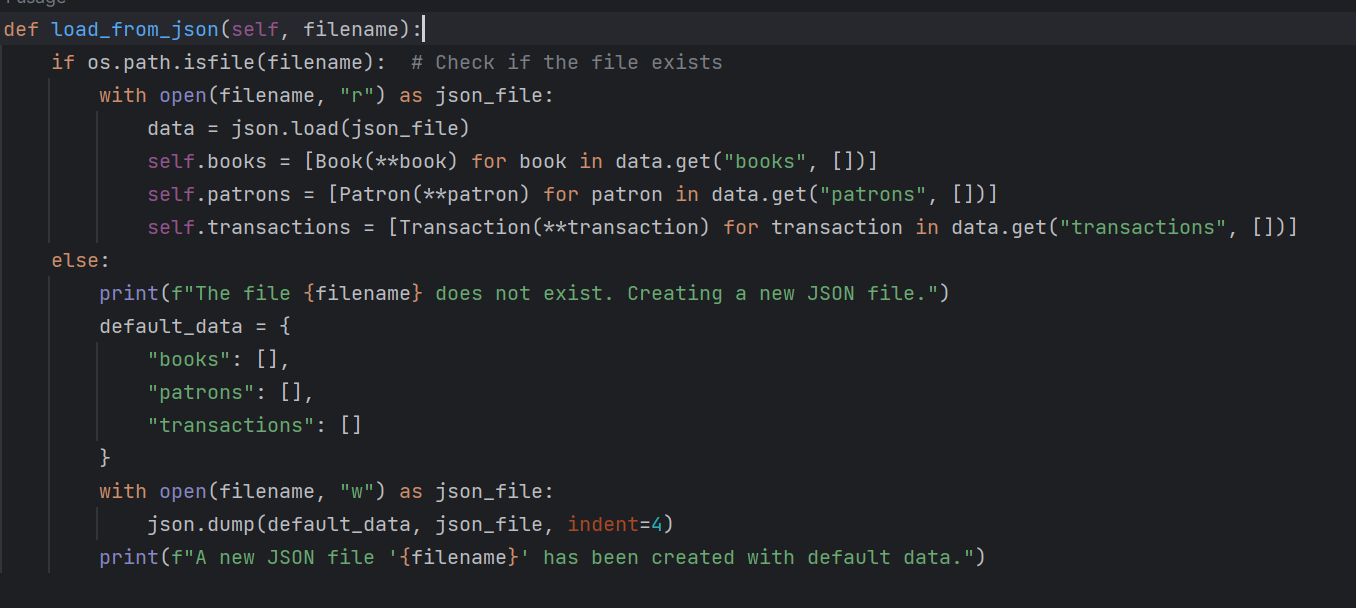


Throughout this program, I was having a lot of trouble with the transaction part, since there are so many things that need to be accounted for. For this method, however, I decided to keep it relatively simple and initialize some things I think I will need later. The name of the book and the name of the patron are information that is given by the user, but the due date and the fine will be calculated separately. That is why I thought it would be better to give them initial values.



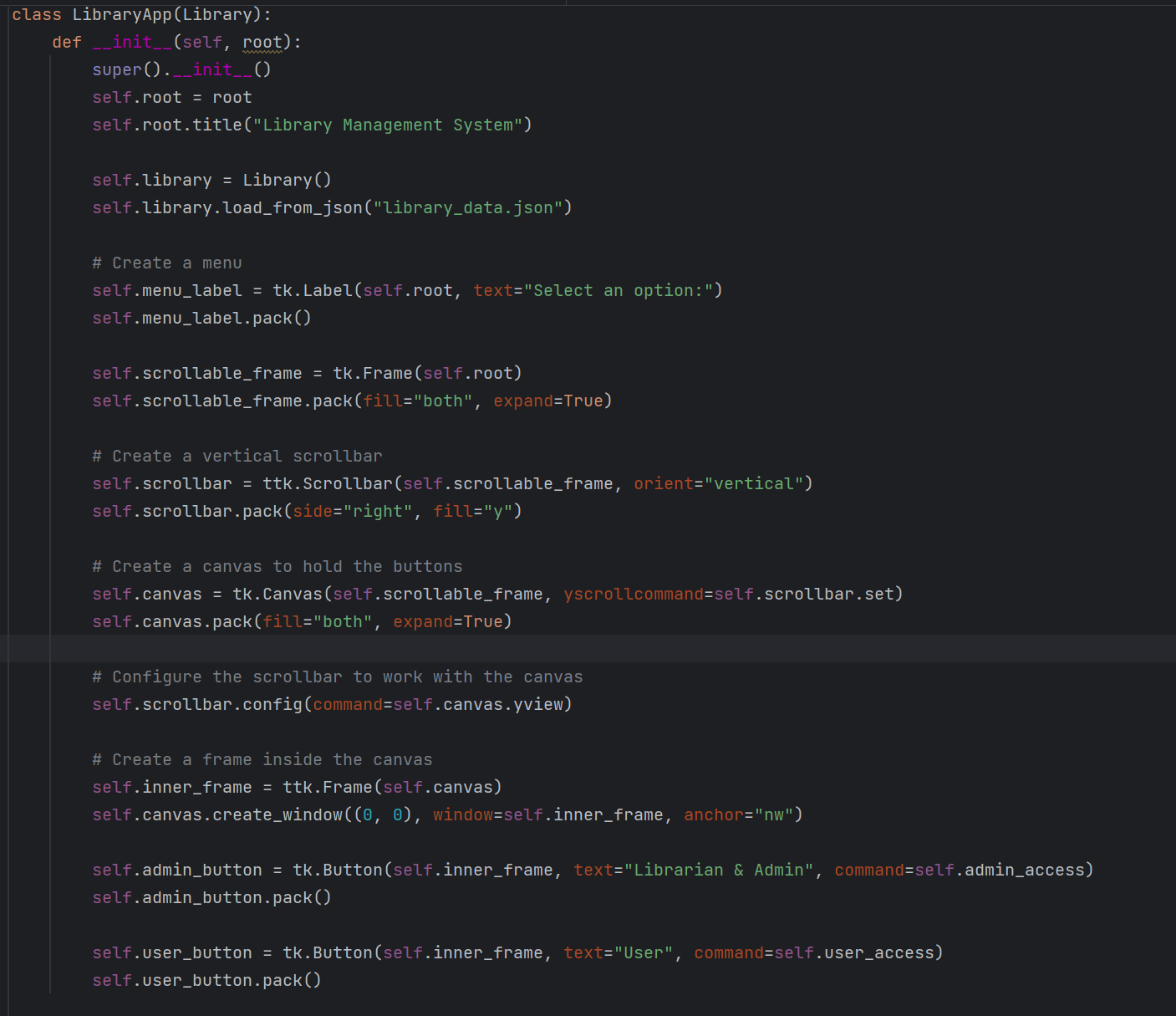
This is the first major section of the library class. After l initialized a list to store books and patrons, I started to work on a method that would go through the books list to see if any attributes associated with the books stored within the library match what the user is looking for. At first, I was having some trouble getting the for loop to work since the information stored in the database about each book was in a dictionary that was nested in a list. But I figured that if the for loop first cycled through the list, and then checked each key of the dictionaries, then I could get the method to work. As it stands now, when some attribute of the user’s input matches some attribute of a book that is in the library, the method will give the user more information regarding the book they are looking for. This includes the title of the book, the name of the author, the book’s isbn, and how many books are currently available.



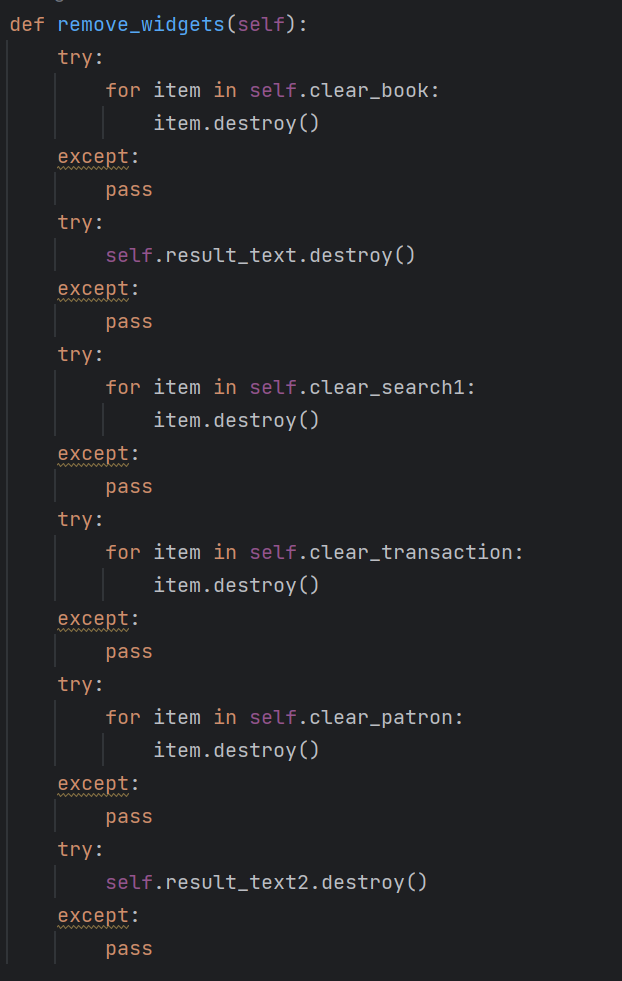
I was not entirely sure how to use json files, so I had to do a lot of research online before I was able to come up with something that worked. Essentially, I am taking all the parameters of books, patrons, and transactions, and creating keys that will be used in dictionaries. This way, whenever new books, patrons, or transactions are added to the database, everything will be nice and organized. The load from json method allows the program to pull information that is already stored in the file, so that certain actions like transactions can be done. If there is not a json file that has already been created, a new one will be initiated so that no errors occur.

Since a user interface was required, I made a new class that handles all the features that involve a GUI as well as altering some implementations so that they can be accessed through a simple user interface.

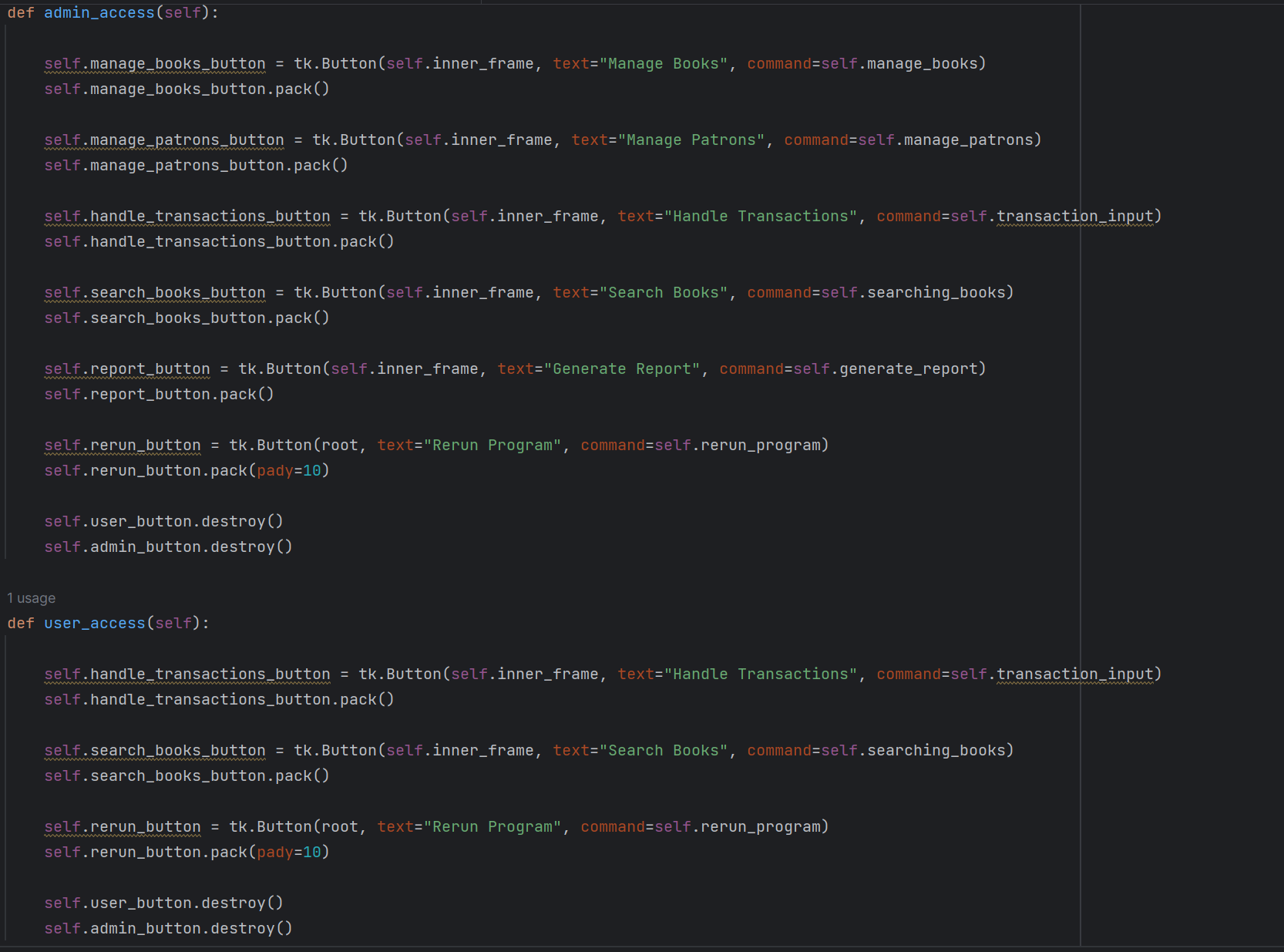
In all honesty, this was the part where I had the most trouble with and had to look up videos online for help. But I wanted to go above and beyond anything I have created before and spend many hours debugging and taking advice from more experienced programmers.



I created this class with the focus being user interaction. To grab some of the of the methods from the library class like the search method and the json methods, I used inheritance and super innit to make the program more efficient. This is also something that was mentioned in the assignment’s instructions. From what I’ve learned while working on this project, self.root represents the window that will pop up when the program starts running. The library is where I store information about books and patrons, although it is probably a bit redundant since the library is the parent class already. However, I did not want to change anything once I had the code working, so it is staying for the time being. I also searched up how to create a scroll bar, so that users can more easily navigate through the program. At the bottom, I created two buttons for users to choose from. This was as close to role-based access control as I could get without having to implement a whole bunch of code that I didn’t understand. So rather than just ripping code that people have already made, I decided to just replicate it in a way that I could do on my own.

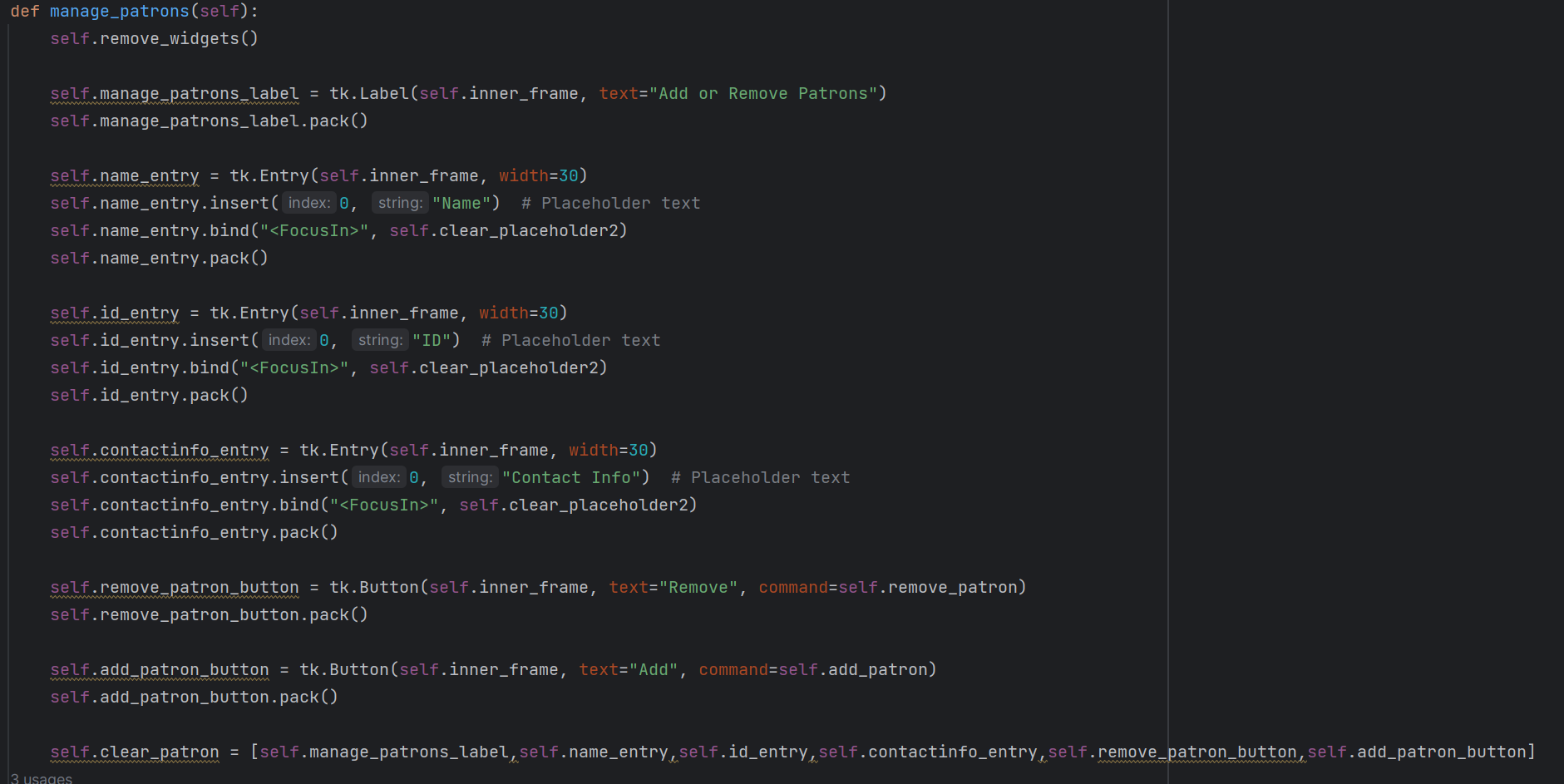


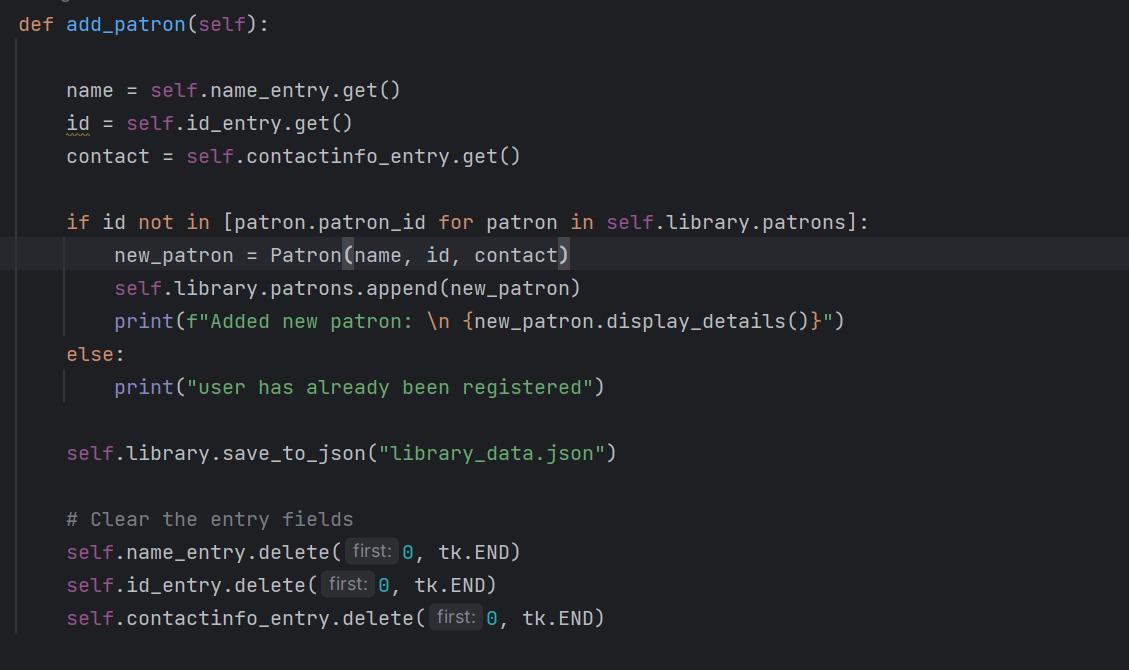
Although it might be a bit out of place, this was a method that I came up with to avoid text boxes from stacking up, even though I had a scroll bar. What it essentially does is delete all the tkinter elements when a button is pressed, so that all that remains are code that comes after the method is called. There might have been a better way to do this, but when I got this to work. I was satisfied.

With these two methods, I tried to create a very basic separation of roles. If the user clicks the regular user button, then they will only have access to a few basic methods. The admin/librarian button, however, will allow the user to use every feature the program has to offer.

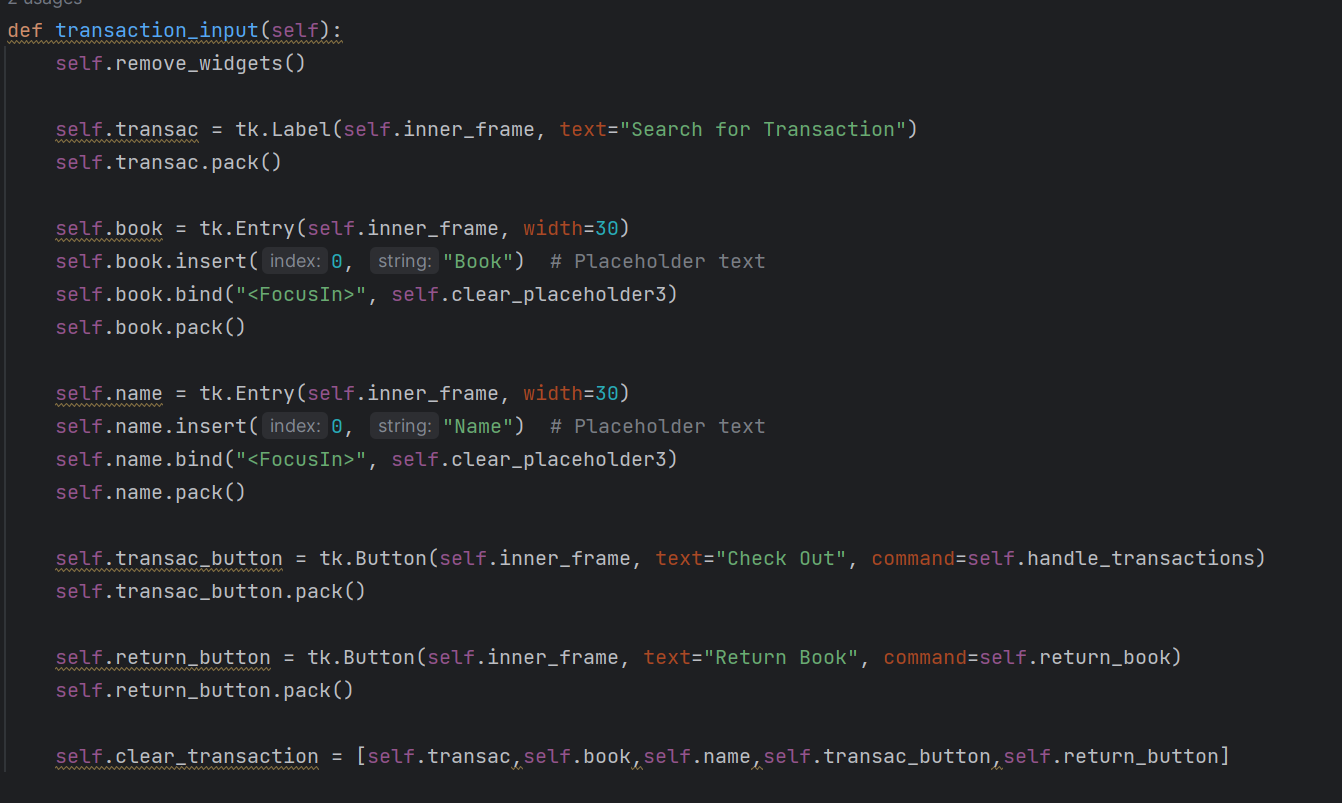


Since this was something that the assignment asked for as well, I created a method that essentially prints everything stored in the json database. It includes information about all the books in the library, all the patrons that are currently added to the database, as well as any transactions that may have been accounted for. This wasn’t too difficult to do, since I am just using basic file operations.

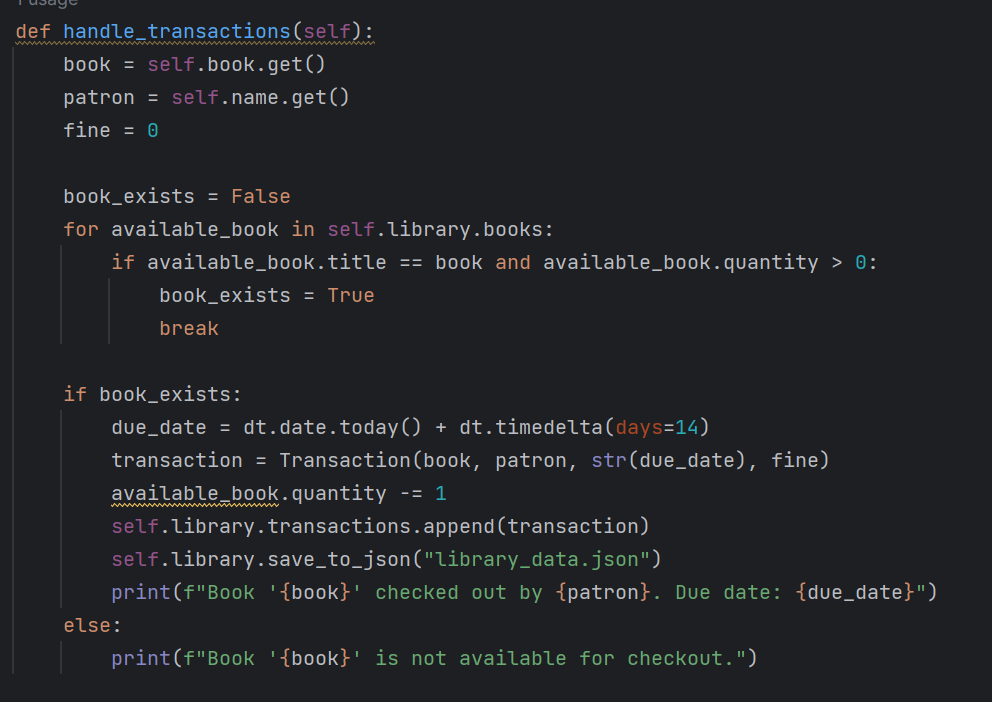
This is the method that will allow users to input information about a patron. Three text boxes will be created, and placeholder text will tell users what to input in each text box. One the user has put in all the information they want to, they have the option of adding, or removing patrons with the information they put in. The clear place holder method makes it so that when a user starts typing in the text box, the words that help the user understand which box corresponds to what information will disappear. I did this so that the words, ”Name”, ”ID”, and ”Contact Info” will not be saved in the library as part of the input.

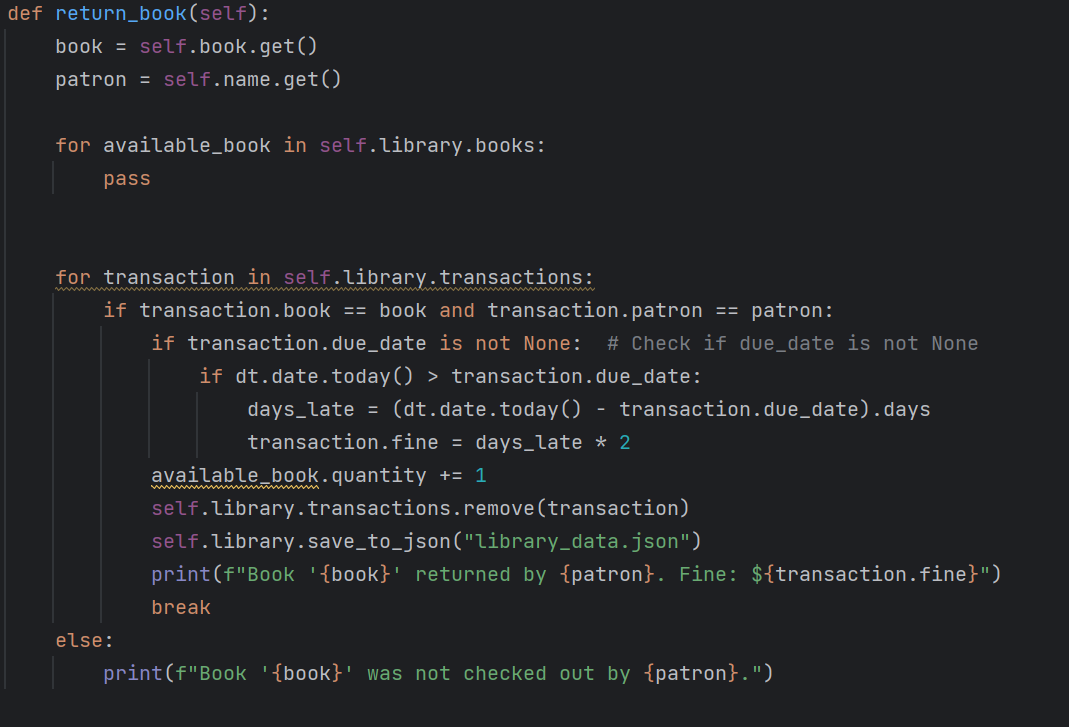
This add\_patron method will use the information that the user puts into the textboxes and add a new set of data into the patron database. But first, I made sure to add an if statement that checks if the patron id, which should be a unique number, is already stored in the system. If it isn’t, then the patron’s information will be appended. The json file will also be updated accordingly. At the bottom, I clear out the inputs so that if the user chooses to, they can add another patron without having to manually delete each input.

Similarly, I added a remove\_patron method that essentially does what the name implies it does. After taking the input from the user, the method will check if the person exists in the database. If they do, then they will be removed from the database. The json file will be updated to reflect this removal.



These two methods set up the method that will be used by users to check out and return books. It’s the same implementation as the previous buttons used to manage patrons, so there’s no need to really go deeper into explaining what these lines of code do.





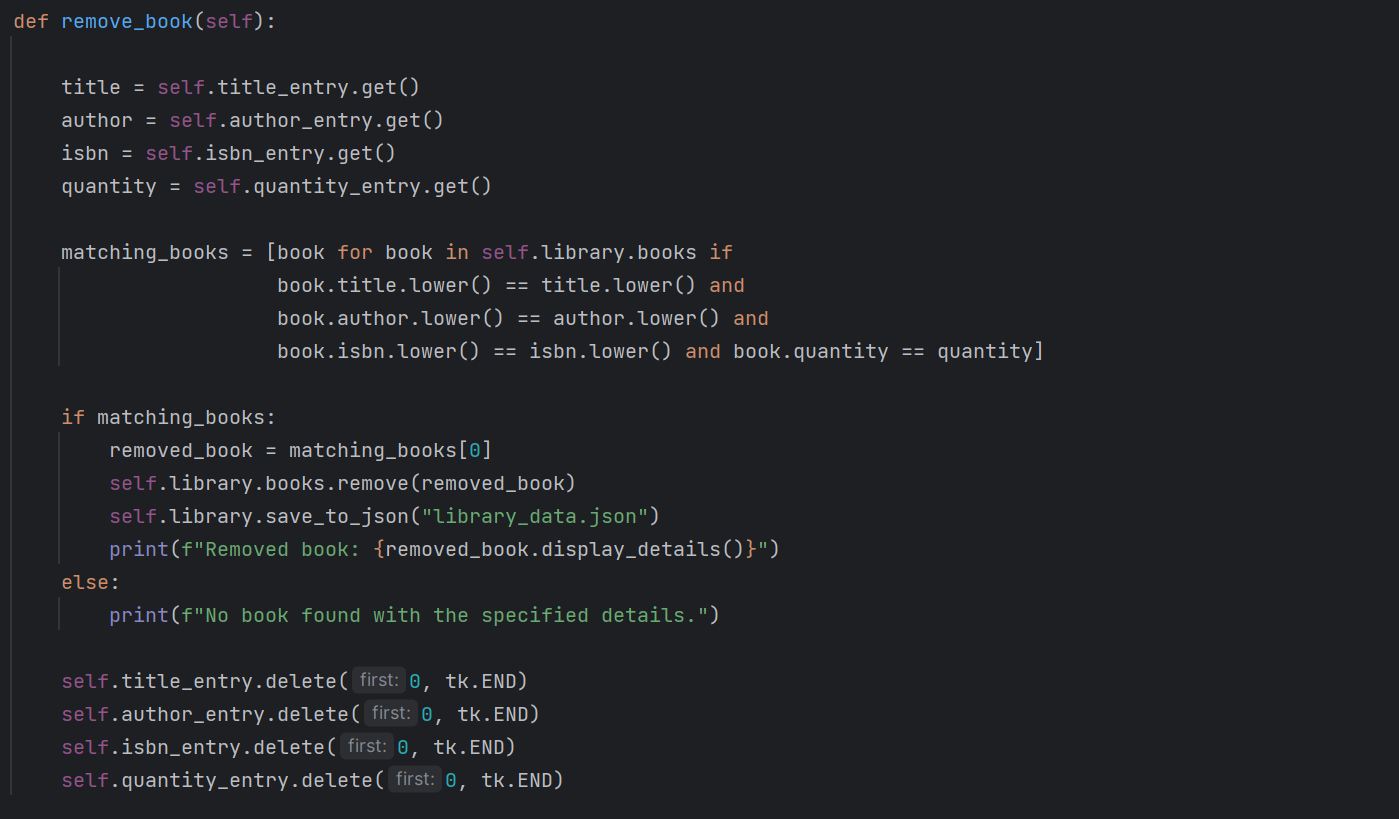
The transaction method was perhaps what I had the most trouble with. But I had a basic idea of what it should look like, and how I should go about implementing it. I take the input of the book's name and check if it is in the database. If it is, then the transaction will proceed. The method will create a due date, which is set to fourteen days from the current date. The transaction is then recorded, and a message is printed to let the user know they have checked the book out along with the due date. If the book does not exist in the database, a message will let the user know the book cannot be checked out. The second method will allow the user to return a book. It checks the library to see if the book has been checked out, and if it has, the book will be returned. However, there is a second criterion that checks if the due date has already been passed. If it has, a transaction fee will be calculated based on the number of days late, times two. The user will be notified of this fine. I also spent a lot of time to implement some basic code that will deduct one from the quantity of the book that was borrowed. If a book is recorded in the library’s database, but the current number of books in the library is zero, the user will be notified that there are no books they can check out.

The next method looks for a textbook when the user clicks on the search books button. This method uses the book search method from the library class, so there isn’t any extra implementation other than the GUI elements. After taking the user input as a value, the second method will check if the book exists in the database. If it does, then the method will display every detail about the book, such as the name of the author, the ISBN, and the quantity of the book in the library. If the book does not exist, the user will be notified.

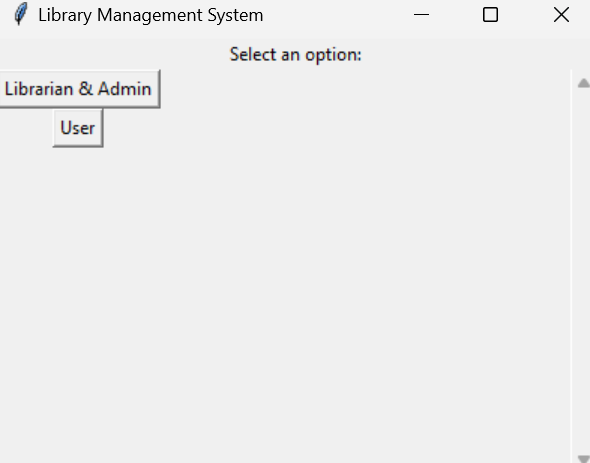
This method uses the same implementation as the other buttons to take user input. I feel like a lot of the program does have redundancies, but it feels necessary to make the program work. If there is a more efficient way of creating these buttons, I would like to learn about it so that I can shave some lines off the program.



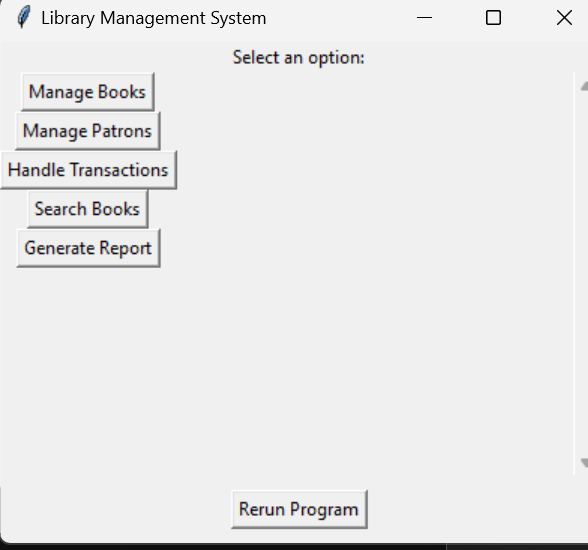
This add\_book method is very similar to the add-patron method. The biggest difference is that there is one more parameter, making the number of user input four instead of three. The method will check if the ISBN of the book exists in the library, and if it does not, then the information given by the user will be uploaded to the database.

The remove\_book method will remove the book that the user wishes to remove. However, if the book does not exist in the library, then the book will not be removed. Implementation for the advanced features is not available because as much as I tried, I could not figure out a way of separating access control for users with different privileges.

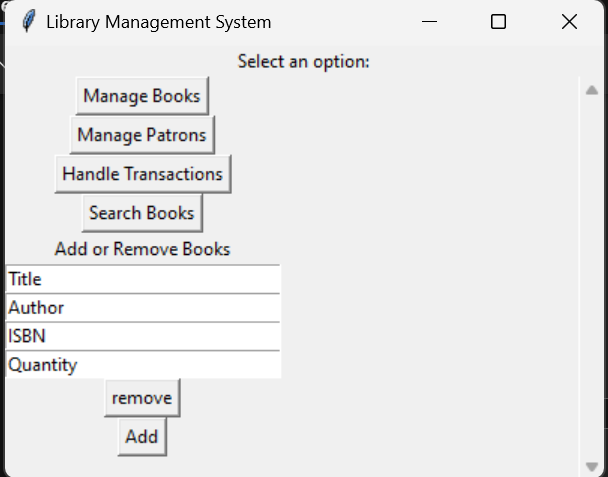
Showcase of program:



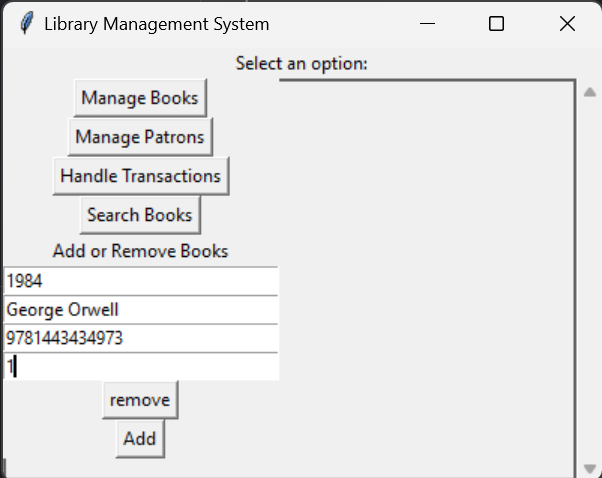
This is the window that pops up when the user launches the program



Once the user has selected the admin role, This is the GUI that the user can interact with to use the library.

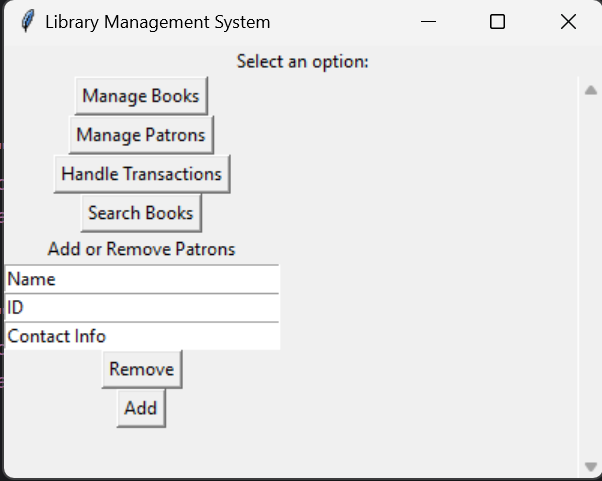


After pressing the manage books button, text boxes will appear which allows the user to add new books or remove existing books from the library.

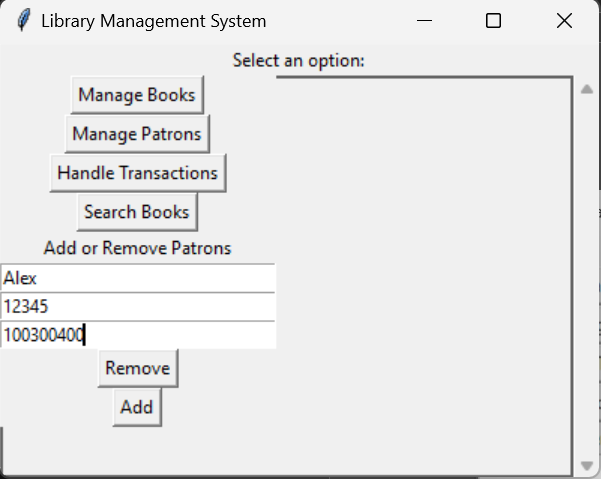


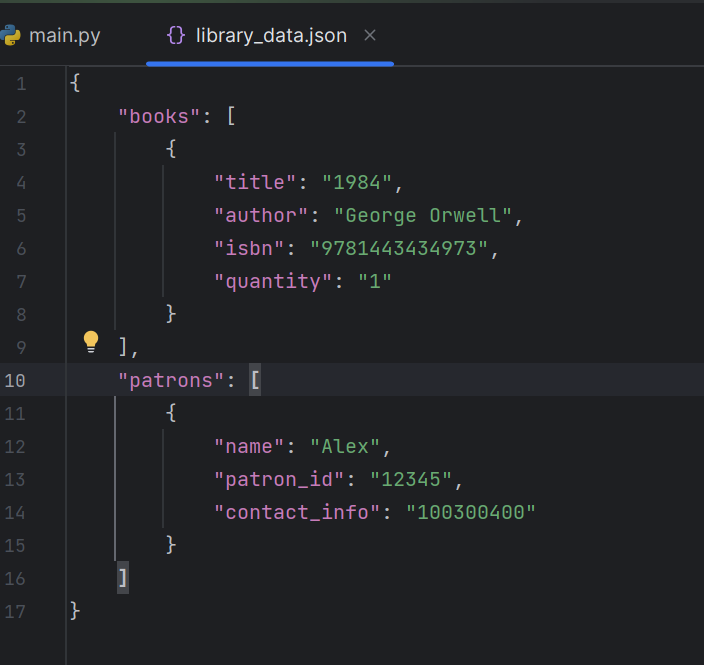


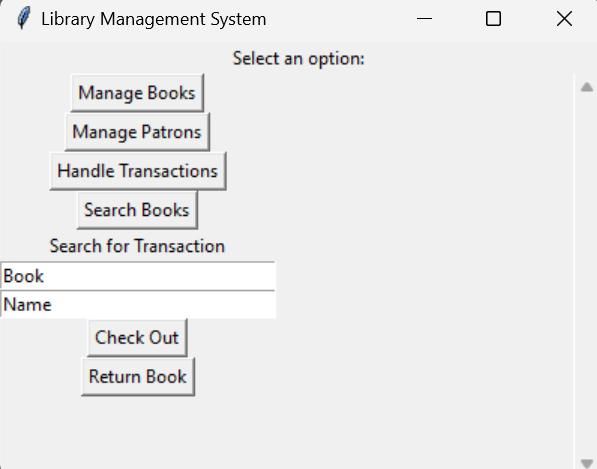
Using the add button will add the information given by the user into the json file. Inputting the same information and pressing the remove button will delete the data from the json file.



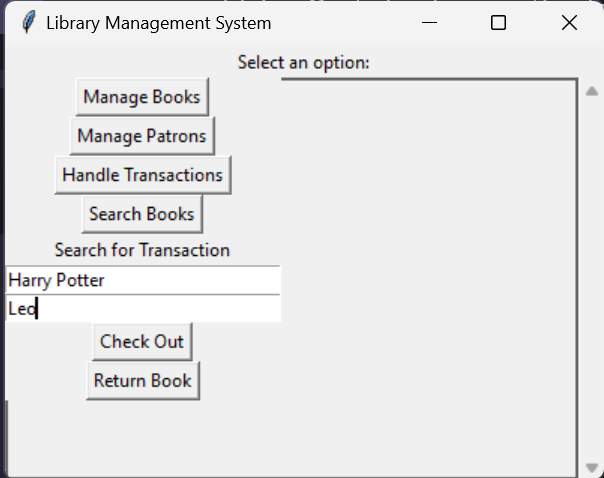
Pressing the manage patrons button will create three text boxes that allow the user to put in information about the patron. The user can either add or delete this patron’s info from the library which will be reflected in the json file.

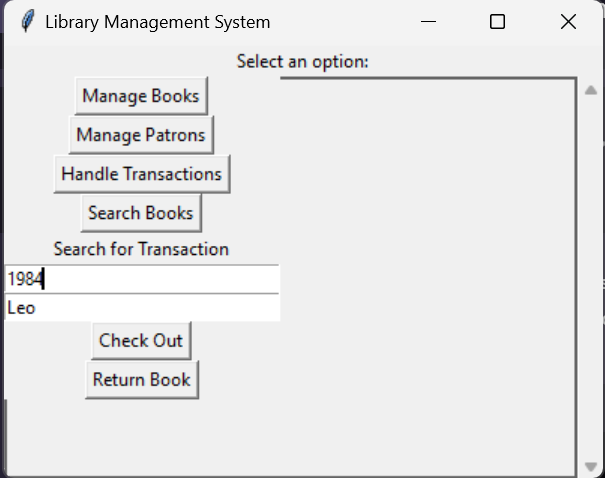
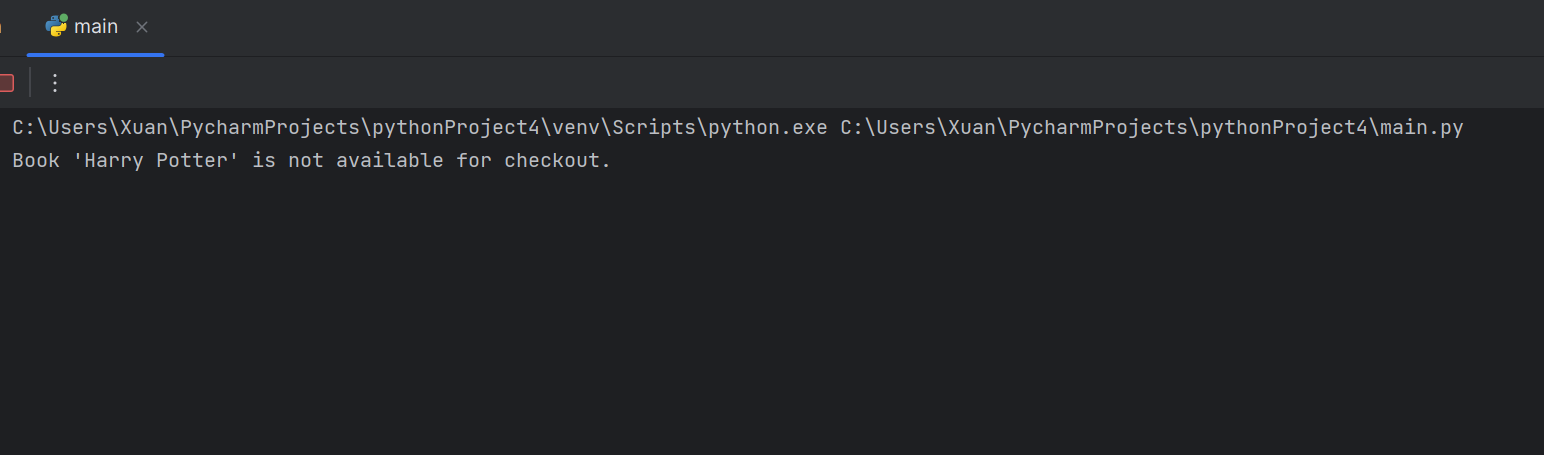


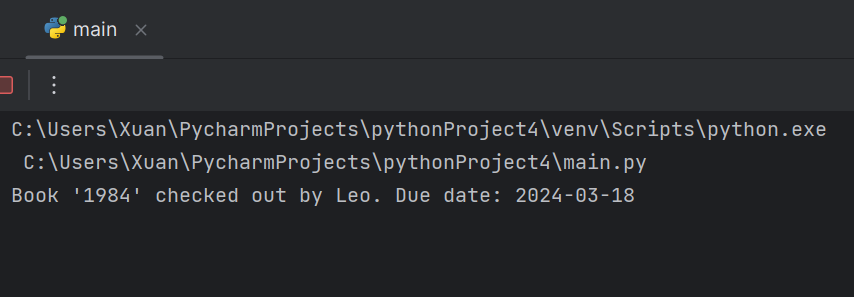




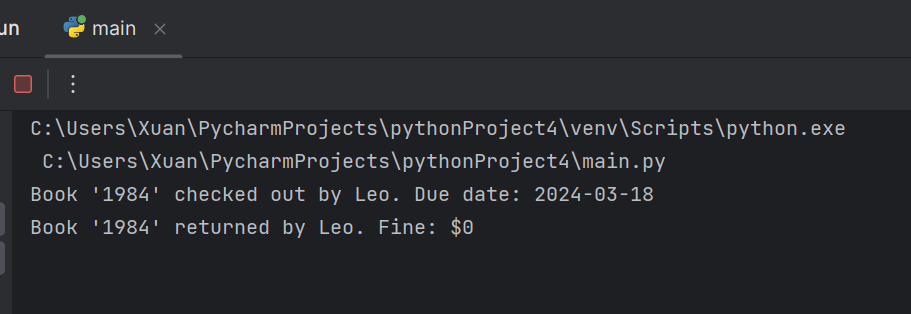
The transaction button will likewise create text boxes for the user to put in information about the name of the book and the name of the patron. If the book's name is in the database and has not been checked out, pressing the check button will allow the user to check the book out under the name of the patron input. If the book is unavailable for checking out, the user will be notified.



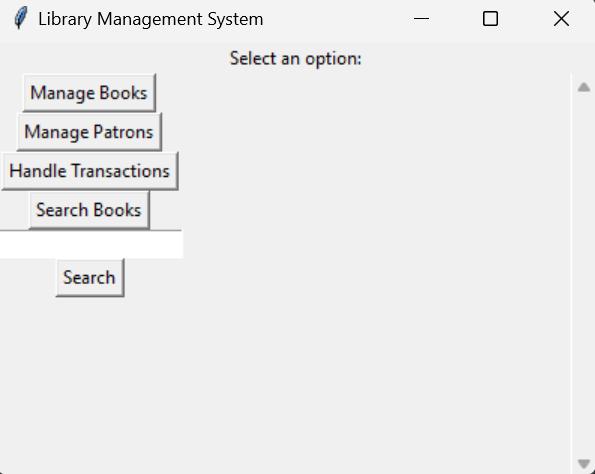




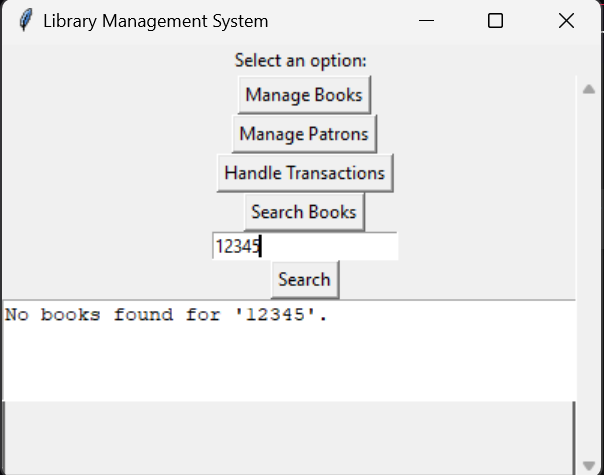
A message like this will appear if the checkout was successful.

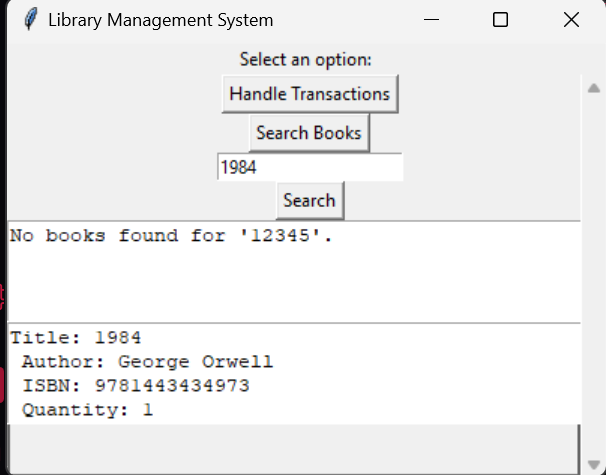


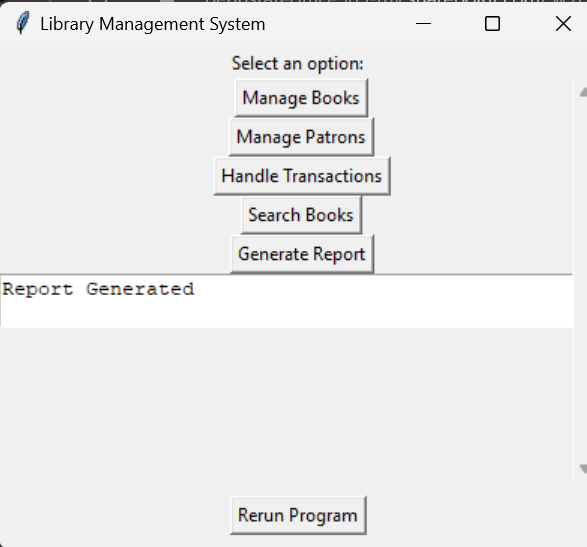
A message like this will appear if the return was successful.



The search books button will create a textbox for input.







Finally, we can generate a report with this button.



Whatever is saved to the json file will be printed out

My experience with this program: This was one of the hardest I worked on and could not have done it on my own. We have learned a lot of the elements that make up this program in class, but there were also certain aspects that we have not covered (i.e. GUI and database). I had to take up many videos on python while browsing YouTube, while also checking resources such as Stack Overflow. These resources proved very useful in my journey, as I was able to complete the project. In the future, I would like to see tighter and more organized code. I would also like to create a similar program in a timelier manner, because this program truly took a great deal of time to put together. Overall, I thought the experience was fun, and I found it extremely satisfying when things worked out the way I wanted them to. On the other hand, it was very frustrating when things did not work out.