Arianna Morgan

November 13, 2022

Foundations of Programming: Python

Assignment 05

Lists and Dictionaries

Introduction

In this module, we continued to learn about lists, including loading data from a file into a list, and we learned about dictionaries. We also learned about some general programming concepts including Separation of Concerns (breaking up code such that each set of information has its own section), functions, script templates, and structured error handling (anticipating potential errors and allowing them to be corrected while keeping the code running). This week we expanded upon last week’s script that allowed the user to create a list of CDs (with index, title, and artist), add to that list, view the list, save the list to a file, and exit the program once done. We changed the data type of the CD list to a list of dictionaries instead of a list of lists and we added functionality to load data from an outside text file and to delete an entry.

Details

The first task to enhance the script was to load data from a text file to be added to our CD list in memory. Because our CD list needs to be a list of dictionaries, we need each row (which has the ID, CD Title, and Artist separated by commas with a new line in between rows) in the text file converted into a dictionary. This is very similar to the address book example on page eleven of our Module 05 PDF. The code we need here matches lines 19 through 24 in the first screenshot on the page with modifications just for the names of the keys. We are opening the file in read mode, taking each row and turning it into a list split by commas and using strip to remove the “\n” at the end, then we are creating a dictionary out of each list row by assigning each item in the list to a key (“ID”, “CD Title”, and “Artist”). We then append each dictionary to the table in our program memory. I clear the CD table memory in the program before loading the file data so that we can start with only our existing saved data. This allows us to load our existing data, add and/or remove data to/from it, save it back to the file, and then reload it without creating duplicates in the program.

We next had to update our add function to turn the user’s inputs into a dictionary before appending it to our CD table. Again, we have seen something very similar in the Module 05 PDF, in this case on page ten. Rows 24 to 29 are exactly what we need, just with the naming and keys changed. I initially tried assigning the inputs directly to the dictionary, as in lines 18-21 of that example code, but even though I had used the int function for my ID assignment, I was getting an error that I couldn’t use a string. I couldn’t figure out why it wasn’t recognizing the use of the int function there, so I swapped to the variable assignment first method, and that was successful.

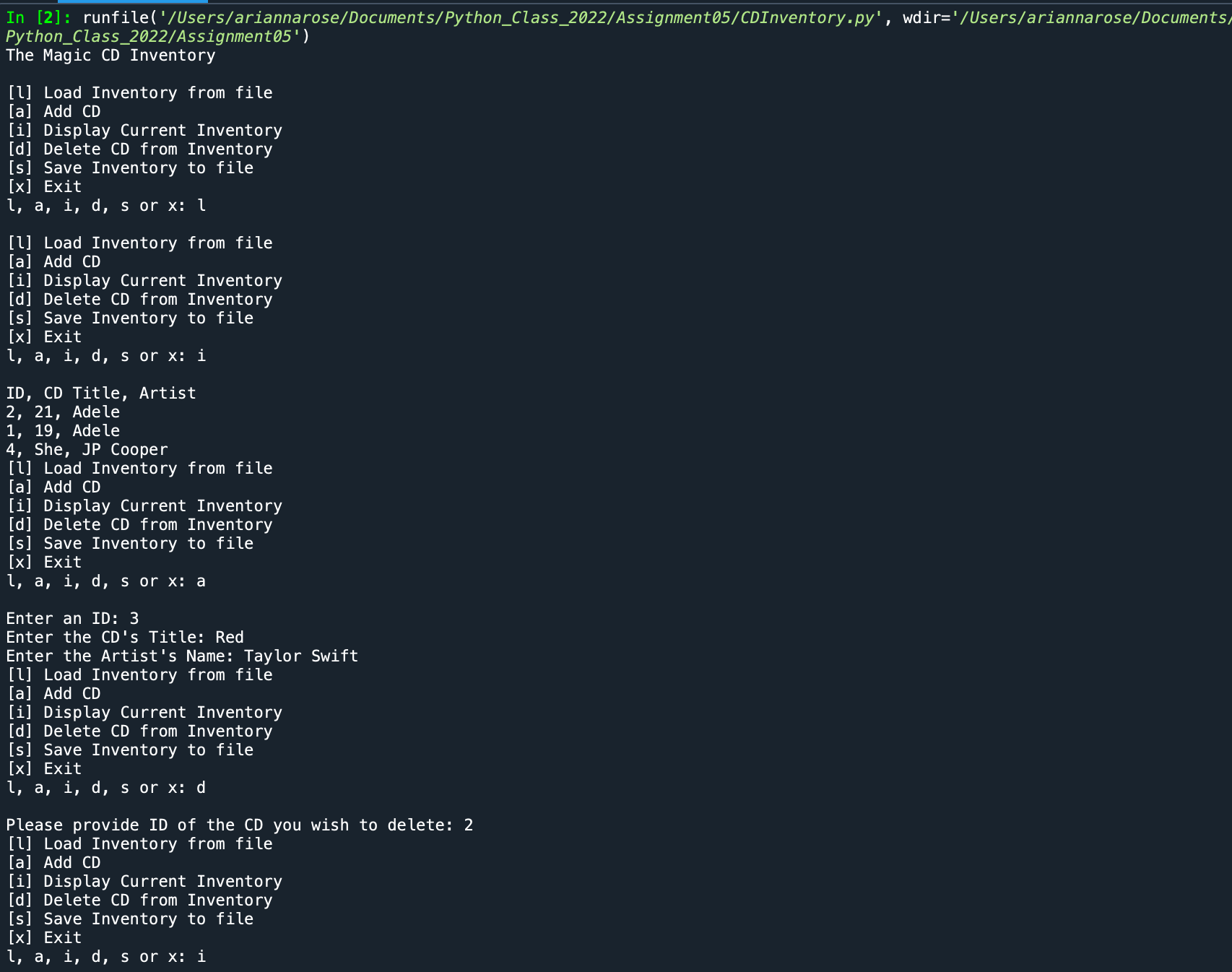
Our display inventory section also needed some updates to account for the new data type of our table rows. I used a similar method to what we have been using to prepare our data to be saved to a text file: I used two for loops to work through the rows and then the items in the row (here using row.values() in the second loop so we are just working with the values and not the keys) so that we can create a string for each row where each item is separated by a comma. I added a space as well so that it would match the header’s format and then when getting the row string ready to print I had it take all but the last two characters so that we skip the extra space and the comma at the end.

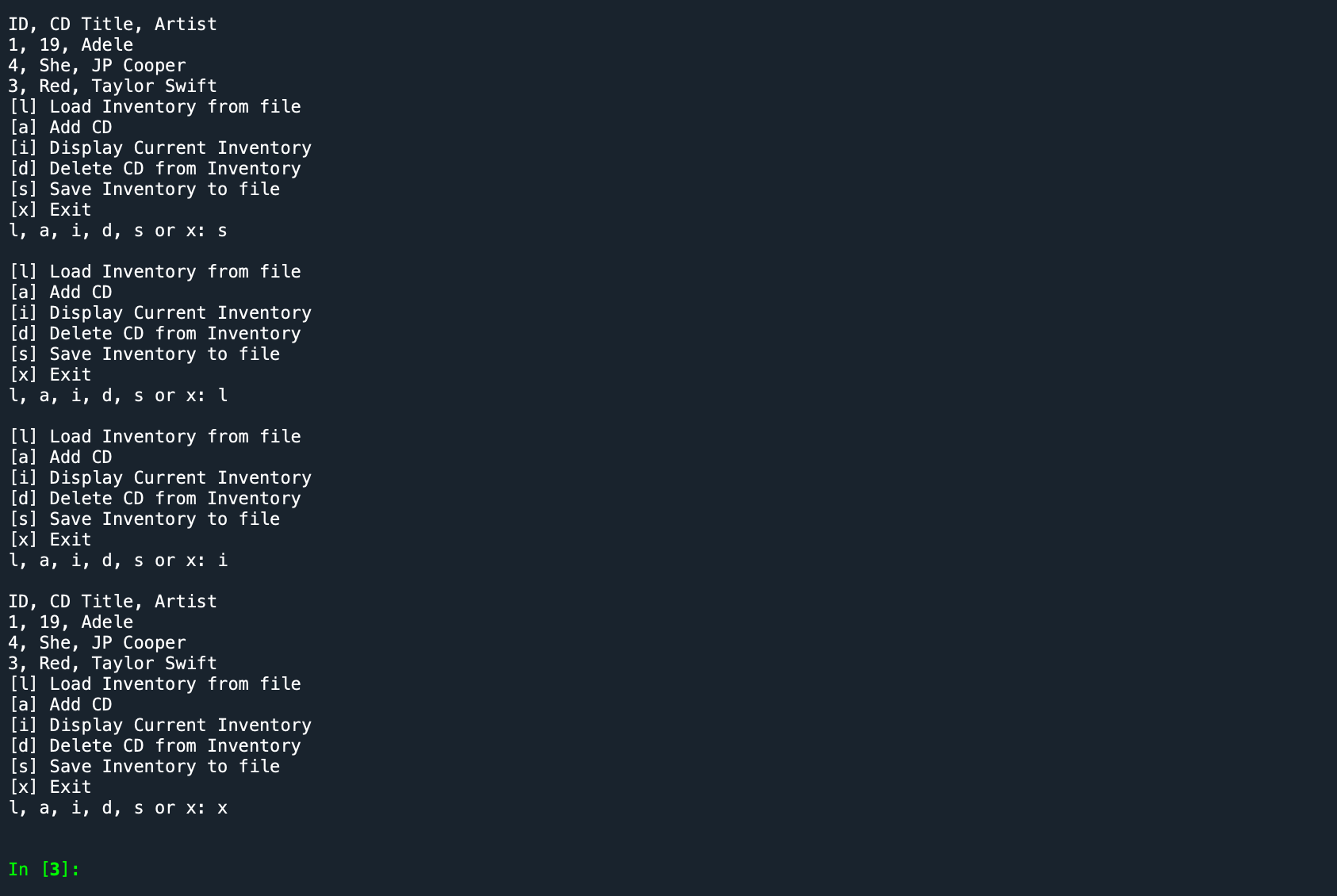
Next was a new functionality, the option for the user to delete a row. I probably spent the most time on this one because I was having trouble figuring out how to refer to the row I wanted deleted when using the del function. I had the user input the ID of the row they want deleted and then I ran a for loop to run through each row and check if the ID is the user input. Since I am using the del function on an item in a list, I realized I needed to use an index to identify where the row to delete sits in the list. I decided a counter in the for loop was a good way to do this. I set the counter as -1 before the loop started and had the counter add one to it before doing the check so that the counter was set at zero for the first item, in line with how list indices function. This allowed me to delete the correct row by having my del function reference the CD table with the index as my counter variable.

The last update needed was for the save function. There were only two edits I needed to make in this section. One was to access the file as write instead of append. Since the user will have loaded what was in that file and added or deleted items from it, we want to be able to save our changes back to the file without creating duplicates. Lastly, as in my display inventory section, I needed to use row.values() in the second for loop so that we are only working on the values in our dictionaries and not the keys when we are creating our row strings.

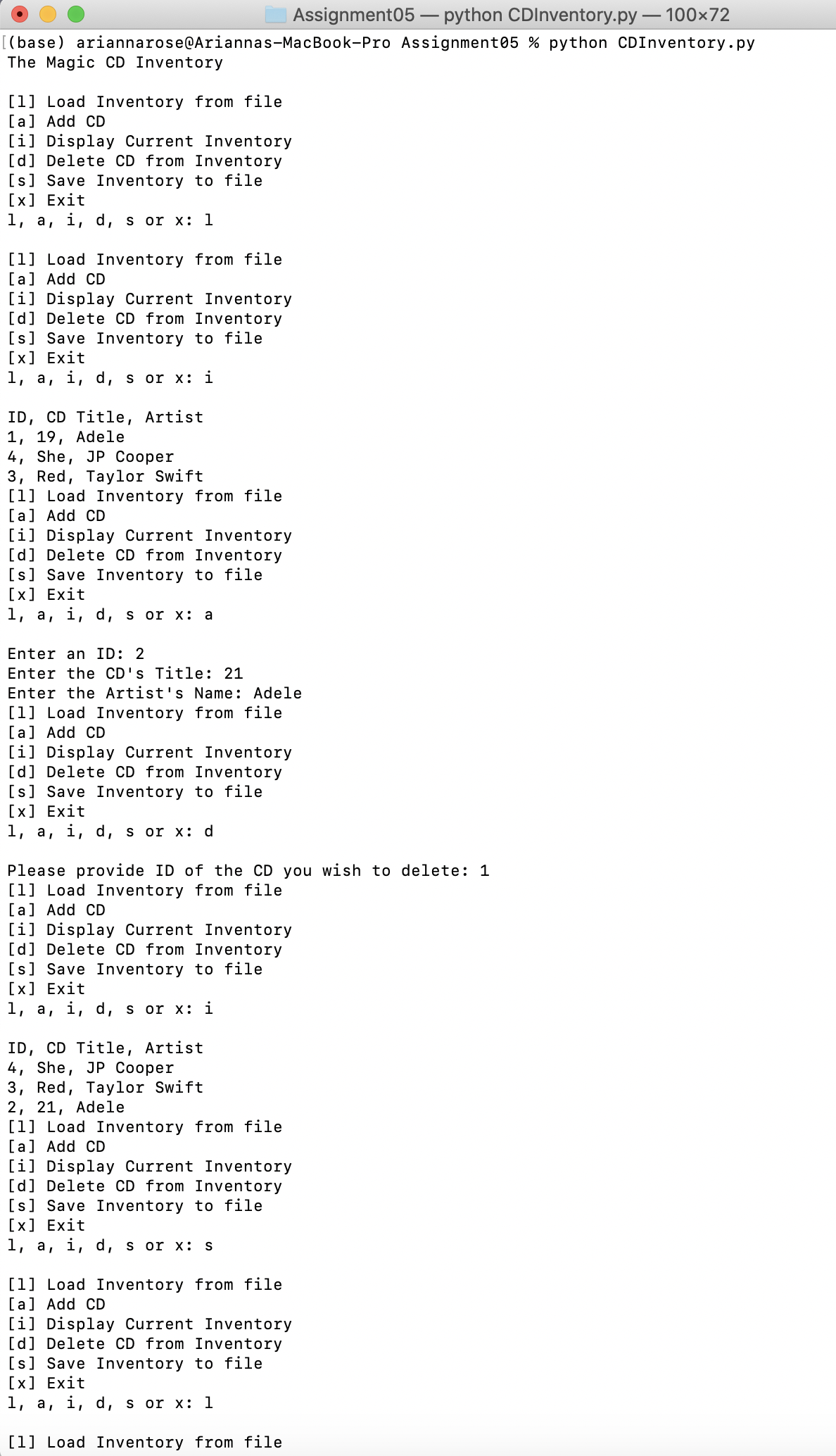
Below are screenshots of my script working.

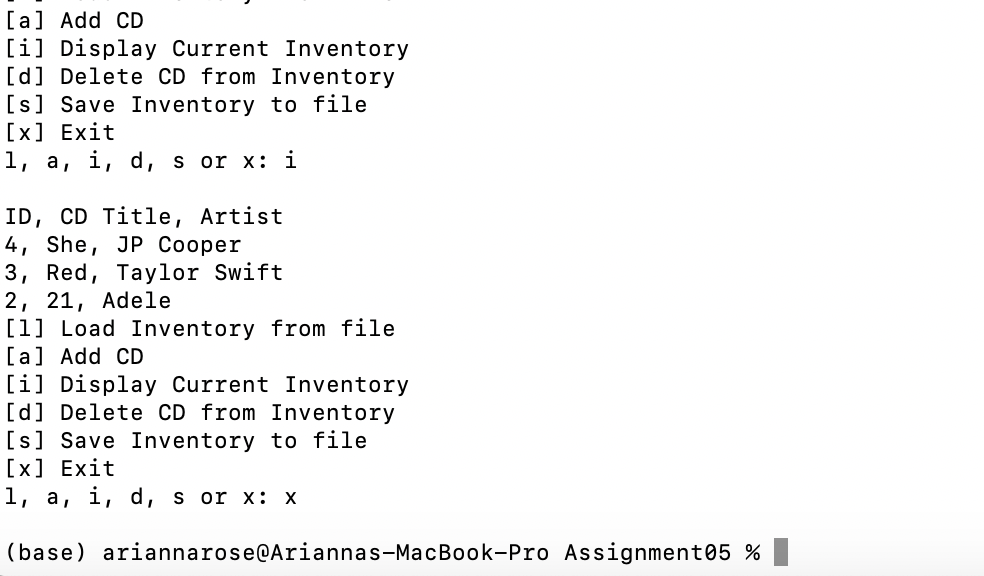
In Spyder:





In Terminal:





Summary

This assignment was great practice working with existing code that we didn’t write and adding value to it. I am enjoying troubleshooting ways to solve a problem and the feeling of accomplishment when I finally get to a solution that works. As with prior weeks, I am eager to move on to the next topic and continue to build on what I am capable of.