Aime Ayers
November 12, 2022
IT FDN 110 B
Assignment 05

Modifying the CD Inventory Program

Introduction

In this document I will describe the concepts used to modify and test the CDInventory.py script as required for assignment 05.

Approach

To begin this assignment, I had to approach it differently than the first few. Since we were directed to modify existing code that was created by a different developer, I needed to spend time reviewing the code and functionality to make sure I understood what was done. I spent time running the existing logic to view the results of the actions taken to verify I understood the program. I also reviewed the original outputs so I would be able to validate I did not change the current outputs with my modifications. In a real world scenario, I felt it would be important not to alter the outputs of the program in case there were potential unintended downstream consequences.

Next, I determined where I would need to adjust the code to store the user inputs into a dictionary, as well as where I would need to adjust existing functionality that would break after transitioning to a dictionary.

Finally, I added the new functionality to delete, and import saved data using a variety of concepts we have learned through module 5.

Dictionaries

The first new concept utilized in Assignment 05 was the python dictionary. Previously, the CDInventory program was developed using a 2D list of lists. The list relies on the sequential order of the items entered. After adjusting the program to use dictionaries, we are now able to reference entries by their key/value pairs instead of their position within the list. In other words, instead of relying on the list being entered in the specific order of “id” (0), “title” (1), “artist (2),” we can now reference the dictionary key “id” and get the proper value regardless of its sequential position.

Dictionary Methods

In addition to learning about the dictionary concept, we also learned about the different built in methods used to work with dictionaries. These were critical to enabling the functionality of the CDInventory.py program.

Clear()

I was able to use the dictionary.clear() function in assignment 05 to delete results from my inventory. I chose clear() to delete entries because I wanted to make sure the entire dictionary was removed, not just specific key/value pairs. The alternative was to delete each key in the dictionary individually. However, this seemed to add unnecessary lines of code as compared to the clear() function when both provide the same result.

Values()

The values() method was required when I began to iterate through the items within my dictionary when saving values to my file. While iterating through my dictionary, I wanted to make sure I only iterated through the values, not the keys, so only the values were written to the file. This was important so the outputs of the program were not changed from the original intended purpose.

Filter()

After using the clear() function, I was left with an empty dictionary existing within the list. To avoid storing this empty dictionary, I wanted to find the most efficient way to remove it from my 2D list of dictionaries. After reviewing a variety of methods found via google searches, I settled on the filter() function due to its readability. Filter() allowed me to search for, and remove, any None values from the list, which resulted in my empty dictionary being removed.

For Loop

In previous modules and assignments we used the for loop() concept to loop through a list of objects. I used this concept in assignment 5 in order to avoid saving duplicates to the CDInventory.txt file. I was able to loop through each dictionary item that exists in the saved file and compare each item to the list that was currently saved in memory. This allowed me to determine, one item at a time, whether something should be saved to the file or skipped.

Testing the Code

To test the program I executed it in both spyder, and the python terminal to validate that the program was running and performing all actions as anticipated. I ran each option individually, comparing the saved list before and after each update, delete, load, and save option to validate the results were what I expected. Testing evidence is included in the screenshots below.

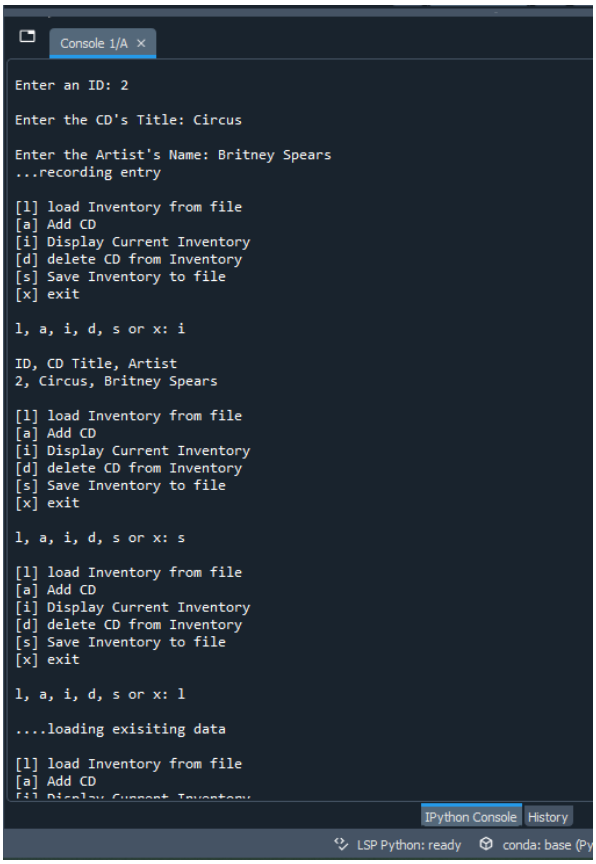

Figure 1 Executing the Add, Display, and Save options in the Spyder IDE.

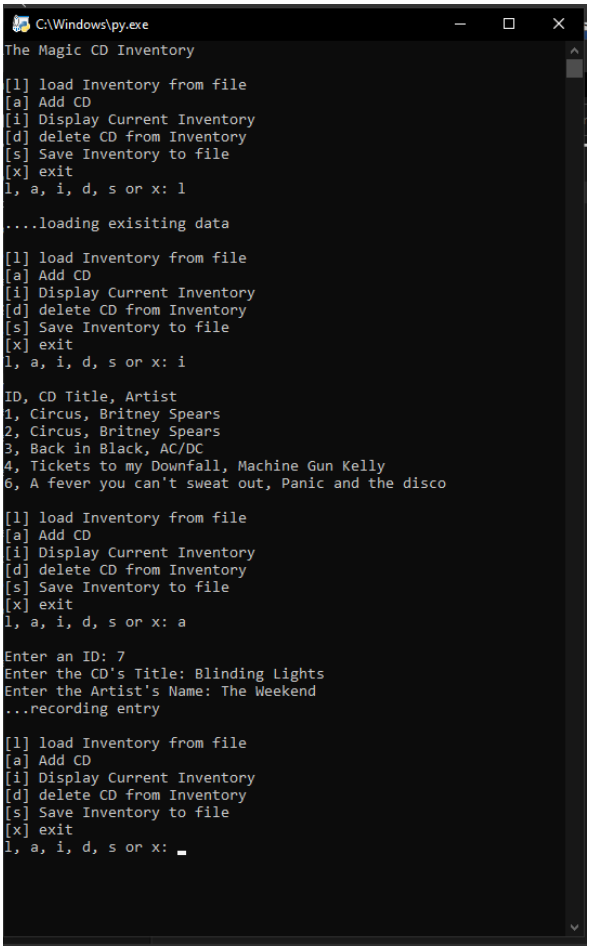

Figure 2 Executing the Load, Display, and add options from the terminal

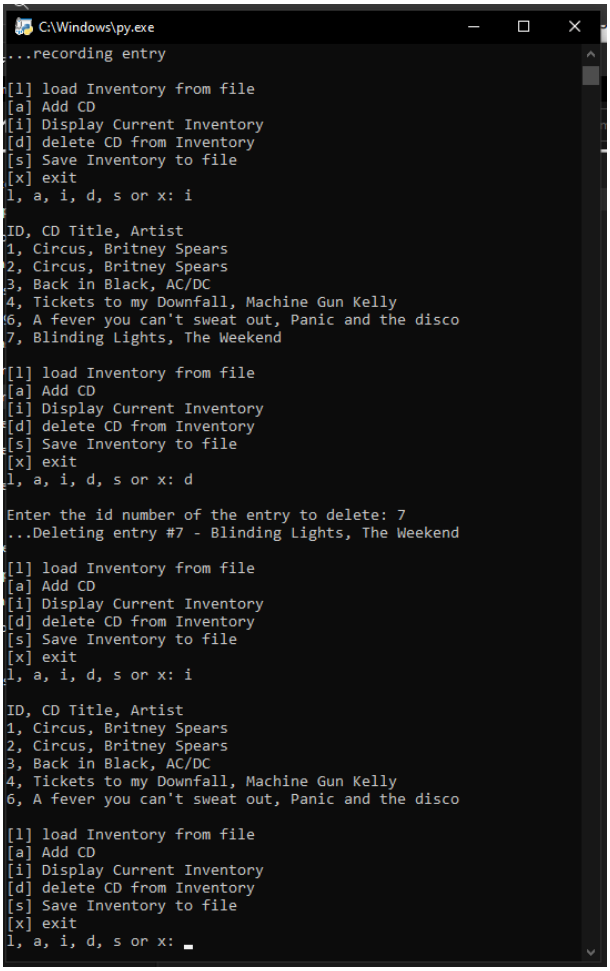

Figure 3 Executing the display and delete functionality from the terminal

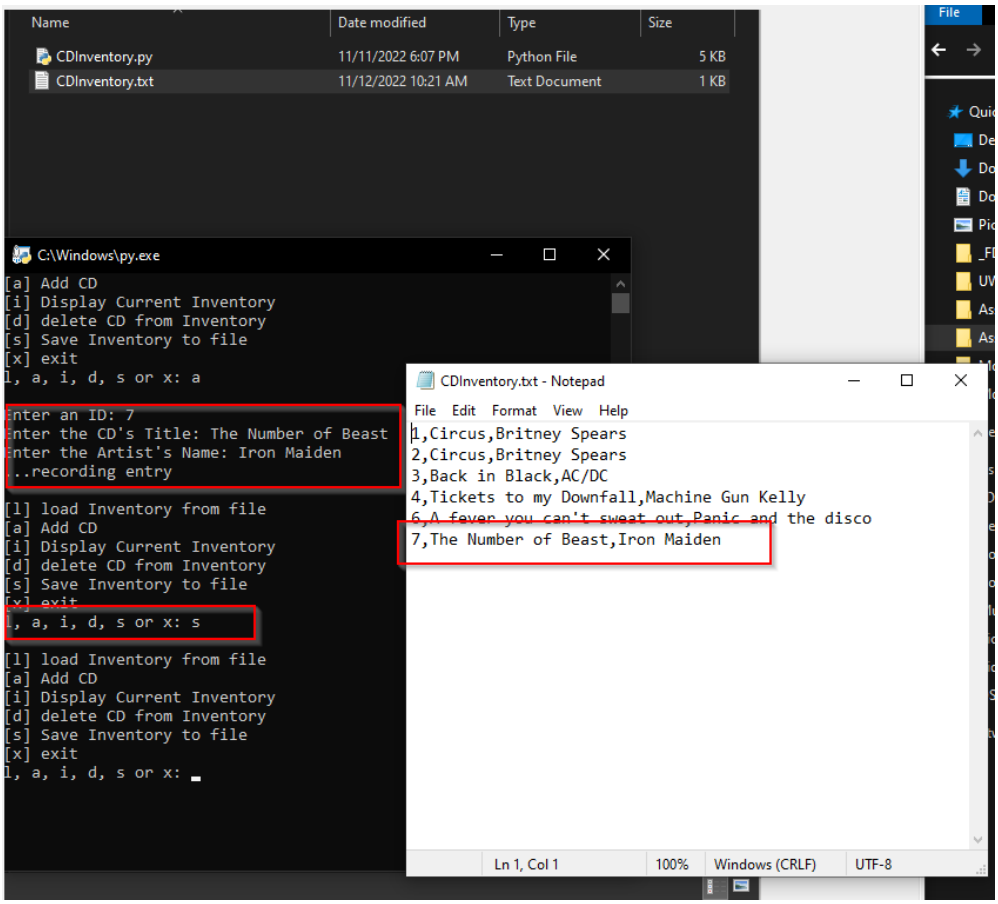

Figure 4 Validating the save functionality works as exepected.