Patrick Danielson

02/14/2021

Foundations of Programming (Python)

Module 05 – Assignment 05

Modifying CD Inventory

# Introduction

This document summarizes the work performed for Module 05 of the Foundations of Programming course. The purpose of this module was to familiarize myself with dictionary object and reading other coding styles. The problem addressed in this assignment was to take an existing starter code for the CDInventory program from Assignment 04 and modify it to operate on dictionary objects rather than list objects. Additional function was also required to be able to read in existing data and delete entries from the inventory.

# Modifying Starter Code

The first step in the assignment was to restructure the existing starter code to accept user input for entries and display the current inventory using dictionaries rather than list objects. The starter code stores the entries in a list table where each “row” is a list object. I kept each of the user input prompts the same, which stored their entries as string objects. In order to make the change from a 2D list of lists to 2D list of dictionaries, I created a dictionary for where the values were the user entries, and the keys were ‘id’, ‘title’, and ‘artist’. This dictionary was then appended to the list Table to store the entry in memory.

In order to display the data, I simply modified the print statement to now when it reads in a row from the list table, it only returns the values of the dictionary. This was done using the .values() method on each row in the list table.

# Loading Existing Data

In order to load data from an existing file, it must first exist. In order to ensure the program would not break if the CDInventory.txt was not already created, I added a section of code at the start of the program that would attempt to open the file using the os.path function[[1]](#footnote-1). If it did not exist, the program would then create the file. This is shown in Figure 1.

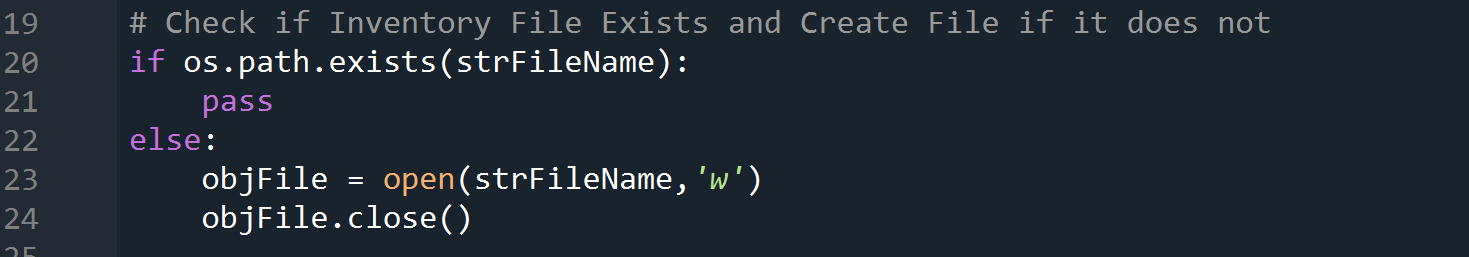


Figure 1: Check for Existing CDInventory File

When the user called to read in existing data, I used a for loop to read in each row of the CDInventory.txt file and pull out each piece of entry information to be temporarily stored in a list. This was done using the .strip() and .split() functions to remove new line characters and create a list with 3 entries. I then assigned each entry as a value to the dictionary that was subsequently appended to the list table of CD entries. After the data had been loaded, I wanted to have some method of tracking when the user pulled in existing information. This was done by creating a Boolean variable set to False at the beginning of the program and became True when data was loaded in by the user. This would become useful in saving the data later in the program.

# Saving Appended List or Entire Inventory

The existing code uses nested for loops to iterate through rows in the data table and entries in the entry lists. Since the program now had to operate on dictionaries, there was no simple index to loop through on each row. However, the .values() method for dictionary types will return the values of a dictionary in a list. By adding this method to the existing for loop, the dictionary entries for each row could be written to the file with minimal changes to the starter script.

Additionally, I recognized that with the starter script was written, if the user loaded existing data into memory and then tried to save the list back to the file, it would append the text file and there would be duplicate entries. In order to prevent this, I created an if statement that would check the Boolean variable previously mentioned for if the user had loaded in data. If they had, then the program would open the CDInventory.txt file for write only functionality. This meant that all existing data would be overridden in the file, assuming that the user wanted to view and/or edit the whole list. If the user had not loaded data from the file, then the save portion of the program would only append the new entries to the file.

# Deleting Entries

In order for the user to delete an entry in the data table stored in memory, I chose to prompt the user to enter the ID of the entry the wished to delete. I then used a for loop to cycle through each row and check for the entered ID against the value stored in the ‘id’ key of the row dictionary. If the values matched, I used the .remove function to delete that row from the data table.[[2]](#footnote-2)

# Executing the Program

The execution of the program is summarized in this section. To begin with, I created a CDInventory.txt file that contained 2 entries. I then executed the script in Spyder to load in the data and display its contents, shown in Figure 2. The next step I took was to remove the second entry, displayed the data, and then add the same the same entry back to the table. This step is shown in Figure 3. After these steps, I ran the program in the terminal window where I simply added an additional entry and saved the data to the file, shown in Figure 4. The CDInventory.txt file submitted for this assignment therefore contains these 3 entries.

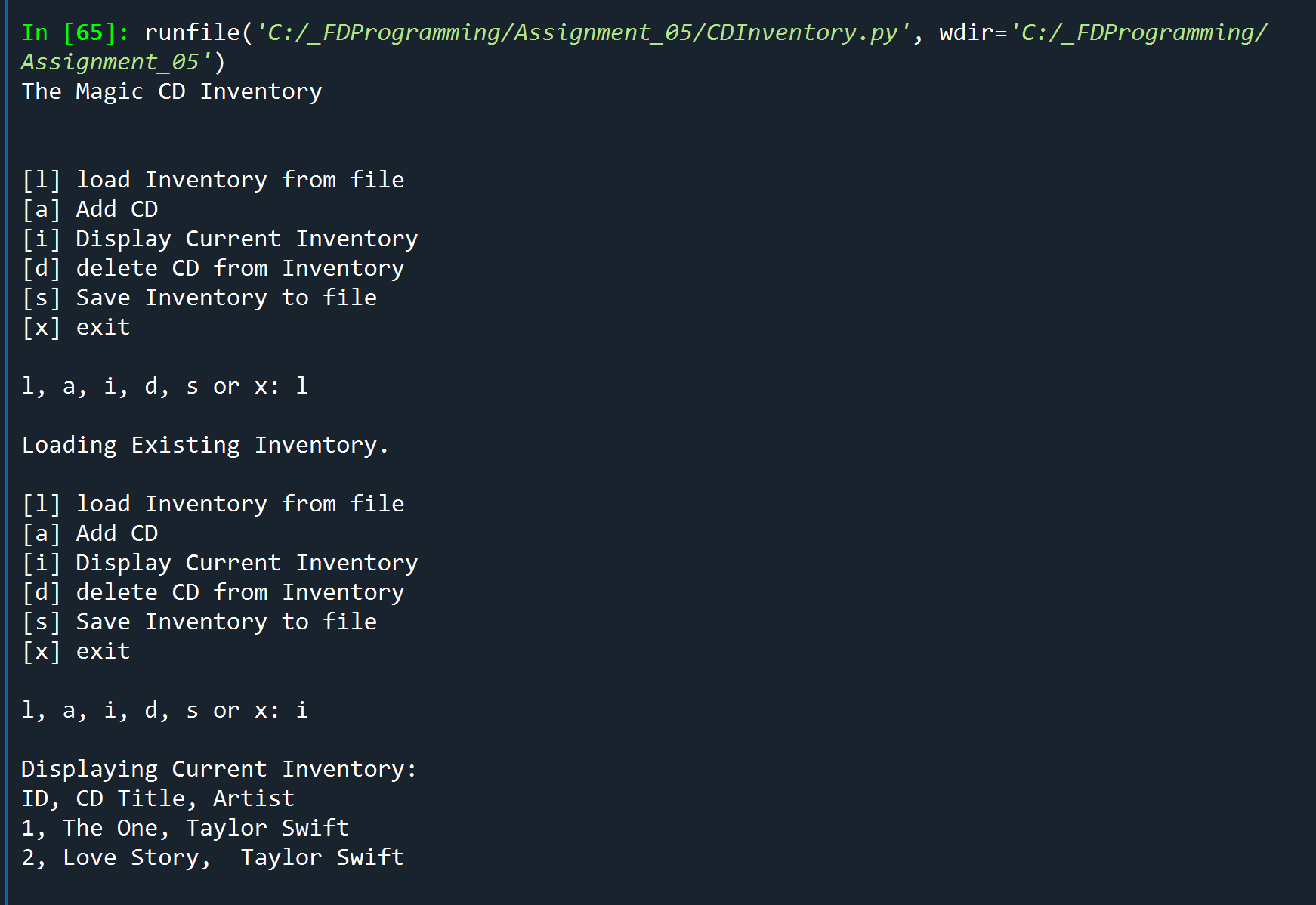
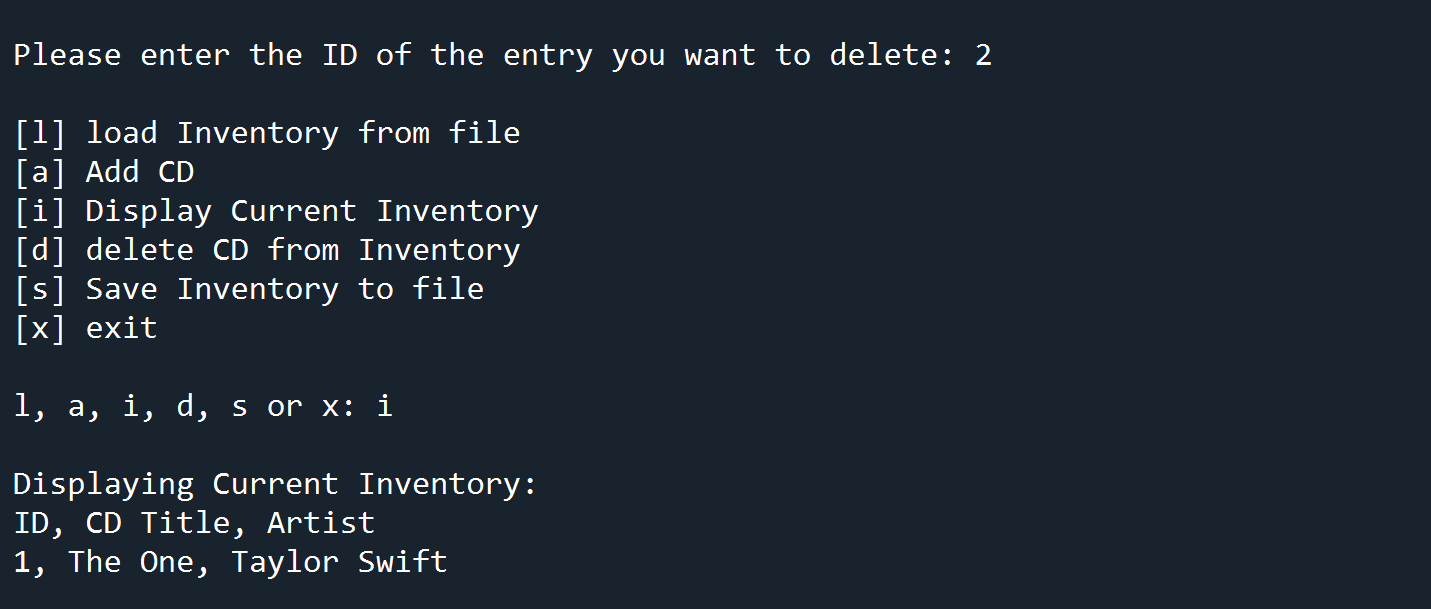


Figure 2: Spyder Execution – Load Data and Display Contents



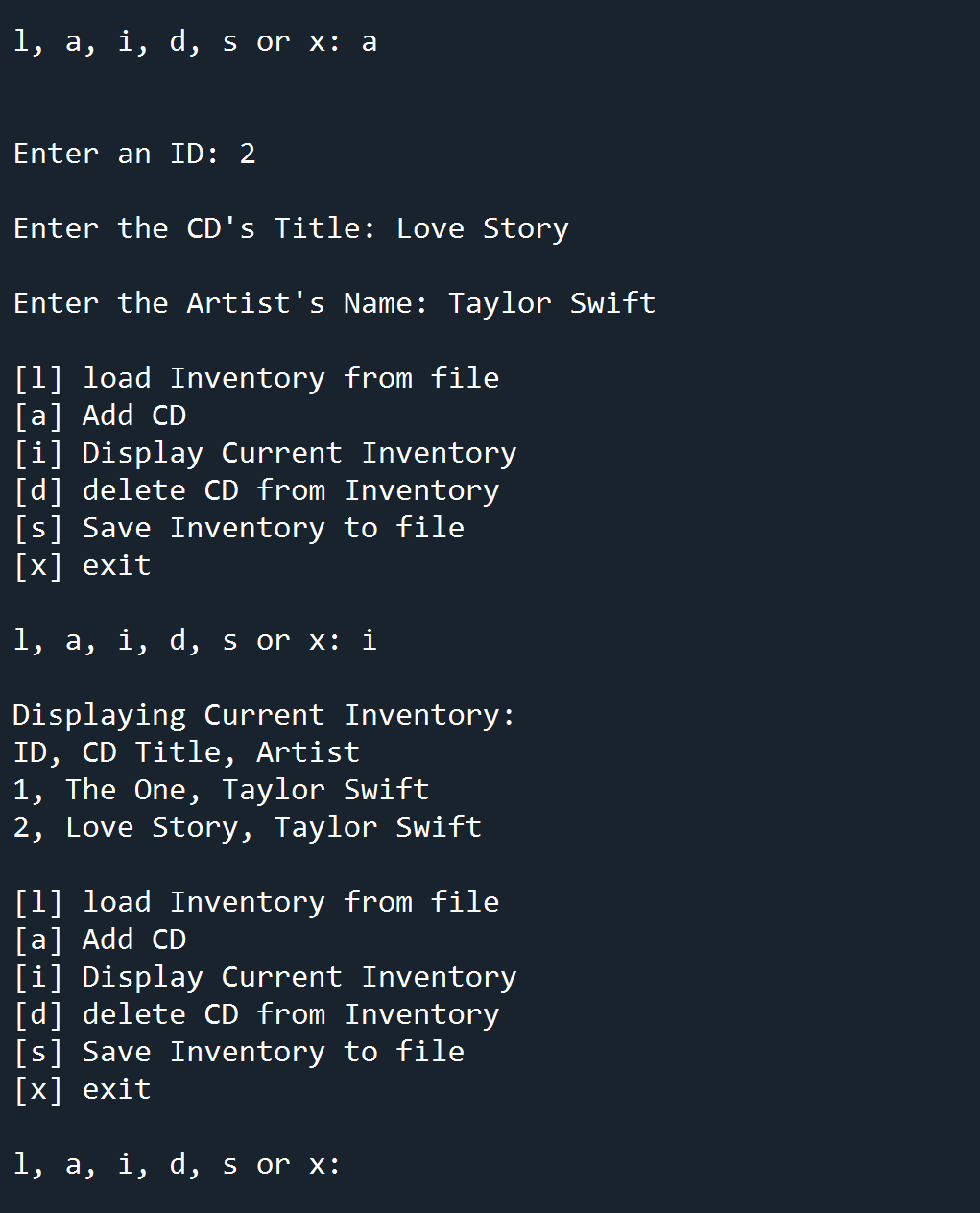


Figure 3: Spyder Execution (cont.) – Delete Entry and Re-Enter Data

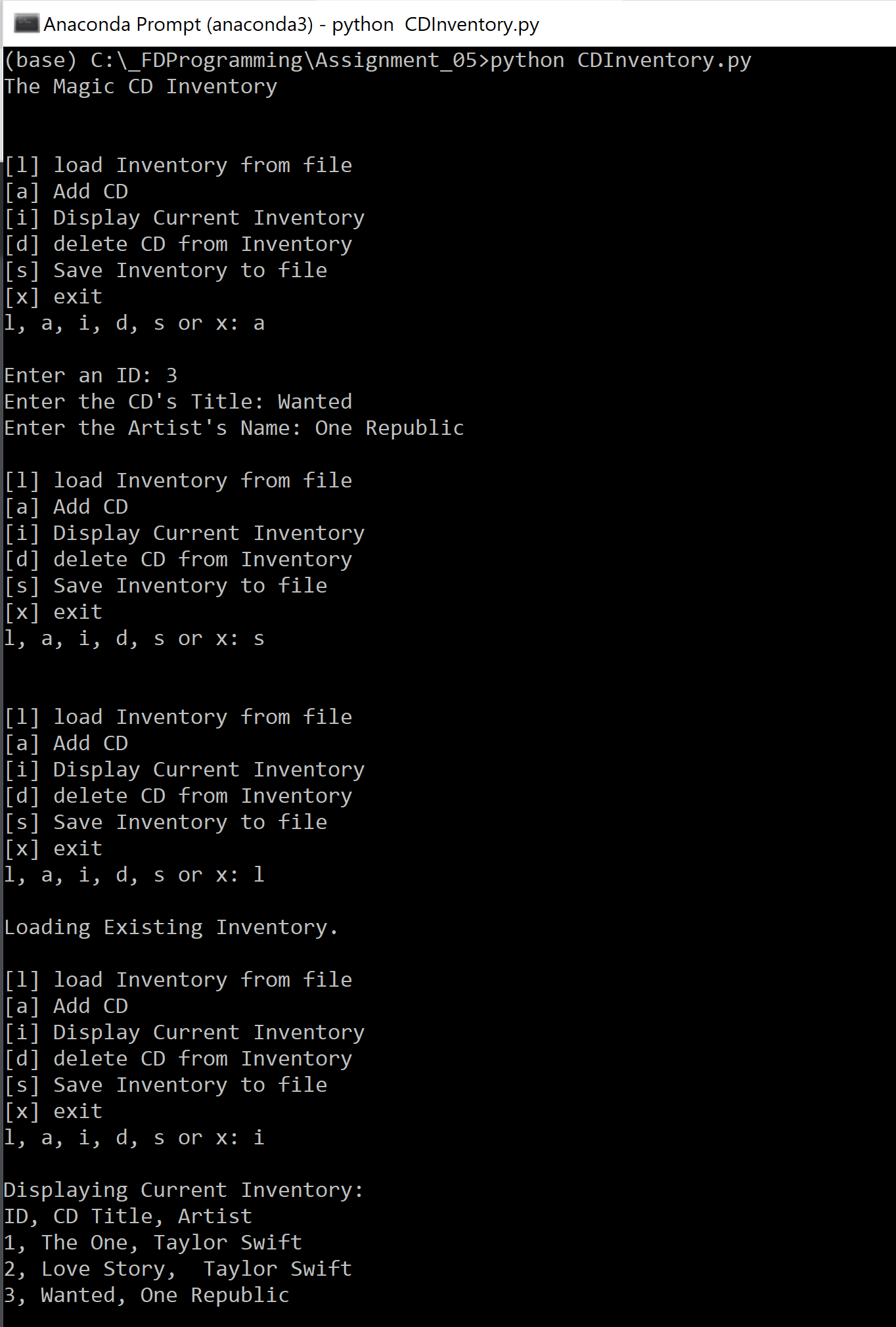


Figure 4: Terminal Execution – Add Data, Save to File, Reload to Display Save Success

# Summary

In this assignment, I was able to successfully take the starter script from Assignment 04 and modify another user’s existing code such that the program operated on dictionary objects rather than list entries. The program successfully adds the capability for the user to load in existing data and to delete entries prior to saving the data back to a text file. By doing so, I learned how manipulation of dictionaries differs from strings.

1. <https://docs.python.org/3/library/os.path.html> [↑](#footnote-ref-1)
2. Python Programming for the Absolute Beginner 3rd Edition, Michael Dawson, Cengage Learning 2010, P. 156 [↑](#footnote-ref-2)