More Loops, Sequences, Tuples, Lists

# Introduction

This module was based around understanding dictionaries, lists, error handling, using Separation of Concerns programming patterns, and creating and using a GitHub account.

# Adapting the CD Inventory Script

The goal of this assignment was to expand upon the CD inventory program made in Assignment 04. The new features include loading data from a text file and deleting a CD of choice from the list. Additionally, all the data will be stored in dictionaries instead of lists.

The final part of the challenge of this assignment is to adapt a script created by another individual. This is challenging since to adapt a program; you first must understand the basis of it and how the creator chose to write it. Additionally, the editor must have the skills to identify what elements of the program will need to be changed when adding new elements and figure out the best way forward.

## Changing Lists to Dictionaries

I started this assignment by converting the list structures to dictionaries. I redefined the variable ‘lstRow’ to ‘dicRow’ and ‘lstTbl’ to ‘dicTbl’. I adapted the structures in each local loop to write to dictionaries vice lists. This was an easy conversion in most cases. When adding new CD information, the list was simply changed to a dictionary. For saving the strings, when extracting info from the dictionaries stored in ‘dicTbl’, I used the .values() function to take the info I wanted from the dictionary and store it into the list. The section for displaying the CD selection did not have to be adapted extensively.

## Loading Data from Existing Text File

To load existing data, I first opened the file ‘CDInventory.txt’ to read only with the ‘open’ function. Then I was able to loop through the data in the text file, process it with the .strip() and .split() methods. These methods worked to separate the line of text into separate strings. I then assigned these strings to key-value in ‘dicRow’. I then appended this to ‘dicTbl’ to save the information to the program to access later.

## Deleting CDs from the Inventory

To allow users to delete a selected CD from the inventory. The easiest method for the user to choose a CD to delete was by ID number. This is the only unique identifier, as there could be multiple albums by the same artist or potentially different albums with the same name. However, the way I wrote the code, it ensured all CDs should have different ID numbers, as the ID number is assigned based on the number of items in the list storing the dictionaries.

Therefore, I first used the ‘input’ function, allowing the user to enter the ID associated with the CD they would like to delete. I then implemented a ‘for’ loop that looped through elements in ‘dicTbl’. I determined the number of elements using the ‘len()’ and ‘range()’ function, to give a start and end point to the for loop. I used an ‘if’ statement to test if the ‘id’ associated with the CD in question was equivalent to the user’s input. To ensure the values were comparable, I used ‘str()’ to convert both to strings. If the user input and ‘id’ value matched, I used the ‘del’ function to delete ‘dicTbl[i]’, which was the row associated with that function. After deleting the row, I used ‘break’ to exit the loop to ensure an error was not generated due to the length of the for loop changing mid iteration.

In order to ensure there weren’t going to be missing ID numbers or duplicates, I had the program loop back through ‘dicTbl’ and renumber the id numbers to equal to the position in the list.

# Running Script in Spyder IDE

The figures below show the result of running the ‘CDinventory.py’ script in Spyder IDE.

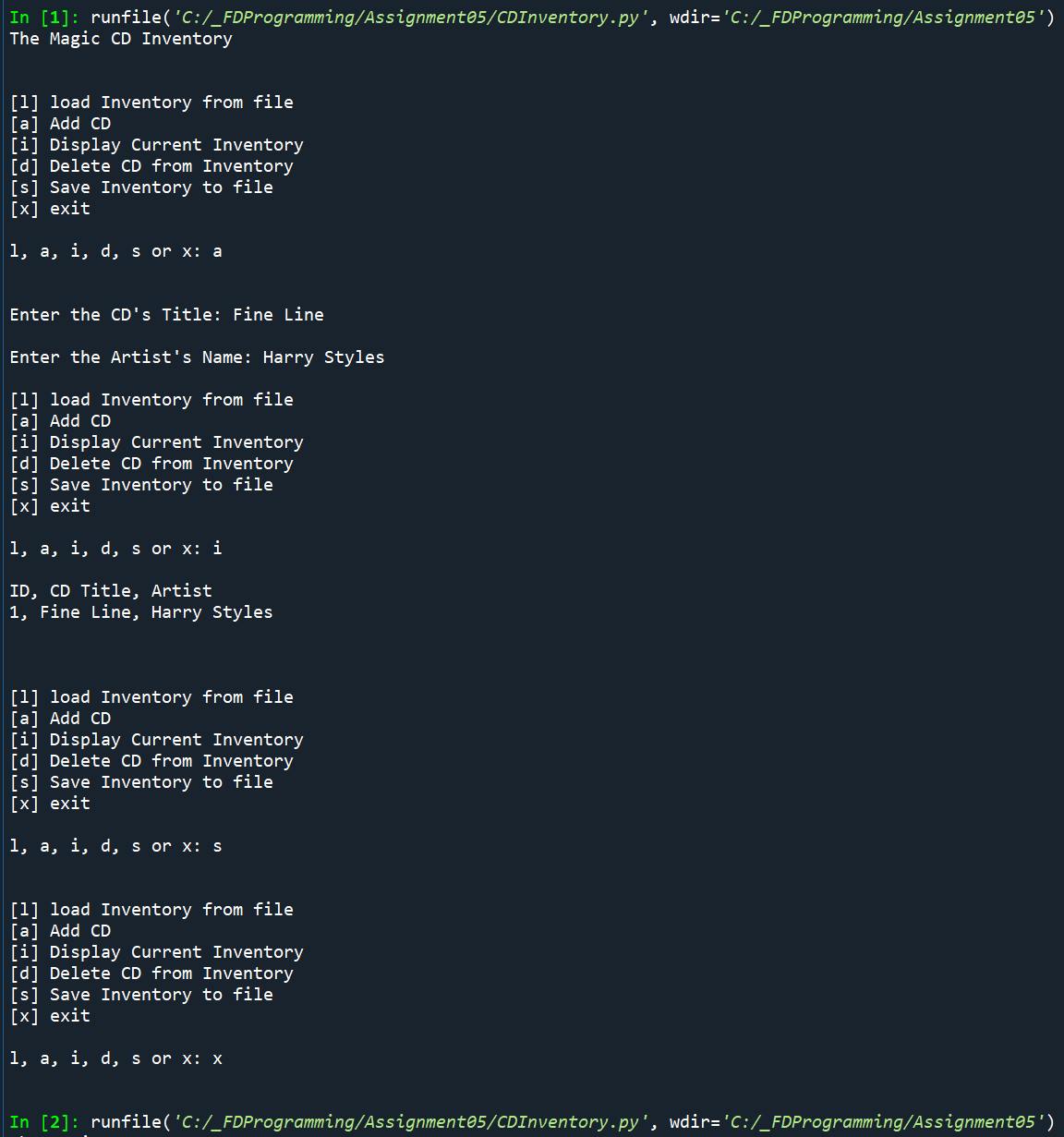


Figure 1 - CDinventory.py program in Spyder

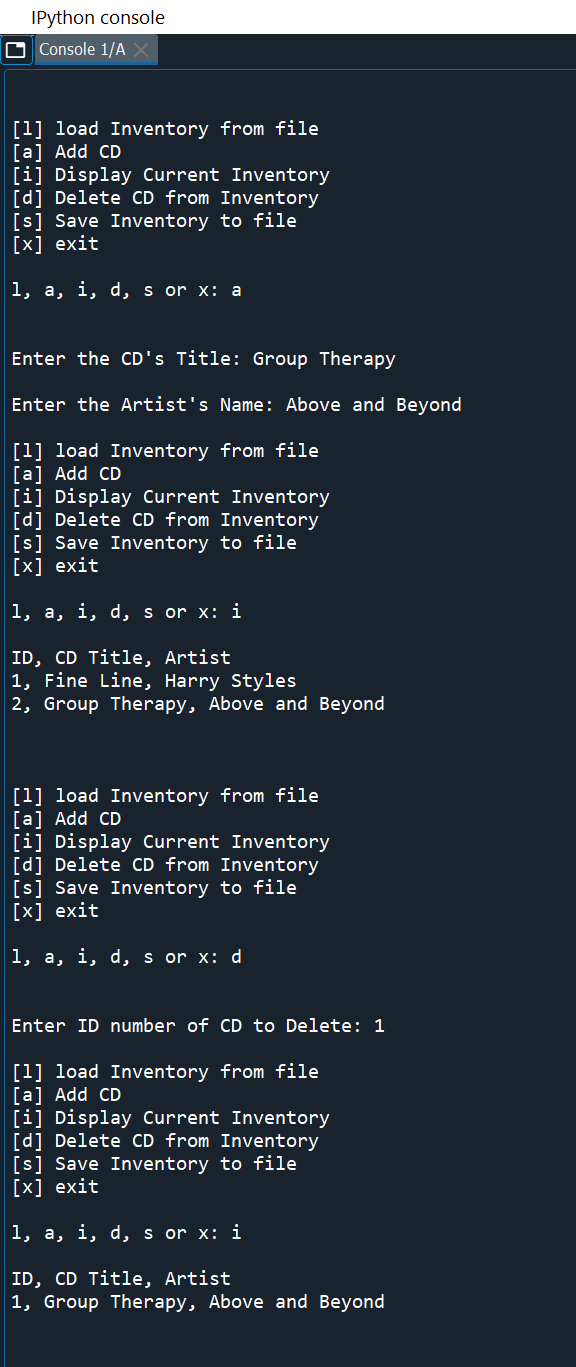


Figure - CDInventory.py in Spyder Cont.

The next figure shows the resulting text file after being run in Spyder:

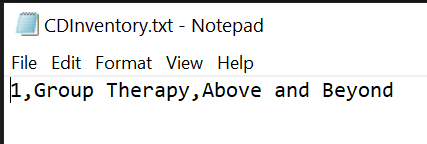


Figure 3 – CDInventory.txt File Generated from CDInventory.py

# Running Script in Terminal Window

The following figures are the result of running ‘CDInventory.py’ script in the terminal window.

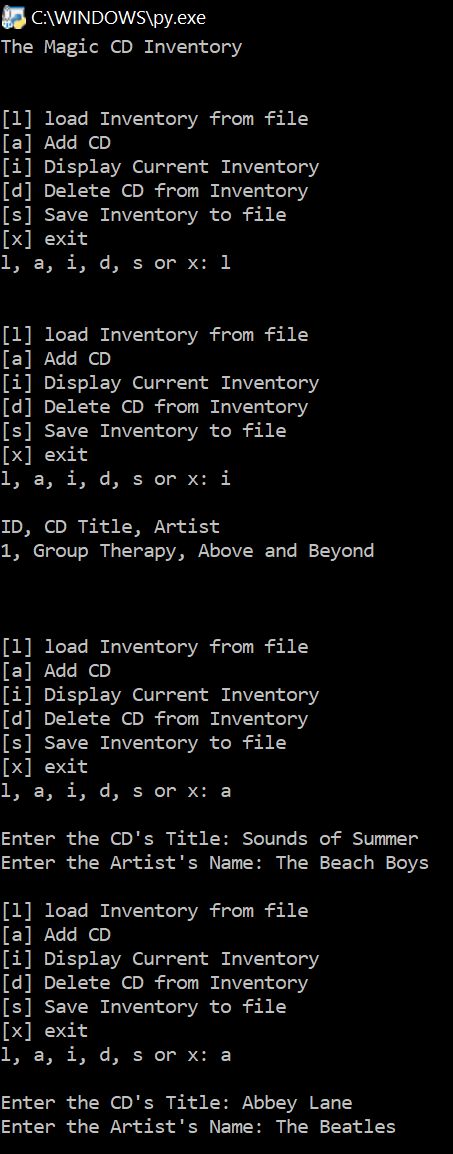


Figure 4 - CDInventory.py Program run in Terminal Window

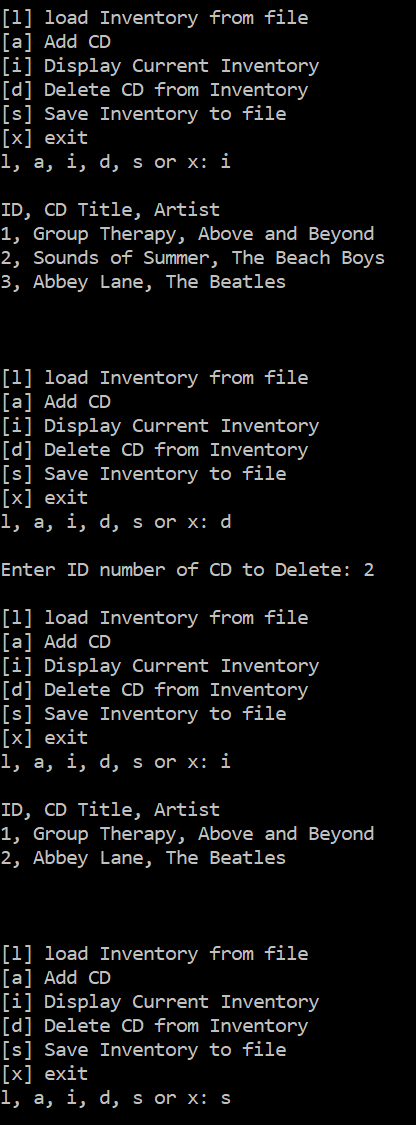


Figure 5 - CDInventory.py in Terminal Window Cont.

# 

Figure 4 – CDinventory.txt created by CDinventory.py in terminal window

# Summary

In this assignment I learned how to work with another individual’s code and create a functional product. I learned how to implement dictionaries, and how to read data from a text file.

Future improvements for this code are to automatically load any data in the text file, as the function I used to store the data in the file overwrites what is already there to prevent any duplicates of the info saving to the file. Additionally, I would consider creating a better method for numbering the CDs so they aren’t all re-numbered every time a CD with a number in the middle is deleted.

# Appendix

CDinventory Python script: