

Modified CD Inventory Script

INTRODUCTION

In Module 5 we learned about the Dictionary type and how to work with it, as well as some strategies for organizing and updating existing code. We then implemented some of these strategies by updating a starter CD Inventory script based on last week's assignment to change the internal data storage from list to dictionary format, while adding a few additional features to the program. Lastly, we were introduced to GitHub – a cloud-based software that facilitates using git version control, and collaboration on software.

GENERAL BACKGROUND

Dictionaries and Lists

Dictionaries and Lists are similar in that they hold pieces of data that are retrievable, but while data in lists are stored by index location, data in dictionaries are stored by key: value pairs. Keys are defined and used to store and access pieces of data (values). Lists are sequence types and dictionaries are mapping types.

A dictionary is indicated using braces "{ }" whereas a list uses brackets "[]".¹

Reading List and Dictionary Data from Files

List and dictionary data stored in text files as strings need to be converted back into the proper data type in memory if they are to function as lists and dictionaries in a program.

One way to read list data from a string file is to use a for loop to apply the split() function to each row, to turn it into a list. The strip() function is used to remove any excess spaces, tabs, or newlines at the beginning and end of each row. For example:

```
objFile = open(strFileName, 'r')
for row in objFile:
    lstRow = row.strip().split(',')
    print(lstRow)
objFile.close()
```

Figure 1 - Example of split & strip functions used while reading data to create lists (from Listing 3, Module 05 Doc, p. 3)

¹ Dirk Biesinger, Module 05 Doc, pp. 7-8.

The process of reading data into dictionaries from a file is similar, but requires an extra step of assigning the items in the list to keys to create dictionary rows. You can use index locations of data in a list to assign that data to a key in a dictionary. For example, from Listing 8:

```
objFile = open(strFileName, 'r')
for row in objFile:
    ... lstRow = row.strip().split(',')
    ... dicRow = {'id': int(lstRow[0]), 'name': lstRow[1], 'email': lstRow[2]}
    ... lstTbl.append(dicRow)
objFile.close()
```

Figure 2 - Example of reading data from text file to dictionary (Module 05 Doc, p. 11)

One of the reasons it makes sense to organize data in a 2-dimensional way is because it makes it easier (by using for loops) to read and write data to files while converting data types as needed.

Process Improvement Strategies

We were introduced to several strategies for improving scripts:

- Separation of Concerns (SoC)
- Functions
- Script Templates
- Structured Error Handling

SEPARATION OF CONCERNS (SOC)

“Separation of Concerns” is a programming pattern design principle that involves separating parts of code into distinct sections based on their primary purpose (addresses a specific “concern” or category of information). In general, programs can be divided into “Data,” “Processing,” and “Presentation” (input-output) sections.²

It makes sense to use these categories from the beginning of planning a script by using them to organize pseudocode.

FUNCTIONS

Functions allow coders to group multiple statements under a single name that can be called multiple times. By assigning variables and functions, it not only cuts down on repetitive statements (and thereby reduces changes for error due to inconsistency and makes editing easier), but it allows code to be grouped more efficiently by separation of concerns. For example, functions could be set ahead of time that will use user data but appear before the user inputs in the structure of the code. This allows the data processing to be separated from the input-output sections of the code.³

² Module 05 Doc, pp. 12-13

³ Module 05 Doc, p. 14

SCRIPT TEMPLATES

We learned how to update a script template in Spyder in order to save time when starting a new script by having a preset header and starter pseudocode using the typical programming pattern of “Data,” “Data-processing,” and “Presentation” sections.⁴

STRUCTURED ERROR HANDLING

Structured error handling is a method of writing ways to handle errors into the code itself to prevent the whole program from crashing in the event of an error. One example is to use the Try-Except construct to account for how the program should behave in the event of an error.⁵ We didn’t work with this much this week but it sounds promising!

GitHub

GitHub is a cloud-based hosting site that facilitates version control for software developers using Git. While Git can be used locally without GitHub, GitHub is a widely used and allows for easy collaboration on software, sharing files, and preserving version history.

This week we set up GitHub accounts and created our first repository in order to share our assignments for peer-review.

GitHub’s mascot is Octocat – part octopus, part cat.⁶

MODIFYING THE CD INVENTORY SCRIPT

This week we were tasked with updating the CD Inventory script from last week (using a starter version provided) by making the following modifications:

- Replace inner data structure with dictionaries
- Add functionality of loading existing data
- Add functionality of deleting an entry
- Use a list of dictionaries as a 2D table

Most of these modifications we practiced already in Lab05_B, for which we were helpfully provided a sample solution. I spent quite a bit of time trying to understand that lab so that helped with part of this assignment.

Replacing Inner Data Structure with Dictionaries

Since we were replacing a list of lists with a list of dictionaries, I changed the initial definitions from `lstTbl = []` and `lstRow = []` to `dictTbl = []` and `dictRow = { }`.

When adding new data, I just needed to change `lstRow` to `dictRow` and change the formatting to that of a dictionary with keys:

⁴ Module 05 Doc, pp. 15-16

⁵ Module 05 Doc, pp. 17-18

⁶ <https://en.wikipedia.org/wiki/GitHub#Mascot>, retrieved 08/08/2020

```
lstRow = [intID, strTitle, strArtist]
lstTbl.append(lstRow)
```

Listing 1 - Adding a row of list data to table in CD Inventory starter

```
dictRow = {'id': intID, 'artist': strArtist, 'title': strTitle}
dictTbl.append(dictRow)
```

Listing 2 - Adding a row of dictionary data in updated CD Inventory

Note: I also changed the order that the artist and title appear in the row because it makes more intuitive sense to me to list artists first before the album title (and I kept entering the data in the wrong places by accident when the order was reversed). This is a personal preference.

Updating the Exit Option

Like I did in last week's version, I added an extra step to the exit sequence to ask the user if they really want to exit. This is to prevent data loss in the event that they type "x" by accident at the menu prompt. (Appendix Listing, lines 36-41.)

Loading Existing Data

I decided loading existing data from a file would replace the data already in memory, but I would warn the user that this would happen and give them a choice whether or not to proceed. Otherwise, it is possible that the data loaded from the file might duplicate data already in memory or would result in duplicate IDs in memory.

I loaded the data from the file using `.strip` and `.split` functions for each row in the file and then using indexes to assign specific items in the row to specific keys in the variable `dictRow`, which I would in turn append to `dictTbl` using a `for` loop. This creates the `dictTbl` inventory in memory as a list of dictionaries.

I added a print statement telling the user that the data had been loaded. (Appendix Listing, lines 44-56.)

Adding New Data

After changing the data type of the new row created by user inputs, I added some "clear" options using `if` statements that would allow the user to stop entering this set of data into memory in the event that they make a mistake or change their minds.

I also added a check for already-existing IDs on the ID entered by the user. If the ID already exists, the user gets an error statement and the loop breaks. (Appendix Listing, lines 59-80.)

Displaying Inventory

I didn't really change this section much, but added formatting using the `.format` function in order to display the table in ordered lists. I created wide columns to account for long band and album names. (Appendix Listing, lines 83-89.)

Deleting a CD Entry (table row)

Thanks to some hints in class, I decided to use the `.remove` function to delete a row if the row's values contained the ID entered by the user. I used the statement:

```
for row in dictTbl:
    if delID in row.values():
        dictTbl.remove(row)
print('\nItem deleted.')
```

Listing 3 - Code to delete a row of data based on user-entered ID

However, I also wanted to return an error message if they entered an ID that didn't exist in the table. This proved a bit trickier for me. I first tried to use "not in" to check if it was a non-existing ID, which would then return an error. However, writing it this way did not end up returning the error I expected.

```
elif strChoice == 'd':
    delID = input('enter the ID of the entry you want to delete: ')

    if delID not in row.values() in dictTbl:
        print('\nSorry, that ID is not in the inventory.')

    else:
        for row in dictTbl:
            if delID in row.values():
                dictTbl.remove(row)
        print('\nItem deleted.')
```

Listing 4 - First attempt to create an error if invalid ID entered by user

```
ID | ARTIST | ALBUM |
1 | R.E.M. | Automatic for the People |
2 | Radiohead | OK Computer |
3 | Smashing Pumpkins | Mellon Collie and the Infinite Sadness |

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit

l, a, i, d, s or x: d

enter the ID of the entry you want to delete: 4

Item deleted.
```

Figure 3 - The code running with bug.

To test the code, I loaded a table with entry IDs 1-3 and then selected “4” as the ID for deletion. The “not in” statement did not work as I thought it would (to return the print statement “Sorry, that ID is not in the inventory.”)

Instead, it went to the else part of the if-else conditional and simply printed “Item deleted” (even though there was nothing to delete. I’m not sure why this didn’t work.

I think there must be an easier way to do this, but instead, I created a more complicated way of achieving this check.

I created an empty list as a checklist (chkLst = []) and then used a for loop to check each row for the presence of the user input ‘delID’ value and assigning either ‘T’ or ‘F’ as the value of a new variable ‘idChck’ which would then be appended to the chkLst. That way it would result in a list of true/false checks for each row. Then, I could use the “not in” keywords to check if ‘T’ not in that checklist, in which case, the error message would print. A bit cumbersome, but it works.

```
elif strChoice == 'd':
    delID = input('enter the ID of the entry you want to delete: ')

    # Sequence to check each row for the entered ID
    # There is probably a better way to set this up
    chkLst = []
    for row in dictTbl:
        if delID in row.values():
            idChck = 'T'
        else:
            idChck = 'F'
        chkLst.append(idChck)
    if 'T' not in chkLst: # Checks if entered ID doesn't exist
        print('\nSorry, that ID is not in the inventory.')

    else:
        for row in dictTbl:
            if delID in row.values():
                dictTbl.remove(row)
        print('\nItem deleted.')
```

Listing 5 - New code to check for non-existing entries

ID	ARTIST	ALBUM
1	R.E.M.	Automatic for the People
2	Radiohead	OK Computer
3	Smashing Pumpkins	Mellon Collie and the Infinite Sadness


```
[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit

l, a, i, d, s or x: d

enter the ID of the entry you want to delete: 4

Sorry, that ID is not in the inventory.
```

Figure 4 - Now the non-existent ID choice "4" returns the proper error message

Saving the Data

The save sequence overwrites what is currently in the data file with whatever data is currently in the program's memory. This part was done using similar methods to those used in the listings on p. 3 of the Module 5 document.

RUNNING THE FINAL CODE

Running in Spyder

```
In [73]: runcell(0, '/Users/brittaanson/_FDNProgramming/Assignment05/CDInventory.py')
The Magic CD Inventory

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit

l, a, i, d, s or x: l

Loading data from file will replace the table data currently in memory.
Do you want to proceed? Choose 'y' or 'n': y

Data loaded from file.

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit

l, a, i, d, s or x: i

ID | ARTIST | ALBUM |
1 | R.E.M. | Automatic for the People |
2 | Radiohead | OK Computer |

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit

l, a, i, d, s or x: a
```

Figure 5 - CDInventory.py running in Spyder Console [1 of 5]. Shows loading data from file, displaying inventory, and selecting "Add CD"

```
Enter new data, or type 'c' to clear and return to menu:

Enter an ID: 2
Error: that ID already exists in the inventory.

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit

l, a, i, d, s or x: a

Enter new data, or type 'c' to clear and return to menu:

Enter an ID: 3

Enter the Artist's Name: Smashing Pumpkins

Enter the CD's Title: Mellon Collie and the Infinite Sadness

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit

l, a, i, d, s or x: a

Enter new data, or type 'c' to clear and return to menu:

Enter an ID: 5

Enter the Artist's Name: c

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
```

Figure 6 – CDInventory.py running in Spyder Console [2 of 5]. Shows error when entering existing ID, adding a new entry, and using the “c” input to clear an entry after making a mistake on the first data input (entered ID “5” instead of “4”).


```
l, a, i, d, s or x: a
Enter new data, or type 'c' to clear and return to menu:
Enter an ID: 3
Error: that ID already exists in the inventory.

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit

l, a, i, d, s or x: a
Enter new data, or type 'c' to clear and return to menu:
Enter an ID: 4
Enter the Artist's Name: Bjork
Enter the CD's Title: Post

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit

l, a, i, d, s or x: a
Enter new data, or type 'c' to clear and return to menu:
Enter an ID: 5
Enter the Artist's Name: Beck
Enter the CD's Title: Odelay
```

Figure 7 - CDInventory.py running in Spyder Console [3 of 5]. Showing another mistake and then adding two more entries.

```
[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
```

l, a, i, d, s or x: i

ID	ARTIST	ALBUM
1	R.E.M.	Automatic for the People
2	Radiohead	OK Computer
3	Smashing Pumpkins	Mellon Collie and the Infinite Sadness
4	Bjork	Post
5	Beck	Odelay

```
[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
```

l, a, i, d, s or x: d

enter the ID of the entry you want to delete: 4

Item deleted.

```
[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
```

l, a, i, d, s or x: i

Figure 8 - CDInventory.py running in Spyder Console [4 of 5]. Showing display inventory after adding two entries, and selecting delete entry "4".

ID	ARTIST	ALBUM
1	R.E.M.	Automatic for the People
2	Radiohead	OK Computer
3	Smashing Pumpkins	Mellon Collie and the Infinite Sadness
5	Beck	Odelay

```
[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
```

l, a, i, d, s or x: s

```
[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
```

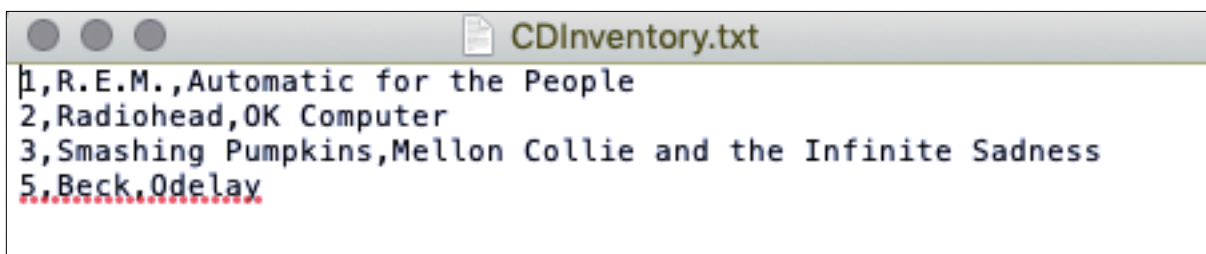
l, a, i, d, s or x: x

Are you sure you want to exit? Choose 'y' or 'n': y

In [74]: |

Figure 9 - CDInventory.py running in Spyder Console [5 of 5]. Showing inventory display after deleting entry with ID "4", save, and exit.

The text file written by the save selection:



```
1,R.E.M.,Automatic for the People
2,Radiohead,OK Computer
3,Smashing Pumpkins,Mellon Collie and the Infinite Sadness
5,Beck,Odelay
```

Figure 10 - CDInventory.txt file after running the program as shown in Figures 5-9

Running in the Terminal

```
Assignment05 — -bash — 94x57
[(base) britta-ansons-computer:Assignment05 brittaanson$ python CDInventory.py
The Magic CD Inventory

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: l

Loading data from file will replace the table data currently in memory.
Do you want to proceed? Choose 'y' or 'n': y

Data loaded from file.

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: i

ID | ARTIST | ALBUM |
1 | R.E.M. | Automatic for the People |
2 | Radiohead | OK Computer |
3 | Smashing Pumpkins | Mellon Collie and the Infinite Sadness |
5 | Beck | Odelay |

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: d

enter the ID of the entry you want to delete: 5

Item deleted.

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: i

ID | ARTIST | ALBUM |
1 | R.E.M. | Automatic for the People |
2 | Radiohead | OK Computer |
3 | Smashing Pumpkins | Mellon Collie and the Infinite Sadness |
```

Figure 11 - CDInventory.py running in the Terminal [1 of 3]. Showing loading data from file, displaying inventory, deleting an item, and displaying updated inventory..

```

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: a

Enter new data, or type 'c' to clear and return to menu:
Enter an ID: 2
Error: that ID already exists in the inventory.

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: a

Enter new data, or type 'c' to clear and return to menu:
Enter an ID: 4
Enter the Artist's Name: Counting Crows
Enter the CD's Title: August and Everything After

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: a

Enter new data, or type 'c' to clear and return to menu:
Enter an ID: 89
Enter the Artist's Name: c

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: a

Enter new data, or type 'c' to clear and return to menu:
Enter an ID: 5
Enter the Artist's Name: Florence and the Machine
Enter the CD's Title: Ceremonials

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: i

```

Figure 12 - CDInventory.py running in the Terminal [2 of 3]. Showing error when attempting to add existing ID, adding an item, clearing out of data entry after entering a mistake, and adding a second new entry.

ID	ARTIST	ALBUM
1	R.E.M.	Automatic for the People
2	Radiohead	OK Computer
3	Smashing Pumpkins	Mellon Collie and the Infinite Sadness
4	Counting Crows	August and Everything After
5	Florence and the Machine	Ceremonials

```

[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: s

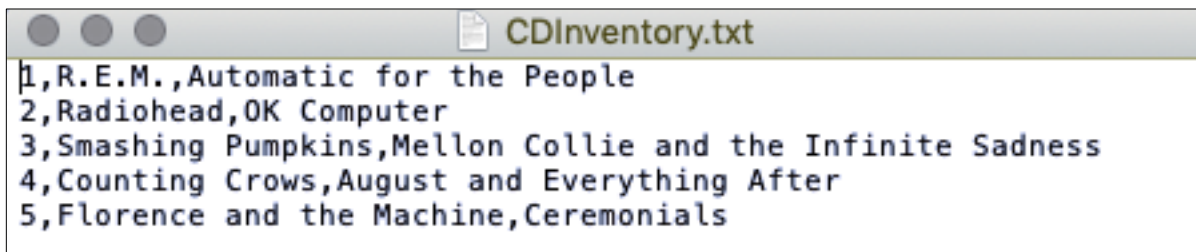
[l] Load inventory from file
[a] Add CD
[i] Display current inventory
[d] Delete CD from inventory
[s] Save inventory to file
[x] Exit
l, a, i, d, s or x: x

Are you sure you want to exit? Choose 'y' or 'n': y
(base) britta-ansons-computer:Assignment05 brittaanson$

```

Figure 13 - CDInventory.py running in the Terminal [3 of 3]. Displaying updated inventory, saving to file, and exiting program.

Here is what the CDInventory.txt file looks like after this second run of the program in the Terminal:



```

1,R.E.M.,Automatic for the People
2,Radiohead,OK Computer
3,Smashing Pumpkins,Mellon Collie and the Infinite Sadness
4,Counting Crows,August and Everything After
5,Florence and the Machine,Ceremonials

```

Figure 14 - CDInventory.txt file after running program in Terminal

POSTED TO GITHUB

The assignment documents are posted to my GitHub repository “Assignment_05” at:

https://github.com/bricolagerie/Assignment_05.

SUMMARY

The process of updating the CD Inventory to use dictionaries for internal storage allowed me to practice using tables made of dictionaries and practicing extracting and converting that data into different types for storage, retrieval, and display. I was not able to organize the code very clearly by separation of concerns in this version due to the order to the various loops. I can see how functions will allow us to reorganize this code differently next week!

APPENDIX

Listing CDInventory.py

```
1. #-----#
2. # Title: CDInventory.py
3. # Desc: Assignment 05 - Updated CD Inventory Script
4. # Change Log: (Who, When, What)
5. # BAnson, 2020-Aug-08, Created File
6. # BAnson, 2020-Aug-08, Update internal data to dictionary type
7. # BAnson, 2020-Aug-08, Added load from file, add data, display inventory, save
8. # BAnson, 2020-Aug-11, Added code for "delete" menu option
9. # BAnson, 2020-Aug-12, Added:
10. #         "continue" options to add data sequence
11. #         "are you sure?" to exit sequence
12. #         error message to delete sequence
13. #         existing ID check in "add CD" sequence
14. #-----#
15.
16.
17. # Declare variables
18.
19. strChoice = '' # User input
20. dictTbl = [] # list of dictionaries to hold data
21. dictRow = {} # dict of data row
22. strFileName = 'CDInventory.txt' # data storage file
23. objFile = None # file object
24.
25.
26. # Get user Input
27. print('The Magic CD Inventory\n')
28.
29. while True:
30.     # Display menu allowing the user to choose:
31.     print('\n[l] Load inventory from file\n[a] Add CD\n[i] Display current inventory')
32.     print('[d] Delete CD from inventory\n[s] Save inventory to file\n[x] Exit')
33.     strChoice = input('l, a, i, d, s or x: ').lower() # convert choice to lower case at time of input
34.     print()
35.
36.     if strChoice == 'x': # exit the program
37.         choice = input('Are you sure you want to exit? Choose \'y\' or \'n\': ') # prevent exiting accidentally
38.         if choice.lower() == 'y':
39.             break
40.         else:
41.             continue
42.
43.
44.     if strChoice == 'l': # load existing data from file
45.         loadChoice = input('Loading data from file will replace the table data currently in memory. \nDo you want to proceed? Choose \'y\' or \'n\': ')
46.         if loadChoice.lower() == 'n':
47.             continue
48.         else:
49.             dictTbl.clear() # clears existing table data in memory before loading data from file
50.             objFile = open(strFileName, 'r')
51.             for row in objFile:
52.                 lstRow = row.strip().split(',')
53.                 dictRow = {'id': lstRow[0], 'artist': lstRow[1], 'title': lstRow[2]}
54.                 dictTbl.append(dictRow)
55.             objFile.close()
56.             print('\nData loaded from file.')
57.
```

```

58.
59.     elif strChoice == 'a': # add data to the table
60.         print('Enter new data, or type \'c\' to clear and return to menu: ')
61.         strID = input('Enter an ID: ')
62.         if strID == 'c': # option to clear data at each stage of entry to prevent writing to memory
63.             continue
64.         # Check if entered ID already in inventory
65.         idInv = []
66.         for row in dictTbl:
67.             if strID in row.values():
68.                 idInv.append('T')
69.         if 'T' in idInv:
70.             print('Error: that ID already exists in the inventory.')
71.             continue
72.
73.         strArtist = input('Enter the Artist\'s Name: ')
74.         if strArtist == 'c':
75.             continue
76.         strTitle = input('Enter the CD\'s Title: ')
77.         if strTitle == 'c':
78.             continue
79.         dictRow = {'id': strID, 'artist': strArtist, 'title': strTitle}
80.         dictTbl.append(dictRow)
81.
82.
83.     elif strChoice == 'i': # Display current inventory in memory
84.         print('{:<4} | {:<30} | {:<40} |'.format('ID', 'ARTIST', 'ALBUM'))
85.         for row in dictTbl:
86.             lstRow = [] # create a list to populate with dictionary values
87.             for item in row.values():
88.                 lstRow.append(item)
89.             print('{:<4} | {:<30} | {:<40} |'.format(lstRow[0], lstRow[1], lstRow[2]))
90.
91.
92.     elif strChoice == 'd':
93.         delID = input('enter the ID of the entry you want to delete: ')
94.
95.         # Sequence to check each row for the entered ID
96.         # There is probably a better way to set this up
97.         chkLst = []
98.         for row in dictTbl:
99.             if delID in row.values():
100.                 idChck = 'T'
101.             else:
102.                 idChck = 'F'
103.             chkLst.append(idChck)
104.             if 'T' not in chkLst: # Checks if entered ID doesn't exist
105.                 print('\nSorry, that ID is not in the inventory.')
106.
107.         else:
108.             for row in dictTbl:
109.                 if delID in row.values():
110.                     dictTbl.remove(row)
111.                 print('\nItem deleted.')
112.
113.
114.     elif strChoice == 's':
115.         objFile = open(strFileName, 'w') # save to file (this overwrites existing file data)
116.
117.         for row in dictTbl:
118.             strRow = ''
119.             for item in row.values():
120.                 strRow += str(item) + ','
121.             strRow = strRow[:-1] + '\n'
122.             objFile.write(strRow)

```



```
122.         objFile.close()
123.
124.
125.     else:
126.         print('Please choose either l, a, i, d, s or x!')
```