Name: Salvador Gavonel

Date: November 11, 2021

Course: IT FDN 110 B

Assignment 05

CD Inventory – Dictionaries Script

Introduction:

Module 05 introduces another sequence called 'dictionaries'. Dictionaries have similarities with lists, with a few exceptions. Dictionaries uses 'key:value' pairs as a way to 'index' its information; a key being any value type that cannot be modified (cannot be a list) that is used to identify the value that it holds.

```
In [217]: dicRow
Out[217]: {'ID': '2', 'Title': 'Runrig', 'Artist': 'The Big Wheel'}
```

Figure 1 - Dictionaries hold information using 'Key:Value' pairs

Assignment 05 continues with the CD:Inventory script to introduce dictionaries as a way to substitute lists, and also adds two more functionalities to the script: deleting entries from the in-memory list and loading CD entries from a text file.

Reading New Code:

Opening the new 'CDInventory_Starter.py' I was surprised to find the structure was different from what the Assignment04 starter pseudocode described. I assume it is part of the challenge and that while code is being developed it makes sense to move things around so that there is an overall sense of direction for the script. I understand now how important it is to add comments for new users to understand how the code works, it was the only way I was able to navigate through the code and understand where was what. With longer scripts, it gets very confusing to understand where is what so I'm very grateful we were able to look into ways to organize and make our code simpler, I'm looking forward to use functions in python.

Converting from Lists to Dictionaries:

Defining Variables:

I did not have a huge number of difficulties performing this task; practicing the LAB exercises and the examples in the modules and videos really help sink in the concepts that are being taught. I started by changing the variable definition at the top of the script: the 'listrow' variable would be switched by the 'dicrow' variable to be used to process data later in the script. The 2D list would still be used as the table to hold the dictionaries in them; I'm assuming

it doesn't really make sense to use a dictionary to hold each dictionary since there is no use for the 'key:value' functionality, the dictionaries all have the same structure.

```
10  strChoice = '' # User input
11  lstTbl = []  # list of lists to hold data
12  # TODO replace list of lists with list of dicts
13  dicRow = {}  # dictionary of data row
14  strFileName = 'CDInventory.txt'  # data storage file
15  objFile = None  # file object
```

Listing 1 - Defining Variables

Adding new Data to the Table:

```
elif strChoice == 'a':  # no elif necessary, as this code is only reached if strChoi
    # 2. Add data to the table (2d-list) each time the user wants to add data
    strID = input('Enter an ID: ')
    strTitle = input('Enter the CD\'s Title: ')
    strArtist = input('Enter the Artist\'s Name: ')
    intID = int(strID)
    dicRow = {'ID':strID, 'Title':strTitle, 'Artist':strArtist}
    lstTbl.append(dicRow)
```

Listing 2 - Adding Entry to the Table with Dictionaries

I had two choices to change how I added new information to the table: using variables or directly adding them to the dictionary. I decided to go with the former to fill out the 'dicRow' variable at the end and visually make sure that the structure is what I want it to be.

Displaying the table Data:

```
    elif strChoice == 'i':
        # 3. Display the current data to the user each time the user wants to display the
        print('ID, CD Title, Artist')
        for row in lstTbl:
            print(row['ID'], row['Title'], row['Artist'],sep = ', ')
```

Listing 3 - Displaying Data with Dictionaries

Displaying the Data is the simplest part of the code, however I had the most difficulty trying to display it. I used all the methods to 'unpack' dictionaries but they all appeared with they key or an additional element I didn't want to show. Going through my notes I found that I could just use display each value in a dictionary by using the key as the index. From here the difficulty would be to format the data so It would be readable to the user.

PosData: I did go back and try to work with the .items() method and was able to make it work with the following code:

```
In [225]: for row in lstTbl:
     ...:     for key, val in row.items():
     ...:         print(val, end='\t\t|')
     ...:     print()
     ...:
2       |Runrig     |The Big Wheel      |
1       |Bad        |Michael Jackson        |
```

Listing 4 - Alternate way to display Table

Writing Data to a File:

```
59      elif strChoice == 's':
60          # 4. Save the data to a text file CDInventory.txt if the user chooses so
61          objFile = open(strFileName, 'a')
62          for row in lstTbl:
63              strName = ''
64              for key, val in row.items():
65                  strName += str(val) + ','
66              strName = strName[:-1] + '\n'
67              objFile.write(strName)
68          objFile.close()
```

Listing 5 - Writing Data to a File using Dictionaries

To write the list into a file I didn't use a dictionary variable so there was no need to replace any variables, but the for statement did need to change to reference the values in a dictionary rather than a list. In the nested for loop, I used the '.items()' method with the 'key, val' elements to pull the value from each dictionary. I had to test the .items() method to figure out the right way to do this, using only 'val' without the 'key' returns a different list:

```
In [230]: for row in lstTbl:
     ...:     for val in row.items():
     ...:         print(val, end='\t\t|')
     ...:     print()
('ID', '2')     |('Title', 'Runrig')        |('Artist', 'The Big Wheel')        |
('ID', '1')     |('Title', 'Bad')       |('Artist', 'Michael Jackson')      |
('ID', '2')     |('Title', 'Runrig')        |('Artist', 'The Big Wheel')        |
('ID', '2')     |('Title', 'Runrig')        |('Artist', 'The Big Wheel')        |

In [231]: for row in lstTbl:
     ...:     for key, val in row.items():
     ...:         print(val, end='\t\t|')
     ...:     print()
2       |Runrig     |The Big Wheel      |
1       |Bad        |Michael Jackson        |
2       |Runrig     |The Big Wheel      |
2       |Runrig     |The Big Wheel      |
```

Figure 2 - Testing .items() method for the for loop

Reading Data from a .txt File:

To implement this functionality I used the example in the Module 05 notes:

```
30          # TODO Add the functionality of loading existing data
31          objFile = open(strFileName, 'r')
32          for row in objFile:
33              lstRow = row.strip().split(',')
34              dicRow = {'ID':lstRow[0], 'Title':lstRow[1], 'Artist':lstRow[2]}
35              lstTbl.append(dicRow)
36          objFile.close()
```

Listing 6 - Reading Data from a .txt file

What the code does is use a list to read every row from the file, using commas to separate the string into separate entries. The dictionary is then filled out using the list, using indexes to determine what value to pull from the list. The dictionary is then appended into the 2D list and the for loop moves on to the next row.

Deleting an Entry from the Table:

This functionality wasn't reviewed in the modules, but I went ahead and looked at the list methods that I could use to delete an entry in the table. I decided to have a prompt that asked what ID the user would like to have deleted, and whichever entry matched the ID would have it's entry deleted. I used a for loop that would go through each row in the table, and an if statement that checked if the ID matched the ID the user wanted deleted.

I initially used the .clear() method to clear the row, but that just cleared that dictionary and not the entry from the table, I kept getting error messages when I tried to display the table after I deleted an entry. I then went ahead to use the .remove() method to remove the dictionary whose entry had the same values as the row in the loop.

```
51      elif strChoice == 'd':
52          # TODO Add functionality of deleting an entry
53          DeleteID = input('Input the ID to delete: ')
54          for row in lstTbl:
55              if row['ID'] == DeleteID:
56                  lstTbl.remove(row)
57              else:
58                  continue
```

Listing 7 - Deleting an Entry

Running Code:

```
In [232]: runfile('C:/Users/Pc/Documents/Fall Classes 2021/Python Programming/
Assignment05/CDInventory_Dictionary.py', wdir='C:/Users/Pc/Documents/Fall Classes
2021/Python Programming/Assignment05')
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:
```

Figure 3 - Running Menu (Spyder)

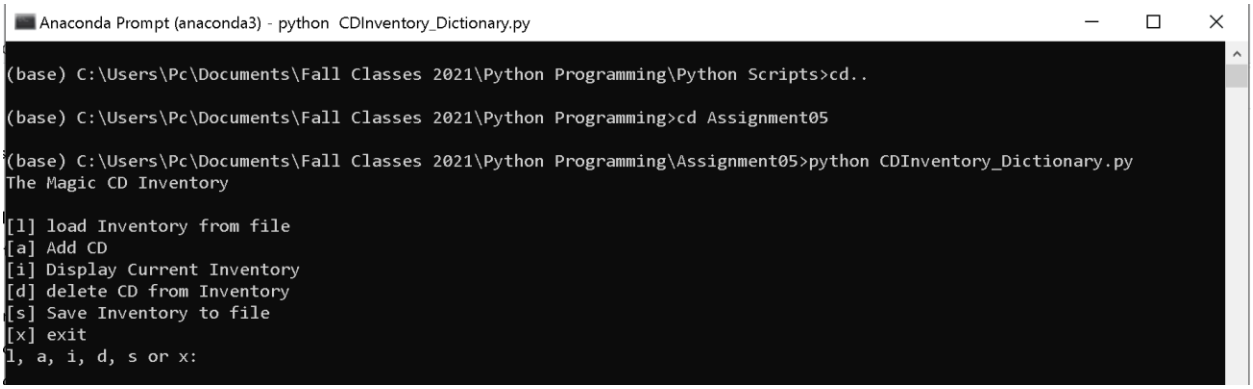

Figure 4 - Running Menu (Terminal)

```
In [232]: runfile('C:/Users/Pc/Documents/Fall Classes 2021/Python Programming/
Assignment05/CDInventory_Dictionary.py', wdir='C:/Users/Pc/Documents/Fall Classes
2021/Python Programming/Assignment05')
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: a


Enter an ID: 1

Enter the CD's Title: Are You Gonna Go My Way

Enter the Artist's Name: Lenny Kravitz
```

Figure 5 - Entering New Data (Spyder)

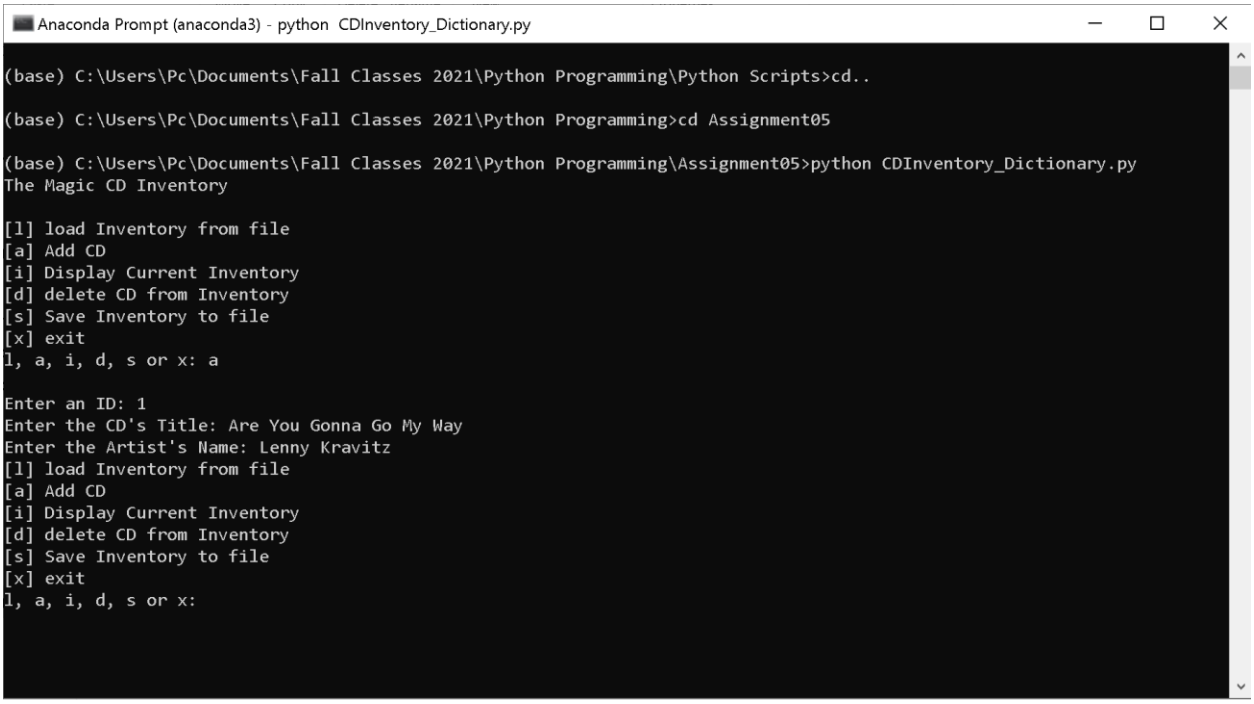

Figure 6 - Entering New Data (Terminal)

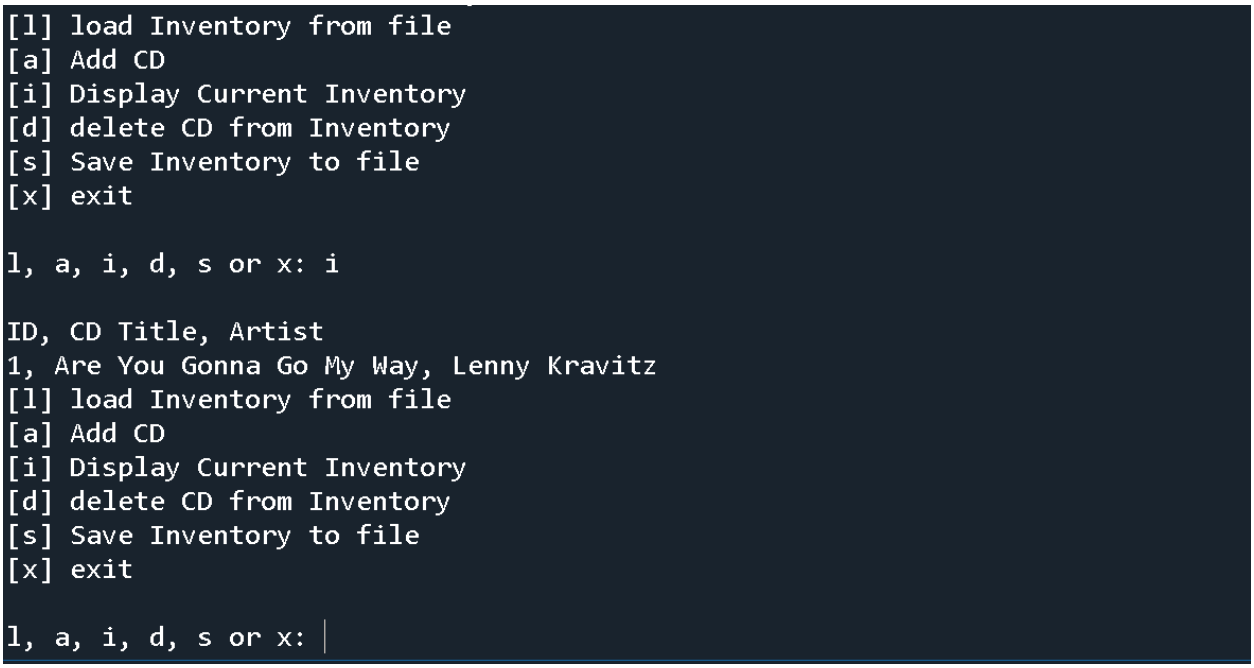

Figure 7 - Displaying Data (Spyder)

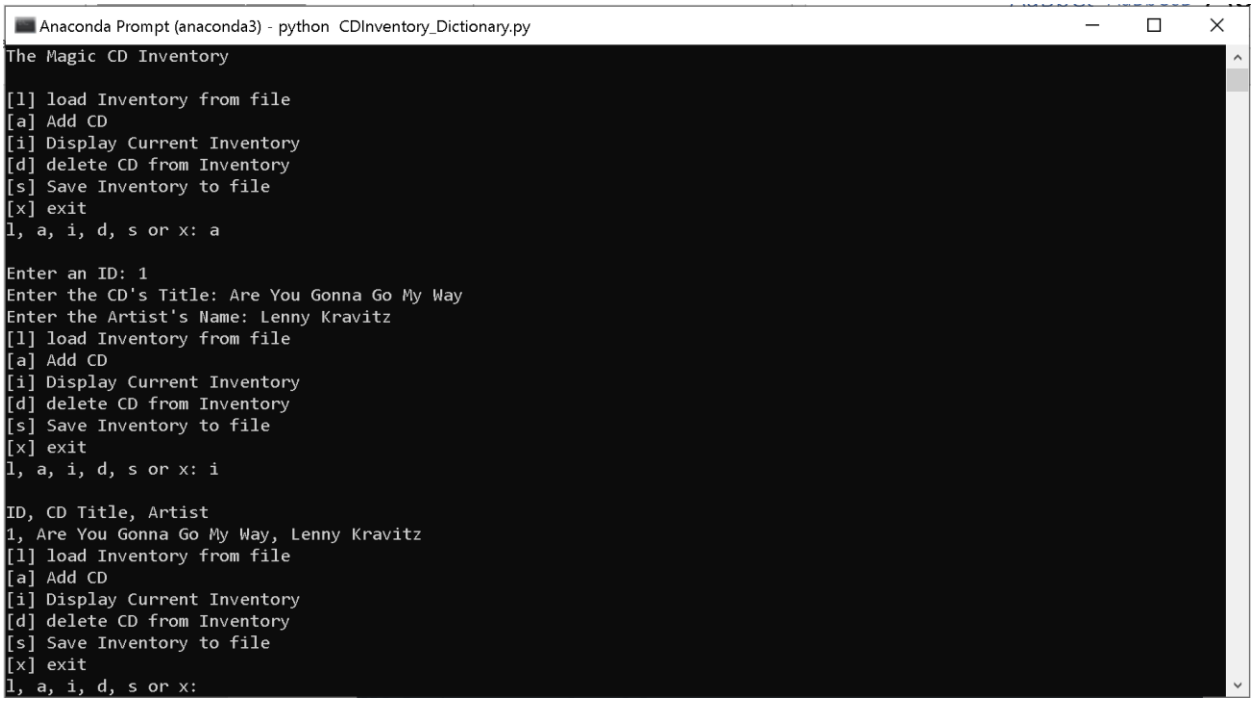

Figure 8 - Displaying Data (Terminal)

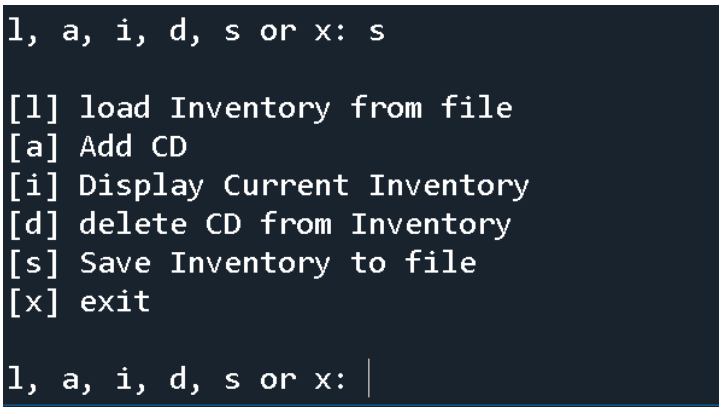

Figure 9 - Writing to .txt file (Spyder)

Figure 10 - .txt File with Entry (Spyder)

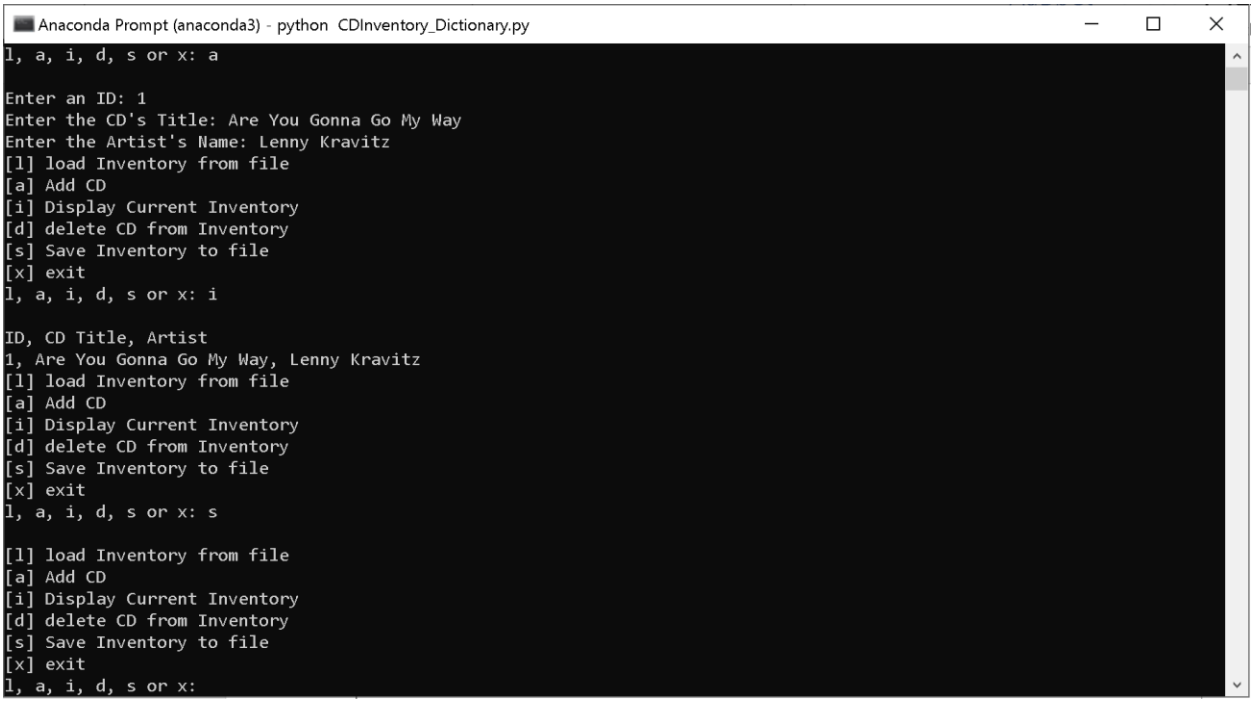

Figure 11 - Writing to .txt File (terminal)

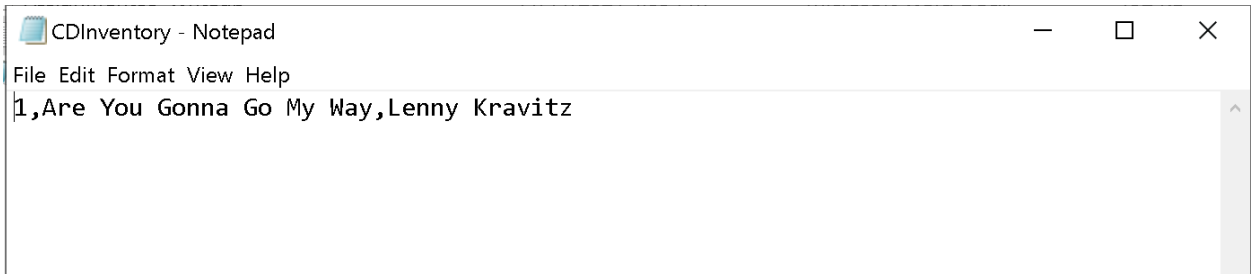

Figure 12 - . txt File with Entry (Terminal)

```
[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


Input the ID to delete: 1
[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, CD Title, Artist
[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:
```

Figure 13 - Deleting Entry from Table (Spyder)

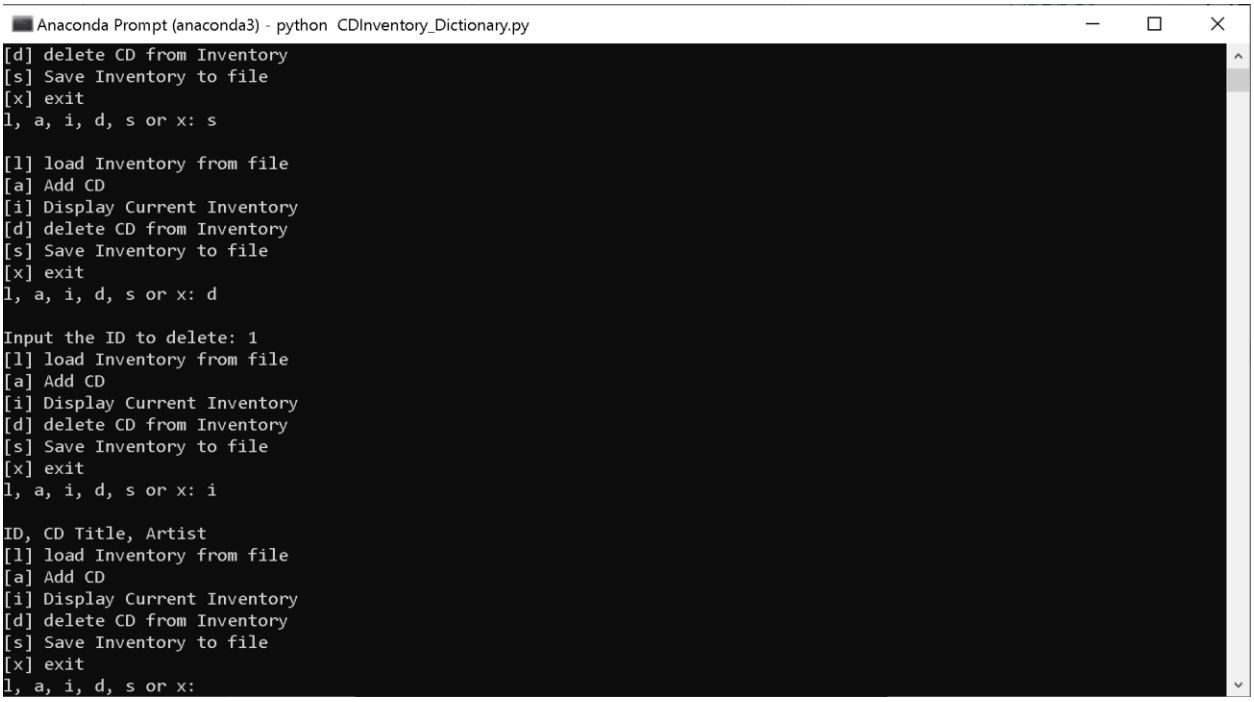

Figure 14 - Deleting Entry from Table (Terminal)

```
ID, CD Title, Artist
[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

[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, CD Title, Artist
1, Are You Gonna Go My Way, Lenny Kravitz
[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:
```

Figure 15 - Loading Data from a File (Spyder)

```
Anaconda Prompt (anaconda3) - python  CDInventory_Dictionary.py
[s] Save Inventory to file
[x] exit
l, a, i, d, s or x: i

ID, CD Title, Artist
[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

[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, CD Title, Artist
1, Are You Gonna Go My Way, Lenny Kravitz
[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:
```

Figure 16 - Loading Data from a File (Terminal)

GitHub Repository Link: https://github.com/Sgavon/Assignment05

Summary:

The CDInventory has evolved since the last assignment, with new functionalities and a new table of dictionaries instead of lists. While the code is better and the script can do additional things, it has also become more cumbersome and harder to read. Separation of concerns and functions(), although they are briefly covered in this assignment, will be needed to create more longer and more complex code.