Joseph Prado

02 March 2020

## IT FDN 100 B

Assignment 05

**Modifying an existing script to use dictionaries on the inner list and add functionalities**

# Introduction

In this assignment, I will do several things.

1. Review and existing script to see how it works.
2. Modify the script to use dictionaries as the inner data structure.
3. Add functionality so that user can load existing data from a text file of CD inventory.
4. Add functionality so that user can delete and entry.

# Reviewing the script

**Okay, there is nothing earth shattering here. This seems very similar the script used in Lab05a where a menu is displayed followed by a while loop and a series if elifs statements that address every possible selection in the menu.**

# Modifying the script to use dictionaries

1. strChoice = '' # User input
2. lstTbl = []  # list of lists to hold data
3. # TODO replace list of lists with list of dicts
4. lstRow = []  # list of data row
5. strFileName = 'CDInventory.txt'  # data storage file
6. objFile = None  # file object

**Defining variables: Ok, nothing to change here so far. I might have to revisit this later as I change references later in the code to dictionaries**

1. # Get user Input
2. **print**('The Magic CD Inventory\n')
3. **while** True:
4. # 1. Display menu allowing the user to choose:
5. **print**('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
6. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
7. strChoice = input('l, a, i, d, s or x: ').lower()  # convert choice to lower case at time of input
8. **print**()

**While statement and menu presentation – Nothing to change here so far, either. No references to lists yet.**

1. **if** strChoice == 'x':
2. # 5. Exit the program if the user chooses so
3. **break**
4. **if** strChoice == 'l':
5. # TODO Add the functionality of loading existing data
6. **pass**

**So far so good. No references to lists yet, but this might change as I add my TODOs.**

1. **elif** strChoice == 'a':  # no elif necessary, as this code is only reached if strChoice is not 'exit'
2. # 2. Add data to the table (2d-list) each time the user wants to add data
3. strID = input('Enter an ID: ')
4. strTitle = input('Enter the CD\'s Title: ')
5. strArtist = input('Enter the Artist\'s Name: ')
6. intID = int(strID)
7. lstRow = [intID, strTitle, strArtist]
8. lstTbl.append(lstRow)

**We need to modify this code to use dictionaries for the inner data structure. We encounter the first list at when we get the list lstRow = [ intID, strTitle, strArtist]. We know this is the inner data structure, since in the next line of code it gets appended inside another list named lstTbl, which is defined, likewise, at the beginning of the script as a table.**

**Let’s first make it clear that we’re using a dictionary by renaming lstRow as dicRow. I’ll need to add this list to the beginning of the script where define variables. Since we’re only using a dictionary for the inner data structure, we don’t need to change the name for the outer list lstTbl.**

**Since each dictionary entry has a key and value**[[1]](#footnote-1)**, we must replace the three items of inner list each with key:value pair and enclose the entire thing in curly brackets.**[[2]](#footnote-2)

**Our dictionary will look like this**

 dicRow = {'ID':intID, 'CD Title':strTitle, 'Artist':strArtist}

1. **elif** strChoice == 'i':
2. # 3. Display the current data to the user each time the user wants to display the data
3. **print**('ID, CD Title, Artist')
4. **for** row **in** lstTbl:
5. **print**(\*row, sep = ', ')

**Here is another place we will need to modify the code, specifically the print statement indented under the ‘for’ loop. The first print statement indicates that only the values of the dictionary will be printed. The second print statement as it currently is will print out both values…we don’t want that. To fix that, I will append the “.values()” dictionary method to to “row” so that only the values in the dictionary are printed.**[[3]](#footnote-3) **Also, note that instead of a second nested “for” statement to unpack the inner dictionary, we’ll use the newly introduced \* operator.**[[4]](#footnote-4)

The result is as follows:

**print**(\*row.values(), sep = ', ')

1. **elif** strChoice == 'd':
2. # TODO Add functionality of deleting an entry
3. **pass**

**Nothing we need to do here right now.**

1. **elif** strChoice == 's':
2. # 4. Save the data to a text file CDInventory.txt if the user chooses so
3. objFile = open(strFileName, 'a')
4. **for** row **in** lstTbl:
5. strRow = ''
6. **for** item **in** row:
7. strRow += str(item) + ','
8. strRow = strRow[:-1] + '\n'
9. objFile.write(strRow)
10. objFile.close()

**Here we’re writing to the text file. The inner data structure is stored to memory as a dictionary. Just like the print statement previously, we only want to write values to the text file. Similar to before, I will append the .values() dictionary method as each row as it’s unpacked.**

**for** item **in** row.values():

1. **else**:
2. **print**('Please choose either l, a, i, d, s or x!')

**Nothing we need to do here.**

**That wraps up modifying the inner data structures with dictionaries.**

# Adding the functionality of loading existing data.

1. **if** strChoice == 'l':
2. # TODO Add the functionality of loading existing data
3. **pass**

**To load existing data in the memory, we need to pull the values out of the text file, associate it with keys and append it to our lstTbl list**

**First, we will clear out the list lstTbl using the clear(). This will prevent the user from duplicating the data in memory if he/she reads it from the file multiple times. This tactic was demonstrated in the example solution for LAB 05-A.**[[5]](#footnote-5)

**﻿**  lstTbl.clear()

**We continue by** opening the existing text file using the open() function with mode ‘r’ to note that we would like to read from file.[[6]](#footnote-6)

﻿  objFile = open(strFileName, 'r')

**We follow up this us by iterating through the rows of the text file with a for statement to unpack the rows. Before assigning the row to the list lstRow, we will apply the .split() functions to split the string into list, defining the ‘,’ as the separator.**[[7]](#footnote-7) **We will also use the ﻿.strip() function to remove the new line (‘\n’) character from the end of each row.**[[8]](#footnote-8)

1. **﻿ for** row **in** objFile:
2. lstRow = row.strip().split(',')

**Then I need to reformat the list as a dictionary. I will do this by accessing items in the list using index values and pairing each with the keys we previously used in dicRow: “ID”, “Album Title”, and “Artist”. This was demonstrated in the sample code in LAB 05-B.**[[9]](#footnote-9)

            dicRow = {'ID':lstRow[0], 'CD Title':lstRow[1], 'Artist':lstRow[2]}

**Then we append this dictionary to as an inner data structure to lstTbl using the append()[[10]](#footnote-10) list method to accomplish this.**

**﻿** lstTbl.append(dicRow)

**We finish off by closing the text file with the close()**[[11]](#footnote-11) **file object method.**

**﻿**  objFile.close()

# Adding the functionality of deleting an entry.

1. **elif** strChoice == 'd':
2. # TODO Add functionality of deleting an entry
3. **pass**

First, we will start by displaying the data to the user. The user needs to know what options there are to delete the data. We’ll accomplish this by recycling the code used to display the current data under the elif strChoice == ‘i’:

1. **print**('ID, CD Title, Artist')
2. **for** row **in** lstTbl:
3. **print**(\*row.values(), sep = ', ')

Then we’ll request the user to type an ‘ID’ to choose which entry to delete.

﻿ deleteChoose = int(input('Which ID would you like to delete?:  '))

Then using the method described at 16:00 in the [Zoom Video for Module 5](https://washington.zoom.us/rec/play/78Ylcev8rD83HtLGswSDVPV9W9W7f6us0SRLr_Rfzh63ASYLZAXyZLEXN7RCtu5dLA09uS8mJF8GGqos?continueMode=true)[[12]](#footnote-12), we will delete the dictionary in lstTbl containing the ID selected by the user. This is accomplished as follows:

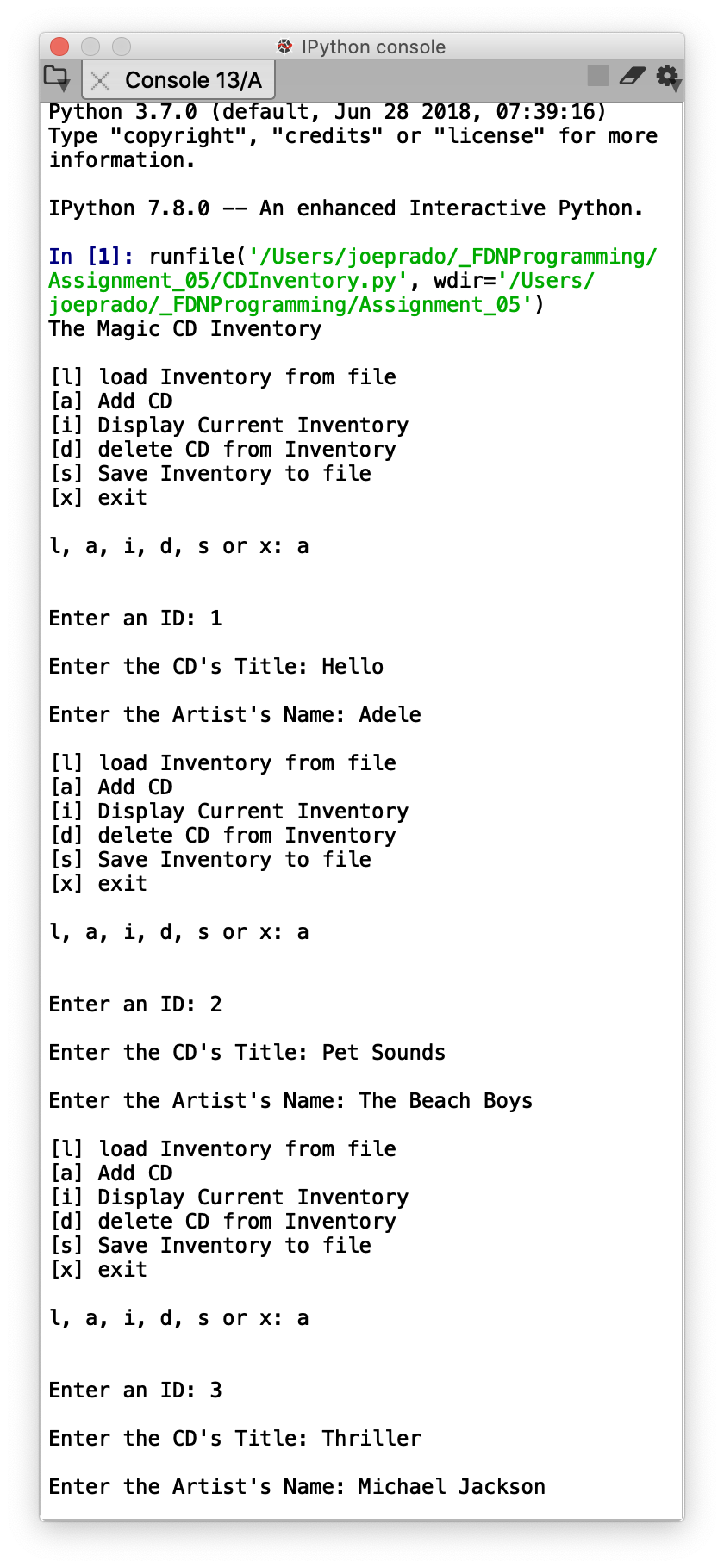
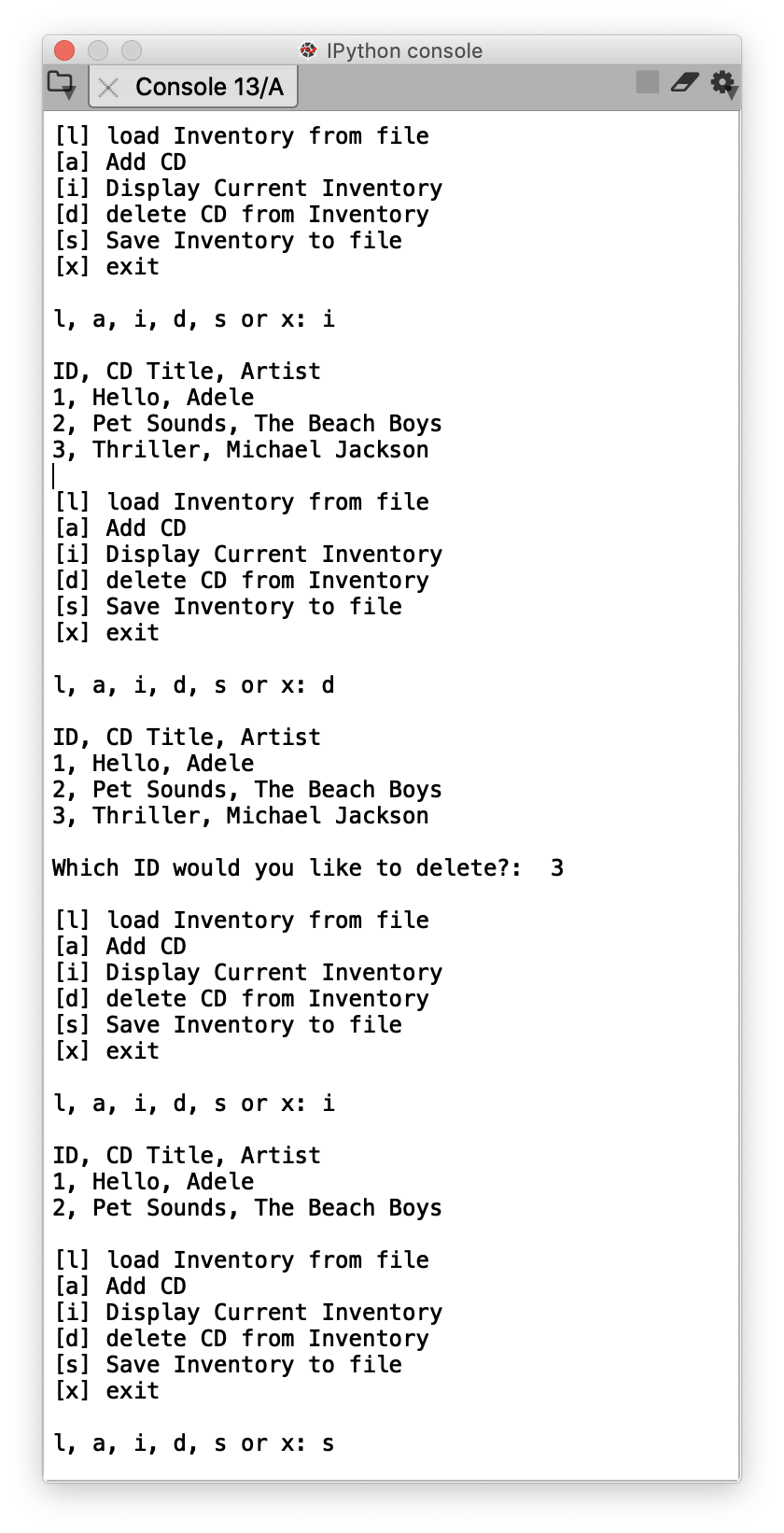
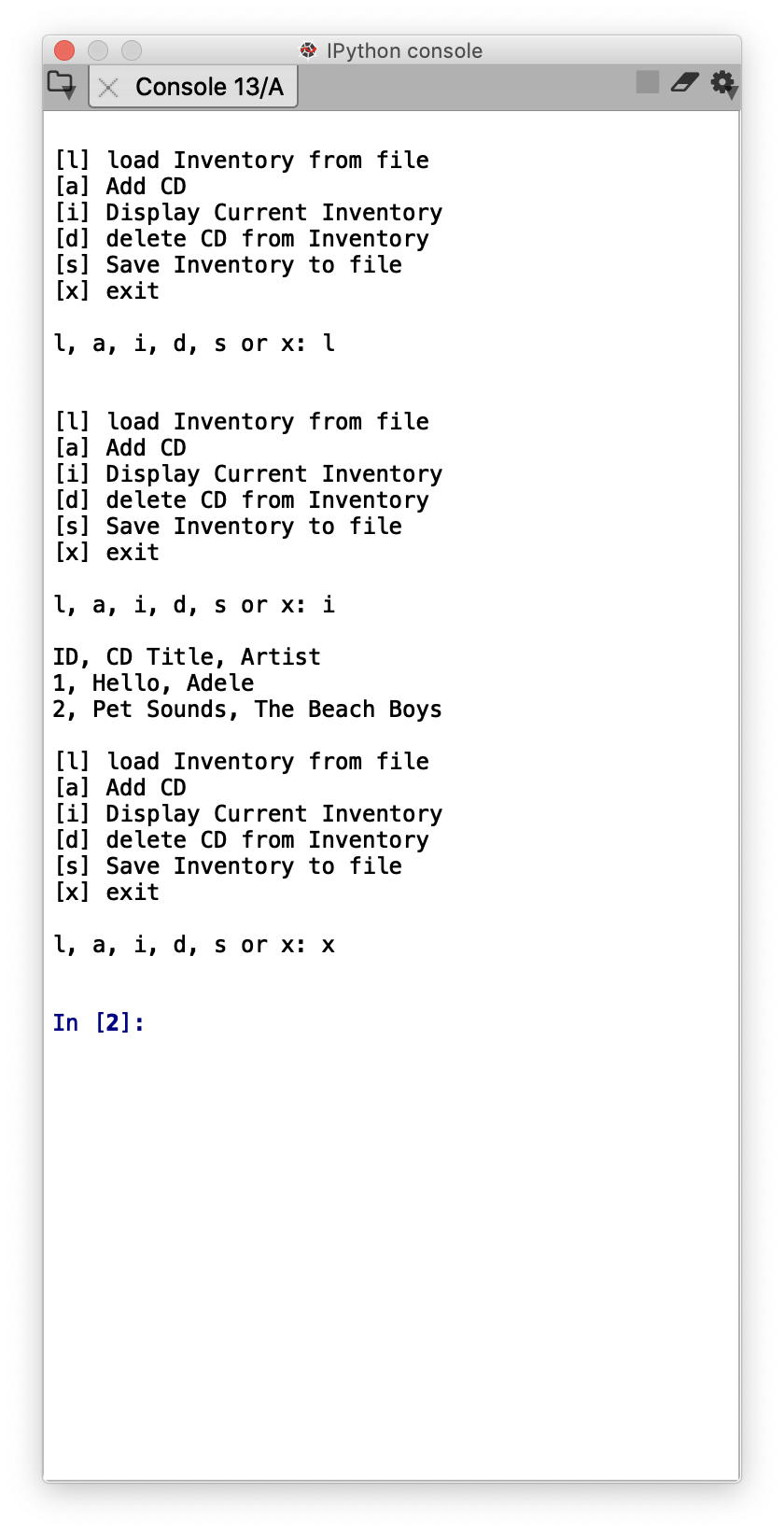
1. **We set a counter variable that will start at ‘0’ when first iterated. That requires us to use rowNbr = -1**
2. **We establish a ‘for’ statement that will interact through the rows in lstTbl**
3. **We advance the counter by one so that rowNbr = 0. This means lstTbl[rowNbr] at this point would correspond to the first dictionary in the lstTbl.**
4. **We set an if statement that becomes true if the user input matches the dictionary value paired with “ID”**
5. **Only if the above is true, do we delete the row indexed by the counter variable.**
6. **Once a row is deleted, the for statement stops with a break statement.**

**The code looks like this:**

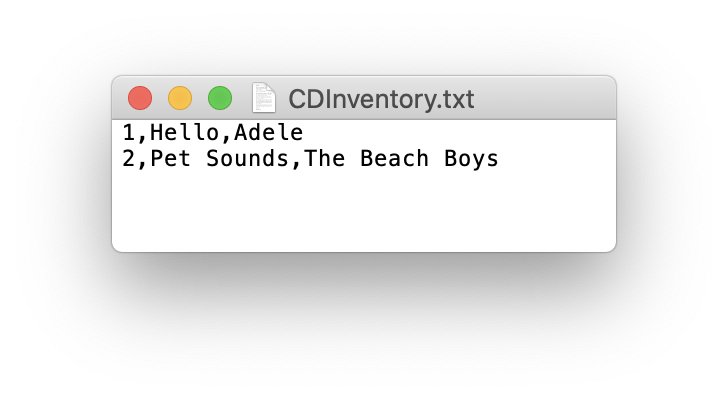
1. rowNbr = -1
2. **for** row **in** lstTbl:
3. rowNbr = rowNbr+1
4. **if** row['ID'] == deleteChoose:
5. **del** lstTbl[rowNbr]
6. **break**

# Running the script

Running the script in Spyder results in the following:

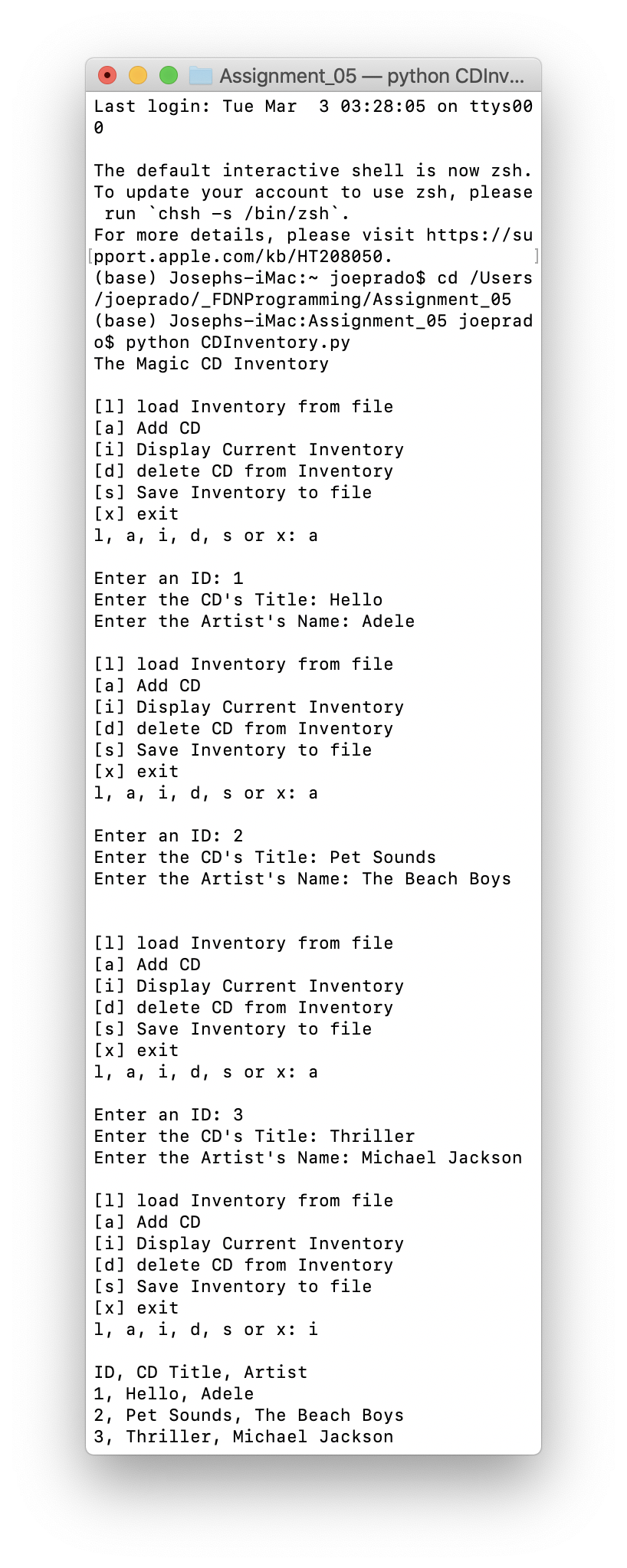
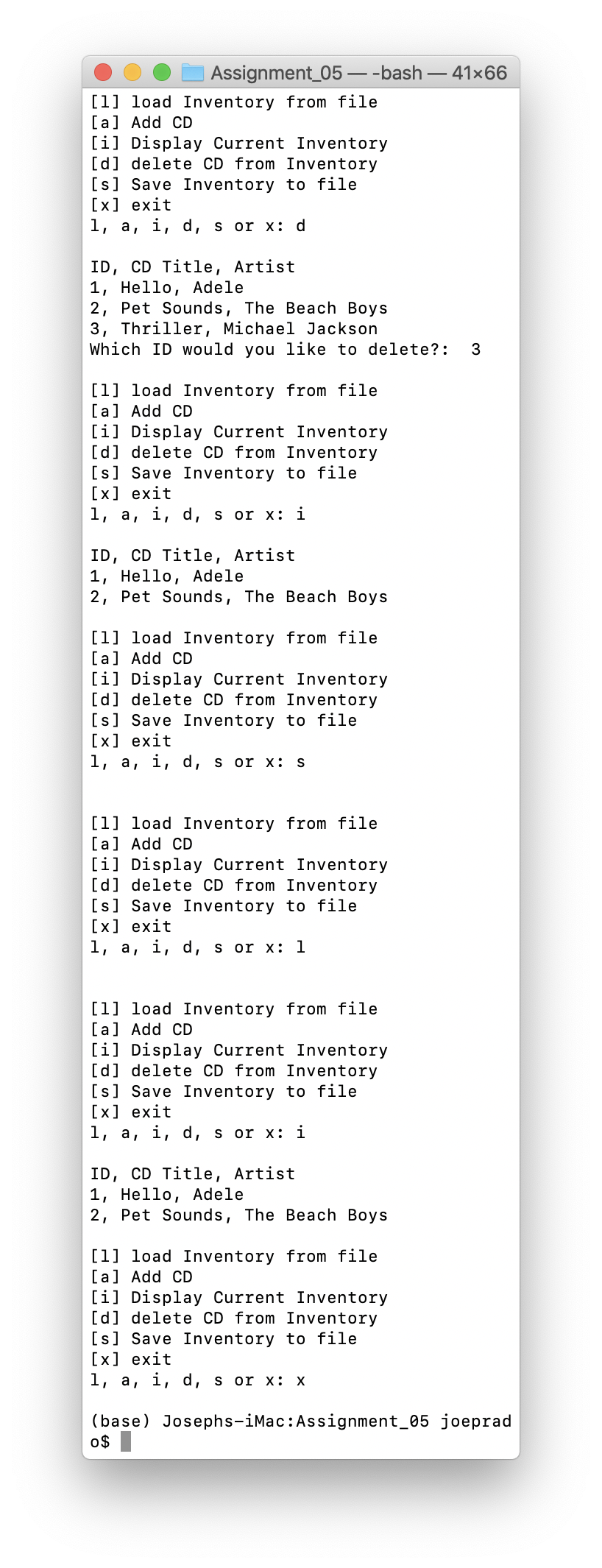
  

*Figure 1 – Output from Spyder*

**

*Figure 2 – CDInventory.txt as written to by Spyder*

Running the script in the Terminal results in the following:

# *Figure 2 – Output from Terminal*

# 

*Figure 3 – CDInventory.txt as written to by Spyder*

# Deficiencies in the script

The script has many limitations. In it very sensitive to order that commands are given to it. I was only able to get it to execute as clean as it did by executing the commands in the order that I did. Certain other execution orders would result in duplicate entries being made to the .txt. file or lists being overwritten before written to the file. I’m sure the logic can be worked to make it clean and more resilient, but I’m not there yet.

Link to GitHub

<https://github.com/joeprado/Assignment_05>

Summary

In this assignment, I took an existing script and:

1. Modified the script to use dictionaries as the inner data structure.
2. Added functionality so that user can load existing data from a text file of CD inventory.
3. Added functionality so that user can delete an entry.
4. Tested the script for functionality in Spyder and the Terminal

# Appendix

## Listing CDInventory.py

1. #------------------------------------------#
2. # Title: CDInventory.py
3. # Desc: Starter Script for Assignment 05
4. # Change Log: (Who, When, What)
5. # DBiesinger, 2030-Jan-01, Created File
6. #joeprado, 2020-Mar-02, Reviewed Script
7. #joeprado, 2020-Mar-02, Replaced inner data with dictionaries
8. #joeprado, 2020-Mar-03, Added functionality of loading existing data
9. #joeprado, 2020-Mar-03, Added functionality to delete an entry
10. #------------------------------------------#
12. # Declare variables
14. strChoice = '' # User input
15. lstTbl = []  # list of lists to hold data
16. dicRow = []  # list of data row
17. strFileName = 'CDInventory.txt'  # data storage file
18. objFile = None  # file object
20. # Get user Input
21. **print**('The Magic CD Inventory\n')
22. **while** True:
23. # 1. Display menu allowing the user to choose:
24. **print**('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
25. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
26. strChoice = input('l, a, i, d, s or x: ').lower()  # convert choice to lower case at time of input
27. **print**()
29. **if** strChoice == 'x':
30. # Exit the program if the user chooses so
31. **break**
32. **if** strChoice == 'l':
33. #Load data from text file
34. lstTbl.clear()  #Clear lstTbl to duplicate entries don't get written to the table.
35. objFile = open(strFileName, 'r') # Open the file in read mode
36. **for** row **in** objFile:  #Read each row from text file
37. lstRow = row.strip().split(',') #Format string in to list with comma as seperator
38. dicRow = {'ID':lstRow[0], 'CD Title':lstRow[1], 'Artist':lstRow[2]} #create keys and values for dictionary
39. lstTbl.append(dicRow) #Append dictionary to list
40. objFile.close() #promptly close text file
41. **elif** strChoice == 'a':  # no elif necessary, as this code is only reached if strChoice is not 'exit'
42. # Add data to the table (2d-list) each time the user wants to add data
43. strID = input('Enter an ID: ') #Request ID from user
44. strTitle = input('Enter the CD\'s Title: ') #Request CD Title from user
45. strArtist = input('Enter the Artist\'s Name: ') # Request Artist Name from user.
46. intID = int(strID) #Convert user ID to integer
47. dicRow = {'ID':intID, 'CD Title':strTitle, 'Artist':strArtist} # Pair keys and values to create dictionary
48. lstTbl.append(dicRow) #Append dictionary to list.
49. **elif** strChoice == 'i':
50. # Display the current data to the user each time the user wants to display the data
51. **print**('ID, CD Title, Artist')
52. **for** row **in** lstTbl: #Unpack dictionaries from list
53. **print**(\*row.values(), sep = ', ') #Unpack items from dictionary; only print values, seperated by comma
54. **elif** strChoice == 'd':
55. #Delete an entry of the user's choosing
56. **print**('ID, CD Title, Artist')
57. **for** row **in** lstTbl:  #Display the current data.
58. **print**(\*row.values(), sep = ', ') #Unpack items from dictionary; only print values, seperated by comma
59. deleteChoose = int(input('Which ID would you like to delete?:  '))#  user select an ID to delete.
60. rowNbr = -1  # set counter variable
61. **for** row **in** lstTbl: #unpack dictionaries from list
62. rowNbr = rowNbr+1 #increase counter variable by 1
63. **if** row['ID'] == deleteChoose: #Find row where value for ID matches row user wants to delete
64. **del** lstTbl[rowNbr] # Delete that row.
65. **break**
66. **elif** strChoice == 's':
67. # Save the data to a text file CDInventory.txt if the user chooses so
68. objFile = open(strFileName, 'a') #Open text file
69. **for** row **in** lstTbl: #Unpack dictionaries from list
70. strRow = ''
71. **for** item **in** row.values(): # Unpack only values from dictionary
72. strRow += str(item) + ',' #Create string of dictionary value seperated items by comma
73. strRow = strRow[:-1] + '\n' # Remove ending comma
74. objFile.write(strRow) #Write string to text file
75. objFile.close() #Close Text File
76. **else**:
77. **print**('Please choose either l, a, i, d, s or x!') #user winds up here if he/she doesn't type in an accepted option

1. Page 140, Python Programming for the absolute beginner, Third Edition. [↑](#footnote-ref-1)
2. Page 142, Python Programming for the absolute beginner, Third Edition. [↑](#footnote-ref-2)
3. Table 5.2, Page 148, Python Programming for the absolute beginner, Third Edition. [↑](#footnote-ref-3)
4. Module 05, Page 2 [↑](#footnote-ref-4)
5. Module 05, Page 21 [↑](#footnote-ref-5)
6. Pages 197-204 (including Table 7.5); Python Programming for the absolute beginner, Third Edition. [↑](#footnote-ref-6)
7. Module 05, Page 04 [↑](#footnote-ref-7)
8. Module 05, Page 04 [↑](#footnote-ref-8)
9. Module 5, Page 22 – Listing 15. [↑](#footnote-ref-9)
10. Table 5.1, Page 132, Python Programming for the absolute beginner, Third Edition. [↑](#footnote-ref-10)
11. Table 7.2. Pag 199, Python Programming for the absolute beginner, Third Edition. [↑](#footnote-ref-11)
12. Accessed 03 March 2020 [↑](#footnote-ref-12)