Troy McFarland

2021-02-14

Foundations of Programming, Python

Assignment\_05

Github link: <https://github.com/DJ-TMC/Assignment_05> (external site)[[1]](#footnote-1)

# Using Lists and Dictionaries to Create a CD Inventory Program

# Introduction

In this paper, I’ll be going over the fifth set of lessons for the Foundations of Programming, Python class taught by Dirk Biesinge and assisted by Douglas Klos. I’ll give a brief overview of things learned in this module, including a brief tangent I took into reading iTunes exported text files, culminating with Assignment 5.

# Overview of New Things Learned in Module 5

Here are the answers to the questions of the main things we learned in this module:

* What is the difference between a Dictionary and a List?
  + Dictionary has pairs of information: a key and a value. The keys are immutable. They can be any type of item that is immutable: A tuple, a string, and integer, etc. In our assignment, we used strings.
  + A list is a group of items, including other lists. Every item in a list is mutable. You can swap out items in a list for other items. Those items may themselves be something immutable, such as a tuple, but you can still remove them from your list.
* What is the difference between an index and a key?
  + Index is a number, starting at 0. If items on a list change, the index number will not necessarily stay the same for remaining items.
  + A key is attached to a specific value. When entering a key, the computer doesn’t have to search through everything as it would in a list.
* How do you read data from a file into a list?
  + Open the file. Using a separator, read in individual entries.
  + For a 2D list, you can use a for loop to grab multiple lines/
* How do you read data from a file into a dictionary?
  + Open the file. Using a separator, read in individual entries, and assign them to keys as they are being read in.
  + If there are multiple entries, then you can use a for loop to search for the keys.
* Why is it making sense to organize data in a 2-dimensional way?
  + Creating a ‘list of lists’ or a ‘list of dictionaries’ etc, is almost like a spreadsheet with multiple cells. It’s a great way to organize multiple entries of similar groups of data. A CD collection database is a great example of this with artist, title, date produced, composers, genre, etc.
* What is the programming pattern “Separation of Concerns”?
  + Grouping similar functions in a program to similar areas whenever possible
    - Three main areas: DATA (declaring variables), PRESENTATION (getting input to user, giving output), and PROCESSING (doing the work of manipulating data based on user input)
* How would you use a function to organize your code?
  + Functions can be put in the Processing section of one’s code, and then called when needed.
* Why is a script template useful?
  + Helps keep coding consistent between programmers on a team. Makes it easier to find things on a larger program.
* Why is error handling (try-except) useful?
  + It helps prevent our data files from getting corrupted, as well as doesn’t kick the user out of the program when they are using it. Also, if enables the programmer to give specific error messages to the user to help them understand why they encountered an error
* What is GitHub and why is it used?
  + A source repository with branching and roll back capabilities.
* What is GitHub’s mascot?
  + Octocat!! I think it is fitting because the arms can represent different branches.

# LAB-05B

I wanted to include Lab-05B in this paper, as I learned a few interesting things

Learning: This code for displaying the Artist & Title works.

1. **elif** strChoice == 'd':
2. # Display data
3. print('Artist, Title')
4. **for** row **in** lstTbl:
5. print(row['Artist'], row['Title'])

Produces something like:

Artist, Title

Sandy Denny, Like an Old Fashoned Waltz

Fairport Convention, UnHalfbricking

Jethro Tull, Aqualung

My original code was:

1. **elif** strChoice == 'd':
2. # Display data
3. print('Artist, Title')
4. **for** row **in** lstTbl:
5. print(dicRow['Artist'], dicRow['Title'])

Which resulted in:

Jethro Tull, Aqualung

Jethro Tull, Aqualung

Jethro Tull, Aqualung

I realized that I wasn’t calling on the actual row info, but rather the last thing I assigned to dicRow.

One of the questions in Lab-05B was:

4. Have a look at this script’s output. Can you think of a way to improve it? Describe what should be done differently and how you would do it.

My Response:

I’d check to see if the text file exists before reading it, to prevent the program from crashing if the user enters an incorrect entry. Not entirely sure how to do this yet.

I’d also have the ability to sort items by title name or artist name. I have a vague idea of creating a separate list to hold the data and then writing the original list over it the new list

And finally, I’d give user feedback when operations take place successfully. This is easily done with print statements, and I integrated this into Assignment 5.

# Importing iTunes Text Files

I took a little tangent and explored how to import other text files. I thought it might be fun to export a playlist text file from iTunes and see if I could read it in Spyder. When I imported it, it totally didn't work in the most spectacular way!



Figure 1: results from reading iTunes text file into Spyder

I found out that this is a different format of text than Spyder was expecting by default. This was a 16 bit text file, and the first characters let me know which type it was.

Resources for understanding text files and byte order:

[Python program will not read a txt file in 16bit characters - Stack Overflow](https://stackoverflow.com/questions/49759183/python-program-will-not-read-a-txt-file-in-16bit-characters) (external site)

More details here show me how to recognize which Encoding is used by which characters

[Byte order mark - Wikipedia](https://en.wikipedia.org/wiki/Byte_order_mark#Byte_order_marks_by_encoding) (external site)

This line fixed my issue:  
objFile = open(strFileName, 'r', encoding='utf-16-le')

This is the code was where I left this tangent exploration

1. strChoice = '' # User input
2. lstTbl = [] # list of lists to hold data
3. lstRow = [] # list of data row
4. strFileName = 'atrax.txt' # data storage file
5. objFile = None # file object
6. objFile = open(strFileName, 'r', encoding='utf-16-le')
7. **for** row **in** objFile:
8. lstRow = row.strip().split('\t')
9. print(lstRow[0:4])
10. objFile.close()
11. print('Notice the \'\\n\' at the end of the last item.\n')

And here is the resulting information. If I revisit it, I will format the results properly.

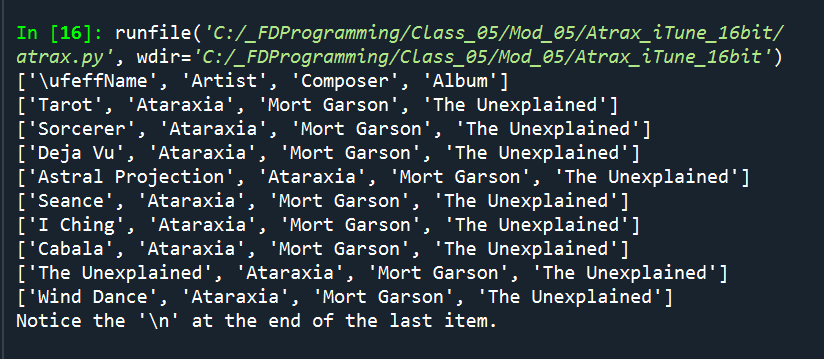


Figure 2: results of printing list rows

# CDInventory.py

Assignment 5 was the first time I really altered anyone else’s code. And, there were a fair number of gotchas in the code that I didn’t immediately see until I started making changes. One thing I need to learn in Spyder is how to change a variable name in all locations with one command (such as find/replace in Word). Keeping track of dictionary entries inside a list format threw me a loop for a while. For deleting rows, on the Saturday office hours, Dirk showed us a way to keep track of which row had the ID number that we wanted to delete. I wanted to come up with a different way to do it. But, my methods were centered around looking up keys in a dictionary! And being that the larger data structure was a list, they were not working. I want to explore solving that same problem by making a dictionary of dictionaries. Not sure if it will work well or not, but I think I could find values faster.

Below are my results in Spyder and the Terminal window. I split to make more legible

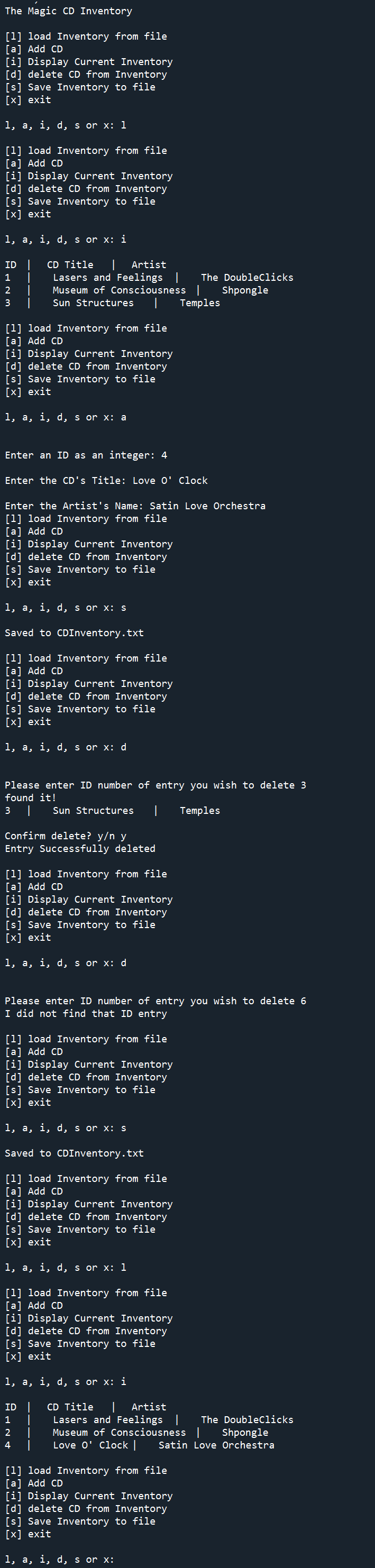
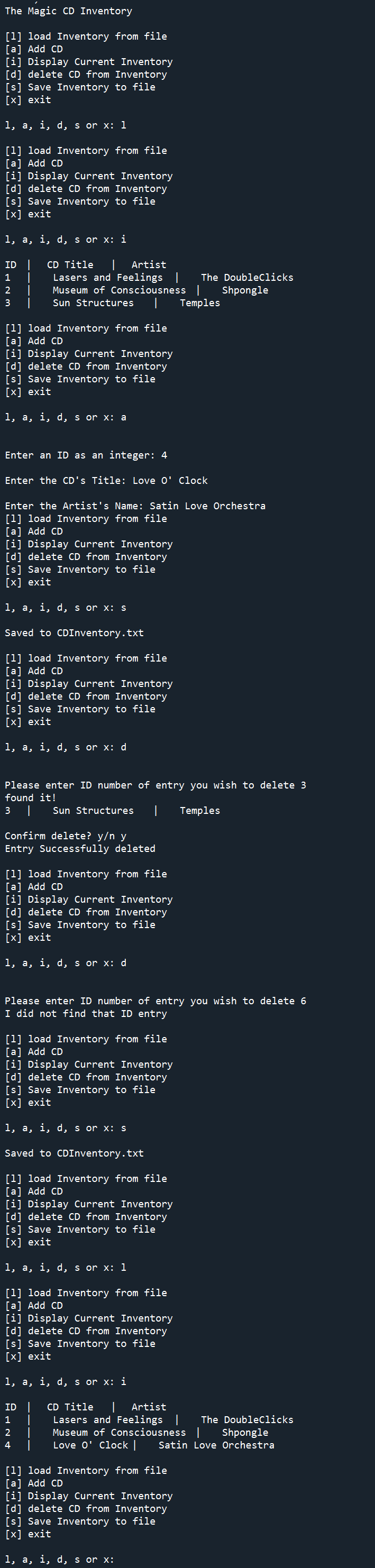
 

Figure 3: CDInventory.py in Spyder

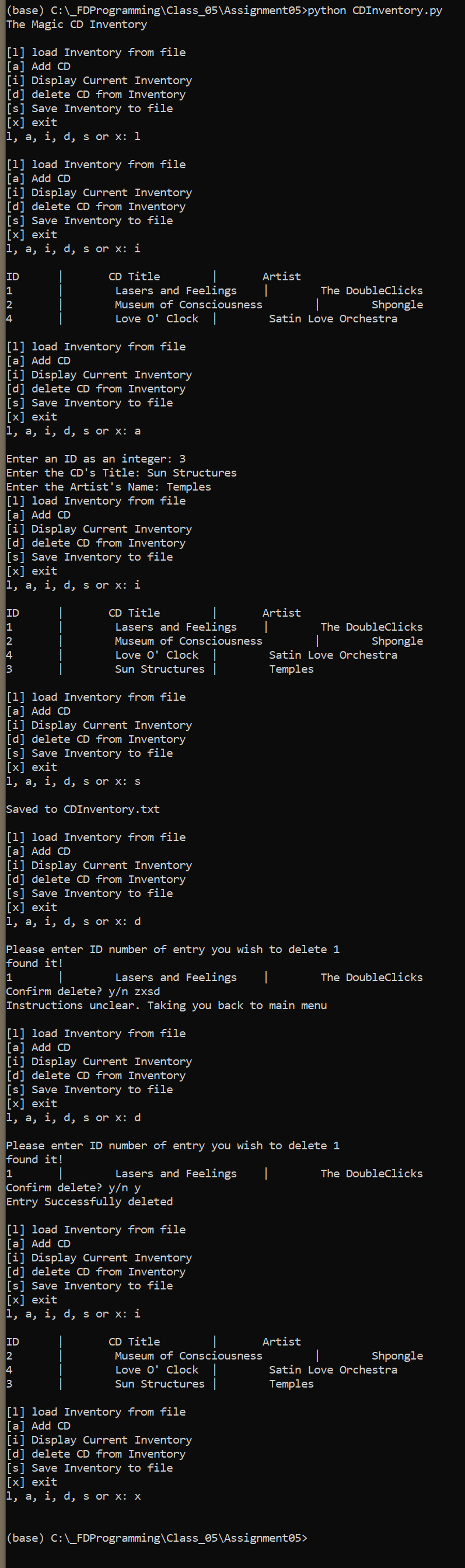
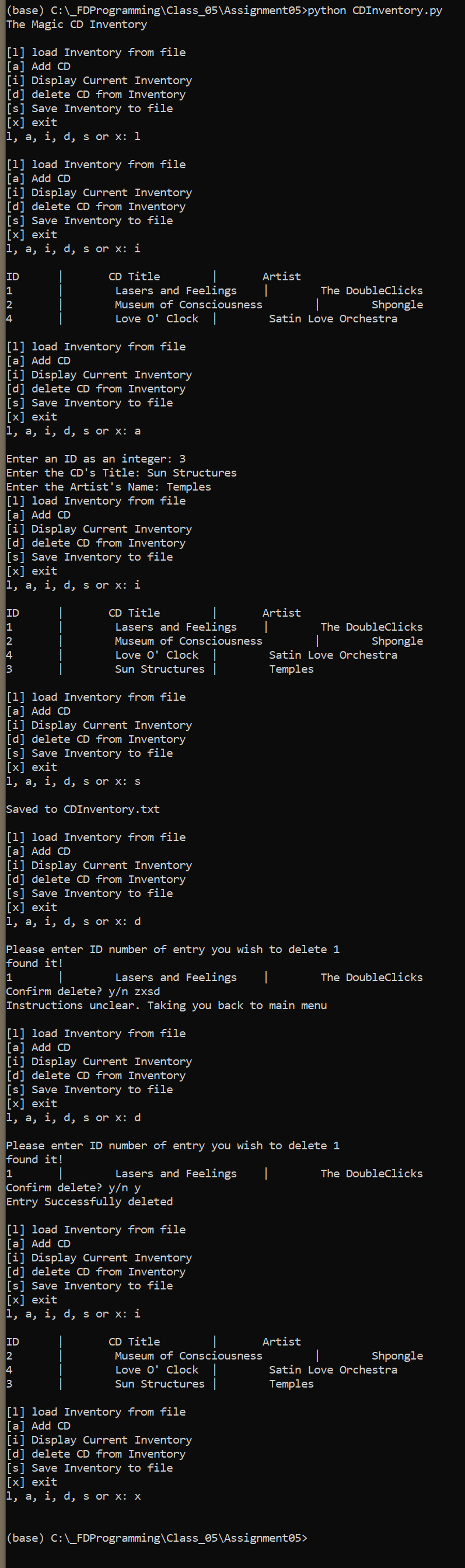
 

Figure 4: CDInventory.py in Spyder

# Summary

In this paper, I went over the fifth set of lessons for the Foundations of Programming. I gave a brief overview of things learned in this module as well as some of my own explorations.

# Appendix

Full Code of Lab05\_B.py:

1. #------------------------------------------#
2. # Title: Lab05\_B.py
3. # Desc: Lab05-B: convert Lab05-A to use dictionaries
4. # Change Log: (Who, When, What)
5. # TMcFarland. 2021-Feb-13, Row converted to Dicitonary solution
6. # TMcFarland, 2021-Feb-10, Added functionality
7. # DBiesinger, 2030-Jan-01, Created File
8. #------------------------------------------#
9. # Declare variables
10. strChoice = '' # User input
11. lstTbl = [] # list of lists to hold data
12. dicRow = {} # dictionary of data row
13. strFileName = 'CDInventory.txt' # data storage file
14. objFile = None # file object
15. # Get user Input
16. print('Write or Read file data.')
17. **while** **True**:
18. print('\n[a] add data to list\n[w] to write data to file\n[r] to read data from file')
19. print('[d] display data\n[exit] to quit')
20. strChoice = input('a, w, r, d, or exit: ').lower() # convert choice to lower case at time of input
21. print('\n\n')
22. **if** strChoice == 'exit':
23. **break**
24. **if** strChoice == 'a': # no elif necessary, as this code is only reached if strChoice is not 'exit'
25. # Add data to list in memory
26. strArtist = input('Enter Artist Name: ')
27. print(strArtist)
28. strTitle = input('Enter CD Title: ')
29. print(strTitle)
30. dicRow = {'Artist': strArtist, 'Title': strTitle}
31. print(dicRow)
32. lstTbl.append(dicRow)
33. print(lstTbl)
34. print('Added to memory')
35. **elif** strChoice == 'w':
36. # Write list to File
37. objFile = open(strFileName, 'w')
38. **for** row **in** lstTbl:
39. strRow = ''
40. **for** item **in** lstTbl:
41. strRow += str(item) + ','
42. strRow = strRow[:-1] + '\n'
43. objFile.write(strRow)
44. objFile.close()
45. **elif** strChoice == 'r':
46. # File to print
47. lstTbl.clear()
48. objFile = open(strFileName, 'r')
49. **for** row **in** objFile:
50. dicRow = row.strip().split(',')
51. lstTbl.append(dicRow)
52. objFile.close()
53. **elif** strChoice == 'd':
54. # Display data
55. print('Artist, Title')
56. **for** row **in** lstTbl:
57. print(row['Artist'], row['Title'])
58. **else**:
59. print('Please choose either a, w, r or exit!')

Full code of Assignment 05’s 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. # TMcFarland, 2021\_Feb\_14, Mod to use Dictionaries, added load inventory & delete inventory functionality
7. #------------------------------------------#
8. # Declare variables
9. #---------------------------DATA------------------------------#
10. strChoice = '' # User input
11. lstTbl = [] # list of dicts to hold data
12. dctRow = {} # dict of data row
13. strFileName = 'CDInventory.txt' # data storage file
14. objFile = None # file object
15. # Get user Input
16. print('The Magic CD Inventory\n')
17. **while** **True**:
18. # 1. Display menu allowing the user to choose:
19. print('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
20. print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
21. #--------------PRESENTATION (Input/Output)---------------#
22. strChoice = input('l, a, i, d, s or x: ').lower() # convert choice to lower case at time of input
23. print()
24. #----------------------PROCESSING------------------------#
25. **if** strChoice == 'x':
26. # 5. Exit the program if the user chooses so
27. **break**
28. **if** strChoice == 'l':
29. # LOAD Add the functionality of loading existing data
30. lstTbl.clear() #clears list. Found this functionality in Lab05\_0B example
31. objFile = open(strFileName, 'r')
32. **for** row **in** objFile:
33. lstRow = row.strip().split(',')
34. dctRow = {'ID': int(lstRow[0]), 'Title': lstRow[1], 'Artist': lstRow[2]} #added ID, add indeces
35. lstTbl.append(dctRow)
36. objFile.close()
37. **pass**
38. **elif** strChoice == 'a':
39. # 2. ADD DATA to the table (2d-dictionary) each time the user wants to add data
40. strID = input('Enter an ID as an integer: ')
41. strTitle = input('Enter the CD\'s Title: ')
42. strArtist = input('Enter the Artist\'s Name: ')
43. intID = int(strID) # turns input into integer.
44. dctRow = {'ID':intID, 'Title':strTitle, 'Artist':strArtist}
45. lstTbl.append(dctRow)
46. **elif** strChoice == 'i':
47. # 3. DISPLAY the current data to the user each time the user wants to display the data
48. print('ID\t|\tCD Title \t|\tArtist')
49. **for** row **in** lstTbl:
50. print(\*row.values(), sep = '\t|\t ')
51. print()
52. **elif** strChoice == 'd':
53. #User enter ID # or CD Title
54. delChoice = int(input('Please enter ID number of entry you wish to delete ').strip())
55. rowIndex = -1 #starting at -1 so will = 0 when counter below kicks in
56. **for** row **in** lstTbl:
57. rowIndex += 1
58. **if** row['ID'] == delChoice:
59. print('found it!')
60. print(\*row.values(), sep = '\t|\t ')
61. delConf = input('Confirm delete? y/n ')
62. **if** delConf == 'y':
63. **del** lstTbl[rowIndex]
64. print('Entry Successfully deleted\n')
65. **break**
66. **elif** delConf == 'n':
67. print('Deletion Canceled. Taking you back to Main menu\n')
68. **break**
69. **else**:
70. print('Instructions unclear. Taking you back to main menu\n')
71. **break**
72. **else**:
73. **if** (rowIndex + 1) == len(lstTbl): #compare entry to length of 2d dicitonary. Adding 1 back to index to match total length
74. print('I did not find that ID entry\n')
75. **pass**
76. **elif** strChoice == 's':
77. # 4. Save the data to a text file CDInventory.txt if the user chooses so
78. objFile = open(strFileName, 'w') #changed to w to overwrite any entry to prevent duplicates
79. **for** row **in** lstTbl:
80. strRow = ''
81. **for** item **in** row.values():
82. strRow += str(item) + ','
83. strRow = strRow[:-1] + '\n'
84. objFile.write(strRow)
85. objFile.close()
86. print('Saved to CDInventory.txt\n')
87. **else**:
88. print('Please choose either l, a, i, d, s or x!')

1. Accessed 2021-02-15 [↑](#footnote-ref-1)