Brandon Wayward

2020-Aug-11

IT FDN 110 B SU 20

Assignment 05

Continued CDInventory.py

Introduction

In this lesson we strengthened our knowledge of lists and added dicts to that. We also expanded our knowledge on saving to and loading data from files as well as separation of concerns, and a beginner’s entry into Github Finally, we save the best for last. We learned how to delete elements of a dictionary based on user input.

# Lists and Dictionaries

From the previous module we learned that lists were a data structure in python that could be used to hold data with comma separated values inside brackets “[]”. Dictionaries[[1]](#footnote-1), unlike lists or tuples, allow you to store data in key:value pairs and are denoted with curly brackets “{}”. This allows quite a bit more fluidity in your data storage as you can look data up by it’s keys as opposed to digging through an entire sequence. We utilized dictionaries in our code to store CD data based on an ID, Artist, and Title

# 

Figure - Defining Dictionaries

# Loading Data from Files

In module 4 we referenced reading data from memory. In this module we take that a step further and consider loading the data from a file into memory to be worked with / manipulated. Reading from a file and writing it line by line into the dict was a bit strange because I hadn’t considered elements and indexing properly. Once I figured out how to push the data from the file to specific keys to build key:value pairs in the dict everything flowed as expected.[[2]](#footnote-2)

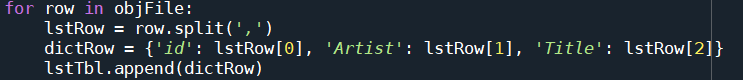


Figure - Loading to Dictionary

# Delete elements from a list of dictionaries based on user input

Deleting elements from a collection of data is good information to know, especially when it comes to needed to remove data based on user input from a data source, such as a list of dictionaries. I had quite a few issues with this part of the assignment. I tried a variety of ways to get the code to function as expected, starting with a code that would delete the item with “user input -1”… this works great for the first item in the list, but a datapoint with ID4 could easily be in position 0, and as such it wouldn’t be deleted since the code was looking for dict[3].

Next attempt was to review the module documentation and try .pop, as that also appeared to be a potential solution. This had it’s own little bundle of surprises for me, as when I attempted to utilize this method it would only delete the exact key:value of the user input. IE: it would delete the ID key, but leave the artist and title keys. I think this method can be used the way I’m envisioning, but I just couldn’t grasp it.

Finally, it hit me…I was treating the list of dictionaries as if I needed to attack each dictionary individually, and not as a complete element within the list. This accidentally revealed itself to me while I was trying to jog my brain. I ran a len(lstTbl) and it returned 6… the amount of test CD’s I had written to the file. Each dictionary was in and of itself one element of the list, and as such I could then utilize user input to delete the dictionary in it’s entirety as a list element. Using range[[3]](#footnote-3)(len(lstTbl)) we could get exactly where we wanted. Pop did what I wanted, how I wanted! Glory was achieved on this day.

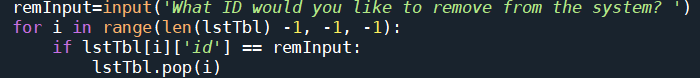


Figure - Deleting from list

# Summary

In summary, with each lesson I grow fonder and fonder of this language. In the beginning it felt like I was just throwing words at a wall and seeing what stuck, in module 4 I had the life drained from me, and now with module 5 I can feel the growth. Certain tasks are still difficult, but I am building a toolset to work around that difficulty and resolve issues outside of praying to the patron saint of code for an epiphany. Working with dictionaries really broadens your ability to store and manipulate data within your program. There are a variety of data removal methods at our disposal. Separation of Concerns helps you plan out your coding into chunks that are easily addressable. Github, well I created an account and pushed my code, but I still don’t understand / recognize how powerful this tool can be,

# Appendix

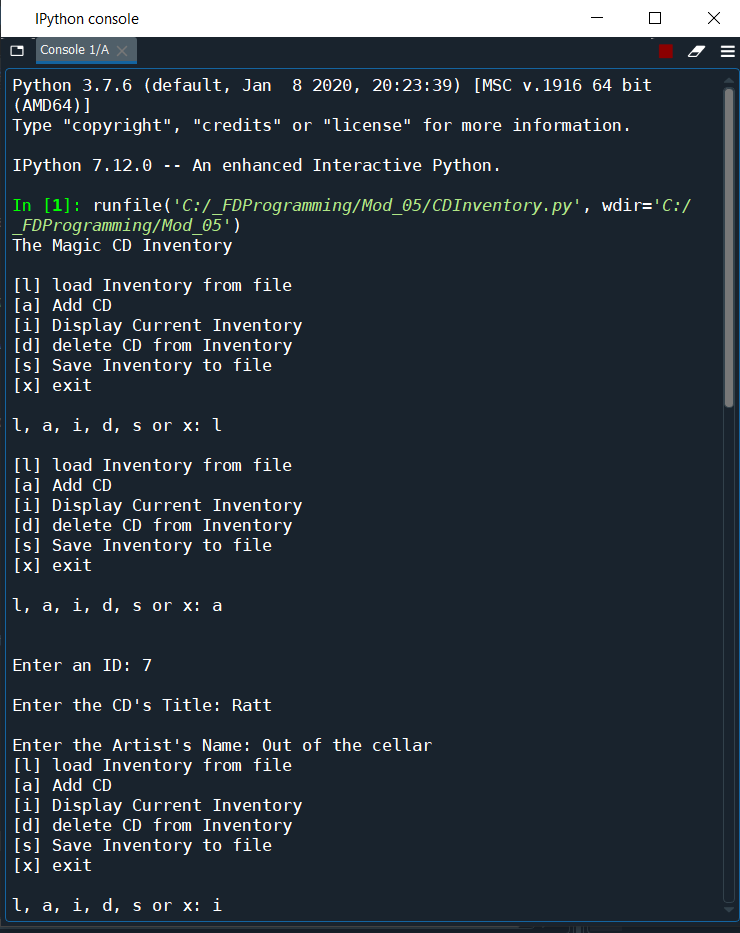


Figure - Spyder Load and Add

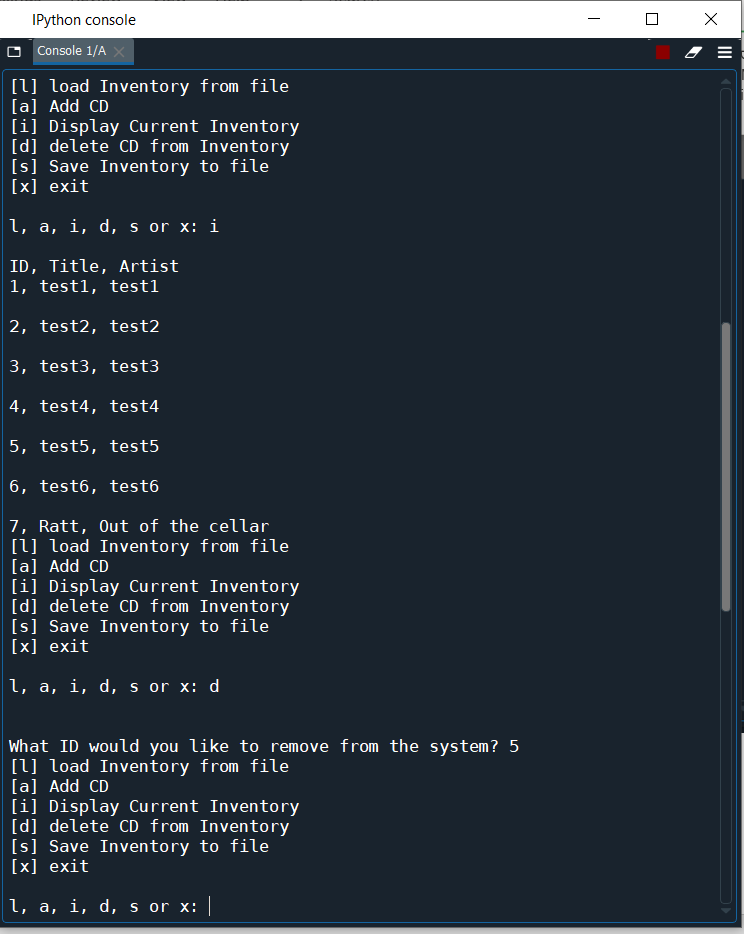


Figure - Spyder display / delete

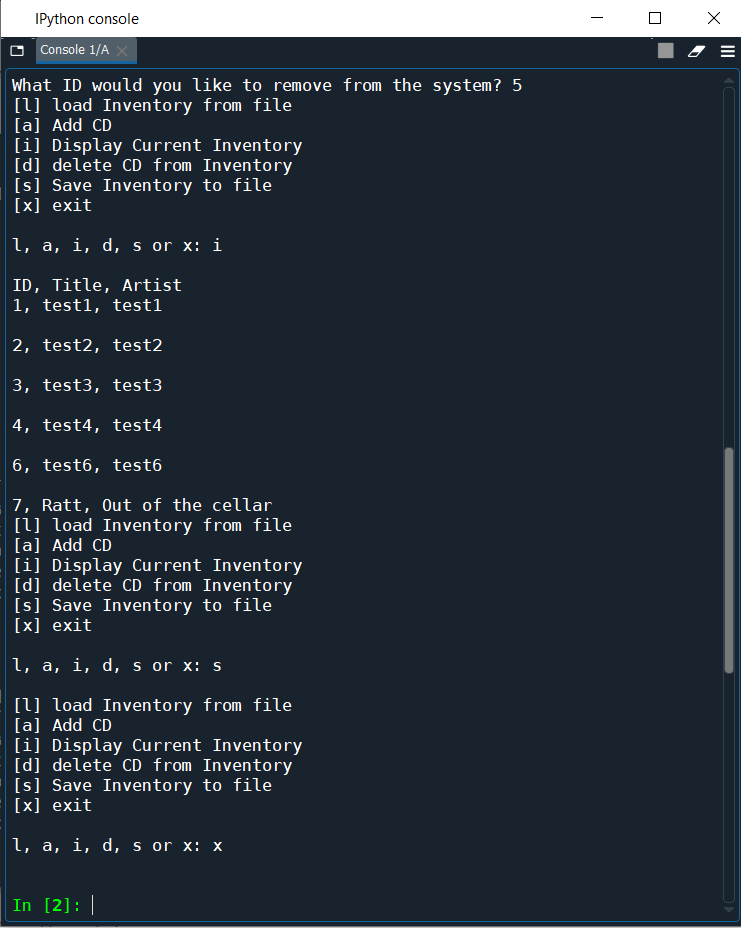


Figure - Spyder display (after delete) save exit

1. #------------------------------------------#
2. # Title: CDInventory.py
3. # Desc: Starter Script for Assignment 05
4. # Change Log: (Who, When, What)
5. # BWayward, 2020-Aug-07, Created File
6. # BWayward, 2020-Aug-07, Added Load Data functionality
7. # Bwayward, 2020-Aug-09, Added and corrected delete functionality
8. # BWayward, 2020-Aug-09, Modified delete functionality to work correctly
9. # with pop instead of delete, as it wouldn't delete every line, just 1st
10. # BWayward, 2020-Aug-09 Cleaned up scraps, removed debug print statements, published
11. # to github
12. #------------------------------------------#
14. # Declare variables
16. dictRow = {}
17. strChoice = '' #User Input
18. lstTbl = [] # List of lists to hold data
19. lstRow = [] # List of data row
20. strFileName = 'CDInventory.txt' #Data Storage File
21. objFile = None #File Object
23. # Get user Input
24. **print**('The Magic CD Inventory\n')
25. **while** True:
26. # 1. Display menu allowing the user to choose:
27. **print**('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
28. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
29. strChoice = input('l, a, i, d, s or x: ').lower()  # convert choice to lower case at time of input
30. **print**()
32. **if** strChoice == 'x':
33. # 5. Exit the program if the user chooses so
34. **break**
35. **if** strChoice == 'l':
36. # TODO Add the functionality of loading existing data
37. lstTbl.clear()
38. objFile = open(strFileName, 'r')
39. **for** row **in** objFile:
40. lstRow = row.split(',')
41. dictRow = {'id': lstRow[0], 'Artist': lstRow[1], 'Title': lstRow[2]}
42. lstTbl.append(dictRow)
43. objFile.close()
44. **elif** strChoice == 'a':  # no elif necessary, as this code is only reached if strChoice is not 'exit'
45. # 2. Add data to the table (2d-list) each time the user wants to add data
46. strID = input('Enter an ID: ')
47. strTitle = input('Enter the CD\'s Title: ')
48. strArtist = input('Enter the Artist\'s Name: ')
49. intID = int(strID)
50. dictRow = {'id': intID, 'Title': strTitle, 'Artist': strArtist, }
51. lstTbl.append(dictRow)
52. **elif** strChoice == 'i':
53. # 3. Display the current data to the user each time the user wants to display the data
54. # Crashes if pressed before l for load. not sure how to check for
55. # validity of data prior to execution
56. **print**('ID, Title, Artist')
57. **for** row **in** lstTbl:
58. **print**(\*row.values(), sep = ', ')
59. **elif** strChoice == 'd':
60. remInput=input('What ID would you like to remove from the system? ')
61. **for** i **in** range(len(lstTbl) -1, -1, -1):
62. **if** lstTbl[i]['id'] == remInput:
63. lstTbl.pop(i)
64. **elif** strChoice == 's':
65. # 4. Save the data to a text file CDInventory.txt if the user chooses so
66. objFile = open(strFileName, 'a')
67. **for** row **in** lstTbl:
68. strRow = ''
69. **for** items **in** row.values():
70. strRow += str(items) + ','
71. strRow = strRow[:-1] + '\n'
72. objFile.write(strRow)
73. objFile.close()
74. **else**:
75. **print**('Please choose either l, a, i, d, s or x!')

1. <https://docs.python.org/3/library/stdtypes.html#mapping-types-dict> Accessed 8/6/2020 [↑](#footnote-ref-1)
2. <https://www.geeksforgeeks.org/read-a-file-line-by-line-in-python/Accessed> 8/7/2020 [↑](#footnote-ref-2)
3. <https://www.geeksforgeeks.org/python-range-function/> Accessed 8/8/2020 [↑](#footnote-ref-3)