Rogger Tovar

11/14/2021

Foundations of Python

Assignment 05

CD Inventory Code Continued

# Introduction

This week’s project includes adding to a similar inventory list just as assignment 4. The difference here is adding our new learned Dictionary Value. The interesting part of the assignment is not building code from scratch but to adding/configuring an already existing Python file.

# CD Inventory

This assignment went a little smoother but went through many trials and errors. In my own error when testing out the project is, I cannot remember what I attempts before getting it right. The conclusion to this idea, the attempts of so many combinations of what I need the syntax to do, that by the time I find the correct method, I just moved on to the next to “elif” statement. To start the assignment I changed the “lstRow” to a dict type. I should have changed the name to “dictRow,” since I hard time determining which variable was typed as a dictionary, figure 1.

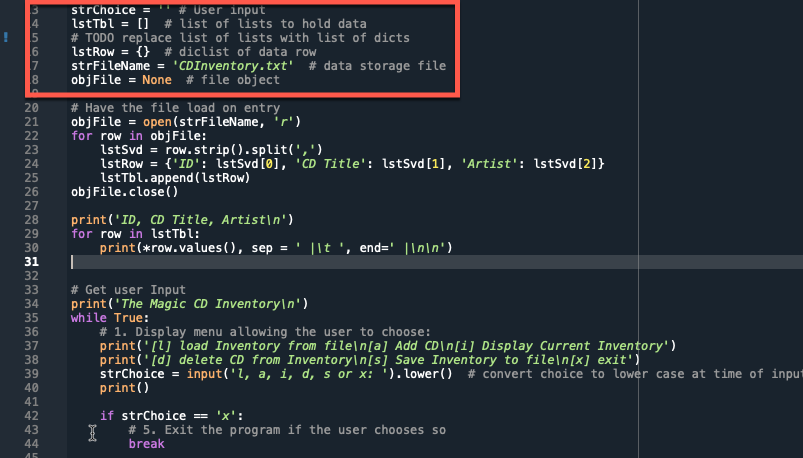


Figure 1 – line 6 change type to dictionary.

Figure 2 displays the menu along with an inventory being loaded into the program. This helps the user already understand if there are any items in the inventory to then either add or delete inventory items.

Text

Description automatically generated

Figure 2 - "MENU"

The real work and head crunching came in the elif statements. Though the statements were straight forward, the matter of having it all working really took the time. Like I said before, I spent extra time trying to understand which variable had the dictionary list. Adding variables went surprisingly well, regarding time, but saving, deleting, and importing took a bit of time. Displaying the inventory was not too bad, I used examples from the reading and combined it with web based reading. I then, set a reminder to add and print the items to the beginning of the code, (figure 3) so when the user boots up the program, the items of a saved text file will indeed print, as referenced in Figure 2.

Text

Description automatically generated

Figure 3 – Print items already in saved file.

Continuing along the code, I reviewed the menu list just to make sure there wasn’t any funny business and double checked the referenced string when to select it, was available in the code and matched with an elif. Starting with the easiest elif state, was in fact easy, which is the exit option. Not much to do, I did add a print function in the end, but towards the end, removed it. I remember the assignment said, “no functions.”

The parts I struggled was the same last week for this week, adding and saving data, but this time as dictionary items. Having read chapter 5, along with the all the suggested reading, made these entries easier. I will quickly include images of an added entry in the run example of the modified code, then displaying all inventory, at the current moment, then saving the file my folder.

Text

Description automatically generated

Figure 4 – Selecting add, adding Sinatra to the inventory.

Text

Description automatically generated

Figure 5 displaying current inventory.

Text

Description automatically generated

Figure 6 - Saving file

Graphical user interface, application, Word

Description automatically generated

Figure 7 - file created content

I kept the formatting in Figure 7 to default from the file, if not, I would have been here another 12 hours trying to figure this out.

The final curveball I will talk about is loading the file; I can successfully load the file. What threw me off was finding out about pre-loading the inventory six hours before I am to submit this project from a conversation in the forum. Just by that alone, I knew another while/else statement would be needed to be added to the top of the file where the file opens and loads the inventory list. The reason for this, when the user opens a file, if the file has yet to be created, the program will crash and not progress to display the menu list. Since I already had been testing the created file, I did not have issues re-opening and run the program. In figure 8, my recent added “Frank Sinatra” items have been loaded from previous saved file, just as the program is run.

Text

Description automatically generated

Figure 8 - Pre-loading the inventory list

# Summary

This assignment was tough by working with an already created script, which felt like a real-world scenario. I can totally envision myself collaborating with other teams at work on a file just like this. This has been a great challenge with a good amount of textbook and web reading in order to complete the code.

# Appendix

1. Listing CDInventory.py
2. *#------------------------------------------#*
3. *# Title: CDInventory.py*
4. *# Desc: Script to create a txt file with user entered data.*
5. *# Change log [Who, When, What]*
6. *# rtovar, 2021-Nov-13, created file*
7. *# rtovar, 2021-Nov-14, Changed lstRow to a dict*
8. *# rtovar, 2021-Nov-14, added code to 'x', 'a', 's'*
9. *# rtovar, 2021-Nov-14, add "press anything to finish”*
10. *#------------------------------------------#*
12. *# Declare variabls*
14. strChoice = '' *# User input*
15. lstTbl = [] *# list of lists to hold data*
16. *# TODO replace list of lists with list of dicts*
17. lstRow = {} *# diclist of data row*
18. strFileName = 'CDInventory.txt' *# data storage file*
19. objFile = None *# file object*
21. *# Have the file load on entry*
22. objFile = open(strFileName, 'r')
23. **for** row **in** objFile:
24. lstSvd = row.strip().split(',')
25. lstRow = {'ID': lstSvd[0], 'CD Title': lstSvd[1], 'Artist': lstSvd[2]}
26. lstTbl.append(lstRow)
27. objFile.close()
29. **print**('ID, CD Title, Artist**\n**')
30. **for** row **in** lstTbl:
31. **print**(\*row.values(), sep = ' |**\t** ', end=' |**\n\n**')

34. *# Get user Input*
35. **print**('The Magic CD Inventory**\n**')
36. **while** True:
37. *# 1. Display menu allowing the user to choose:*
38. **print**('[l] load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')
39. **print**('[d] delete CD from Inventory**\n**[s] Save Inventory to file**\n**[x] exit')
40. strChoice = input('l, a, i, d, s or x: ').lower() *# convert choice to lower case at time of input*
41. **print**()
43. **if** strChoice == 'x':
44. *# 5. Exit the program if the user chooses so*
45. **break**
47. **if** strChoice == 'l':
48. *# TODO Add the functionality of loading existing data*
49. objFile = open(strFileName, 'r')
50. **for** row **in** objFile:
51. lstSvd = row.strip().split(',')
52. lstRow = {'ID': lstSvd[0], 'CD Title': lstSvd[1], 'Artist': lstSvd[2]}
53. lstTbl.append(lstRow)
54. objFile.close()
55. **pass**
57. **elif** strChoice == 'a': *# no elif necessary, as this code is only reached if strChoice is not 'exit'*
58. *# 2. Add data to the table (2d-list) each time the user wants to add data*
59. strID = input('Enter an ID: ')
60. strTitle = input('Enter the CD**\'**s Title: ')
61. strArtist = input('Enter the Artist**\'**s Name: ')
62. intID = int(strID)
63. lstRow = {'ID': intID, 'CD Title': strTitle, 'Artist': strArtist}
64. lstTbl.append(lstRow)
66. **elif** strChoice == 'i':
67. *# 3. Display the current data to the user each time the user wants to display the data*
68. **print**('ID, CD Title, Artist**\n**')
69. **for** row **in** lstTbl:
70. **print**(\*row.values(), sep = ' |**\t** ', end=' |**\n\n**')
72. **elif** strChoice == 'd':
73. *# TODO Add functionality of deleting an entry*
74. delID = int(input('Enter an ID to Delete: '))
75. **for** row **in** lstTbl:
76. **if** delID != row['ID']:
77. lstTbl.remove(row)
78. **print**(row, 'deleted')
79. **else**:
80. delID == row['ID']
81. **print**('nope')
82. **pass**
84. **elif** strChoice == 's':
85. *# 4. Save the data to a text file CDInventory.txt if the user chooses so*
86. objFile = open(strFileName, 'w')
87. **for** row **in** lstTbl:
88. **print**(row)
89. strRow = ''
90. **for** item **in** row.values():
91. strRow += str(item) + ','
92. strRow = strRow[:-1] + '**\n**'
93. objFile.write(strRow)
94. objFile.close()
95. **print**('Your information has been saved! **\n**')
96. input('Press any key to continue: **\n**')
97. **else**:
98. **print**('**\n**Please choose either l, a, i, d, s or x!**\n**')