Nick Culbert

2021 Feb 14

IT FDN 110 A Wi 21

Assignment 05

CD Inventory Version 2

# Introduction

I modified an existing Python script to add the functionalities of inputting data from a file and deleting user entries. I also wrote my script so that when a user loads data from the file but then chooses to delete an entry which was previously made, that entry is deleted not only from the current list of entries but also from the file itself.

# Data

In addition to adding a dictionary to nest into the 2-D list, I also added the variables oldLstTbl and newLstTbl to separate the data loaded from a pre-existing file into working memory and the data inputted by the user but not yet saved to the file (Figure 1). I included this element after an initial round of testing, where I found that if I ran the script, loaded existing data, and then saved, the existing data would be duplicated. By separating the data this way, I could display the entire contents of the ‘working memory’ while only actually saving whatever data in the ‘working memory’ had not yet been written to the file.

1. strChoice = '' # User input
2. oldLstTbl = []
3. newLstTbl = []
4. lstTbl = []  # list of lists to hold data
5. lstRow = []  # list of data row
6. dictRow = {} # dict of list row
7. strDelChoice = ''
8. strFileName = 'CDInventory.txt'  # data storage file
9. objFile = None  # file object

Figure 1 - Global variables

I also added two lines which create the file if it does not yet exist (Figure 2).

1. objFile = open(strFileName, 'a')
2. objFile.close()

Figure 2 - Creating the file

# Processing

## Loading existing data and adding data

If the user chooses to load existing data, they will be creating a 2-D list consisting of dictionaries that define each row of pre-existing data (Figure 3). I needed to use the strip() method to undo my personal aesthetic formatting, which was to add ‘ | ’ as a separator between cells in the table.

1. **if** strChoice == 'l':
2. objFile = open(strFileName, 'r')
3. **for** row **in** objFile:
4. lstRow = row.split('|')
5. dictRow = {'ID':int(lstRow[0].strip()), 'Title':lstRow[1].strip(), 'Artist':lstRow[2].strip()}
6. oldLstTbl.append(dictRow)
7. objFile.close()
8. **print**('Data loaded!\n')

Figure 3 - Loading existing data

If the user chooses to add data, they will be creating a separate 2-D list which adds another dictionary each time the user adds a CD (Figure 4).

1. **elif** strChoice == 'a':
2. strID = input('Enter an ID: ')
3. strTitle = input('Enter the CD\'s Title: ')
4. strArtist = input('Enter the Artist\'s Name: ')
5. intID = int(strID)
6. dictRow = {'ID':intID, 'Title':strTitle, 'Artist':strArtist}
7. newLstTbl.append(dictRow)
8. **print**('\nData added!\n')

Figure 4 - Adding data

## Displaying current data

When the user chooses this option, the existing and new data are combined into a single 2-D list to be displayed via a for loop.

Since the data in ‘working memory’ does not necessarily represent the data that is in the file (if it exists previously) and since the ‘working memory’ needs to be cleared after being saved to the file (to avoid duplication), I termed it the ‘queue’, hoping that this would make intuitive sense to the user. To ensure a smooth program flow, I printed ‘No data in queue!’ if there is no data currently in the ‘working memory’ (Figure 5).

1. lstTbl = oldLstTbl + newLstTbl
3. # Check to see if there is data first
4. **if** **not** lstTbl:
5. **print**('No data in queue!\n')

Figure 5 - Combining 2-D lists and an if statement

After making this check, I used a for loop that iterated across the dictionaries and printed each value (Figure 6).

1. **for** row **in** lstTbl:
2. strRow = ''
3. **for** key, val **in** row.items():
4. strRow += str(val) + ' | '
5. strRow = strRow[:-2] + '\n'
6. **print**(strRow)

Figure 6 - Displaying data

## Deleting an entry

After asking the user to input the ID number of the entry they would like to delete, I then used try: except: to make sure that if they accidentally entered a value that could not be converted into a string, the program would terminate gracefully (Figure 7).

1. **try**:
2. intDelChoice = int(strDelChoice)
3. **except**:
4. **print**('An error occured because you failed to enter an ID number.')

Figure 7 - A try/except statement

Then, in order to add an element of practicality, I wanted to allow the user to not just delete an entry from the ‘working memory’ but to also delete that same entry from the file itself (this might occur if the user loaded existing data, displayed that data, and then realized they wanted to delete an entry from the pre-existing data).

So, after clearing the corresponding dictionary within the 2-D list, I cleared it within the file using a [solution](https://www.kite.com/python/answers/how-to-delete-a-line-from-a-file-in-python)[[1]](#footnote-1) (external reference) that I found online after some research (Figure 8). I start by creating a list consisting of each line in the file, and then I overwrite the data in the file by opening it with the ‘w’ argument and using a for loop to rewrite each line except for the line whose first character corresponds to the ID that the user wanted deleted.

1. **for** row **in** lstTbl:
2. **if** row['ID'] == intDelChoice:
3. row.clear()
4. objFile = open(strFileName, 'r')
5. fileData = objFile.readlines()
6. objFile.close()
7. objFile = open(strFileName, 'w')
8. **for** row **in** fileData:
9. **if** row[0] != strDelChoice:
10. objFile.write(row)
11. objFile.close()
12. **print**('\nCD ' + strDelChoice + ' deleted!\n')

Figure 8 - Deleting an entry from 'working memory' and from the file

This does, however, seem like a clunky solution, and I am sure there must be a better way to do it.

## Saving the data

Finally, I used nested for loops to iterate through each dictionary value within each dictionary in the list of new user entries, and after saving the data to the file I cleared each list so that the user begins with a blank ‘queue’ (Figure 9).

1. **elif** strChoice == 's':
2. objFile = open(strFileName, 'a')
3. **for** row **in** newLstTbl:
4. strRow = ''
5. **for** key, val **in** row.items():
6. strRow += str(val) + ' | '
7. strRow = strRow[:-2] + '\n'
8. objFile.write(strRow)
9. objFile.close()
10. oldLstTbl.clear()
11. newLstTbl.clear()
12. lstTbl.clear()
13. **print**('Data saved! Your queue is cleared!\n')

Figure 9 - Saving data in 'working memory' to the file

# Running the script

Figures 10 and 11 are screenshots of my script running in Spyder and Terminal, respectively. The Terminal window screenshot shows the process of deleting an entry.

Text

Description automatically generated

**Text

Description automatically generated**

# Summary

For this assignment, I expanded upon the script that Professor Biesinger wrote so that it included the options of retrieving data from a pre-existing file and deleting user entries. I also tried to add an element of practicality by allowing the user to delete entries from both the program’s ‘working memory’ and the file itself.

# Appendix

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. # nickculbert, 2021-Feb-14, edited main body
7. #------------------------------------------#
9. # Declare variables
11. strChoice = '' # User input
12. oldLstTbl = []
13. newLstTbl = []
14. lstTbl = []  # list of lists to hold data
15. lstRow = []  # list of data row
16. dictRow = {} # dict of list row
17. strDelChoice = ''
18. strFileName = 'CDInventory.txt'  # data storage file
19. objFile = None  # file object
21. objFile = open(strFileName, 'a')
22. objFile.close()
24. # Get user Input
26. **print**('The Magic CD Inventory\n')
27. **while** True:
29. # Display menu allowing the user to choose:
30. **print**('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
31. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
32. strChoice = input('l, a, i, d, s or x: ').lower()  # convert choice to lower case at time of input
33. **print**()
35. # 'x' Exit the program if the user chooses so
36. **if** strChoice == 'x':
37. **break**
39. # 'l' Load existing data
40. **if** strChoice == 'l':
41. objFile = open(strFileName, 'r')
42. **for** row **in** objFile:
43. lstRow = row.split('|')
44. dictRow = {'ID':int(lstRow[0].strip()), 'Title':lstRow[1].strip(), 'Artist':lstRow[2].strip()}
45. oldLstTbl.append(dictRow)
46. objFile.close()
47. **print**('Data loaded!\n')
49. # 'a' Add data to the table (2d-list) each time the user wants to add data
50. **elif** strChoice == 'a':
51. strID = input('Enter an ID: ')
52. strTitle = input('Enter the CD\'s Title: ')
53. strArtist = input('Enter the Artist\'s Name: ')
54. intID = int(strID)
55. dictRow = {'ID':intID, 'Title':strTitle, 'Artist':strArtist}
56. newLstTbl.append(dictRow)
57. **print**('\nData added!\n')
59. # 'i' Display the current data to the user each time the user wants to display the data
60. **elif** strChoice == 'i':
61. **print**('ID | CD Title | Artist')
62. **print**('----------------------\n')
64. lstTbl = oldLstTbl + newLstTbl
66. # Check to see if there is data first
67. **if** **not** lstTbl:
68. **print**('No data in queue!\n')
70. **for** row **in** lstTbl:
71. strRow = ''
72. **for** key, val **in** row.items():
73. strRow += str(val) + ' | '
74. strRow = strRow[:-2] + '\n'
75. **print**(strRow)
76. **print**()
78. # 'd' Deleting an entry
79. **elif** strChoice == 'd':
80. **print**()
81. strDelChoice = input('Please type the ID of the entry you wish to delete: ')
83. # Account for user error
84. **try**:
85. intDelChoice = int(strDelChoice)
86. **except**:
87. **print**('An error occured because you failed to enter an ID number.')
89. **for** row **in** lstTbl:
90. **if** row['ID'] == intDelChoice:
91. row.clear()
92. objFile = open(strFileName, 'r')
93. fileData = objFile.readlines()
94. objFile.close()
95. objFile = open(strFileName, 'w')
96. **for** row **in** fileData:
97. **if** row[0] != strDelChoice:
98. objFile.write(row)
99. objFile.close()
100. **print**('\nCD ' + strDelChoice + ' deleted!\n')
102. # 's' Save the data to a text file CDInventory.txt if the user chooses so
103. **elif** strChoice == 's':
104. objFile = open(strFileName, 'a')
105. **for** row **in** newLstTbl:
106. strRow = ''
107. **for** key, val **in** row.items():
108. strRow += str(val) + ' | '
109. strRow = strRow[:-2] + '\n'
110. objFile.write(strRow)
111. objFile.close()
112. oldLstTbl.clear()
113. newLstTbl.clear()
114. lstTbl.clear()
115. **print**('Data saved! Your queue is cleared!\n')
117. # User policing
118. **else**:
119. **print**('Please choose either l, a, i, d, s or x!')

1. <https://www.kite.com/python/answers/how-to-delete-a-line-from-a-file-in-python> retrieved 2021-Feb-14 [↑](#footnote-ref-1)