Andrew Hanson

2020 Jul 31

IT FDN 110 B Su 20

Assignment04

CD Inventory revisited

# Intro

This assignment focused on dictionaries and working with starter code. When working with starter code it is important to be able to read the code that you are starting with and identify how it functions. This helps avoid breaking the starter code.

# Input

User input takes two forms in this program, the CD data entries to be saved into the inventory file, and the menu navigation keywords. The program also reads data from the CDInventory file.

1. **if** strChoice == 'l':
2. with open(strFileName, 'r') as objFile:
3. **print**(objFile.read()) #prints entire file

Listing 1 - with

I used a context manager to ensure that the file would close. This block of code displays the current entries saved in the inventory file.

A screenshot of a computer

Description automatically generated

Figure 1 - adding entries in spyder

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 = {'ID':strID,'title':strTitle,'artist':strArtist}
8. #chose to keep name lstRow for dictionary to avoid multiple replacements
9. lstTbl.append(lstRow)

Listing 2 - add cd

This block of code is responsible for taking user input and converting it into cd entries in memory.

# Processing

CD entries are stored in memory before being written to the file and may be deleted. Here is the delete function.

1. strChoice2 = input('enter the id number of the entry to be deleted :')
2. **for** row **in** lstTbl:
3. **if** str(row['ID']) == strChoice2: #allows choosing by id number
4. gone = lstTbl.pop(row) #pop returns the removed value allowing it to be printed after removal
5. **print**(gone['ID'],' ',gone['title'],' ',gone['artist'],' deleted!')
6. **print**()

Listing 3 - delete

# Output

The program can also output the current inventory file to the user in the console. The program also outputs current unsaved entries whenever a new entry is added. The program will create a new inventory file if none exists.

A screenshot of a computer

Description automatically generated

Figure 2 - display file inventory

1. **elif** strChoice == 'i':
2. # 3. Display the current data to the user each time the user wants to display the data
3. **if** lstTbl != []: #check table is not empty
4. **print**('ID, CD Title, Artist') #header for display
5. **for** row **in** lstTbl: #print the values of each dictionary seperated by a comma and space
6. **print**(\*row.values(), sep = ', ')
7. **else**: #just in case
8. **print**('No CD entries!')

Listing 4 - display

This block of code is responsible for displaying the current cd entries saved in memory.

# Challenges

I had some issues parsing through the dictionary entries to make the delete function work until I remembered that dictionaries can be searched by keyword, which is sort of the whole point of dictionaries.

1. strChoice2 = input('enter the id number of the entry to be deleted :')
2. **if** str(row['ID']) == strChoice2: #allows choosing by id number
3. index = 0
4. **print**(row['ID'],' ',row['title'],' ',row['artist'],' deleted!')
5. #print is before deletion because refernced variables will not exist after deletion
6. **del**(lstTbl[index]) #index will equal the index of the list entry currently accessed
7. index += 1

It also took some work to allow the existing functions of the program to work with a dictionary instead of a list.

# Summary

When programs become more complicated it can be hard to keep track of everything. By breaking the program down into smaller parts, it is easier to keep things organized and modify functionality in the future without breaking things. Dictionaries allow directly labeling entries. It is important for code to be readable by humans.

# Questions

## • What is the difference between a Dictionary and a List?

Dictionaries contain keywords.

## • What is the difference between an index and a key?

An index identifies an entry in a list based on its position. If an entry’s position in the list changes so does its index. A keyword pairs a unique identifier with each entry that is irrelevant of the entry’s position in the dictionary.

## • How do you read data from a file into a list?

The method readlines creates a list of the lines of a file.

## • How do you read data from a file into a dictionary?

First you would need to read the data. Then you would need a way to identify district pieces of data from each other. Then you would need to assign appropriate keywords to the data and store it into a dictionary. If multiple related pieces of related pieces of data existed, then a list of dictionaries would allow reusing the same keyword for the related pieces of data.

## • Why is it making sense to organize data in a 2-dimensional way?

Storing data in a list makes sense when the data is similar and can be parsed in the same way. Putting data into a list of lists makes sense when there are multiple instances of multiple related pieces of data that can be parsed as a group in a similar way. Lists are useful when an unknown number of entries are required. Creating a new variable for each entry would be cumbersome. Using a list of dictionaries allows using the same keyword multiple times.

## • What is the programming pattern “Separation of Concerns”?

Separation of concerns advocates for splitting your program into sub parts that each deal with small parts of the overall problem.

## • How would you use a function to organize your code?

Functions allow separating concerns. With functions a loop of functions could be created. Then, if there was a loop inside one of the functions it would not have to be contained in the same part of the program as the original loop. Functions allow code to be stored in separate areas of the program for better organization.

## • Why is a script template useful?

A script template helps break the problem down into smaller pieces.

## • Why is error handling (try-except) useful?

Because it can allow your program to not crash when exceptions would be raised.

## • What is GitHub and why is it used?

Github is a code repository and version software. It is used to keep code projects organized and allow easier collaboration.

## • What is GitHub’s mascot?

octocat

# Appendix

## Sources

<https://www.youtube.com/watch?v=Uh2ebFW8OYM&t=377s>

## Repository

Github: <https://github.com/clichekhfan/IT-FDN-110-B-Su-20/upload/master>

## CDInventory.py repository

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. # Ahanson, 2020-aug-05, draft two started
7. # Ahanson, 2020-aug-05, load function added
8. # Ahanson, 2020-aug-05, added context manager to write function
9. # Ahanson, 2020-aug-05, fixed bug where list not emptied after file write
10. # Ahanson, 2020-aug-05, No CD Entries!
11. # AHanson, 2020-aug-08, Entry deletion added
12. # AHanson, 2020-aug-08, code commented and cleaned up
13. # AHanson, 2020-aug-08, added print of stored entries after addding an entry

16. #TO-DO
17. #1)change the display menu so it prints before or after other information
18. #printed by the program, contextually
19. #2)find a use for exception handling
20. #3)check for valid entries
21. #4)allow deleting entries from the file
22. #5)dont allow duplicate entries
24. #------------------------------------------#
26. # Declare variabls
28. strChoice = '' # User input
29. lstTbl = []  # list of lists to hold data
30. # TODO replace list of lists with list of dicts
31. lstRow = {}  # list of data row
32. strFileName = 'CDInventory.txt'  # data storage file
33. objFile = None  # file object
35. #create file if none exists
36. # with is a context manager
37. # in this context, it will create a variable objFile and reference the opened file,
38. # strFileName in append mode. Then, it will run the code in the indent block and close the file.
39. # with context managers you will never forget to close your file. pun intended.
40. # note strFileName is a variable referncing the string 'CDInventory.txt'
41. with open(strFileName, 'a') as objFile:
42. **pass** #pass is useful when syntax requires code in the indent block but you
43. #don't actually want your program to do anything
45. # Get user Input
46. **print**('The Magic CD Inventory\n')
47. **while** True:
48. # 1. Display menu allowing the user to choose:
49. **print**('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
50. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
51. strChoice = input('l, a, i, d, s or x: ').lower()  # convert choice to lower case at time of input
52. **print**()
54. **if** strChoice == 'x':
55. # 5. Exit the program if the user chooses so
56. **break**
57. **if** strChoice == 'l':
58. **print**('ID, CD Title, Artist') #header for display
59. with open(strFileName, 'r') as objFile:
60. **print**(objFile.read()) #prints entire file
61. **pass**
62. **elif** strChoice == 'a':  # no elif necessary, as this code is only reached if strChoice is not 'exit'
63. # 2. Add data to the table (2d-list) each time the user wants to add data
64. strID = input('Enter an ID: ')
65. strTitle = input('Enter the CD\'s Title: ')
66. strArtist = input('Enter the Artist\'s Name: ')
67. intID = int(strID)
68. lstRow = {'ID':strID,'title':strTitle,'artist':strArtist}
69. #chose to keep name lstRow for dictionary to avoid multiple replacements
70. lstTbl.append(lstRow)
71. #display current entries in memory efter each entry is added
72. **if** lstTbl != []: #check table is not empty
73. **print**('ID, CD Title, Artist') #header for display
74. **for** row **in** lstTbl: #print the values of each dictionary seperated by a comma and space
75. **print**(\*row.values(), sep = ', ')
76. **elif** strChoice == 'i':
77. # 3. Display the current data to the user each time the user wants to display the data
78. **if** lstTbl != []: #check table is not empty
79. **print**('ID, CD Title, Artist') #header for display
80. **for** row **in** lstTbl: #print the values of each dictionary seperated by a comma and space
81. **print**(\*row.values(), sep = ', ')
82. **print**()
83. **else**: #just in case
84. **print**('No CD entries!')
85. **elif** strChoice == 'd':
86. **if** lstTbl != []:
87. **print**('ID, CD Title, Artist')
88. **for** row **in** lstTbl: #display options to be deleted
89. **print**(\*row.values(), sep = ', ')
90. strChoice2 = input('enter the id number of the entry to be deleted :')
91. **for** row **in** lstTbl:
92. **if** str(row['ID']) == strChoice2: #allows choosing by id number
93. gone = lstTbl.pop(row) #pop returns the removed value allowing it to be printed after removal
94. **print**(gone['ID'],' ',gone['title'],' ',gone['artist'],' deleted!')
95. **print**()
96. **else**:#just incase
97. **print**('No CD entries!')
98. **elif** strChoice == 's':
99. # 4. Save the data to a text file CDInventory.txt if the user chooses so
100. with open(strFileName, 'a') as objFile:
101. **for** row **in** lstTbl:
102. strRow = ''
103. **for** item **in** row.values():
104. strRow += str(item) + ','
105. strRow = strRow[:-1] + '\n'
106. objFile.write(strRow)
107. **print**('Entries saved!')
108. lstTbl = []  # list of data row
109. #empties table in memory after written to file to prevent double entries
110. **else**:
111. **print**('Please choose either l, a, i, d, s or x!')