

Modifying a script using Lists and Dictionaries and other useful python programming concepts and tools

Introduction

For the 5th Module, we learned about lists and dictionaries which would be applied to the CDInventory.py script. In the script assignment we were asked to modify an existing script to include additional functionalities such as loading data into memory and deleting data. We also learned about other topics such as separation of concerns, error handling and functions. Lastly, we learned about script templates and Github. The purpose of this document is to explore some of those learnings from Module 5.

Lists and Dictionaries

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.¹ The beauty of using lists is that it is mutable meaning one can add to values to the list as needed. The original script were using list to store the data collected into a sequence. The task is to modify the script to replace the data struction using dictionaries. Dictionaries are also mutable which works well for this assignment. There is currently only one standard mapping type, the dictionary.² One of the features of dictionaries is a key:value pair. The assignment of the data to a key allows for searchability of the input data.

Separation of concerns, error handling and functions

Another topic discussed in this module is Separation of concerns which is a design principle for separating a computer program into distinct sections such that each section addresses a separate concern.³ The main takeaway with the application of separation of concerns is to intentionally break up the code in tiny sections instead of writing the code in one solid block. When the problem is broken up it easier for the programmer to address the individual tasks within the script. Structured error handling is a way for a programmer to minimize the errors within a script or when implementing the script. This can be done by setting up controls or exceptions. As an example, when using division function, the programmer may choose to limit the use of zero as a denominator or even filter for zero to return a scripted error message versus python being unable to process the data.

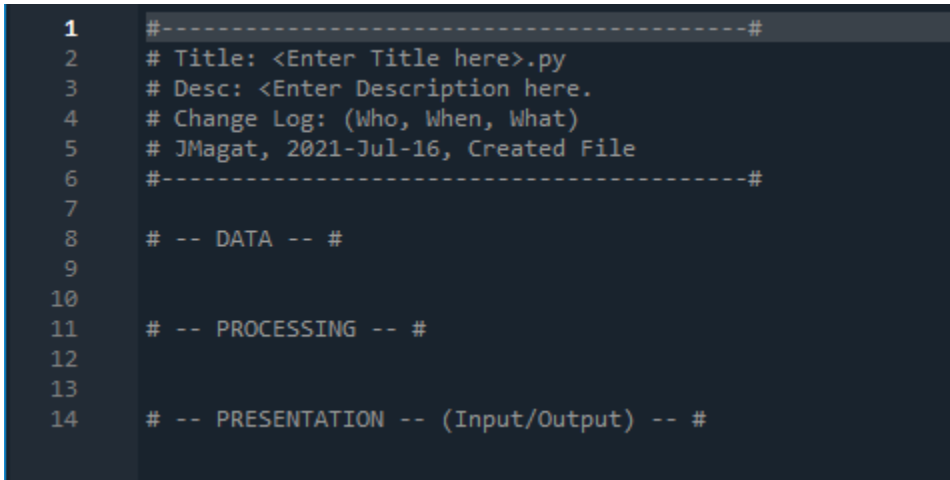
¹ https://www.tutorialspoint.com/python/python_lists.htm, retrieved 2021-Jul-31

² [Built-in Types — Python 3.9.6 documentation](#), retrieved 2021-Aug-08

³ https://en.wikipedia.org/wiki/Separation_of_concerns, retrieved 2021-Aug-08

Script templates and Github

In previous modules, we learned about the importance of using headers to document the who, what and where of a script. Instead of building up the header format every time a new script is created, one can build a script template within spyder preference settings. Figure 1 below displays a sample template in spyder.



```
1  #-----#
2  # Title: <Enter Title here>.py
3  # Desc: <Enter Description here.
4  # Change Log: (Who, When, What)
5  # JMagat, 2021-Jul-16, Created File
6  #-----#
7
8  # -- DATA -- #
9
10
11 # -- PROCESSING -- #
12
13
14 # -- PRESENTATION -- (Input/Output) -- #
```

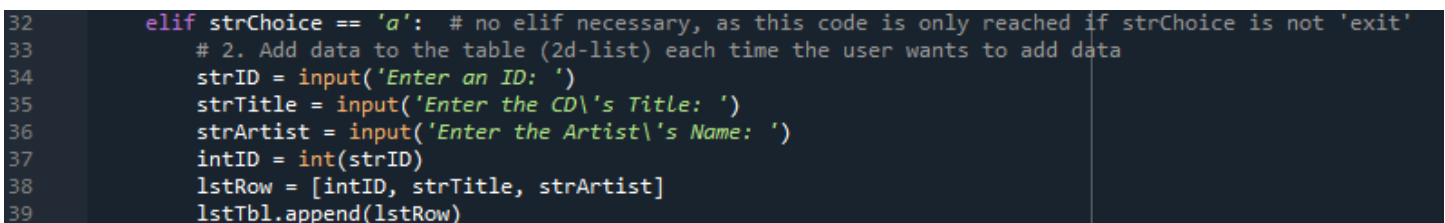
Figure 1 – Sample Script Template in Spyder

Git is software for tracking changes in any set of files, usually used for coordinating work among programmers collaboratively developing source code during software development.⁴ GitHub is a free web-based portal that is particularly helpful when working with a team so that any changes or revisions are tracked and accessible to all. The script assignment for this Module is a good example of a script that that could be uploaded to GitHub since it included modifications to the original script. I created a repository of work related to Assignment 05 available in this link: https://github.com/jlmagat/Assignment_05.

Modifying the CDInventory.py script.

For this assignment, a CDInventory_Starter.py script was provided as a starting to implement changes that included the use of dictionaries, and the added functionality of loading existing data and deleting an entry.

Figure 2 below displays the original script for adding user data to a 2d-list.



```
32 elif strChoice == 'a': # no elif necessary, as this code is only reached if strChoice is not 'exit'
33     # 2. Add data to the table (2d-list) each time the user wants to add data
34     strID = input('Enter an ID: ')
35     strTitle = input('Enter the CD\'s Title: ')
36     strArtist = input('Enter the Artist\'s Name: ')
37     intID = int(strID)
38     lstRow = [intID, strTitle, strArtist]
39     lstTbl.append(lstRow)
```

Figure 2 – Original code for adding user data

⁴ <https://en.wikipedia.org/wiki/Git>, retrieved 2021-Aug-08

To use dictionaries, I modified the inner structure of the data to and replaced the original `lstRow` variable with a `dicRow`. To implement the dictionary, I identified the keys which included `id`, `title`, and `artist` that will be used for mapping the collected data from the user. The `dicRow` variable rolls up to the `lstTbl` variable and appends the table with new information collected from the user. Figure 3 below displays the updated script for adding user data using dictionaries.

```
52     elif strChoice == 'a': # no elif necessary, as this code is only reached if strChoice is not 'exit'
53         # 4. Add data to the table (2d-list) each time the user wants to add data
54         ## Collect user input
55         strID = input('Enter an ID: ')
56         strTitle = input('Enter the CD\'s Title: ')
57         strArtist = input('Enter the Artist\'s Name: ')
58         intID = int(strID)
59         ## Put user input to dictionary
60         dicRow = {'id': intID, 'title': strTitle, 'artist': strArtist}
61         ## Append row of user input into table
62         lstTbl.append(dicRow)
63
```

Figure 3 – Modified code for adding user data using dictionaries.

The modification was easy enough to do but a challenge that I later realized was that changing the structure had a downstream effect in printing the data to terminal and saving the 2D list to file. Thankfully, it was just an easy modification such that when I am unpacking the row in the `lstTbl` using a for loop, I needed use the `values()` method to return values in a given directory. Figure 4 below displays code for displaying user data which included the use of the `values()` method.

```
64     elif strChoice == 'i':
65         # 5. Display the current data to the user each time the user wants to display the data
66         print('This is the Current CD Inventory')
67         print('ID, CD Title, Artist')
68         ## Use a for loop to print list values as separate lines
69         for row in lstTbl:
70             print(*row.values(), sep = ', ')
71         print()
72
```

Figure 4 – Modified code for displaying inventory using the `values()` method.

Next, I created the script to load data from a text file and load it into memory. It was helpful to have access to the sample solutions from one of lab assignments as it provided me a framework for creating the script for this task. This involved using a `split ()` and `strip ()` function to generate a list out of text file which I was not familiar with. The list data is then mapped out to the keys in the dictionary before storing it the `lstTbl` in memory. Figure 5 below displays the code for loading the data from the text file and storing it to memory.

```

31     if strChoice == 'l':
32         # 3.TODO Add the functionality of loading existing data
33         # Read the file line by line into in-memory list.
34         ## Open the text file
35         objFile = open(strFileName, 'r')
36         ## Use for loop to identify the 1st Row
37         for row in objFile:
38             lstRow = row.strip().split(',')
39             ## Map the row items to the dictionary key
40             dicRow = {'id': lstRow[0], 'title': lstRow[1], 'artist': lstRow[2]}
41             ## Append the loaded data into the lstTbl
42             lstTbl.append(dicRow)
43         objFile.close()
44         ## Display the data loaded into the lstTbl
45         print('This is the Current CD Inventory')
46         print('ID, CD Title, Artist')
47         ## Use a for loop to print separate row values as separate lines
48         for row in lstTbl:
49             print(*row.values(), sep = ', ')
50         print()
51

```

Figure 4 – Code for retrieving data from the text file and storing it to memory.

The final modification to the CDInventory.py script is to add a functionality for deleting an entry. It took a long time for me to develop a code for this task. It was challenging not having a starting code to modify and I tried multiple ways of mapping out the user selection of CD delete and implement it so that it is removed from the lstTbl. At first, I required the user to indicate all the available information for the CD that need to be deleted. However, I was getting a value error when implementing the code. Figures 5 and 6 below display the code I testing for deleting an entry and the resulting error.

```

72     elif strChoice == 'd':
73         # TODO Add functionality of deleting an entry
74
75         print('Enter the information for the entry you wish to delete.')
76         ## Identify the CD Id to be deleted
77         strID = input('Enter an ID: ')
78         strTitle = input('Enter the CD\'s Title: ')
79         strArtist = input('Enter the Artist\'s Name: ')
80         intID = int(strID)
81         ## Put user input to dictionary
82         dicRow = {'id': intID, 'title': strTitle, 'artist': strArtist}
83         ## Loop through list to match the CD id
84         lstTbl.remove([dicRow])
85         print('This is the Current CD Inventory')
86         print('ID, CD Title, Artist')
87         for row in lstTbl:
88             print(*row.values(), sep = ', ')
89

```

Figure 5 – Test code for deleting a CD entry

```

Enter the information for the entry you wish to delete.

Enter an ID: 1

Enter the CD's Title: The Big Wheel

Enter the Artist's Name: Runrig
Traceback (most recent call last):

  File "C:\_FDProgramming\Mod_05\CDInventory_with notes.py", line 84, in <module>
    lstTbl.remove([dicRow])
ValueError: list.remove(x): x not in list

```

Figure 6 – Value Error for test code for deleting a CD entry.

After many attempts, I was successful in requiring the user to input one key value which was the CD Title and used the remove() method to filter through the dictionary to remove all the items related to CD selected by the user including the ID, and the Artist name. Figure 7 below displays the final code for deleting a CD entry.

```

73     elif strChoice == 'd':
74         # 6.TODO Add functionality of deleting an entry
75         print('Enter the information for the entry you wish to delete.')
76         ## Identify the CD Id to be deleted
77         strTitle = input('Enter the CD Title to be deleted: ')
78         ## Loop through list to match the CD Title and use remove method
79         for cd in lstTbl:
80             if cd['title'] == strTitle:
81                 lstTbl.remove(cd)
82         print('You have removed a CD. This is the new CD Inventory')
83         print('ID, CD Title, Artist')
84         ## Use a for loop to print remaining list values as separate lines
85         for row in lstTbl:
86             print(*row.values(), sep = ', ')
87         print()

```

Figure 7 – Final code for deleting a CD entry.

Running the Python Script

After saving the file, I ran the CDInventory script in both Spyder and the Terminal. Figures 8 and 9 below displays the script working on the computer that all options are running correctly.

```

Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.22.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/_FDProgramming/Mod_05/Assignment05/CDInventory.py', wdir='C:/_FDProgramming/
Mod_05/Assignment05')
The Magic CD Inventory

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s or x: l

This is the Current CD Inventory
ID, CD Title, Artist
1, The Big Wheel, Runrig
2, Bad, Michael Jackson

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s or x: a

Enter an ID: 3

Enter the CD's Title: Abbey Road

Enter the Artist's Name: The Beattles
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

```

Figure 8 – Running CDInventory.py in Spyder

```
l, a, i, d, s or x: i

This is the Current CD Inventory
ID, CD Title, Artist
1, The Big Wheel, Runrig
2, Bad, Michael Jackson
3, Abbey Road, The Beattles

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s or x: d

Enter the information for the entry you wish to delete.

Enter the CD Title to be deleted: Abbey Road
You have removed a CD. This is the new CD Inventory
ID, CD Title, Artist
1, The Big Wheel, Runrig
2, Bad, Michael Jackson

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s or x: s

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s or x: x
```

Figure 8a – Running CDInventory.py in Spyder – continued

```
Anaconda Prompt (anaconda3)

(base) C:\_FDProgramming\Mod_05\Assignment05>python CDInventory.py
The Magic CD Inventory

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit
l, a, i, d, s or x: l

This is the Current CD Inventory
ID, CD Title, Artist
1, The Big Wheel, Runrig
2, Bad, Michael Jackson

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit
l, a, i, d, s or x: a

Enter an ID: 3
Enter the CD's Title: Abbey Road
Enter the Artist's Name: The Beattles
[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit
l, a, i, d, s or x: i

This is the Current CD Inventory
ID, CD Title, Artist
1, The Big Wheel, Runrig
2, Bad, Michael Jackson
3, Abbey Road, The Beattles
```

Figure 9 – Running CDInventory.py in Anaconda


```
Anaconda Prompt (anaconda3)
[1] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit
l, a, i, d, s or x: d

Enter the information for the entry you wish to delete.
Enter the CD Title to be deleted: Abbey Road
You have removed a CD. This is the new CD Inventory
ID, CD Title, Artist
1, The Big Wheel, Runrig
2, Bad, Michael Jackson

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit
l, a, i, d, s or x: s

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit
l, a, i, d, s or x: x

(base) C:\_FDProgramming\Mod_05\Assignment05>
```

Figure 9a – Running CDInventory.py in Anaconda – continued

```
*CDInventory.txt - Notepad
File Edit Format View Help
1,The Big Wheel,Runrig
2,Bad,Michael Jackson
```

Figure 10 below displays the .txt file where the user data is stored.

Figure 10 – Stored data in CDInventory.txt

Summary

Module 5 continues to stretch my knowledge and my application of various python programming concepts including the use of dictionaries and lists in a code for collecting sequences of data. I found that it quite challenging to modify an existing script as one change could have downstream effects to other sections of the code. Learning about separation of concerns and error handling gives me an idea how to streamline my work and prevent future errors. Having access to tools such as script templates help speed up the process while GitHub allows for version tracking and efficient collaboration with work teams.

Appendix

Listing CDInventory.py – part 1

```
1  #-----#
2  # Title: CDInventory.py
3  # Desc: CD Inventory Script for Assignment 05
4  # Change Log: (Who, When, What)
5  # DBiesinger, 2030-Jan-01, Created File
6  # JMagat, 2021-Aug-08, Modified script to use dictionary, load data from memory and remove entry.
7  #-----#
8
9  # Declare variables
10
11  strChoice = '' # User input
12  lstTbl = [] # list of lists to hold data
13  # TODO replace list of lists with list of dicts
14  dicRow = {} # list of data row
15  strFileName = 'CDInventory.txt' # data storage file
16  objFile = None # file object
17
18  # Get user Input
19  print('The Magic CD Inventory\n')
20  while True:
21      # 1. Display menu allowing the user to choose:
22      print('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
23      print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
24      strChoice = input('l, a, i, d, s or x: ').lower() # convert choice to lower case at time of input
25      print()
26
27      if strChoice == 'x':
28          # 2. Exit the program if the user chooses so
29          break
30
31      if strChoice == 'l':
32          # 3.TODO Add the functionality of loading existing data
33          # Read the file line by line into in-memory list.
34          ## Open the text file
35          objFile = open(strFileName, 'r')
36          ## Use for loop to identify the 1st Row
37          for row in objFile:
38              lstRow = row.strip().split(',')
39              ## Map the row items to the dictionary key
40              dicRow = {'id': lstRow[0], 'title': lstRow[1], 'artist': lstRow[2]}
41              ## Append the loaded data into the lstTbl
42              lstTbl.append(dicRow)
43          objFile.close()
44          ## Display the data loaded into the lstTbl
45          print('This is the Current CD Inventory')
46          print('ID, CD Title, Artist')
47          ## Use a for loop to print separate row values as separate lines
48          for row in lstTbl:
49              print(*row.values(), sep = ', ')
50          print()
51
```

Listing CDInventory.py – part 2

```
52 elif strChoice == 'a': # no elif necessary, as this code is only reached if strChoice is not 'exit'
53     # 4. Add data to the table (2d-list) each time the user wants to add data
54     ## Collect user input
55     strID = input('Enter an ID: ')
56     strTitle = input('Enter the CD\'s Title: ')
57     strArtist = input('Enter the Artist\'s Name: ')
58     intID = int(strID)
59     ## Put user input to dictionary
60     dicRow = {'id': intID, 'title': strTitle, 'artist': strArtist}
61     ## Append row of user input into table
62     lstTbl.append(dicRow)
63
64 elif strChoice == 'i':
65     # 5. Display the current data to the user each time the user wants to display the data
66     print('This is the Current CD Inventory')
67     print('ID, CD Title, Artist')
68     ## Use a for loop to print list values as separate lines
69     for row in lstTbl:
70         print(*row.values(), sep = ', ')
71     print()
72
73 elif strChoice == 'd':
74     # 6.TODO Add functionality of deleting an entry
75     print('Enter the information for the entry you wish to delete.')
76     ## Identify the CD Id to be deleted
77     strTitle = input('Enter the CD Title to be deleted: ')
78     ## Loop through list to match the CD Title and use remove method
79     for cd in lstTbl:
80         if cd['title'] == strTitle:
81             lstTbl.remove(cd)
82     print('You have removed a CD. This is the new CD Inventory')
83     print('ID, CD Title, Artist')
84     ## Use a for loop to print remaining list values as separate lines
85     for row in lstTbl:
86         print(*row.values(), sep = ', ')
87     print()
88
89 elif strChoice == 's':
90     # 7. Save the data to a text file CDInventory.txt if the user chooses so
91     ## Open the text file using an append access
92     objFile = open(strFileName, 'a')
93     ## Unpack the list of list as separate entry per CD
94     for row in lstTbl:
95         strRow = ''
96         for item in row.values():
97             strRow += str(item) + ','
98         strRow = strRow[:-1] + '\n'
99         ## Save the file into the text file
100         objFile.write(strRow)
101     objFile.close()
```

Listing CDInventory.py – part 3

```
103 else:
104     # 8. Display options
105     ## Print the menu option for the user
106     print('Please choose either l, a, i, d, s or x!')
107
```