Makena Mezistrano

February 23, 2020

Foundations of Programming, Python

Assignment 05

CD Inventory Dictionary Python Script

Introduction

In this assignment I will explain the steps I took to modify a pre existing solution to last week's CD inventory program to use dictionaries as the inner data type. The program will allow the user to view current inventory from in memory, load data from a text file, add new data to memory, save data to a text file, and quit the program. This script was saved to Github, an online version control system.

Drafting the script

Reviewing the file

The script for this assignment was based on a pre-existing file called CDInventory_Starter.py. To begin this assignment, I opened the file and reviewed the TODO items.

Defining variables and beginning the dictionary structure

To start the script I first defined all the variables and created the dictionary structure. The first variable, strChoice, would later allow the user to choose from the menu. The dicRow variable, followed by empty curly brackets { }, defined the inner list structure as a dictionary. The lstTbl variable would be the list of dicts holding all the data. I then defined the file name and the objFile. Finally, to make the script run well the first time it's used, I immediately input two dictionary entries as dicRow1 and dicRow2 and used the .append( ) function to add them to the table.

```
# Declare variables

strChoice = '' # User input
dicRow = {}  # inner lists of dicts
lstTbl = []  # list of dicts to hold data
strFileName = 'CDInventory.txt'  # data storage file
objFile = None  # file object
dicRow1 = {'ID': 1, 'CD title': 'Happy', 'Artist name': 'Celine Dion'}
dicRow2 = {'ID': 2, 'CD title': 'Yellow', 'Artist name': 'Coldplay'}
lstTbl.append(dicRow1)
lstTbl.append(dicRow2)

```

Figure 1: Declaring variables.

The menu[1]

This script came preloaded with the beginnings of the while loop and the elif statements required for the user to move through the menu options.

[1] Everything from this section through 'Saving the file' is a combination of processing and presentation SoC, but could not be clearly differentiated in the code at this phase of my knowledge.

The script begins with a print statement to introduce the program followed by the while loop. The first step in the while loop is to display the menu of options. These options are string data types. An input function is used for the user to select from the menu. The script also added the .lower( ) function to the input so that, in the event the user entered a menu option in an uppercase letter, the script would automatically convert it to a lowercase letter.

Exit the program

The first elif statement within this while loop is an **if statement** that allows the user to quit the program. If the user selects this option, it will break the while loop. The option to break the program is listed first so that the program will not have to cycle through all the script in order to reach this statement; if the user wants to quit, the program will quit efficiently.

All of the elif statements that follow are based on this if statement and are short for "else if." Once the first if statement proves False (which it will if the user inputs anything besides for 'x'), the program will run through the next blocks of elif statements until it can run a True statement. If it runs through the if and elif statements and cannot return anything True, the program will default to the final **else statement.** This prints a reminder to the user to only enter one of the letters from the menu.[2]

```
21 # Get user Input
22 print('The Magic CD Inventory\n')
23 while True:
24     # 1. Display menu allowing the user to choose:
25     print('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
26     print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
27     strChoice = input('l, a, i, d, s or x: ').lower()  # convert choice to lower case at time of input
28     print()
29
30     if strChoice == 'x':
31         # 5. Exit the program if the user chooses so
32         break
33
```

Figure 2: The while loop with the menu and the first if statement, which will quit the program.

Load existing data from a file

The next elif statement in the script offers the user the option to load existing data from a file. This is also the next place in the assignment where I had to add a substantial amount of code. To load data from a file, I opened the file and used the read function ['r'], because I only wanted to show what was in the file and did not need to write or append anything. Then I used a for loop to do two things: 1) clean the dictionaries in the list by stripping the space from the end of each row and adding commas between the rows for readability; and 2) define which rows in the dictionary to print via zero-based indexing. To display the file data to the user I used a print statement, then closed the file.[3]

```
34     if strChoice == 'l':
35         # Adds the functionality of loading existing data
36         objFile = open(strFileName, 'r')
37         for row in objFile:
38             lstRow = row.strip().split(',')
39             dicRow = {'ID': lstRow[0], 'CD title': lstRow[1], 'Artist name': lstRow[2]}
40             print(dicRow)
41         objFile.close()
```

Figure 3: Loading existing data from a file.

[2] Adapted from knowledge document on Assignment04.
[3] Script based on the solution from LAB05-B.

Add data

The next elif statement was preloaded into the script and gives the user the option to add data to the inventory. This portion of the script includes a series of inputs that correspond to the **keys** in the CD inventory dictionary: ID, CD title, and Artist name. The ID was cast as an integer and the rest of the data was cast as a string; this will work in the dictionary because dictionaries can hold multiple data types.[4] At this point, each of these inputs are considered single **values**; in order to be added to the dictionary, these items need to be matched with the proper **keys.** This was done by creating a new dictionary with curly brackets. Then, the new dictionary needed to be added to the list; this was done using the .append function. The .append function works because lists are mutable (able to be changed).[5]

```
43    elif strChoice == 'a':  # no elif necessary, as this code is only reached if strChoice is not 'exit'
44        # 2. Add data to the table (2d-list) each time the user wants to add data
45        strID = input('Enter an ID: ')
46        strTitle = input('Enter the CD\'s Title: ')
47        strArtist = input('Enter the Artist\'s Name: ')
48        intID = int(strID)
49        dicRow = {'ID': intID, 'CD title': strTitle, 'Artist name': strArtist}
50        lstTbl.append(dicRow)
```

Figure 4: Elif statement allowing user to add data to dictionary; appends dictionary to the list.

Display current data

The next elif statement also came preloaded in the code but required a small tweak in order to work properly with a dictionary structure. This next section allows the user to display the current inventory from in memory. This section begins with a print statement that lists all the **keys** from the dictionary and acts as a header for the inventory that will be printed out. Next, a for loop goes through the table, upacks the table, and prints only the **values** in each dictionary into neat rows with each value separated by a comma (this is the part that I tweaked). The table is upacked using the * operator, which is a shorthand operator. It's also an overloaded operator since it can also be used for multiplication.[6]

```
52    elif strChoice == 'i':
53        # 3. Display the current data to the user each time the user wants to display the data
54        print('ID, CD Title, Artist')
55        for row in lstTbl:
56            print(*row.values(), sep = ', ') # Changed this to row.values() for readibility
```

Figure 5: Elif statement allows user to display current inventory.

If I had left the for loop unpacking the entire row, then the keys and values would both be printed, and the first print statement would be redundant. The output would also look messy.

Delete an entry

The next elif statement required me to add code in order for the user to be able to delete an entry. I started with a print statement to guide the user and reminded the user about the current inventory (using the same syntax I described in the previous section). Then I used an input function to prompt the user to enter the ID of the CD he/she may want to delete. I then used a for loop with a nested if statement to search through the **values** in the dictionaries to find the ID the user had entered. If the value is found, the delete statement kicks in to remove the

[4] https://realpython.com/python-dicts/#dpopitem, retrieved 2020-Feb-23.
[5] Adapted from knowledge document on Assignment04.
[6] Mod 05 pg. 2

entire row from the table. To show the user what he/she had done, I added another print statement with a for loop just like the one with which this section began to display the new inventory.[7]

```
58    elif strChoice == 'd':
59        # Adds functionality of deleting an entry
60        print('This is your current inventory:')
61        for row in lstTbl:
62            print(*row.values(), sep = ', ')
63        deleteID = int(input('Enter the ID of the CD you want to delete: '))
64        for row in lstTbl:
65            if deleteID in row.values():
66                del lstTbl[lstTbl.index(row)]
67        print('\n This is your new inventory:')
68        for row in lstTbl:
69            print(*row.values(), sep = ', ')
70
```

Figure 6: Elif statement that allows user to delete an entry.

Save data to text file

The final elif statement was also preloaded into the script and enables the user to save data to a text file called CDInventory.txt. All of this data is cast as strings because lists cannot be written to files. This section also required a little bit of tweaking in order to work with the dictionary structure. To save the data, I first opened the file: in this line of code, I changed an append 'a' command to a write 'w' command in order to add the new data to the file. With the 'w' command, if the file does not exist, a new file will be created. I retained the initial lines of the for loop that followed, but in the nested for loop, I specified that the script needed to record the row.values( ) into the file. I separated each item (or value) with a comma for readability, and also used the [:-1] splicing to remove the comma from the end of each line in the file. The .write function then added the new entry (strRow) to the file. The file is then closed.

```
71    elif strChoice == 's':
72        # 4. Save the data to a text file CDInventory.txt if the user chooses so
73        objFile = open(strFileName, 'w') # changed from a to w so that it would only add the new line
74        for row in lstTbl:
75            strRow = ''
76            for item in row.values(): # only row.values() needed changing from list code
77                strRow += str(item) + ','
78            strRow = strRow[:-1] + '\n'
79            objFile.write(strRow)
80        objFile.close()
```

Figure 7: Final elif statement enables user to save data to text file.

Saving the script

I saved the script in a folder created for this course under the user destination on my Mac and renamed the file CDInventory.py. I edited the TODO pseudocodes and added comments throughout to explain my thought process, particularly in places where I had made changes. Finally, I revised the header.

Running the script, verifying functioning

To test the script, I first ran and tested all functions of the script in Spyder. Everything worked.

[7] Based on the solution from: https://github.com/kellyros12/Assignment_05/blob/master/CDInventory.py retrieved 2020-Feb-23.

```
In [81]: runfile('/Users/Makena/Desktop/Mod_05/CDInventory_Starter.py', wdir='/Users/Makena/Desktop/Mod_05')
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

ID, CD Title, Artist
1, Happy, Celine Dion
2, Yellow, Coldplay
[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: Baby

Enter the Artist's Name: Britney Spears
[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: d

This is your current inventory:
1, Happy, Celine Dion
2, Yellow, Coldplay
3, Baby, Britney Spears

Enter the ID of the CD you want to delete: 2

 This is your new inventory:
1, Happy, Celine Dion
3, Baby, Britney Spears
[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


In [82]:
```

Figure 8: The working script in the Spyder terminal.

I checked the text file to make sure the data had been written in properly; it had.

Figure 9: CDInventory.txt with the written data.

I then tested the script from the terminal window, with different data inputs, from my desktop. It also worked properly. The new data was also in the text file.

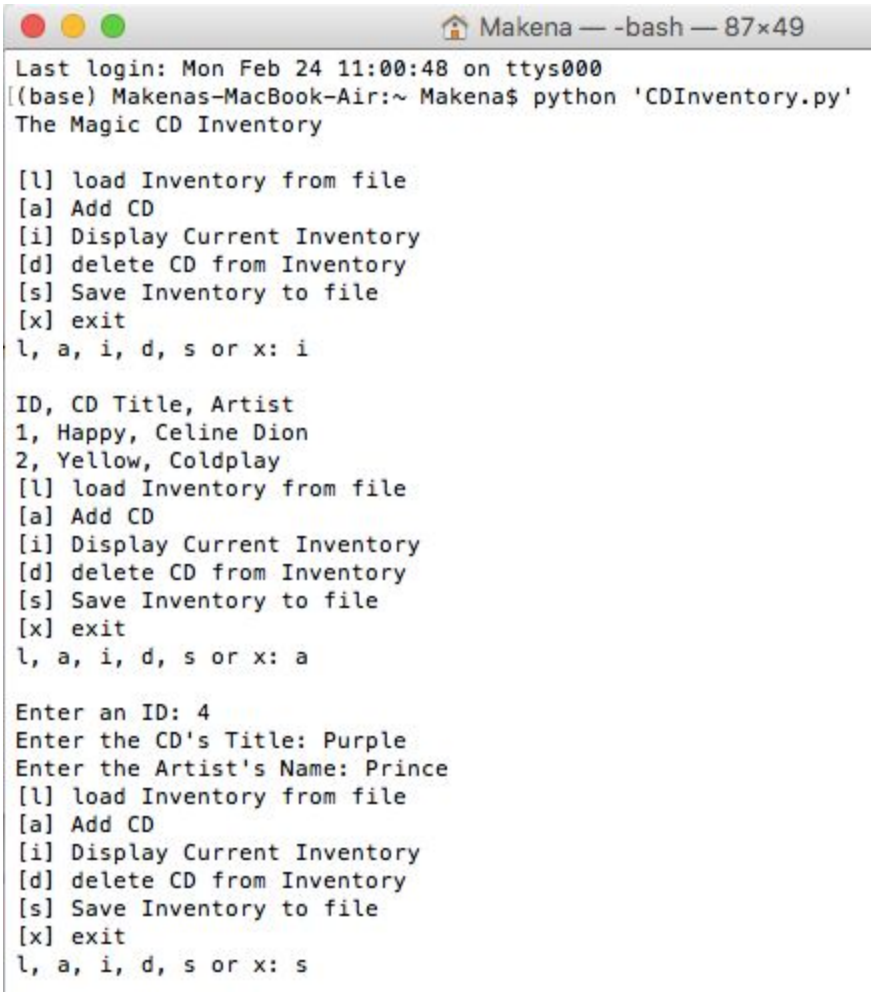

Figure 10: The working script in IDLE.

Figure 11: Data displayed in CDInventory.txt.

Uploading to Github

Part of this assignment also involved saving the script to Github, which is an online, version control system. Github is important because it creates a repository for code so that when teams are working on a project at the same time, members can save versions of the code to Github and return to previous solutions as they work.[8] To save this script (and this knowledge document) to Github, I created a repository called Assignment_05 and placed the files there. I then shared the link to this repository on the discussion board for this part of the assignment on canvas. Here is the link to my repository: https://github.com/makenaflory/Assignment_05

[8] https://youtu.be/uaHH-o83OvQ retrieved 2020-Feb-23.

Summary

In this assignment I modified a pre existing solution to last week's CD inventory program that used dictionaries as the inner data type. The program allowed the user to view current inventory from in memory, load data from a text file, add new data to memory, save data to a text file, and quit the program. This script was saved to Github, an online version control system.

Questions and Comments

Under 'Delete an entry,' I don't fully understand the nested if statement and the logic behind del lstTbl[lstTbl.index(row)]