

Ann Allen

Introduction to Programming – Python

Assignment 5

August 8, 2020

Adding a mapping type and improving code management

Introduction

In Assignment 5 a dictionary type is used in conjunction with a list type, which highlights their similarities and their differences. A dictionary is a Python collection that is a mapping type, rather than being a sequence. When writing the `CDInventory.py` script for Assignment 5, I also practiced more code management with improved error handling and project management using a the GitHub repository.

Dictionaries: A mapping type

To this point, our coursework has used various sequence types for our data: strings, lists, tuples. There is another data type called a dictionary, which is similar to a list, but unlike a list, the dictionary is not a sequence.¹ It is a mapping type that can be used to create a collection of key/value pairs.²

The `CDInventory.py` script uses a 3-key dictionary as each item in a list to story the three values the program needs for each CD entry. Figure 1 shows an example of an inventory list from `CDInventory.py`.

¹ Python.org, <https://docs.python.org/2.3/lib/typesseq.html>, August 10, 2020

² Biesinger, Dirk, Introduction to Programming – Python, `FDN_Py_Module_05.pdf`, August 10, 2020

```

ID, CD Title, Artist
---- No CDs loaded from File Inventory ----

0,Star Wars,John Williams
1,25,Adele

[{'id': 0, 'CD Title': 'Star Wars', 'Artist': 'John Williams'}, {'id': 1, 'CD Title':
'25', 'Artist': 'Adele'}]

[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 1: CDInventory output showing the underlying list-dictionary structure holding CD data

A list variable `lstNewInv` is used to hold new user entries to the CD list. Each entry is a dictionary with three keys – `id`, `CD Title` and `Artist`. This allows the program to access the title or artist of each CD using the *associated* key word rather than relying on a specific index. In fact, a dictionary can be generally referred to as an associative array.³

Dictionaries are similar to lists, but there are a few key differences. Already mentioned is that dictionaries cannot be accessed by index, but rather by key. This difference appears in `CDInventory.py` when the 2D list-dictionary is saved to the file. The 2D list-dictionary needs to be unpacked into a string to be saved to the file (Listing 1).

```

174.     for dictRow in lstNewInv:
175.         strDictRow = ''
176.         for key, value in dictRow.items():
177.             strDictRow += str(value) + ','
178.         # remove last , and add newline
179.         strDictRow = strDictRow[:-1] + '\n'
180.         objFile = open(strFileName, 'a')
181.         objFile.write(strDictRow)
182.         objFile.close()

```

Listing 1: `CDInventory.py` code showing dictionary items being unpacked into a string

Alternatively, when the data from the file is read by the `CDInventory.py` script, the file string needs to be converted into the 2-D list-dictionary format. Using convenient list methods, the file can be read by line as a list and then each comma separated value can be accessed by index and added into a dictionary (Listing 2). This dictionary is then added to the list and represents a single CD entry.

```

236.         lstTbl.clear()
237.         for fileRow in objFile:
238.             lstRow = fileRow.strip().split(sep=',')

```

³ Real Python, <https://realpython.com/python-dicts/>, August 10, 2020

```

239.         dictRow = {'id':int(lstRow[0]), 'CD Title':lstRow[1], \
240.                   'Artist':lstRow[2]}
241.         lstTbl.append(dictRow)

```

Listing 2: CDInventory.py code from saving to file where values are retrieved by list index to add to dictionary subcollection

The code in Listing 2 converts the file contents to a string, then to a list and then to a dictionary. This is made easier by learning during Module 5 that a file object can be iterated line by line.⁴ This convenience lets each line be managed directly, rather than having to read in the entire file as one long string and attempt to separate the lines in the script itself.

Python has other important distinctions between lists and dictionaries. Methods for adding elements and removing elements are both similar and different between the two collections. Knowing how each is edited can help a programmer identify the variable type being used when reading the code. I have found it useful to return to these distinctions multiple times to be sure to understand which options work for each type.

- Removing items:
 - List items are removed by index using the `.remove` method. List items can also be removed using the `del` method (`del list[index]`). Lists also have a `.pop` method, but this method only removes the last item in the list.⁵
 - Dictionary items are removed by key using the `.pop` method. Even though dictionary items are not ordered, the method `.popitem` will remove the most recently added key/value pair from the dictionary. Dictionary items can also be removed using the `del` method (`del dictionary['keyname']`).⁶
- Adding items:
 - List items are added using a variety of methods – `.append`, `.extend` and `.insert`. Each of these changes how the item is placed in the ordered list. Lists also support concatenation using overloaded operators (`list += ['newlistitem']`).
 - Dictionary items are added directly by defining a new key/value for the dictionary – `dictionary['keyname'] = 'value'`. Order is not distinct in a dictionary, so the placement is not a factor. Dictionary also has a convenient method called `.update`, which will merge two dictionaries, adding the unique entries from each dictionary and updating the first dictionary values from any matching keys in the second dictionary passed as a parameter.

List and Dictionary types are both mutable , making them useful resources for managing larger data collections.

⁴ Biesinger, Dirk, Introduction to Programming – Python, FDN_Py_Module_05.pdf, August 10, 2020

⁵ Real Python, <https://realpython.com/python-lists-tuples/>, August 10, 2020

⁶ Real Python, <https://realpython.com/python-dicts/>, August 10, 2020

Code management

Assignment 5 also introduced some more refined code management strategies; separation of concerns, error handling and repositories. I also had some advice from Assignment 4 to work on flattening some deeply nested loops.

Separation of Concerns

Separation of concerns centers around organizing code based on what it is doing, or its “concerns”. “Most programs can be divided into three distinct sections: Data, Processing and Presentation (or Input-Output).”⁷ Assignment 5 makes an early step in this direction by having a beginning section where all key variables are defined (Listing X).

The processing code then follows. Because we are still not using functions, the processing and presentation code is coded together.

Error Handling with Try-Except

Module 5 also introduced the use of Try-Except and more consideration of error handling. Try-Except is a programming construct in Python that lets the developer try a line of code that might cause an exception. If the code fails, rather than ending the script, the developer can manage the exception and keep the script going.⁸

I use the Try-Except construct during file access in `CDInventory.py`. When loading inventory from the saved file, the I would ideally like to open the file in read mode, since the script is only reading data from the file. But the open method in Python will fail with read mode if the file does not exist. To avoid ending the script abruptly in this case, I use the Try-Except construct as seen in Listing X.

```
40.     if strChoice == 'l':
41.         try:
42.             objFile = open(strFileName, 'r')
43.         except:
44.             print('Could not find file to load.\n')
45.             continue
```

Listing 3: try-except used in `CDInventory.py` to catch the case that the file does not yet exist

If the file exists for loading, the file is loaded into the object, `objFile`. If the file does not exist, the code will print an error message to the user that the file could not be found to load. The script then displays the main menu again and the user can continue with a different option.

⁷ Biesinger, Dirk. Introduction to Programming – Python, `FDN_Py_Module_05.pdf`, August 8, 2020

⁸ Biesinger, Dirk, Introduction to Programming – Python, `FDN_Py_Module_05.pdf`, August 8, 2020

Working with repositories - GitHub

Assignment 5 is also saved to a repository. A repository provides structured support for saving versions of code that can be developed synchronously with a group of developers.⁹ Developed by Linus Torvalds, the git program is commonly used to provide this support. As an opensource project, git can be installed by anyone for use by their project.¹⁰

[GitHub.org](https://github.com) is a website that offers an online implementation of git. This site is now owned and supported by Microsoft Corp., but is still the common resource used for git by developers.¹¹ On GitHub, I was able to create a free account and post my script, CDInventory.py, along with this document in a repository called Assignment_05. The document and code can then be reviewed by other with comments and edits can be proposed. If I choose to make an update, I can also update the code there, while preserving a previous copy in case the changes need to be backed out.

Flattening Nested loops

In Assignment 4, our course instructor assistant, Doug Klos, showed some ways that my code could be flattened, meaning that I could reduce the number of times I am nesting loops more than 2 or 3 levels deep. By looking at the suggestions, I found two new code management techniques – doubling up my if statements and using continue.

Using and in if statements

Using the and comparison operator in an if statement can be used to flatten out some nested statements. In one case, I was nesting two if statements. But condensing them and combining them using and, I was able to keep the code flatter for that condition (Listing 4).

```
48.         if objFile != None and objFile == []:
49.             # check if file is empty for good user message
50.             print('Inventory File is empty.\n')
51.             # or proceed to load file contents into lstTbl
```

Listing 4: CDInventory.py code showing an and comparison operator used to flatten nested if statements

Using continue

I also spent time flattening my code for the delete option in the CDInventory.py script (CDInventory.py, Lines 106-164). I used the continue option to return the user to the menu after reaching a conclusion, which helped in flattening my code in this area. Even after working on it, I was still left with a fairly deeply nested loop. In this delete section, I also tried to code it without using the extra boolFound flag, but ultimately came back to using the flag, since the continue option can only bump out one level of a for loop.

⁹ Biesinger, Dirk, Introduction to Programming – Python, FDN_Py_Module_05.pdf, August 8, 2020

¹⁰ Wikipedia – Git, <https://en.wikipedia.org/wiki/Git>, August 8, 2020

¹¹ Biesinger, Dirk, Introduction to Programming – Python, [Module 5, Video 4](#), August 10, 2020

I am anticipating that using functions will also help with code flattening for these types of scenarios. Module 5 provided an introduction to using functions.

CDInventory.py Functioning

CDInventory.py running in Spyder IDE



```
Python 3.7.6 (default, Jan 8 2020, 13:42:34)
Type "copyright", "credits" or "license" for more information.

IPython 7.12.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/annallen/_FDProgramming/Mod_05/Assignment05/
CDInventory.py', wdir='/Users/annallen/_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: a

Enter an ID: 0

Enter the CD's Title: Star Wars

Enter the Artist's Name: John Williams
[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

ID, CD Title, Artist
---- No CDs loaded from saved File Inventory ----

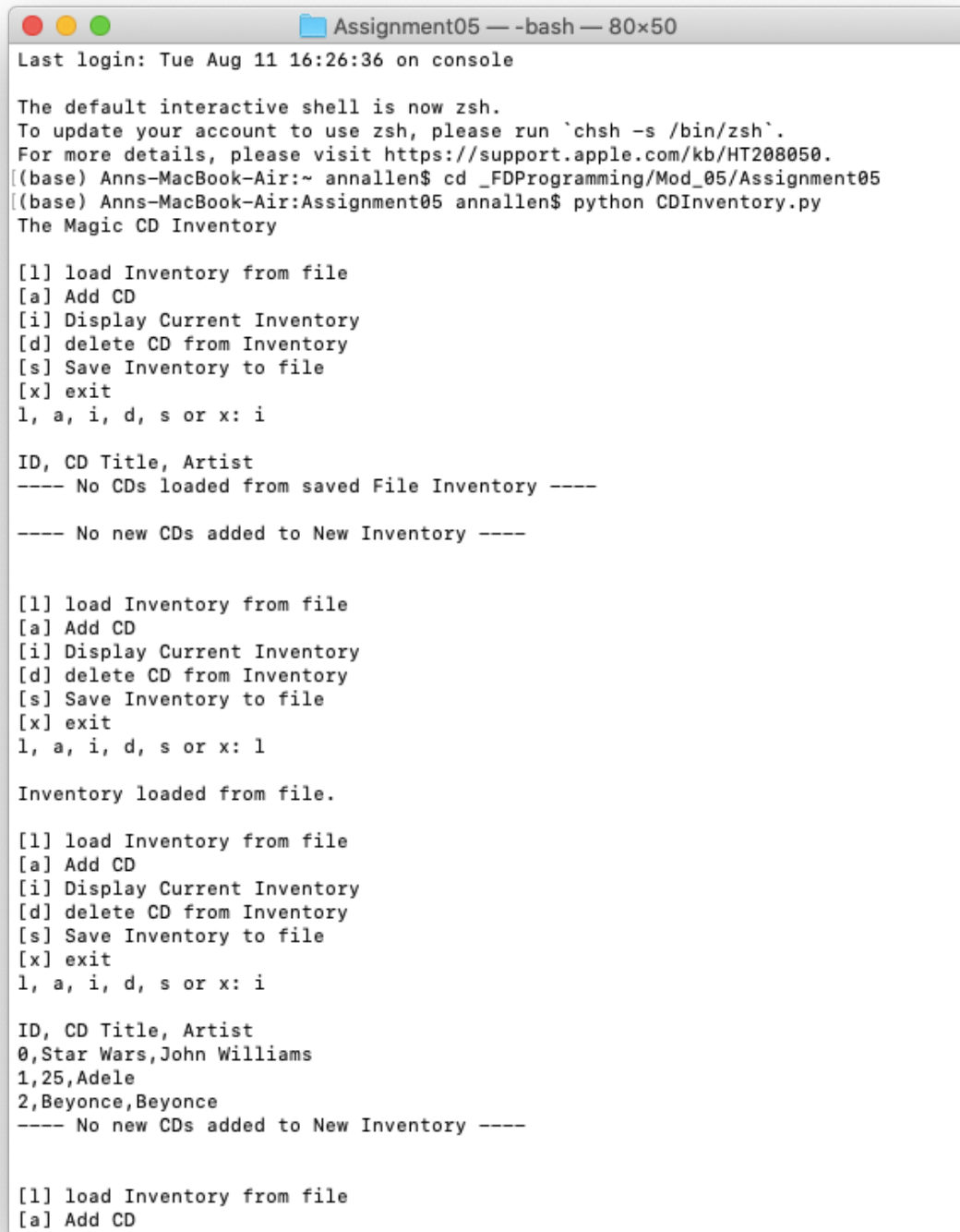
0,Star Wars,John Williams

[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: |
```

Figure 2: CDInventory.py running in the Spyder IDE

CDInventory.py running in MacOS Bash terminal



```
Assignment05 — -bash — 80x50
Last login: Tue Aug 11 16:26:36 on console

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
[(base) Anns-MacBook-Air:~ annallen$ cd _FDProgramming/Mod_05/Assignment05
[(base) Anns-MacBook-Air:Assignment05 annallen$ 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: i

ID, CD Title, Artist
---- No CDs loaded from saved File Inventory ----

---- No new CDs added to New 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

Inventory loaded from file.

[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

ID, CD Title, Artist
0,Star Wars,John Williams
1,25,Adele
2,Beyonce,Beyonce
---- No new CDs added to New Inventory ----

[l] load Inventory from file
[a] Add CD
```

Figure 3: CDInventory.py running in the MacOS Bash terminal

Summary

Assignment 5 left me feeling comfortable with dictionaries and even more proficient with lists and file access. By revisiting the same program used for Assignment 4, I was able to focus more on new ways of working with lists and dictionaries beyond the construct of the menu and perfecting the display. I also spent more time looking at ways to clean up the code and flatten more deeply nested areas. I now look forward to seeing classmate's implementations in GitHub and hearing any feedback on my own work.

Appendix

CDInventory.py

```
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. # AAllen, 2020-Aug-7, replace lists with
7. #         dictionaries inside
8. #         main list, lstTbl
9. # AAllen, 2020-Aug-7, added code to save to
10. #         file
11. # AAllen, 2020-Aug-7, added loading from file
12. # AAllen, 2020-Aug-7, added second table list to
13. #         manage loaded and new data
14. #         separately
15. # AAllen, 2020-Aug-9, removed deeply nested loops
16. #-----#
17.
18. # Declare variables
19.
20. strChoice = '' # User input
21. lstTbl = [] # list of dicts to hold saved data
22. lstNewInv = [] # list of new dicts added by user since last save
23. dictRow = {} # dictionary data row: ID, CD Title, Artist
24. strFileName = 'CDInventory.txt' # data storage file
25. objFile = None # file object
26.
27. # Get user Input
28. print('The Magic CD Inventory\n')
29. while True:
30.     # 1. Display menu allowing the user to choose:
31.     print('[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
32.     print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit')
33.     strChoice = input('l, a, i, d, s or x: ').lower() # convert choice to lower case at t
34.     print()
35.
36.     if strChoice == 'x':
37.         # 5. Exit the program if the user chooses so
38.         break
39.
40.     if strChoice == 'l':
41.         try:
42.             objFile = open(strFileName, 'r')
43.             except:
```

```

44.         print('Could not find file to load.\n')
45.         continue
46.
47.         # if inventory file was loaded successfully
48.     if objFile != None and objFile == []:
49.         # check if file is empty for good user message
50.         print('Inventory File is empty.\n')
51.         # or proceed to load file contents into lstTbl
52.     elif objFile != None:
53.         # start with empty table in case user has already loaded
54.         lstTbl.clear()
55.         for fileRow in objFile:
56.             lstRow = fileRow.strip().split(sep=',')
57.             dictRow = {'id':int(lstRow[0]), 'CD Title':lstRow[1], \
58.                       'Artist':lstRow[2]}
59.             lstTbl.append(dictRow)
60.         print('Inventory loaded from file.\n')
61.     else:
62.         print("----- No CD Inventory has been saved yet.-----\n")
63.     objFile.close()
64.
65.     elif strChoice == 'a': # no elif necessary, as this code is only reached if strChoice
        is not 'exit'
66.         # 2. Add data to the table (2d-list) each time the user wants to add data
67.         strID = input('Enter an ID: ')
68.         strTitle = input('Enter the CD\'s Title: ')
69.         strArtist = input('Enter the Artist\'s Name: ')
70.         intID = int(strID)
71.         dictRow = {'id':intID, 'CD Title':strTitle, 'Artist':strArtist}
72.         lstNewInv.append(dictRow)
73.
74.     elif strChoice == 'i':
75.         # 3. Display the current data to the user each time the user wants to display the
        data
76.
77.         # display header
78.         print('ID, CD Title, Artist')
79.
80.         # display items from file if user has loaded them
81.         # if the CD Inventory is empty, confirm that to user
82.         if lstTbl == []:
83.             print('----- No CDs loaded from saved File Inventory ----- \n')
84.             # for each row from the file, display the inventory
85.             for dictRow in lstTbl:
86.                 # add error catching in case key names incorrect
87.                 try:
88.                     print(dictRow['id'], dictRow['CD Title'], dictRow['Artist'], \
89.                           sep=',', end='\n')
90.                 except:
91.                     print('! Error printing row from CD Inventory File !\n')
92.
93.             # display items the user has added, but not saved in this session
94.             # if new CD Inventory is empty, confirm that to user
95.             if lstNewInv == []:
96.                 print('----- No new CDs added to New Inventory ----- \n')
97.             for dictRow in lstNewInv:
98.                 # add error catching in case key names incorrect
99.                 try:
100.                     print(dictRow['id'], dictRow['CD Title'], dictRow['Artist'], \
101.                           sep=',', end='\n')
102.                 except:

```

```

103.         print('! Error printing row in new CD Inventory\n')
104.     print()
105.
106.     elif strChoice == 'd':
107.         # ask user which CD ID they want to delete (or if they want to cancel)
108.         strDelCD = input('Which CD ID would you like to'+
109.             ' delete? (enter x to cancel): ')
110.
111.         # if user wants to cancel, return to menu
112.         if strDelCD.lower() == 'x':
113.             continue
114.
115.         # or check for valid entry and find entry to delete
116.         try:
117.             intDelCD = int(strDelCD)
118.             # if user entry is not an integer, exit to menu
119.         except:
120.             print('Not a valid CD ID entry\n')
121.             continue
122.
123.         # if user entered a number for CD ID
124.         if intDelCD:
125.             boolFound = False
126.
127.             # Look in all the rows in the new list inventory first
128.             for row in lstNewInv:
129.                 if row.get('id') == intDelCD:
130.                     lstNewInv.remove(row)
131.                     boolFound = True
132.                     print('CD entry with ID=', strDelCD, 'was deleted.\n')
133.                     continue
134.
135.             # if ID row was removed, return to main menu
136.             if boolFound == True:
137.                 continue
138.
139.             # if CD ID has not been found, look in the loaded file table
140.             # and delete from file
141.             for row in lstTbl:
142.                 if row.get('id') == intDelCD:
143.                     lstTbl.remove(row)
144.                     boolFound = True
145.                     # replace file contents with new lstTbl
146.                     for dictRow in lstTbl:
147.                         strDictRow = ''
148.                         for key, value in dictRow.items():
149.                             strDictRow += str(value) + ','
150.                         # remove last , and add newline
151.                         strDictRow = strDictRow[:-1] + '\n'
152.                         objFile = open(strFileName, 'w')
153.                         objFile.write(strDictRow)
154.                         objFile.close()
155.
156.                     print('CD entry with ID= ', strDelCD, ' was',
157.                         ' deleted from the file.\n')
158.                     continue
159.             if boolFound == True:
160.                 continue
161.
162.             # if ID was not found, code reaches here and shows
163.             # message that ID was not found

```

```

164.         print('Could not find CD with ID= ', strDelCD, '\n')
165.
166.     elif strChoice == 's':
167.         # 4. Save the data to a text file CDInventory.txt if the user chooses so
168.         # if there is no new inventory, do not access file
169.         if lstNewInv == []:
170.             print('\n---- No new inventory to save.----\n')
171.
172.         # open file and save new inventory if there are additions
173.         else:
174.             for dictRow in lstNewInv:
175.                 strDictRow = ''
176.                 for key, value in dictRow.items():
177.                     strDictRow += str(value) + ','
178.                 # remove last , and add newline
179.                 strDictRow = strDictRow[:-1] + '\n'
180.                 objFile = open(strFileName, 'a')
181.                 objFile.write(strDictRow)
182.                 objFile.close()
183.             print('New inventory additions saved to file.\n')
184.
185.             # add saved inventory to file inventory, user will expect this
186.             #content for display option or delete option
187.             lstTbl += lstNewInv
188.
189.             # reset new inventory to empty
190.             lstNewInv.clear()
191.         else:
192.             print('Please choose either l, a, i, d, s or x\n')

```