Evan Malina

11/21/2021

Foundations of Programming Python

Assignment 6

# Introduction

For this assignment, I modified an existing script to add the ability to input CDs, delete CDs from inventory and to write the inventory to file. Lastly, I unloaded the code to my Assignment\_06 repository on gitlab. This assignment was mainly focused on functions and classes. The trickiest part for me was variables. I think since I copied the code from the *actual* program, I started with the variables that existed throughout the program, and then I got a bit wrapped up in fixing them. The code worked fine with the shadow variables, and I think that has something to do with, in particular the variable lstTbl, being reference type.

# Code

## Github

Github link to my code and knowledge document here;

<https://github.com/ewmalina/Assignment_06>

## Input CD code and Add Data

It was a little tricky for me to keep the input output and processing sections together mentally for the input CD in particular. The first step was adding *add\_data()* to the I/O section. This one was one of the simplest for me to conceptualize in my head. I changed the variable names, and returned a tuple with the 3 variables (ID, Title, Artist). I used the fast method of ID, Title, Artist = fun() to retrieve them. This is super straightforward for me 😊.

The part that I am still somewhat trying to grasp is how for the next section of processing this data (input\_CD()) actually returns the table. I processed the data, and returned the *local* table, but I don’t understand what the table variable is called once it returns to the code outside of the function. As I mentioned, I wrote the code initially using the initial provided variable names and it still worked. I changed them all and it still seems to work. Also, I didn’t actually put a variable = function to retrieve the return – so where does the return go?

## Delete CD code

This section was fairly simple for me. I moved the provided code into the DataProcessor section, and chose 2 inputs: the ID to delete, and the table. I renamed these variables, and returned the table.

## Write inventory to file

This section of code was more of the same. Add “@staticmethod” prior to the function. Define the new function, type out the doc string, copy and paste the code and change the variable inputs to be unique to the function.

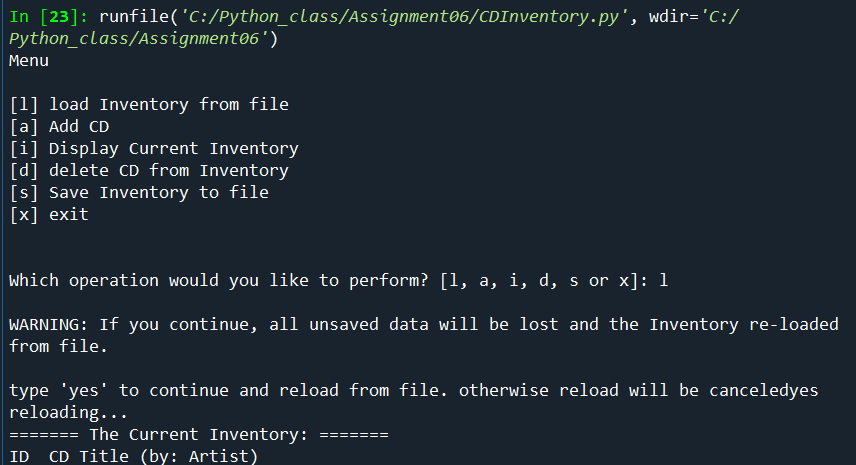


Figure - Spyder Script 1

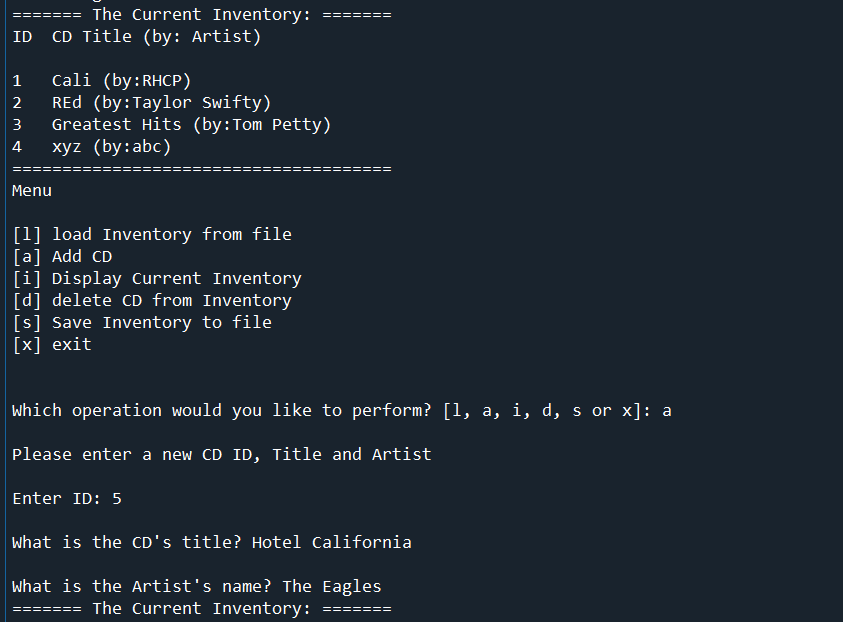


Figure - Spyder Script 2

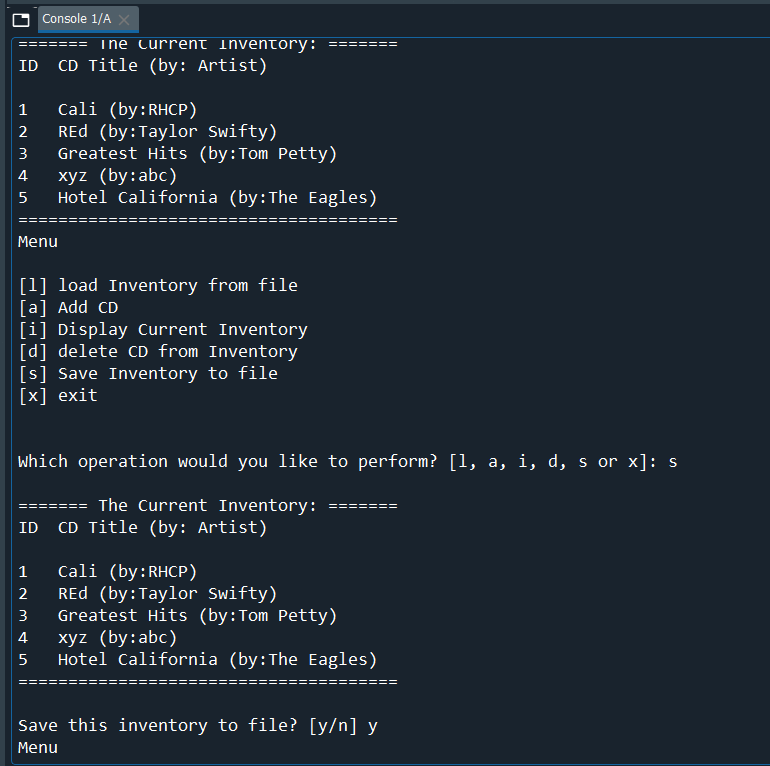


Figure 3 - Spyder script run part 3

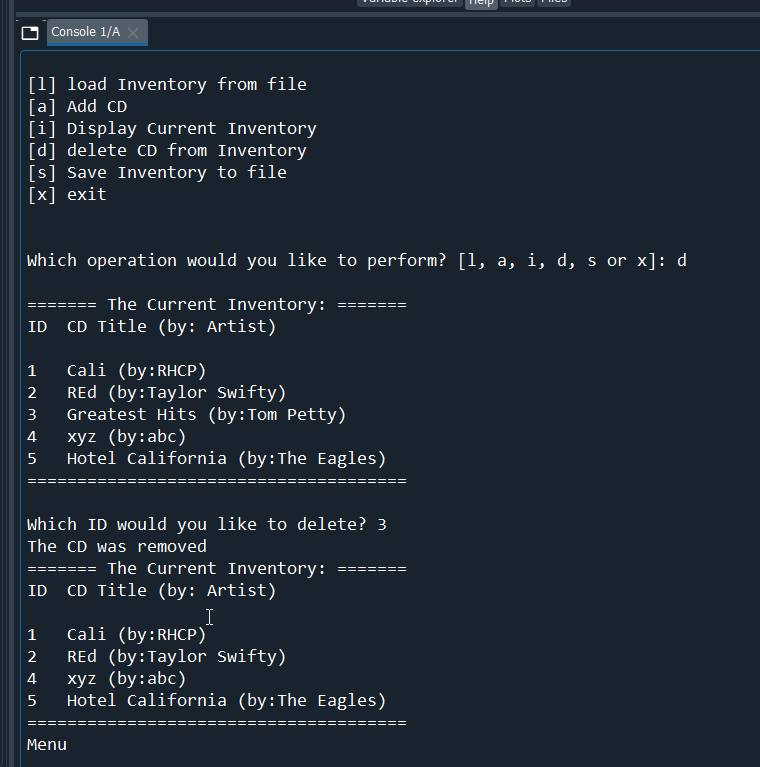


Figure 4 - Spyder script run part 4

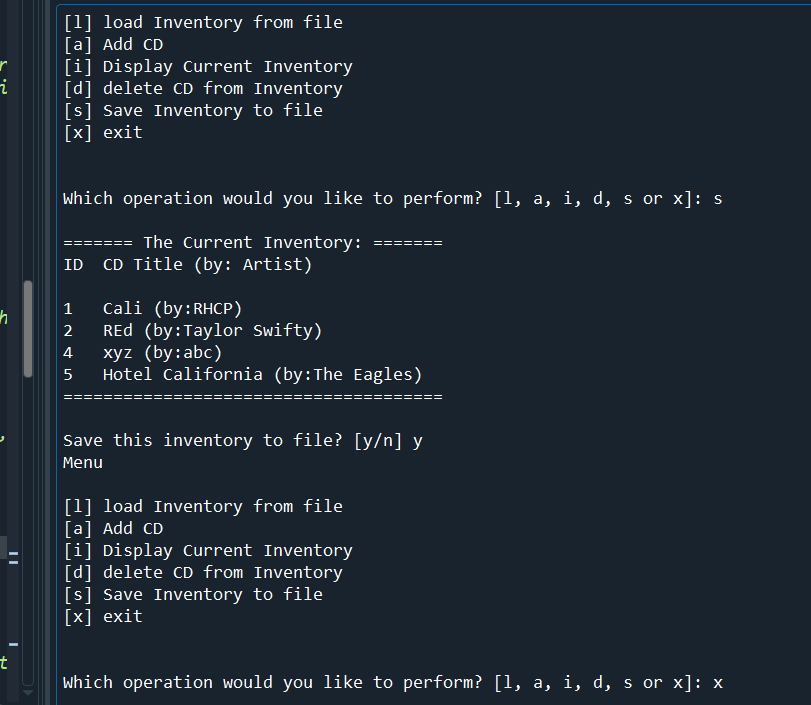


Figure 5 - Spyder script run part 5

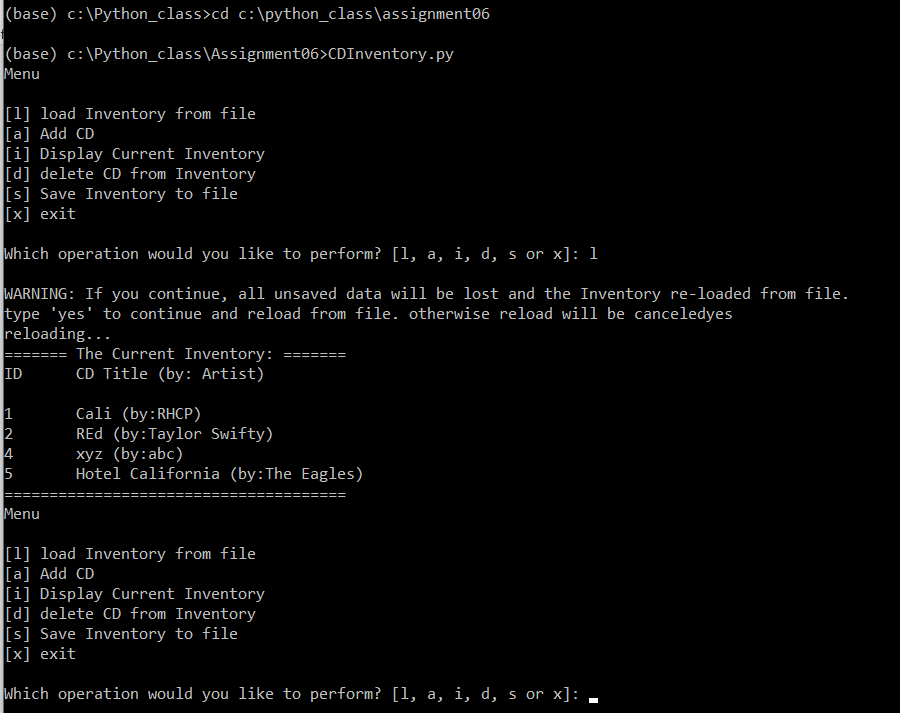


Figure 6 - shell run of script

1. *#------------------------------------------#*
2. *# Title: CDInventory.py*
3. *# Desc: Working with classes and functions.*
4. *# Change Log: (Who, When, What)*
5. ***# DBiesinger, 2030-Jan-01, Created File***
6. *# TODO Evan Malina, 2021-Nov-21, Updated file to add, delete, write CD data*
7. *#------------------------------------------#*
9. *# -- DATA -- #*
10. **strChoice = '' *# User input***
11. lstTbl = [] *# list of lists to hold data*
12. dicRow = {} *# list of data row*
13. strFileName = 'CDInventory.txt' *# data storage file*
14. objFile = None *# file object*

17. *# -- PROCESSING -- #*
18. **class** DataProcessor:
19. *# TODOne add functions for processing here*
20. **"""Functions I added for processing CD data into a table"""**
21. @staticmethod
22. **def** input\_CD(strNum, strCDTitle, str\_Artist, table):
23. """Function to add inputs into table
25. **Args:**
26. strNum (string): First input is the ID number
27. strCDTitle (string): Second input is album Title
28. str\_Artist (string): 3rd input is album artist (TYPO)
29. table (list): current inventory list of dictionaries
31. Returns:
32. table (list): a list of dictionaries of CD entries
33. """
34. intID = int(strNum) *#convert string to integer*
35. **dicRow = {'ID': intID, 'Title': strCDTitle, 'Artist': str\_Artist}**
36. table.append(dicRow)
37. **return** table
39. @staticmethod
40. **def delete\_CD(delNum, table):**
41. """Function to find and delete a specific ID from inventory
42. Args:
43. delNum (int): ID to delete from inventory
44. table (list): cd inventory table
46. Returns:
47. table (list): cd inventory table
49. """
50. **intRowNr = -1**
51. blnCDRemoved = False
52. **for** row **in** table:
53. intRowNr += 1
54. **if** row['ID'] == delNum:
55. **del table[intRowNr]**
56. blnCDRemoved = True
57. **break**
58. **if** blnCDRemoved:
59. **print**('The CD was removed')
60. **else:**
61. **print**('Could not find this CD!')
62. **return** table

65. **class FileProcessor:**
66. """Processing the data to and from text file"""
68. @staticmethod
69. **def** read\_file(file\_name, table):
70. **"""Function to manage data ingestion from file to a list of dictionaries**
72. Reads the data from file identified by file\_name into a 2D table
73. (list of dicts) table one line in the file represents one dictionary row in table.
75. **Args:**
76. file\_name (string): name of file used to read the data from
77. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
79. Returns:
80. **None.**
81. """
82. table.clear() *# this clears existing data and allows to load data from file*
83. objFile = open(file\_name, 'r')
84. **for** line **in** objFile:
85. **data = line.strip().split(',')**
86. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
87. table.append(dicRow)
88. objFile.close()
90. **@staticmethod**
91. **def** write\_file(file\_name, table):
92. *# TODone Add code here*
93. """Function to save the CD to file
94. Args:
95. **file\_name (str): file name for saving**
96. table (list): current inventory of CDs. A list of dictionaries
98. Returns: none
99. """
100. **objFile = open(file\_name, 'w')**
101. **for** row **in** table:
102. lstValues = list(row.values())
103. lstValues[0] = str(lstValues[0])
104. objFile.write(','.join(lstValues) + '**\n**')
105. **objFile.close()**

108. *# -- PRESENTATION (Input/Output) -- #*
110. **class IO:**
111. """Handling Input / Output"""
113. @staticmethod
114. **def** print\_menu():
115. **"""Displays a menu of choices to the user**
117. Args:
118. None.
120. **Returns:**
121. None.
122. """
124. **print**('Menu**\n\n**[l] load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')
125. **print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')**
127. @staticmethod
128. **def** menu\_choice():
129. """Gets user input for menu selection
131. Args:
132. None.
134. Returns:
135. **choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x**
137. """
138. choice = ' '
139. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
140. **choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()**
141. **print**() *# Add extra space for layout*
142. **return** choice
144. @staticmethod
145. **def show\_inventory(table):**
146. """Displays current inventory table

149. Args:
150. **table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.**
152. Returns:
153. None.
155. **"""**
156. **print**('======= The Current Inventory: =======')
157. **print**('ID**\t**CD Title (by: Artist)**\n**')
158. **for** row **in** table:
159. **print**('{}**\t**{} (by:{})'.format(\*row.values()))
160. **print('======================================')**
162. *# TODone add I/O functions as needed*
163. @staticmethod
164. **def** add\_data():
165. **"""requests user to input new CD information**
167. Args: none
169. Returns: tuple of 3;
170. **var1 (int): ID number**
171. var2 (str): CD title
172. var3 (str): CD artist
173. """
174. **print**('Please enter a new CD ID, Title and Artist')
175. **var1 = input('Enter ID: ').strip()**
176. var2 = input('What is the CD**\'**s title? ').strip()
177. var3 = input('What is the Artist**\'**s name? ').strip()
178. **return**(var1, var2, var3)


182. *# 1. When program starts, read in the currently saved Inventory*
183. FileProcessor.read\_file(strFileName, lstTbl)
185. ***# 2. start main loop***
186. **while** True:
187. *# 2.1 Display Menu to user and get choice*
188. IO.print\_menu()
189. strChoice = IO.menu\_choice()
191. *# 3. Process menu selection*
192. *# 3.1 process exit first*
193. **if** strChoice == 'x':
194. **break**
195. ***# 3.2 process load inventory***
196. **if** strChoice == 'l':
197. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
198. strYesNo = input('type **\'**yes**\'** to continue and reload from file. otherwise reload will be canceled')
199. **if** strYesNo.lower() == 'yes':
200. **print('reloading...')**
201. FileProcessor.read\_file(strFileName, lstTbl)
202. IO.show\_inventory(lstTbl)
203. **else**:
204. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
205. **IO.show\_inventory(lstTbl)**
206. **continue** *# start loop back at top.*
207. *# 3.3 process add a CD*
208. **elif** strChoice == 'a':
209. *# 3.3.1 Ask user for new ID, CD Title and Artist*
210. ***# TODOne move IO code into function***
211. strID, strTitle, stArtist = IO.add\_data()
212. *# 3.3.2 Add item to the table*
213. *# TODOne move processing code into function*
214. DataProcessor.input\_CD(strID, strTitle, stArtist, lstTbl)
215. **IO.show\_inventory(lstTbl)**
216. **continue** *# start loop back at top.*
217. *# 3.4 process display current inventory*
218. **elif** strChoice == 'i':
219. IO.show\_inventory(lstTbl)
220. **continue *# start loop back at top.***
221. *# 3.5 process delete a CD*
222. **elif** strChoice == 'd':
223. *# 3.5.1 get Userinput for which CD to delete*
224. *# 3.5.1.1 display Inventory to user*
225. **IO.show\_inventory(lstTbl)**
226. *# 3.5.1.2 ask user which ID to remove*
227. intIDDel = int(input('Which ID would you like to delete? ').strip())
228. *# 3.5.2 search thru table and delete CD*
229. *# TODone move processing code into function*
230. **DataProcessor.delete\_CD(intIDDel, lstTbl)**
231. IO.show\_inventory(lstTbl)
232. **continue** *# start loop back at top.*
233. *# 3.6 process save inventory to file*
234. **elif** strChoice == 's':
235. ***# 3.6.1 Display current inventory and ask user for confirmation to save***
236. IO.show\_inventory(lstTbl)
237. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
238. *# 3.6.2 Process choice*
239. **if** strYesNo == 'y':
240. ***# 3.6.2.1 save data***
241. *# TODO move processing code into function*
242. FileProcessor.write\_file(strFileName, lstTbl)
243. **else**:
244. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
245. **continue *# start loop back at top.***
246. *# 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:*
247. **else**:
248. **print**('General Error')

Listing 1 - CDInventory.py script

# Summary

As previously mentioned, the thing that primarily caught me up was what exactly was going on with the variables as they are passed back and forth through functions. The class part seems like very little change to me, and I like the organization of it. I’m excited to learn what the “staticmethod” is, and what the alternatives are. I thoroughly enjoyed reading the code in this assignment.