Vernon Jackson

Aug 15, 2021

IT FDN 110

Assignment 06

# Modify Program to Store CD Inventory Data

# Introduction

In this document I will be explaining how I added code to the CD Inventory Starter program for assignment 06, that modifies the script as required to add processing and I/O functions. Areas of focus for this document include 1) briefly covering the fundamentals that I learned in this module 2) the steps I took to organize and create the script and 3) a summary of my experience in creating this program.

# Coding Fundamentals

Within this assignment, I learned several new fundamentals of coding, including functions, return values, and classes. I will briefly summarize my own definitions of each used in the assignment in order to substantiate my learning.

**Functions:** Functions are groupings of code that take in a number of values/arguments, that are in a specific order, and processes these values, and then returns another value or completes a task.

**Return Values:** Return values are the result of adding the return statement to a function, which returns a value or assigned to a variable as an object after processing the data. Returning as a variable allows you to use the value throughout the program, instead of having to call the function repeatedly.

**Classes:** Just as functions are groupings of code, a class is a grouping of functions or objects. Classes allow you to organize the functions or objects into groups that make sense, according to what the function does.

# Updating the Program

In order to create the script, I first began with the *Assignment06\_Starter* script and added processing and I/O functions. The main coding fundamentals utilized in this assignment are functions to process and retrieve/send data, return values to obtain new values after being processes, and classes for organizing the functions. The following functionalities were modified for use of separate functions:

**Adding Inventory:** In order to utilize functions within this script for adding inventory, I needed to first create an *add\_choice* function within the *I/O* class, which gets user input for new ID, CD Title, and Artist. I used the existing code that was in 3.3.1 and transferred the code to the separate function. I then created another *add\_inventory* function within the *DataProcessor* class, which is used to take the values from the variable returned from the *add\_choice* function, maps the values to a dictionary, then appends the dictionary to the *lstTbl* list of dictionaries within the program. I used the existing code that was in 3.3.2 and moved the code to the separate *add\_inventory* function.

**Deleting Inventory:** In order to utilize functions within this script for deleting inventory, I needed to first create an *delete\_inventory* function within the *DataProcessor* class. This function references the integer values from user input assigned to the *intIDDel* variable, searches for the element containing the appropriate key, value, and deletes the element from *lstTbl* list of dictionaries. I used the existing code that was in 3.5.2 and transferred the code to the separate function. I did not create a separate function for handling input/output, as this is done directly in the main program.

**Saving Data**: In order to utilize functions within this script for saving inventory to a text file, I used the existing code that was in 3.6.2.1 and transferred the code to the separate *write\_file* function in the DataProcessor class. I then removed the *pass* statement from the existing function, as this was not needed. I did not create a separate function for handling input/output, as this is done directly in the main program.

To test this script, I ran this as a Python script through Spyder IDE and Anaconda Prompt. Executing the script through both, showed that it had the intended functionality of using functions for adding inventory, deleting inventory, and saving the data to a text file. See appendix Listing CDInventory.pyfor highlighted code and follow this link to view the script on GitHub: <https://github.com/Veejster/Assignment_06.git>.

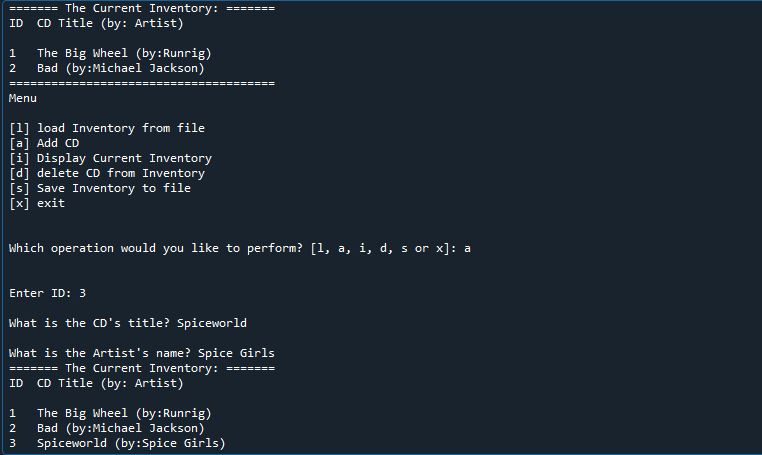


Figure 1 - Showing the program being executed through Spyder.

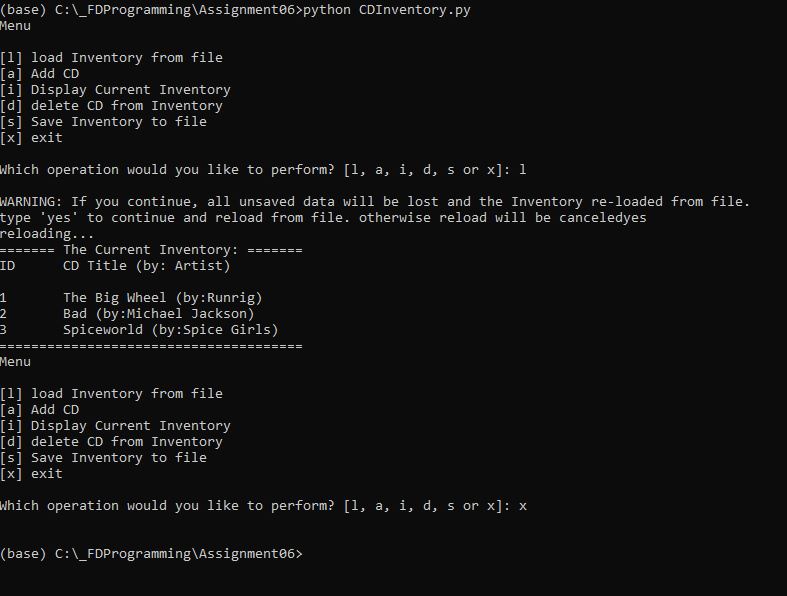


Figure 2 - Shows script being executed through Anaconda prompt.

# Summary

In this assignment we learned about functions, return values, and classes. Since I was already familiar with using functions and return values in my previous assignments, adding the functions was not a challenge for me. In the previous class session, I learned that the names of variables that are passed as arguments within a function did not need be the same. In the previous assignments, I had assumed that these did need to be the same and was very worried that my code would not execute as intended. When reviewing classes in other online resources, I can see that there is more functionality that can be utilized and I will explore these separately in my studies. Additionally, once you understand the basic formatting of someone else’s code, I find it very easy to work with when adding separate functions, instead of writing new code to be added to a function. I assume that this will be harder though, when you are faced with situations where the standard naming conventions and organization are not clearly notated.

# Appendix

## Listing CDInventory.py

1. *#------------------------------------------#*
2. *# Title: Assignment06\_Starter.py*
3. *# Desc: Working with classes and functions.*
4. *# Change Log: (Who, When, What)*
5. ***# DBiesinger, 2030-Jan-01, Created File***
6. *#* ***VJackson, 2021-Aug-15, Updated File to***
7. ***# add functions and code***
8. *#------------------------------------------#*
10. ***# -- DATA -- #***
11. strChoice = '' *# User input*
12. lstTbl = [] *# list of lists to hold data*
13. dicRow = {} *# list of data row*
14. strFileName = 'CDInventory.txt' *# data storage file*
15. **objFile = None *# file object***

18. *# -- PROCESSING -- #*
19. **class** DataProcessor:
20. **"""Processing data to lstTbl"""**
22. @staticmethod
23. **def** add\_inventory(addInput, table):
24. """Function used to add element to a list of dictionaries
26. References the values the return tuple from a function (IO.add\_CD),appends it to a
27. dictionary using defined keys (dicRow),then appends the dictionary to a list of dictionaries (lstTbl).
29. Args:
30. **addInput (Tuple): Tuple containing input values from a fucntion**
31. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
33. Returns:
34. None.
35. **"""**
36. intID = int(addInput[0])
37. dicRow = {'ID': intID, 'Title': addInput[1], 'Artist': addInput[2]}
38. lstTbl.append(dicRow)
39. IO.show\_inventory(lstTbl)
41. @staticmethod
42. **def** delete\_inventory(delID, table):
43. """Function used to delete an element from a list of dictionaries
45. **References the integer values from user input in variable, searches for**
46. element containing key,value, and deletes element from list of dictionaries (lstTbl).
48. Args:
49. delID (value): integer value from user input
50. **table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.**
52. Returns:
53. None.
54. """
55. **intRowNr = -1**
56. blnCDRemoved = False
57. **for** row **in** lstTbl:
58. intRowNr += 1
59. **if** row['ID'] == intIDDel:
60. **del lstTbl[intRowNr]**
61. blnCDRemoved = True
62. **break**
63. **if** blnCDRemoved:
64. **print**('The CD was removed')
65. **else:**
66. **print**('Could not find this CD!')
67. IO.show\_inventory(lstTbl)

70. **class FileProcessor:**
71. """Processing the data to and from text file"""
73. @staticmethod
74. **def** read\_file(file\_name, table):
75. **"""Function to manage data ingestion from file to a list of dictionaries**
77. Reads the data from file identified by file\_name into a 2D table
78. (list of dicts) table one line in the file represents one dictionary row in table.
80. **Args:**
81. file\_name (string): name of file used to read the data from
82. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
84. Returns:
85. **None.**
86. """
87. table.clear() *# this clears existing data and allows to load data from file*
88. objFile = open(file\_name, 'r')
89. **for** line **in** objFile:
90. **data = line.strip().split(',')**
91. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
92. table.append(dicRow)
93. objFile.close()
95. **@staticmethod**
96. **def** write\_file(file\_name, table):
97. """Function to write data in lstTbl to a text file
99. Writes the data from a 2D table into a text file identified by file\_name
100. **(list of dicts) table one line in the file represents one dictionary row in table.**
102. Args:
103. file\_name (string): name of file used to read the data from
104. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
106. Returns:
107. None.
108. """
109. objFile = open(strFileName, 'w')
110. **for row in lstTbl:**
111. lstValues = list(row.values())
112. lstValues[0] = str(lstValues[0])
113. objFile.write(','.join(lstValues) + '**\n**')
114. objFile.close()


118. *# -- PRESENTATION (Input/Output) -- #*
120. **class IO:**
121. """Handling Input / Output"""
123. @staticmethod
124. **def** print\_menu():
125. **"""Displays a menu of choices to the user.**
127. Args:
128. None.
130. **Returns:**
131. None.
132. """
134. **print**('Menu**\n\n**[l] load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')
135. **print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')**
137. @staticmethod
138. **def** menu\_choice():
139. """Gets user input for menu selection.
141. Args:
142. None.
144. Returns:
145. **choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x**
147. """
148. choice = ' '
149. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
150. **choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()**
151. **print**() *# Add extra space for layout*
152. **return** choice
154. @staticmethod
155. **def show\_inventory(table):**
156. """Displays current inventory table.
158. Args:
159. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
161. Returns:
162. None.
164. """
165. **print('======= The Current Inventory: =======')**
166. **print**('ID**\t**CD Title (by: Artist)**\n**')
167. **for** row **in** table:
168. **print**('{}**\t**{} (by:{})'.format(\*row.values()))
169. **print**('======================================')
171. @staticmethod
172. **def** add\_choice():
173. """Gets user input for new ID, CD Title, and Artist to add to inventory
175. **Args:**
176. None.
178. Returns:
179. Tuple: objects with user input values for new ID, CD Title, and Artist
181. """
182. strID = input('Enter ID: ').strip()
183. strTitle = input('What is the CD**\'**s title? ').strip()
184. stArtist = input('What is the Artist**\'**s name? ').strip()
185. **return (strID, strTitle, stArtist)**

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