Sravani Musunuri

August 14th , 2022

IT FDN 110 A Su 22: Foundations of Programming: Python

Assignment 06

Separation of Concerns into Functions

# Introduction

This assignment provides all the knowledge required to write my CD Inventory Python script to convert few lines of code into functions and few functions clubbed into a Classes. By watching module videos and reading book chapter gave clear understanding of Parameters & Arguments, Functions and Classes, Global & Local variables, shadowing, use functions to organize code and Separation of concerns And writing Docstrings are each function. I used Spyder to execute all my lab programs.

# Course Work

Assignment 05 challenged me to convert inner data structures to a 2D List of Dictionaries. Continuing the same example CD Inventory in Assignment 06, the code needs to be modified into several Functions as needed. Few lines of code are converted to a function and few functions are clubbed into a Class.

I started updating the Starter python script by adding the header. And went on moving the few lines of code into functions as suggested. I created two functions add\_item() and delete\_item() in DataProcessor Class with required arguments for each function. Also written the DocString for each function as mentioned in the Assignment. I followed the naming convention for function as recommended. Later these functions are called while adding and deleting an item from the 2D list of dictionaries. A function is called by <<ClassName>>.<<functionName>>, like DataProcessor.add\_item(strID,strTitle,strArtist).

Similarly write\_file() and user\_input() functions are created in FileProcessor and IO classes respectively. Explanatory Header is added at the beginning of each function - DocString as recommended and followed the pattern.

"""Description

Args:

Returns:

"""

# Code dump

Here is full code of the program to store CD inventory details to a list of dictionaries. I updated the header details and also provided the proper comments which make sense to other programmers.

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. # Sravani Musunuri, 2020-Aug-14, Updating the file
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. """Processing the data to add and delete an item from the List of Dictionaries"""
21. @staticmethod
22. **def** add\_item(strID,strTitle,strArtist):
23. """Function to manage user input ingestion to a list of dictionaries
25. User Input is added to a 2D table.
26. (list of dicts) table one line in the file represents one dictionary row in table.
28. Args:
29. strID (string): ID of CD Inventory
30. strTitle (string): Title of CD Inventory
31. strArtist (string): Artist of the CD Inventory
33. Returns:
34. None.
35. """
37. intID **=** int(strID)
38. dicRow **=** {'ID': intID, 'Title': strTitle, 'Artist': strArtist}
39. lstTbl.append(dicRow)
41. @staticmethod
42. **def** delete\_item(lstTbl,intIDDel):
43. """Function to manage deletion of an item from a list of dictionaries
45. Delete an item by parsing thru the 2D table
46. (list of dicts) table one line in the file represents one dictionary row in table.
48. Args:
49. lstTbl (list of dictionaries): List of CD inventories
50. intIDDel (int): ID of CD Inventory
52. Returns:
53. 2D table (list of dict).
55. """
57. intRowNr **=** **-**1
58. blnCDRemoved **=** False
59. **for** row **in** lstTbl:
60. intRowNr **+=** 1
61. **if** row['ID'] **==** intIDDel:
62. **del** lstTbl[intRowNr]
63. blnCDRemoved **=** True
64. **break**
65. **if** blnCDRemoved:
66. print('The CD was removed')
67. **else**:
68. print('Could not find this CD!')
70. **return** lstTbl
72. **pass**

75. **class** FileProcessor:
76. """Processing the data to and from text file"""
78. @staticmethod
79. **def** read\_file(file\_name, table):
80. """Function to manage data ingestion from file to a list of dictionaries
82. Reads the data from file identified by file\_name into a 2D table
83. (list of dicts) table one line in the file represents one dictionary row in table.
85. Args:
86. file\_name (string): name of file used to read the data from
87. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
89. Returns:
90. None.
91. """
92. table.clear()  # this clears existing data and allows to load data from file
93. objFile **=** open(file\_name, 'r')
94. **for** line **in** objFile:
95. data **=** line.strip().split(',')
96. dicRow **=** {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
97. table.append(dicRow)
98. objFile.close()
100. @staticmethod
101. **def** write\_file(file\_name, table):
102. """Function to manage data ingestion from List of dictionaries to a file.
104. Reads the data from 2D table and saves to a file identified by file\_name.
105. (list of dicts) table one line in the file represents one dictionary row in table.
107. Args:
108. file\_name (string): name of file used to read the data from
109. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
111. Returns:
112. None.
113. """
115. objFile **=** open(file\_name, 'w')
116. **for** row **in** table:
117. lstValues **=** list(row.values())
118. lstValues[0] **=** str(lstValues[0])
119. objFile.write(','.join(lstValues) **+** '\n')
120. objFile.close()
121. **pass**

124. # -- PRESENTATION (Input/Output) -- #
126. **class** IO:
127. """Handling Input / Output"""
129. @staticmethod
130. **def** print\_menu():
131. """Displays a menu of choices to the user
133. Args:
134. None.
136. Returns:
137. None.
138. """
140. print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
141. print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
143. @staticmethod
144. **def** menu\_choice():
145. """Gets user input for menu selection
147. Args:
148. None.
150. Returns:
151. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
153. """
154. choice **=** ' '
155. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
156. choice **=** input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
157. print()  # Add extra space for layout
158. **return** choice
160. @staticmethod
161. **def** user\_input():
162. """Gets user input for CD Inventory
164. Args:
165. None.
167. Returns:
168. strID (string): ID of CD Inventory
169. strTitle (string): Title of CD Inventory
170. strArtist (string): Artist of the CD Inventory
172. """
173. strID **=** input('Enter ID: ').strip()
175. strTitle **=** input('What is the CD\'s title? ').strip()
177. strArtist **=** input('What is the Artist\'s name? ').strip()
179. print()
181. **return** strID,strTitle,strArtist
183. @staticmethod
184. **def** show\_inventory(table):
185. """Displays current inventory table

188. Args:
189. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
191. Returns:
192. None.
194. """
195. print('======= The Current Inventory: =======')
196. print('ID\tCD Title (by: Artist)\n')
197. **for** row **in** table:
198. print('{}\t{} (by:{})'.format(**\***row.values()))
199. print('======================================')
201. print()
203. # 1. When program starts, read in the currently saved Inventory
204. FileProcessor.read\_file(strFileName, lstTbl)
206. # 2. start main loop
207. **while** True:
209. # 2.1 Display Menu to user and get choice
210. IO.print\_menu()
212. strChoice **=** IO.menu\_choice()
214. # 3. Process menu selection
215. # 3.1 process exit first
216. **if** strChoice **==** 'x':
217. **break**
219. # 3.2 process load inventory
220. **if** strChoice **==** 'l':
222. print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
224. strYesNo **=** input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')
226. **if** strYesNo.lower() **==** 'yes':
227. print('reloading...')
228. FileProcessor.read\_file(strFileName, lstTbl)
229. IO.show\_inventory(lstTbl)
230. **else**:
231. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
232. IO.show\_inventory(lstTbl)
234. **continue**  # start loop back at top.
236. # 3.3 process add a CD
237. **elif** strChoice **==** 'a':
239. #Getting User Input
240. strID,strTitle,strArtist **=** IO.user\_input()
242. #Calling the function to add item to the table
243. DataProcessor.add\_item(strID,strTitle,strArtist)
245. #Display the CD Inventory
246. IO.show\_inventory(lstTbl)
248. **continue**  # start loop back at top.
250. # 3.4 process display current inventory
251. **elif** strChoice **==** 'i':
253. IO.show\_inventory(lstTbl)
255. **continue**  # start loop back at top.
257. # 3.5 process delete a CD
258. **elif** strChoice **==** 'd':
260. #Get Userinput for which CD to delete
261. #Display Inventory to user
262. IO.show\_inventory(lstTbl)
264. #Ask user which ID to remove
265. intIDDel **=** int(input('Which ID would you like to delete? ').strip())
267. #Calling the function to delete an item from the Inventory table
268. lstTbl **=** DataProcessor.delete\_item(lstTbl,intIDDel)
270. #Display remaining items in the list.
271. IO.show\_inventory(lstTbl)
272. **continue**  # start loop back at top.
274. # 3.6 process save inventory to file
275. **elif** strChoice **==** 's':
277. #Display current inventory and ask user for confirmation to save
278. IO.show\_inventory(lstTbl)
280. strYesNo **=** input('Save this inventory to file? [y/n] ').strip().lower()
282. print()
284. #Process choice
285. **if** strYesNo **==** 'y':
287. #Calling the function to write Inventory details to a file
288. FileProcessor.write\_file(strFileName, lstTbl)
290. **else**:
292. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
294. **continue**  # start loop back at top.
296. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
297. **else**:
298. print('General Error')

Figure 1 - Full Code of loading and deleting items from list of dictionaries.

# Running python script on terminal

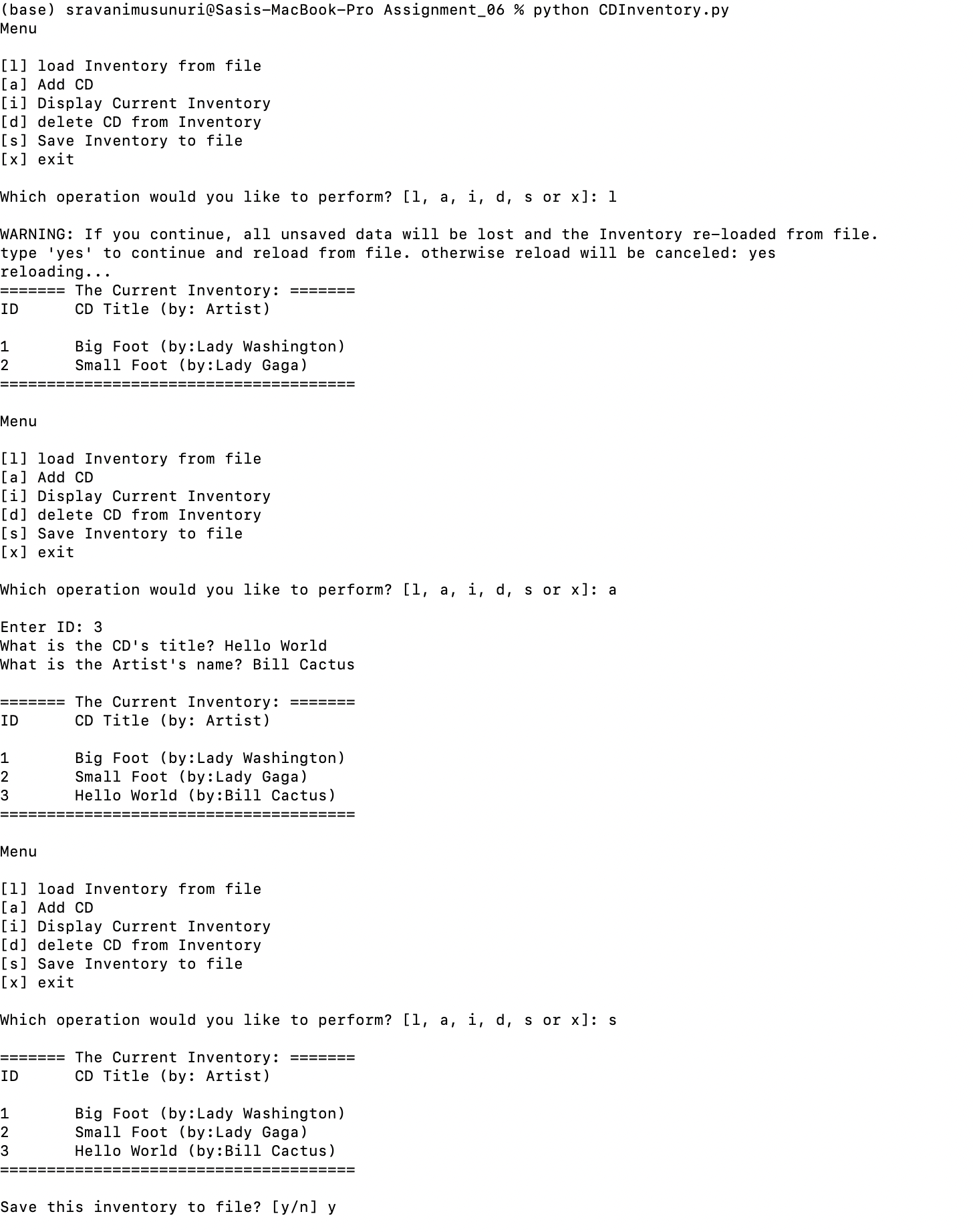


Figure 2 - Screenshot of output of python script running on terminal

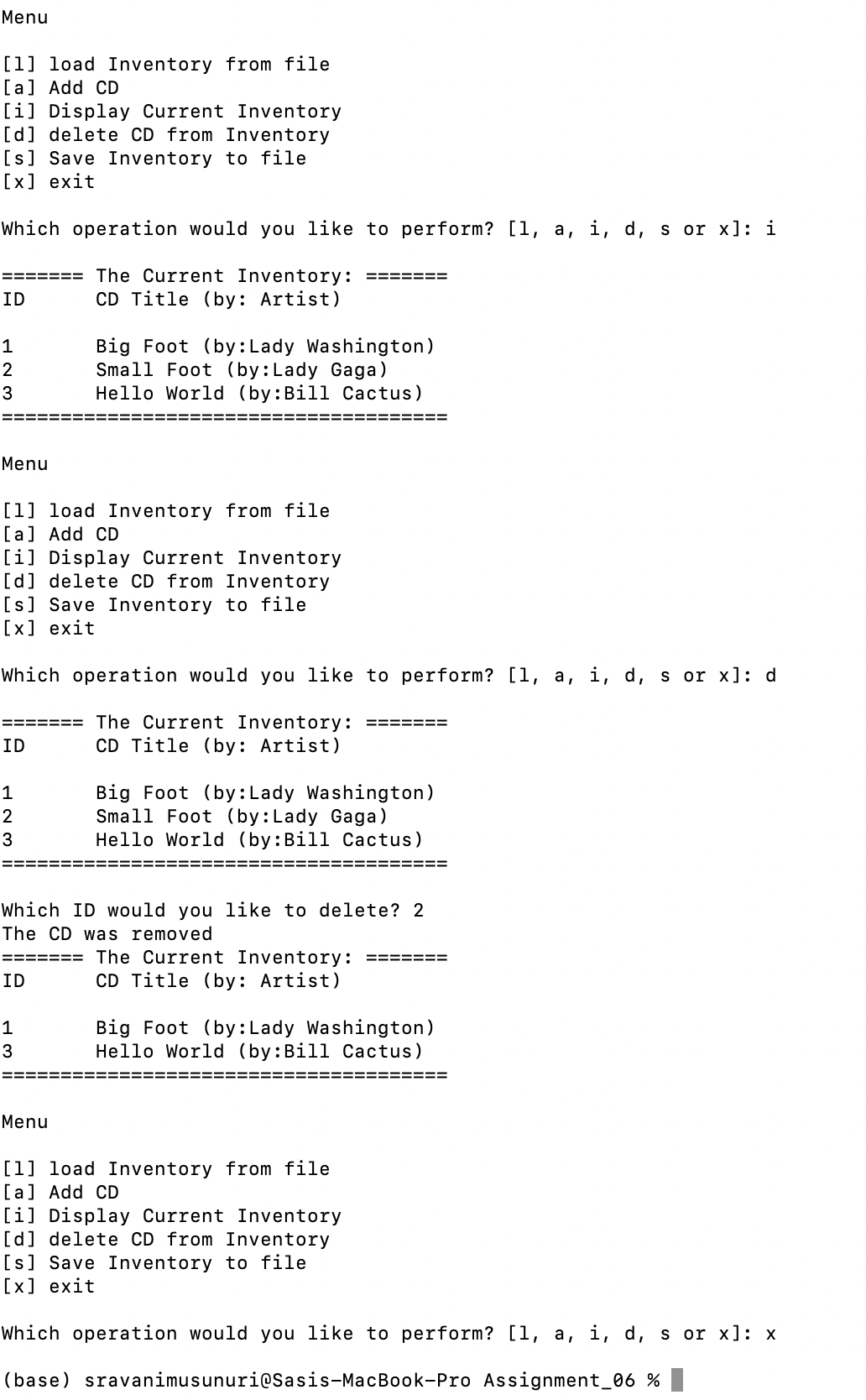
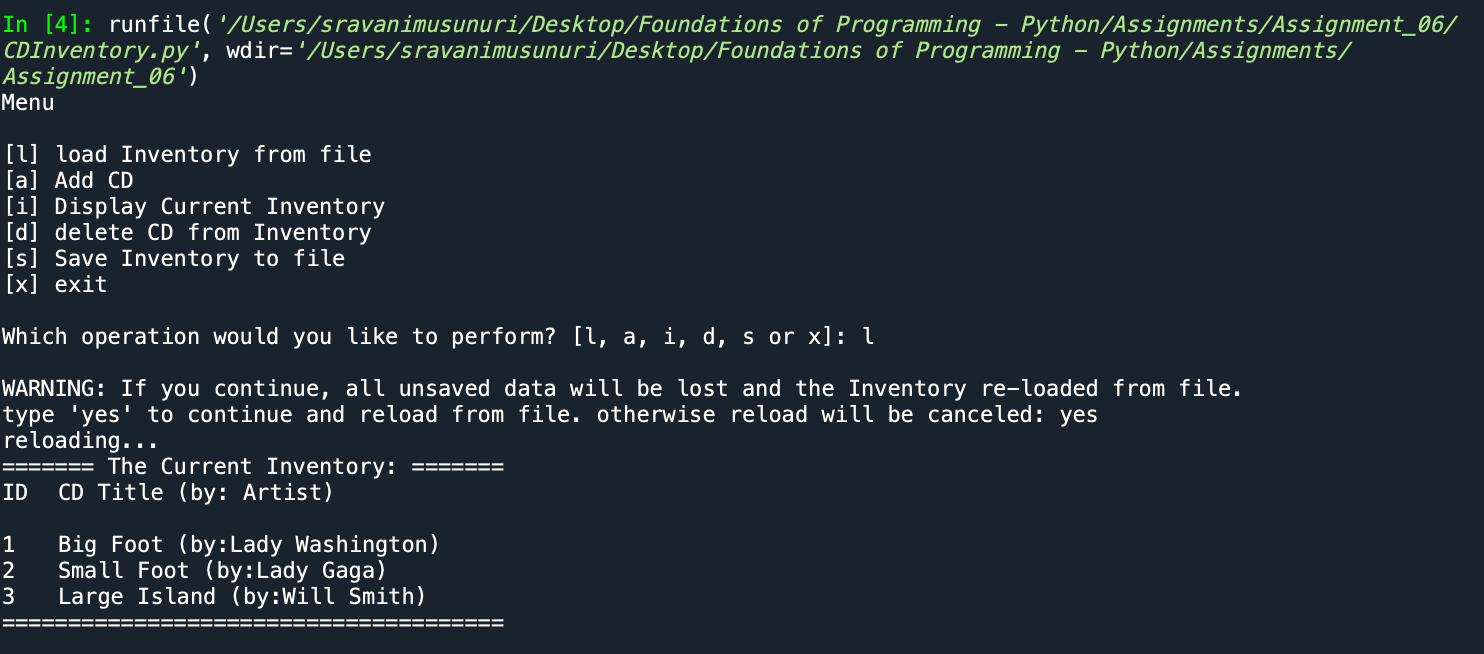
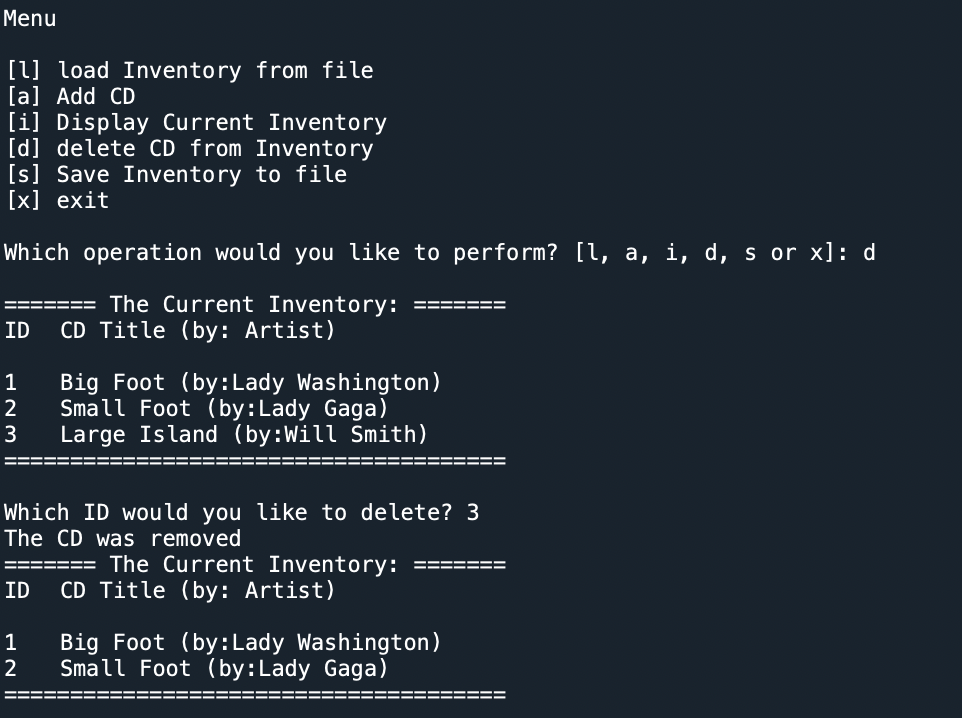


Figure 3 - Screenshot of output of python script running on terminal

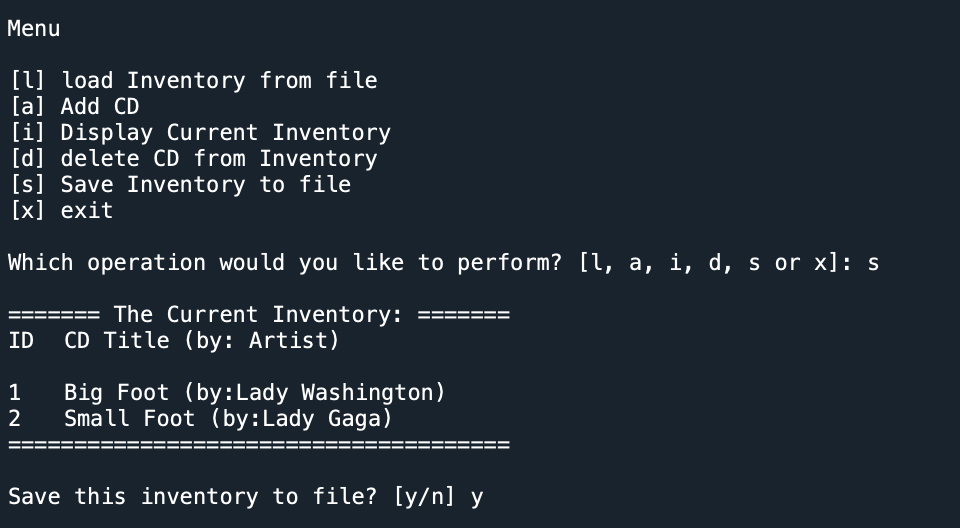
# Running the python script on Spyder



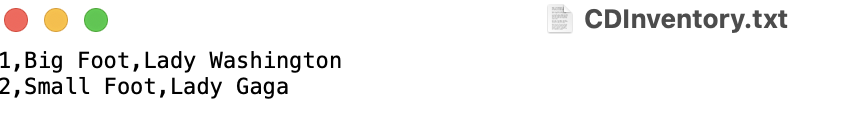
*Figure 4 - Load Inventory from file*

*.*

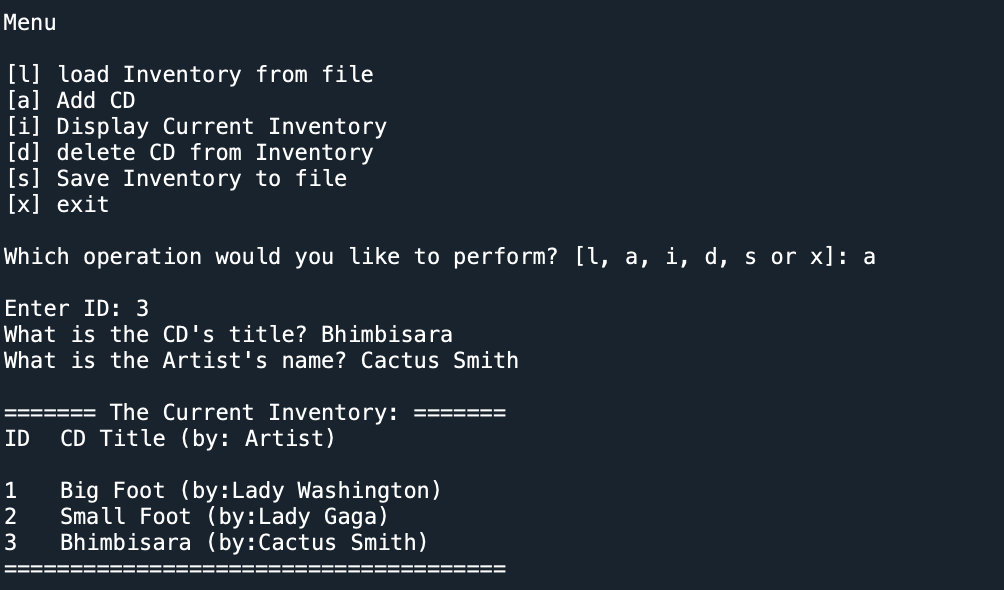
*Figure 5 - Delete CD from Inventory*

**

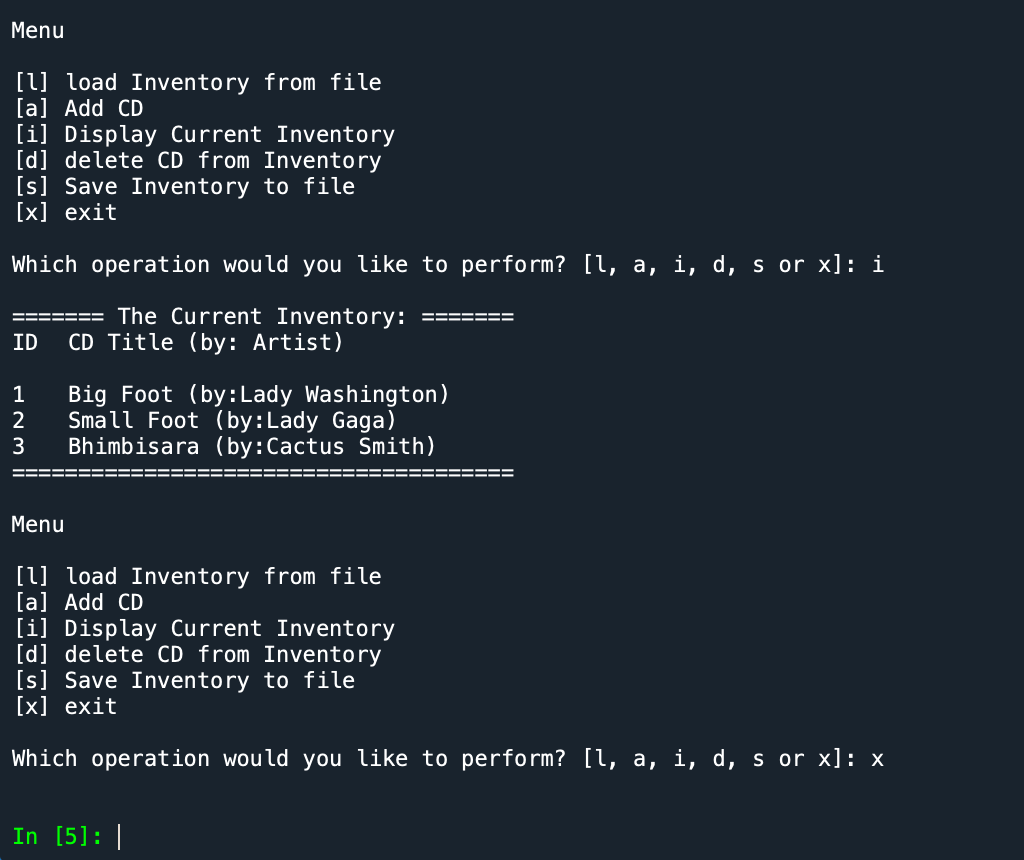
*Figure 6 - Save Inventory to File*

**

*Figure 7 - CDInventory txt file after saving to a file*

**

*Figure 8 - Add CD Inventory*

**

*Figure 9 - Display Current Inventory and Exit*

# GitHub Link

<https://github.com/sravanimusunuri6/Assignment_06>

# Summary

In this module, I got chance to learn:

* Functions and Classes – Few lines of code can be converted to Function and few functions into a Class.
* Difference between Parameters and Arguments
* Difference between global and local variables
* Separation of Concerns to make a Function