Understanding Functions and Classes

# Introduction

This module was based around understanding Functions, Classes, and documentation.

# Adapting the CD Inventory Script

The goal of this assignment was to expand upon the CD inventory program made in Assignment 04 and 05. The primary goal was to change the structure to use functions and follow the Separation of Concerns documentation philosophy in the structure of the code. Like assignment 05, the final part of the challenge of this assignment is to adapt a script created by another individual.

## Reading the Code

I started the assignment by reading the existing code. I went through all the steps that were accomplished by the previous developer. This allowed me to understand the style I was going to write in, and what was left to do.

The previous code writer had conveniently left ‘ToDOs’ in the code, that were highlighted by the Spyder IDE. This made it quick for me to identify what was necessary to accomplish.

The outstanding ‘ToDOs’ in the Functions section were for creating the functions related to adding and deleting data from the list of CDs. Additionally, there was partial code for the function created to save data from the program to the text file. The last ‘ToDO’ was for adding any other function necessary to the Input-Output (IO) section, as necessary.

The outstanding ‘ToDOs’ in the main body of the script were related to creating the adding, deleting, and saving segments. All of these segments had all of the processing code in the main body, which needed to be moved to their respective classes and functions.

## Writing and Documenting Functions

I started modifying the code by relocating the processing scripts from the add, delete, and save segments into the functions. I created the ‘add\_CD’, ‘delete\_CD’, and ‘input\_CD’ functions to carry out the aforementioned functions. The ‘input\_CD’ function was a IO functiont that takes the user’s data to process. The ‘add\_CD’ function uses the strings of user data from the ‘input\_CD’ function, and stored the information into a dictionary row in the list of dictionaries used in the code. The ‘delete\_CD’ function body was simply moved from the body of the script to the ‘delete\_CD’ function.

Additionally, I just moved the existing code for saving data to a file from the main body of the script to the ‘write\_file’ function.

The most time consuming part of this segment was writing the docustrings for the functions. It was important to realize how time consuming documentation and appropriate styling can be when creating functions.

## Modifying the Main Body of the Script

The modifications to the data in the main body were pretty simple, it was a matter of calling the class and the function saved within. I simply had to be sure to create the appropriate number of variables to store the data that was output from function so the data could be used by future functions.

# Running Script in Spyder IDE

The figures below show the result of running the ‘CDinventory.py’ script in Spyder IDE.

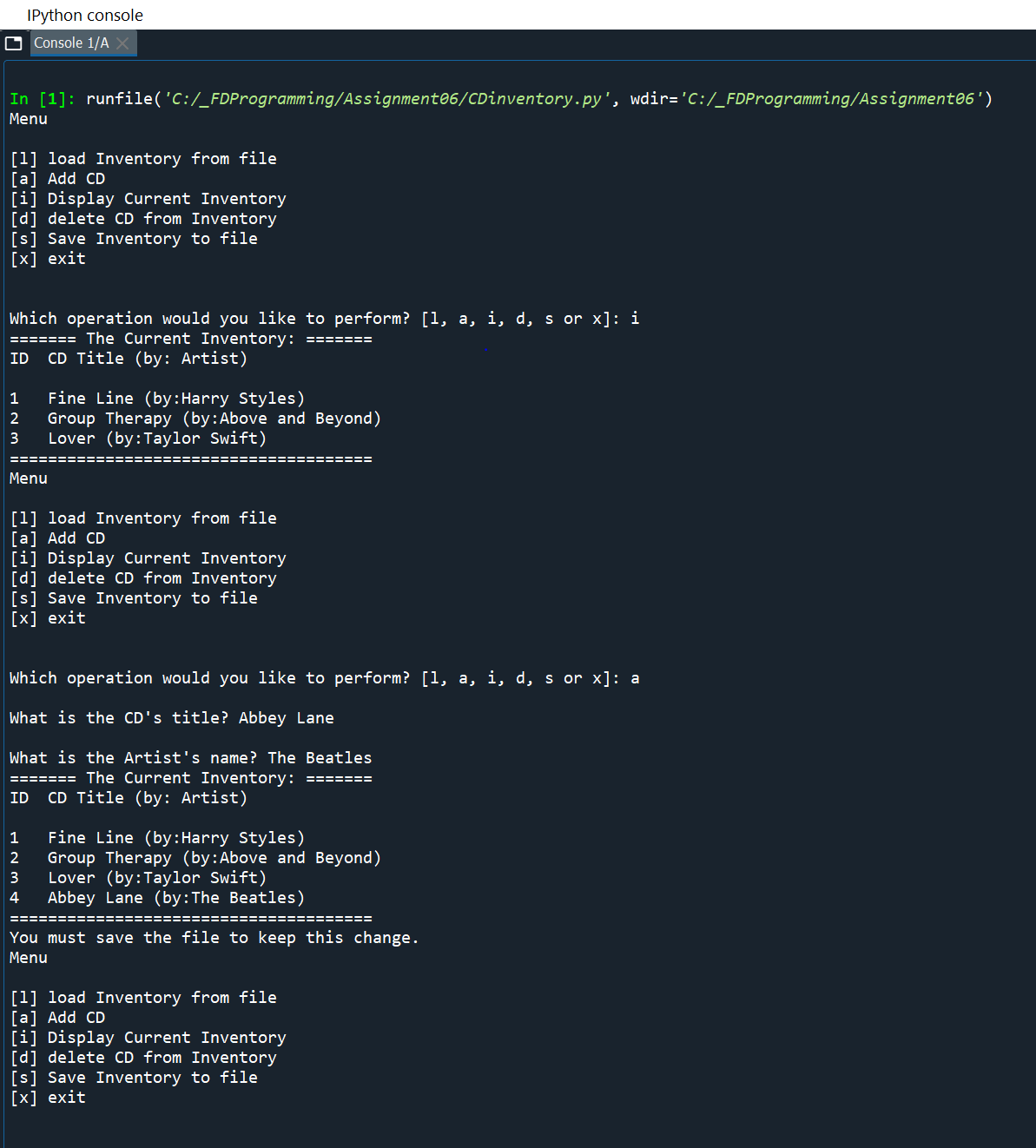


Figure - CDinventory.py program in Spyder

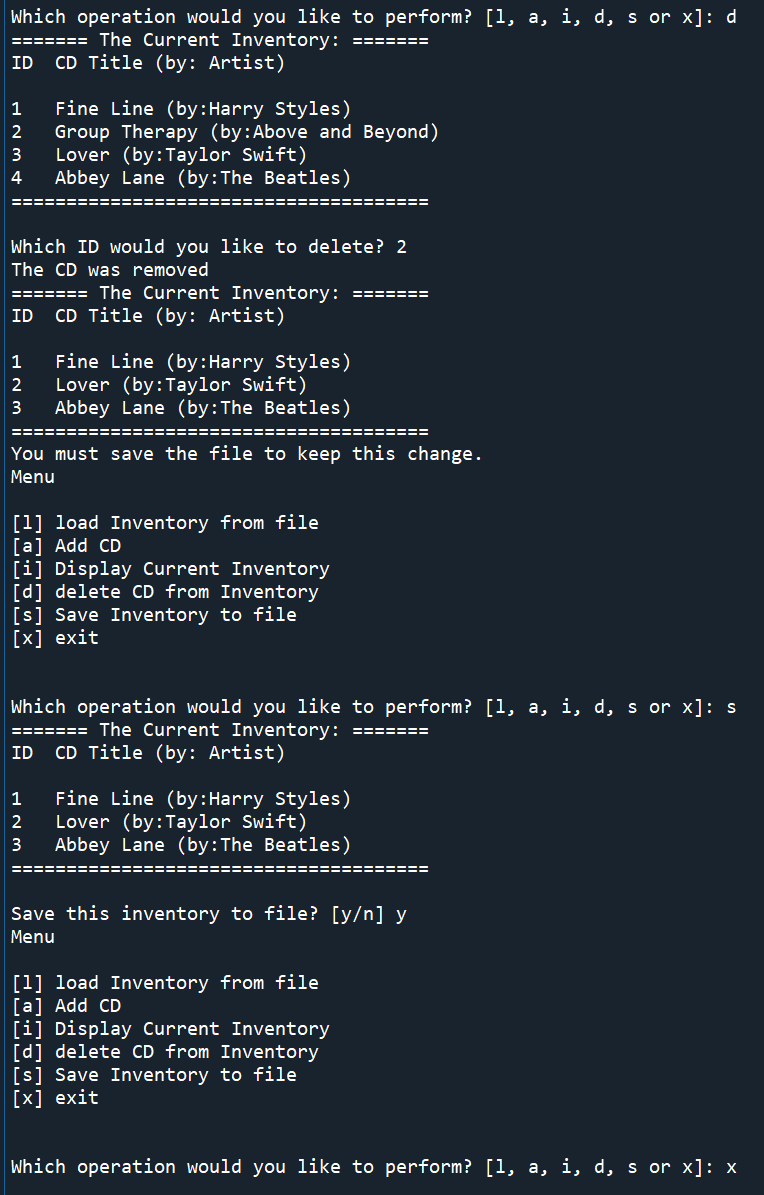


Figure 2 - CDInventory.py in Spyder Cont.

The next figure shows the resulting text file after being run in Spyder:

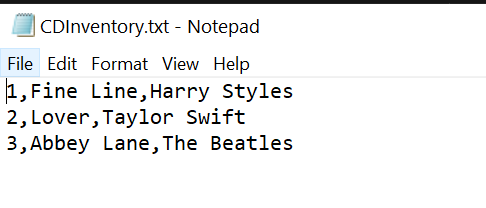


Figure 3 – CDInventory.txt File Generated from CDInventory.py

# Running Script in Terminal Window

The following figures are the result of running ‘CDInventory.py’ script in the terminal window.

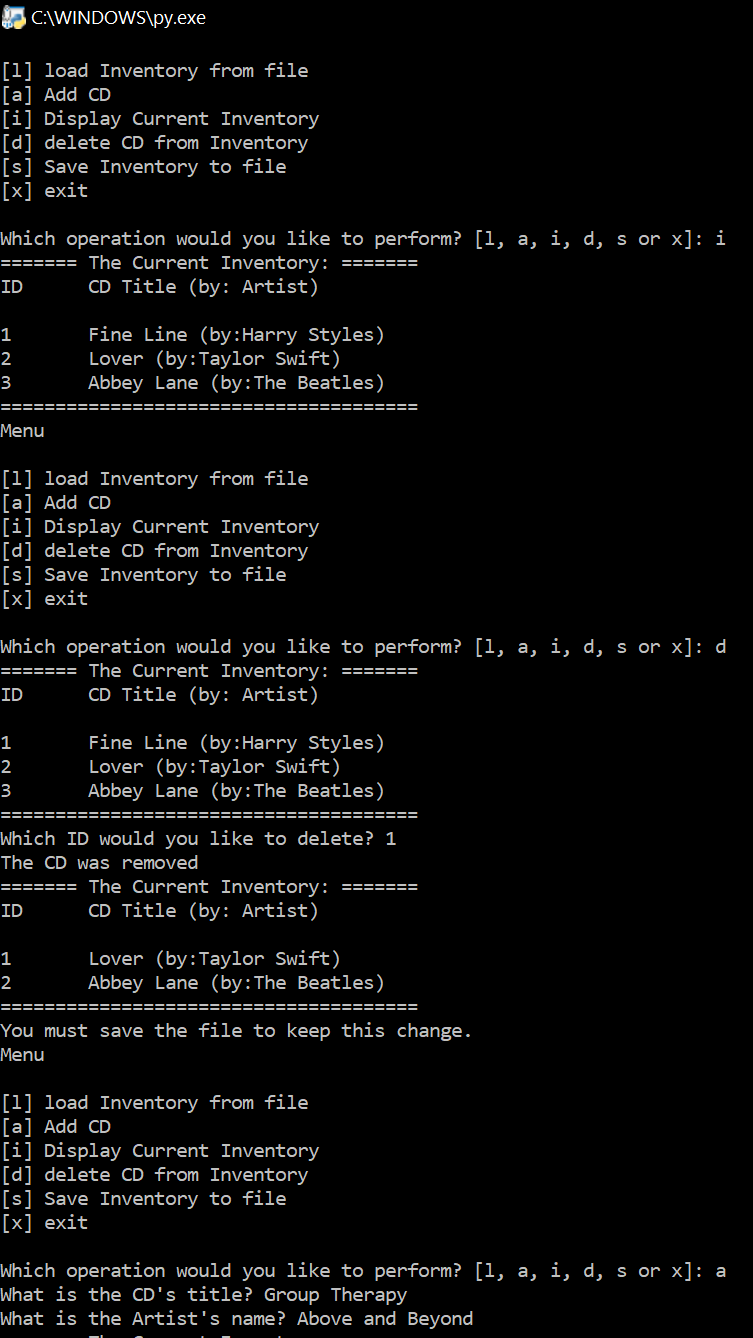


Figure 4 - CDInventory.py Program run in Terminal Window

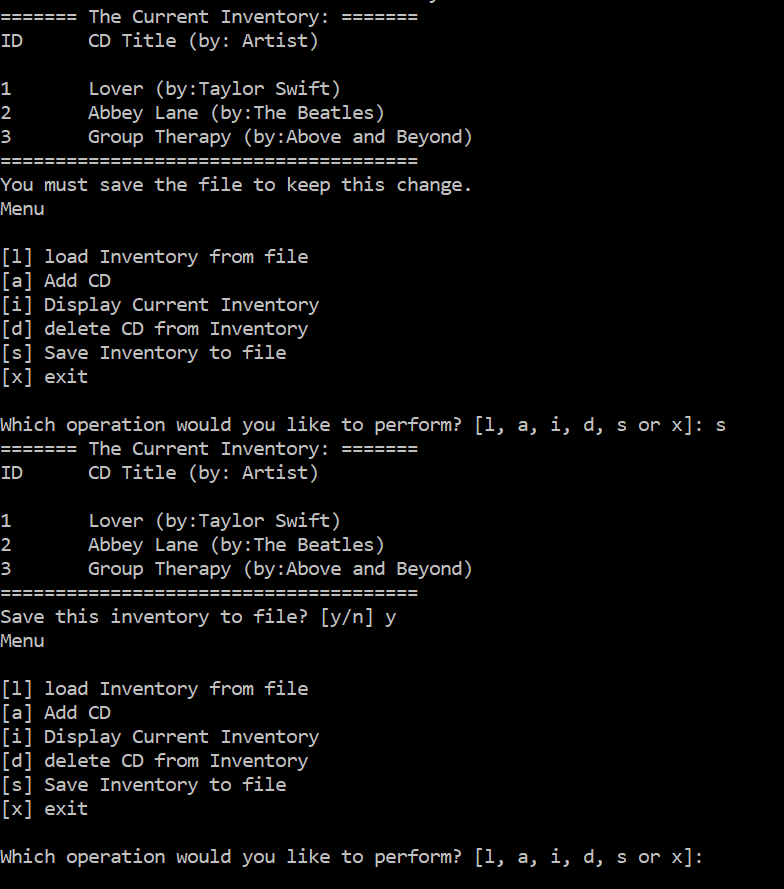


Figure 5 - CDInventory.py in Terminal Window Cont.

# 

Figure 6 – CDinventory.txt created by CDinventory.py in terminal window

# Github Link

Here is the Github link to my code:

<https://github.com/mxlarkin/Assignment_05>

# Summary

In this assignment I learned how to use functions in the code to separate out data retrieval, processing, and presentation. I got more practice in adapting other’s codes as well. I learned how difficult it is to be meticulous with style and documentation. I appreciate being able to practice these skills through the assignments given.

Something I would like would be to have a better handle on how to use the open function. The main issue I encountered was if there wasn’t a file existed with the name needed by the script, it wouldn’t run. This was frustrating, as when I used ‘a+’ or ‘w+’ instead of ‘r’, they could create a new file, but they would delete the information stored or do something unintended.

# Appendix

CDinventory Python script:

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. *#------------------------------------------#*
8. *# -- DATA -- #*
9. strChoice = '' *# User input*
10. **dicTbl = [] *# list of dicationaries to hold data***
11. dicRow = {} *# dictionary of data row*
12. strFileName = 'CDInventory.txt' *# data storage file*
13. objFile = None *# file object*

16. *# -- PROCESSING -- #*
17. **class** DataProcessor:
18. '''Adding and deleting items from the CD inventory table stored during run time'''
19. *# TODone add functions for processing here*
21. @staticmethod
22. **def** add\_CD(title, artist, table):
23. '''Function to add CD to 2D table, list of dics
24. Args:
25. **cdID (string): string to add to dictionary**
26. title (string): string to add to dictionary
27. artist (string): string to add to dictionary
28. table (list of Dics): Table of dics to add new dictionary to
29. Returns:
30. **None.**
31. '''
32. cdID = len(dicTbl) + 1
33. *# intID = int(cdID)*
34. dicRow = {'ID': cdID, 'Title': title, 'Artist': artist}
35. **table.append(dicRow)**
37. @staticmethod
38. **def** delete\_CD(cdID, table):
39. '''Function to delete CD in ED table, list of dics
40. **Args:**
41. cdID (Integer): ID number for CD in dictionary to delete from table
42. table (list of dics): Table of dictionaries to delete dic from
43. Returns:
44. None.
45. **'''**
46. intRowNr = -1
47. blnCDRemoved = False
48. **for** row **in** dicTbl:
49. intRowNr += 1
50. **if int(row['ID']) == intIDDel:**
51. **del** dicTbl[intRowNr]
52. blnCDRemoved = True
53. **break**
54. **for** i **in** range(len(dicTbl)): *# renumbers IDs to prevent duplicates after del row*
55. **j = i + 1**
56. dicTbl[i]['ID'] = j
58. **if** blnCDRemoved:
59. **print**('The CD was removed')
60. **else:**
61. **print**('Could not find this CD!')

64. **class** FileProcessor:
65. **"""Processing the data to and from text file"""**
67. @staticmethod
68. **def** read\_file(file\_name, table):
69. """Function to manage data ingestion from file to a list of dictionaries
71. Reads the data from file identified by file\_name into a 2D table
72. (list of dicts) table one line in the file represents one dictionary row in table.
74. Args:
75. **file\_name (string): name of file used to read the data from**
76. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
78. Returns:
79. None.
80. **"""**
81. table.clear() *# this clears existing data and allows to load data from file*
82. objFile = open(file\_name, 'r')
83. **for** line **in** objFile:
84. data = line.strip().split(',')
85. **dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}**
86. table.append(dicRow)
87. objFile.close()
89. @staticmethod
90. **def write\_file(file\_name, table):**
91. '''Function to write user data from a list of dictionaries to a file
93. Processes data line by line from a dictionary into string form to then write to a
94. text file.
96. Args:
97. file\_name: Name of file to save data from the 2D table used in run time to
98. table: 2D data structure (list of dictionaries) that holds data during run time
99. Returns:
100. **None.**
101. '''
102. *# TODone Add code here*
103. objFile = open(strFileName, 'w')
104. **for** row **in** dicTbl:
105. **lstValues = list(row.values())**
106. lstValues[0] = str(lstValues[0])
107. objFile.write(','.join(lstValues) + '**\n**')
108. objFile.close()
110. ***# -- PRESENTATION (Input/Output) -- #***
112. **class** IO:
113. """Handling Input / Output"""
115. **@staticmethod**
116. **def** print\_menu():
117. """Displays a menu of choices to the user
119. Args:
120. **None.**
122. Returns:
123. None.
124. """
126. **print**('Menu**\n\n**[l] load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')
127. **print**('[d] delete CD from Inventory**\n**[s] Save Inventory to file**\n**[x] exit**\n**')
129. @staticmethod
130. **def menu\_choice():**
131. """Gets user input for menu selection
133. Args:
134. None.
136. Returns:
137. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
139. """
140. **choice = ''**
141. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
142. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
143. *#print('\n') # Add extra space for layout*
144. **return** choice
146. @staticmethod
147. **def** show\_inventory(table):
148. """Displays current inventory table

151. Args:
152. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
154. Returns:
155. **None.**
157. """
158. **print**('======= The Current Inventory: =======')
159. **print**('ID**\t**CD Title (by: Artist)**\n**')
160. **for row in table:**
161. **print**('{}**\t**{} (by:{})'.format(\*row.values()))
162. **print**('======================================')
164. *# TODone add I/O functions as needed*
165. **@staticmethod**
166. **def** input\_CD():
167. '''Get user input for CD to add
168. Args:
169. None
170. **Returns:**
171. strID (string): string of user input for new CD ID
172. strTitle (string): string with user input for new CD title
173. strArtist (string): string with user input for new CD artist
174. '''
175. **title = input('What is the CD\'s title? ').strip()**
176. artist = input('What is the Artist**\'**s name? ').strip()
177. **return** title, artist
179. *# 1. When program starts, read in the currently saved Inventory*
180. **FileProcessor.read\_file(strFileName, dicTbl)**
182. *# 2. start main loop*
183. **while** True:
184. *# 2.1 Display Menu to user and get choice*
185. **IO.print\_menu()**
186. strChoice = IO.menu\_choice()
188. *# 3. Process menu selection*
189. *# 3.1 process exit first*
190. **if strChoice == 'x':**
191. **break**
192. *# 3.2 process load inventory*
193. **if** strChoice == 'l':
194. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
195. **strYesNo = input('Type \'y\' to continue and reload from file; otherwise reload will be cancelled:\n')**
196. **if** strYesNo.lower() == 'y':
197. **print**('reloading...')
198. FileProcessor.read\_file(strFileName, dicTbl)
199. IO.show\_inventory(dicTbl)
200. **else:**
201. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
202. IO.show\_inventory(dicTbl)
203. **continue** *# start loop back at top.*
204. *# 3.3 process add a CD*
205. **elif strChoice == 'a':**
206. *# 3.3.1 Ask user for new ID, CD Title and Artist*
207. *# TODone move IO code into function*
208. strTitle, strArtist = IO.input\_CD()
209. *# 3.3.2 Add item to the table*
210. ***# TODone move processing code into function***
211. DataProcessor.add\_CD(strTitle, strArtist, dicTbl)
212. IO.show\_inventory(dicTbl)
213. **print**('You must save the file to keep this change.')
214. **continue** *# start loop back at top.*
215. ***# 3.4 process display current inventory***
216. **elif** strChoice == 'i':
217. IO.show\_inventory(dicTbl)
218. **continue** *# start loop back at top.*
219. *# 3.5 process delete a CD*
220. **elif strChoice == 'd':**
221. *# 3.5.1 get Userinput for which CD to delete*
222. *# 3.5.1.1 display Inventory to user*
223. IO.show\_inventory(dicTbl)
224. *# 3.5.1.2 ask user which ID to remove*
225. **intIDDel = int(input('Which ID would you like to delete? ').strip())**
226. *# 3.5.2 search thru table and delete CD*
227. *# TODone move processing code into function*
228. DataProcessor.delete\_CD(intIDDel, dicTbl)
229. IO.show\_inventory(dicTbl)
230. **print('You must save the file to keep this change.')**
231. **continue** *# start loop back at top.*
232. *# 3.6 process save inventory to file*
233. **elif** strChoice == 's':
234. *# 3.6.1 Display current inventory and ask user for confirmation to save*
235. **IO.show\_inventory(dicTbl)**
236. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
237. *# 3.6.2 Process choice*
238. **if** strYesNo == 'y':
239. *# 3.6.2.1 save data*
240. ***# TODone move processing code into function***
241. FileProcessor.write\_file(strFileName, dicTbl)
242. **else**:
243. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
244. **continue** *# start loop back at top.*
245. ***# 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:***
246. **else**:
247. **print**('General Error')