Justin Roe

08/19/2020

IT FDN 110 B

Assignment 06

Assignment 06

# Introduction

Assignment 6 was centered around functions; however, variable types, code organization, and classes were also discussed. I struggled with multiple concepts this week, primarily working with multiple / positional arguments, transferring variables from function to function, and incorporating user aids (like the “this ID has already been used” functionality).

# Knowledge Application / Details

See Figures 1-5 for screenshots of the code execution.

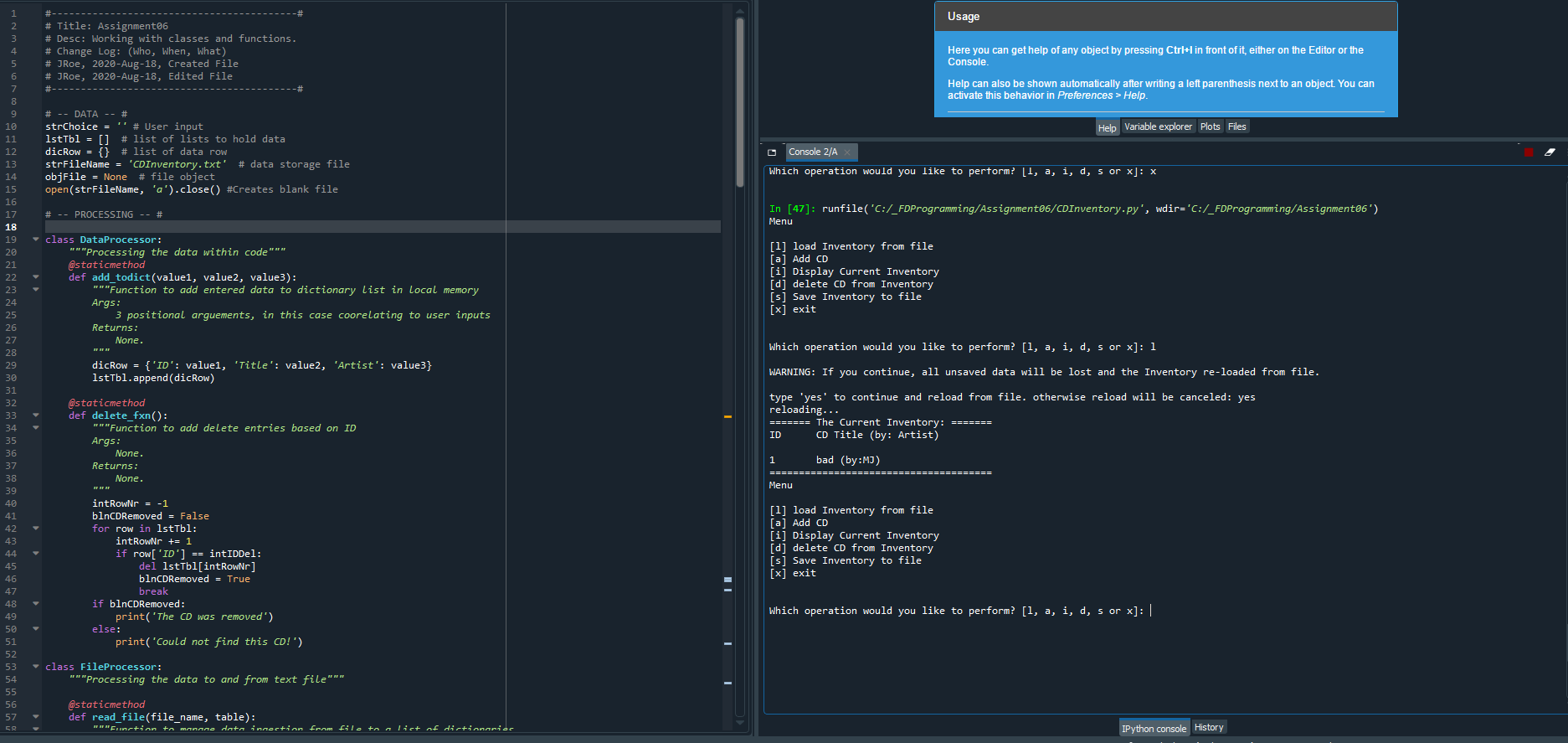


Figure – Spyder run of CDInventory.py script

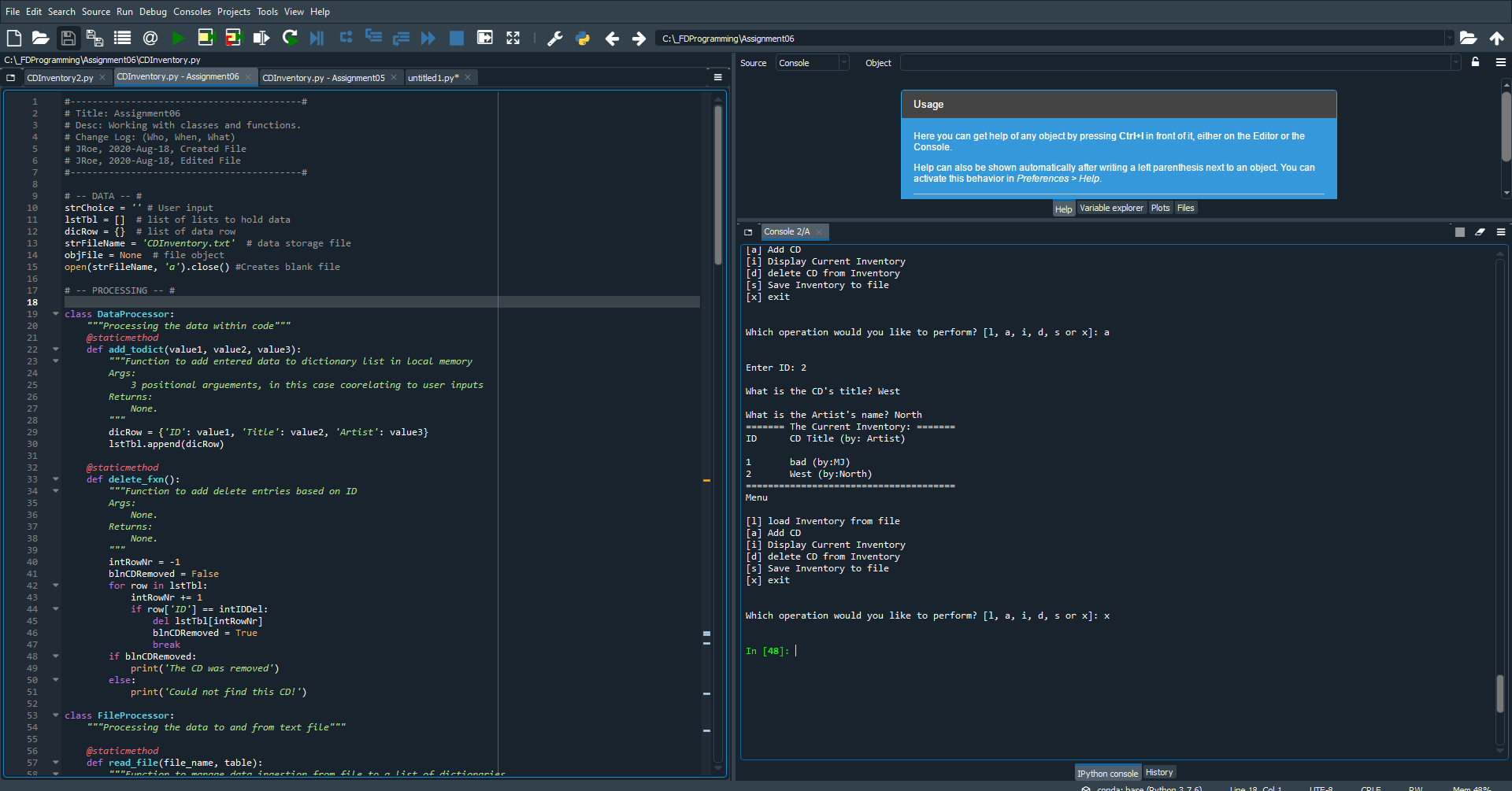


Figure – Spyder run of CDInventory.py script

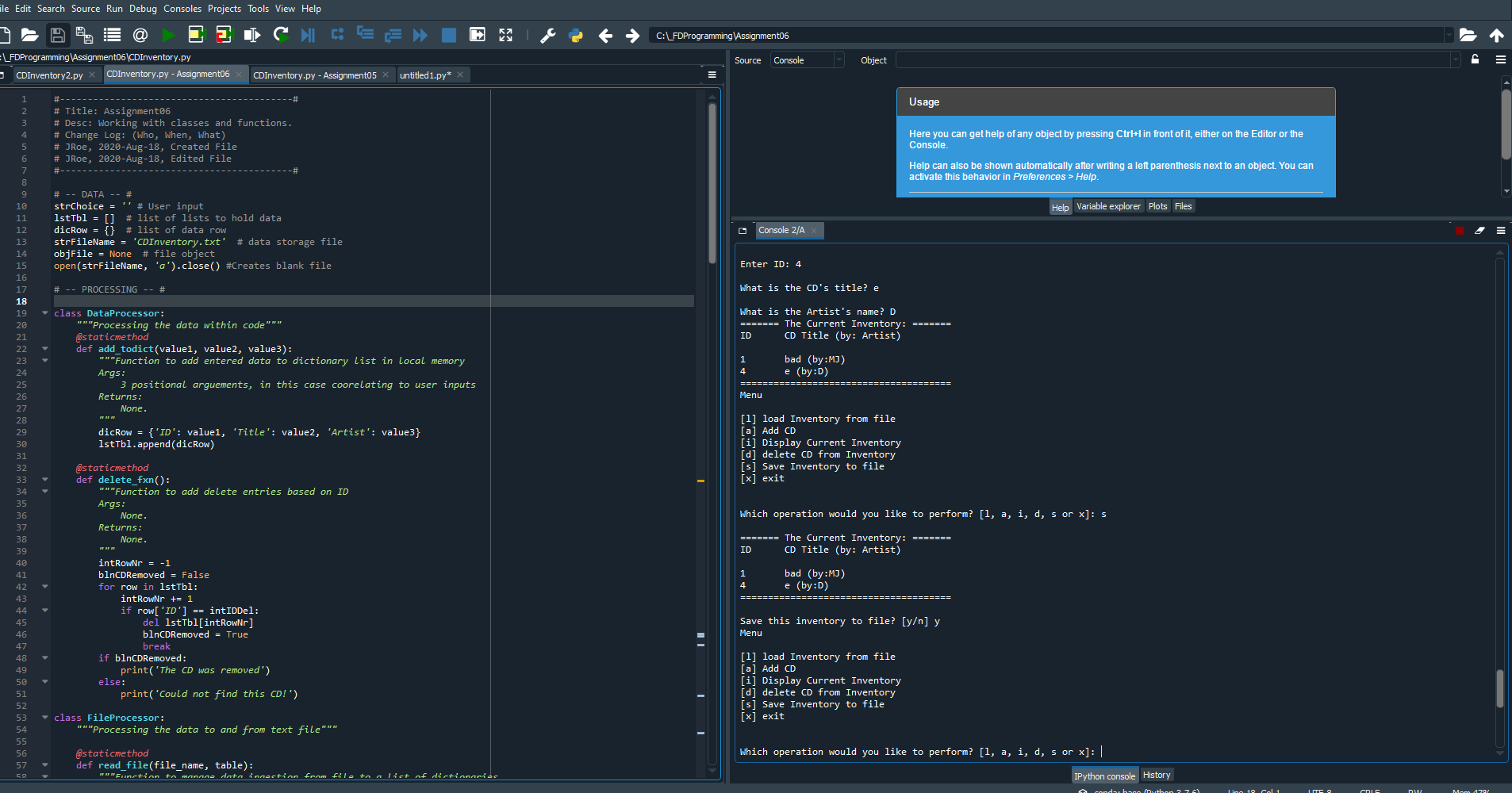


Figure 3 - Spyder run of CDInventory.py script

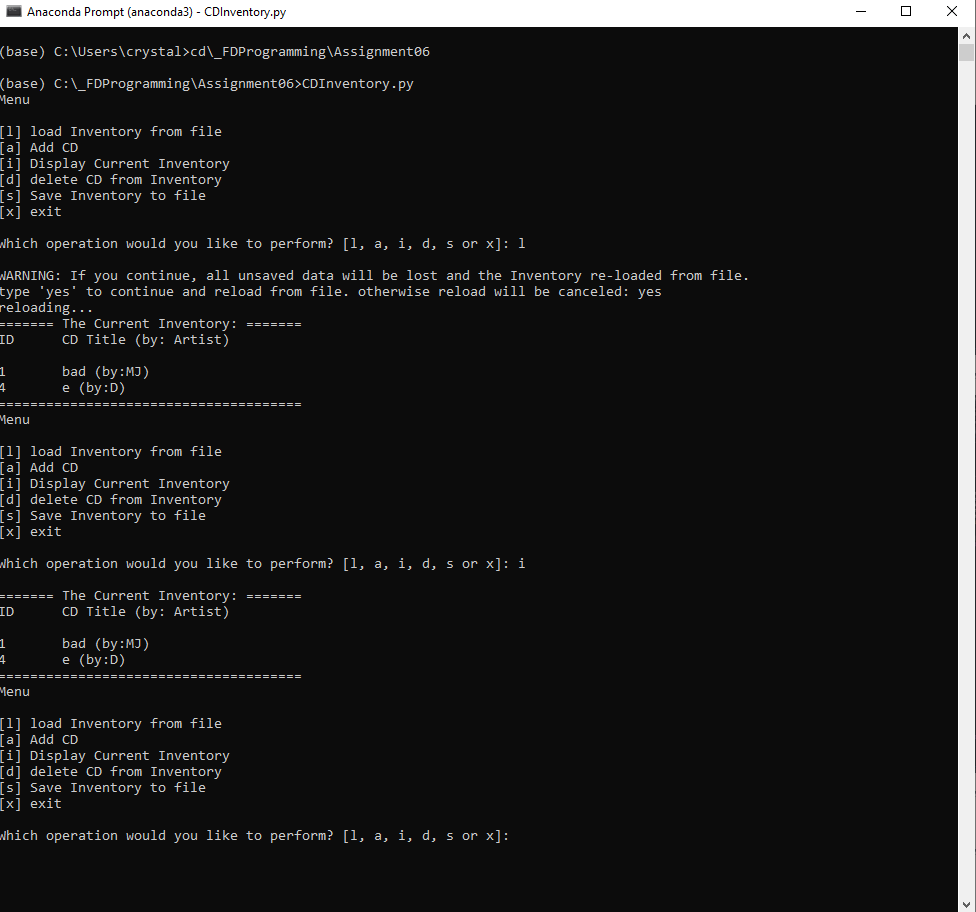


Figure 4 – Representative image of CDInventory.py running in console

# Summary

Overall, this assignment went relatively smoothly. I am still working to understand positional vs named arguments (unpacking the tuple – as reviewed in the notes), these took me a bit of time to understand. The most difficult part, however, was passing the user inputs to the “write” function in the knowledge check. Some sites (like [this one](https://stackoverflow.com/questions/16043797/python-passing-variables-between-functions) (external reference)[[1]](#footnote-1)) handle this by embedding functions in other functions, but it is certainly easier to name variables and set them equal to returned function values – then use positional arguments when passing the values into the write function. As usual, I had my share of issues with general code functionality (try / except statements, while loops, etc) – greatly assisted by this [geek-university](https://geek-university.com/python/the-try-except-statements/) and [python docs](https://docs.python.org/3/tutorial/errors.html) webpages (both external references)[[2]](#footnote-2). My Github link is [here](https://github.com/jusroe/Assignment06).

# Appendix A – Syntax

Generated using [planetb’s webpage](http://planetb.ca/syntax-highlight-word) (external reference)[[3]](#footnote-3) web page

1. #------------------------------------------#
2. # Title: Assignment06
3. # Desc: Working with classes and functions.
4. # Change Log: (Who, When, What)
5. # JRoe, 2020-Aug-18, Created File
6. # JRoe, 2020-Aug-18, Edited 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
15. open(strFileName, 'a').close() #Creates blank file
17. # -- PROCESSING -- #
19. **class** DataProcessor:
20. """Processing the data within code"""
21. @staticmethod
22. **def** add\_todict(value1, value2, value3):
23. """Function to add entered data to dictionary list in local memory
24. Args:
25. 3 positional arguements, in this case coorelating to user inputs
26. Returns:
27. None.
28. """
29. dicRow = {'ID': value1, 'Title': value2, 'Artist': value3}
30. lstTbl.append(dicRow)
32. @staticmethod
33. **def** delete\_fxn():
34. """Function to add delete entries based on ID
35. Args:
36. None.
37. Returns:
38. None.
39. """
40. intRowNr = -1
41. blnCDRemoved = False
42. **for** row **in** lstTbl:
43. intRowNr += 1
44. **if** row['ID'] == intIDDel:
45. **del** lstTbl[intRowNr]
46. blnCDRemoved = True
47. **break**
48. **if** blnCDRemoved:
49. **print**('The CD was removed')
50. **else**:
51. **print**('Could not find this CD!')
53. **class** FileProcessor:
54. """Processing the data to and from text file"""
56. @staticmethod
57. **def** read\_file(file\_name, table):
58. """Function to manage data ingestion from file to a list of dictionaries
59. Reads the data from file identified by file\_name into a 2D table
60. (list of dicts) table one line in the file represents one dictionary row in table.
61. Args:
62. file\_name (string): name of file used to read the data from
63. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
64. Returns:
65. None.
66. """
67. table.clear()  # this clears existing data and allows to load data from file
68. objFile = open(file\_name, 'r')
69. **for** line **in** objFile:
70. data = line.strip().split(',')
71. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
72. table.append(dicRow)
73. objFile.close()
75. @staticmethod
76. **def** save\_fxn(): # AKA "Write\_file" as in starter code
77. """Function to save entered data to designated file
78. Args:
79. None.
80. Returns:
81. None.
82. """
83. objFile = open(strFileName, 'w')
84. **for** row **in** lstTbl:
85. lstValues = list(row.values())
86. lstValues[0] = str(lstValues[0])
87. objFile.write(','.join(lstValues) + '\n')
88. objFile.close()

91. # -- PRESENTATION (Input/Output) -- #
93. **class** IO:
94. """Handling Input / Output"""
96. @staticmethod
97. **def** print\_menu():
98. """Displays a menu of choices to the user
99. Args:
100. None.
101. Returns:
102. None.
103. """
105. **print**('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
106. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
108. @staticmethod
109. **def** menu\_choice():
110. """Gets user input for menu selection
111. Args:
112. None.
113. Returns:
114. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
115. """
116. choice = ' '
117. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
118. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
119. **print**()  # Add extra space for layout
120. **return** choice
122. @staticmethod
123. **def** user\_entry():
124. """Gets user input for "enter data" section
125. Args:
126. None.
127. Returns:
128. ID, song, and title
129. """
130. **while** True:
131. strID = input('Enter ID: ').strip()
132. # Ensure only integer inputs
133. **try**:
134. val = int(strID)
135. **break**
136. **except** ValueError:
137. **print**('\nIntegers Only!\n')
138. **continue**
139. **break**
140. intsrtID = int(strID)
141. strTitle = input('What is the CD\'s title? ').strip()
142. stArtist = input('What is the Artist\'s name? ').strip()
143. **return** intsrtID, strTitle, stArtist
145. @staticmethod
146. **def** show\_inventory(table):
147. """Displays current inventory table
148. Args:
149. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
150. Returns:
151. None.
152. """
153. **print**('======= The Current Inventory: =======')
154. **print**('ID\tCD Title (by: Artist)\n')
155. **for** row **in** table:
156. **print**('{}\t{} (by:{})'.format(\*row.values()))
157. **print**('======================================')
159. # 1. When program starts, read in the currently saved Inventory
160. FileProcessor.read\_file(strFileName, lstTbl)
162. # 2. start main loop
163. **while** True:
164. # 2.1 Display Menu to user and get choice
165. # need to create a blank file
166. IO.print\_menu()
167. strChoice = IO.menu\_choice()
168. # 3. Process menu selection
169. # 3.1 process exit first
170. **if** strChoice == 'x':
171. **break**
172. # 3.2 process load inventory
173. **if** strChoice == 'l':
174. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
175. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')
176. **if** strYesNo.lower() == 'yes':
177. **print**('reloading...')
178. FileProcessor.read\_file(strFileName, lstTbl)
179. IO.show\_inventory(lstTbl)
180. **else**:
181. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
182. IO.show\_inventory(lstTbl)
183. **continue**  # start loop back at top.
185. # 3.3 process add a CD
186. **elif** strChoice == 'a':
187. # 3.3.1 Ask user for new ID, CD Title and Artist
188. # TODO move IO code into function
189. # TODO prevent user from entering same CD
190. ID, Title, Artist = IO.user\_entry()
191. # 3.3.2 Add item to the table
192. # TODO move processing code into function
193. DataProcessor.add\_todict(ID, Title, Artist)
194. IO.show\_inventory(lstTbl)
195. **continue**  # start loop back at top.
197. # 3.4 process display current inventory
198. **elif** strChoice == 'i':
199. IO.show\_inventory(lstTbl)
200. **continue**  # start loop back at top.
201. # 3.5 process delete a CD
203. **elif** strChoice == 'd':
204. # 3.5.1 get Userinput for which CD to delete
205. # 3.5.1.1 display Inventory to user
206. IO.show\_inventory(lstTbl)
207. # 3.5.1.2 ask user which ID to remove
208. intIDDel = int(input('Which ID would you like to delete? ').strip())
209. # 3.5.2 search thru table and delete CD
210. # TODO move processing code into function
211. DataProcessor.delete\_fxn()
212. IO.show\_inventory(lstTbl)
213. **continue**  # start loop back at top.
215. # 3.6 process save inventory to file
216. **elif** strChoice == 's':
217. # 3.6.1 Display current inventory and ask user for confirmation to save
218. IO.show\_inventory(lstTbl)
219. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
220. # 3.6.2 Process choice
221. **if** strYesNo == 'y':
222. # 3.6.2.1 save data
223. # TODO move processing code into function
224. FileProcessor.save\_fxn()
225. **else**:
226. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
227. **continue**  # start loop back at top.
229. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
230. **else**:
231. **print**('General Error')

1. Retrieved 2020-Aug 19 [↑](#footnote-ref-1)
2. Retrieved 2020-Aug 19 [↑](#footnote-ref-2)
3. Retrieved 2020-Aug-19 [↑](#footnote-ref-3)