Ryan Goding

8/19/2020

IT FDN 110: Introduction to Programming (Python)

Assignment 06

# Introduction

Assignment 06 asks us to modify a provided script of CDInventory.py to use functions to address TODO items and use docstrings to document what these functions are doing.

# Body

This assignment introduced me to functions and how to work with existing code written by someone other than yourself. I enjoyed using functions to more organize and structure the code, instead of one large segment. I also found it useful that Spyder allows you to collapse and expand functions or classes. This helped me when I had to debug the code. Figure 1 shows the modified CDInventory\_Starter code being worked in Spyder.

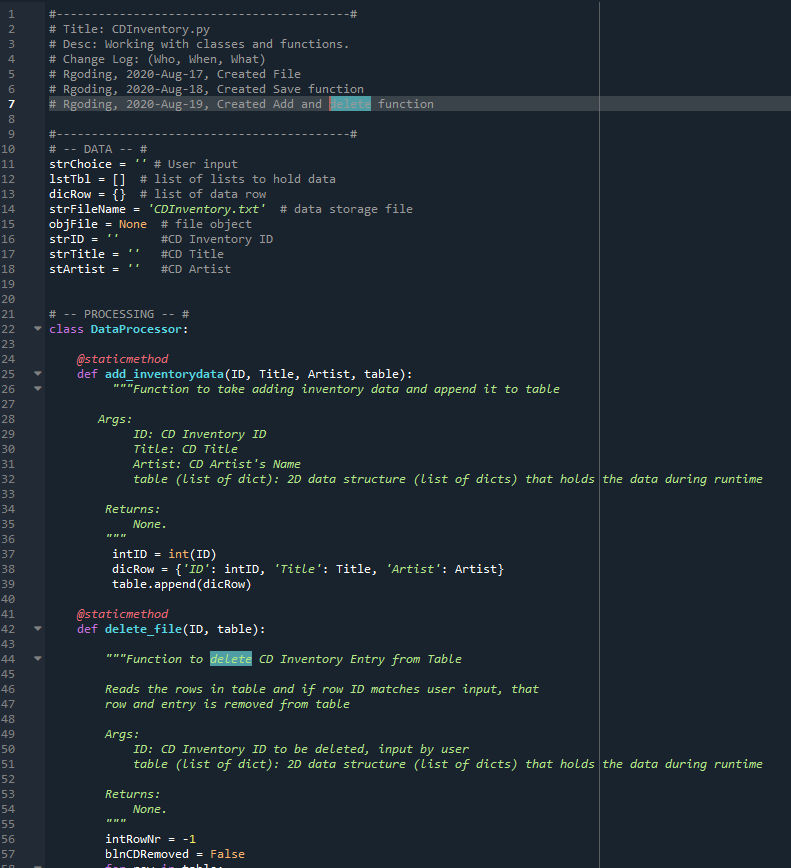


Figure 1: Developing CDInventory.py in Spyder IDE

I first started the assignment by creating the save (write\_file) function. I cut part of the code from the else if statement and brought it up to be a function within the DataProcessor class. I also had to become familiar with the inputs that functions use and how that’s translated into when the function is called. For the write\_file I required that the file name and table would be the input for the function. After this I repeated this process for the other functions, cutting existing code and making sure each function had the required arguments to satisfy the existing code. To test the save functionality two CD entries were created and then saved as shown in Figure 2 and Figure 3. Figure 4 then shows CDInventory.py loading CD entries that were previously saved to CDInventory.txt.

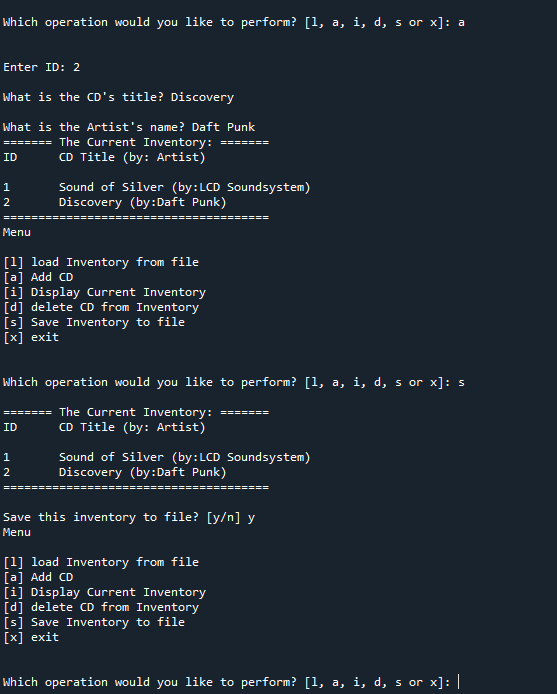


Figure 2: CDInventory.py Script Run in Spyder, Saving two CDs

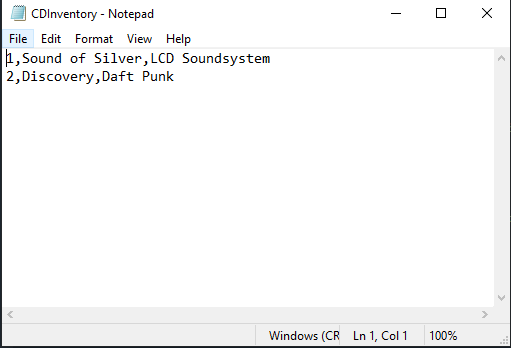


Figure 3: Two CDs Written to CDInvetory.txt

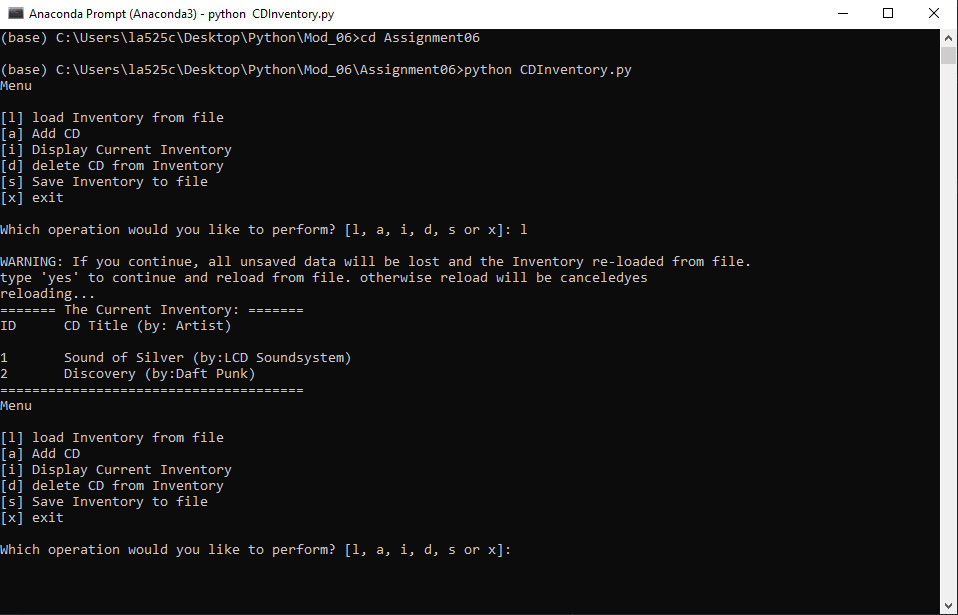


Figure 4: Running CDInventory.py on terminal, loading data from txt file, and displaying Inventory

Figure 5 shows the user deleting CD id 1 after it has been entered.

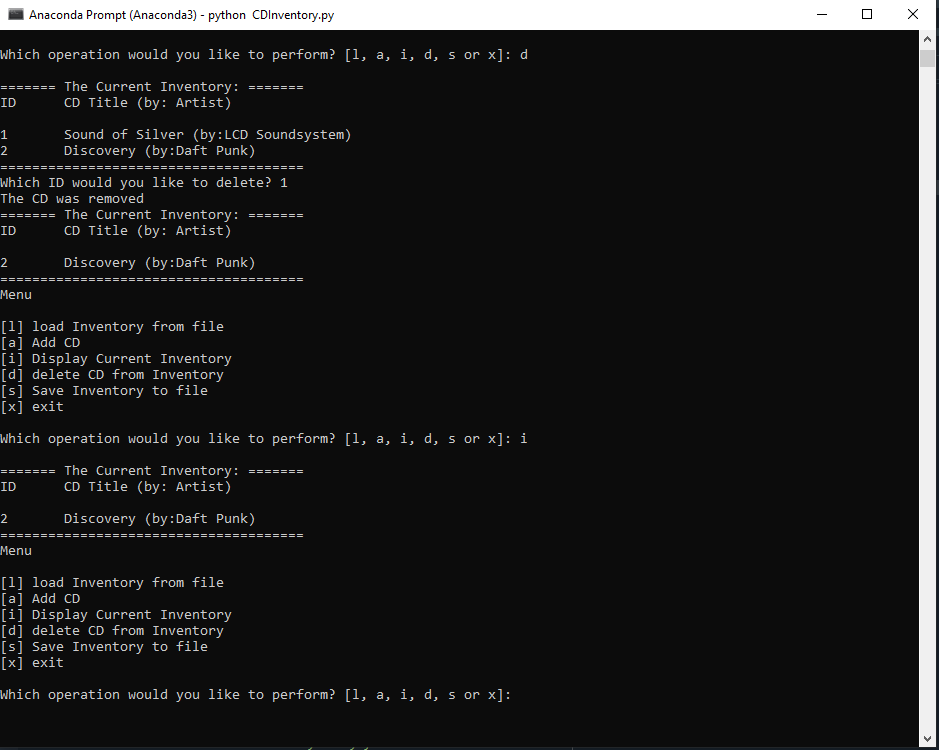


Figure 5: Deleting an Entry, and then Saving to txt file

This assignment also asked us to create an Assignment\_06 repository on Github to share our script and knowledge document. The link for my repository is <https://github.com/rgoding/Assignment_06>

# Summary

I found this assignment rewarding as I do like to organize and make things easier to read and understand. So using functions to make CDInventory.py more readable and workable was enjoyable for me. I wasn’t sure if we should refrain from using functions within functions, as I can see this getting hard to follow, but did use it to handle the different aspects of the save function.

# References

1. Edureka!, How to fix this? ValueError: invalid literal for int() with base 10 error in Python, <https://www.edureka.co/community/30685/this-valueerror-invalid-literal-for-with-base-error-python>
2. Net-informations.com, invalid literal for int() with base 10, <http://net-informations.com/python/err/int.htm>

# Appendix

## Listing CDInventory.py

1. #------------------------------------------#
2. # Title: CDInventory.py
3. # Desc: Working with classes and functions.
4. # Change Log: (Who, When, What)
5. # Rgoding, 2020-Aug-17, Created File
6. # Rgoding, 2020-Aug-18, Created Save function
7. # Rgoding, 2020-Aug-19, Created Add and delete function
9. #------------------------------------------#
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
16. strID = ''      #CD Inventory ID
17. strTitle = ''   #CD Title
18. stArtist = ''   #CD Artist

21. # -- PROCESSING -- #
22. **class** DataProcessor:
24. @staticmethod
25. **def** add\_inventorydata(ID, Title, Artist, table):
26. """Function to take adding inventory data and append it to table
28. Args:
29. ID: CD Inventory ID
30. Title: CD Title
31. Artist: CD Artist's Name
32. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
34. Returns:
35. None.
36. """
37. intID = int(ID)
38. dicRow = {'ID': intID, 'Title': Title, 'Artist': Artist}
39. table.append(dicRow)
41. @staticmethod
42. **def** delete\_file(ID, table):
44. """Function to delete CD Inventory Entry from Table
46. Reads the rows in table and if row ID matches user input, that
47. row and entry is removed from table
49. Args:
50. ID: CD Inventory ID to be deleted, input by user
51. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
53. Returns:
54. None.
55. """
56. intRowNr = -1
57. blnCDRemoved = False
58. **for** row **in** table:
59. intRowNr += 1
60. **if** row['ID'] == ID:
61. **del** table[intRowNr]
62. blnCDRemoved = True
63. **break**
64. **if** blnCDRemoved:
65. **print**('The CD was removed')
66. **else**:
67. **print**('Could not find this CD!')

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 Save the data to CDInventory.txt file
99. Opens the strFileName to write to, then looks at 1stTbl and saves each
100. row
102. Args:
103. file\_name (string): name of file used to write the data to
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(file\_name, 'a')
110. **for** row **in** table:
111. strRow = ''
112. **for** item **in** row.values():
113. strRow += str(item) + ','
114. strRow = strRow[:-1] + '\n'
115. objFile.write(strRow)
116. objFile.close()

119. # -- PRESENTATION (Input/Output) -- #
121. **class** IO:
122. """Handling Input / Output"""
124. @staticmethod
125. **def** print\_menu():
126. """Displays a menu of choices to the user
128. Args:
129. None.
131. Returns:
132. None.
133. """
135. **print**('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
136. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
138. @staticmethod
139. **def** menu\_choice():
140. """Gets user input for menu selection
142. Args:
143. None.
145. Returns:
146. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
148. """
149. choice = ' '
150. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
151. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
152. **print**()  # Add extra space for layout
153. **return** choice
155. @staticmethod
156. **def** show\_inventory(table):
157. """Displays current inventory table

160. Args:
161. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
163. Returns:
164. None.
166. """
167. **print**('======= The Current Inventory: =======')
168. **print**('ID\tCD Title (by: Artist)\n')
169. **for** row **in** table:
170. **print**('{}\t{} (by:{})'.format(\*row.values()))
171. **print**('======================================')
173. @staticmethod
174. **def** add\_inventory():
175. """Add Inventory to CDInventory.txt
177. Args:
178. None
180. Returns:
181. strID, strTitle, stArtist for data processing function to handle
183. """
184. strID = input('Enter ID: ').strip()
185. strTitle = input('What is the CD\'s title? ').strip()
186. stArtist = input('What is the Artist\'s name? ').strip()
187. DataProcessor.add\_inventorydata(strID,strTitle,stArtist,lstTbl)

190. # 1. When program starts, read in the currently saved Inventory
191. FileProcessor.read\_file(strFileName, lstTbl)
193. # 2. start main loop
194. **while** True:
195. # 2.1 Display Menu to user and get choice
196. IO.print\_menu()
197. strChoice = IO.menu\_choice()
199. # 3. Process menu selection
200. # 3.1 process exit first
201. **if** strChoice == 'x':
202. **break**
203. # 3.2 process load inventory
204. **if** strChoice == 'l':
205. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
206. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')
207. **if** strYesNo.lower() == 'yes':
208. **print**('reloading...')
209. FileProcessor.read\_file(strFileName, lstTbl)
210. IO.show\_inventory(lstTbl)
211. **else**:
212. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
213. IO.show\_inventory(lstTbl)
214. **continue**  # start loop back at top.
215. # 3.3 process add a CD
216. **elif** strChoice == 'a':
217. # 3.3.1 Ask user for new ID, CD Title and Artist
218. IO.add\_inventory()
219. # 3.3.2 Add item to the table
221. IO.show\_inventory(lstTbl)
222. **continue**  # start loop back at top.
223. # 3.4 process display current inventory
224. **elif** strChoice == 'i':
225. IO.show\_inventory(lstTbl)
226. **continue**  # start loop back at top.
227. # 3.5 process delete a CD
228. **elif** strChoice == 'd':
229. # 3.5.1 get Userinput for which CD to delete
230. # 3.5.1.1 display Inventory to user
231. IO.show\_inventory(lstTbl)
232. # 3.5.1.2 ask user which ID to remove
233. intIDDel = int(input('Which ID would you like to delete? ').strip())
234. # 3.5.2 search thru table and delete CD
235. DataProcessor.delete\_file(intIDDel,lstTbl)
236. IO.show\_inventory(lstTbl)
237. **continue**  # start loop back at top.
238. # 3.6 process save inventory to file
239. **elif** strChoice == 's':
240. # 3.6.1 Display current inventory and ask user for confirmation to save
241. IO.show\_inventory(lstTbl)
242. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
243. # 3.6.2 Process choice
244. **if** strYesNo == 'y':
245. # 3.6.2.1 save data
246. FileProcessor.write\_file(strFileName,lstTbl)
247. **else**:
248. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
249. **continue**  # start loop back at top.
250. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
251. **else**:
252. **print**('General Error')