Nasser Saber

November 20, 2021

Foundations of Programming, Python

Assignment 06

Creating a Python Program for Managing an Inventory List using Classes and Functions

# Introduction

# In this assignment, I will explain the steps I used to create a Python Script that provides the user with menu options to choose from and performs the corresponding tasks following the user input. This script mainly involves working with classes and functions to handle the data IO, processing, writing to and reading from text files.

# Creating the Script

I started by modifying CDInventory\_Starter.py in Spyder. The work mainly involved moving multiple lines/blocks of code from the main body to the existing classes in the form of functions and then calling them at relevant points. In the DataProcessor class, these newly added functions include add\_table() to process and add the user input data to a 2D list table and row\_delete() to search through table and delete CD per user’s choice. Also in the FileProcessor class, write\_file() function was added to write the data in list of dictionaries (lstTbl) to a text file (CDInventory.txt). Finally in the IO class, user\_input() was created which gets user input for CD and returns the ID, title and artist name. additionaly, Docstrings were added to all functions to provide clear description of tasks, arguments and outputs. The corresponding lines of code in the main body of the program were also adjusted for calling the relevant functions. It should be noted that CDInventory.txt should be provided as part of the first program installation.

# Testing the Script

I ran the script in Spyder first. It runs well by asking the user to make the menu choice and performs the expected action based on the chosen menu number as shown in Figure 1.

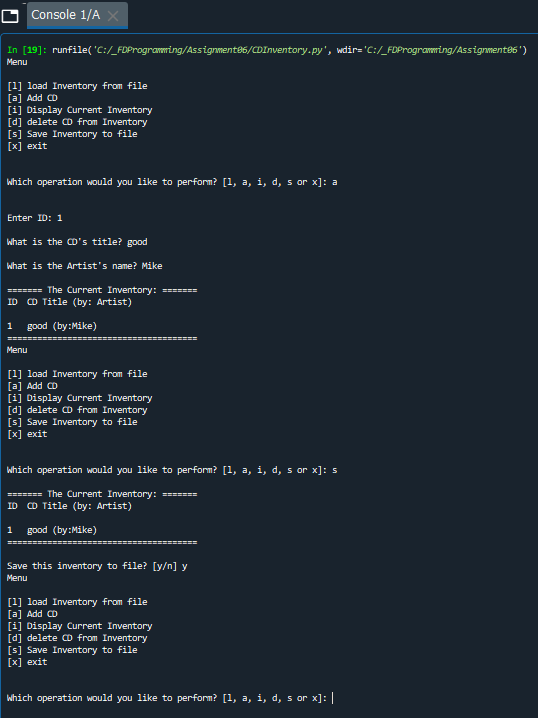


Figure 1 Result of running the AddressBook.py script in Spyder

I then ran the same script in the terminal window, which returned the same results as expected (Figure 2).

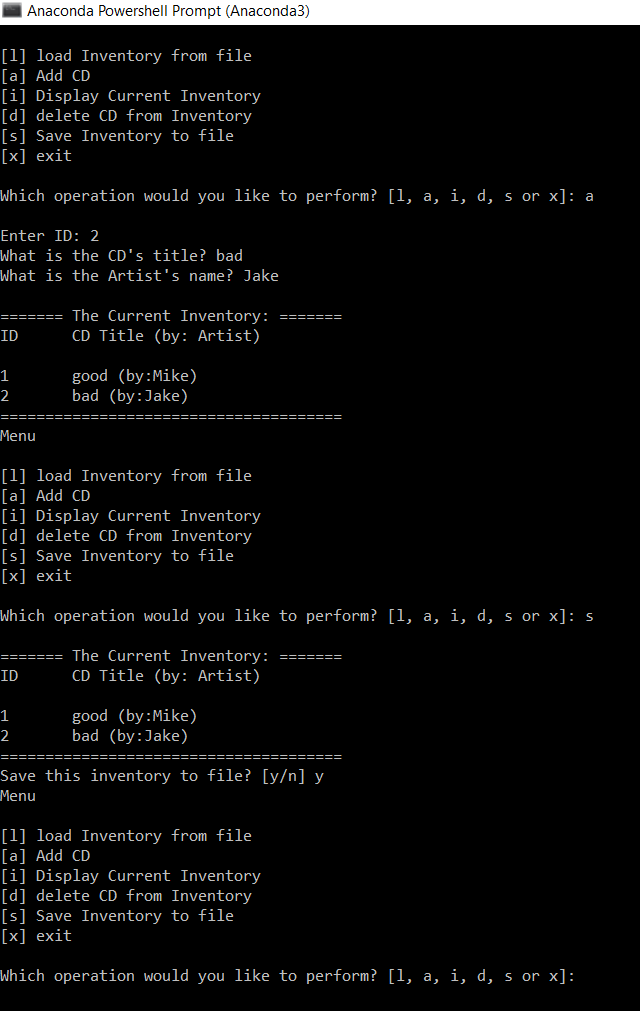


Figure 2 – Result of running the AddressBook.py script in terminal window

Finally, I located the “CDInventory.txt” in the folder and checked that the user inputs were successfully saved in there (Figure 3).

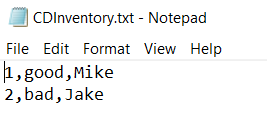


Figure 3 Contents of AddressBook.txt

# Summary

In this assignment, I practiced modifying an existing python program in Spyder, by replacing existing lines of code in the main body by functions for better organization. With the help of classes and functions, the program asks user to choose from several menu options, gets the user input data based on the chosen option and allocates it to a list table using dictionaries, displays and saves the data to a text file and loads data from the file. I tested the script successfully in both Spyder and terminal window.

While having pre-existing code may look like to have made the job a bit easier, spending time on adjusting the code after modifying various blocks is not straightforward, requiring significant amount of time and effort.

# Appendix

CDInventory.py

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. *# NSaber, 2021-Nov-20, Added Functionality*
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. *# TODone add functions for processing here*
20. **@staticmethod**
21. **def** add\_table(ID, title, artist):
22. """Function to process and add the user input data to a 2D list table (list of dicts)
24. Args:
25. **ID (string): CD ID input by the user**
26. title (string): Title of the CD input by the user
27. artist (string): Artist name input by the user
29. Returns:
30. **None.**
31. """
32. intID = int(ID)
33. dicRow = {'ID': intID, 'Title': title, 'Artist': artist}
34. lstTbl.append(dicRow)

37. @staticmethod
38. **def** row\_delete(IDDel):
39. """Function to search thru table and delete CD (row)
41. Args:
42. IDDel (string): CD ID to be deleted (input by the user)

45. **Returns:**
46. None.
47. """
48. intRowNr = -1
49. blnCDRemoved = False
50. **for row in lstTbl:**
51. intRowNr += 1
52. **if** row['ID'] == IDDel:
53. **del** lstTbl[intRowNr]
54. blnCDRemoved = True
55. **break**
56. **if** blnCDRemoved:
57. **print**('The CD was removed')
58. **else**:
59. **print**('Could not find this CD!')
61. **class** FileProcessor:
62. """Processing the data to and from text file"""
64. @staticmethod
65. **def read\_file(file\_name, table):**
66. """Function to manage data ingestion from file to a list of dictionaries
68. Reads the data from file identified by file\_name into a 2D table
69. (list of dicts) table one line in the file represents one dictionary row in table.
71. Args:
72. file\_name (string): name of file used to read the data from
73. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
75. **Returns:**
76. None.
77. """
78. table.clear() *# this clears existing data and allows to load data from file*
79. objFile = open(file\_name, 'r')
80. **for line in objFile:**
81. data = line.strip().split(',')
82. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
83. table.append(dicRow)
84. objFile.close()
86. @staticmethod
87. **def** write\_file(file\_name, table):
88. *# TODone Add code here*
89. """Function to write the data in list of dictionaries to a text file
91. Writes the data to file identified by file\_name from a 2D table
92. (list of dicts) table one line in the file represents one dictionary row in table.
94. Args:
95. **file\_name (string): name of file used to write the data to**
96. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
98. Returns:
99. None.
100. **"""**
101. objFile = open(strFileName, 'w')
102. **for** row **in** lstTbl:
103. lstValues = list(row.values())
104. lstValues[0] = str(lstValues[0])
105. **objFile.write(','.join(lstValues) + '\n')**
106. objFile.close()

109. *# -- PRESENTATION (Input/Output) -- #*
111. **class** IO:
112. """Handling Input / Output"""
114. @staticmethod
115. **def print\_menu():**
116. """Displays a menu of choices to the user
118. Args:
119. None.
121. Returns:
122. None.
123. """
125. **print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')**
126. **print**('[d] delete CD from Inventory**\n**[s] Save Inventory to file**\n**[x] exit**\n**')
128. @staticmethod
129. **def** menu\_choice():
130. **"""Gets user input for menu selection**
132. Args:
133. None.
135. **Returns:**
136. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
138. """
139. choice = ' '
140. **while choice not in ['l', 'a', 'i', 'd', 's', 'x']:**
141. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
142. **print**() *# Add extra space for layout*
143. **return** choice
145. **@staticmethod**
146. **def** show\_inventory(table):
147. """Displays current inventory table

150. **Args:**
151. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
153. Returns:
154. None.
156. """
157. **print**('======= The Current Inventory: =======')
158. **print**('ID**\t**CD Title (by: Artist)**\n**')
159. **for** row **in** table:
160. **print('{}\t{} (by:{})'.format(\*row.values()))**
161. **print**('======================================')
163. *# TODone add I/O functions as needed*
165. **@staticmethod**
166. **def** user\_input():
167. """Gets user input for CD
169. Args:
170. **None.**
172. Returns:
173. strID (string): CD ID input by the user
174. strTitle (string): Title of the CD input by the user
175. **stArtist (string): Artist name input by the user**
176. """
178. strID = input('Enter ID: ').strip()
179. strTitle = input('What is the CD**\'**s title? ').strip()
180. **stArtist = input('What is the Artist\'s name? ').strip()**
181. **print**() *# Add extra space for layout*
182. **return** strID, strTitle, stArtist
184. *# 1. When program starts, read in the currently saved Inventory*
185. **FileProcessor.read\_file(strFileName, lstTbl)**
187. *# 2. start main loop*
188. **while** True:
189. *# 2.1 Display Menu to user and get choice*
190. **IO.print\_menu()**
191. strChoice = IO.menu\_choice()
193. *# 3. Process menu selection*
194. *# 3.1 process exit first*
195. **if strChoice == 'x':**
196. **break**
197. *# 3.2 process load inventory*
198. **if** strChoice == 'l':
199. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
200. **strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')**
201. **if** strYesNo.lower() == 'yes':
202. **print**('reloading...')
203. FileProcessor.read\_file(strFileName, lstTbl)
204. IO.show\_inventory(lstTbl)
205. **else:**
206. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
207. IO.show\_inventory(lstTbl)
208. **continue** *# start loop back at top.*
209. *# 3.3 process add a CD*
210. **elif strChoice == 'a':**
211. *# 3.3.1 Ask user for new ID, CD Title and Artist*
212. *# TODone move IO code into function*
213. strID, strTitle, stArtist = IO.user\_input()
214. *# 3.3.2 Add item to the table*
215. ***# TODone move processing code into function***
217. DataProcessor.add\_table(strID, strTitle, stArtist)
218. IO.show\_inventory(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. *# TODone move processing code into function*
233. DataProcessor.row\_delete(intIDDel)
234. IO.show\_inventory(lstTbl)
235. **continue *# start loop back at top.***
236. *# 3.6 process save inventory to file*
237. **elif** strChoice == 's':
238. *# 3.6.1 Display current inventory and ask user for confirmation to save*
239. IO.show\_inventory(lstTbl)
240. **strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()**
241. *# 3.6.2 Process choice*
242. **if** strYesNo == 'y':
243. *# 3.6.2.1 save data*
244. *# TODone move processing code into function*
245. **FileProcessor.write\_file(strFileName, lstTbl)**
246. **else**:
247. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
248. **continue** *# start loop back at top.*
249. *# 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:*
250. **else:**
251. **print**('General Error')