Nasser Saber

December 05, 2021

Foundations of Programming, Python

Assignment 08

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

# Introduction

# In this assignment, I will explain the steps I used to create a Python Script for maintaining a CD inventory, using objects and classes.

# Creating the Script

I started by following the TODO guidelines and added private attributes, setters and getters as well as the \_\_str\_\_ methods to the CD class. This is to store the main CD data using the appropriate level of abstraction per standard practices. This followed by the FileIO class which handles the read from/write to text file using two @staticmethod’s. The challenge here was to manage the CD objects from reading from file or saving to file. The last class is IO where most of the user data input interaction happens.

In the main body, a cd objects gets instantiated using the user input data and the corresponding tasks gets done per user’s choice.

# Testing the Script

I ran the script in Spyder first. It runs smooth 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.

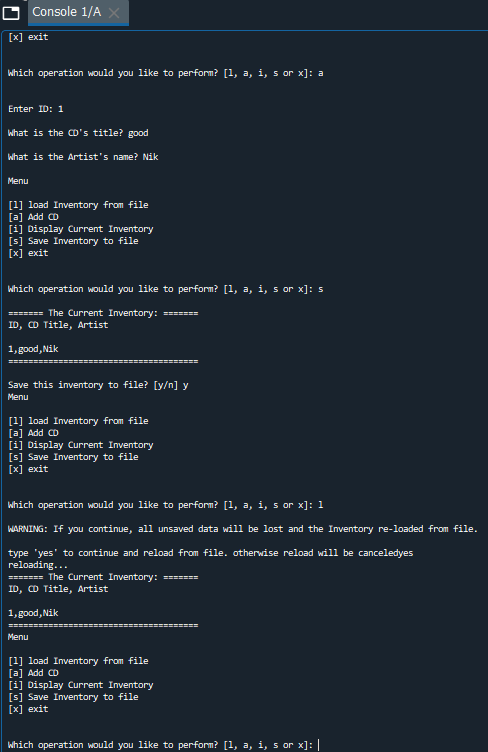


Figure 1 Result of running the AddressBook.py script in Spyder

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

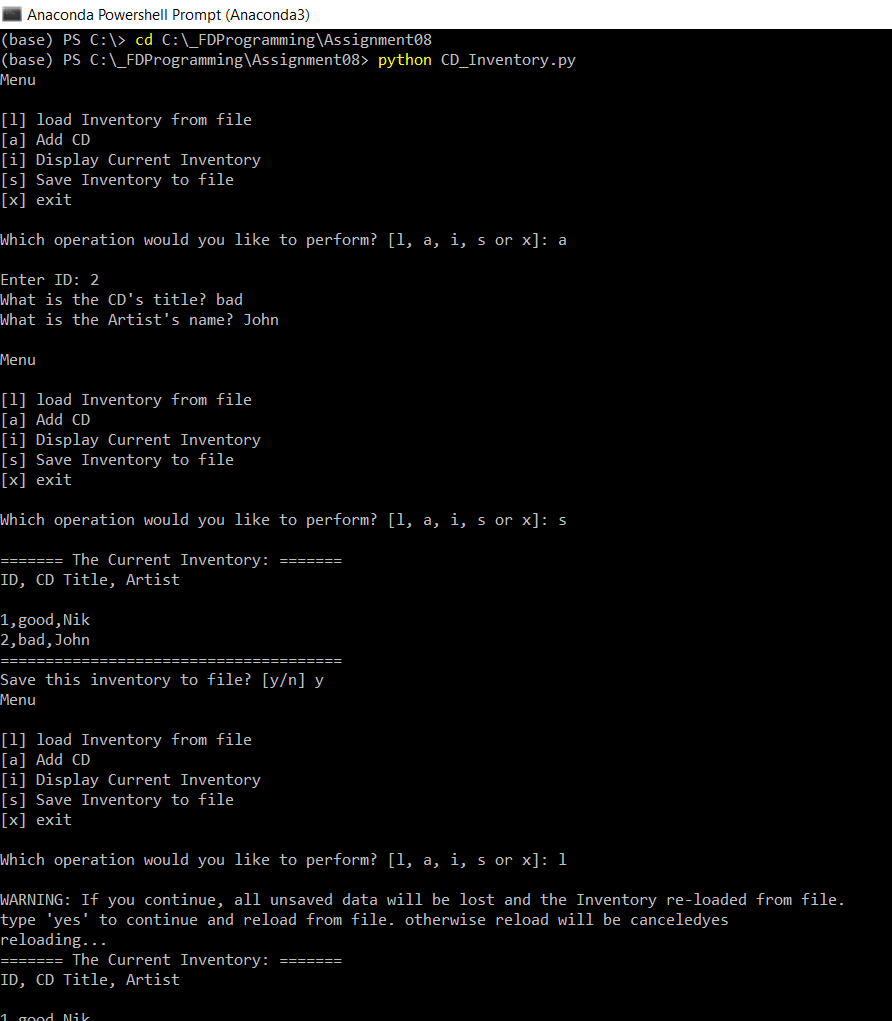


Figure 2 – Result of running the AddressBook.py script in terminal window

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

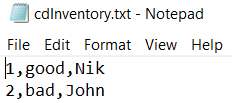


Figure 3 Contents of CDInventory.dat

# Summary

In this assignment, I practiced created a python program in Spyder, by taking advantage of the power of classes and objects, which is the core to OOP like Python. It was a challenging exercise since being the early exposure to such methods and concepts. Structured error handling was also implemented to deal with user and system errors where necessary.

# Appendix

GitHub

CD\_Inventory.py

1. *#------------------------------------------#*
2. *# Title: Assignmen08.py*
3. *# Desc: Assignnment 08 - Working with classes*
4. *# Change Log: (Who, When, What)*
5. ***# DBiesinger, 2030-Jan-01, created file***
6. *# DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08*
7. *# NSaber, 2021-Dec-05, Added Code and Functionality*
8. *#------------------------------------------#*
10. ***# -- DATA -- #***
11. strFileName = 'cdInventory.txt'
12. lstOfCDObjects = []
14. **class** CD:
15. **"""Stores data about a CD:**
17. properties:
18. cd\_id: (int) with CD ID
19. cd\_title: (string) with the title of the CD
20. **cd\_artist: (string) with the artist of the CD**
21. methods:
23. """
24. *# TODone Add Code to the CD class*
25. **def \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):**
26. self.\_\_cd\_id = cd\_id
27. self.\_\_cd\_title = cd\_title
28. self.\_\_cd\_artist = cd\_artist

31. @property
32. **def** cd\_id(self):
34. **return** self.\_\_cd\_id
36. @cd\_id.setter
37. **def** cd\_id(self, value):
38. **if** str(value).isnumeric():
40. **self.cd\_id = value**
41. **else**:
42. **raise** Exception('CD ID can**\'**t be alphabetic')
44. @property
45. **def cd\_title(self):**
47. **return** self.\_\_cd\_title
49. @cd\_title.setter
50. **def cd\_title(self, value):**
51. **if** str(value).isnumeric():
52. **raise** Exception('CD Title can**\'**t be cryptic')
53. **else**:
55. **self.cd\_title = value**

58. @property
59. **def** cd\_artist(self):
61. **return** self.\_\_cd\_artist
63. @cd\_artist.setter
64. **def** cd\_artist(self, value):
65. **if str(value).isnumeric():**
66. **raise** Exception('CD Artist can**\'**t be cryptic')
67. **else**:
69. self.cd\_artist = value
71. **def** \_\_str\_\_(self):
72. **return** self.cd\_id, self.cd\_title, self.cd\_artist





79. *# -- PROCESSING -- #*
80. **class FileIO:**
81. """Processes data to and from file:
83. properties:
85. **methods:**
86. save\_inventory(file\_name, lst\_Inventory): -> None
87. load\_inventory(file\_name): -> (a list of CD objects)
89. """
90. ***# TODone Add code to process data from a file***

93. @staticmethod
94. **def** load\_inventory(file\_name):
95. **"""Function to manage data ingestion from file to a list of objects**
97. Reads the data from file identified by file\_name into a 2D table
98. one line in the file represents one one CD object row in table.
100. **Args:**
101. file\_name (string): name of file used to read the data from
103. Returns:
104. None.
105. **"""**
106. table = []
107. table.clear() *# this clears existing data and allows to load data from file*
108. **try**:
109. **with** open(file\_name, 'r') **as** f:
110. **rows = [r.strip().split(',') for r in f.readlines()]**
111. table = [CD(\*row) **for** row **in** rows]

114. **except** FileNotFoundError **as** e:
115. **print('Text file does not exist!')**
116. **print**('Build in error info:')
117. **print**(type(e), e, sep='**\n**')
118. **return** table

121. *# TODone Add code to process data to a file*
122. @staticmethod
123. **def** save\_inventory(file\_name, lst\_Inventory):
124. *# TODone Add code here*
125. **"""Function to write the data in list of objects to a text file**
127. Writes the data to file identified by file\_name from a 2D table
128. one line in the file represents one CD object row in table.
130. **Args:**
131. file\_name (string): name of file used to write the data to
132. lst\_Inventory (list of objects): 2D data structure that holds the data during runtime
134. Returns:
135. **None.**
136. """
137. **try**:
138. objFile = open(file\_name, 'w')
139. **for** obj **in** lst\_Inventory:
140. **objFile.write(','.join(obj.\_\_str\_\_())+ '\n')**
141. objFile.close()
142. **except** IOError **as** e:
143. **print**('Text file does not exist or cannot be created!')
144. **print**('Build in error info:')
145. **print(type(e), e, sep='\n')**
147. *# -- PRESENTATION (Input/Output) -- #*
148. **class** IO:
149. *# TODone add docstring*
150. **"""Handling input/output:**
152. properties:
154. methods:
155. **print\_menu(): -> Print out menu options**
156. menu\_choice(): -> (user's choice, string)
157. show\_inventory(table) -> None
158. user\_input() -> (CD data input by user)
160. **"""**


164. *# TODone add code to show menu to user*
165. **@staticmethod**
166. **def** print\_menu():
167. """Displays a menu of choices to the user
169. Args:
170. **None.**
172. Returns:
173. None.
174. """
176. **print**('Menu**\n\n**[l] load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')
177. **print**('[s] Save Inventory to file**\n**[x] exit**\n**')
179. *# TODone add code to captures user's choice*
180. **@staticmethod**
181. **def** menu\_choice():
182. """Gets user input for menu selection
184. Args:
185. **None.**
187. Returns:
188. choice (string): a lower case sting of the users input out of the choices l, a, i, s or x
190. **"""**
191. choice = ' '
192. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
193. choice = input('Which operation would you like to perform? [l, a, i, s or x]: ').lower().strip()
194. **print**() *# Add extra space for layout*
195. **return choice**
197. *# TODone add code to display the current data on screen*
198. @staticmethod
199. **def** show\_inventory(table):
200. **"""Displays current inventory table**

203. Args:
204. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
206. Returns:
207. None.
209. """
210. **print('======= The Current Inventory: =======')**
211. **print**('ID, CD Title, Artist**\n**')
212. **for** obj **in** table:
213. **print**(','.join(obj.\_\_str\_\_()))
214. **print**('======================================')
215. ***# TODone add code to get CD data from user***
216. @staticmethod
217. **def** user\_input():
218. """Gets user input for CD
220. **Args:**
221. None.
223. Returns:
224. strID (string): CD ID input by the user
225. **strTitle (string): Title of the CD input by the user**
226. stArtist (string): Artist name input by the user
227. """
228. **while** True:
229. strID = input('Enter ID: ').strip()
230. **strTitle = input('What is the CD\'s title? ').strip()**
231. stArtist = input('What is the Artist**\'**s name? ').strip()
233. **print**() *# Add extra space for layout*
234. **if** strID.isnumeric():
235. **break**
236. **print**('CD ID should be an integer! Enter the data again:**\n**')
237. **return** strID, strTitle, stArtist
239. *# -- Main Body of Script -- #*
240. ***# TODone Add Code to the main body***
241. *# Load data from file into a list of CD objects on script start*
242. lstOfCDObjects = FileIO.load\_inventory(strFileName)
244. **while** True:
245. ***# Display menu to user***
246. IO.print\_menu()
247. strChoice = IO.menu\_choice()
248. *# show user current inventory*
249. **if** strChoice == 'i':
250. **IO.show\_inventory(lstOfCDObjects)**
251. **continue** *# start loop back at top.*
252. *# let user add data to the inventory*
253. **if** strChoice == 'a':
254. strID, strTitle, stArtist = IO.user\_input()
255. **cd = CD(strID, strTitle, stArtist)**
256. lstOfCDObjects.append(cd)
257. *# let user save inventory to file*
258. **if** strChoice == 's':
259. IO.show\_inventory(lstOfCDObjects)
260. **strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()**
261. *# 3.6.2 Process choice*
262. **if** strYesNo == 'y':
263. *# 3.6.2.1 save data*
264. *# TODone move processing code into function*
265. **FileIO.save\_inventory(strFileName, lstOfCDObjects)**
266. *# let user load inventory from file*
267. **if** strChoice == 'l':
268. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
269. strYesNo = input('type **\'**yes**\'** to continue and reload from file. otherwise reload will be canceled')
270. **if strYesNo.lower() == 'yes':**
271. **print**('reloading...')
272. lstOfCDObjects = FileIO.load\_inventory(strFileName)
273. IO.show\_inventory(lstOfCDObjects)
274. **else**:
275. **input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')**
276. IO.show\_inventory(lstOfCDObjects)
277. **continue** *# start loop back at top.*
278. *# let user exit program*
279. **elif** strChoice == 'x':
280. **break**