Ryan Goding

9/2/2020

IT FDN 110: Introduction to Programming (Python)

Assignment 08

# Introduction

Assignment 08 asks us to modify Assignment\_08\_Starter where the existing pseudocode indicates more code is required.

# Body

This assignment introduced us to Object Oriented Programming which was difficult to wrap my head around at first. The structure at first seems less intuitive than the assignments we have done before. I first started with the creation of the CD() class to store all of attributes that our inventory needs to keep track. This was done with setter and getters that set and get the properties of the CD object. Figure 1 shows CD\_Inventory.py in spyder.

Save and Load Inventory was mostly used from previous assignments, and the last function to create was the new\_cd() function. This will prompt the user for the ID, CD title, and Artist. Figure 3 and Figure 4 shows a new CD being added and then saved to the CDInventory.txt file. Both the CD() object and the new\_cd() are now to be used in the main body of the script. Figure 5 shows the load function being used from the command prompt.

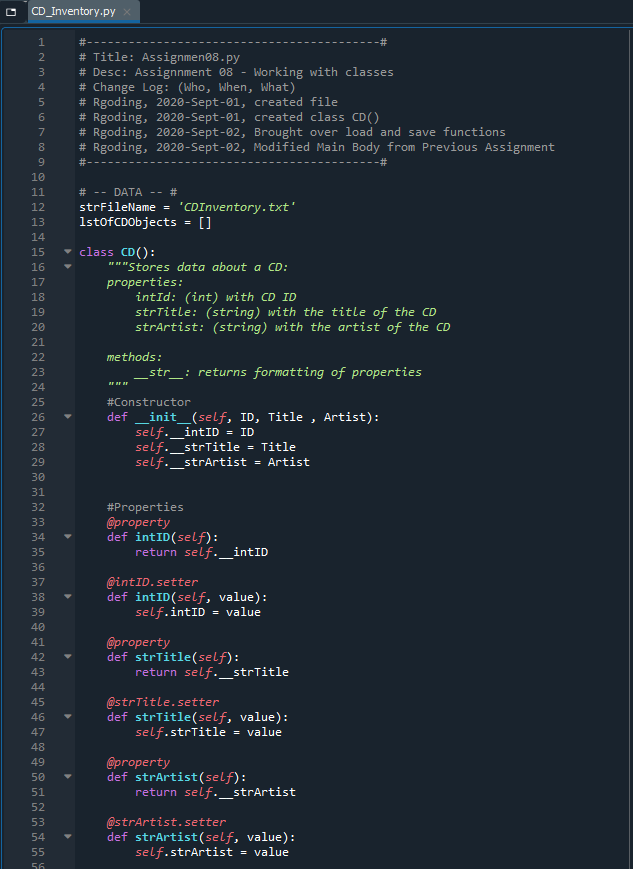


Figure 1: CD\_Inventory.py in Spyder

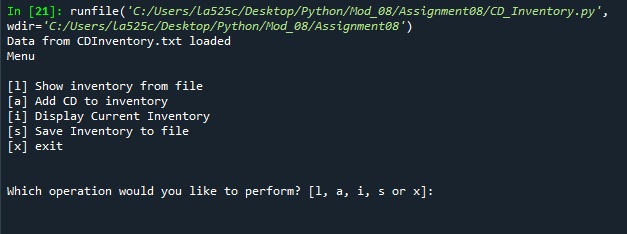


Figure 2: CD\_Inventory.py Script Run in Spyder, with no data file

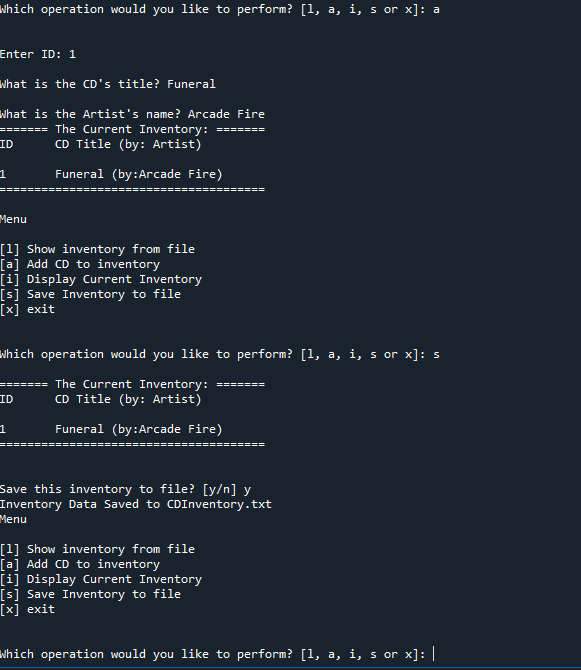


Figure 3: CD\_Inventory.py Adding New CD and Saving to CDInventory.txt

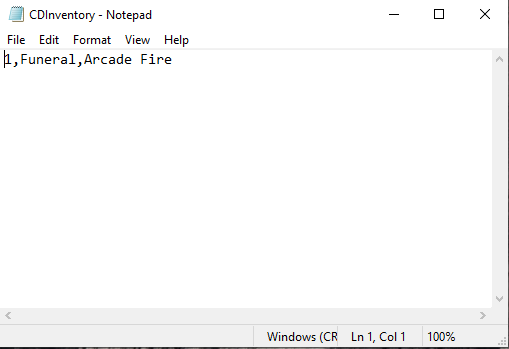


Figure - CDInventory.txt file after CD was added to Inventory

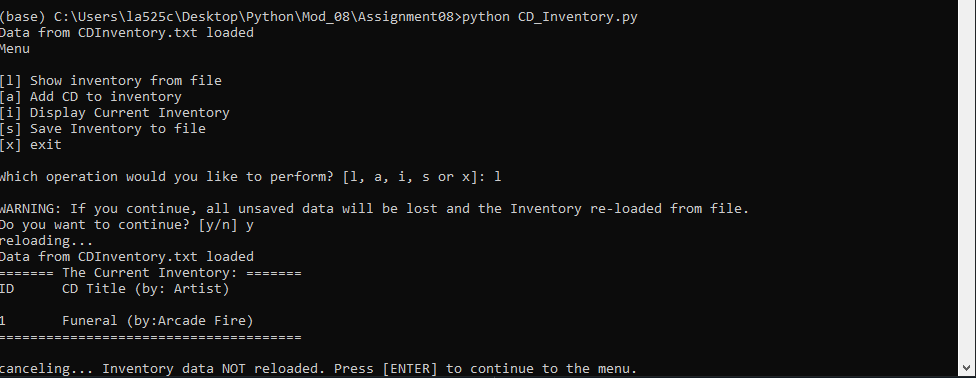


Figure 5: CD\_Inventory running in command window, loading Inventory from CDInventory.txt

This assignment also asked us to create an Assignment\_08 repository on Github to share our script and knowledge document. The link for my repository is https://github.com/rgoding/Assignment\_08.

# Summary

I found this assignment difficult because the object oriented programming was not as intuitive as past assignments. But after learning more about it, it does mirror the real world more in terms of objects that have properties, attributes, and methods and I imagine the more someone gets familiar with these concepts the easier it is to manage and create code.

# References

1. Programiz.com, “Python Object Oriented Programming”, <https://www.programiz.com/python-programming/object-oriented-programming>
2. Python-course.com, “Object Oriented Programming”, https://www.python-course.eu/python3\_object\_oriented\_programming.php

# Appendix

## Listing CD\_Inventory.py

1. #------------------------------------------#
2. # Title: Assignmen08.py
3. # Desc: Assignnment 08 - Working with classes
4. # Change Log: (Who, When, What)
5. # Rgoding, 2020-Sept-01, created file
6. # Rgoding, 2020-Sept-01, created class CD()
7. # Rgoding, 2020-Sept-02, Brought over load and save functions
8. # Rgoding, 2020-Sept-02, Modified Main Body from Previous Assignment
9. #------------------------------------------#
11. # -- DATA -- #
12. strFileName = 'CDInventory.txt'
13. lstOfCDObjects = []
15. **class** CD():
16. """Stores data about a CD:
17. properties:
18. intId: (int) with CD ID
19. strTitle: (string) with the title of the CD
20. strArtist: (string) with the artist of the CD
22. methods:
23. \_\_str\_\_: returns formatting of properties
24. """
25. #Constructor
26. **def** \_\_init\_\_(self, ID, Title , Artist):
27. self.\_\_intID = ID
28. self.\_\_strTitle = Title
29. self.\_\_strArtist = Artist

32. #Properties
33. @property
34. **def** intID(self):
35. **return** self.\_\_intID
37. @intID.setter
38. **def** intID(self, value):
39. self.intID = value
41. @property
42. **def** strTitle(self):
43. **return** self.\_\_strTitle
45. @strTitle.setter
46. **def** strTitle(self, value):
47. self.strTitle = value
49. @property
50. **def** strArtist(self):
51. **return** self.\_\_strArtist
53. @strArtist.setter
54. **def** strArtist(self, value):
55. self.strArtist = value


59. # -- PROCESSING -- #
60. **class** FileIO:
61. """Processes data to and from file:
62. properties:
63. methods:
64. save\_inventory(file\_name, lst\_Inventory): -> None
65. load\_inventory(file\_name): -> (a list of CD objects)
66. """
67. @staticmethod
68. **def** load\_inventory(file\_name, table):
69. """Function to manage data ingestion from file to a list of objects
71. Reads the data from file identified by file\_name into a 2D table
72. (list of objects) 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. with open(file\_name, 'r') as objFile:
83. **for** row **in** objFile:
84. data = row.strip().split(',')
85. CDobj = CD(data[0],data[1],data[2])
86. table.append(CDobj)
87. **print**('Data from ' + strFileName +' loaded')


91. @staticmethod
92. **def** save\_inventory(file\_name, table):
93. """Function to write data from the table to a file
95. Args:
96. file\_name (string): name of file used to read the data from
97. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
99. Returns:
100. None
101. """
103. with open(file\_name, 'w') as objFile:
104. **for** obj **in** table:
105. data = (str(obj.intID) + ',' + str(obj.strTitle) + ',' + str(obj.strArtist) + '\n')
106. objFile.write(data)
107. **print**('Inventory Data Saved to ' + strFileName)



112. # -- PRESENTATION (Input/Output) -- #
113. **class** IO:
114. """Handling Input / Output"""
116. @staticmethod
117. **def** print\_menu():
118. """Displays a menu of choices to the user
119. Args:
120. None.
121. Returns:
122. None.
123. """
125. **print**('Menu\n\n[l] Show inventory from file\n[a] Add CD to inventory')
126. **print**('[i] Display Current Inventory\n[s] Save Inventory to file\n[x] exit\n')

129. @staticmethod
130. **def** menu\_choice():
131. """Gets user input for menu selection
132. Args:
133. None.
134. Returns:
135. choice (string): a lower case string of the users input out of the choices l, a, i s or x
136. """
137. choice = ' '
138. **while** choice **not** **in** ['l', 'a', 'i', 's', 'x']:
139. choice = input('Which operation would you like to perform? [l, a, i, s or x]: ').lower().strip()
140. **print**()  # Add extra space for layout
141. **return** choice

144. @staticmethod
145. **def** show\_inventory(table):
146. """Displays current inventory table
147. Args:
148. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
149. Returns:
150. None.
151. """
152. **print**('======= The Current Inventory: =======')
153. **print**('ID\tCD Title (by: Artist)\n')
154. **for** obj **in** table:
155. **print**('{}\t{} (by:{})'.format(obj.intID,obj.strTitle,obj.strArtist))
156. **print**('======================================\n')
158. @staticmethod
159. **def** new\_cd():
160. """Function to prompt user for new CD attributes
162. Args:
163. strID: user input for CD ID
164. strTitle: CD title
165. strArtist: artist name
167. Returns:
168. intID, strTitle, strArtist
170. """
171. strID = input('Enter ID: ').strip()
172. strTitle = input('What is the CD\'s title? ').strip()
173. strArtist = input('What is the Artist\'s name? ').strip()
174. intID = int(strID)
175. **return** intID, strTitle, strArtist

178. # -- Main Body of Script -- #
179. # 1. When program starts, read in the currently saved Inventory
180. # Add Error exceptions to display error instead of closing program
181. **try**:
182. FileIO.load\_inventory(strFileName, lstOfCDObjects)
183. **except**:
184. **print**("No text file found in local folder, please create CDInventory.txt")
185. **while** True:
186. #Display Menu to user and get choice
187. IO.print\_menu()
188. strChoice = IO.menu\_choice()
190. **if** strChoice == 'x': # process exit request
191. **break**
193. **if** strChoice == 'l': # process load inventory from file
194. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
195. strYesNo = input('Do you want to continue? [y/n] ')
196. #Error Handling if file is not created
197. **try**:
198. **if** strYesNo.lower() == 'y':
199. **print**('reloading...')
200. FileIO.load\_inventory(strFileName, lstOfCDObjects)
201. IO.show\_inventory(lstOfCDObjects)
202. **except**:
203. **print**("CDInventory.txt does not exist in local directory")
204. **else**:
205. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
206. IO.show\_inventory(lstOfCDObjects)
207. **continue**  # start loop back at top.
209. **elif** strChoice == 'a': # process add a CD
210. #IO.add\_cd() will request user for input
211. intID, strTitle, strArtist = IO.new\_cd()
212. # Instantiate CD Class object
213. objCD = CD(intID, strTitle, strArtist)
214. lstOfCDObjects.append(objCD)
215. #Display Inventory
216. IO.show\_inventory(lstOfCDObjects)
217. **continue**
219. **elif** strChoice == 'i': # process display current inventory
220. IO.show\_inventory(lstOfCDObjects)
221. **continue**
223. **elif** strChoice == 's': # process save inventory to txt file
224. IO.show\_inventory(lstOfCDObjects) # Display current inventory and ask user for confirmation to save
225. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
226. #Process choice
227. **if** strYesNo == 'y':
228. #write data
229. FileIO.save\_inventory(strFileName, lstOfCDObjects)
230. **else**:
231. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
232. **continue**
233. # catch-all should not be possible, as user choice gets vetted in IO, but to be save:
234. **else**:
235. **print**('General Error')