Bryce Shepherd

2020-Sep -02

IT FDN 110 B SU 20

Assignment 07

CD\_Inventory

# Introduction

For this lesson we began the first steps into Object Orientated Programming (OOP), and we learned more about Classes. With these new skills we can create more flexible and secure code blocks. With this lesson the concept of OOP is still not fully making sense for me at the time of writing this doc. I also created struggles for myself in Lab08\_A which was built incorrectly.

## Topic 1

For first three labs I thought that I was building things correctly, as I was achieving the results of the lab. However, after the class session when we covered each lab, I realized that I had built the very first lab incorrectly.

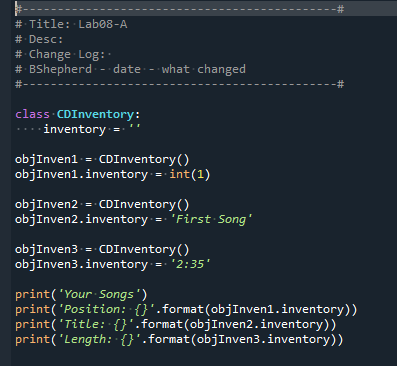


Figure - Incorrect Lab08\_A

The above code is what I originally built for the first lab, which will collect and present that data as the lab instructed. However, after seeing the way the lab was built during class made me realize that I needed to present/collect data differently which came into play in later parts of this lesson.

## Topic 2

With a few tests, I was able to figure out and understand Lab08\_B and Lab08\_C even building out the structure slightly incorrectly. The idea of how Constructor’s and Properties function and work together make sense. In addition, these are more widely and recommended to be used in place of Fields which function similarly.

## Topic 3

Lab08\_D is where I struggled the most in this lesson and this is due to how I originally built Lab08\_A. For this one I was struggling with handling the INT value, I was constantly getting an AttributeError but failed to find how to address this. I tried several changes/updates going mainly I thought that I needed to build a whole new Class to handle the INT values.



Figure - Lab08\_D Attempt

I realized that a new class was not needed, but still couldn’t find why I was still getting AttributeErrors. During class I was able to see the Lab08\_D which helped in the understanding of where I was making my mistake.

## Topic 4

Lastly overall this was laying some of the groundwork for OOP however, I am not certain if I understand this at its core quite yet. I understand that this is a way of allowing your code to be flexible and protect certain values you want to remain as a set value. But I feel like there is something that I am missing but don’t know what it is on this particular lesson.

# Summary

For the core of this lesson we started viewing OOP and extended functionality of the Classes. The different functions of Classes are starting to make sense. I feel that I need quite a bit more practice with OOP to fully use and implement this type of coding effectively.

# Appendix

Using [PlanetB’s](http://www.planetb.ca/syntax-highlight-word) (external reference) web-page[[1]](#footnote-1)

Added code block to [GitHub](https://github.com/shepherdbwork/Assignment_08)[[2]](#footnote-2)

1. #------------------------------------------#
2. # Title: CD\_Inventory.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. # BShepherd, 2020-Aug-31, Begin edit to template, adding Class CD
8. # BShepherd, 2020-Sep-01, Continued edit to Class FileIO
9. #------------------------------------------#
11. # -- DATA -- #
12. **import** pickle
13. strFileName = 'cdInventory.txt'
14. lstOfCDObjects = []
16. **class** CD:
17. """Stores data about a CD:
19. properties:
20. cd\_id: (int) with CD ID
21. cd\_title: (string) with the title of the CD
22. cd\_artist: (string) with the artist of the CD
23. methods:
24. """
25. **def** \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):
26. self.\_\_CDid = cd\_id
27. self.\_\_title = cd\_title
28. self.\_\_artist = cd\_artist
30. @property
31. **def** CDid(self):
32. **return** self.\_\_CDid
34. @CDid.setter
35. **def** CDid(self, value):
36. **if** type(value) == int:
37. self.\_\_CDid = value
38. **else**:
39. **raise** Exception('This value needs to be set as an Integer!')
41. @property
42. **def** title(self):
43. **return** self.\_\_title
45. @title.setter
46. **def** title(self, value):
47. **if** type(value) == str:
48. self.\_\_title = value
49. **else**:
50. **raise** Exception('This value needs to be set as a String!')
52. @property
53. **def** artist(self):
54. **return** self.\_\_artist
56. @artist.setter
57. **def** artist(self, value):
58. **if** type(value) == str:
59. self.\_\_artist = value
60. **else**:
61. **raise** Exception('This value needs to be set as a String!')

64. # -- PROCESSING -- #
65. **class** FileIO:
66. """Processes data to and from file:
68. properties:
70. methods:
71. save\_inventory(file\_name, lst\_Inventory): -> None
72. load\_inventory(file\_name): -> (a list of CD objects)
74. """
75. @staticmethod
76. **def** load\_file(file\_name):
77. **try**:
78. objFile = open(file\_name, 'rb')
79. data = pickle.load(objFile)
80. file\_name.extend(data)
81. objFile.close()
82. **except** FileNotFoundError as e:
83. **print**('===================================================')
84. **print**(type(e), e, e.\_\_doc\_\_, sep='\n')
85. **print**('You will need to use the Add function to Continue')
86. **print**('===================================================\n')
87. **except** EOFError as e:
88. **print**('===================================================')
89. **print**(type(e), e, e.\_\_doc\_\_, sep='\n')
90. **print**('Your file is Empty please use Add to add to your file :\)')
91. **print**('===================================================\n')
92. # TODO Add code to process data to a file
93. @staticmethod
94. **def** write\_file(file\_name):
95. objFile = open(file\_name, 'wb')
96. pickle.dump(objFile)
97. objFile.close()

100. # -- PRESENTATION (Input/Output) -- #
101. **class** IO:
102. @staticmethod
103. **def** print\_menu():
104. """Displays a menu of choices to the user
106. Args:
107. None.
109. Returns:
110. None.
112. """
113. **print**('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
114. **print**('[s] Save Inventory to file\n[x] exit\n')
116. @staticmethod
117. **def** menu\_choice():
118. """Gets user input for menu selection
120. Args:
121. None.
123. Returns:
124. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
126. """
127. choice = ' '
128. **while** choice **not** **in** ['l', 'a', 'i', 's', 'x']:
129. choice = input('Which operation would you like to perform? [l, a, i, s or x]: ').lower().strip()
130. **print**()  # Add extra space for layout
131. **return** choice
133. @staticmethod
134. **def** show\_inventory(table):
135. """Displays current inventory table

138. Args:
139. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
141. Returns:
142. None.
144. """
145. **print**('======= The Current Inventory: =======')
146. **print**('ID\tCD Title (by: Artist)\n')
147. **for** row **in** table:
148. **print**(row)
149. **print**('======================================')
151. @staticmethod
152. **def** add\_info():
153. """ Function to add the CD info to a file
155. Args:
156. None
158. Returns:
159. # None
160. strID: ID input by user to use as an identifier
161. strTitle: Second value captured to identify the Title of an album
162. stArtist: Thrid value captured to identify the Artest of the album
164. """
166. strID = input('Enter ID: ').strip()
167. strTitle = input('What is the CD\'s title? ').strip()
168. strArtist = input('What is the Artist\'s name? ').strip()
169. **return**(strID, strTitle, strArtist)
171. # -- Main Body of Script -- #
172. # Load data from file into a list of CD objects on script start
173. FileIO.load\_file(strFileName)
174. # Display menu to user
175. **while** True:
176. IO.print\_menu()
177. strChoice = IO.menu\_choice()
178. # let user exit program
179. **if** strChoice == 'x':
180. **break**
181. # show user current inventory
182. **if** strChoice == 'l':
183. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
184. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')
185. **if** strYesNo.lower() == 'yes':
186. **print**('reloading...')
187. lst = FileIO.load\_file(strFileName)
188. IO.show\_inventory(lstOfCDObjects)
189. **else**:
190. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
191. IO.show\_inventory(lstOfCDObjects)
192. **continue**  # start loop back at top.
193. # let user add data to the inventory
194. **elif** strChoice == 'a':
195. strID, strTitle, strArtist = IO.add\_info()
196. infoAdd = (strID, strTitle, strArtist)
197. lstOfCDObjects.append(infoAdd)
198. IO.show\_inventory(lstOfCDObjects)
199. # let user save inventory to file
200. **elif** strChoice == 's':
201. IO.show\_inventory(lstOfCDObjects)
202. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
203. **if** strYesNo == 'y':
204. FileIO.write\_file(strFileName)
205. **else**:
206. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
207. **continue**  # start loop back at top.
208. # let user load inventory from file
209. **else** strChoice == 'i':
210. IO.show\_inventory(lstOfCDObjects)
211. **continue**
212. **else**:
213. **print**('General Error')

1. Retrieved 2020-Sep-02 [↑](#footnote-ref-1)
2. Retrieved 2020-Sep -02 [↑](#footnote-ref-2)