Justin Roe

09/04/2020

IT FDN 110 B

Assignment 08

Assignment 08

# Introduction

Assignment 8 introduced the “essence” of object oriented programming in Python. The class construct was discussed, then fields, constructors, properties, and methods of objects were reviewed. This was certainly the most difficult assignment so far (at least, for me). Although I struggled with grasping functions, I had a very difficult time with “dunder” objects (still do), as well as the message setter and getter statements. My primary thought is; why do you need the property? It appears all “actions” are being performed in the setter, getter, and method sections of the class. For the knowledge document, I had the most difficulty switching from utilizing dictionaries to CD objects; however,

# Knowledge Application / Details

See Figures 1-3 for screenshots of the code execution.

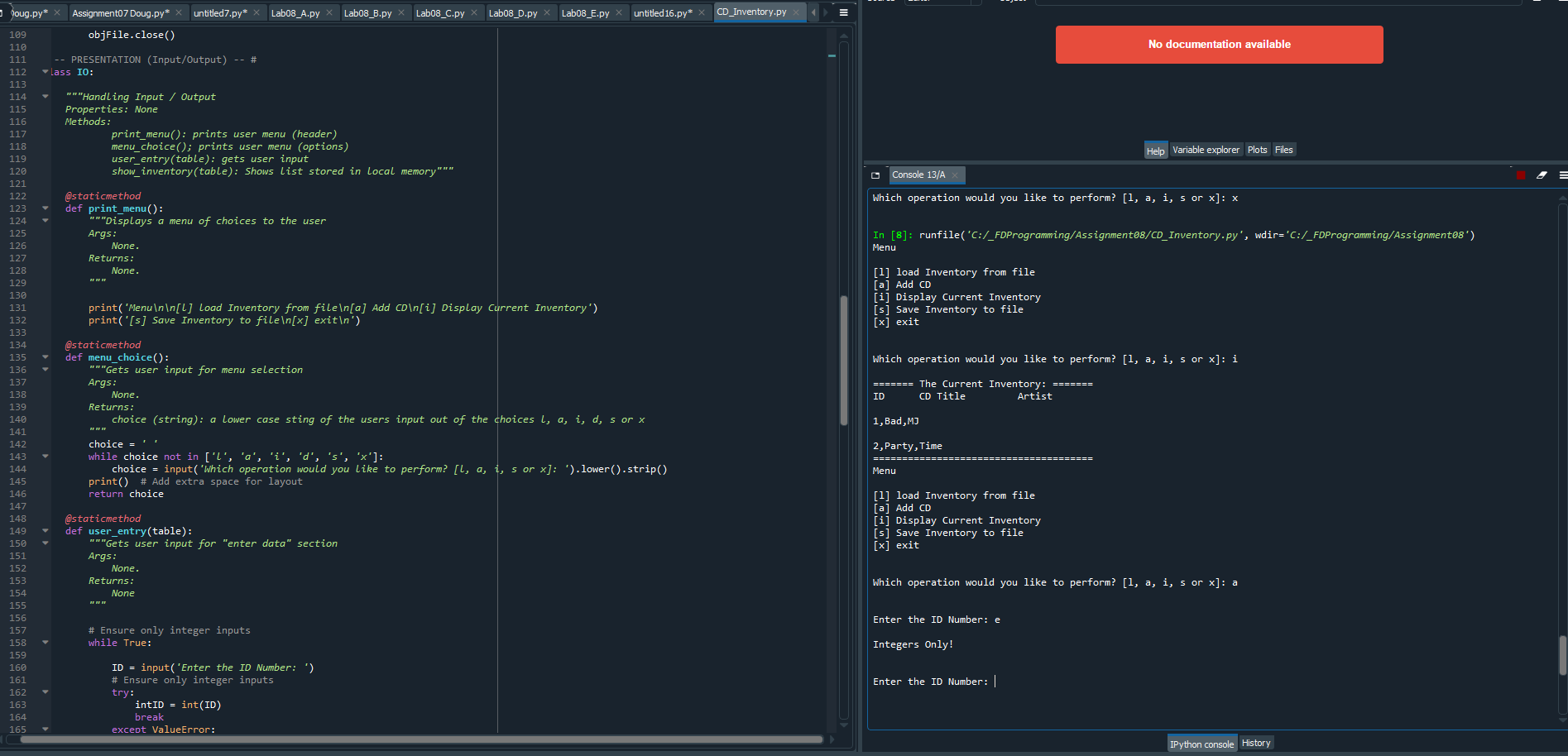


Figure – Representative Spyder run of CDInventory.py script

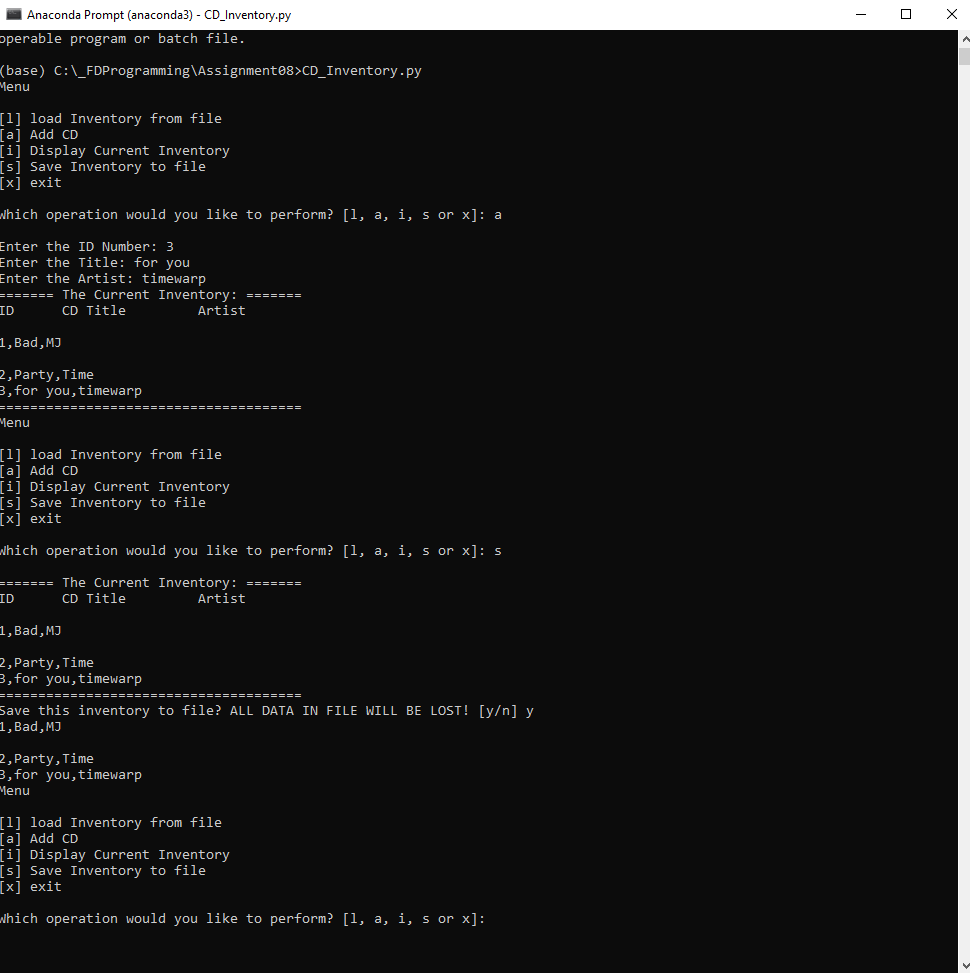


Figure – Representative image of CDInventory.py running in console

# Summary

I had a very difficult time with this assignment until I asked a few questions regarding general structure of the assignment – I didn’t use many external references aside from one to aid saving / loading lists to files [(here)](https://stackabuse.com/reading-and-writing-lists-to-a-file-in-python/)[[1]](#footnote-1). Other than that, I recycled old code from previous assignments, relied heavily on the lectures, and watched the videos recommended in the module instructions.

I chose to load data into the program upon startup as this is how I interpreted the following instruction: “Load data from file into a list of CD objects on script start”; however, I know this was not how we were doing in on assignments 6 and 7. I included a .txt file in my zip file as a “starter”. In addition, I couldn’t call exceptions in position setters when executing code – I’m thinking this was a result of my dunderstring statement (which I found useful when displaying the user input) – so I simply added standard error handling.

# Appendix A – Syntax

Generated using [planetb’s webpage](http://planetb.ca/syntax-highlight-word) (external reference)[[2]](#footnote-2) web page

1. #------------------------------------------#
2. # Title: Assignmen08.py
3. # Desc: Assignnment 08 - Working with classes
4. # Change Log: (Who, When, What)
5. # JRoe, 2020-Aug-31, created file
6. # JRoe, 2020-Sep-04, Added error handling, changed data type to list (of CD objects), added annotation / comments, bolstered docstrings
7. #------------------------------------------#
9. # -- DATA -- #
10. strFileName = 'cdInventory.txt'
11. lstOfCDObjects = []
12. open(strFileName, 'a').close() #Creates blank file
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. #Constructor
25. **def** \_\_init\_\_(self, pos, titl, artist):
27. #Attributes
28. self.\_\_position = pos
29. self.\_\_title = titl
30. self.\_\_artist = artist
32. #Properties
34. @property
35. **def** position(self):
36. **return** self.\_\_position
38. @position.setter
39. **def** position(self, value):
40. **if** type(value) == int:
41. self.\_\_position = value
42. **else**:
43. **raise** Exception('Integers Only!')
45. @property
46. **def** title(self):
47. **return** self.\_\_title
49. @title.setter
50. **def** title(self, value):
51. **if** type(value) == str:
52. self.\_\_title = value
53. **else**:
54. **raise** Exception('String Only!')
56. @property
57. **def** artist(self):
58. **return** self.\_\_artist
60. @artist.setter
61. **def** artist(self, value):
62. **if** type(value) == str:
63. self.\_\_artist = value
64. **else**:
65. **raise** Exception('String Only!')
66. **def** \_\_str\_\_(self):
67. **return** '{},{},{}'.format(self.\_\_position, self.\_\_title, self.\_\_artist)

70. # -- PROCESSING -- #
71. **class** FileIO:
72. """Processes data to and from file:
74. properties:
76. methods:
77. save\_fxn(file\_name, lst\_Inventory): -> None
78. read\_file(file\_name): -> (a list of CD objects)
80. """
81. @staticmethod
82. **def** read\_file(file\_name, table):
83. """Function to read list objects (CD objects) from file
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. lstOfCDObjects.append(line)
91. objFile.close()

94. @staticmethod
95. **def** save\_fxn(file\_name, table):
96. """Function to save entered data to designated file
97. Args:
98. None.
99. Returns:
100. None.
101. """
103. objFile = open(file\_name, 'w')
105. **for** row **in** table:
106. strrow = str(row)
107. **print**(strrow)
108. objFile.write(strrow + '\n')
109. objFile.close()
111. # -- PRESENTATION (Input/Output) -- #
112. **class** IO:
114. """Handling Input / Output
115. Properties: None
116. Methods:
117. print\_menu(): prints user menu (header)
118. menu\_choice(); prints user menu (options)
119. user\_entry(table): gets user input
120. show\_inventory(table): Shows list stored in local memory"""
122. @staticmethod
123. **def** print\_menu():
124. """Displays a menu of choices to the user
125. Args:
126. None.
127. Returns:
128. None.
129. """
131. **print**('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
132. **print**('[s] Save Inventory to file\n[x] exit\n')
134. @staticmethod
135. **def** menu\_choice():
136. """Gets user input for menu selection
137. Args:
138. None.
139. Returns:
140. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
141. """
142. choice = ' '
143. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
144. choice = input('Which operation would you like to perform? [l, a, i, s or x]: ').lower().strip()
145. **print**()  # Add extra space for layout
146. **return** choice
148. @staticmethod
149. **def** user\_entry(table):
150. """Gets user input for "enter data" section
151. Args:
152. None.
153. Returns:
154. None
155. """
157. # Ensure only integer inputs
158. **while** True:
160. ID = input('Enter the ID Number: ')
161. # Ensure only integer inputs
162. **try**:
163. intID = int(ID)
164. **break**
165. **except** ValueError:
166. **print**('\nIntegers Only!\n')
167. **continue**
168. Title = input('Enter the Title: ')
169. Artist = input('Enter the Artist: ')
170. first = CD(intID, Title, Artist)
171. table.append(first)
173. @staticmethod
174. **def** show\_inventory(table):
175. """Displays current inventory table
176. Args:
177. table that holds the data during runtime.
178. Returns:
179. None.
180. """
181. **print**('======= The Current Inventory: =======')
182. **print**('ID\tCD Title\t Artist\n')
183. **for** row **in** table:
184. **print**(row)
185. **print**('======================================')

188. # -- Main Body of Script -- #
190. # Display menu to user
191. # show user current inventory
192. # let user add data to the inventory
193. # let user save inventory to file
194. # let user load inventory from file
195. # let user exit program
197. # 1. When program starts, read in the currently saved Inventory
199. FileIO.read\_file(strFileName, lstOfCDObjects)
201. # 2. start main loop
202. **while** True:
203. # 2.1 Display Menu to user and get choice
204. IO.print\_menu()
205. strChoice = IO.menu\_choice()
206. # 3. Process menu selection
207. # 3.1 process exit first
208. **if** strChoice == 'x':
209. **break**
210. # 3.2 process load inventory
211. **if** strChoice == 'l':
212. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
213. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')
214. **if** strYesNo.lower() == 'yes':
215. **print**('reloading...')
216. FileIO.read\_file(strFileName, lstOfCDObjects)
217. IO.show\_inventory(lstOfCDObjects)
218. **else**:
219. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
220. IO.show\_inventory(lstOfCDObjects)
221. **continue**  # start loop back at top.
223. # 3.3 process add a CD
224. **elif** strChoice == 'a':
225. # 3.3.1 Ask user for new ID, CD Title and Artist
226. IO.user\_entry(lstOfCDObjects)
227. # 3.3.2 Add itiem to the table
229. IO.show\_inventory(lstOfCDObjects)
230. **continue**  # start loop back at top.
232. # 3.4 process display current inventory
233. **elif** strChoice == 'i':
234. IO.show\_inventory(lstOfCDObjects)
235. **continue**  # start loop back at top.
236. # 3.5 process delete a CD

239. # 3.6 process save inventory to file
240. **elif** strChoice == 's':
241. # 3.6.1 Display current inventory and ask user for confirmation to save
242. IO.show\_inventory(lstOfCDObjects)
243. strYesNo = input('Save this inventory to file? ALL DATA IN FILE WILL BE LOST! [y/n] ').strip().lower()
244. # 3.6.2 Process choice
245. **if** strYesNo == 'y':
246. # 3.6.2.1 save data
247. FileIO.save\_fxn(strFileName, lstOfCDObjects)
248. **else**:
249. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
250. **continue**  # start loop back at top.
252. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
253. **else**:
254. **print**('General Error')

1. Retrieved Sept 04 2020 [↑](#footnote-ref-1)
2. Retrieved 2020-Aug-26 [↑](#footnote-ref-2)