Bryce Shepherd

2020-Aug-18

IT FDN 110 B SU 20

Assignment 06

CD Inventory

# Introduction

For this assignment we are learning how to use Functions, Class and DocStrings. With Functions we were able to create a central set of code, that can be called one or multiple times within a code block. Along with the functions we also have Class that allows us to nest a series of functions to be called later in the code. Lastly DocStrings, these allow us to leave an example of what a given function or class does that can be reviewed throughout a code block.

## Topic 1

For the most part how functions work clicked right away for me. I was able to understand that these are essentially groups of code that can be called to run multiple times. Also, that when a function is called it goes directly back to where the function was called. These allow us to pass in multiple values, the one part that caught me off guard was remembering that functions are also Tuples. One area I struggled with was when I tried to call the ID as an INT I was getting an error, remembering that the Tuples don’t work with INT’s I went back and reviewed the material on unpacking tuples.

## Topic 2

Like functions classes made sense right away Classes allow us to group different functions into logical groups. This allows a programmer to identify specific actions being taken by a user of the program.

## Topic 3

Lastly DocStrings are a way for us as programmers to leave ourselves and future programmers notes for our functions and what they are intended to do. Docstrings are like comments in this way where we can identify what something is doing, unlike comments though these are able to be viewed anytime you are able to see the function. However, I think that these docstrings in my code are relatively basic, this feels like a learned skill that will take time to understand the right amount of information to add in a docstring.

# Summary

This lesson taught us how to better structure our code and make logical steps in code to ensure the user goes through all the intended steps. For future work using functions allow us to not have to re-write the same thing multiple times. Classes allow us to better organize the work. Docstrings allow us to better track what certain code blocks do.

# 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/Learning_Python_Basics/blob/master/Mod_06) [[2]](#footnote-2)

1. #------------------------------------------#
2. # Title: Assignment06\_Starter.py
3. # Desc: Working with classes and functions.
4. # Change Log: (Who, When, What)
5. # BShepherd, 2020-Aug-17, Created File
6. # BShepherd, 2020-Aug-17, Completed TODO tasks
7. # BShepherd, 2020-Aug-18, Added DocString's
8. #------------------------------------------#
10. # -- DATA -- #
11. strChoice = '' # User input
12. lstTbl = []  # list of lists to hold data
13. dicRow = {}  # list of data row
14. strFileName = 'CDInventory.txt'  # data storage file
15. objFile = None  # file object
16. strID = None
17. strTitle = None
18. stArtist = None
19. intID = None

22. # -- PROCESSING -- #
23. **class** DataProcessor:
25. @staticmethod
26. **def** add\_cdinfo(intID, strTitle, stArtist):
27. """ Function to add the CD info to a file
29. Args:
30. intID: First value captured to identify the row
31. strTitle: Second value captured to identify the Title of an album
32. stArtist: Thrid value captured to identify the Artest of the album
34. Returns:
35. The added rows to your list.
37. """
38. intID = int(strID)
39. dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
40. lstTbl.append(dicRow)
42. @staticmethod
43. **def** del\_item(table):
44. """ Function to delete CD info from the list
46. Args:
47. Table: 2D Table that holds data during runtime(lstTbl).
49. Returns:
50. Removes the entered row from the lstTbl.
52. """
53. intRowNr = -1
54. blnCDRemoved = False
55. **for** row **in** lstTbl:
56. intRowNr += 1
57. **if** row['ID'] == intIDDel:
58. **del** lstTbl[intRowNr]
59. blnCDRemoved = True
60. **break**
61. **if** blnCDRemoved:
62. **print**('The CD was removed')
63. **else**:
64. **print**('Could not find this CD!')
66. **class** FileProcessor:
67. """Processing the data to and from text file"""
69. @staticmethod
70. **def** read\_file(file\_name, table):
71. """Function to manage data ingestion from file to a list of dictionaries
73. Reads the data from file identified by file\_name into a 2D table
74. (list of dicts) table one line in the file represents one dictionary row in table.
76. Args:
77. file\_name (string): name of file used to read the data from
78. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
80. Returns:
81. None.
82. """
83. table.clear()  # this clears existing data and allows to load data from file
84. objFile = open(file\_name, 'r')
85. **for** line **in** objFile:
86. data = line.strip().split(',')
87. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
88. table.append(dicRow)
89. objFile.close()
91. @staticmethod
92. **def** write\_file(file\_name, table):
93. """ Function to add the CD info to a file
95. Args:
96. file\_name: File to read data from
97. Table: 2D Table that holds data during runtime(lstTbl)
99. Returns:
100. None
102. """
103. objFile = open(strFileName, 'w')
104. **for** row **in** lstTbl:
105. lstValues = list(row.values())
106. lstValues[0] = str(lstValues[0])
107. objFile.write(','.join(lstValues) + '\n')
108. 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.
124. """
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\tCD Title (by: Artist)\n')
159. **for** row **in** table:
160. **print**('{}\t{} (by:{})'.format(\*row.values()))
161. **print**('======================================')

164. @staticmethod
165. **def** cd\_info(strID, strTitle, stArtist):
166. """ Function to add the CD info to a file
168. Args:
169. strID: ID input by user to use as an identifier
170. strTitle: Second value captured to identify the Title of an album
171. stArtist: Thrid value captured to identify the Artest of the album
173. Returns:
174. None
176. """
177. strID = input('Enter ID: ').strip()
178. strTitle = input('What is the CD\'s title? ').strip()
179. stArtist = input('What is the Artist\'s name? ').strip()
180. **return**(strID, strTitle, stArtist)
182. # 1. When program starts, read in the currently saved Inventory
183. FileProcessor.read\_file(strFileName, lstTbl)
185. # 2. start main loop
186. **while** True:
187. # 2.1 Display Menu to user and get choice
188. IO.print\_menu()
189. strChoice = IO.menu\_choice()
190. # 3. Process menu selection
191. # 3.1 process exit first
192. **if** strChoice == 'x':
193. **break**
195. # 3.2 process load inventory
196. **if** strChoice == 'l':
197. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
198. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')
199. **if** strYesNo.lower() == 'yes':
200. **print**('reloading...')
201. FileProcessor.read\_file(strFileName, lstTbl)
202. IO.show\_inventory(lstTbl)
203. **else**:
204. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
205. IO.show\_inventory(lstTbl)
206. **continue**  # start loop back at top.
208. # 3.3 process add a CD
209. **elif** strChoice == 'a':
210. # 3.3.1 Ask user for new ID, CD Title and Artist
211. strID, strTitle, stArtist = IO.cd\_info(strID, strTitle, stArtist)
212. # 3.3.2 Add item to the table
213. DataProcessor.add\_cdinfo(intID, strTitle, stArtist)
214. IO.show\_inventory(lstTbl)
215. **continue**  # start loop back at top.
217. # 3.4 process display current inventory
218. **elif** strChoice == 'i':
219. IO.show\_inventory(lstTbl)
220. **continue**  # start loop back at top.
222. # 3.5 process delete a CD
223. **elif** strChoice == 'd':
224. # 3.5.1 get Userinput for which CD to delete
225. # 3.5.1.1 display Inventory to user
226. IO.show\_inventory(lstTbl)
227. # 3.5.1.2 ask user which ID to remove
228. intIDDel = int(input('Which ID would you like to delete? ').strip())
229. # 3.5.2 search thru table and delete CD
230. DataProcessor.del\_item(lstTbl)
231. IO.show\_inventory(lstTbl)
232. **continue**  # start loop back at top.
234. # 3.6 process save inventory to file
235. **elif** strChoice == 's':
236. # 3.6.1 Display current inventory and ask user for confirmation to save
237. IO.show\_inventory(lstTbl)
238. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
239. # 3.6.2 Process choice
240. **if** strYesNo == 'y':
241. FileProcessor.write\_file(strFileName, lstTbl)
242. **else**:
243. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
244. **continue**  # start loop back at top.
246. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
247. **else**:
248. **print**('General Error')

1. Retrieved 2020-Aug-18 [↑](#footnote-ref-1)
2. Retrieved 2020-Aug-18 [↑](#footnote-ref-2)