Nitasha Woodward

November 21, 2021

IT FDN 110 B Au 21: Foundations of Programming: Python

Assignment 06

Working with Functions and Classes

# Introduction

In this assignment I will explain the steps I used to edit a CD Inventory program to create new functions within pre-existing classes to better organize the code and adhere to ‘Separation of Concerns’ formatting. This program incorporates reading and writing data to and from text files, creating and calling functions, and docstrings.

# Creating the Code

I read through the code first to try and understand where was going to be the easiest place to start in completing the #TODO items. I skipped over the #TODO’s in the classes initially because I didn’t feel they provided enough context on what needed to happen without reading through the main program. Upon my initial run of the program, I received an error that the text file did not exist. To address this, without changing the read\_file function, I added 2 lines of code in step #1 to open/create a file using the append (a) mode.

### Adding a CD

I moved the I/O code for choice a – ‘Add CD’ first. I created a function called ‘get\_cd’ and copied and pasted the 3 input statements into the function. I created a docstring to describe what the function does and added a ‘return’ value after the input statements so that I could return the 3 values (ID, CD Title, Artist) back to my main program.

I struggled with how to call the function and whether I was supposed to put variables into the parenthesis when calling. Inputting values in the parenthesis did not work, so I left them blank. I’m not sure if there should be any arguments listed here or not because we are asking for the user input within the function. Will test when I have more time but would be helpful to discuss.

I also thought there was an issue with my get\_cd function initially because when I was testing this choice, I was being asked twice for CD information. However, what was wrong was in the main program in step 3.3.1 I listed:

IO.get\_cd( )

﻿ strID, strArtist, strTitle = IO.get\_cd( )

which was calling the function twice [I think]. I thought I needed to call the function on its own before being able to unpack it, but this seems to be untrue. I deleted the first call of the function and this function seemed to be operating properly.

I then moved the code to actually add the CD into the lstTbl list of dictionaries into the DataProcessing class. By unpacking the tuple results from step 3.3.1, I was able to use the variables in 3.3.2 to create a dictionary row and append to the list, lstTbl.

### Deleting a CD

I created a new function called ‘del\_cd’ in the DataProcessing class and moved the code that was previously under step 3.5.2 up into this function. I added a variable name to the print statements so that a message could be returned to the user after they tried to delete a CD from the inventory.

### Saving Inventory to the Text File

I updated function called ‘write\_file’ in the FileProcessor class with the code suggested in the feedback on my Assignment 05 submission [I wanted to test it out]. I then replaced the code in 3.6.2.1 with the updated ‘write\_file’ function.

### Completing the #TODO’s in the Classes

As I completed the #TODO’s in the main program, I found that I was simultaneously satisfying the #TODO’s in the classes and therefore did not need to do anything additional for these to be completed

### Other Issues

I do not recall encountering the { } and .format in the materials? I am using it because it was provided as a suggestion in my last homework assignment feedback. I was also unable to figure out how/where to automate the numbering of the ID’s. Is this something we can cover?

# Saving the Script

After writing and testing my code in Spyder, I navigated to the folder in my Home Folder called ‘Python’, in which I created a folder for ‘Assignment06’ and saved my text file as CDInventory.py.

# Running the Script

Once I saved my final script, I ran the program successful in Spyder as well as my Terminal.

Text

Description automatically generated

Figure - Testing Code in Spyder [1 of 4] – Add CD

Text

Description automatically generated

Figure - Testing Code in Spyder [2 of 4] – Add CD + Display Inventory

Text

Description automatically generated

Figure - Testing Code in Spyder [3 of 4] Saving, Loading, + Deleting

Text

Description automatically generated

Figure - Testing Code in Spyder [4 of 4] Deleting + Exiting

Text

Description automatically generated

Figure - Testing in the Terminal [1 of 3]

Text

Description automatically generated with medium confidence

Figure - Testing in the Terminal [2 of 3]

Text

Description automatically generated with low confidence

Figure - Testing in the Terminal [3 of 3]

# Summary

After completing the assigned reading, labs, and videos in Module 06 I was able to edit and execute a script where I created and utilized functions and docstrings to better format and document the CD Inventory program. I have posted the assignment .py file and this document to GitHub for peer review.

# Appendix

## <https://highlight.hohli.com/index.php> (external[[1]](#footnote-1)) web page

1. *#------------------------------------------#*
2. *# Title: Assignment06\_Starter.py*
3. *# Desc: Working with classes and functions.*
4. *# Change Log: (Who, When, What)*
5. ***# DBiesinger, 2030-Jan-01, Created File***
6. *# NWoodward, 2021-Nov-21, Updated lines marked #DONE*
7. *#------------------------------------------#*
9. *# -- DATA -- #*
10. **strChoice = '' *# User input***
11. lstTbl = [] *# list of lists to hold data*
12. dicRow = {} *# list of data row*
13. strFileName = 'CDInventory.txt' *# data storage file*
14. objFile = None *# file object*

17. *# -- PROCESSING -- #*
18. **class** DataProcessor:
19. """ Processing data """
20. ***# DONE add functions for processing here***
22. @staticmethod
23. **def** add\_cd(intID, strTitle, strArtist):
24. """ Function that allows user to add a CD to the inventory in memory. The CD must be
25. **saved, choice 's', in order for the CD to be written to the text file.**
27. Args:
28. intID (string): ID number entered by user
29. strArtist (string): Artist name input by user
30. **strRitle (string): CD title input by user**
32. Returns:
33. lstTbl: list of dictionaries that contains CD data. Each dictionary represents one CD.
34. """
35. **intID = int(strID)**
36. dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}
37. lstTbl.append(dicRow)
38. **return** lstTbl
40. **@staticmethod**
41. **def** del\_cd(table, ID):
42. """ Function that allows user to delete a CD from the inventory in memory.
44. Args:
45. **table (list of dict): 2D data structure (list of dicts) that holds the data during runtime**
46. ID: ID number of the CD the user would like to delete
48. Returns:
49. message: Message to the user to tell them if their attempt to delete a CD was successful
50. **"""**
51. intRowNr = -1
52. blnCDRemoved = False
53. **for** row **in** lstTbl:
54. intRowNr += 1
55. **if row['ID'] == intIDDel:**
56. **del** lstTbl[intRowNr]
57. blnCDRemoved = True
58. **break**
59. **if** blnCDRemoved:
60. **message = print('The CD was removed')**
61. **else**:
62. message = **print**('Could not find this CD!')
63. **return** message
65. **class FileProcessor:**
66. """Processing the data to and from text file"""
68. @staticmethod
69. **def** read\_file(file\_name, table):
70. **"""Function to manage data ingestion from file to a list of dictionaries**
72. Reads the data from file identified by file\_name into a 2D table
73. (list of dicts) table one line in the file represents one dictionary row in table.
75. **Args:**
76. file\_name (string): name of file used to read the data from
77. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
79. Returns:
80. **None.**
81. """
82. table.clear() *# this clears existing data and allows to load data from file*
83. objFile = open(file\_name, 'r')
84. **for** line **in** objFile:
85. **data = line.strip().split(',')**
86. dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
87. table.append(dicRow)
88. objFile.close()
90. **@staticmethod**
91. **def** write\_file(file\_name, table):
92. """Function to write data from a 2D list to a text file
94. Writes data from 2D list (list of dictionaries) identified by lstTbl into a text file indentified
95. **by strFileName. One dictionary represents one line of data in the text file.**

98. Args:
99. file\_name (string): name fo file used to write data to
100. **table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.**
102. Returns:
103. None.
104. """
105. ***# DONE Add code here***
106. objFile = open(strFileName, 'w')
107. **for** row **in** lstTbl:
108. strRow = '{},{},{}**\n**'
109. strRow = strRow.format(row['ID'], row['Title'], row['Artist'])
110. **objFile.write(strRow)**
111. objFile.close()
113. *# -- PRESENTATION (Input/Output) -- #*
115. **class IO:**
116. """Handling Input / Output"""
118. @staticmethod
119. **def** print\_menu():
120. **"""Displays a menu of choices to the user**
122. Args:
123. None.
125. **Returns:**
126. None.
127. """
129. **print**('Menu**\n\n**[l] load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')
130. **print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')**
132. @staticmethod
133. **def** menu\_choice():
134. """Gets user input for menu selection
136. Args:
137. None.
139. Returns:
140. **choice (string): a lower case string 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, d, s or x]: ').lower().strip()
145. **print() *# Add extra space for layout***
146. **return** choice
148. @staticmethod
149. **def** show\_inventory(table):
150. **"""Displays current inventory table**
152. Args:
153. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
155. **Returns:**
156. None.
157. """
158. **print**('======= The Current Inventory: =======')
159. **print**('ID**\t**CD Title (by: Artist)**\n**')
160. **for row in table:**
161. **print**('{}**\t**{} (by:{})'.format(\*row.values()))
162. **print**('======================================')
164. *# DONE add I/O functions as needed*
166. @staticmethod
167. **def** get\_cd(ID = None, Artist = None, Title = None):
168. """ Function to enable user to add a new CD to in memory to a list of dictionaries
170. **Args:**
171. strID (string): ID number of the CD input by user
172. strArtist (string): Artist name input by user
173. strRitle (string): CD title input by user
175. **Returns:**
176. strID (sting): ID number entered by user
177. strArtist (string): Artist name input by user
178. strRitle (string): CD title input by user
180. **"""**
181. strID = input('Enter ID: ').strip()
182. strTitle = input('What is the CD**\'**s title? ').strip()
183. strArtist = input('What is the Artist**\'**s name? ').strip()
184. **return** strID, strTitle, strArtist

187. *# 1. When program starts, read in the currently saved Inventory*
188. objFile = open(strFileName, 'a') *# Added to 'create' file the first time program is run*
189. objFile.close()
190. **FileProcessor.read\_file(strFileName, lstTbl)**
192. *# 2. start main loop*
193. **while** True:
194. *# 2.1 Display Menu to user and get choice*
195. **IO.print\_menu()**
196. strChoice = IO.menu\_choice()
198. *# 3. Process menu selection*
199. *# 3.1 process exit first*
200. **if strChoice == 'x':**
201. **break**
202. *# 3.2 process load inventory*
203. **if** strChoice == 'l':
204. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
205. **strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')**
206. **if** strYesNo.lower() == 'yes':
207. **print**('reloading...')
208. FileProcessor.read\_file(strFileName, lstTbl)
209. IO.show\_inventory(lstTbl)
210. **else:**
211. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
212. IO.show\_inventory(lstTbl)
213. **continue** *# start loop back at top.*
214. *# 3.3 process add a CD*
215. **elif strChoice == 'a':**
216. *# 3.3.1 Ask user for new ID, CD Title and Artist*
217. *# DONE move IO code into function*
218. strID, strArtist, strTitle = IO.get\_cd()
219. **print**(strID, strArtist, strTitle)
221. *# 3.3.2 Add item to the table*
222. *# DONE move processing code into function*
223. DataProcessor.add\_cd(strID, strArtist, strTitle)
224. IO.show\_inventory(lstTbl)
225. **continue *# start loop back at top.***
226. *# 3.4 process display current inventory*
227. **elif** strChoice == 'i':
228. IO.show\_inventory(lstTbl)
229. **continue** *# start loop back at top.*
230. ***# 3.5 process delete a CD***
231. **elif** strChoice == 'd':
232. *# 3.5.1 get Userinput for which CD to delete*
233. *# 3.5.1.1 display Inventory to user*
234. IO.show\_inventory(lstTbl)
235. ***# 3.5.1.2 ask user which ID to remove***
236. intIDDel = int(input('Which ID would you like to delete? ').strip())
237. *# 3.5.2 search thru table and delete CD*
238. *# DONE move processing code into function*
239. DataProcessor.del\_cd(lstTbl,intIDDel)
240. **print() *# Add extra space for layout***
241. IO.show\_inventory(lstTbl)
242. **continue** *# start loop back at top.*
243. *# 3.6 process save inventory to file*
244. **elif** strChoice == 's':
245. ***# 3.6.1 Display current inventory and ask user for confirmation to save***
246. IO.show\_inventory(lstTbl)
247. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
248. *# 3.6.2 Process choice*
249. **if** strYesNo == 'y':
250. ***# 3.6.2.1 save data***
251. *# DONE move processing code into function*
252. FileProcessor.write\_file(strFileName, lstTbl)
253. **else**:
254. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
255. **continue *# start loop back at top.***
256. *# 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:*
257. **else**:
258. **print**('General Error')

1. Accessed November 21, 2021 [↑](#footnote-ref-1)