Bryce Shepherd

2020-Aug-24

IT FDN 110 B SU 20

Assignment 07

CDInventory

# Introduction

With this lesson we have tackled learning to save in a different file type with the pickle method. This allows us to store data in a .dat file rather than a .txt file which has more flexibility for later file/data structures. Additionally, we have learned how to add in error handling allowing our code to account for different types of generic errors and not crash.

## Topic 1

For this lesson one of the first things we learned about is [pickling](https://docs.python.org/3/library/pickle.html)[[1]](#footnote-1), this is a new form of saving data to a binary file. This form of saving doesn’t add in any encryption but what this does allow for is more condensed file forms. We are also able to store more data than we would be able to store in a normal .txt file.

## Topic 2

One struggle I had with this one is getting a starting file to build off of, I found a [Tutorials Point](https://www.tutorialspoint.com/python-pickling)[[2]](#footnote-2) this allowed me to build some test files to create a base file. I have found that what works for me is to take small bits of code and put them into another script to run that to see how it functions.

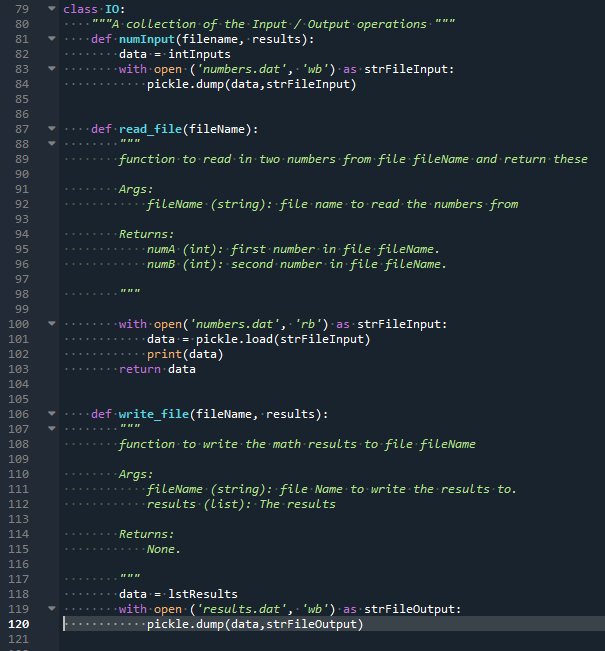


Figure - Pickle Functions

## Topic 3

Lab07-B was the biggest hurdle in this lesson, I believe this was difficult because I completed lesson 6 incorrectly. In the previous lesson instead of asking for inputs outside of the function, I was asking for the user inputs within the function. After testing with various places to add the user inputs, I finally landed on putting it outside of the function which worked as intended.

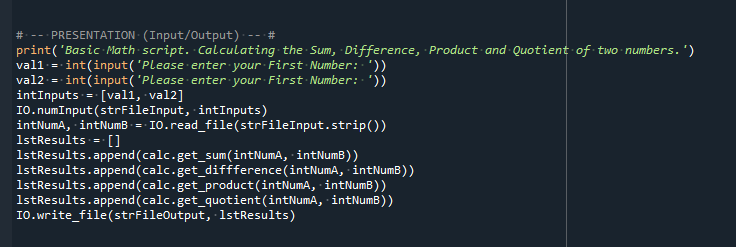


Figure - Added code for Lab07\_B

## Topic 4

In Lab07-C adding in error handling was very easy to understand and review. One thing that I tested this with after successfully adding in error handling was removing the TRY and EXCEPT out of a while loop. This helped build error handling for when a file doesn’t exist directing the user to use a specific function. One item error that I managed to get was an [EOFError](https://docs.python.org/3/library/exceptions.html)[[3]](#footnote-3), looking for this I found that my file had no data. I added this to my assignment, as part of the error handler to help address this issue.

# Summary

This lesson was interesting as we were able to learn a new format of data saving. This new method allows us to save more complex data sets. We also learned about error handling, which has been something I have been looking forward to learning more about.

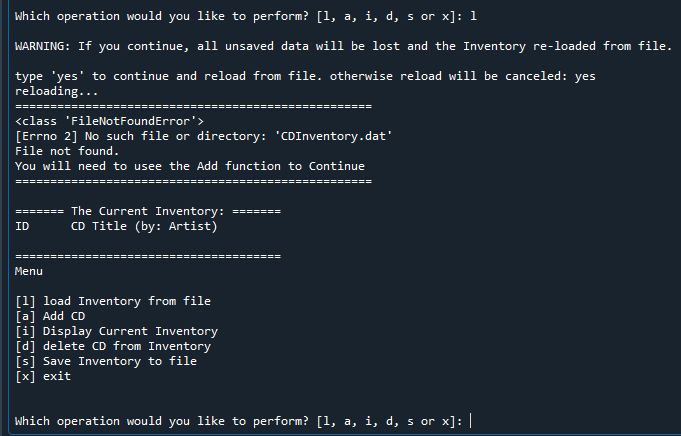


Figure - Error Handling in Spyder

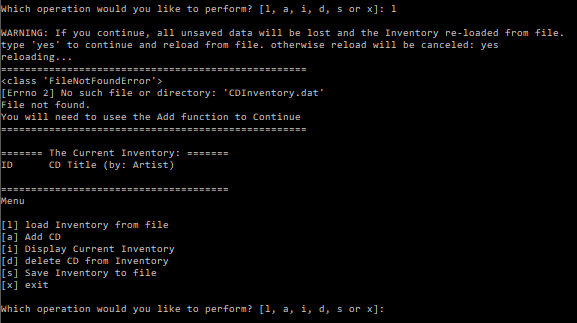


Figure - Error Handling in Terminal

# Appendix

Using [PlanetB’s](http://www.planetb.ca/syntax-highlight-word) (external reference) web-page[[4]](#footnote-4)

Added code block to [GitHub](https://github.com/shepherdbwork/Assignment_07)[[5]](#footnote-5)

1. #------------------------------------------#
2. # Title: CDInventory.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. # BShepherd, 2020-Aug-23, Updated the script to process DAT file type
9. # BShepherd, 2020-Aug-24, Adding in additional error handling
10. #------------------------------------------#
12. # -- DATA -- #
13. **import** pickle
14. strChoice = '' # User input
15. lstTbl = []  # list of lists to hold data
16. dicRow = {}  # list of data row
17. strFileName = 'CDInventory.dat'  # data storage file
18. objFile = None  # file object

21. # -- PROCESSING -- #
22. **class** DataProcessor:
24. @staticmethod
25. **def** add\_cdinfo(str, strTitle, stArtist, table):
26. """ Function to add the CD info to a file
28. Args:
29. intID: First value captured to identify the row
30. strTitle: Second value captured to identify the Title of an album
31. stArtist: Thrid value captured to identify the Artest of the album
33. Returns:
34. # Without a return statement in the function, it's not actually returning anything.
35. # The added rows to your list.
36. None
38. """
39. # We had to change intID to strID in the function declaration to make this line valid.
40. intID = int(strID)
41. dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
42. table.append(dicRow)
44. @staticmethod
45. **def** del\_item(id\_to\_remove, table):
46. """ Function to delete CD info from the list
48. Args:
49. Table: 2D Table that holds data during runtime(lstTbl).
51. Returns:
52. # Removes the entered row from the lstTbl.
53. None
55. """
56. **try**:
57. intRowNr = -1
58. blnCDRemoved = False
59. **for** row **in** table:
60. intRowNr += 1
61. **if** row['ID'] == id\_to\_remove:
62. **del** table[intRowNr]
63. blnCDRemoved = True
64. **break**
65. **except** ValueError as e:
66. **print**('===================================================')
67. **print**(type(e), e, e.\_\_doc\_\_, sep='\n')
68. **print**('Please enter an integer to delete a row')
69. **print**('===================================================\n')
70. **if** blnCDRemoved:
71. **print**('The CD was removed')
72. **else**:
73. **print**('Could not find this CD!')
75. **class** FileProcessor:
76. """Processing the data to and from text file"""
78. @staticmethod
79. **def** read\_file(file\_name, table):
80. """Function to manage data ingestion from file to a list of dictionaries
82. Reads the data from file identified by file\_name into a 2D table
83. (list of dicts) table one line in the file represents one dictionary row in table.
85. Args:
86. file\_name (string): name of file used to read the data from
87. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
89. Returns:
90. None.
91. """
92. table.clear()  # this clears existing data and allows to load data from file
93. **try**:
94. objFile = open(file\_name, 'rb')
95. data = pickle.load(objFile)
96. table.extend(data)
97. objFile.close()
98. **except** FileNotFoundError as e:
99. **print**('===================================================')
100. **print**(type(e), e, e.\_\_doc\_\_, sep='\n')
101. **print**('You will need to usee the Add function to Continue')
102. **print**('===================================================\n')
103. **except** EOFError as e:
104. **print**('===================================================')
105. **print**(type(e), e, e.\_\_doc\_\_, sep='\n')
106. **print**('Your file is Empty please use Add to add to your file :\)')
107. **print**('===================================================\n')
109. @staticmethod
110. **def** write\_file(file\_name, table):
111. """ Function to add the CD info to a file
113. Args:
114. file\_name: File to read data from
115. Table: 2D Table that holds data during runtime(lstTbl)
117. Returns:
118. None
120. """
121. objFile = open(file\_name, 'wb')
122. pickle.dump(table, objFile)
123. objFile.close()
125. # -- PRESENTATION (Input/Output) -- #
127. **class** IO:
128. """Handling Input / Output"""
130. @staticmethod
131. **def** print\_menu():
132. """Displays a menu of choices to the user
134. Args:
135. None.
137. Returns:
138. None.
140. """
141. **print**('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
142. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
144. @staticmethod
145. **def** menu\_choice():
146. """Gets user input for menu selection
148. Args:
149. None.
151. Returns:
152. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
154. """
155. choice = ' '
156. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
157. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
158. **print**()  # Add extra space for layout
159. **return** choice
161. @staticmethod
162. **def** show\_inventory(table):
163. """Displays current inventory table

166. Args:
167. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
169. Returns:
170. None.
172. """
173. **print**('======= The Current Inventory: =======')
174. **print**('ID\tCD Title (by: Artist)\n')
175. **for** row **in** table:
176. **print**('{}\t{} (by:{})'.format(\*row.values()))
177. **print**('======================================')

180. @staticmethod
181. # You do not need to pass any values to this function.
182. **def** cd\_info():
183. """ Function to add the CD info to a file
185. Args:
186. None
188. Returns:
189. # None
190. strID: ID input by user to use as an identifier
191. strTitle: Second value captured to identify the Title of an album
192. stArtist: Thrid value captured to identify the Artest of the album
194. """
196. strID = input('Enter ID: ').strip()
197. strTitle = input('What is the CD\'s title? ').strip()
198. stArtist = input('What is the Artist\'s name? ').strip()
199. **return**(strID, strTitle, stArtist)
201. # 1. When program starts, read in the currently saved Inventory
202. FileProcessor.read\_file(strFileName, lstTbl)
204. # 2. start main loop
205. **while** True:
206. # 2.1 Display Menu to user and get choice
207. IO.print\_menu()
208. strChoice = IO.menu\_choice()
209. # 3. Process menu selection
210. # 3.1 process exit first
211. **if** strChoice == 'x':
212. **break**
214. # 3.2 process load inventory
215. **if** strChoice == 'l':
216. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
217. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')
218. **if** strYesNo.lower() == 'yes':
219. **print**('reloading...')
220. FileProcessor.read\_file(strFileName, lstTbl)
221. IO.show\_inventory(lstTbl)
222. **else**:
223. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
224. IO.show\_inventory(lstTbl)
225. **continue**  # start loop back at top.
227. # 3.3 process add a CD
228. **elif** strChoice == 'a':
229. # 3.3.1 Ask user for new ID, CD Title and Artist
230. strID, strTitle, stArtist = IO.cd\_info()
231. # 3.3.2 Add item to the table
232. DataProcessor.add\_cdinfo(strID, strTitle, stArtist, lstTbl)
233. IO.show\_inventory(lstTbl)
234. **continue**  # start loop back at top.
236. # 3.4 process display current inventory
237. **elif** strChoice == 'i':
238. IO.show\_inventory(lstTbl)
239. **continue**  # start loop back at top.
241. # 3.5 process delete a CD
242. **elif** strChoice == 'd':
243. # 3.5.1 get Userinput for which CD to delete
244. # 3.5.1.1 display Inventory to user
245. IO.show\_inventory(lstTbl)
246. # 3.5.1.2 ask user which ID to remove
247. intIDDel = int(input('Which ID would you like to delete? ').strip())
248. # 3.5.2 search thru table and delete CD
249. DataProcessor.del\_item(intIDDel, lstTbl)
250. IO.show\_inventory(lstTbl)
251. **continue**  # start loop back at top.
253. # 3.6 process save inventory to file
254. **elif** strChoice == 's':
255. # 3.6.1 Display current inventory and ask user for confirmation to save
256. IO.show\_inventory(lstTbl)
257. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
258. # 3.6.2 Process choice
259. **if** strYesNo == 'y':
260. FileProcessor.write\_file(strFileName, lstTbl)
261. **else**:
262. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
263. **continue**  # start loop back at top.
265. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
266. **else**:
267. **print**('General Error')

1. Retrieved 2020-Aug-22 [↑](#footnote-ref-1)
2. Retrieved 2020-Aug-23 [↑](#footnote-ref-2)
3. Retrieved 2020-Aug-23 [↑](#footnote-ref-3)
4. Retrieved 2020-Aug-23 [↑](#footnote-ref-4)
5. Retrieved 2020-Aug-23 [↑](#footnote-ref-5)