Rogger Tovar

11/28/2021

Foundations of Python

Assignment 07

CD Inventory Exception and Pickling

# Introduction

This project of mine has many errors I couldn’t fix. This week’s project modifying week 6 assignment by adding exceptions and pickling the file to binary. What I thought was going to be an easy assignment, turned out to be harder that I thought. The labs didn’t work out for me. I really couldn’t progress past lab A; lab B doesn’t work without A, so lab C was out of the question. The progress of this assignment was slow. I manage to complete about half of the assignment, but mostly spent most of the time troubleshooting my own errors and unable to fix the mistakes. I skipped and just had to move on, or I would spend too much time in one section.

# CD Inventory with exceptions

The idea of adding Exceptions was simple to start and finished off tedious. I will quickly note that I managed to add the try/except exception rather quickly by running the code and catching which is the first official input made by the user; it was the menu. In the menu, the user can add whatever they want without notice they entered the wrong input. So looked for the menu in the code and added the try, then except where needed. I ran the code and already shown in Figure 1.

Text

Description automatically generated

Figure - Added try/except to the menu option

Text

Description automatically generated

Idle - “General Error” appeared

Idle 1 displays the error when entering the wrong key information. The next option I tackled was the user input when deleting an ID. In the Figure 2, try/except statement was added and executed properly in image of Idle 2.

Text

Description automatically generated

Figure

Text

Description automatically generated

Idle successful exception method

In Figure 3 is where I began to run into trouble, I just couldn’t figure out how to loop it properly so the end user could stop giving the incorrect input. In Idle 3, the shows, the program does not continue, but it never shows my “except” print message.

Text

Description automatically generated

Figure saving the file

Text

Description automatically generated

Idle Proper loop, but no error message

At this point, sending too much time on this. I moved on to the pickle phase.

# Pickling CDInventory

Everything I read, and listened to in the videos, lead me to believe this was going to be easy. I only added the pickle feature to the “read\_file” and “write\_file” functions of the code. When importing the pickle module, I already knew, the data file needed changed to a “.dat” file. Again I spent too much time on this, that when I created the binary file, it was a jump for joy. Figure 4

Graphical user interface, application, Word

Description automatically generated

Figure dat file created in the folder.

However, the case, I spent about 9 hours trying to troubleshoot my pickle problem. I was able to save only one line of data. No matter what I kept changing, the data stayed at one. At one point, my file was erased, so I undid everything as long as the code captured one line of data, I was happy. The issue continued somewhere in the save portion. I could not save additional entries. I could see the entries in real time by adding the ID’s, and finally I was able to save the same number of entries, but the function was not saving or, when reading, the file came back with the same entries but not the same data that was entered, see Idle 4 which saves, then Idle 5 returns the loaded file with 3 entries but not the same info.

Text

Description automatically generated

Idle ID's entered then saved.

Text

Description automatically generated

Idle Loaded data returns with the same entries.

The figure below shows pickle was added to the code when saved.

A screenshot of a computer

Description automatically generated with medium confidence

Figure added pickle to load the file.

Figure 6 shows the sequence to save a file.

Text

Description automatically generated

Figure - write file function with added pickle.

# Summary

This assignment was tougher than the last. Getting the resources needed for this project was not the hard part. The hard part was applying what I learned.

# Link to github

<https://github.com/roggergt/Assignment07>

# Other resources

A quick resource I used that I really liked talking about exceptions:

[The New Boston](https://www.youtube.com/watch?v=1cCU0owdiR4&list=PL6gx4Cwl9DGAcbMi1sH6oAMk4JHw91mC_&index=28) – I liked the examples and metaphors to explain the Exceptions.

I had a hard time finding a good topic about pickling. Most the examples were using extensions I am not familiar with, such as, pkl.

[Mark Jay](https://www.youtube.com/watch?v=Pl4Hp8qwwes) on youtube.com talked about pickle but the examples didn’t make any sense to me given our assignment.

# Appendix

Straight copy from Anaconda

1. *# Title: CDInventory.py*
2. *# Desc: Working with classes and functions.*
3. *# Change Log: (Who, When, What)*
4. *# rtovar, 2021-Nov-21, Copied from Original File*
5. *# rtovar, 2021-Nov-21, Created/added functions*
6. *# rtovar, 2021-Nov-21, Added docstrings*
7. *# rtovar, 2021-Nov-28, Added structured error handling where needed.*
8. *# rtovar, 2021-Nov-28, Pickle import and editing file*
9. *#------------------------------------------#*
10. *# -- DATA -- #*
11. *strChoice = '' # User input*
12. *lstTbl = [] # list of lists to hold data*
13. *dicRow = {} # list of data row*
14. *strFileName = 'CDInventory.dat' # data storage file*
15. *objFile = None # file object*
16. *import pickle*
17. *# -- PROCESSING -- #*
18. *class DataProcessor:*
19. *""" A collection of processing data fuctions:*
20. *Adding user inventory and delete inventory."""*
22. *def adding\_user\_inventory(strID, strTitle, stArtist):*
24. *"""Adding user inventory grabs the user inputs of ID, Title,*
25. *and Artist to then add to the list of data (lstTble)*
27. *arg: strID - ID number selected by the end user.*
28. *strTitle - The name of the title of song selected by the end user.*
29. *strArtist -The Artist of the song selected by the end user.*
31. *Returns: The added inventory list will be updated and displayed to the end user.*
32. *"""*
33. *intID = int(strID)*
34. *dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}*
35. *lstTbl.append(dicRow)*
36. *IO.show\_inventory(lstTbl)*
38. *def delete\_inventory():*
39. *"""Delete inventory function is designed to process a delete selection by the*
40. *end user then to update the over inventory list CDInventory text tile.*
42. *arg: none*
44. *return: updated list minus the selected ID choice from the end user, which*
45. *will remove the rove including, title and artist.*
46. *"""*
47. *intRowNr = -1*
48. *blnCDRemoved = False*
49. *for row in lstTbl:*
50. *intRowNr += 1*
51. *if row['ID'] == intIDDel:*
52. *del lstTbl[intRowNr]*
53. *blnCDRemoved = True*
54. *break*
55. *if blnCDRemoved:*
56. *print('\nThe CD was removed. \n')*
57. *else:*
58. *print('Could not find this CD!')*
59. *IO.show\_inventory(lstTbl)*
60. *class FileProcessor:*
61. *"""Processing the data to and from text file"""*
62. *@staticmethod*
63. *def read\_file(strFileName, table):*
64. *"""Function to manage data ingestion from file to a list of dictionaries*
65. *Reads the data from file identified by strFileName into a 2D table*
66. *(list of dicts) table one line in the file represents one dictionary row in table.*
67. *Args:*
68. *file\_name (string): name of file used to read the data from*
69. *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*
70. *Returns:*
71. *None.*
72. *"""*
73. *table.clear() # this clears existing data and allows to load data from file*
74. *try:*
75. *objFile = open(strFileName, 'rb')*
76. *data = pickle.load(objFile).strip().split(',')*
77. *for line in objFile:*
78. *dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}*
79. *table.append(dicRow)*
80. *objFile.close()*
81. *except FileNotFoundError:*
82. *print('No the file does not exist')*
83. *except ValueError:*
84. *print('The file is closed but now open.')*
85. *except EOFError:*
86. *print('Ooops')*
87. *@staticmethod*
88. *def write\_file(strFileName, lstTbl):*
89. *"""Write file saves the text file with CD Inventory data onto a destinaion on the hard drive.*
91. *arg: strFileName - is a variable for the CDInventory.txt file used to store the data entries.*
92. *lstTbl - is the current table of list with rows added with ID, Title, and Artist information.*
94. *return: None*
95. *"""*
96. *objFile = open(strFileName, 'wb')*
97. *for row in lstTbl:*
98. *lstValues = list(row.values())*
99. *lstValues[0] = str(lstValues[0])*
100. *pickle.dump(','.join(lstValues) + '\n', objFile)*
101. *objFile.close()*
102. *print('\nYou have successfully saved your file.\n')*
104. *# -- PRESENTATION (Input/Output) -- #*
105. *class IO:*
106. *"""Handling Input / Output"""*
107. *@staticmethod*
108. *def print\_menu():*
109. *"""Displays a menu of choices to the user*
110. *Args:*
111. *None.*
112. *Returns:*
113. *None.*
114. *"""*
115. *print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')*
116. *print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')*
117. *@staticmethod*
118. *def menu\_choice():*
119. *"""Gets user input for menu selection*
120. *Args:*
121. *None.*
122. *Returns:*
123. *choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x*
124. *"""*
125. *choice = ' '*
126. *try:*
127. *while choice not in ['l', 'a', 'i', 'd', 's', 'x']:*
128. *choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()*
129. *print() # Add extra space for layout*
130. *return choice*
131. *except:*
132. *print('That is not an option.')*
134. *@staticmethod*
135. *def show\_inventory(table):*
136. *"""Displays current inventory table*
137. *Args:*
138. *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.*
139. *Returns:*
140. *None.*
141. *"""*
142. *print('======= The Current Inventory: =======')*
143. *print('ID\tCD Title (by: Artist)\n')*
144. *for row in table:*
145. *print('{}\t{} (by:{})'.format(\*row.values()))*
146. *print('======================================')*
147. *def add\_inventory():*
148. *"""This function adds an inventory row using 3 inputs: ID, Title, Artist.*
150. *arg: none*
152. *return: An input selection is offered to the end user if they use to add a row.*
153. *"""*
155. *# 3.3.1 Ask user for new ID, CD Title and Artist*
157. *while True:*
158. *try:*
159. *strID = int(input('Enter ID: ').strip())*
160. *break*
161. *except Exception as e:*
162. *print(e)*
163. *print('Please enter a number and try again. ')*
165. *strTitle = input('What is the CD\'s title? ').strip()*
166. *stArtist = input('What is the Artist\'s name? ').strip()*
167. *DataProcessor.adding\_user\_inventory(strID, strTitle, stArtist)*
169. *# 1. When program starts, read in the currently saved Inventory*
170. *FileProcessor.read\_file(strFileName, lstTbl)*

173. *# 2. start main loop*
174. *while True:*
175. *# 2.1 Display Menu to user and get choice*
176. *IO.print\_menu()*
177. *strChoice = IO.menu\_choice()*
178. *# 3. Process menu selection*
179. *# 3.1 process exit first*
180. *if strChoice == 'x':*
181. *break*
182. *# 3.2 process load inventory*
183. *if strChoice == 'l':*
184. *print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')*
185. *strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')*
186. *try:*
187. *strYesNo.lower() == 'yes'*
188. *print('reloading...')*
189. *FileProcessor.read\_file(strFileName, lstTbl)*
190. *IO.show\_inventory(lstTbl)*
191. *except Exception as e:*
192. *print(e)*
193. *input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')*
194. *IO.show\_inventory(lstTbl)*
195. *continue # start loop back at top.*
196. *# 3.3 process add a CD*
197. *# I have already moved ADD CD*
198. *elif strChoice == 'a':*
199. *IO.add\_inventory()*
200. *continue*
201. *# 3.4 process display current inventory*
202. *elif strChoice == 'i':*
203. *IO.show\_inventory(lstTbl)*
204. *continue # start loop back at top.*
205. *# 3.5 process delete a CD*
206. *elif strChoice == 'd':*
207. *# 3.5.1 get Userinput for which CD to delete*
208. *# 3.5.1.1 display Inventory to user*
209. *IO.show\_inventory(lstTbl)*
210. *# 3.5.1.2 ask user which ID to remove*
211. *while True:*
212. *try:*
213. *intIDDel = int(input('Which ID would you like to delete? ').strip())*
214. *break*
215. *except Exception as e:*
216. *print(e)*
217. *print('No no, that\'s not right.')*
219. *# 3.5.2 search thru table and delete CD*
220. *DataProcessor.delete\_inventory()*
221. *continue # start loop back at top.*
222. *# 3.6 process save inventory to file*
223. *elif strChoice == 's':*
224. *# 3.6.1 Display current inventory and ask user for confirmation to save*
225. *IO.show\_inventory(lstTbl)*
226. *while True:*
227. *try:*
228. *strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()*
230. *# 3.6.2 Process choice*
231. *if strYesNo == 'y':*
232. *# 3.6.2.1 save data*
233. *FileProcessor.write\_file(strFileName, lstTbl)*
234. *break*
235. *if strYesNo == 'n':*
236. *break*
237. *else:*
238. *input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')*
239. *continue # start loop back at top.*
240. *except:*
241. *print('oops')*
242. *# 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:*
243. *else:*
244. *print('General Error. \n')*