Nasser Saber

November 27, 2021

Foundations of Programming, Python

Assignment 07

Creating a Python Program for Managing an Inventory List using Structured Error Handling and Binary Files

# Introduction

# In this assignment, I will explain the steps I used to modify the Assignment06 Python Script to add structured error handling as well as altering save/read functions to work with binary files for data store.

# Creating the Script

I started by modifying CDInventory.py in Spyder. The work mainly involved identifying the parts with user input, interactions with files and function calls in order to add structured error handling. Examples include when user inputs a CD ID other than an integer or when the data file is not available upon first load call. The added except classes like FileNotFoundError and ValueError helped not to break the program when such errors occur.

# Testing the Script

I ran the script in Spyder first. It runs smooth by asking the user to make the menu choice and performs the expected action based on the chosen menu number as shown in Figure 1.

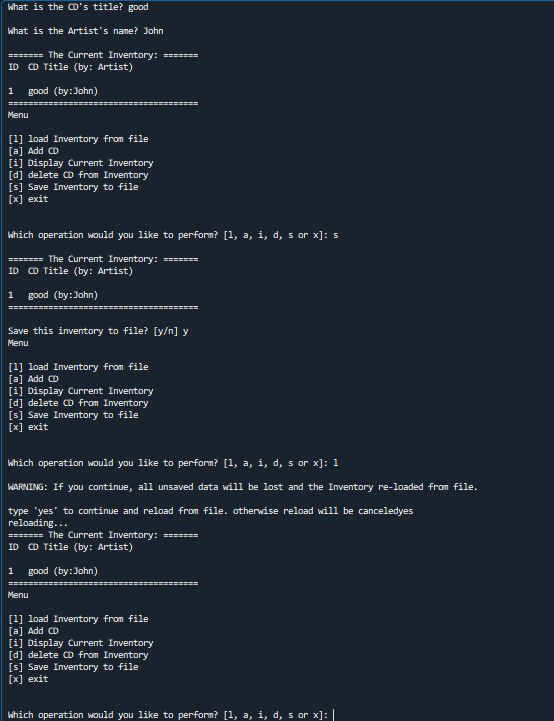


Figure 1 Result of running the AddressBook.py script in Spyder

I then ran the same script in the terminal window, which returned the same results as expected (Figure 2).

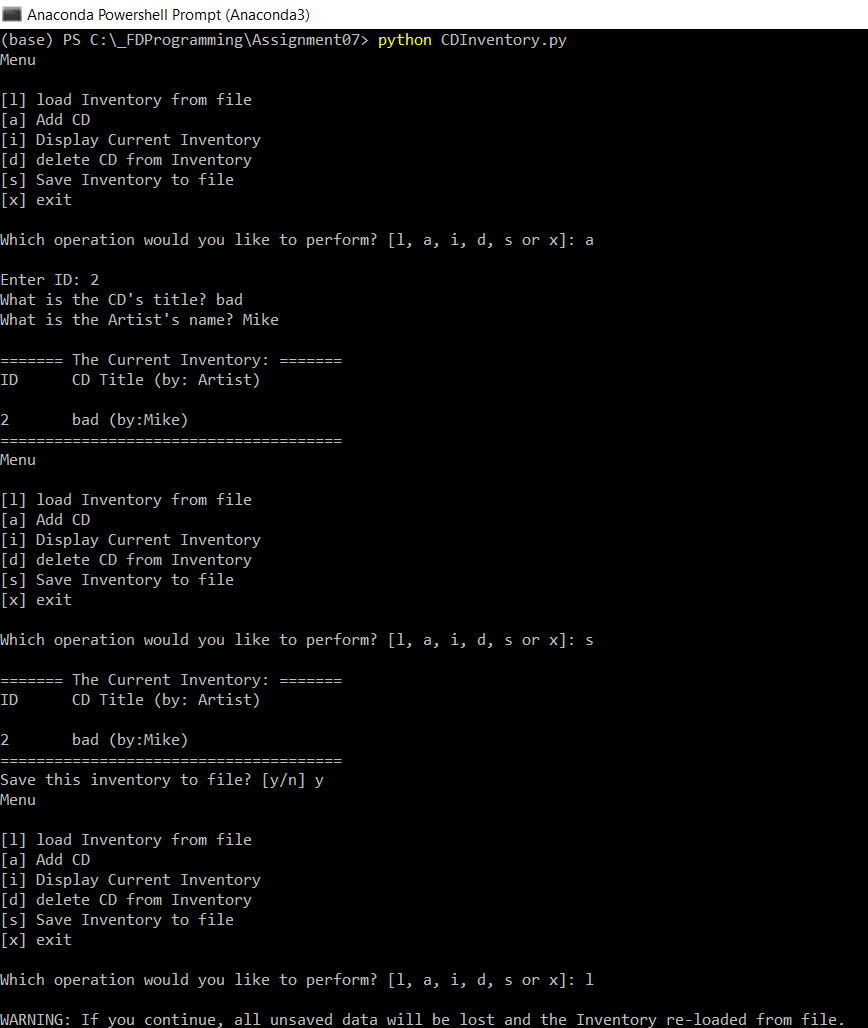


Figure 2 – Result of running the AddressBook.py script in terminal window

Finally, I located the “CDInventory.dat” in the folder and checked that the user inputs were successfully saved in there (Figure 3).

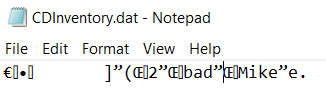


Figure 3 Contents of CDInventory.dat

# Summary

In this assignment, I practiced modifying an existing python program in Spyder, by adding structured error handling and also altering the store functions to work with binary data.

# Appendix

GitHub

CDInventory.py

1. *#------------------------------------------#*
2. *# Title: CDInventory.py*
3. *# Desc: Working with classes and functions.*
4. *# Change Log: (Who, When, What)*
5. ***# DBiesinger, 2030-Jan-01, Created File***
6. *# NSaber, 2021-Nov-20, Added Functionality*
7. *# NSaber, 2021-Nov-27, Added Structured Error Handling*
8. *#------------------------------------------#*
9. **import** pickle
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***

18. *# -- PROCESSING -- #*
19. **class** DataProcessor:
20. ***# TODone add functions for processing here***
21. @staticmethod
22. **def** add\_table(ID, title, artist):
23. """Function to process and add the user input data to a 2D list table (list of dicts)
25. **Args:**
26. ID (string): CD ID input by the user
27. title (string): Title of the CD input by the user
28. artist (string): Artist name input by the user
30. **Returns:**
31. None.
32. """
33. intID = int(ID)
34. dicRow = {'ID': intID, 'Title': title, 'Artist': artist}
35. **lstTbl.append(dicRow)**

38. @staticmethod
39. **def** row\_delete(IDDel):
40. **"""Function to search thru table and delete CD (row)**
42. Args:
43. IDDel (string): CD ID to be deleted (input by the user)

46. Returns:
47. None.
48. """
49. intRowNr = -1
50. **blnCDRemoved = False**
51. **for** row **in** lstTbl:
52. intRowNr += 1
53. **if** row['ID'] == IDDel:
54. **del** lstTbl[intRowNr]
55. **blnCDRemoved = True**
56. **break**
57. **if** blnCDRemoved:
58. **print**('The CD was removed')
59. **else**:
60. **print('Could not find this CD!')**
62. **class** FileProcessor:
63. """Processing the data to and from text file"""
65. **@staticmethod**
66. **def** read\_file(file\_name):
67. """Function to manage data ingestion from file
69. Reads the data from binary file identified by file\_name
71. Args:
72. file\_name (string): name of file used to read the data from

75. **Returns:**
76. table(object): the data read from the file
77. """
79. **with** open(file\_name, 'rb') **as** objFile:
80. **table = pickle.load(objFile)**
81. **return** table
83. @staticmethod
84. **def** write\_file(file\_name, table):
85. ***# TODone Add code here***
86. """Function to write the data to a binary file
88. Writes the data to file identified by file\_name from a 2D table
89. (list of dicts) table one line in the file represents one dictionary row in table.
91. Args:
92. file\_name (string): name of file used to write the data to
93. table (Object): the data to be saved to the file
95. **Returns:**
96. None.
97. """
99. **for** row **in** table:
100. **lstValues = list(row.values())**
101. lstValues[0] = str(lstValues[0])
103. **with** open(file\_name, 'wb') **as** objFile:
104. pickle.dump(lstValues, objFile)
106. 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.
123. """
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**\t**CD Title (by: Artist)**\n**')
159. **for** row **in** table:
160. **print('{}\t{} (by:{})'.format(\*row.values()))**
161. **print**('======================================')
163. *# TODone add I/O functions as needed*
165. **@staticmethod**
166. **def** user\_input():
167. """Gets user input for CD
169. Args:
170. **None.**
172. Returns:
173. strID (string): CD ID input by the user
174. strTitle (string): Title of the CD input by the user
175. **stArtist (string): Artist name input by the user**
176. """
178. strID = input('Enter ID: ').strip()
179. strTitle = input('What is the CD**\'**s title? ').strip()
180. **stArtist = input('What is the Artist\'s name? ').strip()**
181. **print**() *# Add extra space for layout*
182. **return** strID, strTitle, stArtist
184. *# 1. When program starts, read in the currently saved Inventory*
185. **try:**
186. FileProcessor.read\_file(strFileName)
187. **except** FileNotFoundError **as** e:
188. **print**('Text file does not exist!')
189. **print**('Build in error info:')
190. **print(type(e), e, e.\_\_doc\_\_, sep='\n')**

193. *# 2. start main loop*
194. **while** True:
195. ***# 2.1 Display Menu to user and get choice***
196. IO.print\_menu()
197. strChoice = IO.menu\_choice()
199. *# 3. Process menu selection*
200. ***# 3.1 process exit first***
201. **if** strChoice == 'x':
202. **break**
203. *# 3.2 process load inventory*
204. **if** strChoice == 'l':
205. **print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')**
206. strYesNo = input('type **\'**yes**\'** to continue and reload from file. otherwise reload will be canceled')
207. **if** strYesNo.lower() == 'yes':
208. **print**('reloading...')
209. **try**:
210. **FileProcessor.read\_file(strFileName)**
211. IO.show\_inventory(lstTbl)
212. **except** FileNotFoundError **as** e:
213. **print**('Text file does not exist!')
214. **print**('Build in error info:')
215. **print(type(e), e, e.\_\_doc\_\_, sep='\n')**
216. **else**:
217. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
218. IO.show\_inventory(lstTbl)
219. **continue** *# start loop back at top.*
220. ***# 3.3 process add a CD***
221. **elif** strChoice == 'a':
222. *# 3.3.1 Ask user for new ID, CD Title and Artist*
223. *# TODone move IO code into function*
224. strID, strTitle, stArtist = IO.user\_input()
225. ***# 3.3.2 Add item to the table***
226. *# TODone move processing code into function*
227. **try**:
228. DataProcessor.add\_table(strID, strTitle, stArtist)
229. IO.show\_inventory(lstTbl)
230. **except ValueError as e:**
231. **print**('ID should be an integer!')
232. **print**('Build in error info:')
233. **print**(type(e), e, e.\_\_doc\_\_, sep='**\n**')
234. **continue** *# start loop back at top.*
235. ***# 3.4 process display current inventory***
236. **elif** strChoice == 'i':
237. IO.show\_inventory(lstTbl)
238. **continue** *# start loop back at top.*
239. *# 3.5 process delete a CD*
240. **elif strChoice == 'd':**
241. *# 3.5.1 get Userinput for which CD to delete*
242. *# 3.5.1.1 display Inventory to user*
243. IO.show\_inventory(lstTbl)
244. *# 3.5.1.2 ask user which ID to remove*
245. **intIDDel = int(input('Which ID would you like to delete? ').strip())**
246. *# 3.5.2 search thru table and delete CD*
247. DataProcessor.row\_delete(intIDDel)
248. IO.show\_inventory(lstTbl)
249. **continue** *# start loop back at top.*
250. ***# 3.6 process save inventory to file***
251. **elif** strChoice == 's':
252. *# 3.6.1 Display current inventory and ask user for confirmation to save*
253. IO.show\_inventory(lstTbl)
254. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
255. ***# 3.6.2 Process choice***
256. **if** strYesNo == 'y':
257. *# 3.6.2.1 save data*
258. *# TODone move processing code into function*
259. FileProcessor.write\_file(strFileName, lstTbl)
260. **else:**
261. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
262. **continue** *# start loop back at top.*
263. *# 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:*
264. **else**:
265. **print('General Error')**