Justin Roe

08/23/2020

IT FDN 110 B

Assignment 07

Assignment 07

# Introduction

Assignment 7 introduced students to binary files / pickling, exception types (and custom exceptions), and error handling. Although I’ve liked using the try / except error handling technique throughout many assignments, I struggled with adding the “this ID has been used!” feature in my “user\_entry” function. I also had difficulty pickling the list of dicts, and outputting it correctly. I then struggled with embedding the “filenotfound” error into that portion of the script. I used try/except statements in many cases, but struggles with syntax occasionally.

# Knowledge Application / Details

See Figures 1-3 for screenshots of the code execution.

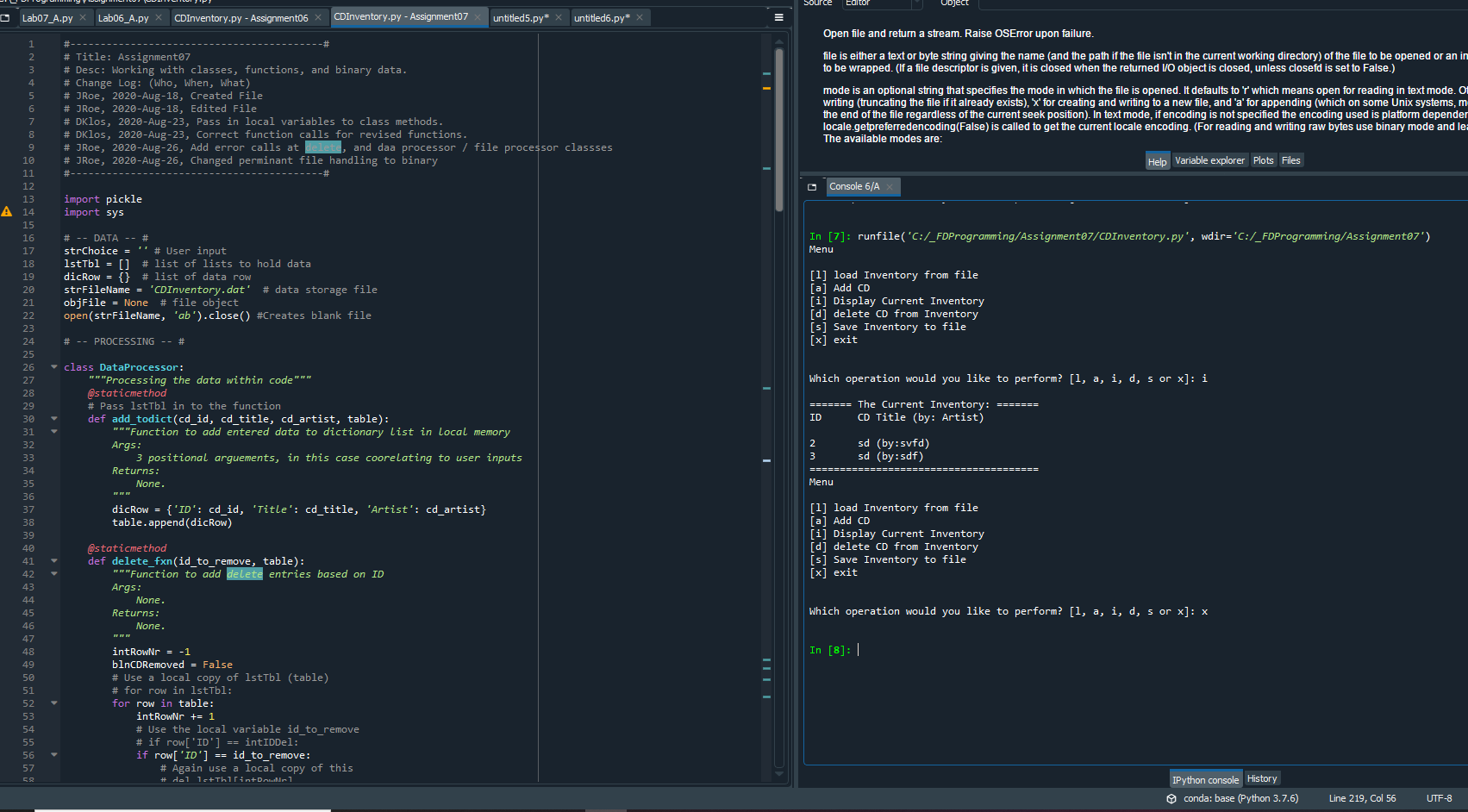


Figure – Representative Spyder run of CDInventory.py script

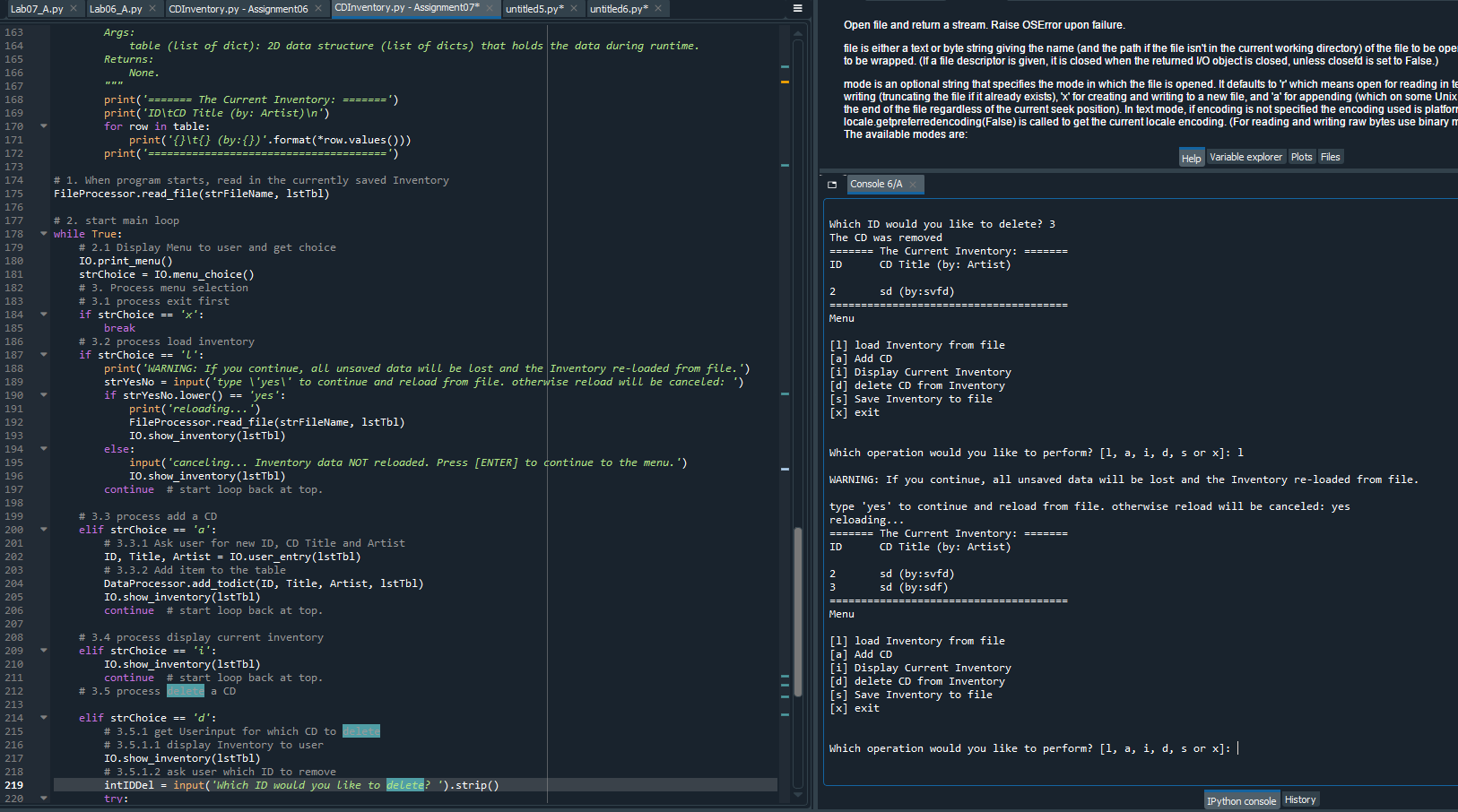


Figure – Representative Spyder run of CDInventory.py script

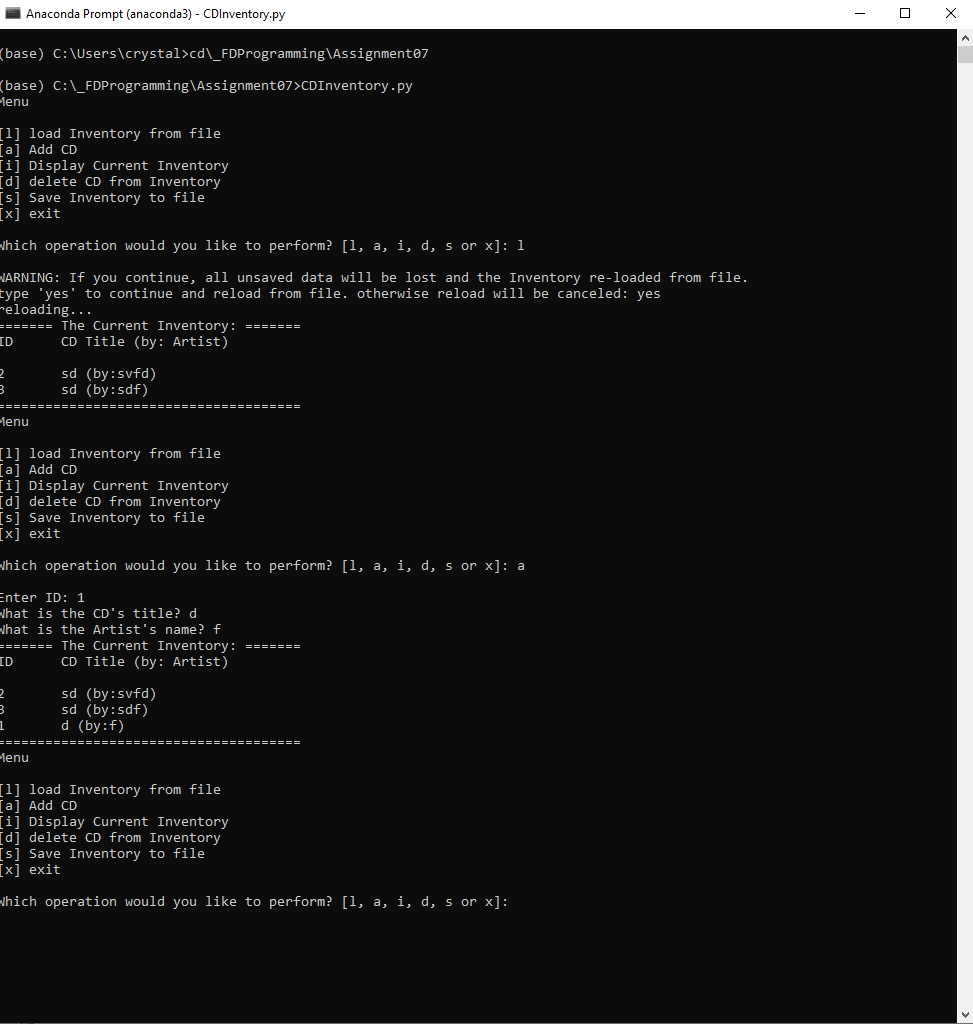


Figure – Representative image of CDInventory.py running in console

# Summary

I spent the most time this week on a silly error – while writing my binary data to the file referenced near the header, I forgot to change the file type to a .dat… this was causing many strange errors (that I cant say I 100% understand). I also utilized the list.extend(iterable) function to ensure all rows were output (see FDN\_Py\_Module\_04.pdf, page 18[[1]](#footnote-1)). Other sites that helped me were this [pickling youtube video](https://www.youtube.com/watch?v=2Tw39kZIbhs)[[2]](#footnote-2), [this](https://intellipaat.com/community/28834/python-error-string-index-out-of-range-python#:~:text=The%20string%20index%20out%20of,not%20inside%20of%20the%20string.)[[3]](#footnote-3) page that aided me with formatting my binary output, and [this](https://docs.python.org/3/tutorial/errors.html)[[4]](#footnote-4) page that helped greatly with a few of my try / except statements. My Github link is [here](https://github.com/jusroe/Assignment_07).

# Appendix A – Syntax

Generated using [planetb’s webpage](http://planetb.ca/syntax-highlight-word) (external reference)[[5]](#footnote-5) web page

1. #------------------------------------------#
2. # Title: Assignment07
3. # Desc: Working with classes, functions, and binary data.
4. # Change Log: (Who, When, What)
5. # JRoe, 2020-Aug-18, Created File
6. # JRoe, 2020-Aug-18, Edited File
7. # DKlos, 2020-Aug-23, Pass in local variables to class methods.
8. # DKlos, 2020-Aug-23, Correct function calls for revised functions.
9. # JRoe, 2020-Aug-26, Add error calls at delete, and daa processor / file processor classses
10. # JRoe, 2020-Aug-26, Changed perminant file handling to binary
11. #------------------------------------------#
13. **import** pickle
14. **import** sys
16. # -- DATA -- #
17. strChoice = '' # User input
18. lstTbl = []  # list of lists to hold data
19. dicRow = {}  # list of data row
20. strFileName = 'CDInventory.dat'  # data storage file
21. objFile = None  # file object
22. open(strFileName, 'ab').close() #Creates blank file
24. # -- PROCESSING -- #
26. **class** DataProcessor:
27. """Processing the data within code"""
28. @staticmethod
29. # Pass lstTbl in to the function
30. **def** add\_todict(cd\_id, cd\_title, cd\_artist, table):
31. """Function to add entered data to dictionary list in local memory
32. Args:
33. 3 positional arguements, in this case coorelating to user inputs
34. Returns:
35. None.
36. """
37. dicRow = {'ID': cd\_id, 'Title': cd\_title, 'Artist': cd\_artist}
38. table.append(dicRow)
40. @staticmethod
41. **def** delete\_fxn(id\_to\_remove, table):
42. """Function to add delete entries based on ID
43. Args:
44. None.
45. Returns:
46. None.
47. """
48. intRowNr = -1
49. blnCDRemoved = False
50. # Use a local copy of lstTbl (table)
51. # for row in lstTbl:
52. **for** row **in** table:
53. intRowNr += 1
54. # Use the local variable id\_to\_remove
55. # if row['ID'] == intIDDel:
56. **if** row['ID'] == id\_to\_remove:
57. # Again use a local copy of this
58. # del lstTbl[intRowNr]
59. **del** table[intRowNr]
60. blnCDRemoved = True
61. **break**
62. **if** blnCDRemoved:
63. **print**('The CD was removed')
64. **else**:
65. **print**('Could not find this CD!')
67. **class** FileProcessor:
68. """Processing the data to and from text file"""
70. @staticmethod
71. **def** read\_file(file\_name, table):
72. """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.
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
78. Returns:
79. None.
80. """
81. **try**:
82. table.clear()  # this clears existing data and allows to load data from file
83. with open(file\_name, 'rb') as item:
84. list1 = pickle.load(item)
85. table.extend(list1)
87. **except**:
88. **print**('\nYou must enter data first!\n')

91. @staticmethod
92. **def** save\_fxn(file\_name, table):
93. """Function to save entered data to designated file
94. Args:
95. None.
96. Returns:
97. None.
98. """
99. with open(file\_name, 'wb') as item:
100. pickle.dump(table, item)

103. # -- PRESENTATION (Input/Output) -- #
105. **class** IO:
106. """Handling Input / Output"""
108. @staticmethod
109. **def** print\_menu():
110. """Displays a menu of choices to the user
111. Args:
112. None.
113. Returns:
114. None.
115. """
117. **print**('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
118. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
120. @staticmethod
121. **def** menu\_choice():
122. """Gets user input for menu selection
123. Args:
124. None.
125. Returns:
126. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
127. """
128. choice = ' '
129. **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:
130. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
131. **print**()  # Add extra space for layout
132. **return** choice
134. @staticmethod
135. **def** user\_entry(table):
136. """Gets user input for "enter data" section
137. Args:
138. None.
139. Returns:
140. ID, song, and title
141. """
142. **while** True:
143. # TODO, no repeats
144. strID = input('Enter ID: ').strip()
145. # Ensure only integer inputs
146. **try**:
147. intID = int(strID)
148. **break**
149. **except** ValueError:
150. **print**('\nIntegers Only!\n')
151. **continue**



156. strTitle = input('What is the CD\'s title? ').strip()
157. stArtist = input('What is the Artist\'s name? ').strip()
158. **return** intID, strTitle, stArtist
160. @staticmethod
161. **def** show\_inventory(table):
162. """Displays current inventory table
163. Args:
164. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
165. Returns:
166. None.
167. """
168. **print**('======= The Current Inventory: =======')
169. **print**('ID\tCD Title (by: Artist)\n')
170. **for** row **in** table:
171. **print**('{}\t{} (by:{})'.format(\*row.values()))
172. **print**('======================================')
174. # 1. When program starts, read in the currently saved Inventory
175. FileProcessor.read\_file(strFileName, lstTbl)
177. # 2. start main loop
178. **while** True:
179. # 2.1 Display Menu to user and get choice
180. IO.print\_menu()
181. strChoice = IO.menu\_choice()
182. # 3. Process menu selection
183. # 3.1 process exit first
184. **if** strChoice == 'x':
185. **break**
186. # 3.2 process load inventory
187. **if** strChoice == 'l':
188. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
189. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled: ')
190. **if** strYesNo.lower() == 'yes':
191. **print**('reloading...')
192. FileProcessor.read\_file(strFileName, lstTbl)
193. IO.show\_inventory(lstTbl)
194. **else**:
195. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
196. IO.show\_inventory(lstTbl)
197. **continue**  # start loop back at top.
199. # 3.3 process add a CD
200. **elif** strChoice == 'a':
201. # 3.3.1 Ask user for new ID, CD Title and Artist
202. ID, Title, Artist = IO.user\_entry(lstTbl)
203. # 3.3.2 Add item to the table
204. DataProcessor.add\_todict(ID, Title, Artist, lstTbl)
205. IO.show\_inventory(lstTbl)
206. **continue**  # start loop back at top.
208. # 3.4 process display current inventory
209. **elif** strChoice == 'i':
210. IO.show\_inventory(lstTbl)
211. **continue**  # start loop back at top.
212. # 3.5 process delete a CD
214. **elif** strChoice == 'd':
215. # 3.5.1 get Userinput for which CD to delete
216. # 3.5.1.1 display Inventory to user
217. IO.show\_inventory(lstTbl)
218. # 3.5.1.2 ask user which ID to remove
219. intIDDel = input('Which ID would you like to delete? ').strip()
220. **try**:
221. intDDel1 = int(intIDDel)
222. **except** ValueError:
223. **print**('\nIntegers Only!\n')
225. # 3.5.2 search thru table and delete CD
226. DataProcessor.delete\_fxn(intDDel1, lstTbl)
227. IO.show\_inventory(lstTbl)
228. **continue**  # start loop back at top.
230. # 3.6 process save inventory to file
231. **elif** strChoice == 's':
232. # 3.6.1 Display current inventory and ask user for confirmation to save
233. IO.show\_inventory(lstTbl)
234. strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
235. # 3.6.2 Process choice
236. **if** strYesNo == 'y':
237. # 3.6.2.1 save data
238. FileProcessor.save\_fxn(strFileName, lstTbl)
239. **else**:
240. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
241. **continue**  # start loop back at top.
243. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
244. **else**:
245. **print**('General Error')

1. Retrieved 2020-Aug-26 [↑](#footnote-ref-1)
2. Retrieved 2020-Aug-26 [↑](#footnote-ref-2)
3. Retrieved 2020-Aug-26 [↑](#footnote-ref-3)
4. Retrieved 2020-Aug-26 [↑](#footnote-ref-4)
5. Retrieved 2020-Aug-26 [↑](#footnote-ref-5)