|  |  |
| --- | --- |
| Name | Bruck Assefa |
| Date | 2021-Feb-03 |
| Course | IT FDN 110: Introduction to Programming (Python) |
| Assignment | Assignment 07 |
| Githun | **https://github.com/BruckAssefa/Assignment7** |

Assignment 07

# Introduction

This document covers steps involved in using binary file and using pickle module.

The IDE we will be using is Spyder. We will also be working with Python 3.8.5 version as shown on the below console.

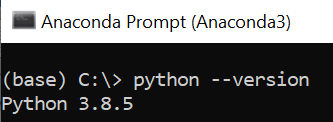


Figure "Python version" script

Research Exception Handling in Python

* <https://docs.python.org/3/tutorial/errors.html>
* <https://www.w3schools.com/python/python_try_except.asp>

The python.org documentation provides foundational knowledge on Syntax errors, raising exceptions, creating user defined exceptions. It is neatly laid out and easy to understand. After having the foundational knowledge, I used the w3schools resource. The w3schools allows you to read about the syntax and provides exercise that you can test out online. It is easy to follow and helps you practice the concepts learned.

# Create a Python Script

We will use Assignment06 solutions scriupt and test the following added scripts demonstrating using binary file and pickling:

1. Reassign strFileName from a text file to a binary file.
2. Use pickle module to load and dump data.
3. Use try except error handling to catch wrong user inputs.

First open a note pad document and write down the below scripts.

1. #------------------------------------------#
2. # Title: Assignment06\_Starter.py
3. # Desc: Working with classes and functions.
4. # Change Log: (Who, When, What)
5. # ABruck, 2021-Feb-2, Created File
6. # ABruck, 2021-Mar-2, Modified File
7. #------------------------------------------#
9. **import** pickle
10. # -- DATA -- #
11. strChoice = '' # User input
12. lstTbl = []  # list of lists to hold data
13. dicRow = {}  # list of data row
14. # Change text file to binary file
15. strFileName = 'CDInventory.dat'  # data storage file
16. objFile = None  # file object

19. # -- PROCESSING -- #
20. **class** DataProcessor:
22. # Need to put @staticmethod before all your method declaration for these classes.
23. @staticmethod
24. # def delete\_inventory(table):
25. # Since we should not be prompting for user input inside of DataProcessor
26. #   we need to also accept as an argument the ID we with to remove.
27. **def** delete\_inventory(table, id\_to\_remove):
29. # We do not want to make any calls to outside class methods.
30. # IO.show\_inventory(table)
31. # 3.5.1.2 ask user which ID to remove
32. # This violates SoC and should be in IO or the main while loop.
33. # intIDDel = int(input('Which ID would you like to delete? ').strip())
34. # 3.5.2 search thru table and delete CD
35. intRowNr = -1
36. blnCDRemoved = False
37. **for** row **in** table:
38. intRowNr += 1
39. **if** row['ID'] == id\_to\_remove:
40. **del** table[intRowNr]
41. blnCDRemoved = True
42. **break**
43. **if** blnCDRemoved:
44. **print**('The CD was removed')
45. **else**:
46. **print**('Could not find this CD!')
48. # No outside class calls.
49. # IO.show\_inventory(table)
51. @staticmethod
52. **def** add\_to\_inventory(cd\_id, cd\_title, cd\_artist, table):
53. new\_cd = {
54. 'ID': int(cd\_id),
55. 'Title': cd\_title,
56. 'Artist': cd\_artist
57. }
58. table.append(new\_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 to a list of dictionaries
69. Reads the data from file identified by file\_name into a 2D table
70. (list of dicts) table one line in the file represents one dictionary row in table.
72. Args:
73. file\_name (string): name of file used to read the data from
74. table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
76. Returns:
77. None.
78. """
79. # Using pickle.load to read data from binary file
80. with open(file\_name, 'rb') as fileobj:
81. data = pickle.load(fileobj)
83. @staticmethod
84. **def** write\_file(data, file\_name):
86. # No function calls to external class methods
87. # IO.show\_inventory(table)
89. # No prompting for user input inside of FileProcessor
90. # strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
91. # if strYesNo == 'y':

94. # Using pickle.dump to write data on binary file
95. with open(file\_name, 'wb') as fileobj:
96. pickle.dump(data, fileobj)
98. # else:
99. #     input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')

102. # -- PRESENTATION (Input/Output) -- #
104. **class** IO:
105. """Handling Input / Output"""
107. @staticmethod
108. **def** print\_menu():
109. """Displays a menu of choices to the user
111. Args:
112. None.
114. Returns:
115. None.
116. """
118. **print**('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
119. **print**('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
121. @staticmethod
122. **def** menu\_choice():
123. """Gets user input for menu selection
125. Args:
126. None.
128. Returns:
129. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
131. """
132. choice\_list = ['l', 'a', 'i', 'd', 's', 'x']
133. choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()
134. **try** :
135. choice\_list.index(choice)
136. res = "Element found"
137. **except** ValueError :
138. res = "Element not in list !"
139. **print**()  # Add extra space for layout
141. # Printing result
142. **print**("The value after catching error : " + str(res))
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\tCD Title (by: Artist)\n')
159. **for** row **in** table:
160. **print**('{}\t{} (by:{})'.format(\*row.values()))
161. **print**('======================================')
163. # We need to have @staticmethod before each of our methods
164. @staticmethod
165. **def** add\_inventory():
167. strID = input('Enter ID: ').strip()
168. strTitle = input('What is the CD\'s title? ').strip()
169. stArtist = input('What is the Artist\'s name? ').strip()
171. **return** strID, strTitle, stArtist
173. # intID = int(strID)
174. # dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
175. # lstTbl.append(dicRow)
176. # IO.show\_inventory(lstTbl)


180. # 2. start main loop
181. **while** True:
182. # 2.1 Display Menu to user and get choice
183. IO.print\_menu()
184. strChoice = IO.menu\_choice()
186. # 3. Process menu selection
187. # 3.1 process exit firstcls
188. **if** strChoice == 'x':
189. **break**
191. # 3.2 process load inventory
192. **if** strChoice == 'l':
193. **print**('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
194. strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')
195. **if** strYesNo.lower() == 'yes':
196. **print**('reloading...')
197. FileProcessor.read\_file(strFileName)
198. IO.show\_inventory(lstTbl)
199. **else**:
200. input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
201. IO.show\_inventory(lstTbl)
202. **continue**  # start loop back at top.
204. # 3.3 process add a CD
205. **elif** strChoice == 'a':
206. # The IO function call should get the data from the user and return it to here.
207. # IO.add\_inventory()
208. # 3.3.1 Ask user for new ID, CD Title and Artist
209. cd\_id, cd\_title, cd\_artist = IO.add\_inventory()
210. # 3.3.2 Add item to the table
211. DataProcessor.add\_to\_inventory(cd\_id, cd\_title, cd\_artist, lstTbl)
213. # 3.4 process display current inventory
214. **elif** strChoice == 'i':
215. IO.show\_inventory(lstTbl)
216. **continue**  # start loop back at top.
218. # 3.5 process delete a CD
219. **elif** strChoice == 'd':
220. # 3.5.1 get Userinput for which CD to delete
221. # 3.5.1.1 display Inventory to user
222. IO.show\_inventory(lstTbl)
223. intIDDel = int(input('Which ID would you like to delete? ').strip())
224. # DataProcessor.delete\_inventory(lstTbl)
225. DataProcessor.delete\_inventory(lstTbl, intIDDel)
227. # 3.6 process save inventory to file
228. **elif** strChoice == 's':
229. # 3.6.1 Display current inventory and ask user for confirmation to save
230. # First we display inventory
231. IO.show\_inventory(lstTbl)
232. # Then prompt for verification.
233. #   This could also be made into a method in the IO class or left here.
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.write\_file(lstTbl, strFileName)
240. **else**:
241. input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')

244. **continue**  # start loop back at top.
246. # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
247. **else**:
248. **print**('General Error')

Figure 1 python script

# Save your script to a folder

Inside the course folder create another folder called Assignment07 and save the script document as “Assignment07.py”.

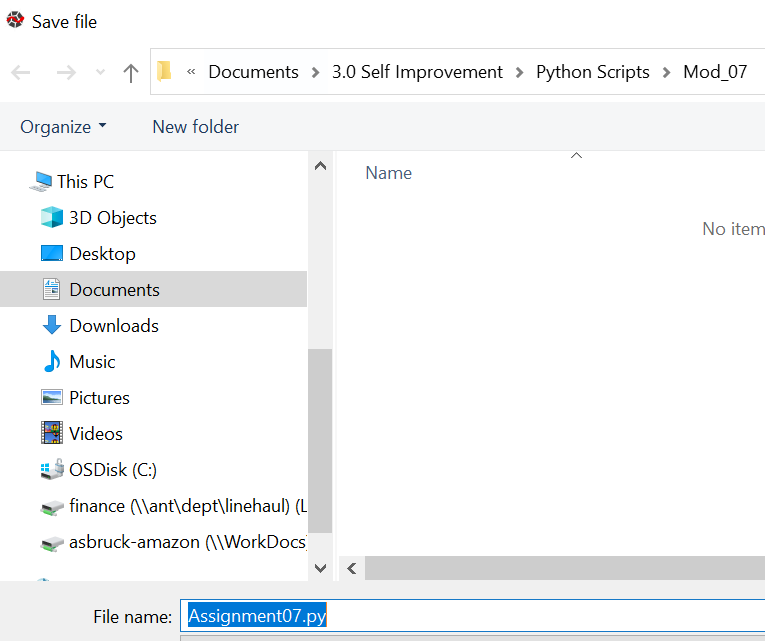


Figure 2 saving a .py extension file

# Run and Test Your Script

1. **Pickle module**
2. Test the pickle.dump using the FileProcessor.write\_file function

We will first add a cd information In order to test the pickle.dump using the FileProcessor.write\_file function.



Figure 3 defining the binary file

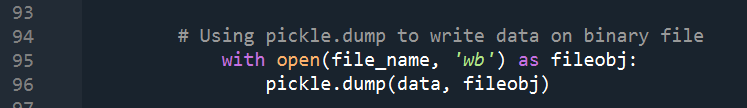


Figure 4 pickle.dump module

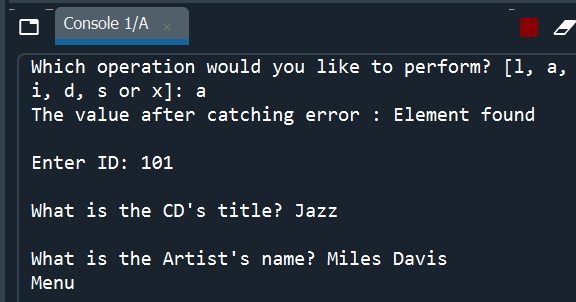
We will create a cd with ID number 101, Title Jazz, Artist Miles Davis  


Figure 5 creating cd information

We will write the data by calling the FileProcessor.write\_file function by providing a user input ‘S’.

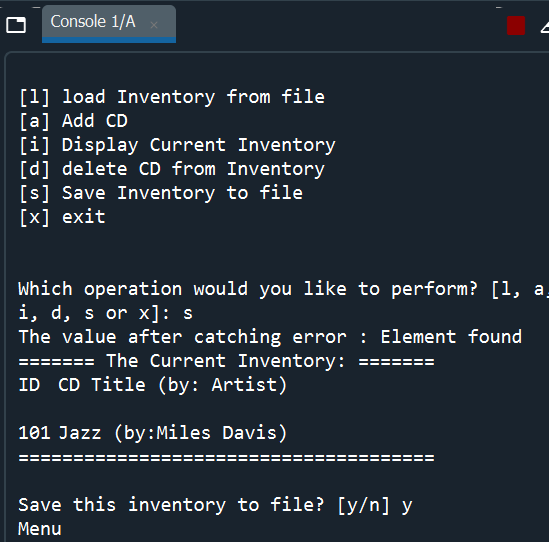


Figure 6 saving to binary file

1. Test the pickle.load using the FileProcessor.read\_file function

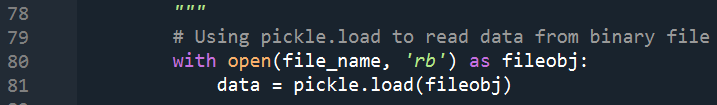


Figure 7 pickle.load module

We will load the cd information we have added by using the user input ‘l’. If the IDE returns the CD information we provided, it should serve as a proof that the pickle.dump and pickle.load module we used under the FileProcessor.write\_file and FileProcessor.read\_file works.

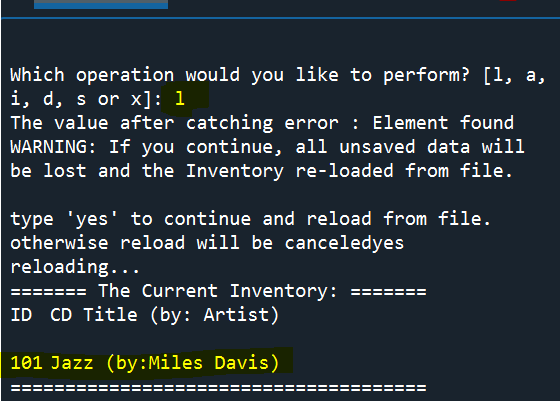


Figure 8 loading cd information

1. **Error handling using Try Except**

We will use Try Except to catch a wrong user input that is outside of the provided menu choices.

We will create a choice\_list and catch any user inputs provided outside of the Menu\_list provided and return an except error handling message “Element not in list !”. If the user inputs a choice among the menu\_list provided it will return “Element found” and return choice.

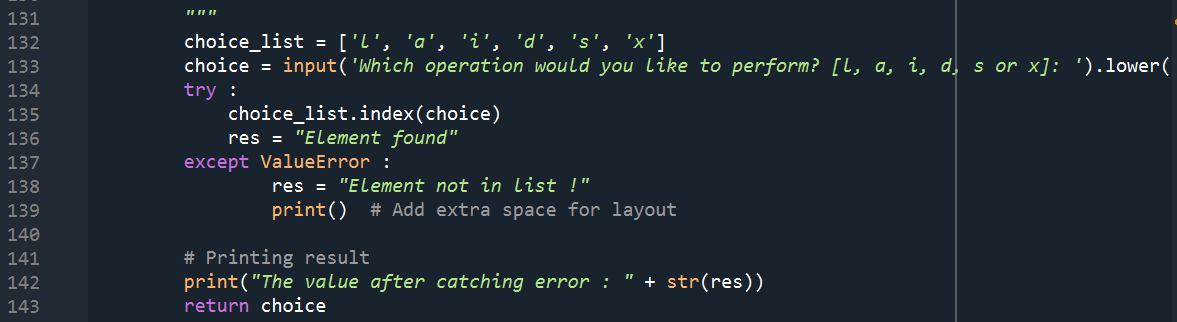


Figure 9 script for try except error handling

Testing the try except error handling script by providing an input that is not found under the list.

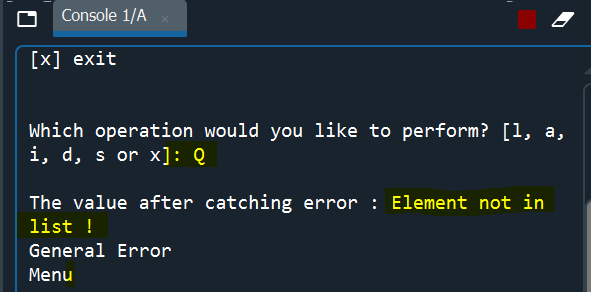


Figure 9 trying to break the code by providing a wrong input

Testing the try except error handling by providing an input that is found in the list.

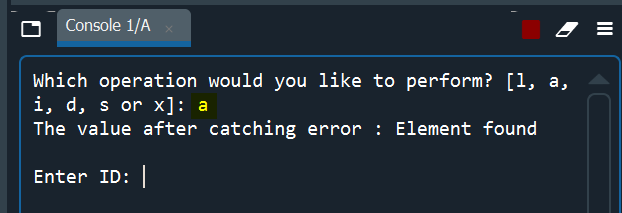


Figure 10 testing with a correct input

# Summary

The main challenge I had starting out the assignment was not removing line 177 from the original cdinventory-DK-5 file.

# 

Figure 11 trying to break the code by providing a wrong input

I believe this function was executing before I was able to call the write\_file function. It would return the below message.

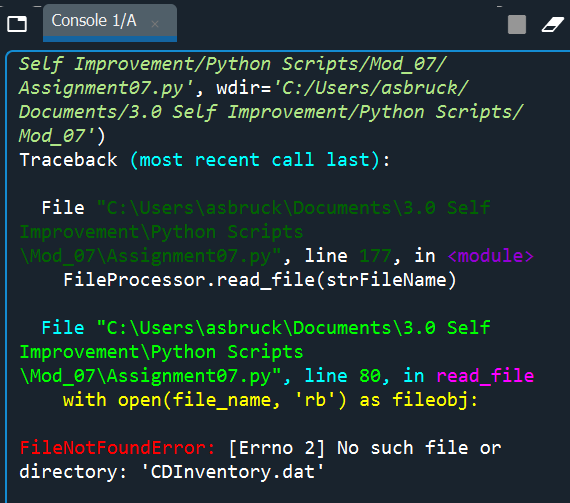


Figure 12 error message

I also spent several hours creating a script that would catch d/nt character types. Instead I was able to think of creating a list and catching an input provided by the user outside of the menu\_list. I believe this approach is most appropriate as we have already provided permittable menu list.

# Reference

* FDN\_Py\_Module\_06 file from IT FDN 110 A Module 07 Overview materials
* <https://docs.python.org/3/tutorial/errors.html>
* <https://docs.python.org/3/tutorial/errors.html>
* <https://www.w3schools.com/python/python_try_except.asp>