Amanda Larson

November 28, 2021

IT FDN 110 – Foundations of Programming: Python

Assignment 07

Module 07 – Assignment 07

Introduction

Assignment 07 focused on handling errors using try statements with an except clause as well as saving(writing) and retrieving(reading) the data using pickling. We've been using starter scripts for a few weeks while working on our CD inventory script so that we don't have to bother with fixing errors before moving on. This week, however, we were supposed to use our own script from the previous week and modify it with the required additions. In addition to the program modification we needed to our own research on both pickling and handling errors. Usually, the research is done and given to us in a pdf as part of the assignment, but this week it seems very little information was given to us about these topics aside from the module videos. Th book was helpful and explained in detail both topics and gave a number of examples in their scripts, but they seemed basic and their try/exception examples didn't seem to be actually handling anything because it would just move on, without letting the user try again.

Here are a few of the other resources I found that were helpful in learning both topics:

Try/Except:

https://stackoverflow.com/questions/43572508/try-except-and-while-loop-in-python

https://stackoverflow.com/questions/36432954/python-validation-to-ensure-input-only-contains-characters-a-z

https://www.idkrtm.com/error-handling-in-python-using-with-and-try/

https://youtu.be/NIWwJbo-9_8

Pickling:

https://stackoverflow.com/questions/7501947/understanding-pickling-in-python

https://docs.python.org/3/library/pickle.html

https://stackoverflow.com/questions/24791987/why-do-i-get-pickle-eoferror-ran-out-of-input-reading-an-empty-file

What went well

I really enjoyed learning about pickling. It kind of made everything we've done for the past couple of weeks seem a lot more complicated time consuming and pickling seems like the easier way to do things. That is, if you don't need to see your data in a word document or csv. If you don't need to see your data, you just want to save it, pickling is the way to go because it saves space and takes less time than formatting it into a string with commas to put in a .txt or .csv. I was scared to delete the code we had worked on that formatted the data in order to write to file but since you can pickle anything in any format, we really don't need it to be formatted as a string with commas and we can delete that section of code from our program. Once I figured that out, it was easy: just pickle.dump() or pickle.load() and you've got it saved and retrieved from file! Pretty cool!

```
104             objFile = open(strFileName, 'w')
105             for row in lstTbl:
106                 lstValues = list(row.values())
107                 lstValues[0] = str(lstValues[0])
108                 objFile.write(',',.join(lstValues) + '\n')
109             objFile.close()
```

Figure 1 - A snippet of code from the prior week where we needed to format the data for saving to a text file.

```
119             try:
120                 objFile = open(strFileName, 'wb')
121                 pickle.dump(table, objFile)
122             except IOError:
123                 print('Unable to open/write to file.')
124             finally:
125                 objFile.close()
```

Figure 2 - A snippet of code from this week where we didn't need to format for a text file, but we needed to both pickle and use a try/exception.

```
Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit

Which operation would you like to perform? [l, a, i, d, s or x]: a

Enter ID: g
The ID must be a number. Please enter a numeric ID.
Enter ID: 3
```

Figure 3 - A screenshot of the program running in Anaconda prompt where the try/exception leads the user to enter in only a number for the ID.

```
Menu

[l] load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] delete CD from Inventory
[s] Save Inventory to file
[x] exit


Which operation would you like to perform? [l, a, i, d, s or x]: a


Enter ID: y
The ID must be a number. Please enter a numeric ID.

Enter ID: 5
```

Figure 4 - The same section of code as in Figure 3, but running in Spyder.

What didn't go well

The error handling isn't as clear to me. I understand how to write it into code, but I don't know why you'd use it. It seemed redundant in a lot of place in my program to add error handling when there was already code that wouldn't allow for errors to exist. I'm not sure why you would use a try/except when you could use an if/elif statement, for example. Another example is in the menu choice function… the code uses a while loop that prevents the user from

selecting anything but the menu choices. It doesn't seem like you need a try/except block here because the loop will continue to ask the user for an input if they don't input one of the correct menu options. I talked about it with Armando for a while this weekend and any real-world example we came up with for a try/except we could see a different way to handle it using a while loop or an if loop or some other way. I was able to get the try/exceptions to work in my script. I was even able to go a bit beyond and use try/except/finally. I just don't see the point. Even some forums online were advising people on their code and removing the try/except in exchange for an if/elif. Hopefully we can go over this in class on Tuesday.

Summary

The assignments seem to be getting easier as we practice using python to improve upon our same CDInventory program. I'm starting to "think in python" which makes it a little bit more intuitive. I've also been enjoying meeting with other students and talking about it in the zoom meetings and the discord chat. It helps to solidify what I know when I can help teach other students. I know that it helps others to do the same, so I feel like there are a few of us, at least, who are starting to really get it and who are able to start asking better questions during class time. Really enjoying the class and I'm glad that I'm starting to "get it". Its impressive to me to look back and our CDInventory program from even just a few weeks ago and see how much more sophisticated it is now.

Appendix

I used a site called (Highlight your source code, 2021)[1] to format the code into the colored version. See below:

```
1.  #------------------------------------------#
2.  # Title: Assignment06.py
3.  # Desc: Working with classes and functions.
4.  # Change Log: (Who, When, What)
5.  # DBiesinger, 2030-Jan-01, Created File
6.  # ALarson, 2021-Nov-20, Moved code blocks into functions and created doctext
7.  # ALarson, 2021-Nov-21, Troubleshooted new code errors after moving into functions
8.  # ALarson, 2021-Nov-27, Modified file processing to pickling/unpickling data instead
9.  #                       of writing to textfile, added error handling functionality,
10. #                       updated docstrings to reflect changes in processsing.
11. #------------------------------------------#
12.
13. import sys
14. import pickle
15.
16.
17.
18. # -- DATA -- #
19. strChoice = '' # User input
20. lstTbl = []  # list of lists to hold data
21. dicRow = {}  # list of data row
22. strFileName = 'CDInventory.dat'  # pickled data storage file
23. objFile = None  # file object
24. strTitle = ''
25. stArtist = ''
26. lstValues = []
```

[1] https://highlight.hohli.com/index.php

```
27.
28. # -- PROCESSING -- #
29. class DataProcessor:
30.     """The DataProcessor class processes the user data."""
31.
32.     @staticmethod
33.     def add_CD(strID, strTitle, stArtist):
34.         """Function to add a new user-defined CD to the table in-memory.
35.
36.         Args:
37.             intID: the CD ID number as provided by the user
38.             strTitle: the CD title as provided by the user
39.             stArtist: the CD artist as provided by the user
40.
41.         Returns:
42.             None.
43.         """
44.
45.         intID = int(strID)
46.         dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
47.         lstTbl.append(dicRow)
48.
49.
50.     @staticmethod
51.     def delete_Row(intIDDel):
52.         """ Function to delete row in table.
53.
54.         Args:
55.             intIDDel: the ID of the CD to delete
56.
57.         Returns:
58.             None.
59.
60.         """
61.         intRowNr = -1
62.         blnCDRemoved = False
63.         for row in lstTbl:
64.             intRowNr += 1
65.             if row['ID'] == intIDDel:
66.                 del lstTbl[intRowNr]
67.                 blnCDRemoved = True
68.                 break
69.         if blnCDRemoved:
70.             print('The CD was removed')
71.         else:
72.             print('Could not find this CD!')
73.
74.
75. class FileProcessor:
76.     """Processing (pickling/unpickling) the data to and from datfile"""
```

```
77.
78.     @staticmethod
79.     def read_file(file_name, table):
80.         """Function to manage data ingestion from a pickled datfile to a list of dictionaries
81.
82.         Reads the data from datfile identified by file_name into a 2D table
83.         (list of dicts) table one line in the file represents one dictionary row in table.
84.
85.         Args:
86.             file_name (string): name of datfile used to read the data from
87.             table (list of dict): 2D data structure (list of dicts) that holds the data
    during runtime
88.
89.         Returns:
90.             None.
91.         """
92.         table.clear()  # this clears existing data and allows to load data from file
93.
94.         try:
95.             objFile = open(file_name, 'rb')
96.             table = pickle.load(objFile)
97.             return table
98.         except IOError:
99.             print('Unable to open the file', file_name,'.'.strip(), 'Ending program.\n')
100.                input('\n Press Enter to exit.')
101.                sys.exit()
102.            except:
103.                print('Unknown Error')
104.            finally:
105.                objFile.close()
106.
107.
108.        @staticmethod
109.        def write_file(file_name, table):
110.            """Function to manage the saving of pickled data to a datfile.
111.
112.            Args:
113.                file_name (string): name of file to be saved, presumably the same as has
    already been loaded in.
114.                table (list of dictionaries): 2D data structure (list of dicts) that holds
    the data during runtime
115.
116.            Returns:
117.                None.
118.                """
119.            try:
120.                objFile = open(strFileName, 'wb')
121.                pickle.dump(table, objFile)
122.            except IOError:
123.                print('Unable to open/write to file.')
```

```
124.             finally:
125.                 objFile.close()
126.
127.
128.
129.     # -- PRESENTATION (Input/Output) -- #
130.
131.     class IO:
132.         """Handling Input / Output"""
133.
134.         @staticmethod
135.         def print_menu():
136.             """Displays a menu of choices to the user
137.
138.             Args:
139.                 None.
140.
141.             Returns:
142.                 None.
143.             """
144.
145.             print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current
    Inventory')
146.             print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
147.
148.         @staticmethod
149.         def menu_choice():
150.             """Gets user input for menu selection
151.
152.             Args:
153.                 None.
154.
155.             Returns:
156.                 choice (string): a lower case sting of the users input out of the choices
    l, a, i, d, s or x
157.
158.             """
159.             try:
160.                 choice = ' '
161.                 while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
162.                     choice = input('Which operation would you like to perform? [l, a, i,
    d, s or x]: ').lower().strip()
163.                 print()  # Add extra space for layout
164.                 return choice
165.             except TypeError:
166.                 print('Numbers aren\'t allowed, please choose a letter from the menu
    choices above.')
167.
168.
169.         @staticmethod
```

```
170.         def show_inventory(table):
171.             """Displays current inventory table
172.
173.
174.             Args:
175.                 table (list of dict): 2D data structure (list of dicts) that holds the
    data during runtime.
176.
177.             Returns:
178.                 None.
179.
180.             """
181.             print('======= The Current Inventory: =======')
182.             print('ID\tCD Title (by: Artist)\n')
183.             for eachDict in table:
184.                 print('{}\t{} (by:{})'.format(*eachDict.values()))
185.             print('======================================')
186.
187.
188.         @staticmethod
189.         def get_CD():
190.             """Function to get user input/information for the CD ID number, the CD title,
    and the CD artist.
191.
192.             Args: None
193.
194.             Returns:
195.                 a tuple of the user's input: strID, strTitle, stArtist
196.
197.             """
198.             while True:
199.                 try:
200.                     strID = int(input('Enter ID: ').strip())
201.                     strID = str(strID)
202.                     break
203.                 except ValueError:
204.                     print('The ID must be a number. Please enter a numeric ID.')
205.
206.             strTitle = input('What is the CD\'s title? ').strip()
207.             stArtist = input('What is the Artist\'s name? ').strip()
208.             return strID, strTitle, stArtist #returns a tuple
209.
210.     # 1. When program starts, read in the currently saved Inventory
211.     lstTbl = FileProcessor.read_file(strFileName, lstTbl)
212.
213.
214.     # 2. start main loop
215.     while True:
216.         # 2.1 Display Menu to user and get choice
217.         IO.print_menu()
```

```
218.          strChoice = IO.menu_choice()
219.
220.          # 3. Process menu selection
221.          # 3.1 process exit first
222.          if strChoice == 'x':
223.              break
224.          # 3.2 process load inventory
225.          if strChoice == 'l':
226.              print('WARNING: If you continue, all unsaved data will be lost and the
    Inventory re-loaded from file.')
227.              strYesNo = input('type \'yes\' to continue and reload from file. otherwise
    reload will be canceled')
228.              if strYesNo.lower() == 'yes':
229.                  print('reloading...')
230.                  lstTbl = FileProcessor.read_file(strFileName, lstTbl)
231.                  IO.show_inventory(lstTbl)
232.              else:
233.                  input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue
    to the menu.')
234.                  IO.show_inventory(lstTbl)
235.              continue  # start loop back at top.
236.          # 3.3 process add a CD
237.          elif strChoice == 'a':
238.              # 3.3.1 Ask user for new ID, CD Title and Artist
239.              strID, strTitle, stArtist = IO.get_CD() #unpack and call at the same time
240.              # 3.3.2 Add item to the table
241.              DataProcessor.add_CD(strID, strTitle, stArtist)
242.              IO.show_inventory(lstTbl)
243.              continue  # start loop back at top.
244.          # 3.4 process display current inventory
245.          elif strChoice == 'i':
246.              IO.show_inventory(lstTbl)
247.              continue  # start loop back at top.
248.          # 3.5 process delete a CD
249.          elif strChoice == 'd':
250.              # 3.5.1 get Userinput for which CD to delete
251.              # 3.5.1.1 display Inventory to user
252.              IO.show_inventory(lstTbl)
253.              # 3.5.1.2 ask user which ID to remove
254.              while True:
255.                  try:
256.                      intIDDel = int(input('Which ID would you like to delete? ').strip())
257.                      break
258.                  except ValueError:
259.                      print('Please type an ID (number) to delete')
260.              # 3.5.2 search thru table and delete CD
261.              blnCDRemoved = False
262.              DataProcessor.delete_Row(intIDDel)
263.              IO.show_inventory(lstTbl)
264.              continue  # start loop back at top.
```

```
265.            # 3.6 process save inventory to file
266.            elif strChoice == 's':
267.                # 3.6.1 Display current inventory and ask user for confirmation to save
268.                IO.show_inventory(lstTbl)
269.                strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
270.                # 3.6.2 Process choice
271.                if strYesNo == 'y':
272.                    FileProcessor.write_file(strFileName, lstTbl)
273.                else:
274.                    input('The inventory was NOT saved to file. Press [ENTER] to return to the
    menu.')
275.                continue  # start loop back at top.
276.            # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to
    be save:
277.            else:
278.                print('General Error')
```