Sheena Harms

Nov. 29, 2022

IT FDN 110 B

Assignment 07

Error Handling and Pickling

# Introduction

I will talk about some of the places I found good information on pickling and error handling for Python. I will also lay out the steps I took to incorporate those two concepts to my existing script.

# Pickling Research

I felt the pickling process was best described by the youtube tutorial (<https://youtu.be/XzkhtWYYojg>) and webpage (<https://realpython.com/python-pickle-module/>) at RealPython. The two are in different formats but reiterate similar points across platforms which helped me to understand. The presenter defined a class of different types, made an instance of that object, and pickled it. He then changed the original object to make a property = None. Then he unpickled the byte string and looked at the object’s property. A few things he glossed over, which I don’t now about, were setting environments and f string formatting. He did a good job of explaining the four key methods of the pickle module. The only other problem with the Realy Python site is that there is a limit to how many times you can view their pages without an account.

I also watched a video (<https://youtu.be/Pl4Hp8qwwes>) by Mark Jay, which I didn’t learn much from. The difference really was that he ran through a process and spent much less time explaining any of the reasons behind it.

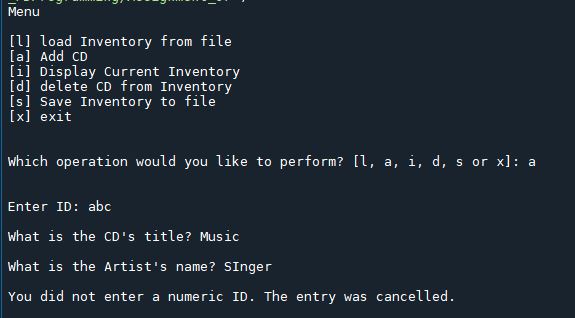


Figure 1: Exception Handling – adding to the inventory..

# Exception Handling Research

The book did a really good job of explaining this concept to me. I learned the most from adding try-except clauses to different areas in the script and testing various errors. The Real Python site (<https://realpython.com/python-exceptions/#the-try-and-except-block-handling-exceptions>) again did a nice job of explaining exceptions. I found it helpful that they differentiated between exceptions and syntax errors. The try and except block was clearly laid out in a graphic. The did mention that using bare except clauses isn’t a best practice, because it tells the user nothing about what went wrong. I think I left a few of those in my code because I wanted to use them as a catch-all, and the user would be returned to the menu anyway.

There were several other sites which did a similarly good job of explaining error handling. One which wasn’t particularly helpful, because it was so sparse and gave no context, was w3schools wiki (<https://www.w3schools.com/python/gloss_python_error_handling.asp>) on the subject of error handling.

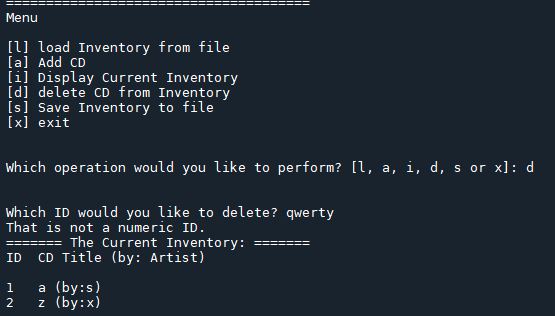


Figure – Exception handling – deleting a row.

# Putting it Into Action

For assignment 7 I began by cleaning up the mistakes I had made the week prior. Refining a script every week can be a very slippery slope if you don’t grasp a concept.

I placed some error handling in the processing script at the bottom, for deleting a row. I then placed some error handling in the functions for adding data. I realized that most of the error handling I wanted to add could be put into the functions. For the sake of consistency I moved the delete row scripting into an IO function and added the error handling there. I tested each try-except clause to ensure that my error messages would come through. When I was satisfied I re-read the assignment and noticed that I needed to add a pickling component to the script.

Here I was flummoxed for quite a while. I wanted to use load near the read file and dump near the write file. However, I wasn’t sure how to use a predefined function within my functions. It turned out to be fairly simple. I created a .dat file to replace the .txt I had been using. Then I replaced that within the strfileName definition. For writing I used the dump method and for reading I used load.

It is worth noting that although the script worked in Spyder I couldn’t get it to run in the power shell. I believe this is because I am using my work laptop and don’t have the admin privileges to write things in with a .dat file. It will be interesting to find out if this is actually an incorrect assumption, once I get my 7th assignment graded and commented on.

# Summary

Error handling is an excellent way to keep your user informed while allowing them to continue using a program. Best practices will require I become more comfortable with handling exceptions in the future. Pickling is a more niche process, which allows you to serialize objects.

\*See python and knowledge document for assignment 6 at <https://github.com/sheenahar/Assignment_07>

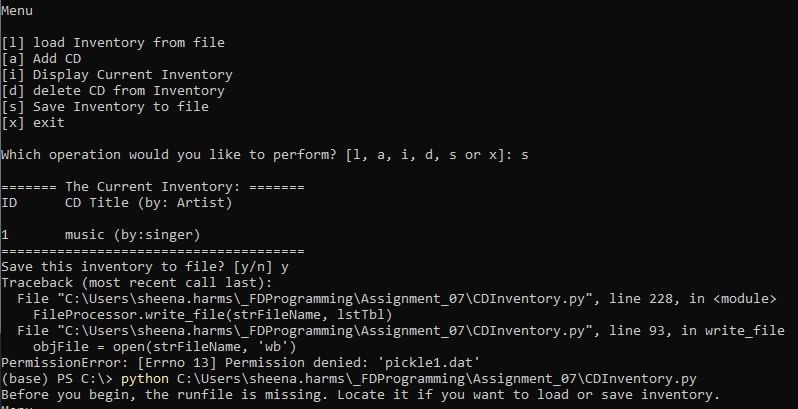


Figure 3: CD Inventory script in terminal.

# Appendix

## Listing CD\_Inventory.py

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235 | *#------------------------------------------#*  *# Title: Assignment06\_Starter.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# SHarms, 2022-Nov-18, Edited file*  *# SHARMS, 2022-Nov 27, Edited to add exception handling*  *#------------------------------------------#*  **import** **pickle**  *# -- DATA -- #*  strChoice = '' *# User input*  lstTbl = [] *# list of lists to hold data*  dicRow = {} *# list of data row*  strFileName = 'pickle1.dat' *# data storage file*  objFile = **None** *# file object*  *# -- PROCESSING -- #*  **class** **DataProcessor**:    @staticmethod  **def** del\_Row(intIDDel, lstTbl):  *"""Function to remove files from table*  *Reads through rows for the input ID*  *Deletes row if input ID is found*  *Returns:*  *None"""*  intRowNr = -1  blnCDRemoved = **False**  **for** row **in** lstTbl:  intRowNr += 1  **if** row['ID'] == intIDDel:  **del** lstTbl[intRowNr]  blnCDRemoved = **True**  **break**  **if** blnCDRemoved:  print('The CD was removed')  **else**:  print('Could not find this CD!')    @staticmethod  **def** add\_cd(strID, strTitle, strArtist, lstTbl):  *"""Function adds data to dictionary row*  *Appends to list*  *Returns:*  *None."""*  **try**:  intID = int(strID)  **except** **ValueError**:  print("**\n**You did not enter a numeric ID. The entry was cancelled.**\n**")  **else**:  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}  lstTbl.append(dicRow)  **class** **FileProcessor**:  *"""Processing the data to and from binary file"""*  @staticmethod  **def** read\_file(file\_name, table):  *"""Function to manage data ingestion from file to a list of dictionaries*  *Reads the data from file identified by file\_name into a 2D table*  *(list of dicts) table one line in the file represents one dictionary row in table.*  *Args:*  *file\_name (string): name of file used to read the data from*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *Returns:*  *None.*  *"""*  **try**:  objFile = open(file\_name, 'rb')  table.clear()  data = pickle.load(objFile)  **for** row **in** data:  table.append(row)  *#print(table)*  **except**:  print("Before you begin, the runfile is missing. Locate it if you want to load or save inventory.")  **else**:  objFile.close()  @staticmethod  **def** write\_file(strfileName, lstTbl):  *"""Writes from list of dicts to a binary file.*  *Args:*  *strfileName: file name*  *lstTbl: list of lists*  *Returns:*  *None"""*  objFile = open(strFileName, 'wb')  pickle.dump(lstTbl, objFile)  objFile.close()  *# -- PRESENTATION (Input/Output) -- #*  **class** **IO**:  *"""Handling Input / Output"""*  @staticmethod  **def** print\_menu():  *"""Displays a menu of choices to the user*  *Args:*  *None.*  *Returns:*  *None.*  *"""*  print('Menu**\n\n**[l] load Inventory from file**\n**[a] Add CD**\n**[i] Display Current Inventory')  print('[d] delete CD from Inventory**\n**[s] Save Inventory to file**\n**[x] exit**\n**')  @staticmethod  **def** menu\_choice():  *"""Gets user input for menu selection*  *Args:*  *None.*  *Returns:*  *choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x*  *"""*  choice = ' '  **while** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:  choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: ').lower().strip()  print() *# Add extra space for layout*  **return** choice  @staticmethod  **def** show\_inventory(table):  *"""Displays current inventory table*  *Args:*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.*  *Returns:*  *None.*  *"""*  print('======= The Current Inventory: =======')  print('ID**\t**CD Title (by: Artist)**\n**')  **for** row **in** table:  print('**{}\t{}** (by:**{}**)'.format(\*row.values()))  print('======================================')  @staticmethod  **def** cd\_Data():  *"""Asks user for cd data inputs*  *Args:*  *None.*  *Returns:*  *None.*  *"""*  *# 3.3.1 Ask user for new ID, CD Title and Artist*  strID = input('Enter ID: ').strip()  strTitle = input('What is the CD**\'**s title? ').strip()  strArtist = input('What is the Artist**\'**s name? ').strip()  DataProcessor.add\_cd(strID, strTitle, strArtist, lstTbl)  IO.show\_inventory(lstTbl)    @staticmethod  **def** del\_Choice():  *"""Asking user what they want to delete from the inventory.*  *Args:*  *None.*  *Returns:*  *None.*  *"""*  *# 3.5.1.2 ask user which ID to remove*  **try**:  intIDDel = int(input('Which ID would you like to delete? ').strip())  **except** **ValueError**:  print("That is not a numeric ID.")  **except**:  print("There was an error. Please begin again")  *# 3.5.2 search thru table and delete CD*  **else**:  DataProcessor.del\_Row(intIDDel, lstTbl)  IO.show\_inventory(lstTbl)    *# 1. When program starts, read in the currently saved Inventory*  FileProcessor.read\_file(strFileName, lstTbl)  *# 2. start main loop*  **while** **True**:  *# 2.1 Display Menu to user and get choice*  IO.print\_menu()  strChoice = IO.menu\_choice()  *# 3. Process menu selection*  *# 3.1 process exit first*  **if** strChoice == 'x':  **break**    *# 3.2 process load inventory*  **if** strChoice == 'l':  print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')  strYesNo = input('type **\'**yes**\'** to continue and reload from file. otherwise reload will be canceled')  **if** strYesNo.lower() == 'yes':  print('reloading...')  FileProcessor.read\_file(strFileName, lstTbl)  IO.show\_inventory(lstTbl)  **else**:  input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')  IO.show\_inventory(lstTbl)  **continue** *# start loop back at top.*    *# 3.3 process add a CD*  **elif** strChoice == 'a':  IO.cd\_Data()  **continue** *# start loop back at top.*    *# 3.4 process display current inventory*  **elif** strChoice == 'i':  IO.show\_inventory(lstTbl)  **continue** *# start loop back at top.*    *# 3.5 process delete a CD*  **elif** strChoice == 'd':  *# 3.5.1 get Userinput for which CD to delete*  IO.del\_Choice()  *# 3.5.1.1 display Inventory to user*  IO.show\_inventory(lstTbl)  **continue** *# start loop back at top.*    *# 3.6 process save inventory to file*  **elif** strChoice == 's':  *# 3.6.1 Display current inventory and ask user for confirmation to save*  IO.show\_inventory(lstTbl)  strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()  *# 3.6.2 Process choice*  **if** strYesNo == 'y':  *# 3.6.2.1 save data*  FileProcessor.write\_file(strFileName, lstTbl)  **else**:  input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')  **continue** *# start loop back at top.*    *# 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be safe:*  **else**:  print('General Error') |