Miroslava Meza

03/13/2022

IT FDN 110B Wi 22: Foundations of Programming Python

Assignment 07, Module 0

# Reading and Writing Text File vs. Binary File; Structured Error Handling, Exception Handling, Pickling

## Introduction

This assignment requires\d to modify the previous assignment by adding structured error handling around the areas where there is user interaction, type casting (string to int) or file access operations. This assignment requested to use python built in error classes and not creating custom error classes.

Finally, this script was modified to store cd-inventory using binary data.

## Github repository

This assignment is stored in two different places, (1) <https://github.com/Mirka3m-Sea/Assignment_07.git> and (2) CANVAS.

## Error Handling

The first task for this assignment was to identify the areas where there is user input that may produce errors. In this step the first challenge was to properly identify where the error handling was necessary. I identified two areas for error handling (a) inputting ID number for the CD both at adding new entries and deleting information. [[1]](#footnote-1)

Figure 1Handling the error of non-numerical data input for CD-ID.

Text

Description automatically generated

Figure 2Handling the error in the function for deleting data.

Text

Description automatically generated

Using the Help window in Spyder to find definitions and correct arguments.

Graphical user interface, text, application

Description automatically generated *A screenshot of a computer

Description automatically generated with medium confidence*

*Challenge: Identifying the exception type to be used, there were multiple options. For this assignment I used the following exception types: Exception, and ValueError.*

*Solution: I tested both and kept the one that I produced the expected values.*

|  |
| --- |
|  |

## Storing into binary data

For storing data into binary protocols, I researched and used Python’s [**pickle**](https://docs.python.org/3/library/pickle.html#module-pickle)**module. [[2]](#footnote-2)**

*“The pickle module implements binary protocols for serializing and de-serializing a Python object structure.*“Pickling”*is the process whereby a Python object hierarchy is converted into a byte stream, and*“unpickling”*is the inverse operation, whereby a byte stream (from a*[*binary file*](https://docs.python.org/3/glossary.html#term-binary-file)*or*[*bytes-like object*](https://docs.python.org/3/glossary.html#term-bytes-like-object)*) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” or “flattening”; however, to avoid confusion, the terms used here are “pickling” and “unpickling”.”* 3.10.02 Python documentation.

*Challenges: When using pickle module, I needed to modify the data structure from “directory” to binary. I tried unsuccessfully to have the data as binary-type within a directory. At the end, striping the dictionary and not reading lines, but rows was what worked. Second, I could not load the file (reading from file) without using the extension ‘wb”, I also tried with ‘ar”.*

Figure 3. Reading a Binary File

*Text

Description automatically generated*

Figure 4. Writing inventory into a binary data file

*Text

Description automatically generated*

## Running the Script in Spider

The first challenge I faced was that the script would yield an error message if I did not have a .txt file on the same repository. I eliminated that issue by opening/closing or creating/closing the txt file at the beginning of the script. That way, regardless of the sequence chosen by the user there will always be a destination file.

Testing sequence.

1. Started using the option [a] adding one more inventory entry.
2. [d] to erase an entry, handling data input error by the user.
3. [s] to save the information to the binary-data external file.
4. [l] from the menu, uploading binary data.
5. [x] to close the script.

Figure 5. Loading information from file. Binary data form.

Text

Description automatically generated

On figure 5, you can see that when the user is prompted back to the menu when they input an invalid CD Id to remove from the list.

Figure 6. Handling input error, wrong CD ID.

Text

Description automatically generated

.

## Running the Script in Terminal

Testing sequence.

1. Started using the option [a] adding one more inventory entry.
2. [d] to erase an entry, handling data input error by the user.
3. [s] to save the information to the binary-data external file.
4. [l] from the menu, uploading binary data.
5. [x] to close the script.

Figure 7. Error handling, deleting an entry that does not exist.

Text

Description automatically generated

Figure 8. Saving and Loading information from file, binary data type..

Text

Description automatically generated

## Conclusions

This assignment required to modify the previous assignment by adding structured error handling around the areas where there is user interaction, type casting (string to int) or file access operations.

In this assignment I learned to use python built in error classes to handle errors introduced by the user, and how to store information as a binary data type. The main challenges I faced were to identify the type of error that the user may introduce and to find the correct way to handle it. Also, it was difficult to modify the while loops when using the “pickle” module.

## Appendix Full Script

|  |  |
| --- | --- |
| 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  236  237  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253  254  255  256  257  258  259  260  261  262 | *#------------------------------------------#*  *# Title: Assignment07.py*  *# Desc: Starting with Assig06,*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *#Miroslava Meza, 2022-Mar-06, Included classes and functions*  *#Miroslava Meza, 2022-Mar-06, Eliminated TODO marks, pass, and redundant code lines*  *#Miroslava Meza, 2022-Mar-12, Corrected from HW6, clearing any previously saved inventory txt by using append.*  *#Miroslava Meza, 2022-Mar-13, Modified lines for user input to include error handling, and data storage on binary type*  *#------------------------------------------#*  *# -- DATA -- #*  strChoice = '' *# User input*  lstTbl = [] *# list of lists to hold data*  dicRow = {} *# list of data row*  strFileName = 'CDInventory.txt' *# data storage file*  objFile = **None** *# file object*  cdData =[**None**, **None**, **None**] *# Saving data in memory as a list, starts with no values.*  dicRow= {'ID': **None**, 'Title':**None**, 'Artist':**None**} *#start the dictionary with no values*  *#Importing module*  **import** **pickle**  *# -- PROCESSING -- #*  **class** **DataProcessor**:  *#"""*  *# Action-1 add functions for processing here*  *#using the functions identified in the starter code as:*  *# 3.3.2, 3.5.2, 3.6.2.1*  **def** add\_data():  *"""*  *Add and save data entries*  *Using the list type variable named 'cdData'*  *variable names were already in dicRow*    *Args: intID, strTitle, stArtist, IO.cdData, IO.show\_inventory, lstable, dicRow*  *Returns: inventory list with new entry.*    *Note: read and write functions are on the next class.*  *"""*  intID, strTitle,stArtist =IO.data\_input()  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}  lstTbl.append(dicRow)  IO.show\_inventory(lstTbl)      **def** delete\_data(intIDDel):  *"""*  *This function allows to erase a data entry.*  *Args: lstTbl, intIDDel, intRowNr*  *IO.show\_inventory*    *ASSIGNMENT 07- If the ID provided by user is invalid or not Int type.*  *Substituted the previous option of "else:"l*    *"""*  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**\n**')  **try**:  **if** blnCDRemoved == **True**:  print('CD removed**\n**')  **if** blnCDRemoved == **False**:  **raise** **Exception**  **except** **Exception**:  print('CD not found **\n**Going to main menu**\n** ')  **return** intIDDel  IO.show\_inventory(lstTbl)    **class** **FileProcessor**:  *"""Processing the data to and from text file*  *ASSIGNMENT 07- Storing the information into binary data*  *"""*  @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=strFileName into a 2D table*  *(list of dicts) table one line in the file represents one dictionary row in table.*  *Args:*  *strFileName (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.*  *\*\*\* file\_name will be tied to the variable strFileName*    *ASSIGNMENT 07: Store inventory into binary data type. Use Picking module*  *"""*  objFile = open(file\_name, 'ab') *# It did not work if I used 'rb'*  pickle.dump(table, objFile) *# pickle.dump(obj, file, protocol=None, \*, fix\_imports=True, buffer\_callback=None)*  *# #Return the pickled representation of the object obj as a bytes object, instead of writing it to a file.*  print('You have a new binary file ready to work.**\n**')  objFile.close()    @staticmethod  **def** write\_file(file\_name, table):  *"""*    *Parameters*  *file\_name (String) : destination file.*  *table (List): Inventory data appended through out the script*  *Returns: None*  *#-----------------Assignment 07 ----------------#*  *#First I corrected my missuse of a global variable within the function. Followed*  *#Laura's feedback to Assignment 06*  *ASSIGNMENT 07: Modified this function to store/write data into a binary type.*  *"""*  objFile = open(file\_name, 'wb') *# Writing on a binary data type file. #@Laura, I corrected my error from HW6*  pickle.dump(table, objFile) *### @Laura: I made the mistake of adding file\_name instead of objFile and corrected!*  objFile.close() *# Closing as fast as possible.*  print('Inventory saved**\n**')  *# -- 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('======================================')    **def** data\_input() :  *"""*  *This function request the user to input data for each CD.*  *Arg: strID, intID, strTitle, stArtist. All of these entries are held*  *in the internal memory of this function. THey are no global variables.*  *Returns: None*    *ASSIGNMENT 07: User Input Error Handling for ID, when user adds a non numerical ID.*  *Used the following operators: while, try, break*  *"""*  **while** **True**:  strID = input('Enter ID: ').strip()  **try**:  intID= int(strID)  **break**  **except** **ValueError**:  print('Invalid ID entry, type a number**\t**')  strTitle = input('What is the CD**\'**s title? ').strip()  stArtist = input('What is the Artist**\'**s name? ').strip()  **return** [intID, strTitle, stArtist]    *# 1. When program starts, read in the currently saved Inventory*  *"""*  *The script did not run without creating the file to store data. To avoid issues, I will start by*  *ensuring CDInventory.txt exists.*  *"""*  file\_name2=open('CDInventory.txt', 'a') *#@Laura: I fixed this as you recommended, thanks!.*  file\_name2.close()  *#Script starts*  *# 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**\t**')  **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':  *# 3.3.1 Ask user for new ID, CD Title and Artist*  *# DONE move IO code into function, Calling the add data function from DataProcessor*  DataProcessor.add\_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*  *# 3.5.1.1 display Inventory to user*  IO.show\_inventory(lstTbl)  *# 3.5.1.2 ask user which ID to remove*  intIDDel = int(input('Which ID would you like to delete? ').strip())  *# 3.5.2 search thru table and delete CD*  *# DONE moved processing code into function*  DataProcessor.delete\_data(intIDDel)  **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':  *# DONE Called the function write\_file from the Class "FileProcessor"*  FileProcessor.write\_file(strFileName, lstTbl) *# 3.6.2.1 save data*  **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 save:*  **else**:  print('General Error') |

## Appendix II. Sources reviewed for this assignment

[Python - Exceptions Handling (tutorialspoint.com)](https://www.tutorialspoint.com/python/python_exceptions.htm)

[contextlib — Utilities for with-statement contexts — Python 3.10.2 documentation](https://docs.python.org/3/library/contextlib.html?highlight=try)

[pickle — Python object serialization — Python 3.10.2 documentation](https://docs.python.org/3/library/pickle.html)

[Understanding Python Pickling with example - GeeksforGeeks](https://www.geeksforgeeks.org/understanding-python-pickling-example/)

[Python File readline() Method (w3schools.com)](https://www.w3schools.com/python/ref_file_readline.asp)

1. Tutorials point. Base class for all exceptions; [Python - Exceptions Handling (tutorialspoint.com)](https://www.tutorialspoint.com/python/python_exceptions.htm#:~:text=Python%20-%20Exceptions%20Handling%20%20%20%20Sr.No.,all%20built-i%20...%20%2025%20more%20rows%20) [↑](#footnote-ref-1)
2. Source: [pickle — Python object serialization — Python 3.10.2 documentation](https://docs.python.org/3/library/pickle.html) [↑](#footnote-ref-2)