< Sara Lam >

< 11/28/2021 >

< Foundations of Programming (Python) >

< Assignment07>

How I performed Assignment07

# Introduction

In Assignment07, we will research exception handling in Python and pickling in Python and search the web to find examples.

For programming, we will take the CDInventory.py file from Assignment06, add structured error handling around the areas where there is user interaction, type casting (string to int) or file access operations. Also modify the permanent data store to use binary data.

We will continue to use GitHub. We will upload the python script and knowledge document on the GitHub repository named “Assignment\_07”, commit changes, share a link on canvas, add link to this knowledge document. We will also perform a peer review of another student’s assignment.

We then save the script file as CDIntentory.py into folder “Assignment07”. Submit this folder as a zip file, including the python script as well as this knowledge document.

# Assignment07\_A – research exception handling in Python

This link is helpful because it shows different kinds of exceptions, and how to handle: <https://docs.python.org/3/tutorial/errors.html>

# Assignment07\_B – research pickling in Python

This link provides good examples because it contrasts pickling with a file and pickling without a file: <https://www.geeksforgeeks.org/understanding-python-pickling-example/>

Here’s a YouTube video that’s helpful: <https://www.youtube.com/watch?v=2Tw39kZIbhs>

# Assignment07\_C - coding

## Step 1 – examine CDInventory.py from Assignment06

The script contains the solution of Assignment06 as a starter file.

## Step 2 – add and change code based on requirements

I created and modified the code for the following:

* Added exception handling where there is user interaction, including asking user for integer as ID for the CD, asking user for integer to delete the CD, checking whether file exists for loading data.
* Used pickling to write to and read from .dat file.
* Saved the script as CDInventory.py to the Assignment07 folder.

## Step 3 – run the python script file in Spyder, and verify it worked

Click the run button to run the file. It describes what the program does, allows the user to choose from menu, and performs the functions based on user choice. It repeats till user enters exit. When no file exists it handles the error and displays the message. Screenshot is below:

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Figure 1 - Script run in Spyder – when program first started without file and an attempt was made to load the file, it correctly displays message from error handling

Text

Description automatically generated

Figure - Script run in Spyder – when trying to add CDs but not entering integer for ID, it correctly displays error messages

Text

Description automatically generated

Figure - Script run in Spyder – added 2 CDs correctly

Text

Description automatically generated

Figure - Script run in Spyder – saved and loaded file

Text

Description automatically generated

Figure - Script run in Spyder – when user tries to delete but entering invalid CD number, it displays messages correctly

Text

Description automatically generated

Figure 6 - Script run in Spyder – deleted a CD correctly

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Figure – Saving to file. File displays updated CD records

## Step 4 – run the python script file in a terminal window, and verify it worked

In Windows search, I type cmd to open the Command Prompt. Then I change to the directory with my python script, which is C:\\_FDProgramming\Assignment07. I type CDInventory.py to run the program.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Figure 8 - Script run in terminal window – loading CDInventory.dat created in Spyder, displaying error messages, adding, deleting CDs and saving back to file

Graphical user interface, application

Description automatically generated

Figure 9 - Script run in terminal window updated CDInventory.dat

Graphical user interface, text, application

Description automatically generated

Figure 10 - CDInventory.dat correctly shows updates

## Step 5 – upload script and knowledge document to GitHub

# <https://github.com/Lamcloud/Assignment_06>

# Summary

In this assignment we practiced the following -

* Using dictionary to store data
* Using the dictionary values function to get the values
* Using while loop and for loop
* Using conditional statement to control the flow
* Using break to break out of a section
* Reading from and writing to file
* Creating classes and functions for separation of concerns
* Creating docstrings for functions to explain their use
* Using structured error handling
* Writing to and reading from binary file

# Challenges

Below are the challenges I encountered and how I handled them -

* It was challenging to write to and read from a binary file. At first I tried to use a for loop to write a string each time to the binary file like I did for text file, it was written, but I could not read the file properly, probably due to ‘\n’ for line return at the end. I googled for examples, and realized I could just pickle.dump the entire list of dictionaries into the file. Then when reading it, I would read the entire file into a list of dictionaries. Then later process data from that list of dictionaries.
* When using read\_file to read the file while it doesn’t exist, even though I used exception handling, at first it would lead to error in adding items. I realized when the file doesn’t exist, I needed to return an empty table, instead of not returning anything because without return anything the table would have type of None and the append function would not work.

# Appendix – the code

I used this syntax highlighting application, <http://hilite.me/>.

|  |  |
| --- | --- |
| 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  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286 | *#------------------------------------------#*  *# Title: CDInventory.py*  *# Desc: Working with classes, functions, exceptions. Write to and read from binary file.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# SLam, 2020-Nov-28*  *#------------------------------------------#*  **import** **pickle**  *# -- DATA -- #*  strChoice = '' *# User input*  lstTbl = [] *# list of lists to hold data*  dicRow = {} *# list of data row*  strFileName = 'CDInventory.dat' *# data storage file*  objFile = None *# file object*  *# -- PROCESSING -- #*  **class** **DataProcessor**:  *# TODone add functions for processing here*  *"""Processing the data in list of dicts"""*    @staticmethod  **def** add\_item(strID, strTitle, strArtist, lstTbl):  *"""Add CD to list of dicts*    *Args:*  *strID (string): ID of the CD*  *strTitle (string): Title of CD*  *strArtist (string): Artist of CD*  *Returns*  *-------*  *lstTbl.*  *"""*  intID = int(strID)  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}  lstTbl.append(dicRow)  **return** lstTbl    @staticmethod  **def** delete\_item(lstTbl):  *"""Delete CD from list of dicts*    *Args:*  *lstTbl: 2D data structure (list of dicts) that holds the data during runtime*    *Returns*  *-------*  *lstTbl: 2D data structure (list of dicts) that holds the data during runtime*  *"""*  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!')  **return** lstTbl  **class** **FileProcessor**:  *"""Processing the data to and from text 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.*  *"""*    *# Check if file exists*    **try**:  table.clear() *# this clears existing data and allows to load data from file*  objFile = open(file\_name, 'rb')  table = pickle.load(objFile)  **return** table  **except** FileNotFoundError **as** e:  **print**('Text file does not exist!')  **print**('Build in error info:')  **print**(type(e), e, e.\_\_doc\_\_, sep='**\n**')  *# If no data, still return empty table, otherwise throws exception for None*  table = []  **return** table      @staticmethod  **def** write\_file(file\_name, table):  *# TODone Add code here*  *"""Function to save data to file*    *Args:*  *file\_name (string): name of file used to write the data to*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*    *Returns:*  *None*  *"""*  objFile = open(strFileName, 'wb')  pickle.dump(table, 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**')  *# Should not have TypeError but just in case*  **try**:  **for** row **in** table:  **print**('{}**\t**{} (by:{})'.format(\*row.values()))  **print**('======================================')  **except** **TypeError** **as** e:  **print**('Nothing in inventory!')  **print**('Build in error info:')  **print**(type(e), e, e.\_\_doc\_\_, sep='**\n**')    *# TODone add I/O functions as needed*  @staticmethod  **def** ask\_user():  *"""Ask user to enter ID, Title, and Artist of CD*    *Args:*  *None*  *Returns*  *-------*  *strID (string): ID of the CD*  *strTitle (string): Title of CD*  *strArtist (string): Artist of CD*  *"""*  **try**:  strID = int(input('Enter an integer for ID: ').strip())  strTitle = input('What is the CD**\'**s title? ').strip()  strArtist = input('What is the Artist**\'**s name? ').strip()  **return** strID, strTitle, strArtist  **except** **ValueError** **as** e:  **print**('That is not an integer! <<< Customer Message')  **print**('Build in error info:')  **print**(type(e), e, e.\_\_doc\_\_, sep='**\n**')  **print**()    *# 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...')  lstTbl = 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*  *# Check if strings are empty from ask\_user() function*  **try**:  strID, strTitle, strArtist = IO.ask\_user()  lstTbl = DataProcessor.add\_item(strID, strTitle, strArtist, lstTbl)  IO.show\_inventory(lstTbl)  **except** **TypeError** **as** e:  **print**('Nothing to add.')  **print**('Build in error info:')  **print**(type(e), e, e.\_\_doc\_\_, sep='**\n**')  **print**()  **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*  **try**:  intIDDel = int(input('Enter an integer for the ID you would like to delete: ').strip())  *# 3.5.2 search thru table and delete CD*  lstTbl = DataProcessor.delete\_item(lstTbl)  IO.show\_inventory(lstTbl)  **except** **ValueError** **as** e:  **print**('That is not an integer! <<< Customer Message')  **print**('Build in error info:')  **print**(type(e), e, e.\_\_doc\_\_, sep='**\n**')  **print**()  **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*  *# TODone move file processing code into function*  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 save:*  **else**:  **print**('General Error') |