<Songli Zhu>

<2022-Mar-13>

<Foundations of Programming (Python)>

<Assignment 07>

# Introduction

The assignment required me to continue modify a CD inventory program by handing possible errors, including file not exist when you try to open it, ID numbers are not integer, also you have to save the file into binary file (.dat) not text file (.txt). The goal here to modify the code, add more code to finish the assignment.

First thing, add try and except function when you try to open the file. Then, of course, taking care of integer of ID issues (when you add CD information or delete CD information). I also noticed that the CD numbers might be duplicate when you typed in the same CD\_ID (with or without same CD title and artist). However, I haven’t dig into it and handle it with error when you tried to add or delete it. I don’t know if this is a big issue.

The next thing is to save and read the .dat file by importing pickle and use dump and load function. I don’t want to change the number of variables in read\_file and write\_file. So, I just leave them although they are not used in the function. Shortcut to modify the codes, I think.

Lastly, run the script, check the dat file in a text editor. Cannot tell if it is correct in text editor. But it works when you load the file and read it in both terminal and Spyder.

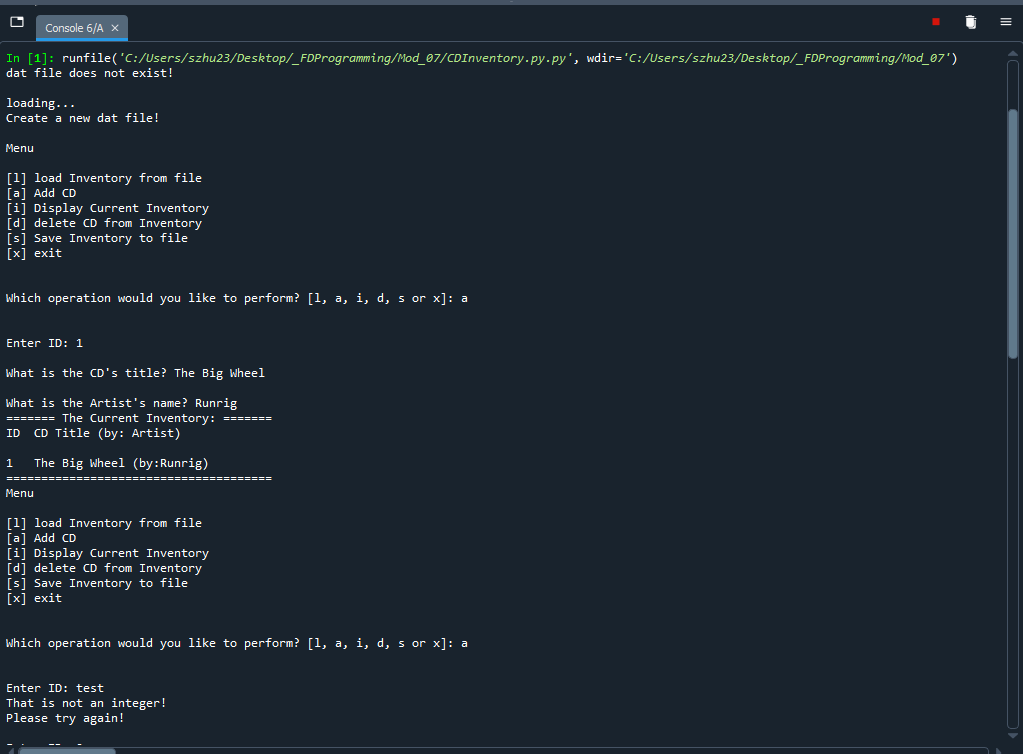


Figure Screen capture of input from CDInventory script in Spyder (Note: handling error when dat file is not exist and ID input is not integer).

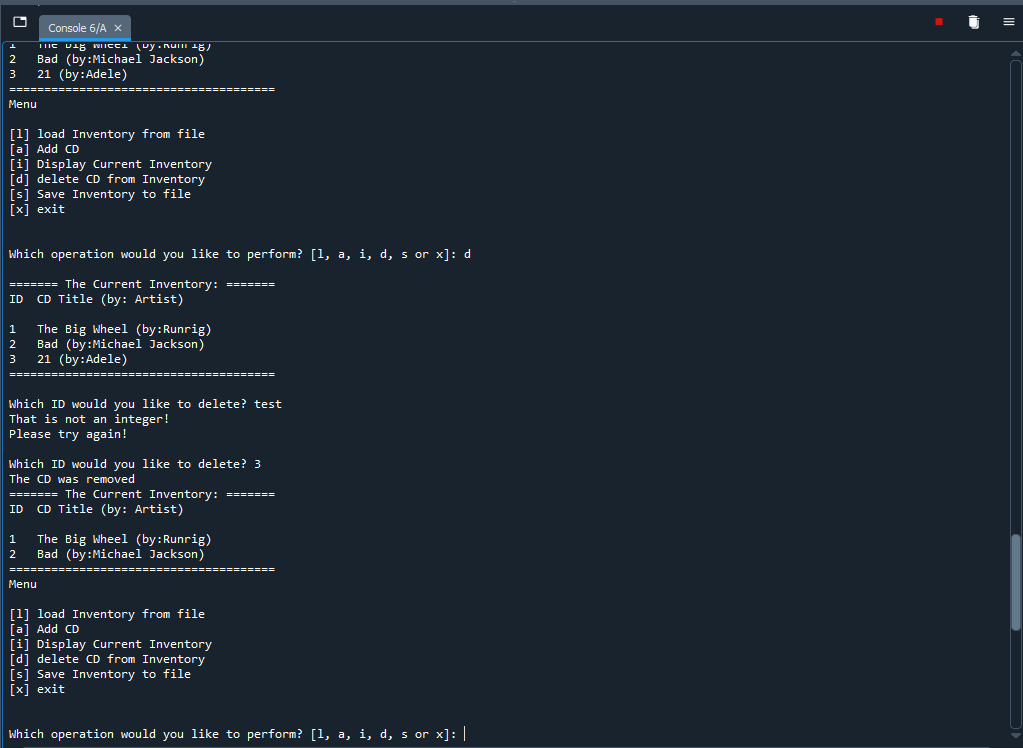


Figure 2 Screen capture of deletion from CDInventory script in Spyder (Note: handling error when ID input deletion is not integer).

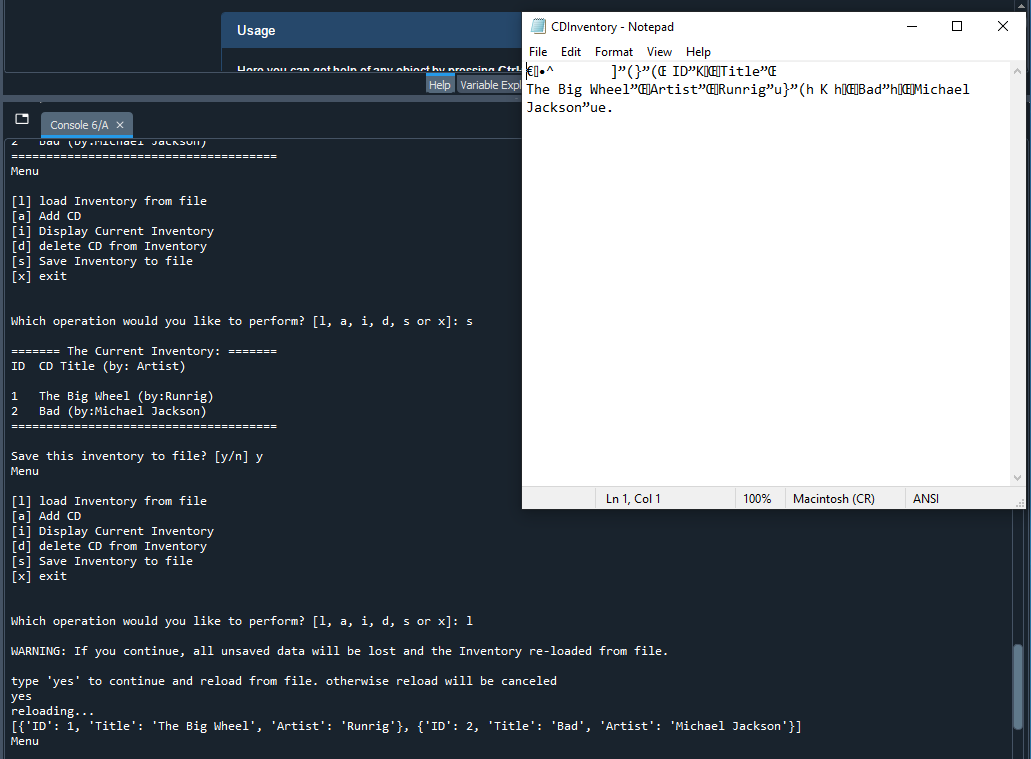


Figure Screen capture of load and save function of CDInventory scripts in Spyder (Note: CDInventory.dat file also included).

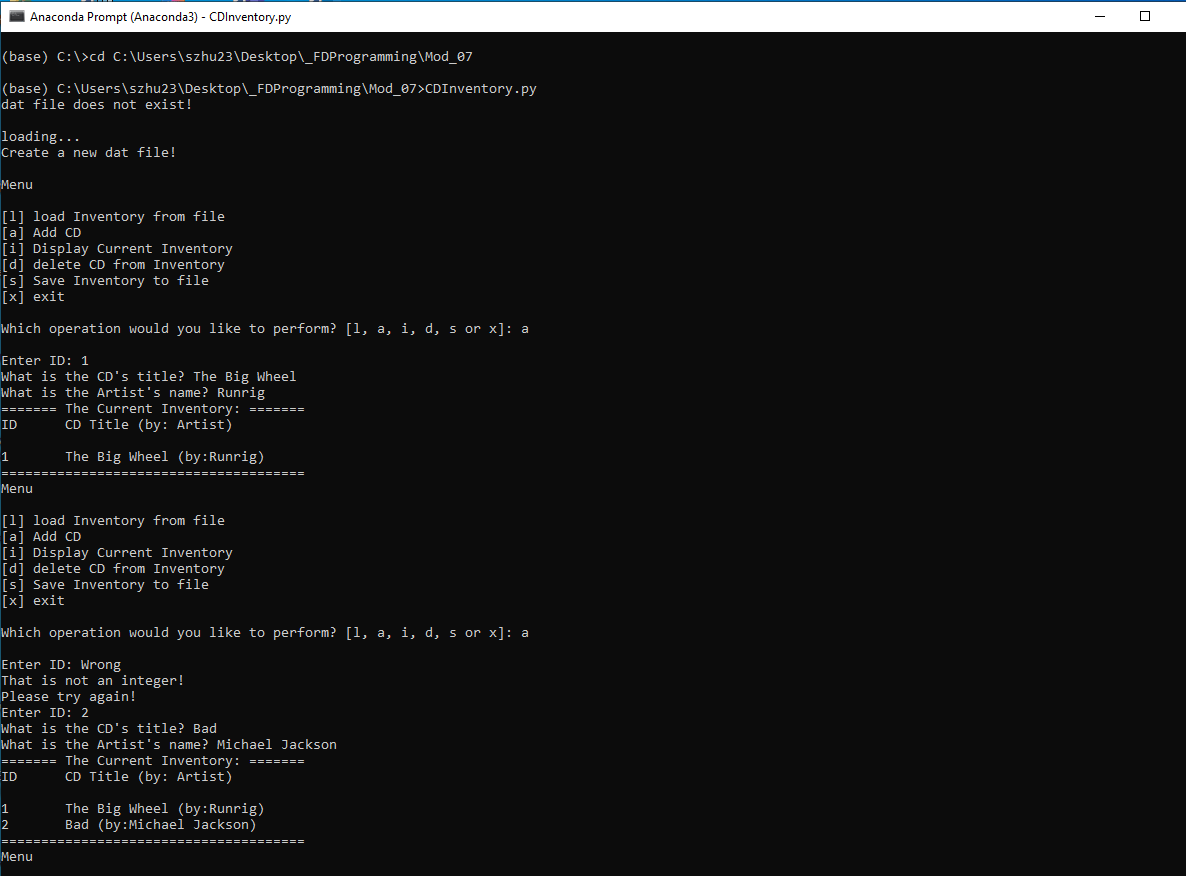


Figure Screen capture of input from CDInventory script in Terminal (Note: handling error when dat file is not exist and ID input is not integer).

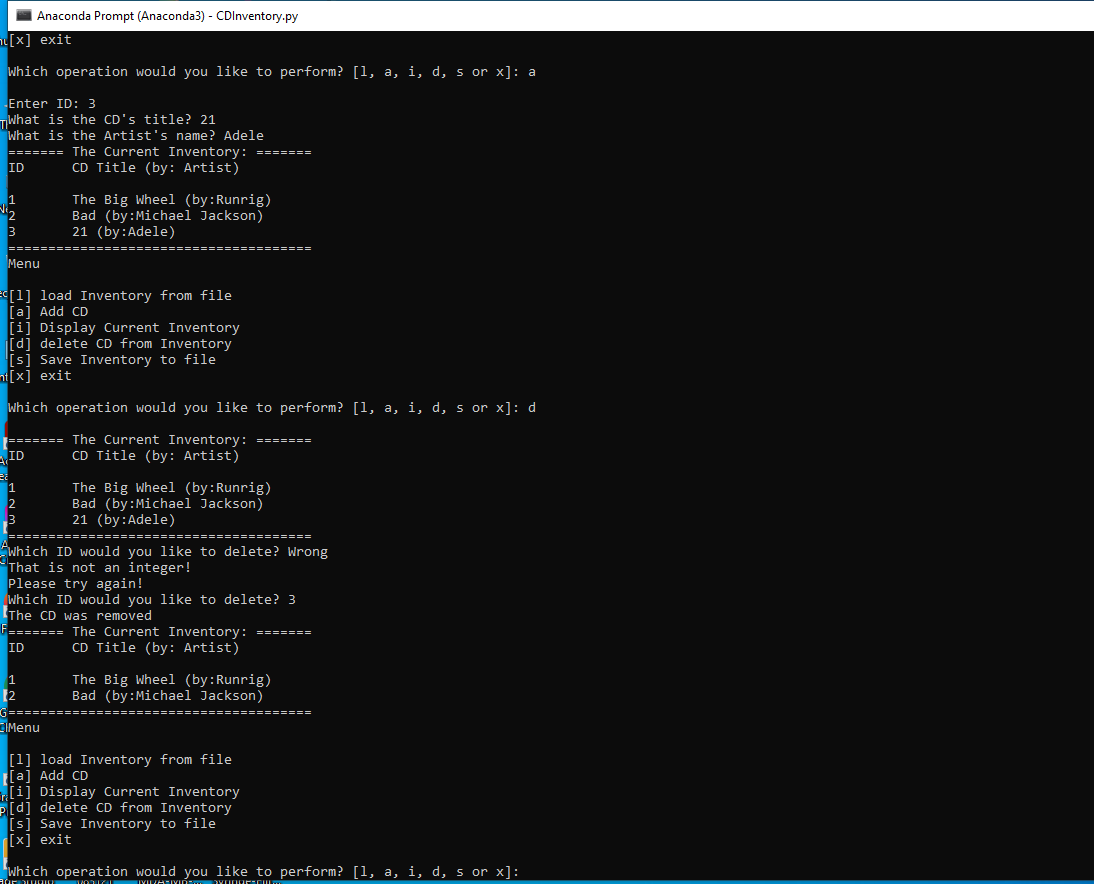


Figure Screen capture of deletion from CDInventory script in Terminal (Note: handling error when ID input deletion is not integer).

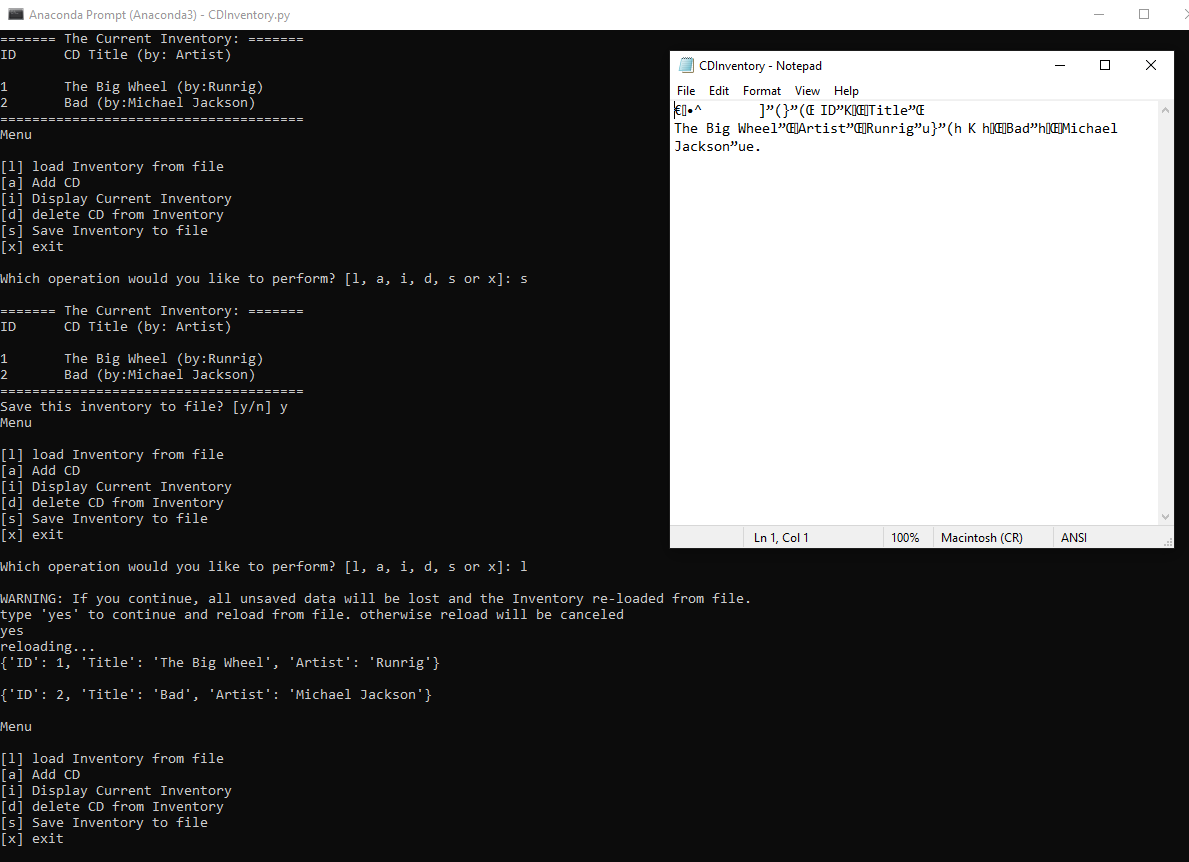


Figure Screen capture of load and save function of CDInventory scripts in Terminal (Note: CDInventory.dat file also included).

# Summary

From module 7, I learnt three modes when working with text files. You can write, read or append by using ‘w’, ‘r’, and ‘a’ command. To read data, you can also use readline() or readlines(). When you are using readline() function, you have to use while or for loop when you have multiple lines in your text file. I found for loop better because you don’t need to worry about how many lines you have in your text file. If you are using while loop, you should have a counter to count all lines which is unknown for sometimes. Also, you can use with…as option instead which seems odd to me at the beginning. Will get used to it for future practice.

Instead of text file, I also learnt about binary files (.dat). Working with binary files are similar to text files. You should use ‘wb’, ‘rb’, and ‘ab’ instead. Also, you should import pickle and use dump and load function instead of append function.

The next thing is Errors. You have to be careful about some commands because they are raising errors easily if input or calculation are invalid. For example, you cannot divide by 0, transfer a string to integer by int() function or open a file which doesn’t not exist. They will raise different errors, ZeroDivisionError, ValueError, FileNotFoundError… Besides that, you can also raise custom errors when you set certain conditions. Use try except is helpful.

It mentions more about class and \_\_init\_\_, \_\_str\_\_, \_\_repr\_\_(self) a little when talking about Error class. If you have no idea about class, it is hard to understand.

Appendix

GitHub link: <https://github.com/synbiomotif/Assignment_07>

## List function assignment06.py

Using [Saravjishut](https://saravjishut.org/syntax) (external reference) [[1]](#footnote-1)web page

|  |  |
| --- | --- |
| 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  287  288 | *#------------------------------------------#*  *# Title: CDInventory.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# Songli Zhu, 2022-Mar-05, Modify the script by calling functions*  *# Songli Zhu, 2022-Mar-13, Modify the script to handle structured errors and use binary data to store*  *#------------------------------------------#*  **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 from user input"""*  @staticmethod  **def** cd\_addition():  *"""Function to add data from user input to a 2D table (list of dictionaries)*    *Reads the data from user input and store the data into dictionaries*  *save the dictionaries into a list.*  *Args:*  *None*  *Returns:*  *None.*  *"""*  intID, strTitle, stArtist = IO.cd\_info()  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}  lstTbl.append(dicRow)  IO.show\_inventory(lstTbl)    @staticmethod  **def** cd\_deletion(intIDDel):  *"""Function to delete data from user input to a 2D table (list of dictionaries)*    *Reads the data from user input and search the data from a 2D table (list of dicts)*  *Try matching the user input to the value of each dictionary in a 2D table*  *if found, delete the dictionary from the 2D table*  *otherwise, return error message*  *Args:*  *intIDDel (int): deletion ID of the CD used to find its match from*  *from a 2D table (list of dictionaries) that holds the data*  *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!')  IO.show\_inventory(lstTbl)    @staticmethod  **def** inventory\_save():  *"""Function to save data from a 2D table (list of dicts) to file,*  *Reads the data from user input and store the data into dictionaries*  *save the dictionaries into a list.*  *Args:*  *None.*    *Returns:*  *None.*  *"""*  FileProcessor.write\_file(strFileName,lstTbl)  **class** **FileProcessor**:  *"""Processing the data to and from dat 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:*  *data (list): 2D data structure (list of dicts) that holds the data during runtime.*  *"""*  **with** open(file\_name,'rb') **as** objFile:  data = pickle.load(objFile)  **return** data  @staticmethod  **def** write\_file(file\_name, table):  *"""Function to write data from a 2D table (list of dicts) to file,*  *Reads the data from a 2D table (list of dicts) into a file*  *one dictionary row in table represents one line in the file.*  *Args:*  *file\_name (string): name of file used to save the data from*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*    *Returns:*  *None.*  *"""*  *# TODONE Add code here*  **with** open(file\_name, 'wb') **as** objFile:  pickle.dump(table, objFile)  *# -- 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('======================================')  *# TODONE add I/O functions as needed*    @staticmethod  **def** cd\_info():  *"""Gets user input for cd information*  *Args:*  *None.*  *Returns:*  *intID (int): an interger of the user input for the id of a cd*  *strTitle (string): a string of the user input for the title of a cd*  *stArtist (string): a string of the user input for the artist of a cd*  *"""*  flag = 1  **while** flag:  strID = input('Enter ID: ').strip()  **try**:  intID = int(strID)  flag = 0  **except** **ValueError**:  print('That is not an integer!')  print('Please try again!')  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*  **try**:  data = FileProcessor.read\_file(strFileName, lstTbl)  **for** row **in** data:  print(row)  **except** **FileNotFoundError**:  print('dat file does not exist!**\n**')  print('loading...')  print('Create a new dat file!**\n**')  open(strFileName,'x')    *# 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**\n**') *# add extra \n*  **if** strYesNo.lower() == 'yes':  print('reloading...')  data = FileProcessor.read\_file(strFileName, lstTbl)  **for** row **in** data:  print(row)  print()  **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*  *# TODONE move IO code into function*  *# 3.3.2 Add item to the table*  *# TODONE move processing code into function*  DataProcessor.cd\_addition()  **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*  flag = 1  **while** flag:  **try**:  IDDel = input('Which ID would you like to delete? ').strip()  intIDDel = int(IDDel)  flag = 0  **except** **ValueError**:  print('That is not an integer!')  print('Please try again!')  *# 3.5.2 search thru table and delete CD*  *# TODONE move processing code into function*  DataProcessor.cd\_deletion(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':  *# 3.6.2.1 save data*  *# TODONE move processing code into function*  DataProcessor.inventory\_save()  **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') |

1. Retrieved 2022-Feb-26 [↑](#footnote-ref-1)