Name: Fabio Sacca

Date: Nov-20, 2022

Course: Foundations of Programming (Python)

Title: Assignment 06

# Assignment 06

## Introduction:

In this document, I will describe the steps I took to modify the ‘CD Inventory’ program from a starter script in order to organize the code, as needed. I will then show the script running in Spyder and Terminal Mac window.

## Topic 1: Modifying a Python script

I opened the 'CDInventory\_Starter.py' starter script and worked through the various sections of the program focusing on one requirement at a time.

### 1.1. Adding functions for data processing

I created a class named ‘DataProcessor’ to group the functions that process data held in memory during runtime, and added two main functions within it:

* add\_cd(data, table): this is a function that manages data ingestion from user input to a list of dictionaries
* del\_cd(ID, table): this is a function that manages data removal from a list of dictionaries based on user input

**class** **DataProcessor**:

*# TODone add functions for processing here*

*"""Processing the data held in memory during runtime"""*

@staticmethod

**def** add\_cd(data, table):

*"""Function to manage data ingestion from user input to a list of dictionaries*

*Adds data from user entry into a 2D table (list of dicts) in memory during runtime.*

*Args:*

*data (list): values entered by user for ID, CD Title, Artist Name*

*table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*

*Returns:*

*Confirmation message the CD was added to inventory.*

*"""*

strID = input('Enter ID: ').strip()

strTitle = input('What is the CD**\'**s title? ').strip()

stArtist = input('What is the Artist**\'**s name? ').strip()

intID = int(strID)

dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}

lstTbl.append(dicRow)

print('The CD was added to Inventory')

@staticmethod

**def** del\_cd(ID, table):

*"""Function to manage data removal from list of dictionaries based on user input*

*Remove a row from a 2D table (list of dicts) in memory during runtime.*

*Args:*

*ID: ID to be deleted*

*table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*

*Returns:*

*Confirmation message the CD was removed or not found.*

*"""*

intIDDel = int(input('Which ID would you like to delete? ').strip())

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!')

### 1.2. Adding function for writing data on file

Within the class ‘FileProcessor’ that groups functions that process data to and from text file, I added one function:

* write\_file(file\_name, table): this is a function that manages data storage from a list of dictionaries to a file

**class** **FileProcessor**:

*"""Processing the data to and from text file"""*

@staticmethod

**def** write\_file(file\_name, table):

*# ToDONE Add code here*

*"""Function to manage data storage from a list of dictionaries to a file*

*Saves the data to file identified by file\_name from a 2D table*

*(list of lists).*

*Args:*

*file\_name (string): name of file used to write the data from*

*table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*

*Returns:*

*None.*

*"""*

objFile = open(strFileName, 'w')

**for** row **in** lstTbl:

lstValues = list(row.values())

lstValues[0] = str(lstValues[0])

objFile.write(','.join(lstValues) + '**\n**')

objFile.close()

### 1.3. Adding I/O functions as needed

Finally, I added the functions to display input/output in the program UI based on the program design

* To load data from file: FileProcessor.read\_file(strFileName, lstTbl)
* To write data to file: FileProcessor.write\_file(strFileName, lstTbl)
* To add CD to table: DataProcessor.add\_cd(dicRow, lstTbl)
* To delete CD from table: DataProcessor.del\_cd(dicRow, lstTbl)
* To display inventory: IO.show\_inventory(lstTbl)
* To print menu: IO.print\_menu()

## Topic 2: Running my script

I saved the script as 'CDInventory.py' in a folder called 'Assignment06' within the course folder '\_FDProgramming', and ran the script in Spyder. The image below shows that the script works in my computer.

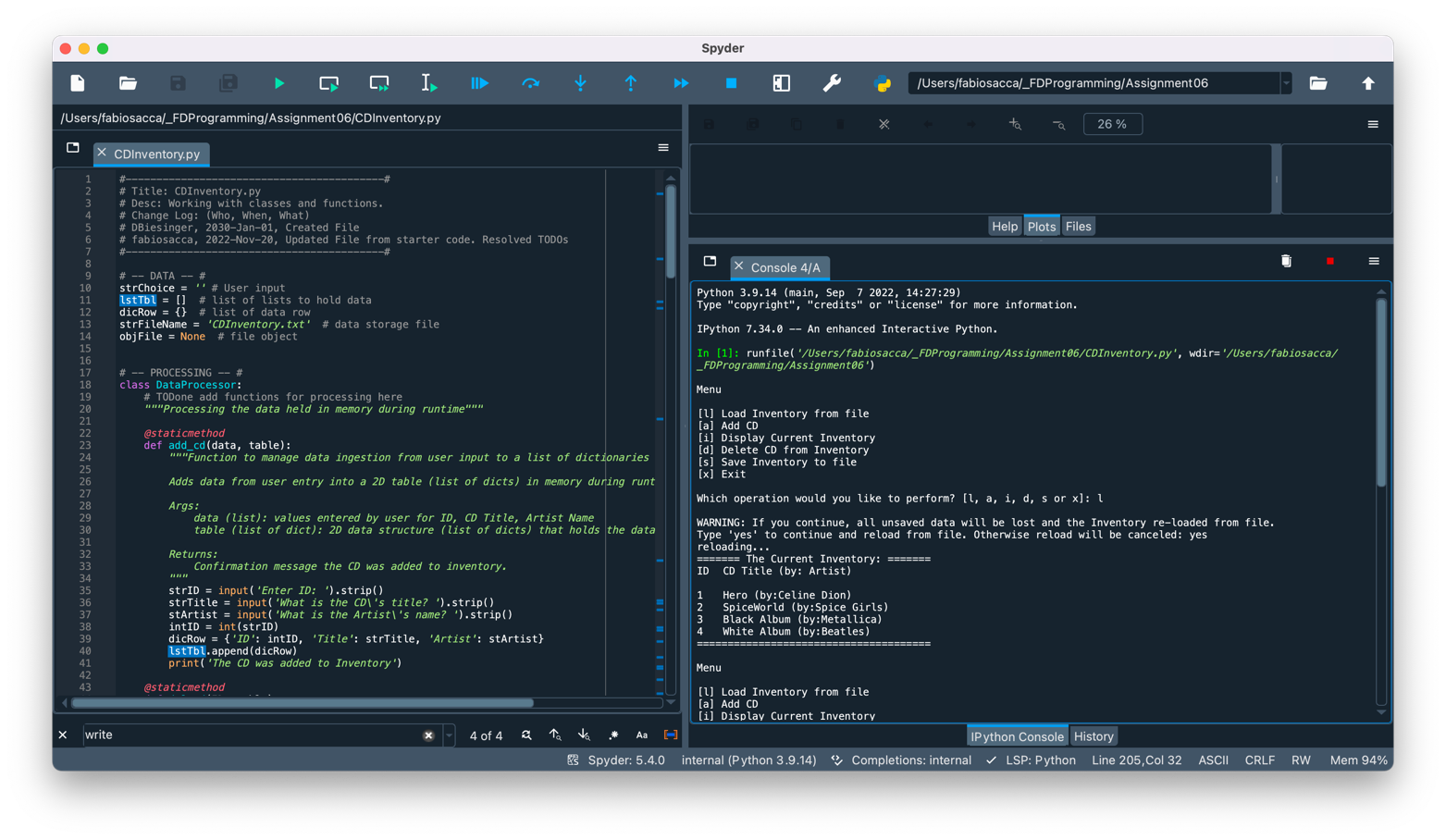


Figure 1 - Screen capture of Spyder returning CDInventory.py script (Load Data)

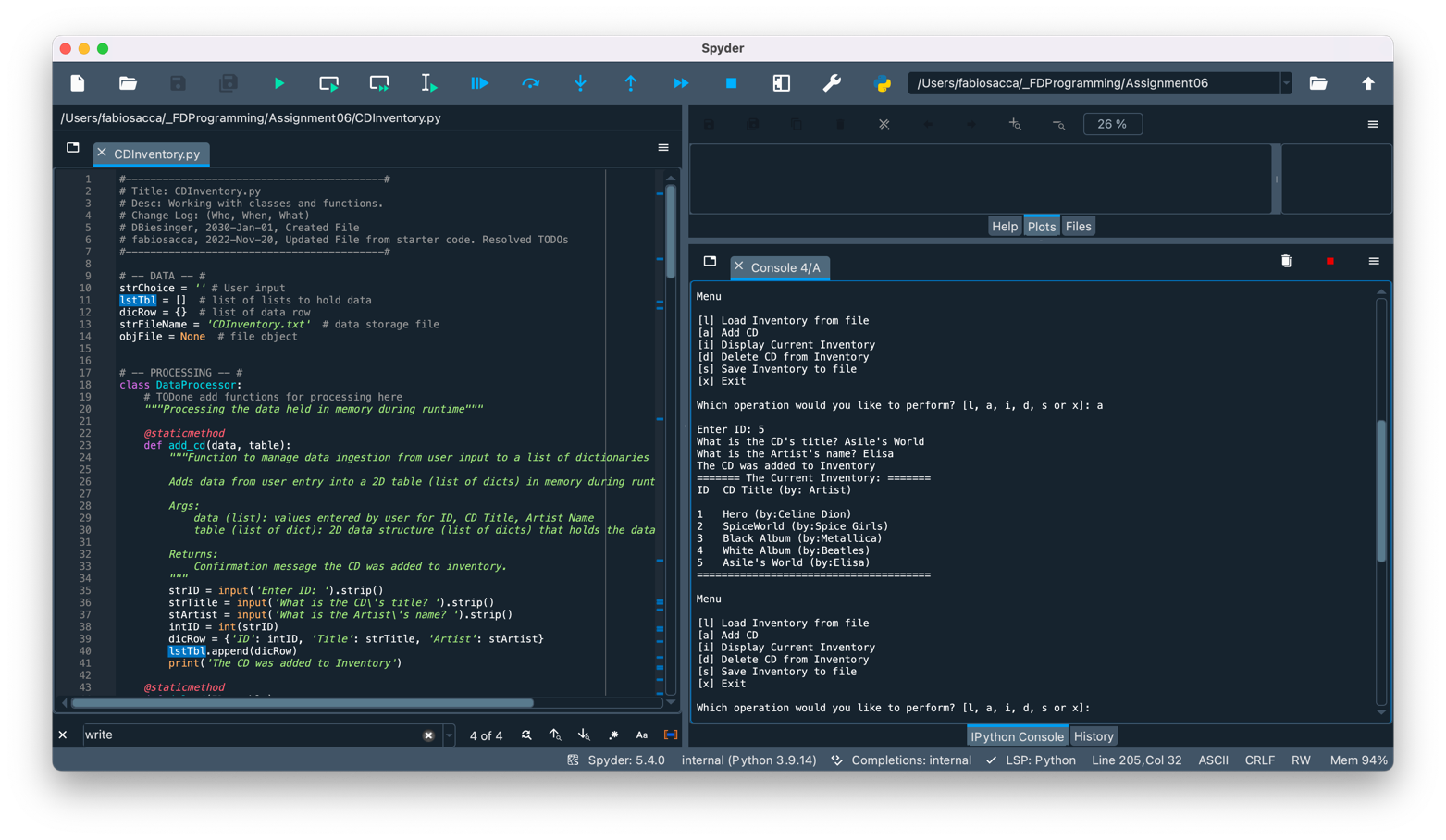


Figure 2 - Screen capture of Spyder returning CDInventory.py script (Add CD)

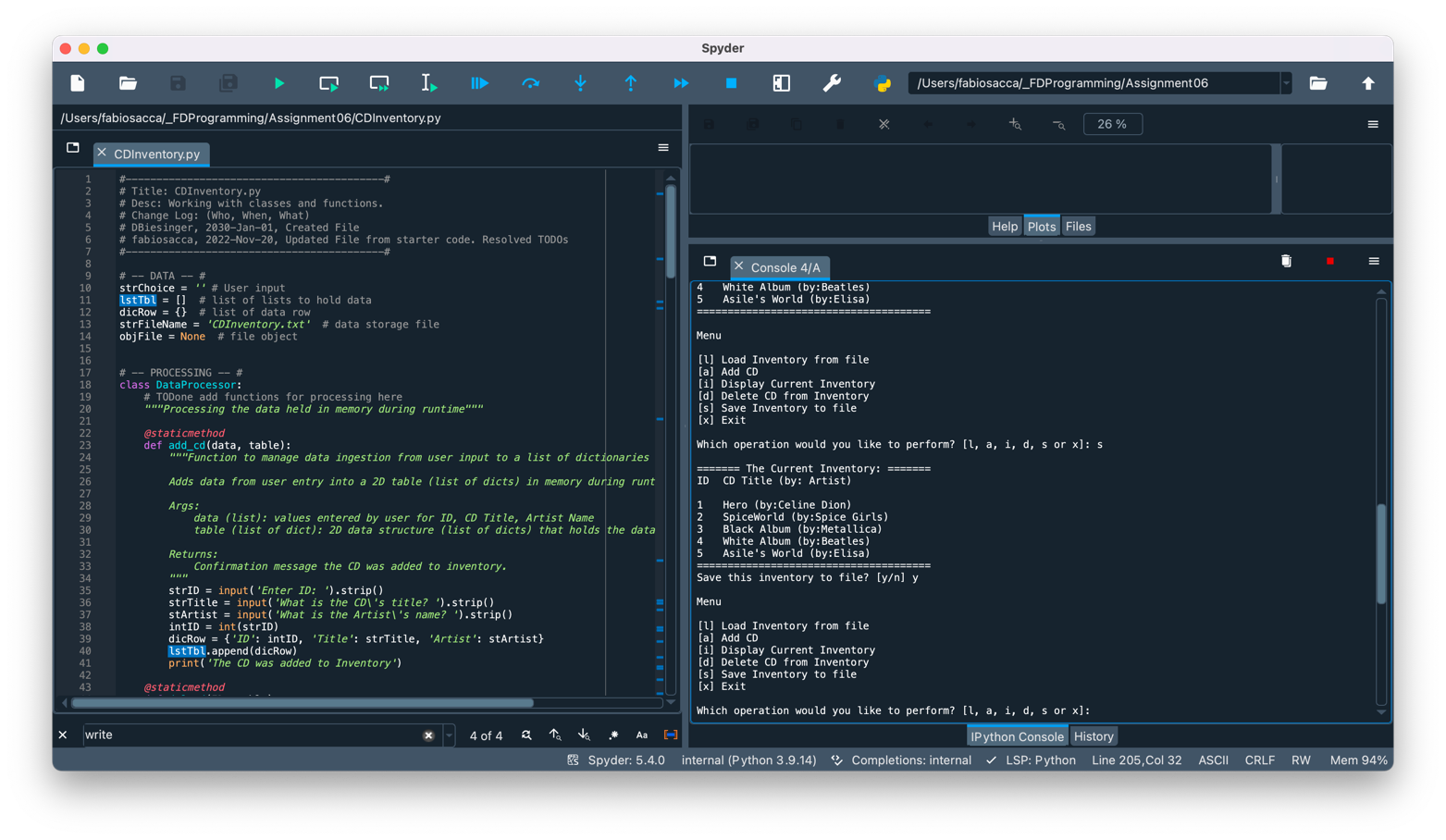


Figure 3 - Screen capture of Spyder returning CDInventory.py script (Save File)

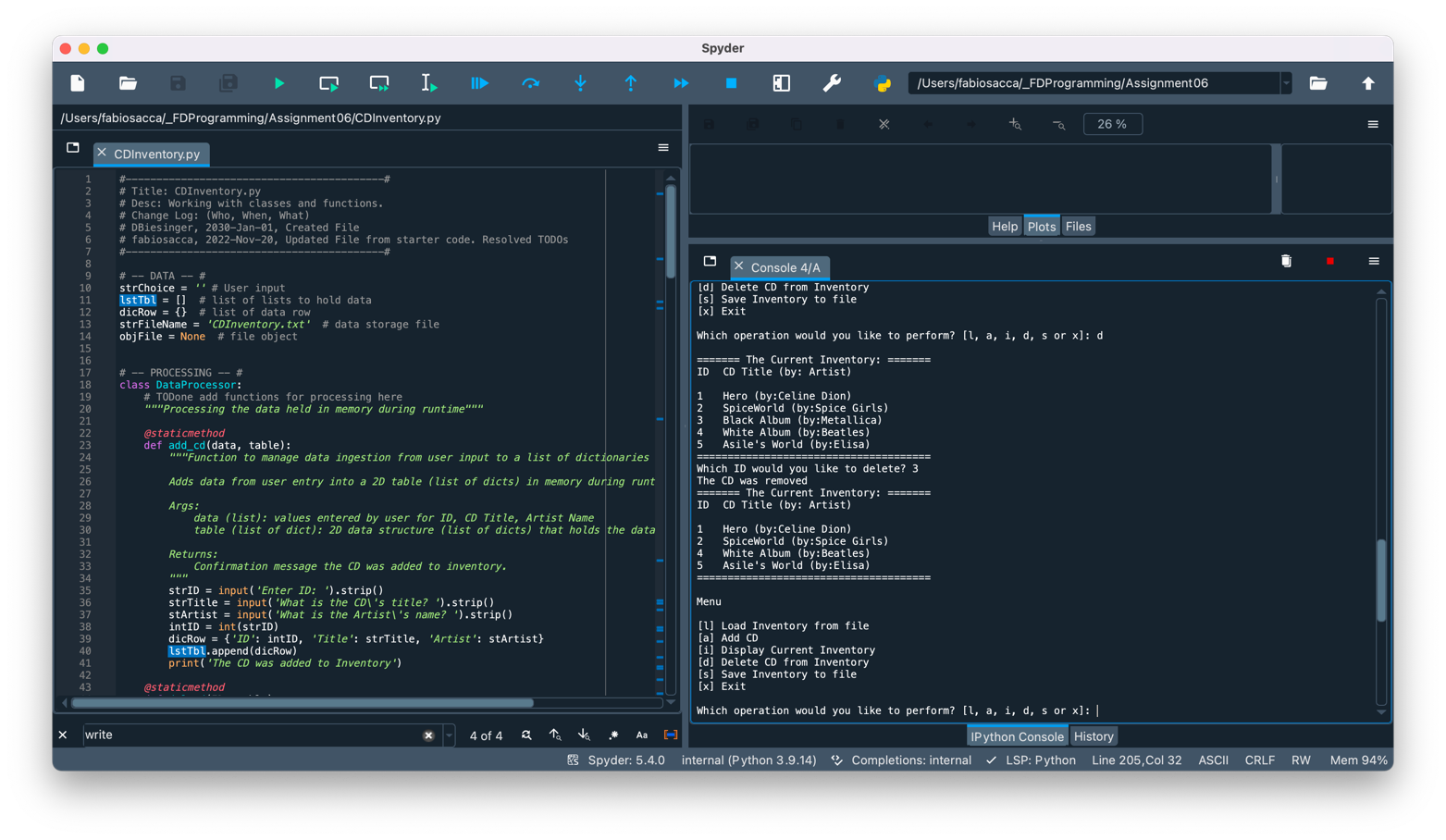


Figure 4 - Screen capture of Spyder returning CDInventory.py script (Delete CD)

I repeated this step in a terminal window. The image below shows that the script works in my computer.

Text

Description automatically generated with medium confidence

Figure 5 - Screen capture of OS Terminal returning CDInventory.py script

## Text Description automatically generated

Figure 6 - Screen capture of OS Terminal returning CDInventory.py script

## Topic 3: Verifying that my script worked

I located the file ‘CDInventory.txt’ and opened it in a text editor. The image below shows that the data I entered has been written to the file in the correct format. Text

Description automatically generated with medium confidence

Figure 7 - Screen capture of CDInventory.txt displaying data entered.

## Topic 4: Submitting my work on GitHub

Both the CDInventory.py and the present knowledge document are posted on the following public repository

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

## Summary

In this assignment, I covered the steps needed to:

* Modify ‘CD Inventory’ program from a starter script adding classes and functions to process data in memory and data to and from a text file, and calling those functions as needed throughout the script.
* Run the script in Spyder and a terminal window.
* Verify that the script in fact wrote in CDInventory.txt the data I entered in the correct format.
* Submit my work on a public GitHub repository.

# Appendix A

## Code of CDInventory.py script presented with [**Syntax Highlighter**](https://saravjishut.org/syntax)**[[1]](#footnote-1)**

|  |  |
| --- | --- |
| 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 | *#------------------------------------------#*  *# Title: CDInventory.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# fabiosacca, 2022-Nov-20, Updated File from starter code. Resolved TODOs*  *#------------------------------------------#*  *# -- 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*  *# -- PROCESSING -- #*  **class** **DataProcessor**:  *# TODone add functions for processing here*  *"""Processing the data held in memory during runtime"""*    @staticmethod  **def** add\_cd(data, table):  *"""Function to manage data ingestion from user input to a list of dictionaries*  *Adds data from user entry into a 2D table (list of dicts) in memory during runtime.*  *Args:*  *data (list): values entered by user for ID, CD Title, Artist Name*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *Returns:*  *Confirmation message the CD was added to inventory.*  *"""*  strID = input('Enter ID: ').strip()  strTitle = input('What is the CD**\'**s title? ').strip()  stArtist = input('What is the Artist**\'**s name? ').strip()  intID = int(strID)  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}  lstTbl.append(dicRow)  print('The CD was added to Inventory')    @staticmethod  **def** del\_cd(ID, table):  *"""Function to manage data removal from list of dictionaries based on user input*  *Remove a row from a 2D table (list of dicts) in memory during runtime.*  *Args:*  *ID: ID to be deleted*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *Returns:*  *Confirmation message the CD was removed or not found.*  *"""*  intIDDel = int(input('Which ID would you like to delete? ').strip())  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!')  **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.*  *"""*  table.clear() *# this clears existing data and allows to load data from file*  objFile = open(file\_name, 'r')  **for** line **in** objFile:  data = line.strip().split(',')  dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}  table.append(dicRow)  objFile.close()  @staticmethod  **def** write\_file(file\_name, table):  *# ToDONE Add code here*  *"""Function to manage data storage from a list of dictionaries to a file*  *Saves the data to file identified by file\_name from a 2D table*  *(list of lists).*  *Args:*  *file\_name (string): name of file used to write the data from*  *table (list of dict): 2D data structure (list of dicts) that holds the data during runtime*  *Returns:*  *None.*  *"""*  objFile = open(strFileName, 'w')  **for** row **in** lstTbl:  lstValues = list(row.values())  lstValues[0] = str(lstValues[0])  objFile.write(','.join(lstValues) + '**\n**')  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('**\n**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*  *# 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':  *# 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.add\_cd(dicRow, lstTbl)  IO.show\_inventory(lstTbl)  **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*  *# 3.5.2 search thru table and delete CD*  *# TODone move processing code into function*  DataProcessor.del\_cd(dicRow, lstTbl)  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*  *# TODone move 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') |

1. Last viewed: Nov-20, 2022 (9:20pm PT) [↑](#footnote-ref-1)