<Songli Zhu>

<2022-Mar-05>

<Foundations of Programming (Python)>

<Assignment 06>

# Introduction

The assignment required me to continue modify a CD inventory program by using class and function. From last Module, we already have some codes. The goal here to modify the code, move code, and try to call function from class to finish the assignment.

First thing, I found some TODOs in the code and try to answer each of them one by one. Move some codes from the main body of the script to each class. Since we already have three classes: DataProcessor, FileProcessor, and IO. I have to try to find correct position for each function, such as addition, deletion, saving option.

I thought it is easier at first place, just need to move some codes and give a name for each function. Then I realized that I have to consider the arguments and returns if I want to call them correctly. Also, I need to add a new function (cd\_info) in class IO to make the script works.

I am confused about the save the inventory function. It appears that you have to move some codes (TODOs) into class DataProcessor to have a save function and I found a similar function under class FileProcessor (default write\_file function (TODOs)). I found they are exactly the same to me when you want to save inventory to file. To my understanding, save the inventory to file means write the data into file. What I did here is to call write\_file function when calling save\_inventory function. Seems redundant to me. Maybe I am wrong.

Overall, I found it is easier than previous assignments when I figure out the arguments and returns for some functions.

Lastly, run the script, check the text file in a text editor. Make sure the information is correct.

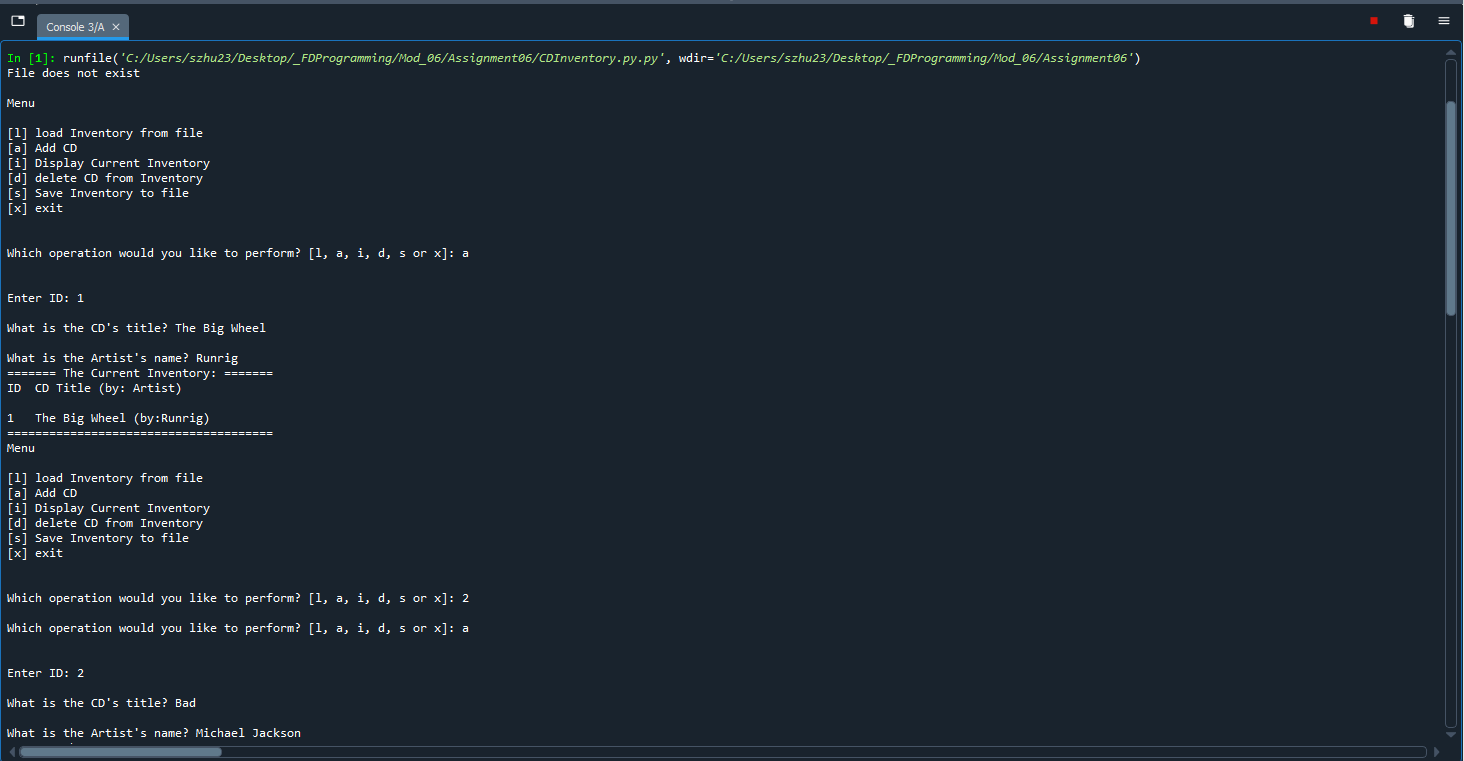


Figure 1 Screen capture of output from CDInventory script in Spyder (Note: input 3 CD information).

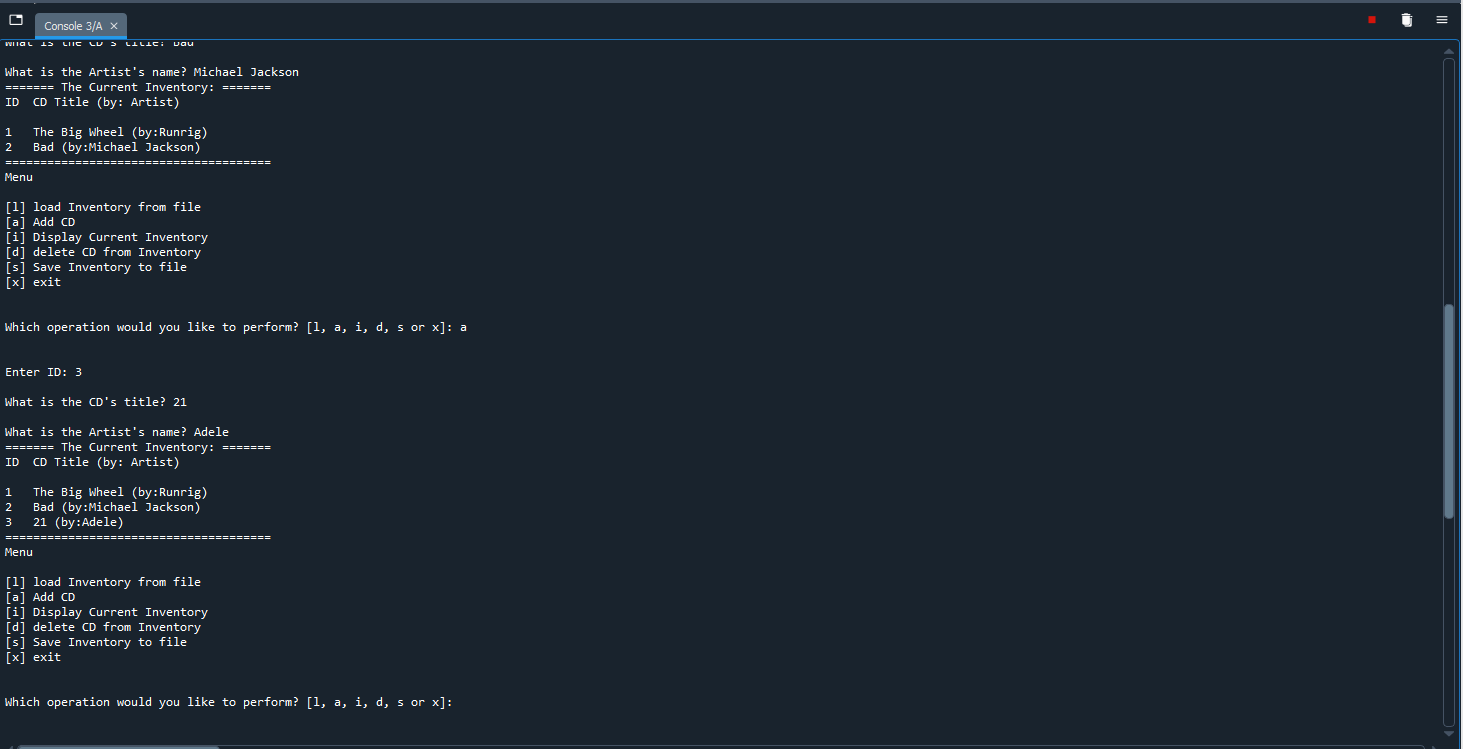
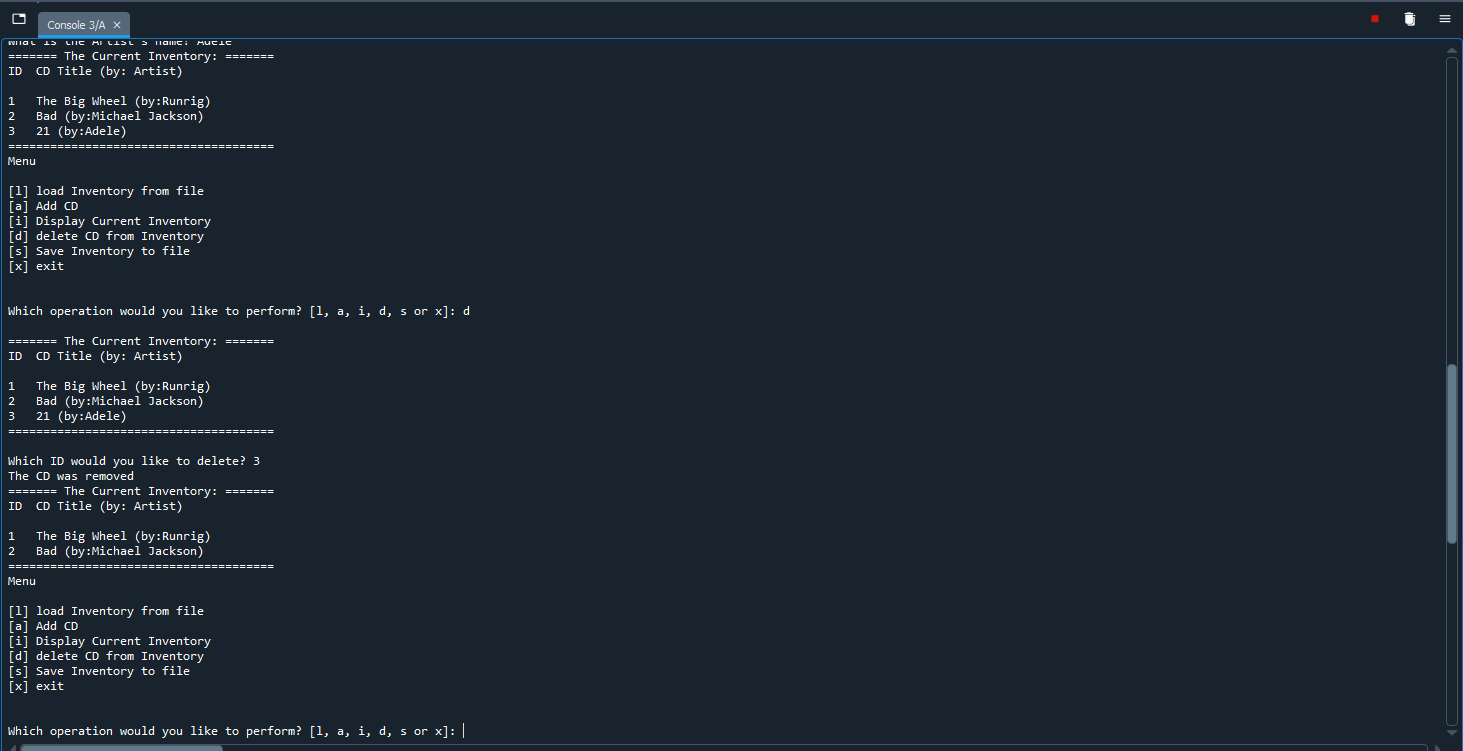


Figure Screen capture of output from CDInventory script in Spyder (Note: input 3 CD information)-Continued.

Figure 3 Screen capture of output from CDInventory script in Spyder (Note: delete one CD from inventory and display the remaining inventory).

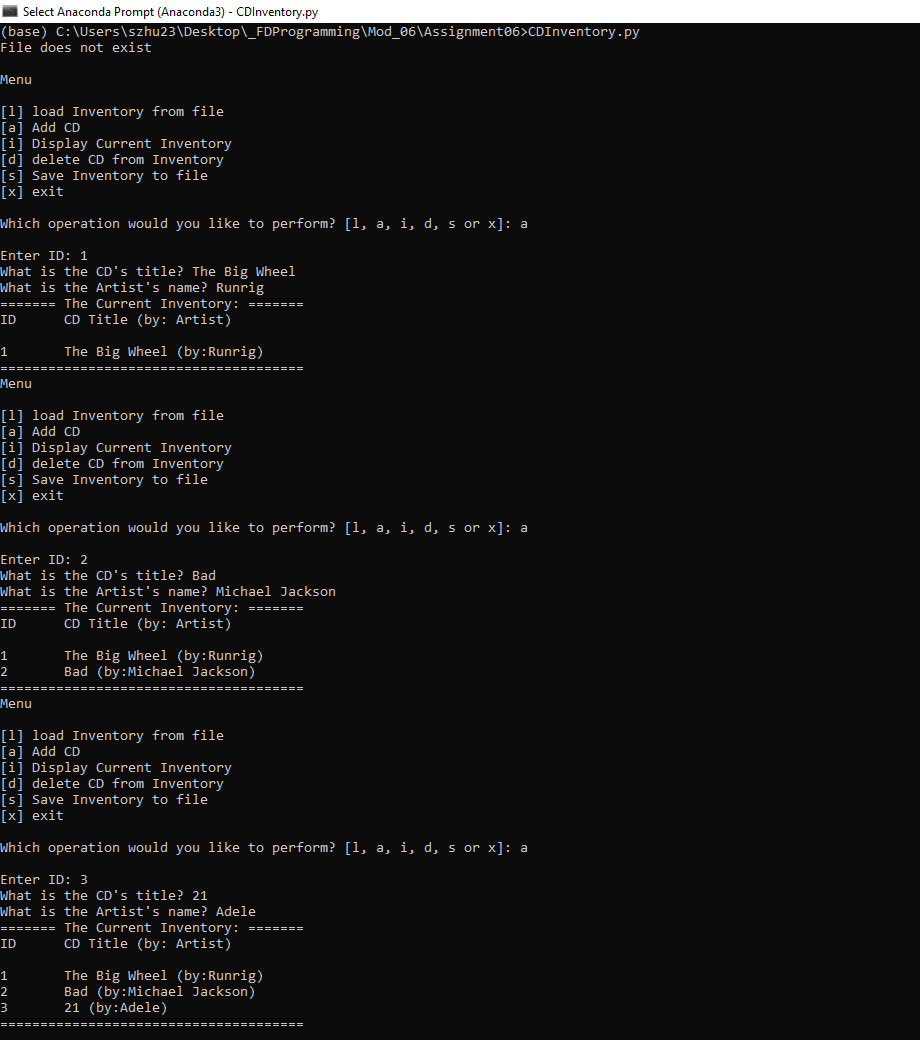


Figure 4 Screen capture of output from CDInventory script in Terminal (Note: input 3 CD information).

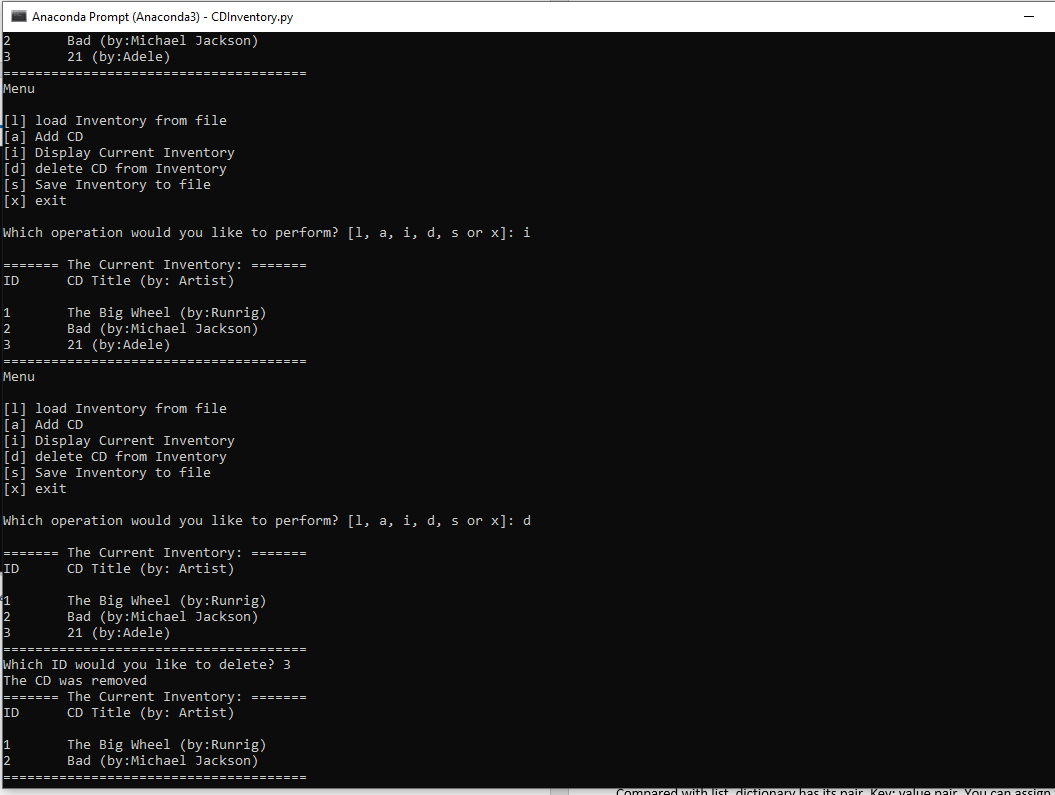


Figure 5Screen capture of output from CDInventory script in Terminal. (Note: delete one CD from inventory and display the remaining inventory).

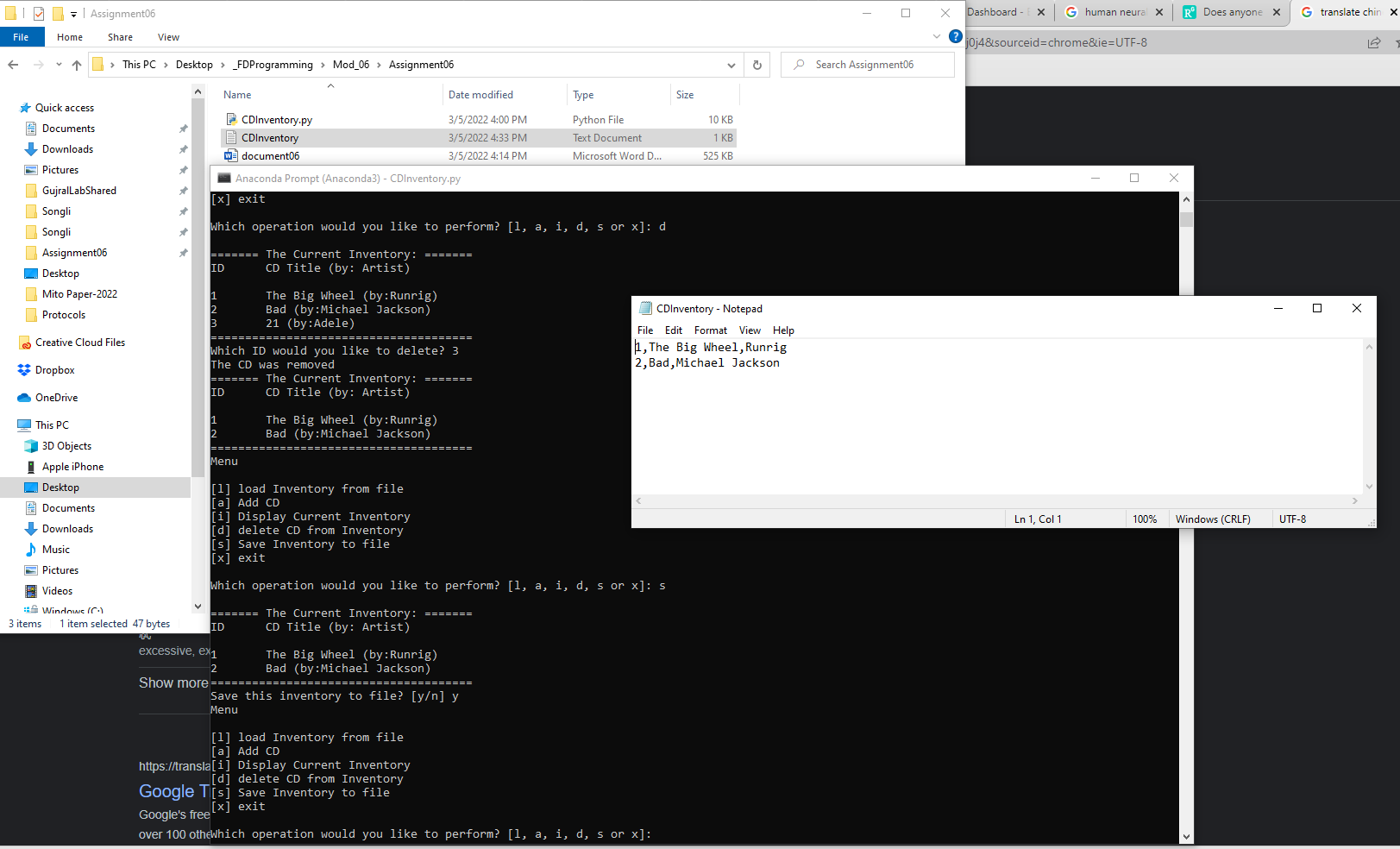


Figure 6 Screen capture of text file generated by CDInventory script in text editor

# Summary

From module 6, I learnt the function and some information of class. Function is important and save your time and effort when you repeat calling the same function. Be aware of arguments and return values when you tried to define a function. Especially when you call a function, the arguments are important to be noticed when you try to use positional and key arguments. The examples from the module 6 really help a lot when you want to mix positional and named arguments. You should have positional argument follows keyword argument. Also, you can set default values for some arguments for convenience. To return values, you could have multiple return values. However, when you have multiple return values and want to call this function, you should have the same number of variables to be assigned.

Next thing is the local and global variables, if the program cannot find the variable in its scope of the function, it will search the main part to find out. Also, you can also set a local variable with the same name of the variable in the main part of the codes. If you want to change the variable in the scope of the function, use global. However, the materials don’t recommend it.

Lastly, the docstrings, when you have a large amount of codes and multiple functions, it is better to have some explanatory headers at the beginning of a function or class. It helps you a lot to understand what the function does. I really did a lot of docstrings for the assignment.

At the end of the module, it shadows a little on class and just @staticmethod as the example (Also shown in the assignment). That’s what I learnt so far about class.

# Appendix

GitHub link: <https://github.com/synbiomotif/Assignment_06>

## 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 | *#------------------------------------------#*  *# 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*  *#------------------------------------------#*  *# -- 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 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 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):  *"""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*  objFile = open(strFileName, 'w')  **for** row **in** table:  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('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*  *"""*  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)  **return** intID, strTitle, stArtist  *# 1. When program starts, read in the currently saved Inventory*  **try**:  FileProcessor.read\_file(strFileName, lstTbl)  **except** **FileNotFoundError**:  print('File does not exist**\n**')    *# 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...')  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.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*  intIDDel = int(input('Which ID would you like to delete? ').strip())  *# 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)