Anna Willie

November 21, 2021

IT FDN 110

Assignment 06

CD Inventory with Functions

# Introduction

The goal for this week’s assignment was to add functions to our menu loop. This was done by classifying each function and defining each function before the menu in the code flow. We also used arguments to carry over global variables.

# Topic 1

Creating the functions for delete and add was not too hard, just had to define them before the menu definition. I originally had some issues with the functions that Dirk created, because he didn’t add the file and list variable names. Once I changed those, my script ran fine. I decided to add an ID counting function, so there could not be a human error where we use the same ID twice or skip lines. I do, however, have some faults with this function, because when the delete function is used it will leave number gaps. I’m not quite sure how to fix that now, but I’m sure we can fix that later.

Here is a screenshots of my script working on Spyder:

Text

Description automatically generated

Figure 1 Load and Add Functions

Text

Description automatically generated

Figure 2 Save and Delete Functions

Here is my saved text file:

Graphical user interface, text, application

Description automatically generated

Figure 3 Text File

Here is my script running in a shell:

Graphical user interface, text, application, email

Description automatically generated

Figure 4 Script in Shell

Here is a link to my GitHub post.

# Summary

With this assignment we were able to see how streamlined code can be with the use of functions and classes of functions. I’m excited to see how this can be further utilized in future classes. To further help my study I am curious to see how I can improve my ID function by preventing the gaps that would occur during my delete functions. Can you help with this?

# Appendix

Generated from [Syntax Higlighter.](https://ajblk.github.io/SyntaxHighlightGenerator-v3.0/OnlineGenerator.html)[[1]](#footnote-1)

|  |  |
| --- | --- |
| 001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049  050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099  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 | #------------------------------------------#  # Title: Assignment06\_Starter.py  # Desc: Working with classes and functions.  # Change Log: (Who, When, What)  # DBiesinger, 2030-Jan-01, Created File  # AWillie, 2021-Nov-17, Added add function and delete function  # AWillie, 2021-Nov-18, Worked on save and load function  # AWillie, 2021-Nov-21, Added ID naming function  #------------------------------------------#    # -- 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  ID **=** 0    # -- PROCESSING -- #  **class** DataProcessor:      @staticmethod  **def** New\_ID():  **global** ID          ID **+=** 1      @staticmethod  **def** add\_Title(strTitle, strArtist):          """ Function to add additonal CD's to inventory            Uses the input as variable to add to a dictionary to add to the inventory            Args:              strTitle: Variable for the CD title              strArtist: Variable for the CD\'s Artist            Returns:              None.          """  **global** ID          DataProcessor.New\_ID()          dicRow **=** {'ID': ID, 'CD Title': strTitle, 'Artist': strArtist}          lstTbl.append(dicRow)          print('Your CD has been added')          print()      @staticmethod  **def** del\_Title(IDIntDel):          """Function to delete a chosen CD title            User input the desired ID they would like to delete            Args:              IDIntDel: The ID # that is chosen to be deleted            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!')  **class** FileProcessor:      """Processing the data to and from text file"""        @staticmethod  **def** read\_file(strFileName, lstTbl):          """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.          """  **global** ID          lstTbl.clear()  # this clears existing data and allows to load data from file          objFile **=** open(strFileName, 'r')  **for** line **in** objFile:              data **=** line.strip().split(',')              dicRow **=** {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}              lstTbl.append(dicRow)              ID **=** int(data[0])          objFile.close()        @staticmethod  **def** write\_file(strFileName, lstTbl):          # Added code here          """Finction to save data to ta text file            Takes the current memory and moves it to a text file            Args:              file\_name(string): name of file used to copy memory to              table (list of dict): 2D data structure holding the inventory          Returns:              None.          """  **if** strYesNo **==** 'y':              objFile **=** open(strFileName, 'w')  **for** row **in** lstTbl:                  lstValues **=** list(row.values())                  lstValues[0] **=** str(lstValues[0])                  objFile.write(','.join(lstValues) **+** '\n')              objFile.close()  **else**:              input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')      # -- 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\tCD Title (by: Artist)\n')  **for** row **in** table:              print('{}\t{} (by:{})'.format(**\***row.values()))          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...')              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          # moved IO code into function          strTitle **=** input('What is the CD\'s title? ').strip()          strArtist **=** input('What is the Artist\'s name? ').strip()          DataProcessor.add\_Title(strTitle, strArtist)          IO.show\_inventory(lstTbl)          # 3.3.2 Add item to the table          #moved processing code into function  **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          # moved processing code into function          DataProcessor.del\_Title(intIDDel)          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()          FileProcessor.write\_file(strFileName, lstTbl)  **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. Accessed November 21, 2021 [↑](#footnote-ref-1)