Kent Bloodsworth

March 06, 2022

Foundation of Programming (Python)

Assignment06

CD Inventory Continued

# Introduction

This module covered how to work with functions, their arguments and return values, and how we can use functions to organize our code. We also covered what is meant by shadowing and the difference between global and local variables. To complete this assignment, we used the starter code that was provided. The starter code was an example solution for last week's work. In this starter code script, there were several TODO's throughout the code, and we were tasked with completing these to modify the script to use functions and further organize the code.

# Completing the TODO’s

It was overwhelming when first opening the starter script and looking at the TODO’s. I started by completing the ones I thought were the most straightforward, and even the quote UN quote easy ones took work. There’s a lot to wrap my mind around just the concept of the functions and using arguments and return values and identifying what they should be and how they interact with the code. I had the most challenging time with the “a” option in the menu, adding a CD entry. I believe this was because of breaking up the input from the user into the input-output class and then moving the second part of that code to the data processor class. The information was captured in one function, but the data was processed and appended to the list table in another function. I got several error messages trying to get everything working right. I found that the information provided with the error message was usually enough to point me in the right direction for where I went wrong.

# Screenshot of working script

Figures 1, 2, and 3 are screenshots of the script working in Spyder, and figures 4, and 5 are screenshots of the script working in Terminal. The script ran fine on both consoles.

Text

Description automatically generated

Figure - Spyder Screenshot

Text

Description automatically generated

Figure - Spyder Screenshot

Text

Description automatically generated

Figure - Spyder Screenshot

Text

Description automatically generated

Figure - Screenshot of Terminal

Text

Description automatically generated

Figure - Screenshot of Terminal

# Summary

For this assignment, I reviewed the modules, lectures, reading material, and videos before starting. Once I had done this I started by completing what I perceived to be the easiest TODO's in the code. This included taking blocks of code and defining functions for the blocks. The functionality for the code was tested in Spyder in Terminal and worked as expected. It was mentioned in class that working with somebody else’s code can in some ways be easier and then in other ways be harder, I now understand.

# Appendix

GitHub Link - <https://github.com/KentBloods/Assignment06>

|  |  |
| --- | --- |
| 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 | *#------------------------------------------#*  *# Title: Assignment06\_Starter.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# KBloodsworth, 2022-Mar-06, Completed TODO's*  *#------------------------------------------#*  *# -- 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*  @staticmethod  **def** add\_item(strID, strTitle, strArtist):  *"""Function to add item to table*    *Args:*  *strID(string): CD ID, input from user to add to table*  *strTitle(string): CD title, input from user to add to table*  *strArtist(string): artist name, input from user to add to table*    *Returns:*  *None.*  *"""*  intID = int(strID)  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}  lstTbl.append(dicRow)  IO.show\_inventory(lstTbl)      @staticmethod  **def** del\_CD(ntIDDel):  *"""Function to delete CD entry from 2D list of dictionary table*    *Allows the user to choose to delete a CD entry from the table*    *Args:*  *None*    *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(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 the table to file.*    *Writes the 2D data structure table consisting of list of dicts to file*    *Args:*  *file\_name (string): name of file used to write the 2D data structure to, from table: list of dicts*  *written to file*    *Returns:*  *None*  *"""*  *# TODONE Add code here*  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('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** add\_input():  *"""Function asks user to input new ID, Title, and Artist*    *Args:*  *None.*    *Returns:*  *strID(string): CD ID, input from user*  *strTitle(string): CD title, input from user*  *strArtist(string): artist name, input from user*    *"""*    strID = input('Enter ID: ').strip()  strTitle = input('What is the CD**\'**s title? ').strip()  strArtist = input('What is the Artist**\'**s name? ').strip()  **return** strID, strTitle, strArtist    *# 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*  strID, strTitle, strArtist = IO.add\_input()  *# 3.3.2 Add item to the table*  *# TODONE move processing code into function*  DataProcessor.add\_item(strID, strTitle, strArtist)  **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*  *# TODO move processing code into function*  DataProcessor.del\_CD(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()  *# 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') |