Kent Bloodsworth

March 20, 2022

Foundation of Programming (Python)

Assignment08

Title

# Introduction

In this assignment, we looked at the differences between class and the objects made in a class, and the different components that make up a typical class. We also looked at when to use the keyword self and static method. After reviewing the materials for module 8, the assignment was to complete the starter script included for the module. The starter script had a pseudocode, and the goal was to read and understand the pseudocode and then add code to the script to make the application work.

# Completing the TODOs

To complete this assignment, we had to read and understand the pseudocode included in the starter code. This seems straightforward enough; however, it wasn't as easy as it may sound, and I know there are better ways to do it than I did it. A lot of the pseudocode looks like I would be able to use code from previous CD inventory assignments, which seemed to be the case for some of it. However, the CD class was something new, and incorporating it with the rest of the code was challenging. I spent much time looking back at the module PDF and think that I have the bulk of the code there but struggled with defining the methods and how to get them to interact and be happy with the rest of the script.

# Screenshots of Working Script

Below are figures containing screenshots of the script working in the Spyder IDE and in the terminal console. Figures 1, 2, and 3 show the script working in the Spyder IDE, and Figure 4 shows the script working in the terminal console.

Text

Description automatically generated

Figure - Screenshot of script working in Spyder

Text

Description automatically generated

Figure - Screenshot of script working in Spyder

Text

Description automatically generated

Figure - Screenshot of script working in Spyder

Text

Description automatically generated

Figure - Screenshot of script working in terminal console

# Summary

For this assignment, I looked at the pseudocode provided in the starter script and started working through the TODOs. Working through the TODOs helped me understand the relation between the class and objects and how it all ties together. in addition to learning about the class constructor in the keyword self. Although I'm not confident the script is currently as intended, I have a functioning script.

# Appendix

GitHub link – <https://github.com/KentBloods/Assignment08>

Code –

|  |  |
| --- | --- |
| 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  289  290  291  292  293  294  295  296  297  298  299  300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324 | *#------------------------------------------#*  *# Title: Assignmen08.py*  *# Desc: Assignnment 08 - Working with classes*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, created file*  *# DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08*  *# KBloodsworth, 2022-Mar-20, Completed TODO's*  *#------------------------------------------#*  *# -- DATA -- #*  strFileName = 'cdInventory.txt'  lstOfCDObjects = []  **class** **CD**:  *"""Stores data about a CD:*  *properties:*  *cd\_id: (int) with CD ID*  *cd\_title: (string) with the title of the CD*  *cd\_artist: (string) with the artist of the CD*  *methods:*  *"""*  *# TODone Add Code to the CD class*    *# --Fields-- #*  *# --Constructor-- #*  **def** \_\_init\_\_(self, ID, Title, Artist):  self.\_\_ID = int(ID)  self.\_\_Title = str(Title)  self.\_\_Artist = str(Artist)    *# --Properties-- #*  @property  **def** ID(self):  **return** self.\_\_ID    @ID.setter  **def** ID(self, value):  self.\_\_ID = int(value)    @property  **def** Title(self):  **return** self.\_\_Title    @Title.setter  **def** Title(self, value):  self.\_\_Title = str(value)    @property  **def** Artist(self):  **return** self.\_\_Artist    @Artist.setter  **def** Artist(self, value):  self.\_\_Artist = str(value)      *# -- PROCESSING -- #*  **class** **FileIO**:  *"""Processes data to and from file:*  *properties:*  *methods:*  *save\_inventory(file\_name, lst\_Inventory): -> None*  *load\_inventory(file\_name): -> (a list of CD objects)*  *"""*    @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.*  *"""*  *# TODone Add code to process data from a file*  *#Lets the user know that the file is missing*  **try**:  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()  **except** **FileNotFoundError**:  print('**\n**File not found!**\n**')  **except** **EOFError**:  print('**\n**File is empty!**\n**')    @staticmethod  *# TODone Add code to process data to a file*  **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*  *"""*  *#Lets the user know that the file is missing*  **try**:  objFile = open(strFileName, 'w')  **for** row **in** table:  lstValues = list(row.values())  lstValues[0] = str(lstValues[0])  objFile.write(','.join(lstValues) + '**\n**')  objFile.close()  **except** **FileNotFoundError**:  print('**\n**File not found!**\n**')    *# -- PRESENTATION (Input/Output) -- #*  **class** **IO**:  *"""Handling Input / Output"""*  @staticmethod  *# TODone add code to show menu to user*  **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  *# TODone add code to captures user's choice*  **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  *# TODone add code to display the current data on screen*  **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('======================================')  @staticmethod  *# TODone add code to get CD data from user*  **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      **class** **DataProcessor**:  *# TODone add functions for processing here*  @staticmethod  **def** add\_item(strID, strTitle, strArtist):  *"""Function to add item to lstOfCDObjects*    *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.*  *"""*  *#Tell user the value must be an integer*  **try**:  intID = int(strID)  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}  print('ID must be an integer')  lstOfCDObjects.append(dicRow)  IO.show\_inventory(lstOfCDObjects)  **except** **ValueError**:  print('**\n**Must be an integer!**\n**')    @staticmethod  **def** del\_CD(ntIDDel):  *"""Function to delete CD entry from the lstOfCDObjects*    *Allows the user to choose to delete a CD entry from the lstOfCDObjects*    *Args:*  *None*    *Returns:*  *None*    *"""*  intRowNr = -1  blnCDRemoved = **False**  **for** row **in** lstOfCDObjects:  intRowNr += 1  **if** row['ID'] == intIDDel:  **del** lstOfCDObjects[intRowNr]  blnCDRemoved = **True**  **break**  **if** blnCDRemoved:  print('The CD was removed')  **else**:  print('Could not find this CD!')    *# TODone add docstring*    *# -- Main Body of Script -- #*  *# TODone Add Code to the main body*  *# 1. When program starts, read in the currently saved Inventory*  FileIO.read\_file(strFileName, lstOfCDObjects)  *# 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...')  FileIO.read\_file(strFileName, lstOfCDObjects)  IO.show\_inventory(lstOfCDObjects)  **else**:  input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')  IO.show\_inventory(lstOfCDObjects)  **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*  strID, strTitle, strArtist = IO.add\_input()  *# 3.3.2 Add item to the table*  DataProcessor.add\_item(strID, strTitle, strArtist)  **continue** *# start loop back at top.*  *# 3.4 process display current inventory*  **elif** strChoice == 'i':  IO.show\_inventory(lstOfCDObjects)  **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*  **try**:  IO.show\_inventory(lstOfCDObjects)  *# 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*  DataProcessor.del\_CD(intIDDel)  IO.show\_inventory(lstOfCDObjects)  **continue** *# start loop back at top.*  **except** **ValueError**:  print('**\n**Must be an integer!**\n**')  *# 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(lstOfCDObjects)  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*  FileIO.write\_file(strFileName, lstOfCDObjects)  **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') |