< Sara Lam >

< 12/4/2021 >

< Foundations of Programming (Python) >

< Assignment08>

How I performed Assignment08

# Introduction

In Assignment08, we will use a starter script with pseudocode, and add code and docstrings to complete the program.

We will continue to use GitHub. We will upload the python script and knowledge document on the GitHub repository named “Assignment\_08”, commit changes, share a link on canvas, add link to this knowledge document. We will also perform a peer review of another student’s assignment.

We then save the script file as CD\_Intentory.py into folder “Assignment08”. Submit this folder as a zip file, including the python script as well as this knowledge document.

# Step 1 – examine starter code

The script contains the structure and pseudocode as a starter file.

# Step 2 – add code based on requirements

I created the code for the following:

* Added code to class CD with \_\_init\_\_() method to create private attributes with getter properties for id, title and artist. Also include static methods for processing data, i.e. add\_item() to add CDs, and delete\_item() to delete CDs.
* Added code to class FileIO with static methods read\_file() and write\_file(). Change code from previous assignments to handle data in list of CD objects.
* Added code to class IO. This is largely the same as Assignment07. The only difference is in static method show\_inventory(), I used list of CD objects instead of list of dictionaries to display inventory.
* The main section is largely the same as in previous assignment, except I had to change the class name to reflect the name used in this assignment.
* Saved the script as CD\_Inventory.py to the Assignment08 folder.

# Step 3 – run the python script file in Spyder, and verify it worked

Click the run button to run the file. It describes what the program does, allows the user to choose from menu, and performs the functions based on user choice. It repeats till user enters exit. When no file exists it handles the error and displays the message. Screenshot is below:

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Figure - Script run in Spyder – when program first started without file and an attempt was made to load the file, it correctly displays message from error handling

Text

Description automatically generated

Figure 2 - Script run in Spyder – when trying to add CDs but not entering integer for ID, or not entering value for title and artist, it correctly displays error messages

Text

Description automatically generated

Figure 3 - Script run in Spyder – added 2 CDs correctly

Text

Description automatically generated

Figure 4 - Script run in Spyder – saved and loaded file

Text

Description automatically generated

Figure 5 - Script run in Spyder – when user tries to delete but entering invalid CD number, it displays messages correctly

Text

Description automatically generated

Figure - Script run in Spyder – deleted a CD correctly

Text

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Figure 7 – Saving to file. File displays updated CD records

# Step 4 – run the python script file in a terminal window, and verify it worked

In Windows search, I type cmd to open the Command Prompt. Then I change to the directory with my python script, which is C:\\_FDProgramming\Assignment08. I type CD\_Inventory.py to run the program.

Text

Description automatically generated

Text

Description automatically generated

Figure - Script run in terminal window – loading cdInventory.txt created in Spyder, adding, deleting CDs and saving back to file

Graphical user interface, application

Description automatically generated

Figure - Script run in terminal window updated cdInventory.txt

Graphical user interface, text

Description automatically generated

Figure - cdInventory.txt correctly shows updates

# Step 5 – upload script and knowledge document to GitHub

<https://github.com/Lamcloud/Assignment_08>

# Summary

In this assignment we practiced the following -

* Using object to store data
* Using while loop and for loop
* Using conditional statement to control the flow
* Using break to break out of a section
* Reading from and writing to a text file
* Creating classes and functions for separation of concerns
* Creating docstrings for classes and functions to explain their use
* Using structured error handling

# Challenges

Below are the challenges I encountered and how I handled them -

* It took me a while to think of how to start writing the program. Then I start from previous assignment07. For Assignment07 I actually first started with using a text file, and when it ran successfully I changed to binary file. So I just started with the version with text file because that’s the file type used here. I looked at the entire structure of the program, which largely stayed the same. The main difference is using objects instead of dictionaries to store data. So I made changes in those particular areas involving data storage, and updated the other impacted parts of the program.
* When user doesn’t enter anything for title and artist, I added the error handling in the Constructor. I don’t think that’s the best place to put error handling, because Dirk said we should place error handling closest to where it can happen. I think it should be placed in the ask\_user() function when user enters the information. However I couldn’t figure out how to handle more errors after handling the ValueError (when user doesn’t enter an integer for id), and still return the id, title and artist (strID, strTitle, strArtist). I thought of another option of creating 3 functions to ask the user, e.g. ask\_id(), ask\_title(), ask\_artist(), handling error in each function, but then thought I should not break up this function because it should be grouped together. So I chose the Constructor to handle errors for title and artist.

# Appendix – the code

I used this syntax highlighting application, <http://hilite.me/>.

|  |  |
| --- | --- |
| 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  325  326  327  328  329  330  331  332  333  334  335 | *#------------------------------------------#*  *# 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*  *# SLam, 2021-Dec-04, added code to complete assignment 08*  *#------------------------------------------#*  *# -- DATA -- #*  strFileName = 'cdInventory.txt'  lstOfCDObjects = []  **class** **CD**:  *"""Stores data about a CD:*    *properties:*  *id: (int) with CD ID*  *title: (string) with the title of the CD*  *artist: (string) with the artist of the CD*  *methods:*  *"""*  *# TODone Add Code to the CD class*  *# -- Constructor --*  **def** \_\_init\_\_(self, id, title, artist):  *# -- Attributes --*  self.\_\_id = id    **if** title.strip() != '':  self.\_\_title = title  **else**:  **raise** **Exception**('CD title can**\'**t be empty!')    **if** artist.strip() != '':  self.\_\_artist = artist  **else**:  **raise** **Exception**('Artist name can**\'**t be empty!')    *# -- Properties --*  @property *# Getter property*  **def** id(self):  **return** self.\_\_id    @property *# Getter property*  **def** title(self):  **return** self.\_\_title    @property *# Getter property*  **def** artist(self):  **return** self.\_\_artist    *# -- Methods --*  @staticmethod  **def** add\_item(strID, strTitle, strArtist):  *"""Add CD to list of dicts*    *Args:*  *strID (string): ID of the CD*  *strTitle (string): Title of CD*  *strArtist (string): Artist of CD*  *Returns*  *-------*  *lstOfCDObjects.*  *"""*  intID = int(strID)  cd = CD(intID, strTitle, strArtist)  lstOfCDObjects.append(cd)  **return** lstOfCDObjects    @staticmethod  **def** delete\_item(lstOfCDObjects):  *"""Delete CD from list of dicts*    *Args:*  *lstOfCDObjects: 2D data structure (list of objects) that holds the data during runtime*    *Returns*  *-------*  *lstOfCDObjects: 2D data structure (list of objects) that holds the data during runtime*  *"""*  intRowNr = -1  blnCDRemoved = False  **for** cd **in** lstOfCDObjects:  intRowNr += 1  **if** cd.id == intIDDel:  **del** lstOfCDObjects[intRowNr]  blnCDRemoved = True  **break**  **if** blnCDRemoved:  **print**('The CD was removed')  **else**:  **print**('Could not find this CD!')  **return** lstOfCDObjects    *# -- PROCESSING -- #*  **class** **FileIO**:  *"""Processes data to and from file:*  *properties:*  *methods:*  *read\_file(file\_name, table): -> None*  *write\_file(file\_name, lstOfCDObjects): -> (a list of CD objects)*  *"""*  *# TODone Add code to process data from a 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 CD objects): 2D data structure that holds the data during runtime*  *Returns:*  *None.*  *"""*  *# Check if file exists*  **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(',')  cd = CD(int(data[0]),data[1],data[2])  table.append(cd)  *# adding for clarity*  **print**('table from read\_file function:',table)  objFile.close()  *# If file doesn't exist, tell user*  **except** FileNotFoundError **as** e:  **print**('Text file does not exist!')  **print**(e)  *# TODone Add code to process data to a file*  @staticmethod  **def** write\_file(file\_name, table):  *# TODone Add code here*  *"""Function to save data to file*    *Args:*  *file\_name (string): name of file used to write the data to*  *table (list of CD objects): 2D data structure that holds the data during runtime*    *Returns:*  *None*  *"""*  objFile = open(file\_name, 'w')  **for** cd **in** table:  tplValues = (str(cd.id), cd.title, cd.artist)  objFile.write(','.join(tplValues) + '**\n**')  objFile.close()  *# -- PRESENTATION (Input/Output) -- #*  **class** **IO**:  *# TODone add docstring*  *"""Class to process user input and display output"""*    *"""Handling Input / Output"""*  *# TODone add code to show menu to user*  @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**')  *# TODone add code to captures user's choice*  @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  *# TODone add code to display the current data on screen*  @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** cd **in** table:  **print**('{}**\t**{} (by:{})'.format(cd.id, cd.title, cd.artist))  **print**('======================================')  *# TODone add code to get CD data from user*  @staticmethod  **def** ask\_user():  *"""Ask user to enter ID, Title, and Artist of CD*    *Args:*  *None*  *Returns*  *-------*  *strID (string): ID of the CD*  *strTitle (string): Title of CD*  *strArtist (string): Artist of CD*  *"""*  **try**:  strID = int(input('Enter an integer for ID: ').strip())  strTitle = input('What is the CD**\'**s title? ').strip()  strArtist = input('What is the Artist**\'**s name? ').strip()  **return** strID, strTitle, strArtist  **except** **ValueError** **as** e:  **print**('ID has to be an integer!')  **print**(e)    *# -- Main Body of Script -- #*  *# TODone Add Code to the main body*  *# 1. When program starts, read in the currently saved Inventory*  *# Load data from file into a list of CD objects on script start*  FileIO.read\_file(strFileName, lstOfCDObjects)  *# 2. start main loop*  **while** True:  *# 2.1 Display Menu to user and get choice*  *# Display menu to user*  *# show user current inventory*  *# let user add data to the inventory*  *# let user save inventory to file*  *# let user load inventory from file*  *# let user exit program*    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*  *# Check if strings are empty from ask\_user() function*  **try**:  strID, strTitle, strArtist = IO.ask\_user()  lstOfCDObjects = CD.add\_item(strID, strTitle, strArtist)  IO.show\_inventory(lstOfCDObjects)  **except** **Exception** **as** e:  **print**(e)  **print**('Nothing to add.')  **print**()  **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*  IO.show\_inventory(lstOfCDObjects)  *# 3.5.1.2 ask user which ID to remove*  **try**:  intIDDel = int(input('Enter an integer for the ID you would like to delete: ').strip())  *# 3.5.2 search thru table and delete CD*  lstOfCDObjects = CD.delete\_item(lstOfCDObjects)  IO.show\_inventory(lstOfCDObjects)  **except** **ValueError** **as** e:  **print**('That is not an integer! <<< Customer Message')  **print**(e)  **print**()  **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(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*  *# TODone move file processing code into function*  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') |