< Sara Lam >

< 11/20/2021 >

< Foundations of Programming (Python) >

< Assignment06>

How I performed Assignment06

# Introduction

Assignment06 asks us to take the Assignment06\_Starter.py file which has the solution from previous assignment, replace certain sections with classes and functions.

We will continue to use GitHub. We will upload the python script and knowledge document on the GitHub repository named “Assignment\_06”, 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 CDIntentory.py into folder “Assignment06”. Submit this folder as a zip file, including the python script as well as this knowledge document.

# Step 1 – examine Assignment06\_Starter.py

The script contains the solution of Assignment05. Additionally, it provides the structure for us to add code to create the classes and functions to perform certain tasks.

# Step 2 – add code based on requirements

I create the statements for the following:

* For the loading section, used exists function to check if file exists.
* For the deleting section, used for loop and conditional statement to check if CD number is in inventory for deletion.
* Converted inner data structure from list to dict.

I save the script as CDInventory.py to the Assignment05 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. Screenshot is below:

Graphical user interface, application, Word

Description automatically generated

Text

Description automatically generated

Figure 1 - Script run in Spyder – when program first started without file, it correctly displays “File CDInventory.txt does not exist.”

Text

Description automatically generated

Figure - Script run in Spyder – adding 2 CDs

Text

Description automatically generated

Figure - Script run in Spyder – displaying CDs. Trying to delete CD 8 that does not exist

Text

Description automatically generated

Figure - Script run in Spyder – deleted CD2

Text

Description automatically generated

Figure - Script run in Spyder – adding CD3 and displaying

Text

Description automatically generated

Figure 6 - Script run in Spyder – saving to CDInventory.txt

Text

Description automatically generated

Figure – 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\Assignment06. I type CDInventory.py to run the program.

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Figure 8 - Script run in terminal window – loading CDInventory.txt created in Spyder, added CD4, deleted CD1

Graphical user interface, application

Description automatically generated

Figure 9 - Script run in terminal window updated CDInventory.txt

Graphical user interface, text

Description automatically generated

Figure 10 - CDInventory.txt correctly shows CDs 2 and 1 remaining, and CD 3 has been deleted.

# Step 5 – upload script and knowledge document to GitHub

# <https://github.com/Lamcloud/Assignment_06>

# Summary

In this assignment we practiced the following -

* Using dictionary to store data
* Using the dictionary values function to get the values
* 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 file
* Creating classes and functions for separation of concerns
* Creating docstrings for functions to explain their use

# Challenges

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

* It first seemed daunting to see a program with 200+ lines. So I looked closely at the “TODO” sections, and found there are 7 “TODOs”. When I carefully examined them, and looked at how the existing codes inside the classes and functions and how they are called as examples, I followed the pattern to move the code into the appropriate places of functions within the classes. In Friday’s session Laura also pointed out we need to replace current sections with the calling of the functions, which was very helpful in clarifying what I needed to do.
* At first I didn’t understand the difference between docstrings and regular comments because they look similar. I asked in class and Dirk explained docstrings appear as tooltips for the functions.

# 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 | *#------------------------------------------#*  *# Title: CDInventory.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# SLam, 2020-Nov-20*  *#------------------------------------------#*  **import** **os.path** *# to use exists function to check if file exists for loading*  *# -- 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 in list of dicts"""*    @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*  *-------*  *lstTbl.*  *"""*  intID = int(strID)  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}  lstTbl.append(dicRow)  **return** lstTbl    @staticmethod  **def** delete\_item(lstTbl):  *"""Delete CD from list of dicts*    *Args:*  *lstTbl: 2D data structure (list of dicts) that holds the data during runtime*    *Returns*  *-------*  *lstTbl: 2D data structure (list of dicts) that holds the data during runtime*  *"""*  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!')  **return** 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.*  *"""*  *# Check if file exists*  **if** os.path.exists(strFileName) == True:  *# Load existing data if file exists*    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()  *# If file doesn't exist, tell user*  **else**:  **print**('File', strFileName, 'does not exist.')    @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 dict): 2D data structure (list of dicts) that holds the data during runtime*    *Returns:*  *None*  *"""*  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** 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*  *"""*  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.ask\_user()  *# 3.3.2 Add item to the table*  *# TODone move data processing code into function*  lstTbl = DataProcessor.add\_item(strID, strTitle, strArtist)  IO.show\_inventory(lstTbl)  **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 data processing code into function*  lstTbl = DataProcessor.delete\_item(lstTbl)  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 file 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') |