Kenneth Chiu

March 06, 2022

Foundations of Programming: Python

Assignment 06

The Magic CD Inventory Program - Continued

# Introduction

In this module, we learned about programmer defined functions and how to pass through different types of parameters/arguments, positional vs. named, for processing. We also learned how to use function to organize codes using the SoC principles to further improve the Magic CD Inventory Program.

# What are Functions?

Functions are statements, which are grouped together and can be accessed by programmer defined names following the “def” syntax. Similar to variables, functions need to be defined before they’re called. When a function is called, the codes are executed, programmer defined tasks are performed and results are returned.

# What are Parameters/Arguments?

Parameters, which are also called arguments, are values that can be passed through functions to be processed. Arguments can be positional, which means the values are assigned to the arguments based on the order they’re passed through the function, or named, which allows the values to be explicitly assigned to specific arguments, before they’re processed in the programmer defined tasks performed within the function.

# The Magic CD Inventory program

GitHub link - https://github.com/kennchiu/Assignment\_06

To complete the Magic CD Inventory program, I loaded the Assignment06\_Starter.py file and updated the script header – List 1.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | *#------------------------------------------#*  *# Title: Assignment06\_Starter.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# KChiu, 2022-Mar-06, Updated codes to complete the Magic CD Inventory Program*  *#------------------------------------------#* |

List - Script Header

Next, I converted codes throughout the program main body into custom functions and added to DataProcessor and FileProcessor Class sections wherever appropriate and made sure each function has proper docstring.

In the DataProcessor Class, I added the add\_cd() and delete\_cd() functions to perform tasks for option ‘a’ and ‘d’ in the program – List 2.

|  |  |
| --- | --- |
| 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 | **class** **DataProcessor**:  *# TODone add functions for processing here*  *"""Processing user input to add or delete CDs"""*  @staticmethod  **def** add\_cd(cd\_id, cd\_title, cd\_artist):  *"""Function to add a new cd to lstTbl table if user chooses to*  *Append the CD ID, Title and Artist Name of the new CD to the lstTbl table*  *in a dictionary list format.*  *Args:*  *cd\_id (integer): ID number of the new CD*  *cd\_title (string): Title of the new CD*  *cd\_artist (string): Arist name of the new CD*  *Returns:*  *None.*  *"""*  *# Convert ID to integer format since default user input is in string format*  intID = int(cd\_id)  *# Add item to the lstTbl table*  dicRow = {'ID': intID, 'CD Title': cd\_title, 'Artist': cd\_artist}  lstTbl.append(dicRow)    @staticmethod  **def** delete\_cd(cd\_id):  *"""Function to delete an existing cd if user chooses to*  *Deleted the user chosen CD from the lstTbl table.*  *An error is rasied to the user if the CD ID doesn't exist*  *Args:*  *cd\_id (integer): ID number of the existing CD to be deleted*  *Returns:*  *None.*  *"""*  intRowNr = -1  blnCDRemoved = **False**  **for** row **in** lstTbl:  intRowNr += 1  **if** row['ID'] == cd\_id:  **del** lstTbl[intRowNr]  blnCDRemoved = **True**  **break**  **if** blnCDRemoved:  print('The CD was removed')  **else**:  print('Could not find this CD!') |

List - DataProcessor Class with Add\_CD and Remove\_CD Functions

In the FileProcessor Class, I added the write\_file() function to perform tasks for option ‘d’ in the program – List 3. Note that I excluded the read\_file() function purposely since the codes were not modified.

|  |  |
| --- | --- |
| 67  68    92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115 | **class** **FileProcessor**:  *"""Processing the data to and from text file"""*  @staticmethod  **def** write\_file(file\_name, table):  *# TODone Add code here*  *"""Function to sync the data in memory to file by saving current table in csv format*    *Writes the data in the lstTbl in memory to the CDInventory.txt file.*    *Args:*  *file\_name (string): name of file used to write/save the data to*  *table (multiple lists): 2D data structure (values of multiple lists of dicts)*  *that holds the current data during runtime*  *Returns:*  *None.*    *"""*  *# save data*  objFile = open(file\_name, 'w')  **for** row **in** table:  lstValues = list(row.values())  lstValues[0] = str(lstValues[0])  objFile.write(','.join(lstValues) + '**\n**')  objFile.close() |

List - Write\_File Function in FileProcessor Class

Next, I replaced the codes in the main body of program with these new functions.

For option ‘a’, the add\_cd() function from the DataProcessor class is called and strID, strTitle and strArtist from user inputs are passed through the function to add the new cd to the CD Inventory table (lstTbl) – List 4.

|  |  |
| --- | --- |
| 198  199  200  201  202  203  204  205  206 | *# 3.3 process add a CD*  **elif** strChoice == 'a':  *# 3.3.1 Ask user for new ID, CD Title and Artist*  strID = input('Enter ID: ').strip()  strTitle = input('What is the CD**\'**s title? ').strip()  strArtist = input('What is the Artist**\'**s name? ').strip()  DataProcessor.add\_cd(strID, strTitle, strArtist)  IO.show\_inventory(lstTbl)  **continue** *# start loop back at top.* |

List - Option 'a' with add\_cd() Function

For option ‘d’, the delete\_cd() function from the DataProcessor class is called and intIDDel from user input is passed through the function. Before the chosen ID number is passed through the delete\_cd() function, I added a try/except loop to raise an error message if user provides a value that can’t be converted to an integer and the program doesn’t proceed with deletion until a valid input is provided – List 5.

|  |  |
| --- | --- |
| 211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227 | *# 3.5 process delete a CD*  **elif** strChoice == 'd':  *# 3.5.1 get User input 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*  **while** **True**:*# continues to ask until an integer is provided.*  **try**:  intIDDel = int(input('Which ID would you like to delete? ').strip())  **break**  **except**:  intIDDel = **None**  print('Please enter integers only!')  **continue***# start loop back and ask for an ID number.*  DataProcessor.delete\_cd(intIDDel)  IO.show\_inventory(lstTbl)  **continue** *# start loop back at top.* |

List - Option 'd' with delete\_cd() Function

For option ‘s’, the write\_file() function from the FileProcessor class is called and strFileName and lstTble values are passed through the function to sync and save the current CD inventory table to the CDInventory.txt file – List 6.

|  |  |
| --- | --- |
| 228  229  230  231  232  233  234  235  236  237  238 | *# 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':  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.* |

List - Option 's' with save\_file() Function

Finally, in the menu\_choice() function, I added an if statement to validate the user chosen option and raise an error if user did not provide an input from the menu – List 7.

|  |  |
| --- | --- |
| 118  119  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152 | **class** **IO**:  *"""Handling Input / Output"""*  @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()  **if** choice **not** **in** ['l', 'a', 'i', 'd', 's', 'x']:  print('Please enter a valid option!')  print() *# Add extra space for layout*  **return** choice |

List - menu\_choice() Function with Error Message for Invalid Choice

# Summary

This assignment appeared to be daunting at first; however, once I followed the instructions carefully, I was able to complete the assignment without issues. The custom function concept is definitely the most challenging topic so far and the concept was difficult to comprehend while I was going through the Tic-Tac-Toe program in the book. Even with the pseudocode laid out in Chapter 6 in the book, I feel like I’m still not able to write a program that is as complex and sophisticated as the Tic-Tac-Toe program at this point. Hopefully, as the program continues and with more practices, I’ll be able to write a new program on my own at the end of this class.

Everything below is required screenshots per assignment instructions.

Below are screenshots showing the CD Inventory program working on Spyder – Figure 1 through 9.

Text

Description automatically generated

Figure - Main Menu

Text

Description automatically generated

Figure - Load Existing CD Inventory from External File

Text

Description automatically generated

Figure - Display Current CD Inventory

Text

Description automatically generated

Figure - Add A New CD

Text

Description automatically generated

Figure - Add Another New CD

Text

Description automatically generated

Figure - Delete An Existing CD

Text

Description automatically generated

Figure - Save Current CD Inventory to External File

Text

Description automatically generated

Figure - Exist CD Inventory Program

Graphical user interface, text, application

Description automatically generated

Figure - CDInventory File after Running Through CD Inventory Program in Spyder

Below screenshots show Magic CD Inventory program run in the Terminal – Figure 10 through 14.

Text

Description automatically generated

Figure - CD Inventory in Terminal Part 1

Text

Description automatically generated

Figure - CD Inventory in Terminal Part 2

Text

Description automatically generated

Figure - CD Inventory in Terminal Part 3

Text

Description automatically generated

Figure - CD Inventory in Terminal Part 4

Graphical user interface, text, application

Description automatically generated

Figure - CDInventory File after Running Through CD Inventory Program in Terminal