**Kasisi Harris**

**2022-Aug-13**

**IT FDN 110 A Su 22: Foundations of Programming**

**Assignment\_06**

**GitHub link:** [kasisi79/Assignment\_06 (github.com)](https://github.com/kasisi79/Assignment_06)

Functions

# Introduction

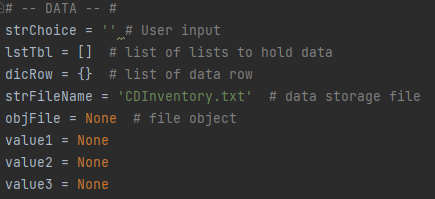
Building on the lessons from the previous week, using Dictionaries, the current task is to incorporate create and call Functions. It is easy to see how functions will be extremely useful moving forward as they can provide a clean way to organize code in a manner that is easily understood by others. They also allow organization in code completion as each function can be tackled independently and placed into a larger script, allowing for multiple coders to complete a single piece of code in a production line fashion. Furthermore, they can make troubleshooting code much easier being able to isolate errors to specific functions rather than trying to comb through an entire script.

# Script Creation in PyCharm

To modify this code, three functions were created, AddInfoInput, DeleteEntry, and write\_file. I will cover the details of these functions below and their placement below. However, first a few variables were added into the “—DATA --” section of the code to ensure the scripts functionality.

## Declaring Variables

Value1, Value2, and Value3 were declared as “None” in the data declaration section of the code in order to enable the AddInfoInput function to process the data correctly. Below is a figure displaying their addition.

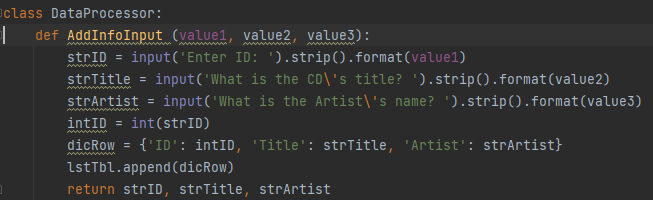


Listing - CDInventory.py - Declaration of Variables

## Class Data Processor Function Additions

### Adding Information: AddInfoInput

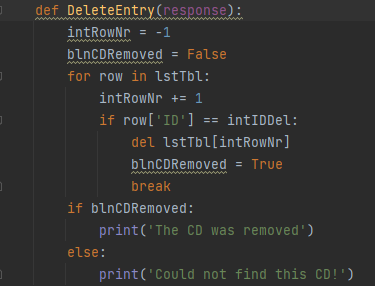
As mentioned earlier, the bulk of the changes this week can be found in the creation of functions. The first of these functions to be created facilitates the addition of CD information into dictionaries should the user select to add information. The function, AddInfoInput, uses three positional parameters: value1, value2, and value3. These parameters correlate to the three pieces of information requested by the user, ID, CD Title, and Artist which are then associated with a dictionary row and appended.



Listing - CDInventory.py - AddInfoInput

### Deleting Data: DeleteEntry

Should the user decide to delete one of the entries within the CD Inventory a pre identified function, IO.show\_inventory displays all entries available for deletion. The added function DeleteEntry prompts the user to enter the ID number associated with the row that they would like to delete. If successful, a confirmation text is displayed. Likewise, if the user chooses something outside of the range of what can be deleted an error message appears.

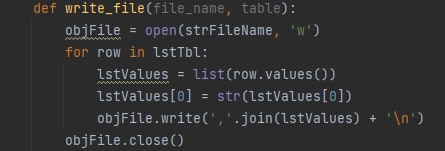


Listing - CDInventory.py - DeleteEntry

## Class File Processor Function Additions

### Saving Data: write\_file

The final function to be added was the write\_file function that saves the information to the CDInventory.txt file associated with this code.



Listing - CDInventory.py - write\_file

# Terminal Run Images

While most of the functionality was explained above, below are images of the program working in an Anaconda Terminal for reference.

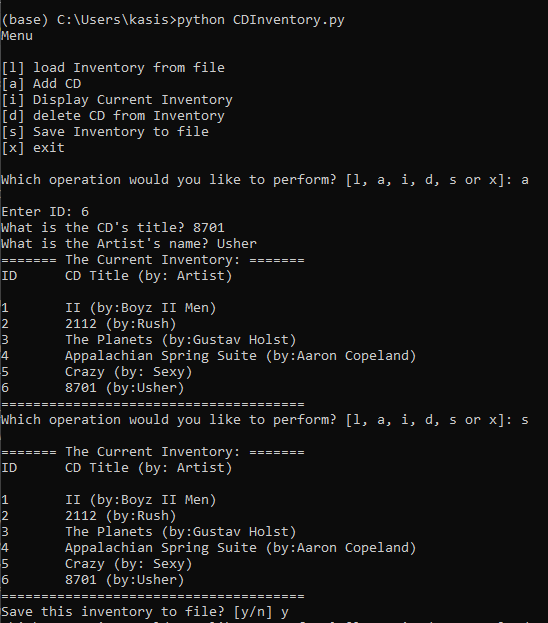


Figure - Terminal Run - Adding Data and Saving

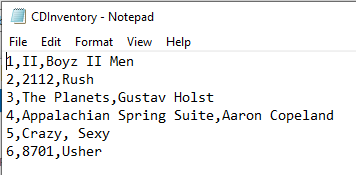


Figure - Addition 6 to text file

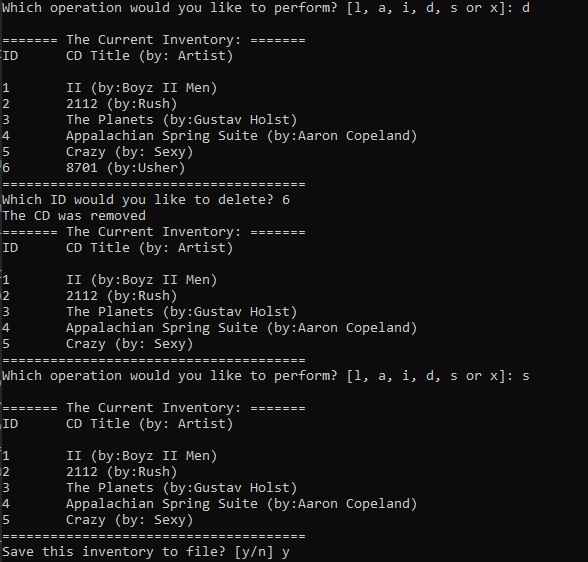


Figure - Terminal Run - Delete ID 6

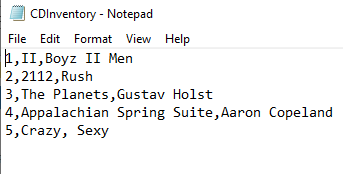


Figure - Deletion of ID 6 from text file

# Summary

While the incorporation of functions proved challenging parameter wise, the concept of using them feels straight forward. Again, the future use of functions would seem to have many benefits in terms of organization, clarity, code production, and troubleshooting. I look forward to implementing more of these in future lessons.

# Appendix

## Listing of CDInventory.py

|  |  |
| --- | --- |
| 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 | *#------------------------------------------#*  *# Title: Assignment06\_Starter.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# KHarris, 2022-Aug-13, Added code to replace TODO tasks*  *#------------------------------------------#*  *# -- 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*  value1 = **None**  value2 = **None**  value3 = **None**  *# -- PROCESSING -- #*  **class** **DataProcessor**:  **def** AddInfoInput (value1, value2, value3):  strID = input('Enter ID: ').strip().format(value1)  strTitle = input('What is the CD**\'**s title? ').strip().format(value2)  strArtist = input('What is the Artist**\'**s name? ').strip().format(value3)  intID = int(strID)  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}  lstTbl.append(dicRow)  **return** strID, strTitle, strArtist  **def** DeleteEntry(response):  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):  objFile = open(strFileName, 'w')  **for** row **in** lstTbl:  lstValues = list(row.values())  lstValues[0] = str(lstValues[0])  objFile.write(','.join(lstValues) + '**\n**')  objFile.close()  **pass**  *# -- 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:*  *Return strChoice*  *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('======================================')  *# 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*  *# 3.3.2 Add item to the table*  DataProcessor.AddInfoInput(value1,value2, value3)  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*  DataProcessor.DeleteEntry(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*  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') |