< Jayr Padilla >

< 11-20-22 >

< Foundations of Programming; Python >

< Assignment 06>

Working with classes and functions

# Introduction

The purpose of this assignment was to modify and organize the provided starter code as required to their proper classes and functions. This was to be completed with the information learned in module 6 such as defined functions, parameters and return values in functions, and classes.

# Functions: Return Values

**Going more in depth with functions is something I am glad we finally got to. I remember learning about them in an earlier module and knew that they would come back around. With this module and assignment, I could really see how powerful and helpful defining functions. Within the assignment the section that I took the most time organizing, and dissecting was the I/O section prompting the user to input an ID number, CD title, and Artist name. At first, I simply had one input function with all three prompts. Additionally, I had the return for that function include all the data inputted by the user (intID, strTitle, stArtist). It was not until I ran my script that: 1) I needed to store the return values with their respective variables so they may be properly added to the list table. 2) I needed to separate out each input to do so. I had defined intID, strTitle, stArtist all with blank strings but doing so even after trying to add data, the list table appeared blank. Another indicator that I wasn’t capturing the variables properly. Looking at Listing 2, you can see how I created different functions for each input that was prompted to the user. In Listing 1, in the main code, I assigned the return values for each function to their respective variable. In Figure 1 you can see the script properly saving the inputs to the table in Sypder.**

Text

Description automatically generated

Listing 1 - Assigning return values

Text

Description automatically generated

Listing 2 - Separated input functions w/ return

Text

Description automatically generated

Figure - Script saving input to table (Spyder)

# DocStrings

**DocStrings have also proven useful and tied in with last week’s module separations of concerns. It allows the programmer to include additional information about a function. This includes a header at the beginning of the function and then arguments and returns. Writing out why we are creating the function, arguments, and what will be returned allowed a clearer picture on why the function was being created and where it will be used. For this specific assignment, writing out a DocString made me think if the function needed parameters or if return values were needed. At first, I thought that created DocStrings had no value. But when going through and moving functions to their proper classes and creating new ones, I saw how constructing the code became easier. It does add more time when creating a script with functions, but it is a good coding practice to continue to do.**

Text

Description automatically generated

Figure -CDInventory.py running in terminal (Delete Function)

Using [Saravijust Syntax](https://saravjishut.org/syntax) (external reference)[[1]](#footnote-1) web page

# Summary

In this assignment I learned various topics to help me build this script. The main part of this module focused on how to properly build defined functions. This included looking at parameters within a function and return values. With return values, storing the value in a variable was a huge part of building my script and allowed me to properly troubleshoot I/O functions in my file. Finally, DocStrings for functions gave me a clearer picture on what each functions purpose was and to also think through if parameters and return values were needed. Looking at Figure 1 and 2, you can see parts of the script (saving entries and deleting entries) being executed in Spyder and in Terminal, respectively

# Appendix

Web link to GitHub Depository: <https://github.com/jayrpad/Assignment06>

## Listing 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  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 | *#------------------------------------------#*  *# Title: CDInventory.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *#JPadilla, 2022-Nov-20, Moved appropiate functions into classes*  *#JPadilla, 2022-Nov-20, Created add\_data function under DataProcessor class w/ DocString*  *#JPadilla, 2022-Nov-20, Created delete\_data function under DataProcessor class w/ DocString*  *#JPadilla, 2022-Nov-20, Added function calls to the main code (Lines 251-253, 255, 270)*  *#------------------------------------------#*  *# -- 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*  intID = '' *#user input ID number*  strTitle = '' *#user input CD title*  stArtist = '' *#user input artist name*  *# -- PROCESSING -- #*  **class** **DataProcessor**:  *"""Function to process/ save / delete any data entry made by user"""*  @staticmethod  **def** add\_data(ID, title, artist):  *"""Function that adds user input of ID, CD Title, Artist Name to a dictionary and then adds to table*  *Args:*  *ID: identification number for entry*  *Title: user inputted CD title*  *Artist: user inputted artist name*  *Returns:*  *None*  *"""*  dicRow = {'ID': intID, 'Title': title, 'Artist': artist}  lstTbl.append(dicRow)  @staticmethod  **def** delete\_data(ID):  *"""Function that finds desired entry to delete based on ID number*  *Args:*  *ID: identification number for entry*  *Returns:*  *None*  *"""*  intRowNr = -1  blnCDRemoved = **False**  **for** row **in** lstTbl:  intRowNr += 1  **if** row['ID'] == ID:  **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):  *"""Function that writes string data to file*  *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*  *"""*  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()  *# -- 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('======================================')  @staticmethod  **def** user\_idInput():  *"""Function that prompts user for new ID*  *Args:*  *None*  *Returns:*  *strID: user inputted identification number for entry*  *"""*  intID = int(input('Enter ID: ').strip())  **return** intID    @staticmethod  **def** user\_TitleInput():  *"""Function that prompts user for new CD Title*  *Args:*  *None*  *Returns:*  *strTitle: user inputted CD title*  *"""*  strTitle = input('What is the CD**\'**s title? ').strip()  **return** strTitle    @staticmethod  **def** user\_ArtInput():  *"""Function that prompts user for new CD Title*  *Args:*  *None*  *Returns:*  *stArtist: user inputted artist name*  *"""*  stArtist = input('What is the Artist**\'**s name? ').strip()  **return** stArtist  *# 1. When program starts, read in the currently saved Inventory*  objFile = open(strFileName, 'a')  objFile.close()  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*  intID = IO.user\_idInput()  strTitle = IO.user\_TitleInput()  stArtist = IO.user\_ArtInput()  *# 3.3.2 Add item to the table*  DataProcessor.add\_data(intID, strTitle, stArtist)  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.delete\_data(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') |

1. Retrieved 2022-Nov-20 [↑](#footnote-ref-1)