Sheena Harms

Nov. 20, 2022

IT FDN 110 B

Assignment 06

Functions!

# Introduction

This week I gained a decent understanding of how to use functions in Python. I demonstrated that knowledge by rearranging a CD Inventory script to include functions. I will explain how the functions in the code work, and some of the things that slowed me down.

# Adding Functions

The best thing about the code I was given to work with was that it was fully functional at the start. This was also the worst thing about it. Since there was nothing to correct, I didn’t really know how to begin.

I started by moving the open and write action into a function. I called that function in the program below and was greeted with an error. It took me a while to realize that I needed to list class first, because we just barely covered classes in the module so I didn’t understand yet how they needed to be used.

After figuring out the reading and writing I moved on to adding cds to the inventory, which took me a few tries to figure out. Initially I tried to move the inputs up as their own function, and then the row and append as its own function. This didn’t work because of encapsulation. All the user defined inputs in the first function didn’t mean anything to that second function. To fix this I moved the user inputs back down to the main program.

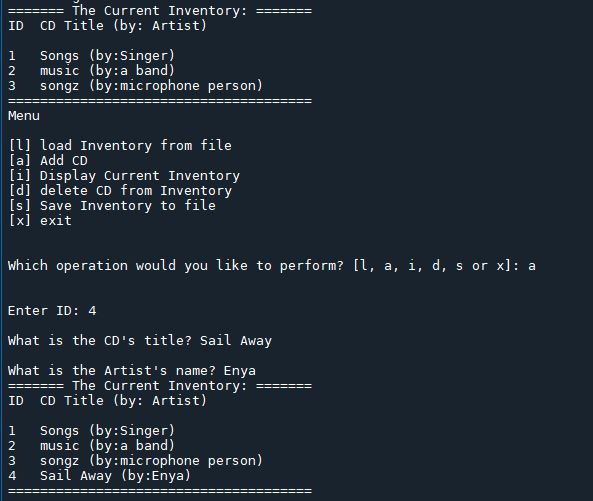


Figure : Adding a CD to CD Inventory, in Spyder.

# Some Struggles

The delete function was difficult for me to understand. I moved the action up into a function and it seemed to work just as well as before. However, I realized it needed to use arguments to ensure that it was going to delete records from the right spot. I suppose that if the code dealt with more tables then the arguments would be an important thing to implement. Before I added those arguments the delete function was working from the global variable. So, it still removed the record as requested, but that was only because it had one table to pull it from.

I also struggled with docstrings, primarily because I like pared down scripts and adding those strings made the code more bothersome for me to work in. There is a lot more hunting and scrolling. I can understand though how they would be helpful to the next programmer. I did appreciate the breakdown by numeric part.

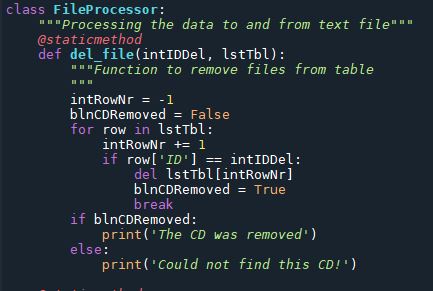


Figure 2 – Delete function (del\_file) in Spyder.

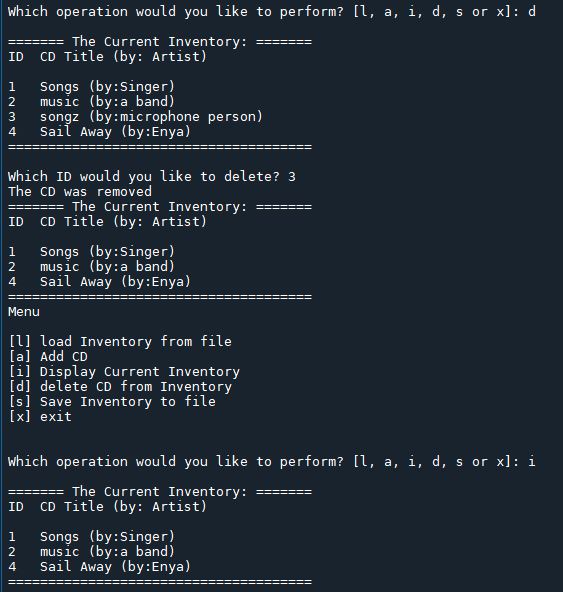


Figure 3 – Delete function running in Spyder console.

# Insight

I work in GIS and am a very spatial thinker. The more complicated these scripts become the more desperately I want to draw schematics for them. Not being able to see exactly how things cycle has been very difficult for me. I found a video on Python functions which did a good job of outlining exactly how functions work. The video tutorial is by Programiz and can be found here: https://youtu.be/-Bkupx9gX0o

# Summary

Adding your own functions to a script is an excellent way to control and streamline a program. Organizing a program into different parts and then commenting or adding docstrings makes your program easier to comprehend for future users.

\*See python and knowledge document for assignment 6 at <https://github.com/sheenahar/Assignment_06>

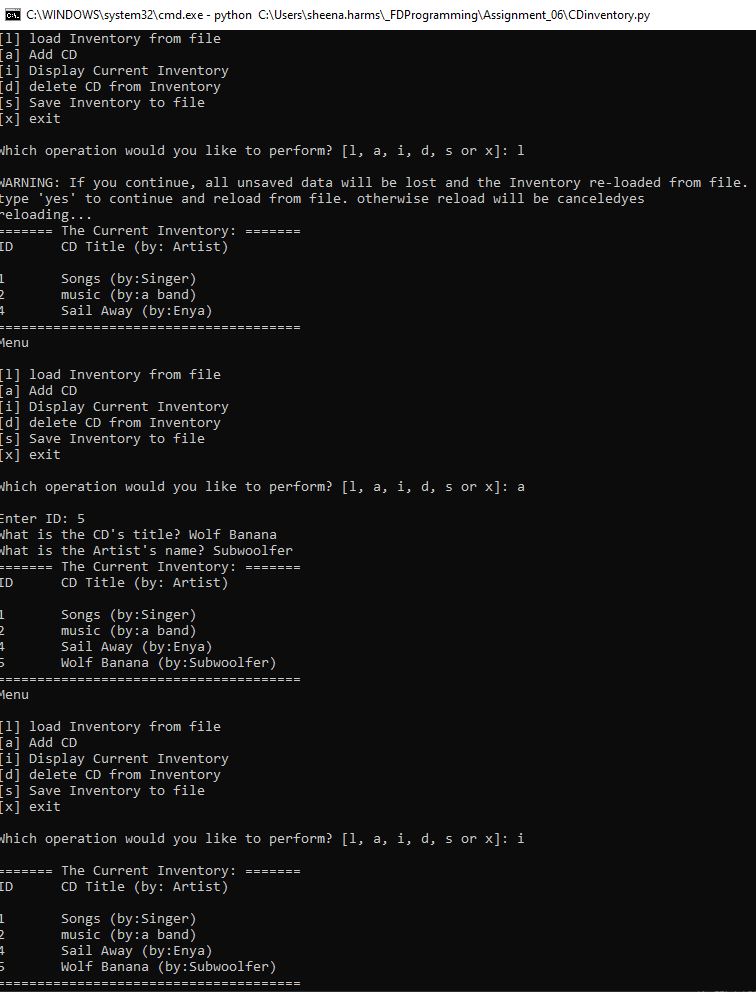


Figure 4: CD Inventory script in terminal.

# Appendix

## Listing CD\_Inventory.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 | *#------------------------------------------#*  *# Title: Assignment06\_Starter.py*  *# Desc: Working with classes and functions.*  *# Change Log: (Who, When, What)*  *# DBiesinger, 2030-Jan-01, Created File*  *# SHarms, 2022-Nov-18, Edited file*  *#------------------------------------------#*  *# -- 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**:  @staticmethod  **def** add\_cd(strID, strTitle, strArtist, lstTbl):  *"""Function adds data to dictionary row*  *Appends to list*  *Returns:*  *None."""*  intID = int(strID)  dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}  lstTbl.append(dicRow)  **pass**  **class** **FileProcessor**:  *"""Processing the data to and from text file"""*  @staticmethod  **def** del\_file(intIDDel, lstTbl):  *"""Function to remove files from table*  *Reads through rows for the input ID*  *Deletes row if input ID is found*  *Returns:*  *None"""*  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!')  @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(strfileName, lstTbl):  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:*  *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('======================================')  *# TODO add I/O functions as needed*  *# 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*  strID = input('Enter ID: ').strip()  strTitle = input('What is the CD**\'**s title? ').strip()  strArtist = input('What is the Artist**\'**s name? ').strip()  *# 3.3.2 Add item to the table*  DataProcessor.add\_cd(strID, strTitle, strArtist, lstTbl)  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*  FileProcessor.del\_file(intIDDel, 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*  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') |