**Dimitrios Tsakalos**

Sunday, February 14, 2021

IT FDN 110 A Wi 21: Foundations of Programming: Python

[Assignment 05](https://github.com/tsakdimi/Assignment_05)

**Creating a Magic CD Inventory program using Python**

# Introduction

In Assignment 05 for this course, I reviewed the material for Module 05 which included watching the module videos, reading book chapter 05, reading the suggested webpages and watching the additional videos. I read through the Foundations of Programming (Python) PDF and completed the assignment by applying my knowledge to create a script in Python that collects input from the user for a CD ID number, CD Title, Artist Name, displays the inventory, saves it, loads from file data and deletes user selected entries according to what the user inputs. The knowledge document and Python file were uploaded to [github](https://github.com/tsakdimi/Assignment_05) (external reference)[[1]](#footnote-1).

# Reviewing the material

The material I reviewed for Module 05 included watching the module videos, reading book chapter 05, looking at the suggested web pages and searching online for extra information for how to make my CD inventory program more functional for the user. I first looked at the Foundations of Programming (Python) PDF provided in module 05, which seemed a logical continuation from last week’s Assignment 04. After reviewing the materials and the assignment requirements, I started working on the labs step by step. The labs and appendix helped greatly and I gained an understanding of the new concepts of dictionaries and the usefulness of keys. This week’s challenges were:

* Understanding the importance of clearing data in memory before loading data from file
* Paying attention to data types and keeping data types in the same category (i.e. strings to strings, integers to integers)
* Finding a way to work within a 2D table to select specific values
* Understanding dictionary operations and how to access values

# Working on the Foundations of Programming (Python) Module 05

After reviewing the above materials, I proceeded to work on the Foundations of Programming Module 04 Labs. I started with LAB 05-A. I first reviewed the code and read the TODO descriptions. The program allows the user to perform five functions: exit, add data to memory, write data from memory to file, load data from file to memory and display data from memory to the user on the console. After opening the starter Python file, saving as and adding to the header I replaced the TODOs with my script (see Figure 1). I then proceeded to test the program in the Spyder console (see Figure 2) and verified the text file saved in the desired format (see Figure 3). The code performs the following:

* Lines 9 to 13 declare the variables
* Lines 16 to 21 contain the menu and introduce the while loop
* Lines 24 and 25 break the while loop when exit is entered by the user
* Lines 28 to 32 collect user input when the user inputs a and appends the information to a 2D table in memory
* Lines 35 to 43 execute when a user enters w and writes the information in memory to text file in a comma separated value format
* Lines 46 to 52 load to memory the data existing on file when a user enters r. The code reads the text file stripping trailing and leading characters (in our case \n) and splits rows into items on the , character. It is important to clear existing memory by using lstTbl.clear() so as not to get a concatenation of in memory and on file data when displaying the data.
* Lines 55 to 58 execute when a user inputs d and displays the data in memory to the user
* Lines 60 and 61 are there to account for user input out of the specified values and continue the loop after reminding the user of their options.

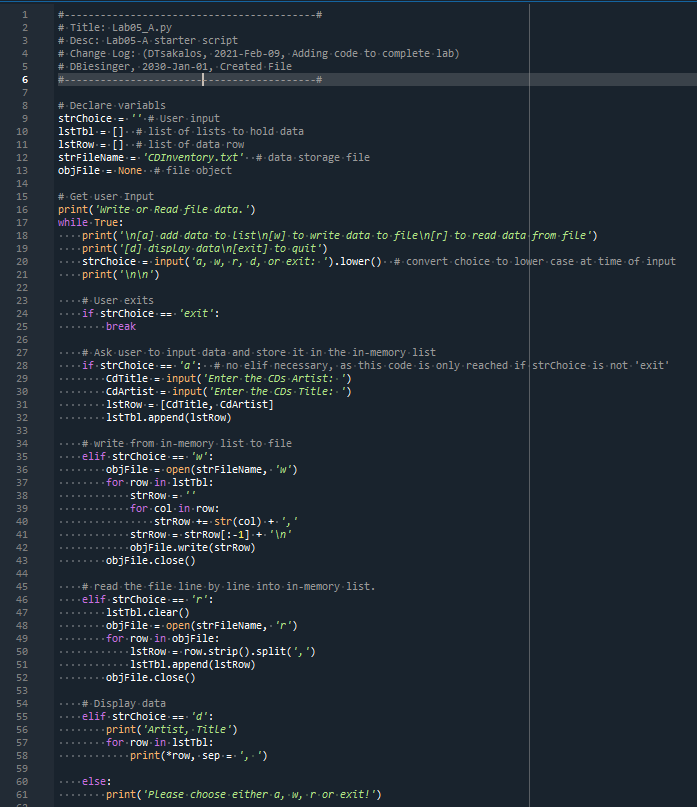


Figure - LAB05-A code

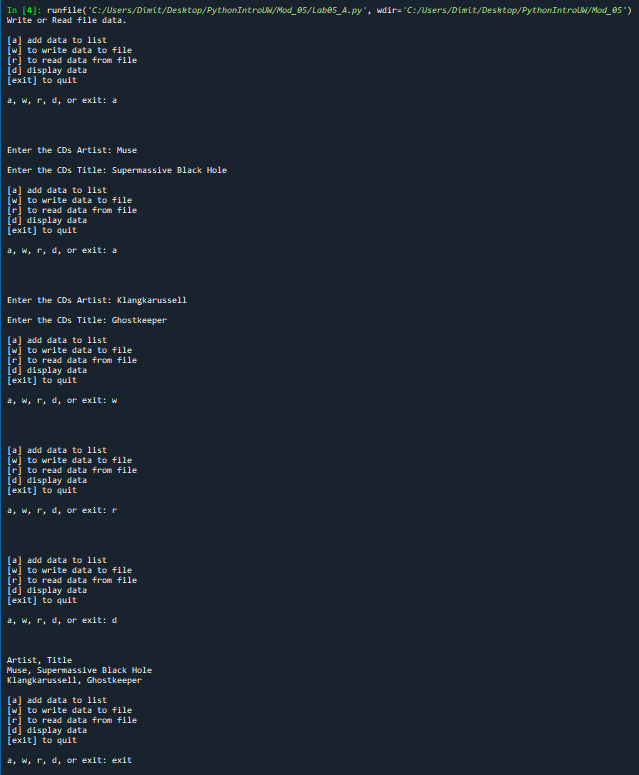


Figure - LAB05-A console

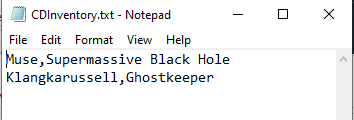


Figure - LAB05-A CDInventory.txt file saving in desired format

I used the code in LAB-05A as input to LAB-05B and modified it to use dictionaries instead of lists to hold data within the list 2D table. After replacing the inner lists with dictionaries, I added the element of an ID number withing each row (see Figure 4). This means that we cannot use the data as is from the text file created in LAB-05A, as our range has increased from 2 to 3. After modifying the text file to include indices I then tested the program in the Spyder console (see Figure 5) and verified the text file saved in the desired format (see Figure 6). The code performs the following:

* Lines 9 to 13 declare the variables
* Lines 16 to 21 contain the menu and introduce the while loop
* Lines 24 and 25 break the while loop when exit is entered by the user
* Lines 28 to 33 collect user input when the user inputs a as a string choice, define the dictionary keys and corresponding variable values and append the information to a 2D table in memory
* Lines 36 to 44 execute when a user enters w and write the information in memory to text file in a comma separated value format
* Lines 47 to 54 load to memory the data existing on file when a user enters r. The code reads the text file stripping trailing and leading characters (in our case \n) and splitting rows into items on the , character. It then lists the keys and instructs the program to treat the incoming three elements per row and assign them to the corresponding keys. It is important to clear existing memory by using lstTbl.clear() so as not to get a concatenation of in memory and on file data when displaying later the data.
* Lines 57 to 60 execute when a user inputs d and display the data in memory to the user. We make use of \*row.values() function to only unzip the values of the dictionary.
* Lines 62 and 63 are there to account for user input out of the specified values and continue the loop after reminding the user of their options

The script’s output in the provided solution (see Figure 7) can be improved mainly by adding user functionality. In the read section it would be helpful to include a printout of the data loaded into memory to let the user know that loading was successful and what exactly has loaded. I would do this by adding a print statement after the data loaded into memory. Second, information display could benefit from formatting to make it easier for the user to read the information from their console screen. The way to do this is covered in lines 58 to 60 in my shared code from LAB-05B (see Figure 4). Both of my improvements are related to more frequent and better visual representations of the data. Users rely heavily on visual cues to process information, so adding code that provides more information to the user increases functionality at minimal computing cost and is a good practice.

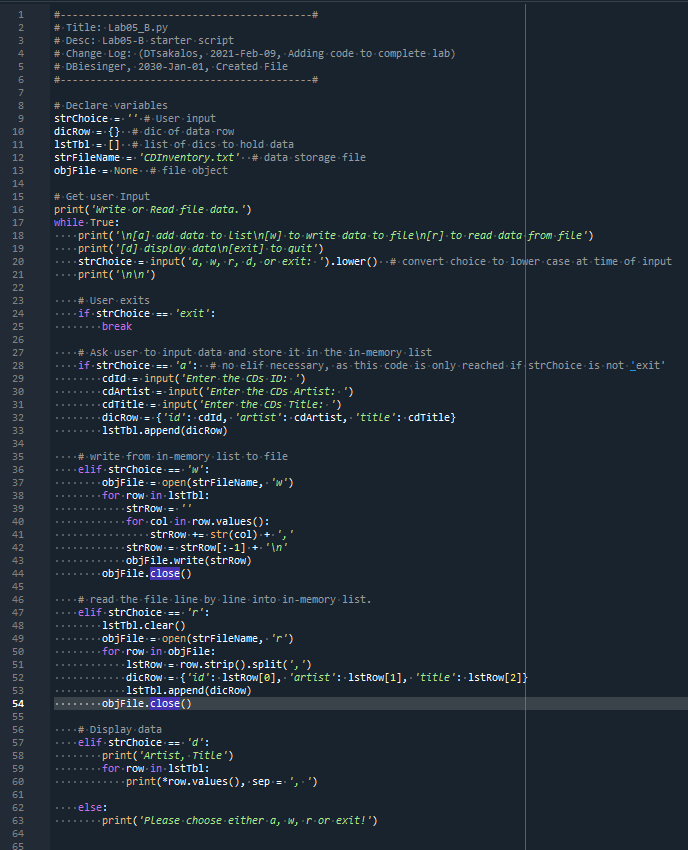


Figure - LAB05-B code

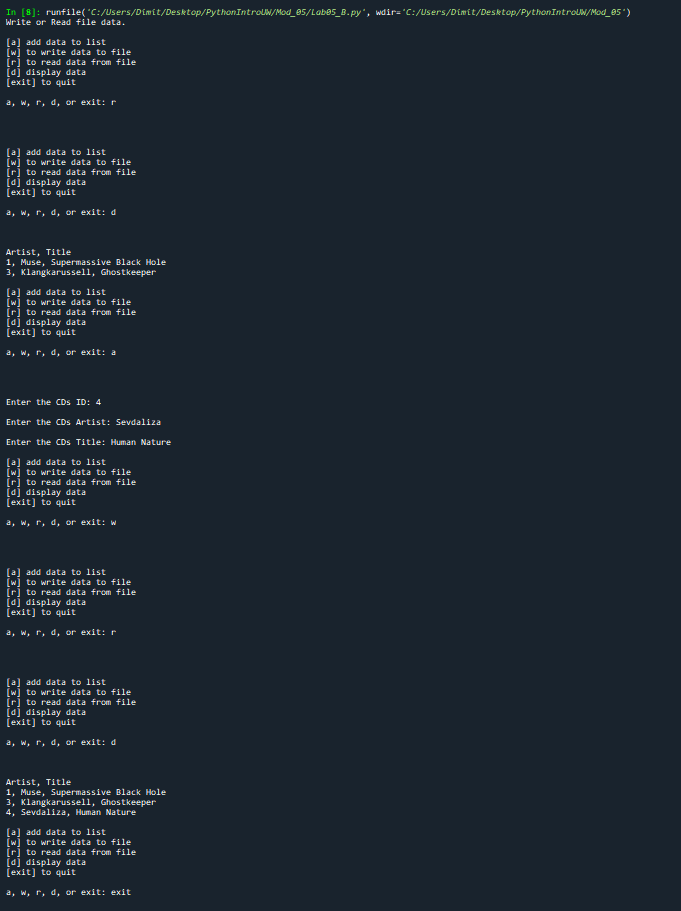


Figure - LAB05-B console

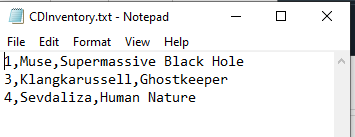


Figure - LAB05-B CDInventory.txt file saving in desired format

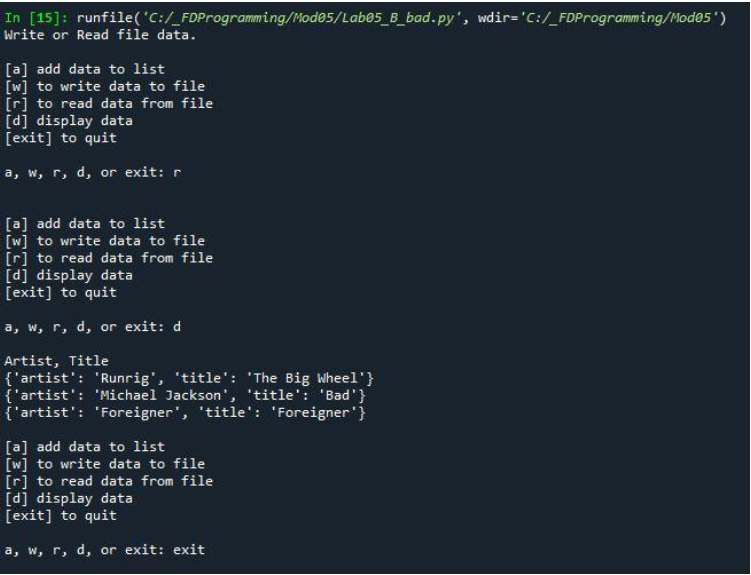


Figure - LAB05 - B solution with potential provided in module 05 PDF

# Creating a Magic CD Inventory program with increased functionality

After completing the labs, I moved onto Assignment05. After going through all the steps of folder creation and header modifications I started working on the code. The code performs the following:

* Lines 9 to 13 declare the variables
* Lines 16 and 17 introduce the program title and while loop
* Lines 20 to 24 display a menu and collect user input
* Lines 27 to 29 execute when the user selects x and exit the program after displaying a thank you message
* Lines 33 to 46 execute when the user selects l. The existing memory is erased, and the data is loaded externally from file. The data is read line by line from the text file and converted from string to a 2D table of dictionaries within a list. The data of each row is assigned to the different dictionary keys using , as the splitting character. The data is displayed with a header and formatted to be more presentable to the user.
* Lines 49 to 59 execute when the user selects a. After including a while, try, except statement to prevent crashing when the user does not enter a number, the code collects the entered data and adds to memory using the append method.
* Lines 62 to 67 execute when the user selects i and display the current in-memory data to the user with appropriate formatting to be more legible
* Lines 70 to 95 execute when the user selects d. First, they display a formatted version of the existing data, and then prompt the user to select an ID number to delete the corresponding entries. This block of code includes a while, try, except statement to avoid crashing when a user inputs something other than a number. After the user selects their ID number the program prints the number of rows deleted followed by the new in-memory data. The script used to delete *t*he row is listed on line 85. I came to this solution after searching online and found code posted by Alex Martelli on [stackoverflow](https://stackoverflow.com/questions/1235618/remove-dictionary-from-list) (external reference)[[2]](#footnote-2). The script in the line copies the entire list and traverses through it removing the rows that contain the matching elements.
* Lines 98 to 107 execute when the user selects s. They allow the user to save the data in-memory to file and overwrite any existing data in the text file. The program then prints a line of code that notifies the user that their data has been saved.
* Lines 109 and 110 execute when a user selects something other than the menu options provided, and loops again to the menu after informing the user of incorrect input.

After that I ensured the program ran correctly in the Spyder console (see Figure 9). I then confirmed the data saved correctly in the CDInventory.txt file (see Figure 10). I ran a last check in the Anaconda Prompt (see Figure 11) and made sure the last table entry saved correctly to file (see Figure 12).

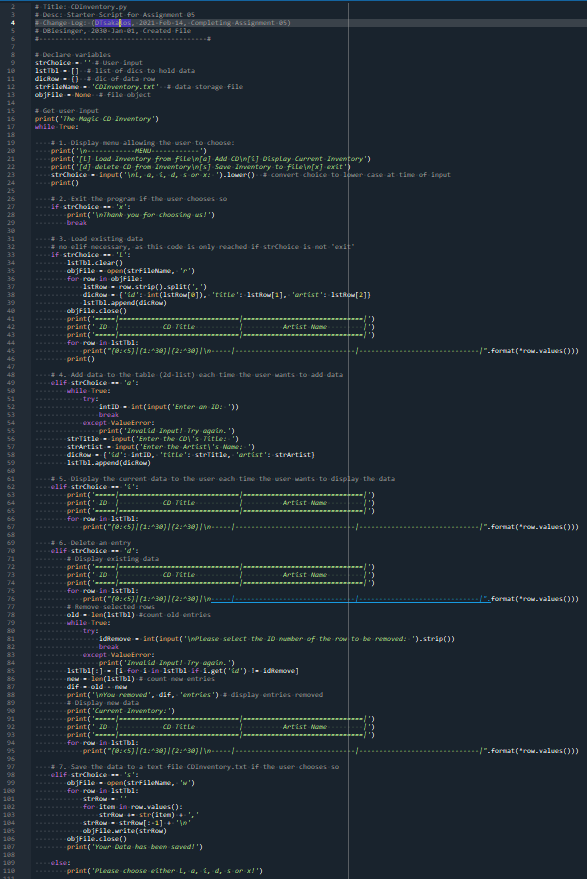
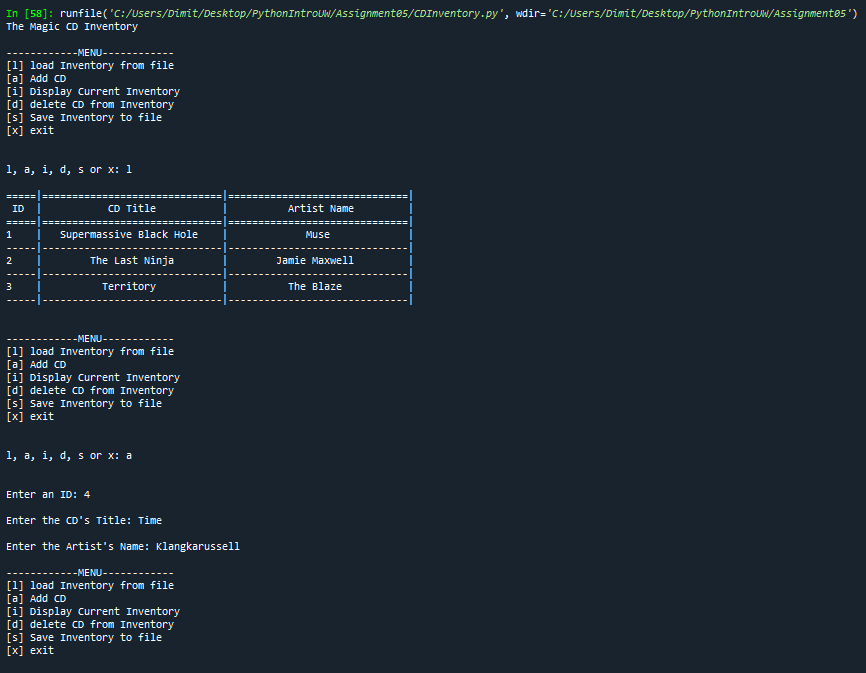


Figure 8 - CDInventory.py code



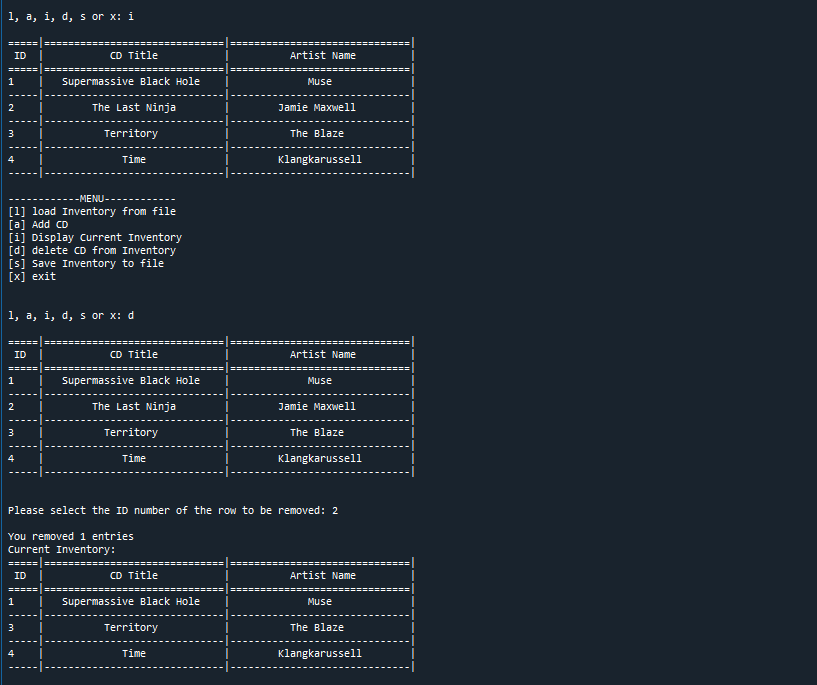




Figure 9 - CDInventory.py console

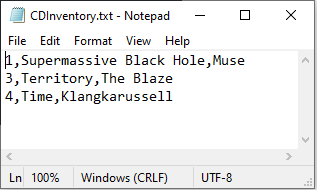
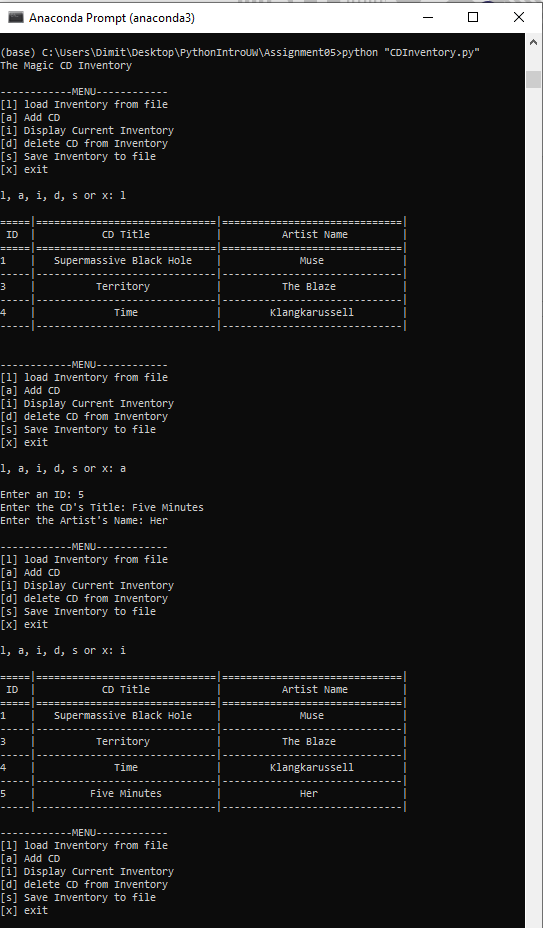


Figure 10 - CDInventory.txt saving in correct format



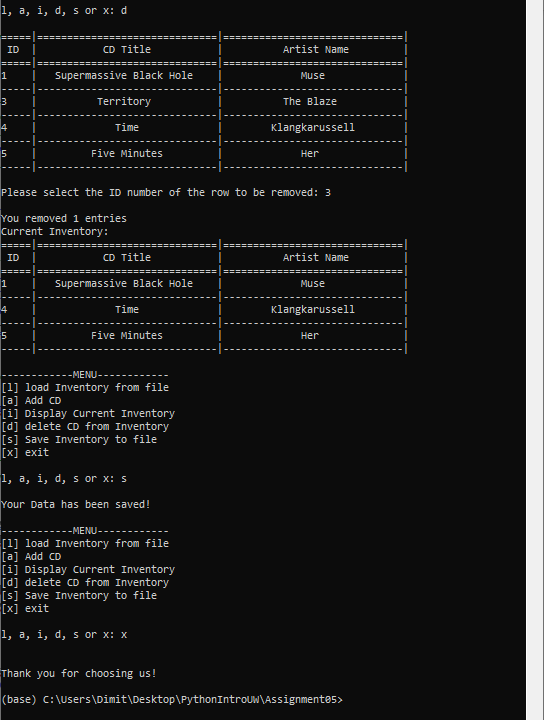


Figure 1 - CDInventory.py running in Anaconda Prompt

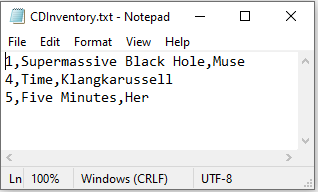


Figure 2 - CDInventory.txt saving in correct format

# Summary

After reviewing the material and going through the Labs, I worked on Assignment 05. While it seemed a bit easier than Assignment 04, it is the most lines of code that I have written. I enjoyed the challenge on working on some other person’s code, and constantly was thinking how much to modify the existing code. Trying to strike a balance must be something programmers often think about. It was also interesting to see how cumbersome and repetitive code can get without using the def for defining functions. Even if the code is simple and repetitive it seems that more lines of code make it more difficult to read. At the same time, if code is too condensed it is hard for it to be legible. It is another instance of balance that must be struck. Also, I have changed the way I view comments and empty space in code. I now consider it of equal importance and try and make use of it to help with organizing my code in a document. The hardest two parts of the assignment were deleting a dictionary based on a value and counting how many entries a user removed. The internet helped with the first solution and simple thinking helped with the second.

# Appendix

## Listing Github

<https://github.com/tsakdimi/Assignment_05>

## 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 | *#------------------------------------------#*  *# Title: CDInventory.py*  *# Desc: Starter Script for Assignment 05*  *# Change Log: (DTsakalos, 2021-Feb-14, Completing Assignment 05)*  *# DBiesinger, 2030-Jan-01, Created File*  *#------------------------------------------#*  *# Declare variables*  strChoice = '' *# User input*  lstTbl = [] *# list of dics to hold data*  dicRow = {} *# dic of data row*  strFileName = 'CDInventory.txt' *# data storage file*  objFile = None *# file object*  *# Get user Input*  **print**('The Magic CD Inventory')  **while** True:  *# 1. Display menu allowing the user to choose:*  **print**('\n------------MENU------------')  **print**('[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')  strChoice = input('\nl, a, i, d, s or x: ').lower() *# convert choice to lower case at time of input*  **print**()  *# 2. Exit the program if the user chooses so*  **if** strChoice == 'x':  **print**('\nThank you for choosing us!')  **break**  *# 3. Load existing data*  *# no elif necessary, as this code is only reached if strChoice is not 'exit'*  **if** strChoice == 'l':  lstTbl.clear()  objFile = open(strFileName, 'r')  **for** row **in** objFile:  lstRow = row.strip().split(',')  dicRow = {'id': int(lstRow[0]), 'title': lstRow[1], 'artist': lstRow[2]}  lstTbl.append(dicRow)  objFile.close()  **print**('=====|==============================|==============================|')  **print**(' ID | CD Title | Artist Name |')  **print**('=====|==============================|==============================|')  **for** row **in** lstTbl:  **print**("{0:<5}|{1:^30}|{2:^30}|\n-----|------------------------------|------------------------------|".format(\*row.values()))  **print**()  *# 4. Add data to the table (2d-list) each time the user wants to add data*  **elif** strChoice == 'a':  **while** True:  **try**:  intID = int(input('Enter an ID: '))  **break**  **except** ValueError:  **print**('Invalid Input! Try again.')  strTitle = input('Enter the CD\'s Title: ')  strArtist = input('Enter the Artist\'s Name: ')  dicRow = {'id': intID, 'title': strTitle, 'artist': strArtist}  lstTbl.append(dicRow)  *# 5. Display the current data to the user each time the user wants to display the data*  **elif** strChoice == 'i':  **print**('=====|==============================|==============================|')  **print**(' ID | CD Title | Artist Name |')  **print**('=====|==============================|==============================|')  **for** row **in** lstTbl:  **print**("{0:<5}|{1:^30}|{2:^30}|\n-----|------------------------------|------------------------------|".format(\*row.values()))  *# 6. Delete an entry*  **elif** strChoice == 'd':  *# Display existing data*  **print**('=====|==============================|==============================|')  **print**(' ID | CD Title | Artist Name |')  **print**('=====|==============================|==============================|')  **for** row **in** lstTbl:  **print**("{0:<5}|{1:^30}|{2:^30}|\n-----|------------------------------|------------------------------|".format(\*row.values()))  *# Remove selected rows*  old = len(lstTbl) *#count old entries*  **while** True:  **try**:  idRemove = int(input('\nPlease select the ID number of the row to be removed: ').strip())  **break**  **except** ValueError:  **print**('Invalid Input! Try again.')  lstTbl[:] = [i **for** i **in** lstTbl **if** i.get('id') != idRemove]  new = len(lstTbl) *# count new entries*  dif = old - new  **print**('\nYou removed', dif, 'entries') *# display entries removed*  *# Display new data*  **print**('Current Inventory:')  **print**('=====|==============================|==============================|')  **print**(' ID | CD Title | Artist Name |')  **print**('=====|==============================|==============================|')  **for** row **in** lstTbl:  **print**("{0:<5}|{1:^30}|{2:^30}|\n-----|------------------------------|------------------------------|".format(\*row.values()))  *# 7. Save the data to a text file CDInventory.txt if the user chooses so*  **elif** strChoice == 's':  objFile = open(strFileName, 'w')  **for** row **in** lstTbl:  strRow = ''  **for** item **in** row.values():  strRow += str(item) + ','  strRow = strRow[:-1] + '\n'  objFile.write(strRow)  objFile.close()  **print**('Your Data has been saved!')  **else**:  **print**('Please choose either l, a, i, d, s or x!') |

1. Retrieved 2021-Feb-14 [↑](#footnote-ref-1)
2. Retrieved 2021-Feb-14 [↑](#footnote-ref-2)