**Kasisi Harris**

**2022-Aug-7**

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

**Assignment\_05**

**GitHub link:** [kasisi79/Assignment\_05 (github.com)](https://github.com/kasisi79/Assignment_05)

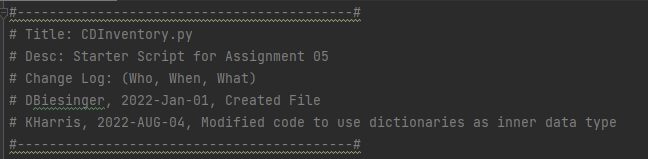
Dictionaries

# Introduction

This assignment focused on the use of dictionaries. As a storage tool, dictionaries are different than their list counterparts as the user does not reference data in the same way. This repository relies on the combination of Keys and Values to store and reference data. This week the class was tasked with improving the CDInventory.py code introduced during Week 4, to incorporate dictionaries in lieu of 2D lists for data storage.

# Script Creation in PyCharm

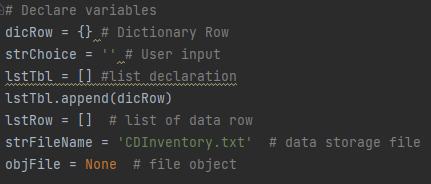
Starting with the CDInventory\_Starter code supplied with the assignment, I revised the header to show my additions of code under the directing pseudo code presented.



Listing - CDInvnentory.py - Header

## Declaring Variables

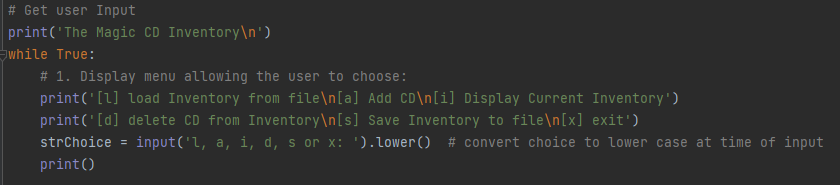
Upon completion of the header updates, I began declaring the variables necessary to complete this task. This establishes the List Table that will house the dictionaries and presents variables that allow the manipulation of rows, file names, and appendixes to row within the table. See below for declarations:



Listing - CDInventory - Variable Declaration

## Processing Data

The bulk of the script revolves around six choices that can be entered by the user: loading existing file data, adding file data, displaying file data, deleting file data, saving file data, and exiting the program. Each section will be highlighted below.



Listing - CDInventory.py - User Input

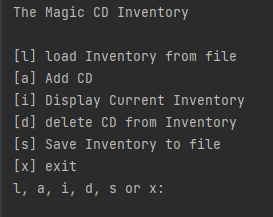
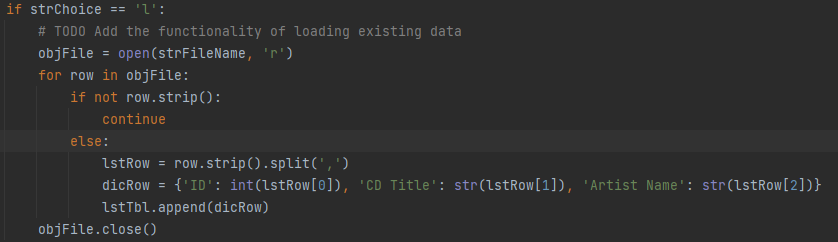


Figure - User Input - PyCharm

### Loading Inventory from File

Should the user choose to, (by entering “l”), they can load any data that has been previously saved to the text file CDInventory.txt. This loaded data can be displayed, new entries added, and previous entries deleted. In my particular solution, I encountered a blank row starts my data, as seen in Figure 3. In order to delete this row, incorporate an “if not” statement to delete the empty row and facilitate the smooth reading of the file for the user.



Listing - CDInventory.py - Loading Data

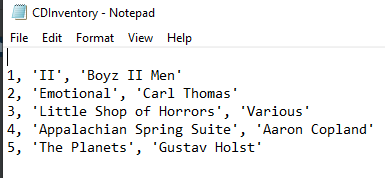
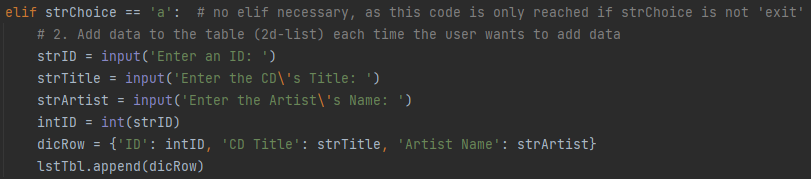


Figure - CDInventory.txt - Prior Data

### Adding Data

Should the user choose to add data to the CD Inventory, they are prompted to enter three pieces of information: ID number, CD Title, and Artist Name. These pieces areas will serve as the Keys to our dictionary, and the data entered by the user will be the Values associated with those keys.



Listing - CDInventory.py - Adding Data

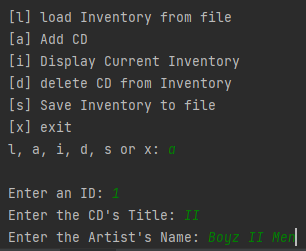


Figure - Adding Data

Combining the Values displayed in Listing 5 with the code in figure 4 we can surmise that for the first row the keys, ‘ID’, ‘CD Title’, and ‘Artist Name’, pertain to the values, ‘1’, ‘II’, and ‘Boyz II Men.’ The entry of data will continue as long as the user selects the ‘a’ option when prompted.

### Displaying Current Data

Should the user choose to display current data, the code prints rows in memory via a FOR loop and displays the for the reader. In my particular code, I chose only to display the dictionaries Values, as this holds the most meaning to the user.

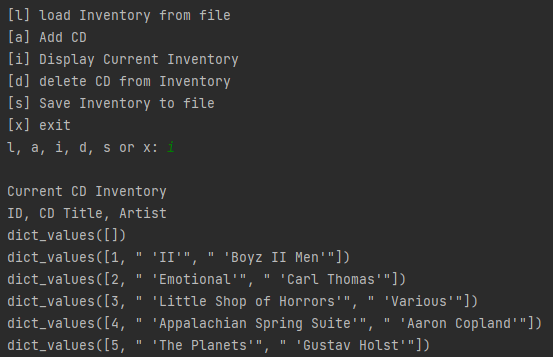
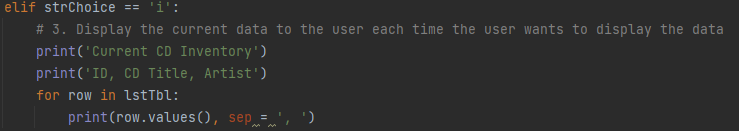


Figure - Display of Current Data

The coding to achieve this utilized a FOR loop that displayed to the user every row within our variable lstTbl which houses all the dictionary information.



Listing - CDInventory.py - Displaying Data

### Deleting Data

In the event that the use wants to delete data, they need only select “d” when prompted. Once selected the code will display the current CD inventory and ask the use to select the ID for the row that they would like deleted.

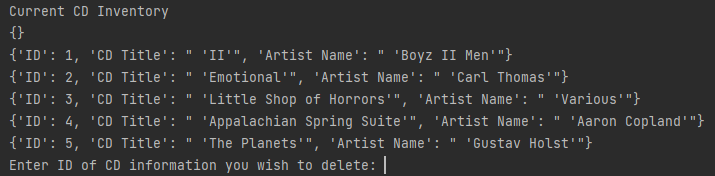


Figure - Displaying Options to Delete

In this example, I’ll select the last entry, ID = 5. Once entered, a message will be displayed that the requested info was deleted.

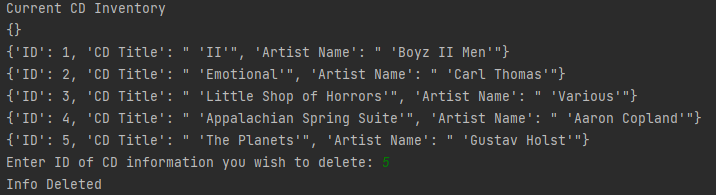


Figure - Selecting ID to Delete

Displaying the inventory again, we will see that the 5th entry in our data has been deleted from memory.

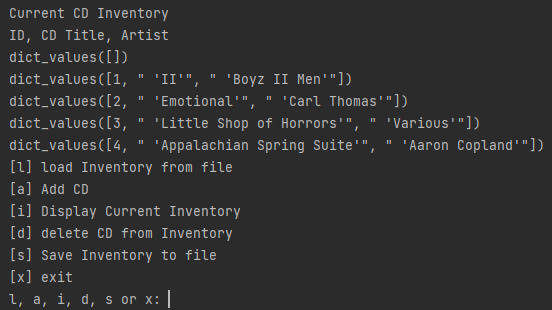
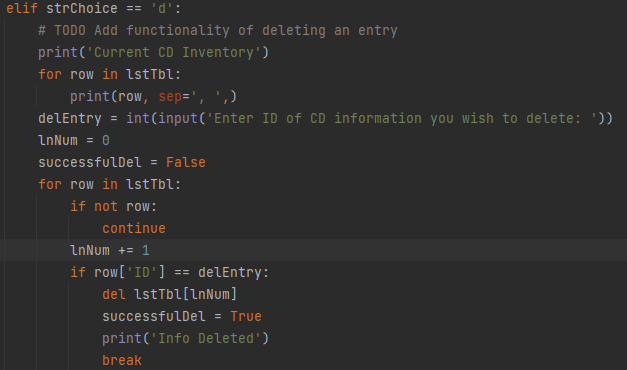


Figure - Data Deleted

Perhaps the hardest section to code, the delete functionality focuses on three things: identifying what information can be deleted, understanding what row of data the user is interested in deleting, and confirming for the user the information has been deleted. In order to achieve the first item, a print statement shows all rows capable of being deleted. I use the *lnNum* variable to keep track of the rows that are currently in our inventory by incrementing its count within the FOR loop. Finally, a print statement “Info Deleted” is presented once a line of code has successfully been deleted.



Listing - CDInventory.py - Deleting Data

### Saving Data

In order to keep track of their progress, the user may elect to save the inventory thus far. To do this, they need only select “s” when prompted. Once selected, the program will display what information has been saved to the CDInventory.txt file as a way to confirm what the user expected.

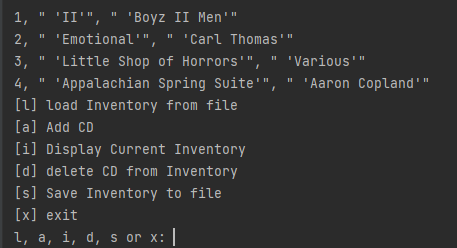


Figure - Saving Data

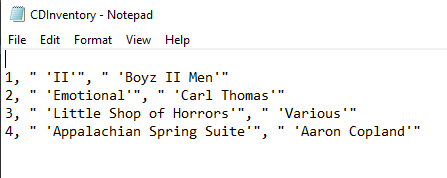
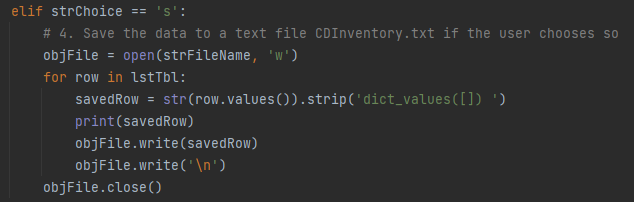


Figure - CDInventory.txt - Saved Data

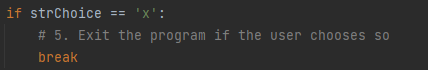
The coding for this portion is straight forward and simply writes the rows within *lstTbl* to our CDInventory.txt file. I elected to only save dictionary Values, as these entries can be assigned to Keys upon being loaded into the program.



Listing - CDInventory.py - Saving Data

### Exiting the Program

Perhaps the easiest selection is exiting the program altogether which can be achieved by entering “x” when prompted.



Listing - CDInventory.py - Exiting the Program

# Terminal Run Images

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

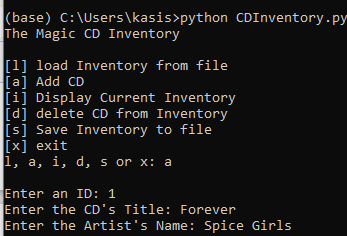


Figure - Terminal Run - Adding Data 1

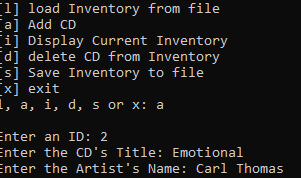


Figure - Terminal Run - Adding Data 2

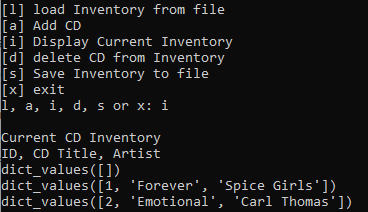


Figure - Terminal Run - Displaying Data

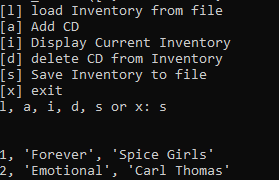


Figure - Terminal Run - Saving Data

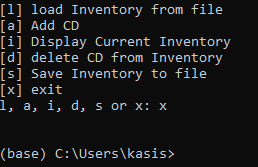


Figure - Terminal Run - Exiting the Program

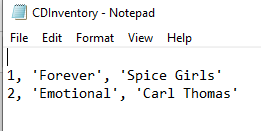


Figure - Saved Data to TXT file

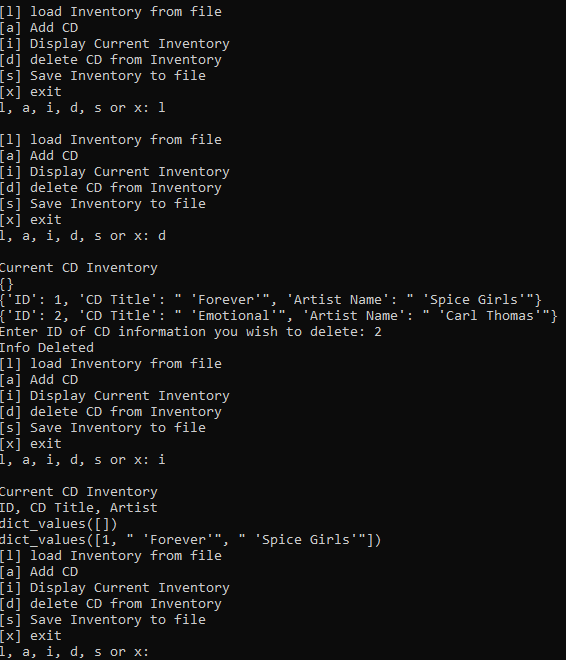


Figure - Terminal Run - Deleting Information

# Summary

While the concept of switching storage methods may seem simple, I found it to be quite challenging to incorporate the dictionaries as repositories. I did learn the importance of incorporating ways to clean data, (i.e. removing empty rows), prior to loading and manipulating data. I’m sure there is a better way to achieve the results. However, I am excited that I was able to tackle the task.

# Appendix

## Listing of CDInventory.py

*#------------------------------------------#*

*# Title: CDInventory.py*

*# Desc: Starter Script for Assignment 05*

*# Change Log: (Who, When, What)*

*# DBiesinger, 2022-Jan-01, Created File*

*# KHarris, 2022-AUG-04, Modified code to use dictionaries as inner data type*

*#------------------------------------------#*

*# Declare variables*

dicRow = {} *# Dictionary Row*

strChoice = '' *# User input*

lstTbl = [] *#list declaration*

lstTbl.append(dicRow)

lstRow = [] *# list of data row*

strFileName = 'CDInventory.txt' *# data storage file*

objFile = **None** *# file object*

*# Get user Input*

print('The Magic CD Inventory**\n**')

**while** **True**:

*# 1. Display menu allowing the user to choose:*

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('l, a, i, d, s or x: ').lower() *# convert choice to lower case at time of input*

print()

**if** strChoice == 'x':

*# 5. Exit the program if the user chooses so*

**break**

**if** strChoice == 'l':

*# TODO Add the functionality of loading existing data*

objFile = open(strFileName, 'r')

**for** row **in** objFile:

**if** **not** row.strip():

**continue**

**else**:

lstRow = row.strip().split(',')

dicRow = {'ID': int(lstRow[0]), 'CD Title': str(lstRow[1]), 'Artist Name': str(lstRow[2])}

lstTbl.append(dicRow)

objFile.close()

*#pass*

**elif** strChoice == 'a': *# no elif necessary, as this code is only reached if strChoice is not 'exit'*

*# 2. Add data to the table (2d-list) each time the user wants to add data*

strID = input('Enter an ID: ')

strTitle = input('Enter the CD**\'**s Title: ')

strArtist = input('Enter the Artist**\'**s Name: ')

intID = int(strID)

dicRow = {'ID': intID, 'CD Title': strTitle, 'Artist Name': strArtist}

lstTbl.append(dicRow)

**elif** strChoice == 'i':

*# 3. Display the current data to the user each time the user wants to display the data*

print('Current CD Inventory')

print('ID, CD Title, Artist')

**for** row **in** lstTbl:

print(row.values(), sep = ', ')

**elif** strChoice == 'd':

*# TODO Add functionality of deleting an entry*

print('Current CD Inventory')

**for** row **in** lstTbl:

print(row, sep=', ',)

delEntry = int(input('Enter ID of CD information you wish to delete: '))

lnNum = 0

successfulDel = **False**

**for** row **in** lstTbl:

**if** **not** row:

**continue**

lnNum += 1

**if** row['ID'] == delEntry:

**del** lstTbl[lnNum]

successfulDel = **True**

print('Info Deleted')

**break**

**elif** strChoice == 's':

*# 4. Save the data to a text file CDInventory.txt if the user chooses so*

objFile = open(strFileName, 'w')

**for** row **in** lstTbl:

savedRow = str(row.values()).strip('dict\_values([]) ')

print(savedRow)

objFile.write(savedRow)

objFile.write('**\n**')

objFile.close()

**else**:

print('Please choose either l, a, i, d, s or x!')