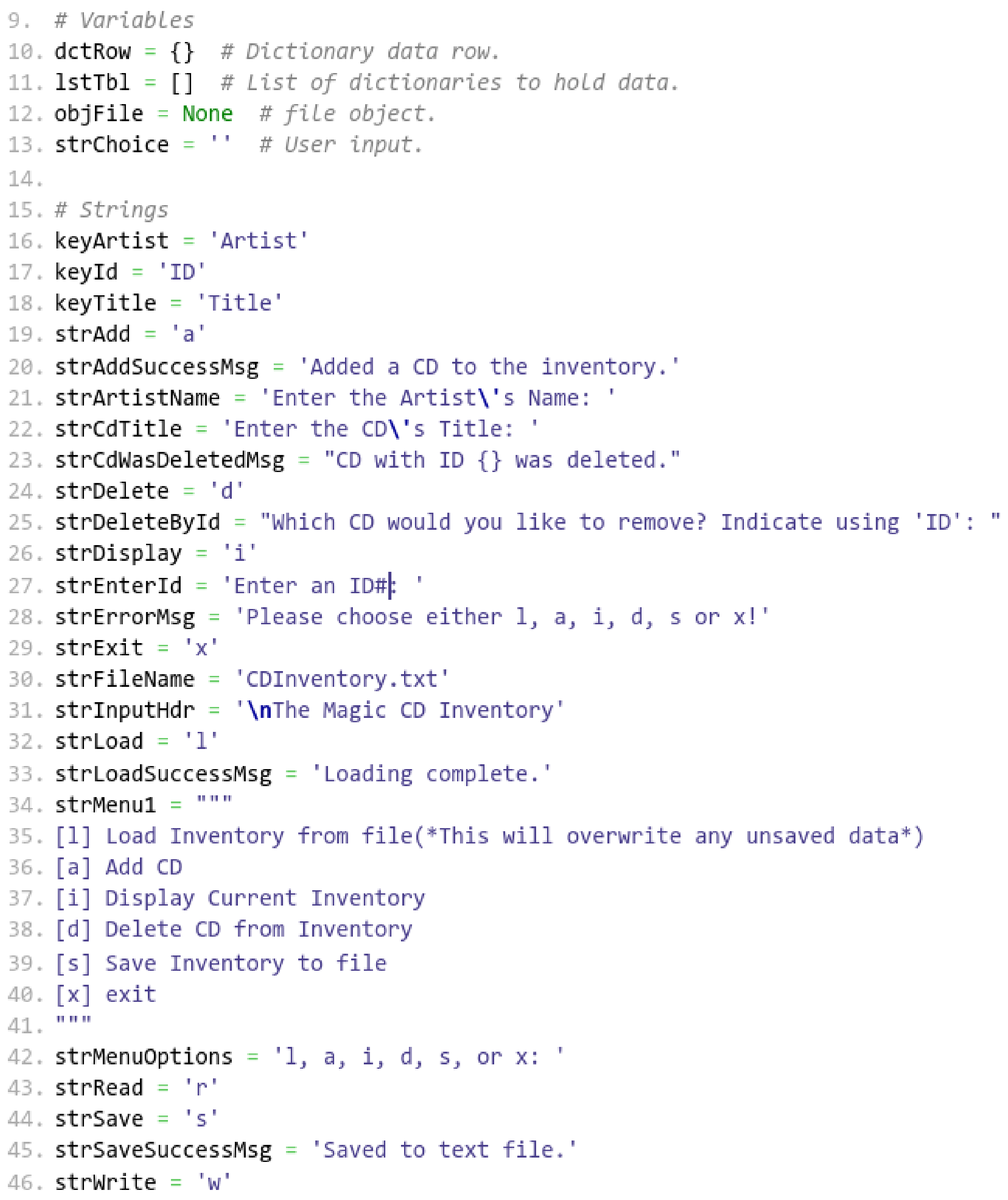
Charles Hodges(hodges11@uw.edu)
August 8, 2021
IT FDN 110: Introduction to Programming (Python)
Link to Assignment in GitHub: https://github.com/chodges11/Assignment_05
Assignment 05

CD Inventory with a Starter Script

Introduction

In this assignment, we were given a starter script of a possible solution to last week's assignment. We were instructed to fill in the missing sections, using the template, and without using functions. The specifics of the assignment were about using lists and dictionaries for the CD Inventory projects, as opposed to just tuples.

Variables and Strings

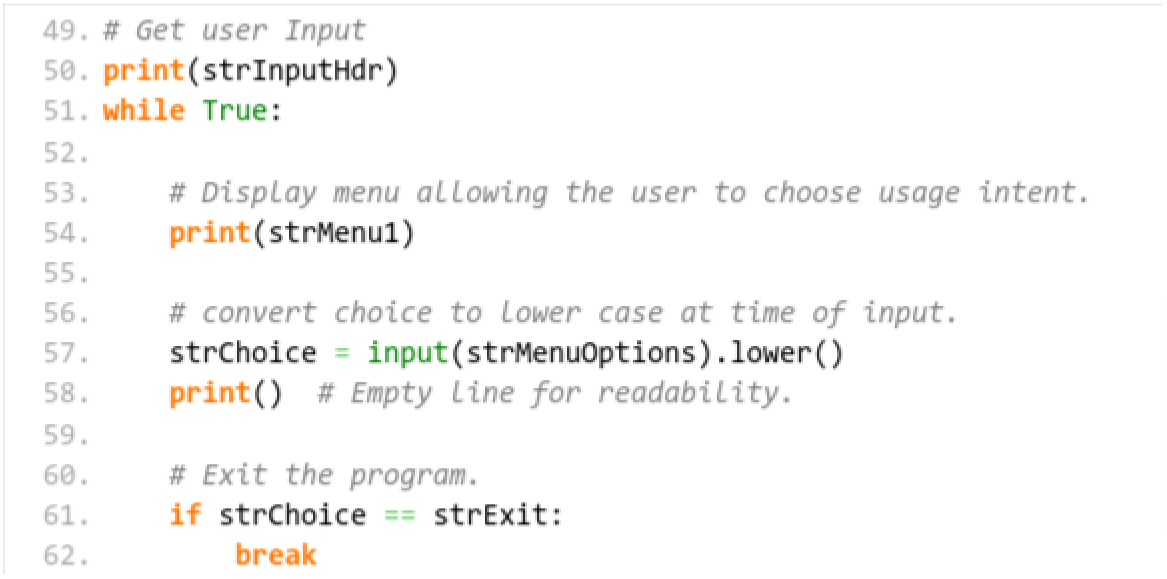


```python
# Variables
dctRow = {}  # Dictionary data row.
lstTbl = []  # List of dictionaries to hold data.
objFile = None  # file object.
strChoice = ''  # User input.

# Strings
keyArtist = 'Artist'
keyId = 'ID'
keyTitle = 'Title'
strAdd = 'a'
strAddSuccessMsg = 'Added a CD to the inventory.'
strArtistName = 'Enter the Artist\'s Name: '
strCdTitle = 'Enter the CD\'s Title: '
strCdWasDeletedMsg = "CD with ID {} was deleted."
strDelete = 'd'
strDeleteById = "Which CD would you like to remove? Indicate using 'ID': "
strDisplay = 'i'
strEnterId = 'Enter an ID#: '
strErrorMsg = 'Please choose either l, a, i, d, s or x!'
strExit = 'x'
strFileName = 'CDInventory.txt'
strInputHdr = '\nThe Magic CD Inventory'
strLoad = 'l'
strLoadSuccessMsg = 'Loading complete.'
strMenu1 = """
[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit
"""
strMenuOptions = 'l, a, i, d, s, or x: '
strRead = 'r'
strSave = 's'
strSaveSuccessMsg = 'Saved to text file.'
strWrite = 'w'
```

Figure 1 - Strings & Variables

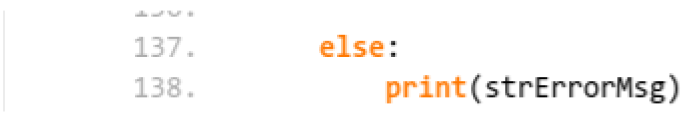
I kept the four previously declared variables where they were, and added all of my strings as well. There are a lot of strings, so making sure their names were useful was important. Also, I alphabetized both the variables and the strings.

Top and Bottom of the Script



```python
# Get user Input
print(strInputHdr)
while True:

    # Display menu allowing the user to choose usage intent.
    print(strMenu1)

    # convert choice to lower case at time of input.
    strChoice = input(strMenuOptions).lower()
    print()  # Empty line for readability.

    # Exit the program.
    if strChoice == strExit:
        break
```

Figure 2 - Top of the script



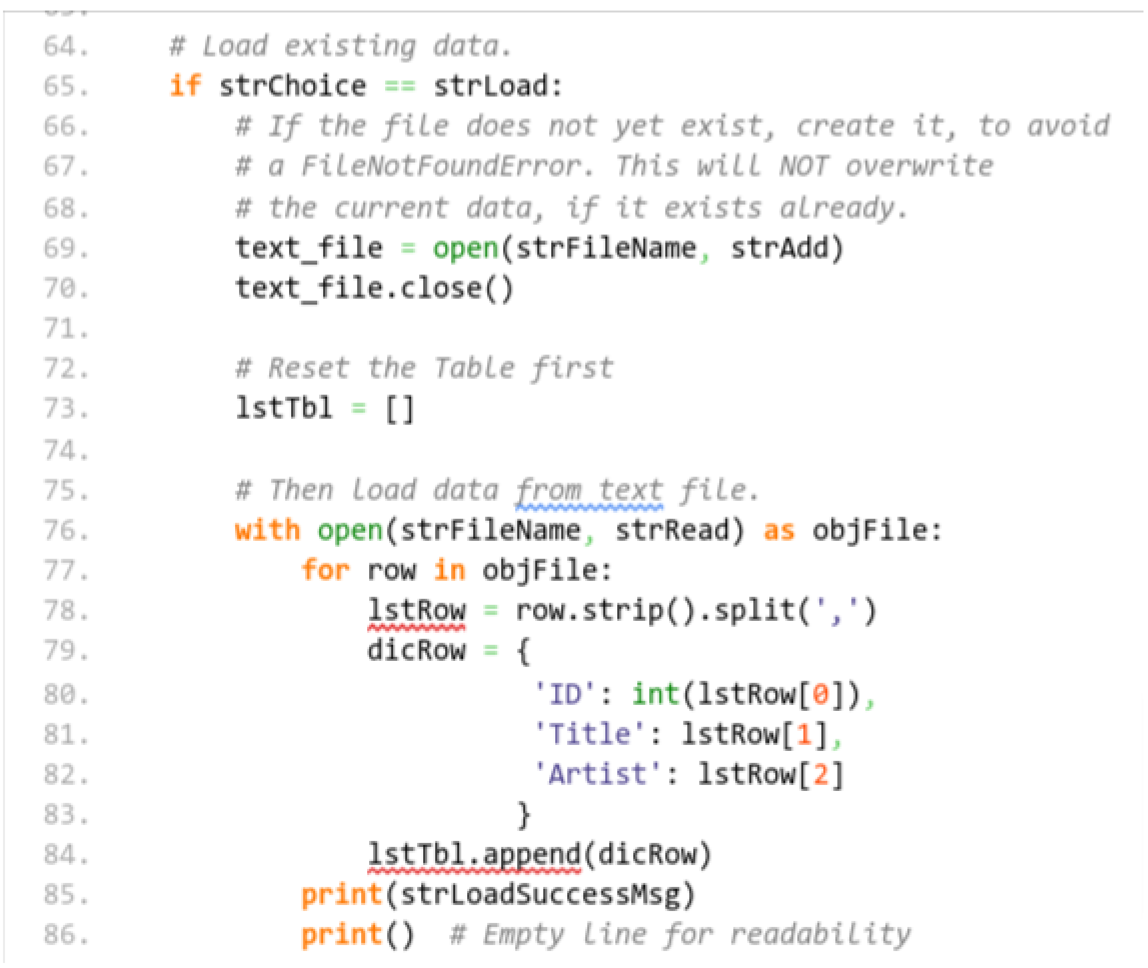
```python
        else:
            print(strErrorMsg)
```

Figure 3 - Bottom of the script

Here we begin the script by printing a header for the top of the menu of options. Next we print the menu, and request input from the user. There's a smidge of error handling here, with the "*lower()*" string method. This method moves the User entered string to lowercase, in case they have Caps Lock on. Also, we give the User a chance to bail out early, by allowing them to input an exit command.

Lastly, we have an else statement at the bottom of this script. This conditional validates that the User enters only one of the expected values for interaction. A pithy error message is provided if the User goes off script(*pun intended!*).

Load



```
    # Load existing data.
    if strChoice == strLoad:
        # If the file does not yet exist, create it, to avoid
        # a FileNotFoundError. This will NOT overwrite
        # the current data, if it exists already.
        text_file = open(strFileName, strAdd)
        text_file.close()

        # Reset the Table first
        lstTbl = []

        # Then load data from text file.
        with open(strFileName, strRead) as objFile:
            for row in objFile:
                lstRow = row.strip().split(',')
                dicRow = {
                          'ID': int(lstRow[0]),
                          'Title': lstRow[1],
                          'Artist': lstRow[2]
                         }
                lstTbl.append(dicRow)
            print(strLoadSuccessMsg)
            print()  # Empty line for readability
```

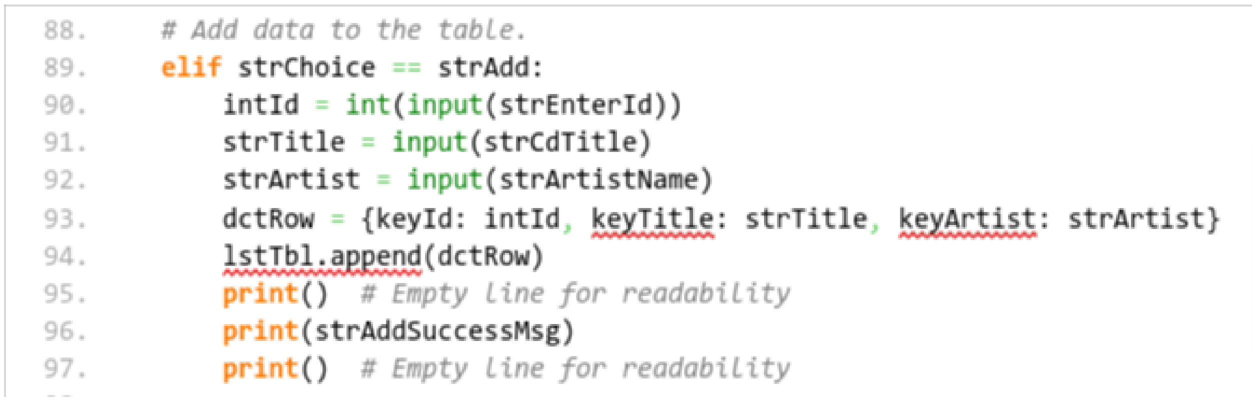
Figure 4 - Load logic

For the Load logic, the script starts by creating the text file. I open it as ‘append’ and immediately close it again. The reason we do this is that if the User enters Load as their selection, upon the initial running of this program, there is a FileNotFoundError, and the program closes. But, ‘append’ is used so that if there is an existing file already, the contents are not overwritten.

Next I reset the 2D Table in memory. If I may take a step back for a second, I do warn the User that selecting load will delete all unsaved changes. The reason why we need to reset the table here is that, if we just load the text file’s contents into memory, we would append the file data to the memory data. That creates unwanted duplicate entries. So the table must be cleared to load the file’s data freshly.

Finally, we open the file and read from it. Then we go row by row and create the list of dictionaries, for each CD. Finally, we printed out a message saying that the Loading task had been completed. There is almost no perceptible time between the loading action and the display of the message, but such messages provide a more delightful user experience. In a larger much Inventory, there may be a perceptible delay.

Add

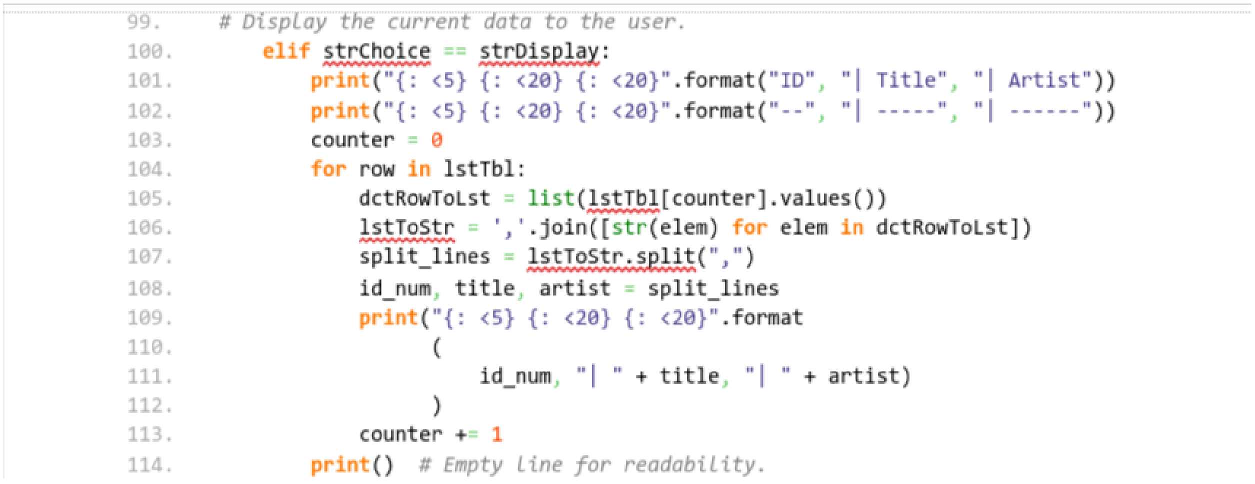


```
    # Add data to the table.
    elif strChoice == strAdd:
        intId = int(input(strEnterId))
        strTitle = input(strCdTitle)
        strArtist = input(strArtistName)
        dctRow = {keyId: intId, keyTitle: strTitle, keyArtist: strArtist}
        lstTbl.append(dctRow)
        print()  # Empty line for readability
        print(strAddSuccessMsg)
        print()  # Empty line for readability
```

Figure 5 - Add logic

For the Add logic, we user is prompted to input the expected information, of ID#, CD Title, and Artist Name. Normally, I would have added a format validation for the ID#, but since there has not been any mention of that being necessary, I skipped it this time. Then we create a dictionary entry with the information, and append that entry to the list Table. FInally, we add another message at the end informing the User of success.

Display

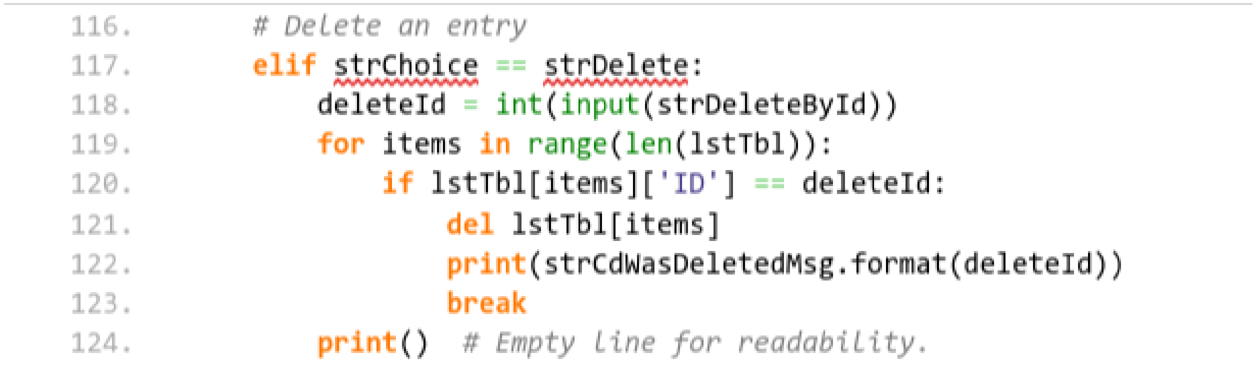


```
    # Display the current data to the user.
    elif strChoice == strDisplay:
        print("{: <5} {: <20} {: <20}".format("ID", "| Title", "| Artist"))
        print("{: <5} {: <20} {: <20}".format("--", "| -----", "| ------"))
        counter = 0
        for row in lstTbl:
            dctRowToLst = list(lstTbl[counter].values())
            lstToStr = ','.join([str(elem) for elem in dctRowToLst])
            split_lines = lstToStr.split(",")
            id_num, title, artist = split_lines
            print("{: <5} {: <20} {: <20}".format
                (
                    id_num, "| " + title, "| " + artist)
                )
            counter += 1
        print()  # Empty line for readability.
```

Figure 6 - Display logic

For the Display logic, I reused some of the formatting code from Assignment 04, for the header and table formatting. Then iterate through the list of dictionaries, convert their type to lists and then to strings, so that their output was the format I wanted. No success message needed here, as the table tells its own story.

Delete

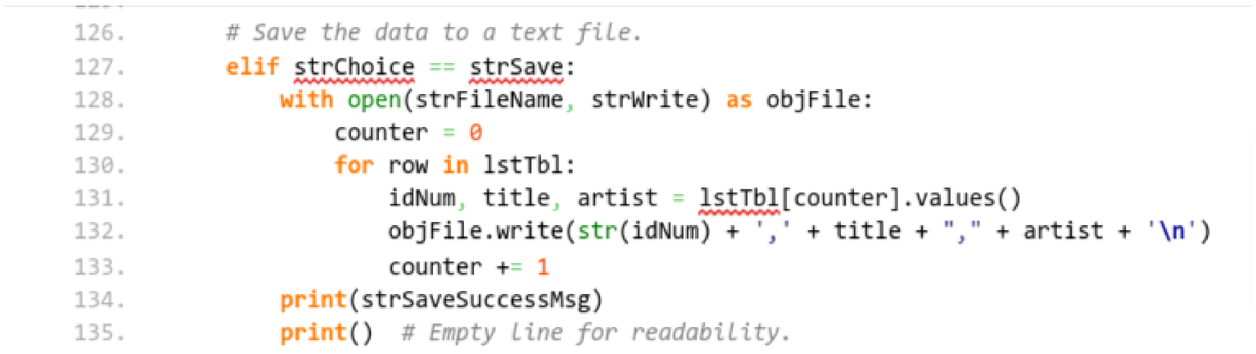


```
        # Delete an entry
        elif strChoice == strDelete:
            deleteId = int(input(strDeleteById))
            for items in range(len(lstTbl)):
                if lstTbl[items]['ID'] == deleteId:
                    del lstTbl[items]
                    print(strCdWasDeletedMsg.format(deleteId))
                    break
            print()  # Empty line for readability.
```

Figure 7 - Delete logic

For the Delete logic, I ask the User to select which entry to input by ID#. Firstly, here is where I found the lack of functions annoying. I would have re-printed the table out, for the User, before they could interact with the delete functionality, so that they could make the best choice. I refused to copy/paste the whole Display logic here, for that purpose. So I hope the User remembers to display the inventory first. I just iterate through the Table, and find the entry which matches the ID# that the User entered. Then I delete the entry and inform the User that it has been done.

Save



```
        # Save the data to a text file.
        elif strChoice == strSave:
            with open(strFileName, strWrite) as objFile:
                counter = 0
                for row in lstTbl:
                    idNum, title, artist = lstTbl[counter].values()
                    objFile.write(str(idNum) + ',' + title + "," + artist + '\n')
                    counter += 1
            print(strSaveSuccessMsg)
            print()  # Empty line for readability.
```

Figure 8 - Save logic

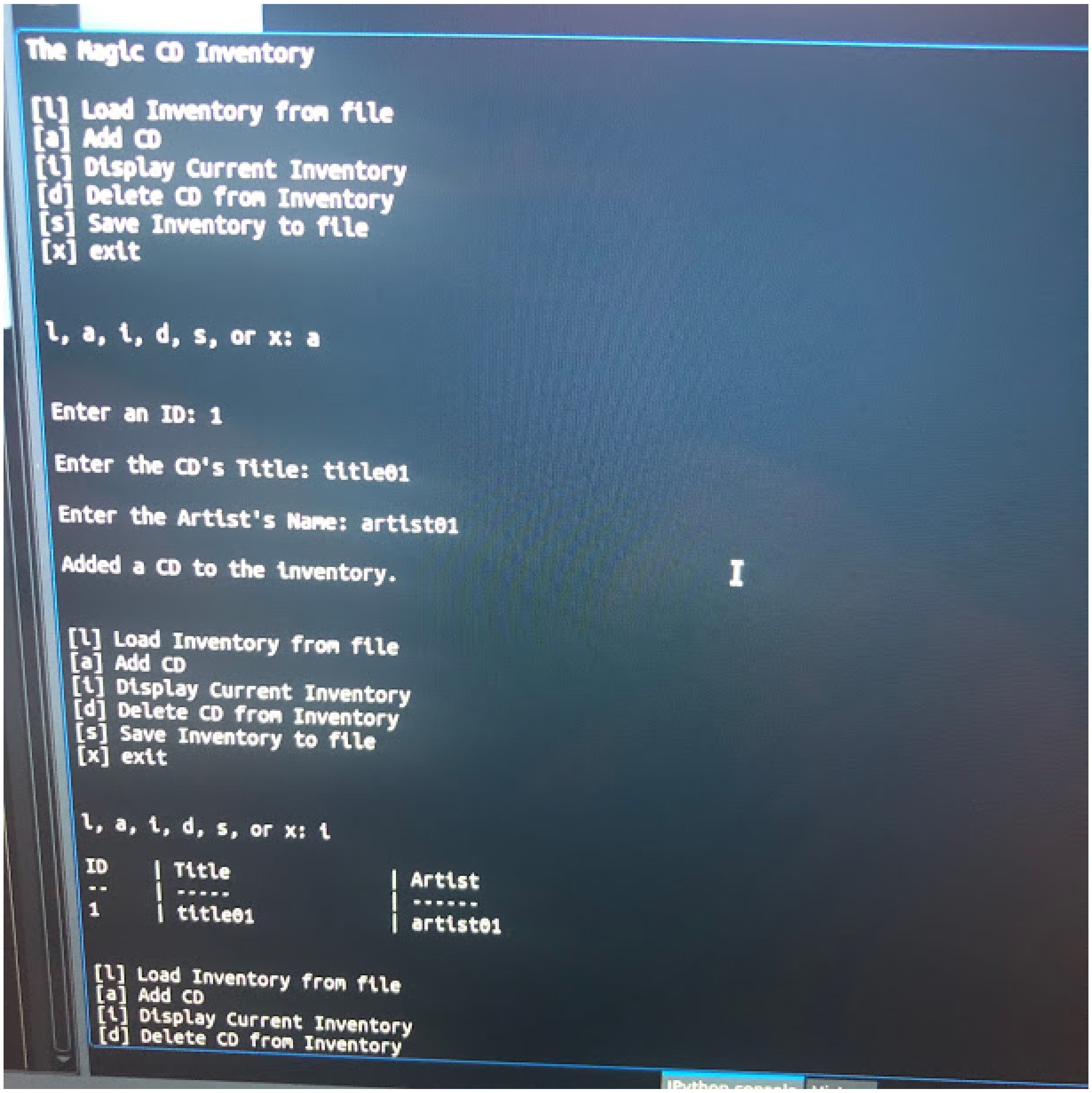
For the Save logic, I had to make sure that I was not saving the Table to the file as rows of dictionaries. Initially I worked for a long time on figuring out how to do that, and I made that work. But the display logic worked better when the Text file only contained the Values, without the keys. So I wasted a bunch of time there, but I learned how to do that, if I need to later. So this logic saves comma separated values to the text file, with each dictionary's values having their own line in the file.

Summary

In this project, we took a sample page of starter code, and was asked to make it work. We were expected to use 2D Tables, which I did this time, and not use functions, which I didn't this time. We upgraded the CD Inventory script from last time, by using lists and dictionaries, instead of tuples.

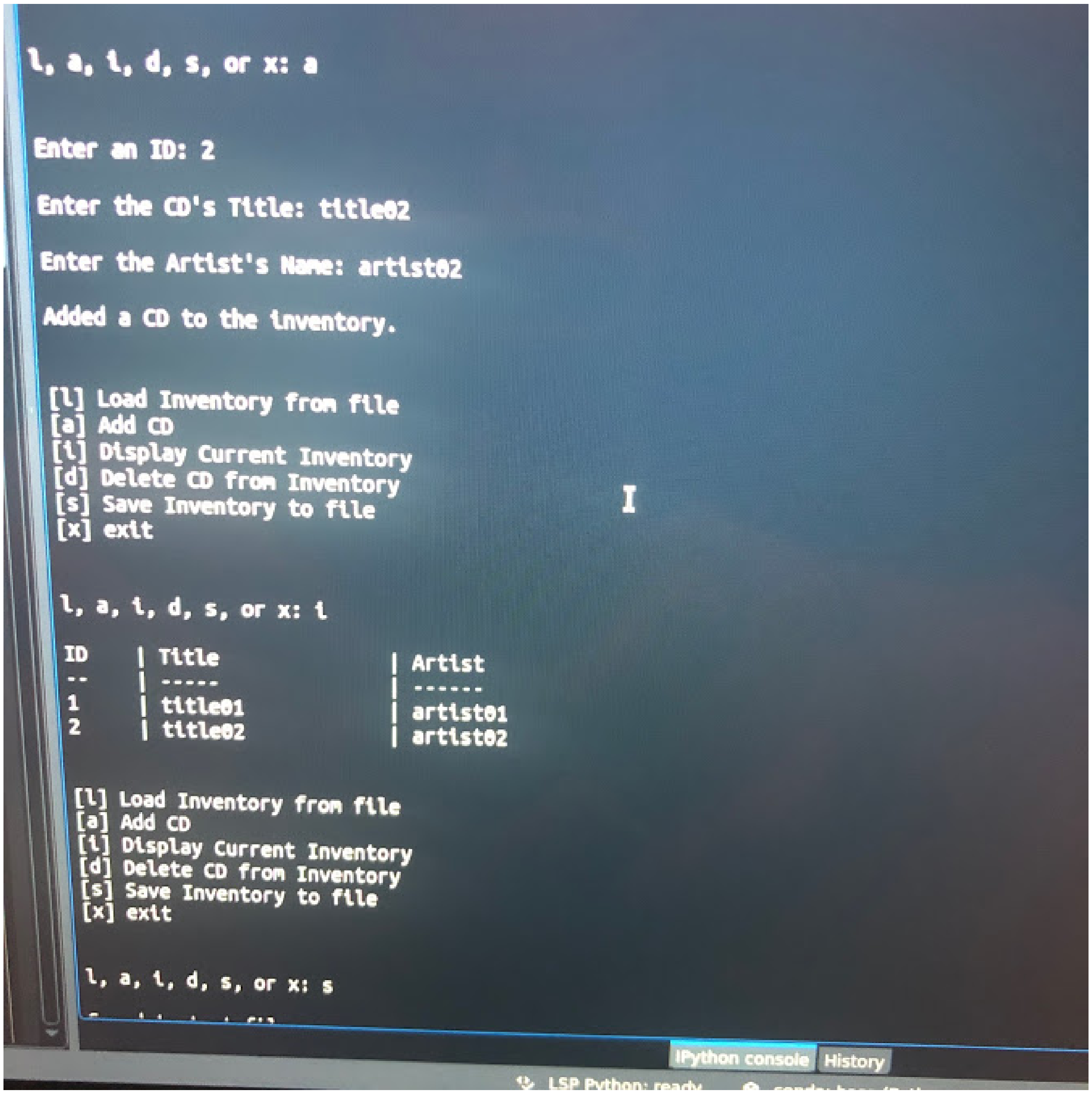
Script Running

Spyder



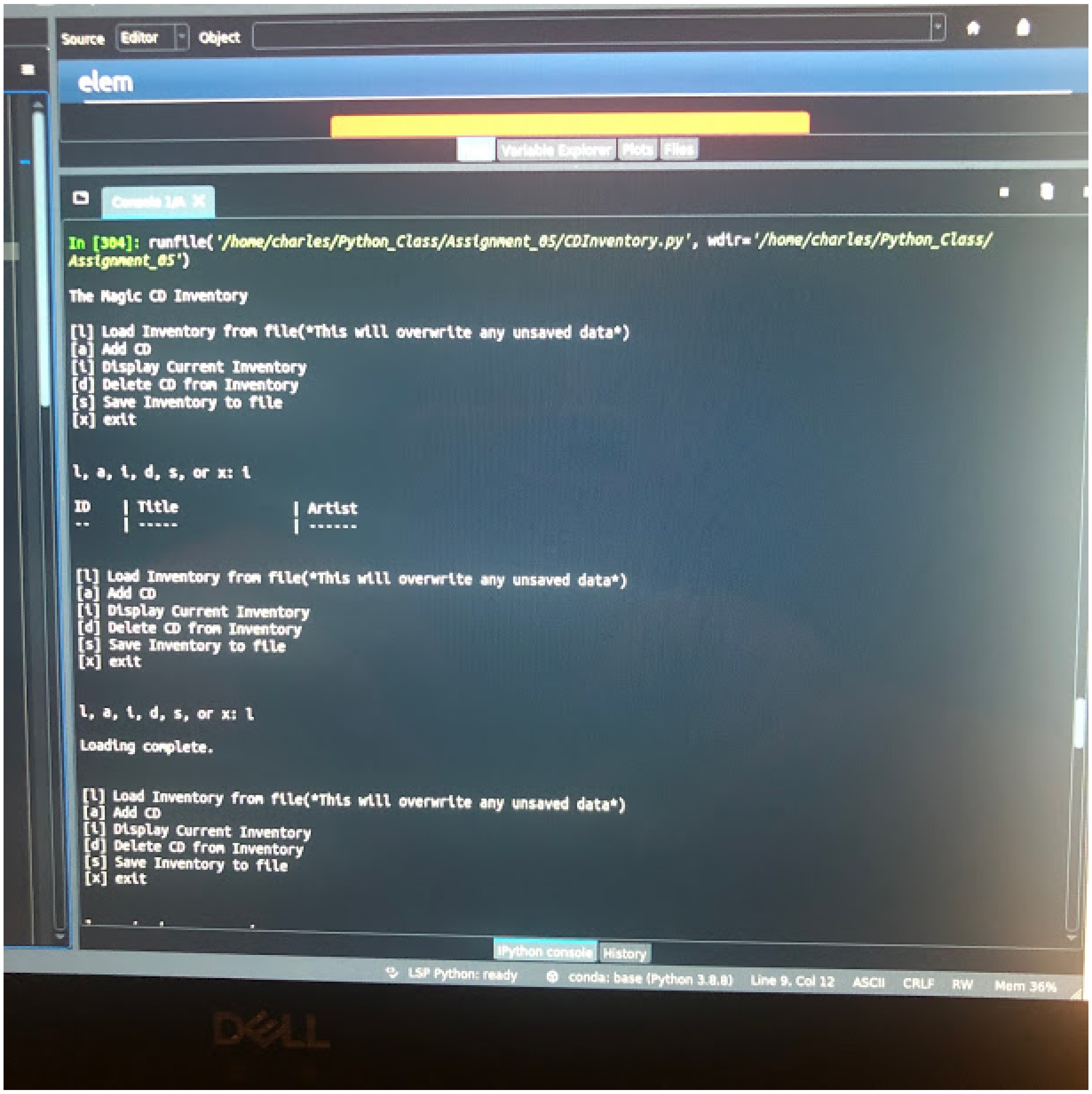
```
The Magic CD Inventory

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: a


Enter an ID: 1

Enter the CD's Title: title01

Enter the Artist's Name: artist01

Added a CD to the inventory.


[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: i
ID      | Title               | Artist
--      | -----               | ------
1       | title01             | artist01

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
```

Figure 9 - Running in Spyder



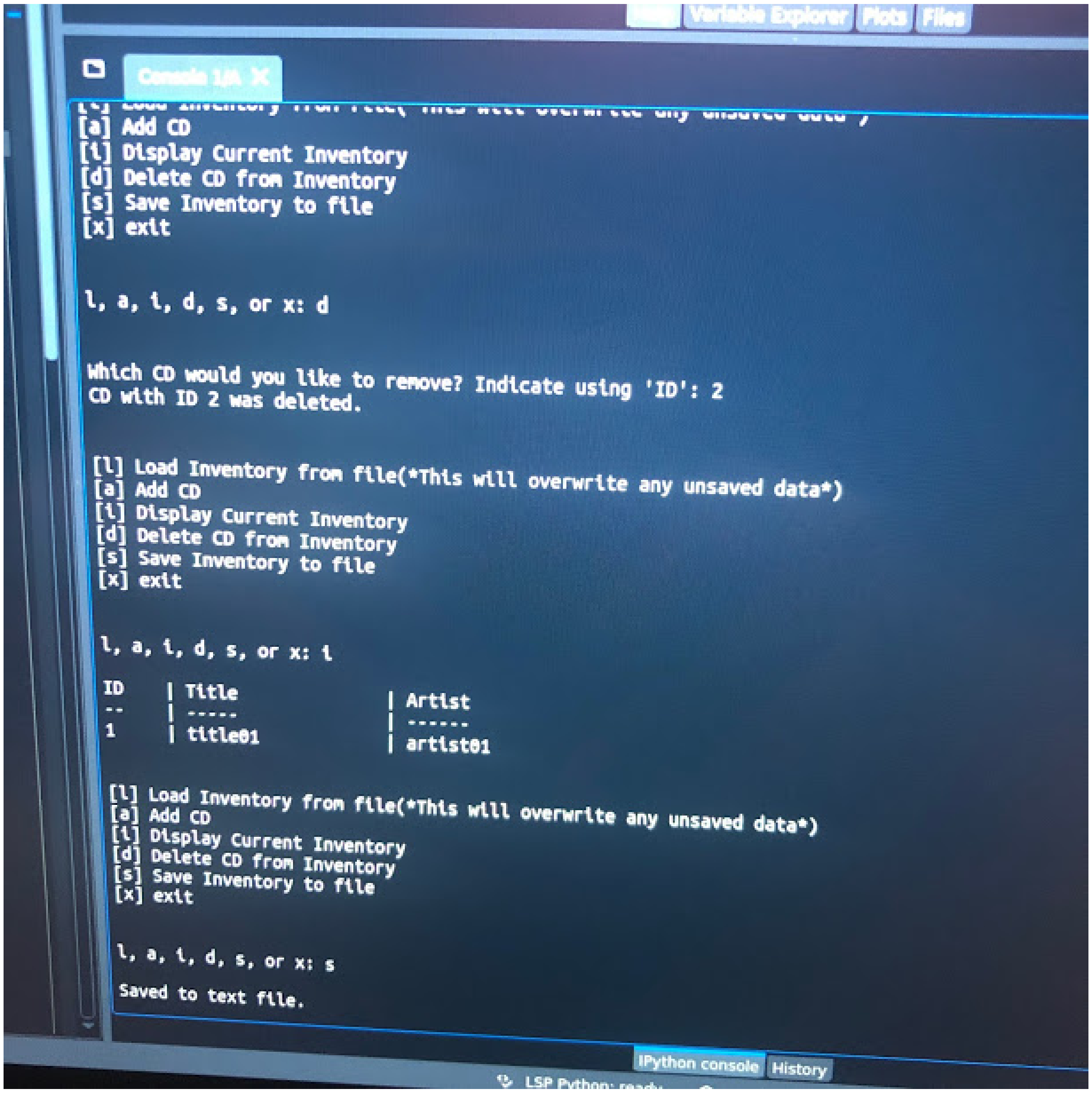
```
l, a, i, d, s, or x: a


Enter an ID: 2

Enter the CD's Title: title02

Enter the Artist's Name: artist02

Added a CD to the inventory.


[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: i

ID     | Title                | Artist
--     | -----                | ------
1      | title01              | artist01
2      | title02              | artist02

[l] Load Inventory from file
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: s
```

Figure 10 - Running in Spyder



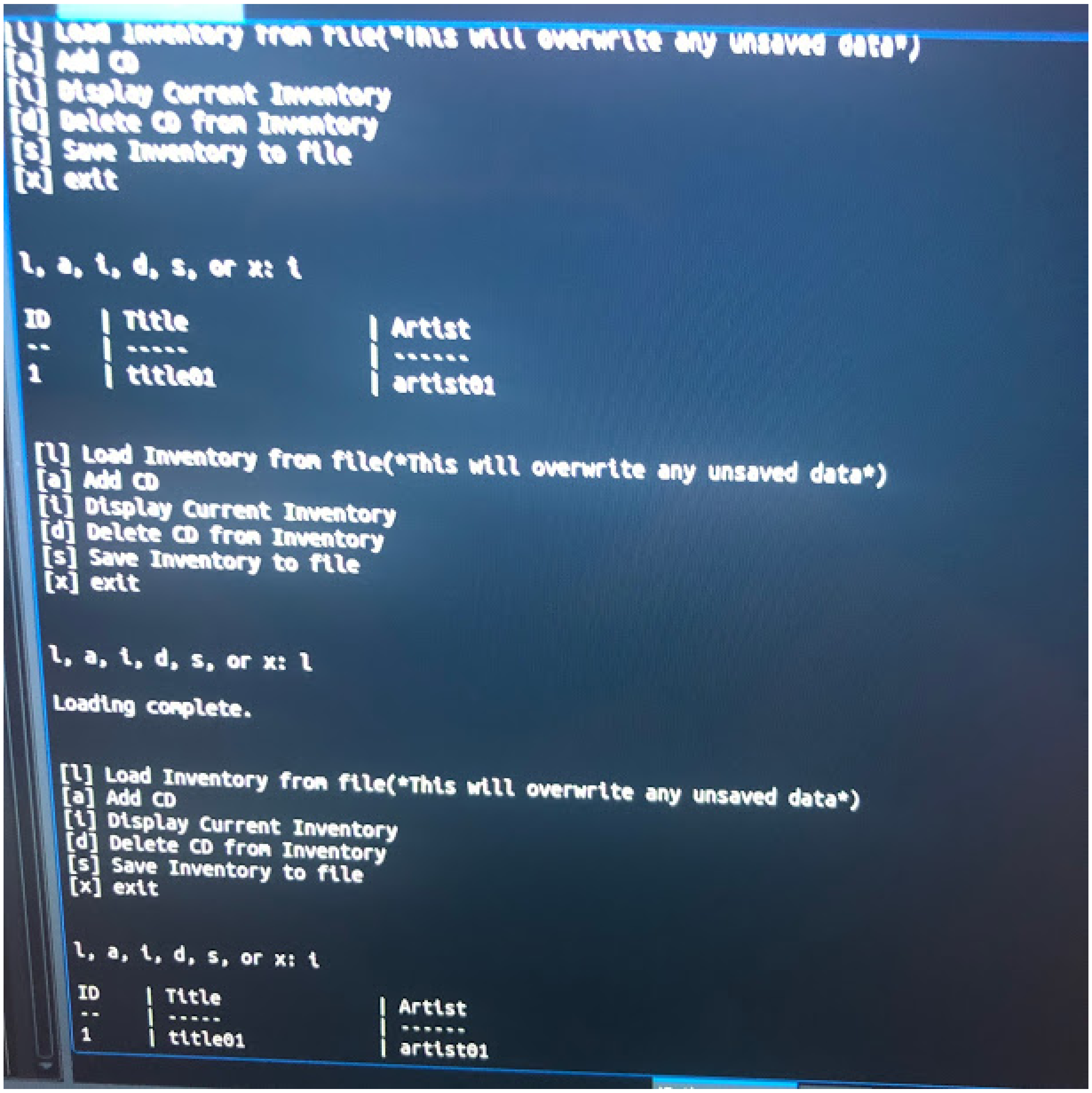
```
In [304]: runfile('/home/charles/Python_Class/Assignment_05/CDInventory.py', wdir='/home/charles/Python_Class/
Assignment_05')

The Magic CD Inventory

[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: i

ID    | Title             | Artist
--    | -----             | ------


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: l

Loading complete.


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit
```

Figure 11 - Running in Spyder



```
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: d


Which CD would you like to remove? Indicate using 'ID': 2
CD with ID 2 was deleted.


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: i

ID     | Title              | Artist
--     | -----              | ------
1      | title01            | artist01

[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: s

Saved to text file.
```

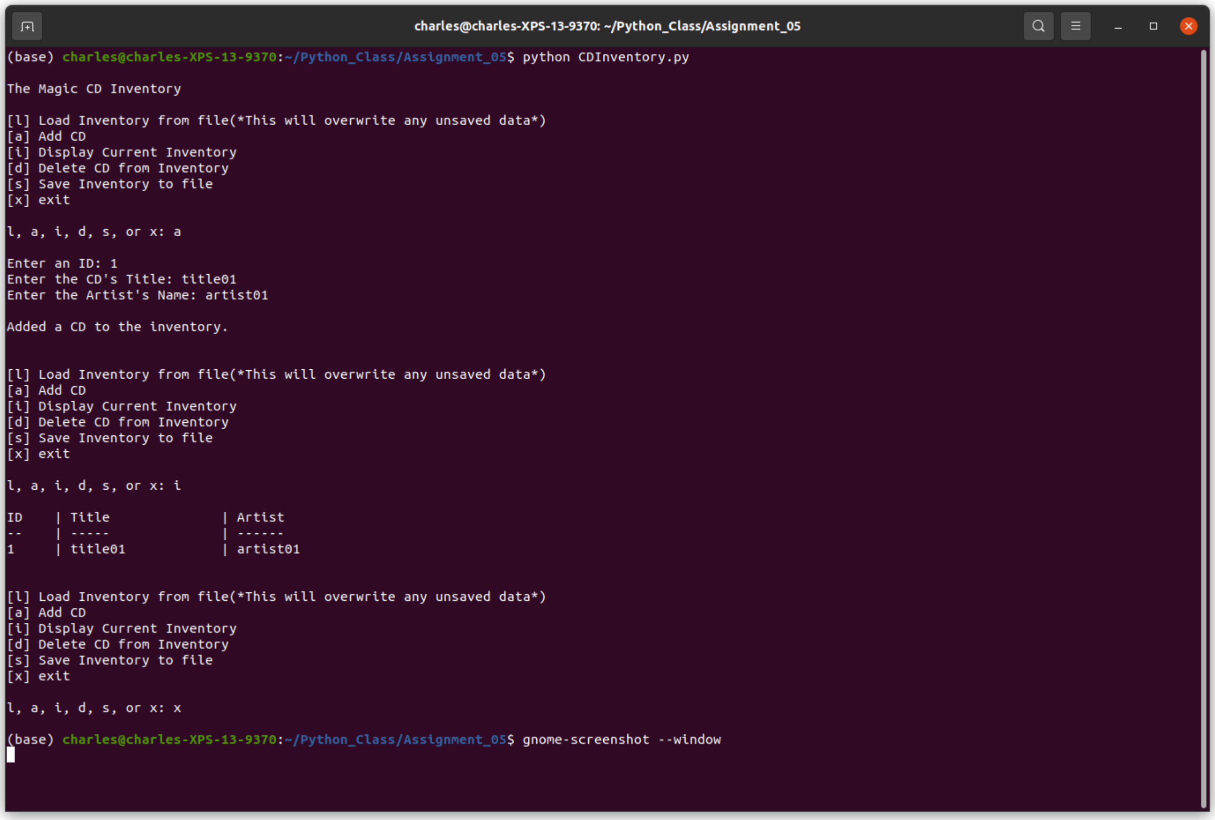
Figure 12 - Running in Spyder



```
[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: i

ID     | Title                | Artist
--     | -----                | ------
1      | title01              | artist01


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: l

Loading complete.


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit


l, a, i, d, s, or x: i

ID     | Title                | Artist
--     | -----                | ------
1      | title01              | artist01
```

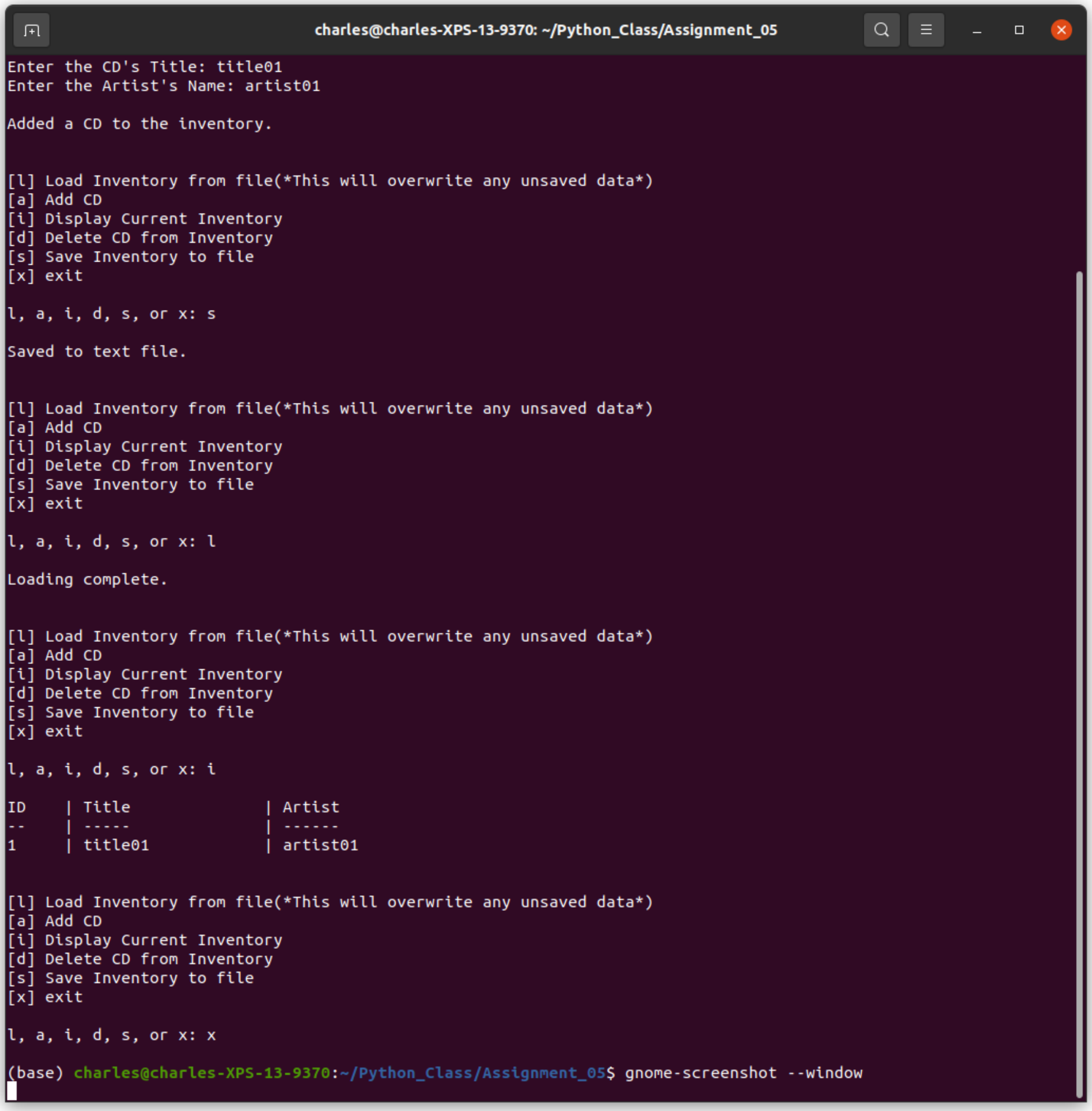
Figure 13 - Running in Spyder

Terminal



```
(base) charles@charles-XPS-13-9370:~/Python_Class/Assignment_05$ python CDInventory.py

The Magic CD Inventory

[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: a

Enter an ID: 1
Enter the CD's Title: title01
Enter the Artist's Name: artist01

Added a CD to the inventory.


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: i

ID      | Title                 | Artist
--      | -----                 | ------
1       | title01               | artist01


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: x

(base) charles@charles-XPS-13-9370:~/Python_Class/Assignment_05$ gnome-screenshot --window
```

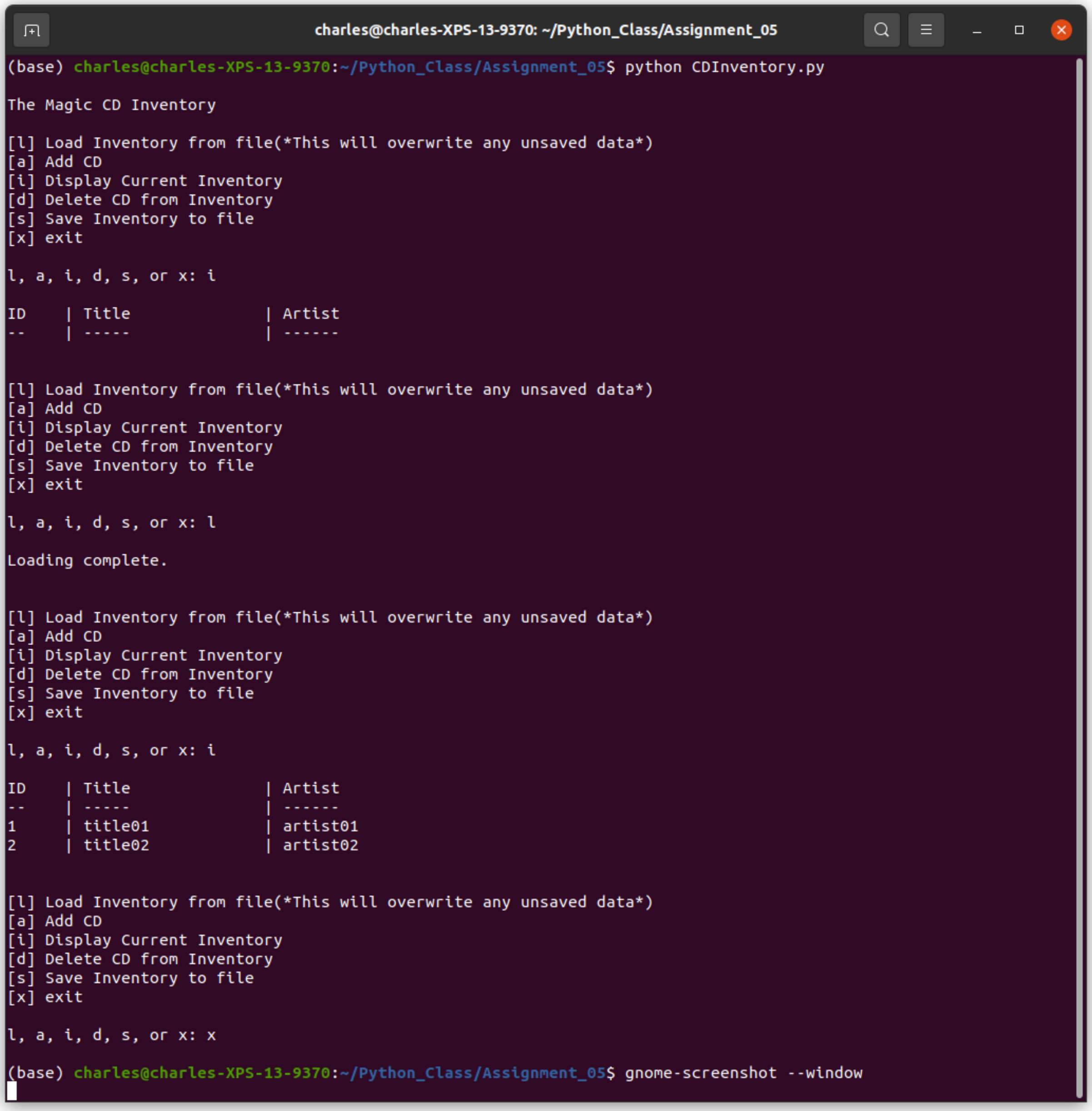
Figure 14 - Running in the Terminal



charles@charles-XPS-13-9370: ~/Python_Class/Assignment_05

```
Enter the CD's Title: title01
Enter the Artist's Name: artist01

Added a CD to the inventory.


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: s

Saved to text file.


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: l

Loading complete.


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: i

ID    | Title             | Artist
--    | -----             | ------
1     | title01           | artist01


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: x

(base) charles@charles-XPS-13-9370:~/Python_Class/Assignment_05$ gnome-screenshot --window
```

Figure 15 - Running in the Terminal



```
(base) charles@charles-XPS-13-9370:~/Python_Class/Assignment_05$ python CDInventory.py

The Magic CD Inventory

[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: i

ID     | Title                | Artist
--     | -----                | ------


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: l

Loading complete.


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: i

ID     | Title                | Artist
--     | -----                | ------
1      | title01              | artist01
2      | title02              | artist02


[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit

l, a, i, d, s, or x: x

(base) charles@charles-XPS-13-9370:~/Python_Class/Assignment_05$ gnome-screenshot --window
```

Figure 16 - Running in the Terminal

Appendix

```
# ------------------------------------------#
# Title: CDInventory.py
# Desc: Starter Script for Assignment 05
# Change Log: (Who, When, What)
# Charles Hodges(hodges11@uw.edu), 2021-Aug-08, Created File
# ------------------------------------------#

```

```python

# Variables
dctRow = {}  # Dictionary data row.
lstTbl = []  # List of dictionaries to hold data.
objFile = None  # file object.
strChoice = ''  # User input.

# Strings
keyArtist = 'Artist'
keyId = 'ID'
keyTitle = 'Title'
strAdd = 'a'
strAddSuccessMsg = 'Added a CD to the inventory.'
strArtistName = 'Enter the Artist\'s Name: '
strCdTitle = 'Enter the CD\'s Title: '
strCdWasDeletedMsg = "CD with ID {} was deleted."
strDelete = 'd'
strDeleteById = "Which CD would you like to remove? Indicate using 'ID': "
strDisplay = 'i'
strEnterId = 'Enter an ID#: '
strErrorMsg = 'Please choose either l, a, i, d, s or x!'
strExit = 'x'
strFileName = 'CDInventory.txt'
strInputHdr = '\nThe Magic CD Inventory'
strLoad = 'l'
strLoadSuccessMsg = 'Loading complete.'
strMenu1 = """
[l] Load Inventory from file(*This will overwrite any unsaved data*)
[a] Add CD
[i] Display Current Inventory
[d] Delete CD from Inventory
[s] Save Inventory to file
[x] exit
"""
strMenuOptions = 'l, a, i, d, s, or x: '
strRead = 'r'
strSave = 's'
strSaveSuccessMsg = 'Saved to text file.'
strWrite = 'w'


# Get user Input
print(strInputHdr)
while True:

    # Display menu allowing the user to choose usage intent.
    print(strMenu1)

    # convert choice to lower case at time of input.
    strChoice = input(strMenuOptions).lower()
    print()  # Empty line for readability.

    # Exit the program.
    if strChoice == strExit:
        break

    # Load existing data.
    if strChoice == strLoad:
        # If the file does not yet exist, create it, to avoid
        # a FileNotFoundError. This will NOT overwrite
        # the current data, if it exists already.
        text_file = open(strFileName, strAdd)
        text_file.close()

        # Reset the Table first
        lstTbl = []

        # Then load data from text file.
        with open(strFileName, strRead) as objFile:
            for row in objFile:
                lstRow = row.strip().split(',')
                dicRow = {
                          'ID': int(lstRow[0]),
                          'Title': lstRow[1],
                          'Artist': lstRow[2]
                         }
                lstTbl.append(dicRow)
            print(strLoadSuccessMsg)
            print()  # Empty line for readability

    # Add data to the table.
    elif strChoice == strAdd:
        intId = int(input(strEnterId))
        strTitle = input(strCdTitle)
        strArtist = input(strArtistName)
        dctRow = {keyId: intId, keyTitle: strTitle, keyArtist: strArtist}
        lstTbl.append(dctRow)
        print()  # Empty line for readability
        print(strAddSuccessMsg)
        print()  # Empty line for readability

    # Display the current data to the user.
    elif strChoice == strDisplay:
        print("{: <5} {: <20} {: <20}".format("ID", "| Title", "| Artist"))
        print("{: <5} {: <20} {: <20}".format("--", "| -----", "| ------"))
        counter = 0
        for row in lstTbl:
            dctRowToLst = list(lstTbl[counter].values())
            lstToStr = ','.join([str(elem) for elem in dctRowToLst])
            split_lines = lstToStr.split(",")
            id_num, title, artist = split_lines
            print("{: <5} {: <20} {: <20}".format
                  (
                      id_num, "| " + title, "| " + artist)
                  )
            counter += 1
        print()  # Empty line for readability.

    # Delete an entry
    elif strChoice == strDelete:
        deleteId = int(input(strDeleteById))
        for items in range(len(lstTbl)):
            if lstTbl[items]['ID'] == deleteId:
                del lstTbl[items]
                print(strCdWasDeletedMsg.format(deleteId))
                break
        print()  # Empty line for readability.

    # Save the data to a text file.
    elif strChoice == strSave:
        with open(strFileName, strWrite) as objFile:
            counter = 0
            for row in lstTbl:
                idNum, title, artist = lstTbl[counter].values()
                objFile.write(str(idNum) + ',' + title + "," + artist + '\n')
                counter += 1
        print(strSaveSuccessMsg)
        print()  # Empty line for readability.

    else:
        print(strErrorMsg)
```