Laura Qin
August 14 2022
IT FDN 110 A
Assignment 06

Functions, Variable Scope and DocString

Introduction

For this assignment we're given some starter codes for a program that manages a CD inventory system. We're required to complete the "TODO"s in the script, which mostly entails writing new code, and reorganizing existing code to put some data, IO, and presentation steps in functions, which ensures better separation of concerns.

Move Processing Code into Functions

In the main part of the script, there are four sections that needs to be moved into functions: (1) asking user to enter ID, title and artist for a new data row, (2) add the new row to the data table, (3) delete a row from the data table, and (4) save data table into the file. For each of the four sections, I created a new function in the corresponding class based on the functionality it serves, so (1) would be in IO class, (2) and (3) would be in DataProcessor class, and (4) would be in FileProcessor class. For each function I try to re-use the code that's already in the starter script, but modify it so that the variables in the main scripts can be passed in as arguments in the function if needed, and made sure that the argument names in the function definition are not the same as the variable names in the main script to avoid running into variable scope issues. I also added DocStrings to the functions to make the code more readable. As an example, here's a comparison between the code in the main script in the starter file and what it looks like after it's been moved into a function.

```
        # 3.3.2 Add item to the table
        # TODO move processing code into function
        intID = int(strID)
        dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
        lstTbl.append(dicRow)
```
Script 1. Starter script - 3.3.2 Add item to the table

```
class DataProcessor:
    '''Processing the data stored in memory'''

    @staticmethod
    def add_row_to_data(id, title, artist, table):
        '''Function to add a new row of data to the table in memory
```

```
       Taking the id, title and artist entered by the user, turn it into a dictionary
and append it to
       the table in memory.

       Args:
           id (string): the id of the new row
           title (string): the title of the cd
           artist (string): the artist of the cd
           table (list of dict): 2D data structure (list of dicts) that holds the data
during runtime

       Returns:
           None.
       '''
       int_id = int(id)
       dicRow = {'ID': int_id, 'Title': title, 'Artist': artist}
       table.append(dicRow)
```
Script 2. New Script - function to add new row to the table

```
       # 3.3.2 Add item to the table
       DataProcessor.add_row_to_data(strID, strTitle, strArtist, lstTbl)
```
Script 3. New script - 3.3.2 Add item to the table

Note that for the function above, I didn't choose to return a value, because it is supposed to do some operations on the data table *lstTbl*, which is a reference type variable, and changing it in the function instead of having the function return a copy of *lstTbl* helps makes the program much faster, especially if *lstTbl* is large. For functions that don't directly deal with the data table, such as the get row input function, I chose to return the id, title, and artist entered by the user as string variables, and then pass them onto the next step, for readability and easy access.

```
       # 3.3.1 Ask user for new ID, CD Title and Artist
       # TODO move IO code into function
       strID = input('Enter ID: ').strip()
       strTitle = input('What is the CD\'s title? ').strip()
       stArtist = input('What is the Artist\'s name? ').strip()

```
Script 4. Starter script - 3.3.1 Ask user for new ID, CD Title and Artist

```
   @staticmethod
   def get_input_row():
       '''Gets user input for a new row of data

```

```
        Prompts the user to enter the id, title and artist for a new row of data, and
then store the entered values
        into three seperate string variables

        Args:
            None.

        Returns:
            id, title, artist (a tuple of strings): the id, title and artist entered by
a user, stored in strings
            inside a tuple
        '''
        id = input('Enter ID: ').strip()
        title = input('What is the CD\'s title? ').strip()
        artist = input('What is the Artist\'s name? ').strip()
        return id, title, artist
```
Script 5. New Script - function to get user's input for a new row

```
        # 3.3.1 Ask user for new ID, CD Title and Artist
        strID, strTitle, strArtist = IO.get_input_row()
```
Script 6. New script - 3.3.1 Ask user for new ID, CD Title and Artist

Improvements

I want to try to make the main code cleaner, so I decided to put into functions all the codes that ask the user to enter a value and capture them in variables. For example, I created a function for asking user which row they would like to delete from the data.

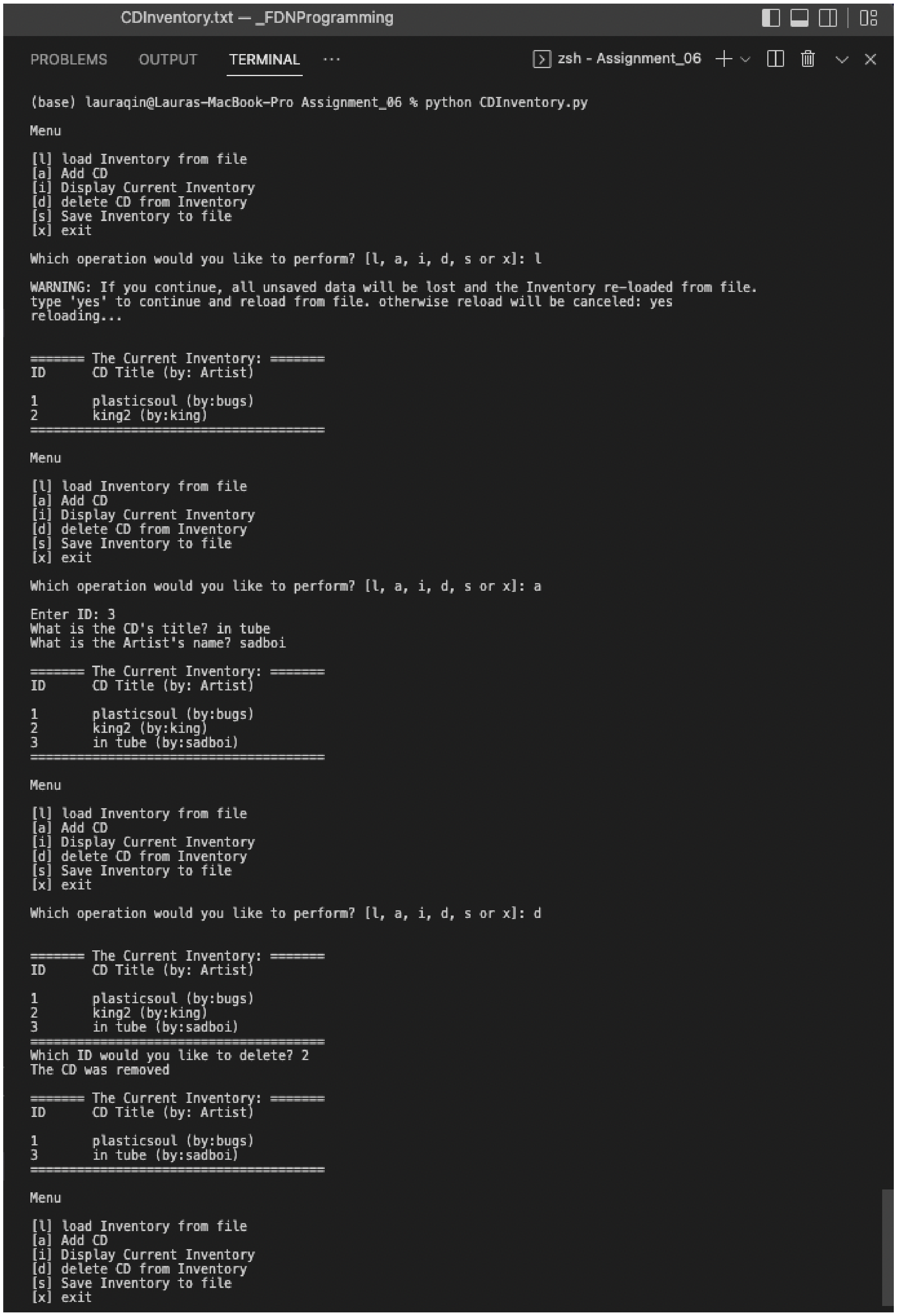
```
        # 3.5.1.2 ask user which ID to remove
        intIDDel = int(input('Which ID would you like to delete? ').strip())
```
Script 7. Starter script - 3.5.1.2 ask user which ID to remove

```
    @staticmethod
    def get_delete_id():
        '''Ask user to enter the id of the row they would like to delete, and store it
in a string

        Args:
            None.

        Returns:
            del_id (string): the id of the row to be deleted from the table
        '''
        del_id = input('Which ID would you like to delete? ').strip()
        return del_id
```
Script 8. New script - function to ask user for the ID of the row to delete, and return the string ID

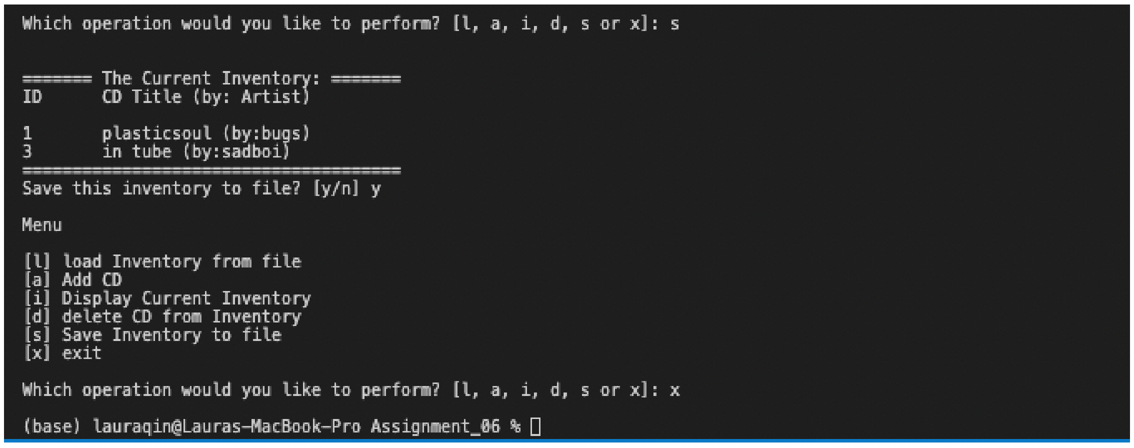
```
        # 3.5.1.2 ask user which ID to remove
        intIDDel = IO.get_delete_id()
```
Script 9. New script - 3.5.1.2 ask user which ID to remove

Result

I ran the updated script in VS Code, it's working as expected. It prints out a menu, and asks the user to enter their selection. It was able to execute the user's request, and generate a text file that includes the data entered by the user. It also behaves as expected when run on the Terminal.

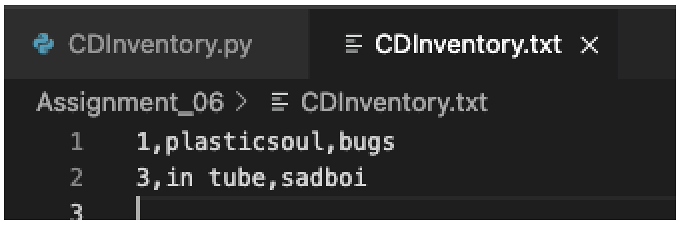


```
(base) lauraqin@Lauras-MacBook-Pro Assignment_06 % python CDInventory.py

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: l

WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be canceled: yes
reloading...


======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
2       king2 (by:king)
======================================

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: a

Enter ID: 3
What is the CD's title? in tube
What is the Artist's name? sadboi

======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
2       king2 (by:king)
3       in tube (by:sadboi)
======================================

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: d


======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
2       king2 (by:king)
3       in tube (by:sadboi)
======================================
Which ID would you like to delete? 2
The CD was removed

======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
3       in tube (by:sadboi)
======================================

Menu

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



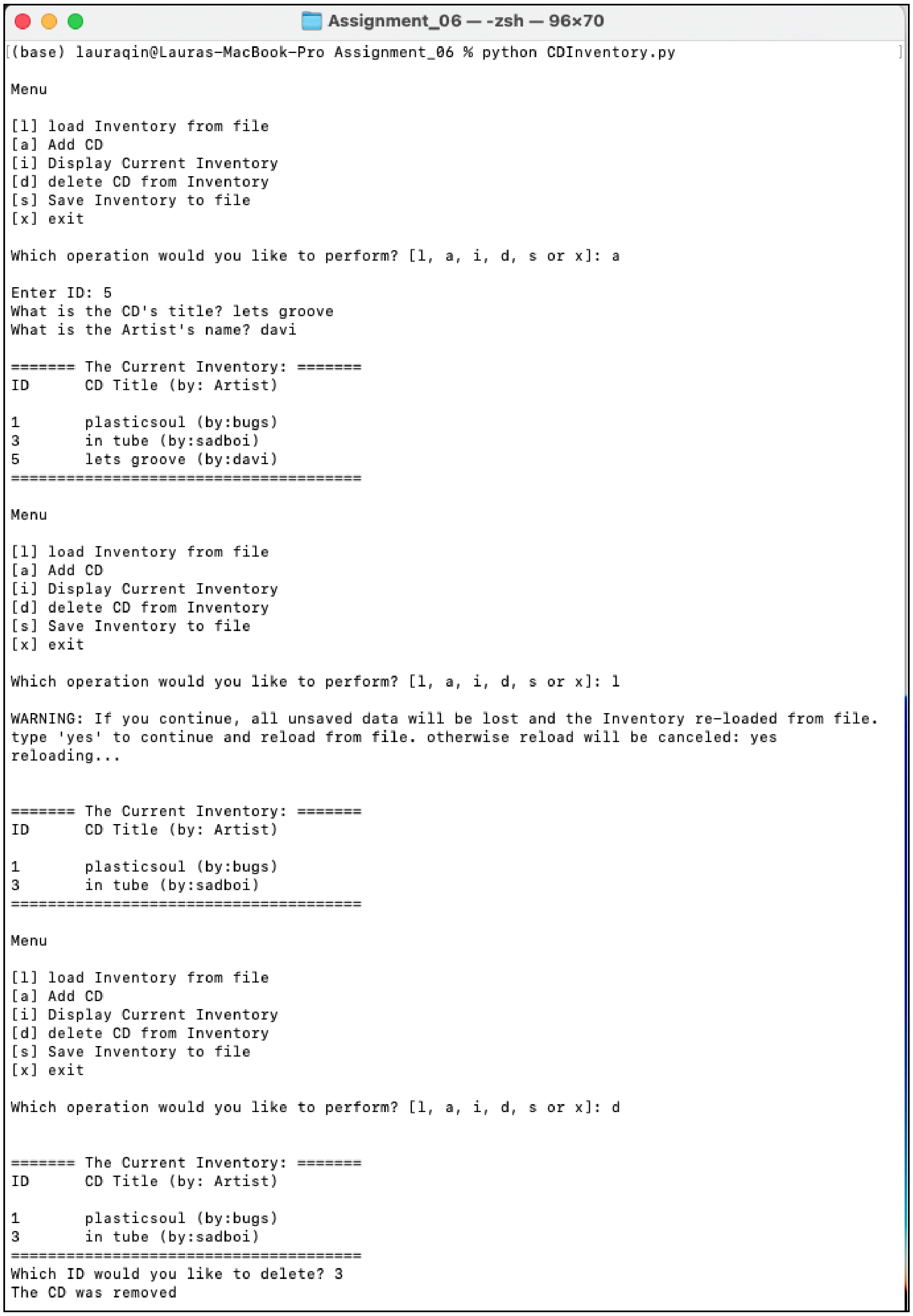
```
Which operation would you like to perform? [l, a, i, d, s or x]: s


======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
3       in tube (by:sadboi)
======================================
Save this inventory to file? [y/n] y

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: x

(base) lauraqin@Lauras-MacBook-Pro Assignment_06 % 
```

Screenshot 1. VSCode Run Result

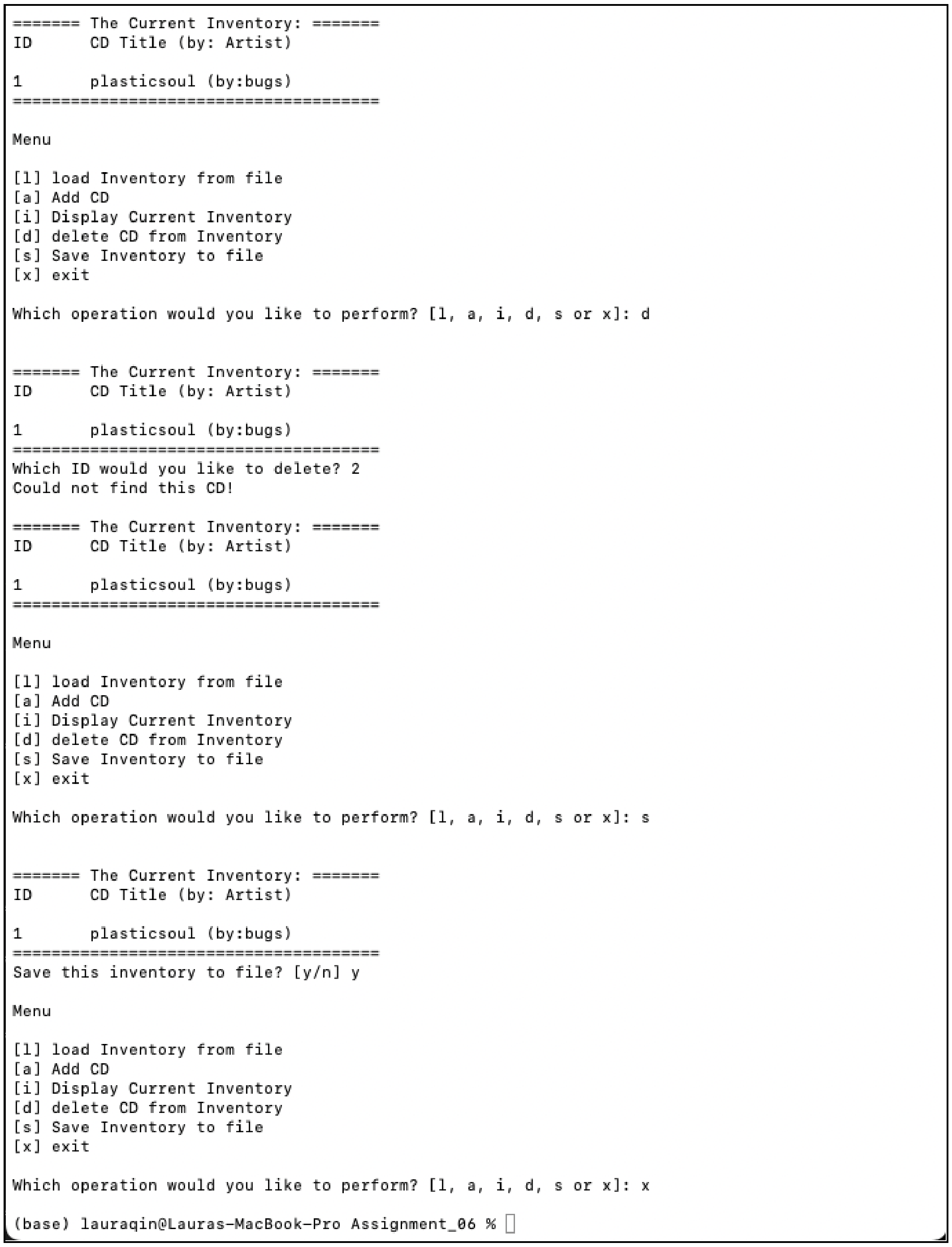


```
1,plasticsoul,bugs
3,in tube,sadboi
```

Screenshot 2. CDInventory.txt after VSCode Run

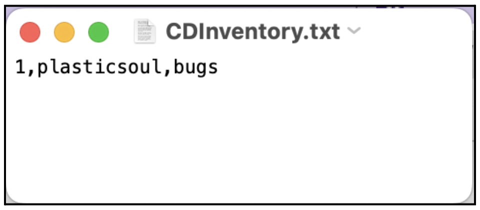

Assignment_06 — -zsh — 96×70

```
(base) lauraqin@Lauras-MacBook-Pro Assignment_06 % python CDInventory.py

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: a

Enter ID: 5
What is the CD's title? lets groove
What is the Artist's name? davi

======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
3       in tube (by:sadboi)
5       lets groove (by:davi)
======================================

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: l

WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.
type 'yes' to continue and reload from file. otherwise reload will be canceled: yes
reloading...


======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
3       in tube (by:sadboi)
======================================

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: d


======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
3       in tube (by:sadboi)
======================================
Which ID would you like to delete? 3
The CD was removed
```



```
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
======================================

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: d


======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
======================================
Which ID would you like to delete? 2
Could not find this CD!

======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
======================================

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: s


======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       plasticsoul (by:bugs)
======================================
Save this inventory to file? [y/n] y

Menu

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

Which operation would you like to perform? [l, a, i, d, s or x]: x

(base) lauraqin@Lauras-MacBook-Pro Assignment_06 % 
```

Screenshot 3. Terminal Run Result



```
1,plasticsoul,bugs
```

Screenshot 4. CDInventory.txt after Terminal Run

Summary

Through this assignment I was able to get more familiar with functions, variable scopes and DocStrings, and also understand the challenges that come with working on and optimizing pre-existing code rather than starting from scratch. Also, I was able to practicing good code design and documentation practice.

Appendix

CDInventory.py

```
#------------------------------------------#
# Title: CDInventory.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# qinlaura, 2023-Aug-14, Updated Filee
#------------------------------------------#

# -- 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:
   '''Processing the data stored in memory'''

   @staticmethod
   def add_row_to_data(id, title, artist, table):
       '''Function to add a new row of data to the table in memory

       Taking the id, title and artist entered by the user, turn it into a dictionary
and append it to
       the table in memory.

       Args:
           id (string): the id of the new row
           title (string): the title of the cd
           artist (string): the artist of the cd
           table (list of dict): 2D data structure (list of dicts) that holds the data
during runtime

       Returns:
           None.
       '''
       int_id = int(id)
       dicRow = {'ID': int_id, 'Title': title, 'Artist': artist}
       table.append(dicRow)

   @staticmethod
   def delete_row(delete_id, table):
       '''Function to delete a row from the table in memory

       Taking the id of the row the user wants to delete, search through the table in
memory,
       check each row to see if the id matches the id of the row that the user wants
to delete,
       and remove that row from the table.

       Args:
           delete_id (string): the id of the row that user wants to delete
           table (list of dict): 2D data structure (list of dicts) that holds the data
during runtime

       Returns:
           None.
       '''
       int_delete_id = int(delete_id)
       row_num = -1
       cd_removed = False
       for row in table:
           row_num += 1
           if row['ID'] == int_delete_id:
               del table[row_num]
               cd_removed = True
               break
       if cd_removed:
           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 to save the table in memory into the file

       Go through each row in the table that is a 2D list of dictionaries, extract the
values in each dictionary
       row, combine them into one comma-seperated string and write it to the file

       Args:
           file_name (string): name of file to save data to
           table (list of dict): 2D data structure (list of dicts) that holds the data
during runtime
```

```python
        Returns:
            None.
        '''
        obj_file = open(file_name, 'w')
        for row in table:
            lst_values = list(row.values())
            lst_values[0] = str(lst_values[0])
            obj_file.write(','.join(lst_values) + '\n')
        obj_file.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('\nMenu\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 get_input_row():
        '''Gets user input for a new row of data

        Prompts the user to enter the id, title and artist for a new row of data, and
then store the entered values
        into three seperate string variables

        Args:
            None.

        Returns:
            id, title, artist (a tuple of strings): the id, title and artist entered by
a user, stored in strings
            inside a tuple
        '''
        id = input('Enter ID: ').strip()
        title = input('What is the CD\'s title? ').strip()
        artist = input('What is the Artist\'s name? ').strip()
        return id, title, artist

    @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('\n======= The Current Inventory: =======')
        print('ID\tCD Title (by: Artist)\n')
        for row in table:
            print('{}\t{} (by:{})'.format(*row.values()))
        print('======================================')

    @staticmethod
    def get_reload_yes_no():
        '''Ask user to confirm if they want to reload data, save user entry in a string

        Prints out a warning that says if the user procees to reload data from file,
 all unsaved changes in
        memory will be discarded. Then it asks user to confirm they want to proceed by
 typing in 'yes',
        otherwise abort the reload.

        Args:
            None.

        Returns:
            reload_yes_no (string): whether or not user wants to proceed with reloading
 data
        '''
        print('WARNING: If you continue, all unsaved data will be lost and the
 Inventory re-loaded from file.')
        reload_yes_no = input('type \'yes\' to continue and reload from file. otherwise
 reload will be canceled: ')
        return reload_yes_no

    @staticmethod
    def get_delete_id():
        '''Ask user to enter the id of the row they would like to delete, and store it
 in a string

        Args:
            None.

        Returns:
            del_id (string): the id of the row to be deleted from the table
        '''
        del_id = input('Which ID would you like to delete? ').strip()
        return del_id

    @staticmethod
    def get_save_yes_no():
        '''Ask user to whether they want to save the current inventory to file, and
 store it in a string

        Args:
            None.

        Returns:
            save_yes_no (string): whether or not the user wants to save current
 inventory to file
        '''
        save_yes_no = input('Save this inventory to file? [y/n] ').strip().lower()
        return save_yes_no



 # 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':
        strYesNo = IO.get_reload_yes_no()
        if strYesNo.lower() == 'yes':
            print('reloading...\n')
            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, strTitle, strArtist = IO.get_input_row()
        # 3.3.2 Add item to the table
        DataProcessor.add_row_to_data(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 = IO.get_delete_id()
        # 3.5.2 search thru table and delete CD
        DataProcessor.delete_row(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 = IO.get_save_yes_no()
        # 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')
```