

# Week 6 Knowledge Document: CD Inventory

## Introduction

For this week's assignment, I'm modifying the CD Inventory app yet again. But this time, I'm going to use functions. Functions should be used whenever we're repeating codes, which also helps with organization and legibility. I'm also going to demonstrate my knowledge of docstrings, global vs local variables and class. Below is my process documenting how I completed the program modifying a given template, and my learnings along the way.

## Process

### Set up the program

For this week's assignment, we are provided with a template as the starting off point with many TODOs in the comments. To begin the program, I first added more detail to the header. Since I wasn't the original creator of the code, I executed and read over the script several times to familiar myself with the codes and its structure.

### Understand the script and tasks

Upon first glance, this script looks quite daunting – there were way more lines of code than I used since the start of this course. But upon closer examination, I realized that there were many comments intertwined in the script. But more importantly, most of the space were taken docstrings which explain how the function works. I also assume once I move block of scripts to functions, I'll be saving lines of codes.

The script is nicely divided into three sections: data, processing, and presentation. Inventory data is stored in dictionaries with ID, Title and Artist as the keys.

For this assignment, my tasks are the following:

Move blocks codes from the main loop to their appropriate sections as functions. Then calling the functions when appropriate. Specifically...

#### [a] Add a CD

- Migrate code to ask for user input into a function in the IO class and call it
- Move code to convert the user input as a row in the table into a function in the DataProcess class and call it

#### [d] Delete a CD

- Migrate code to search through the table and delete the CD into a function in the DataProcess class and call it

#### [s] Save inventory to file

- Migrate the section of the code that save the inventory to a file a function by updating the `FileProcessor.write_file()` code

### [a] Move IO code into function

Here, I needed to move the block of code into a function and put the function under the IO class.

```

1. 2tride = input('Enter ID: ').strip()
2. strTitle = input('What is the CD\'s title? ').strip()
3. stArtist = input('What is the Artist\'s name? ').strip()

```

The first thing I did was to define a function called `ask_new_entry()` under the `IO` class since this is asking for input. Then I added a docstring to explain the function. A docstring is a useful explanatory header at the beginning of a function normally including a brief explanation of the function, its arguments and returns. In order to use the value in the second part of the task under this section – add the entry to table – I returned the value as a list.

To call the function under `[a]`, I simply called the class `IO` followed by `.` then the function. Then I assigned the return result as `userInput` for the next task.

#### [a] Add entry to table

Next step is to add the user input into a table by moving the following code to a function.

```

1. intID = int(strID)
2. dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
3. lstTbl.append(dicRow)

```

First I defined a function called `add_entry_to_table()` in the `DataProcessor` class with `entry` as the parameter. Because in the previous step I have put the user input into a list in order of ID, Title, Artist, here I'm using the values to replace `intID`, `strTitle` and `stArtist` for the dictionary values respectively.

To call the function, I first called the class `DataProcessor`, follow by a `.`, then the function name with `userInput` as the argument.

#### [d] Move processing code into function

Here, I need to move the following block of code into the `DataProcessor` class.

```

1. intRowNr = -1
2. blnCDRemoved = False
3. for row in lstTbl:
4.     intRowNr += 1
5.     if row['ID'] == intIDDel:
6.         del lstTbl[intRowNr]
7.         blnCDRemoved = True
8.         break
9. if blnCDRemoved:
10.    print('The CD was removed')
11. else:
12.    print('Could not find this CD!')

```

To create the function, I first defined a function in the `DataProcessor` class called `remove_id()` and with a parameter of `deleteID`. Then I pasted the code within that function and replaced `intIDDel` with the parameter.

To call the function, I followed the pattern of `class.function(argument)` and used `intIDDel` as the argument, which is the ID number the user wishes to remove.

#### [s] Save inventory to file

In this section, I need to migrate the saving process code into a function. There's already a function `write_file()` defined for this block of code <sup>1</sup> in the `FileProcessor` class.

---

<sup>1</sup> Change of formatting due to PlanetB having an internal server error, Feb-21-2021 at 11pm

```

objFile = open(strFileName, 'w')
for row in lstTbl:
    lstValues = list(row.values())
    lstValues[0] = str(lstValues[0])
    objFile.write(','.join(lstValues) + '\n')
objFile.close()
else:
    input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')

```

All I did was first pasting the code from the main loop to the function then replacing the parameters with `file_name` and `table`.

To call the function, I used `FileProcessor.write_file()` and using `strFileName` and `lstTbl` as the arguments respectively.

### Polish and test

Now that I tested the code and made sure everything ran properly, I added the final touches: added docstrings where needed, provided more context in the script, checked spelling, etc. I ran the script one more time in Spyder, and in the Terminal to make sure it's working as expected.

Here's how the script looked in Spyder:

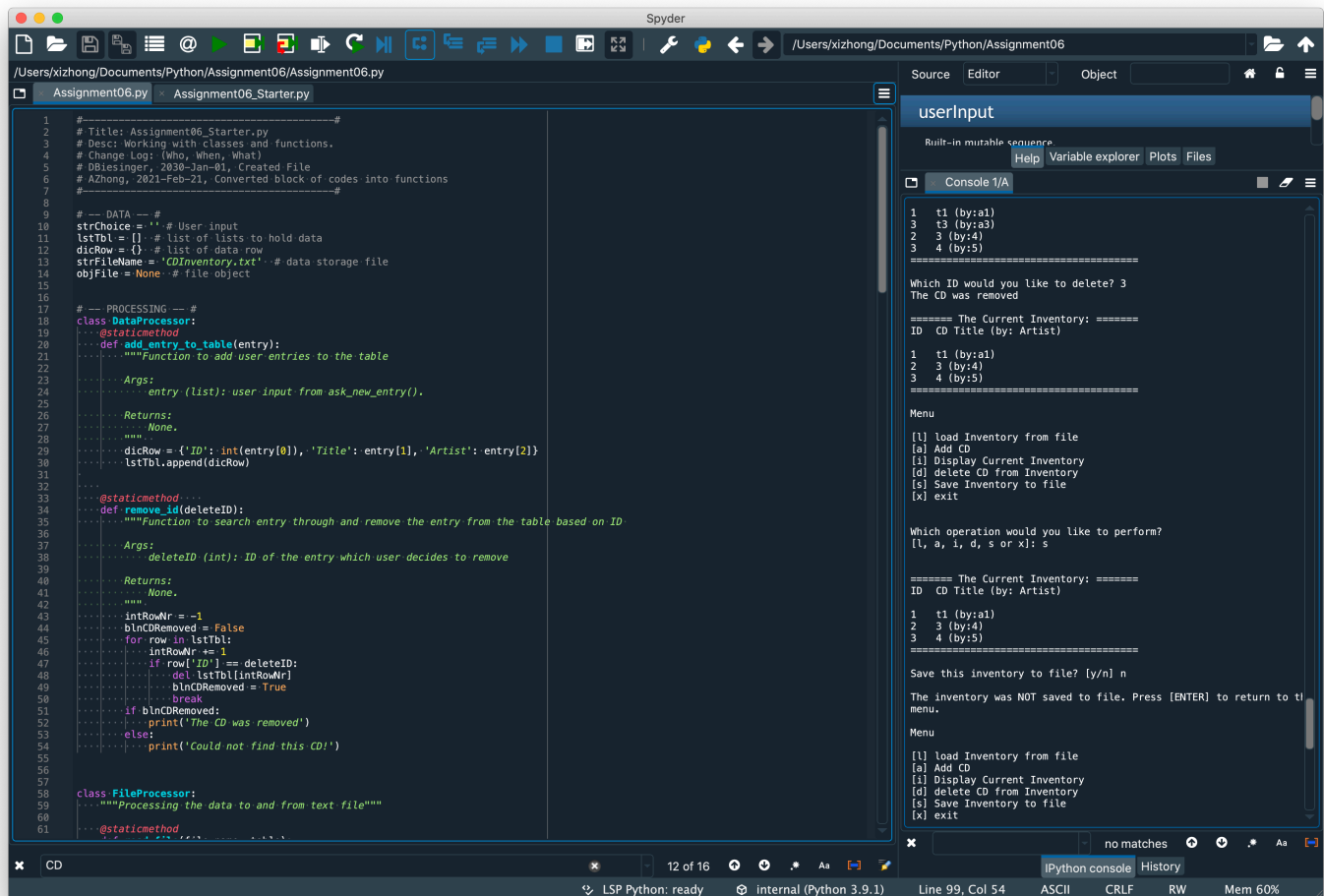
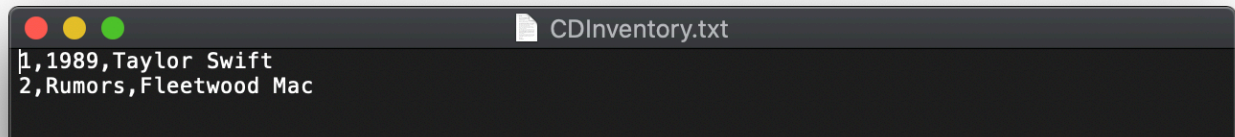


Figure 1: final code in Spyder

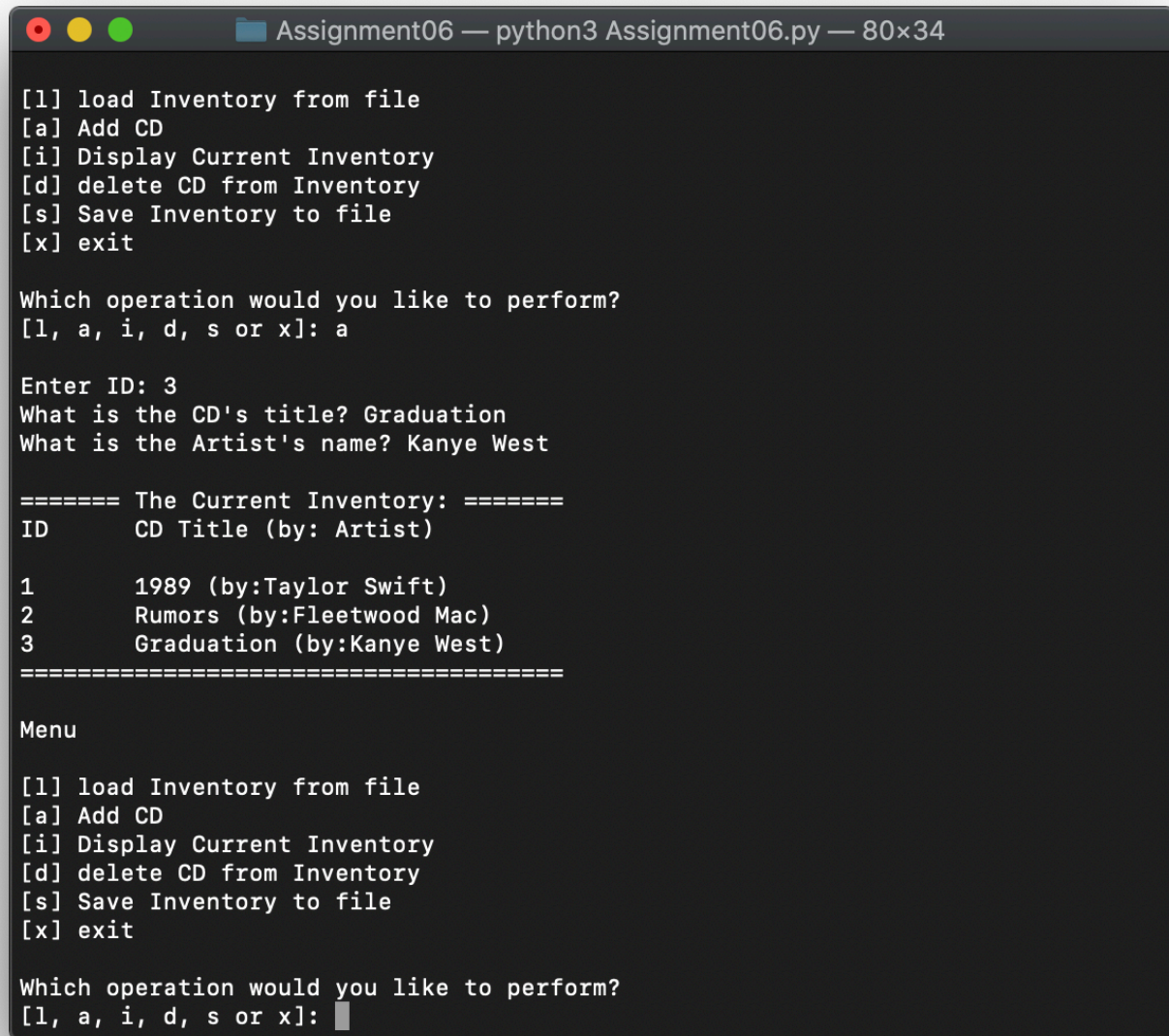
With the corresponding text file:



```
1,1989,Taylor Swift
2,Rumors,Fleetwood Mac
```

Figure 2: Corresponding text file with new data

And here's how it looks in Terminal:



```
Assignment06 — python3 Assignment06.py — 80x34

[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? Graduation
What is the Artist's name? Kanye West

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

1       1989 (by:Taylor Swift)
2       Rumors (by:Fleetwood Mac)
3       Graduation (by:Kanye West)
=====

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]:
```

Figure 3: Final script in Terminal

## Summary

In this module, I learned how to create my own functions and how to use functions and classes to keep code organized using Separations of Concerns pattern. As a rule of thumb, we should be using functions whenever we are repeating blocks of codes. I also learned about docstrings and how useful it is to include them when working with others or when we need to keep track of many functions. This was the first time I worked with

code length this large. It was definitely daunting at first but I'm glad that I was able to plow through the noise and make the script work.

## Appendix

Github repo

<https://github.com/ameszhong/assignment06>

### Final code

```
#-----#
# Title: Assignment06_Starter.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# AZhong, 2021-Feb-21, Converted block of codes into functions
#-----#

# -- 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_entry_to_table(entry):
        """Function to add user entries to the table

        Args:
            entry (list): user input from ask_new_entry().

        Returns:
            None.
        """
        dicRow = {'ID': int(entry[0]), 'Title': entry[1], 'Artist': entry[2]}
        lstTbl.append(dicRow)

    @staticmethod
    def remove_id(deleteID):
        """Function to search entry through and remove the entry from the table based on ID

        Args:
            deleteID (int): ID of the entry which user decides to remove

        Returns:
            None.
```

```

"""
intRowNr = -1
blnCDRemoved = False
for row in lstTbl:
    intRowNr += 1
    if row['ID'] == deleteID:
        del lstTbl[intRowNr]
        blnCDRemoved = True
        break
if blnCDRemoved:
    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 write save entries to an external text file

```

```

        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.

"""

```
objFile = open(file_name, 'w')
```

```
for row in table:
```

```
    lstValues = list(row.values())
```

```
    lstValues[0] = str(lstValues[0])
```

```
    objFile.write(','.join(lstValues) + '\n')
```

```
objFile.close()
```

```
print('The inventory has been saved to file.')
```

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

```

```

@staticmethod
def ask_new_entry():
    """Ask the user to enter the CD ID, title and artist name

    Args:
        strID: ID value
        strTitle: song title
        strArtist: artist name

    Returns: list of user inputs in order of ID, Title, and Artist name

    """
    strID = input('Enter ID: ').strip()
    strTitle = input('What is the CD\'s title? ').strip()
    strArtist = input('What is the Artist\'s name? ').strip()
    return [strID, strTitle, strArtist]

```

# 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':
```



```

        print('Thanks for using the program. See you later!')
        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
    userInput = IO.ask_new_entry()
    # 3.3.2 Add item to the table
    DataProcessor.add_entry_to_table(userInput)
    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
    DataProcessor.remove_id(intIDDel);
    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')
```