

Assignment 06

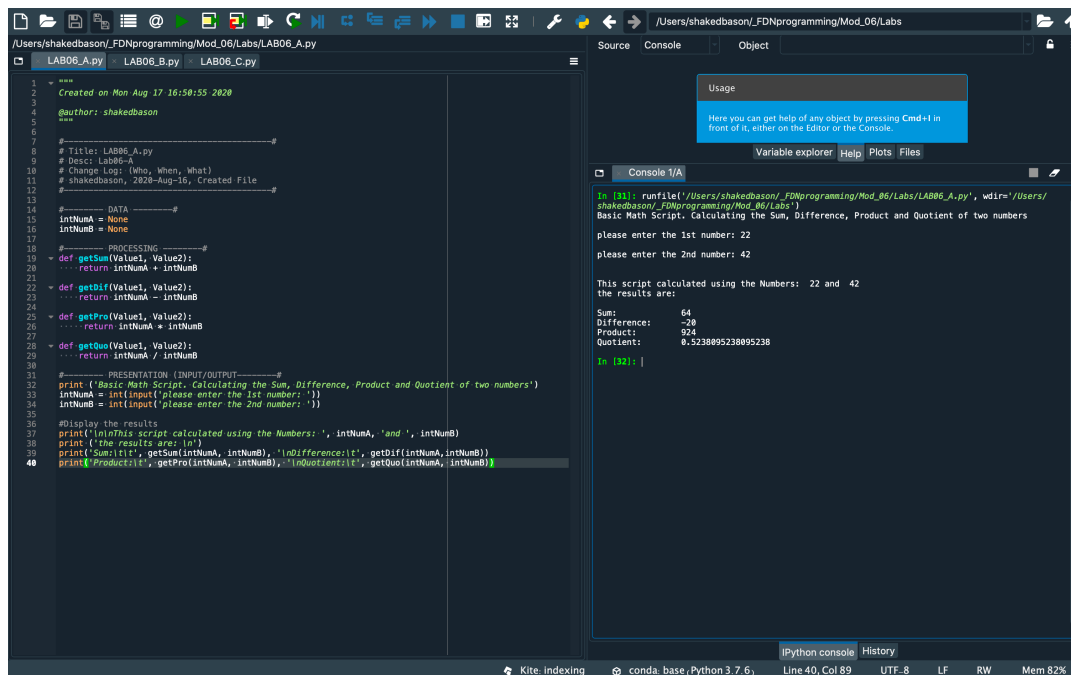
Instruction

In this assignment, questions such as “What is a function?,” and “What are return values?” were answered. Throughout the assignment, I learned the difference between a global and a local variable, how to use functions, and how to organize my code. I also got familiar with the difference between a function and a class.

Module 6 and Labs – Step by step

1. First, I went to the Module 06 materials list and read the instructions. I watched the module videos and followed them step by step. Then, I continued learning about functions and how to write and define one.

I learned that calling the function executes the code in the function, returns the results, and continues at the calling point. I used parameters, which in functions allow the option to pass. Variables can also be used as Arguments in functions. Later, I did my first Lab for this module. The lab asked me to create 4 different functions that calculate the sum, difference, product, and quotient of two numbers. After defining these functions, I asked the user to provide 2 numbers and then I called the function over the number that the user provided. Finally, my code printed the answer that the functions formulated.



The screenshot shows a Jupyter Notebook interface with a dark theme. The left pane displays a Python script for LAB06_A.py. The script includes a header with author information, a description, and four functions: getSum, getDiff, getPro, and getQuo. The main execution block prompts the user for two numbers (22 and 42) and prints the results of the functions. The right pane shows the console output, which includes the user input and the calculated results: Sum: 64, Difference: -20, Product: 928, and Quotient: 0.5238095238095238. The bottom status bar indicates the file is at Line 40, Col 89, using Python 3.7.6.

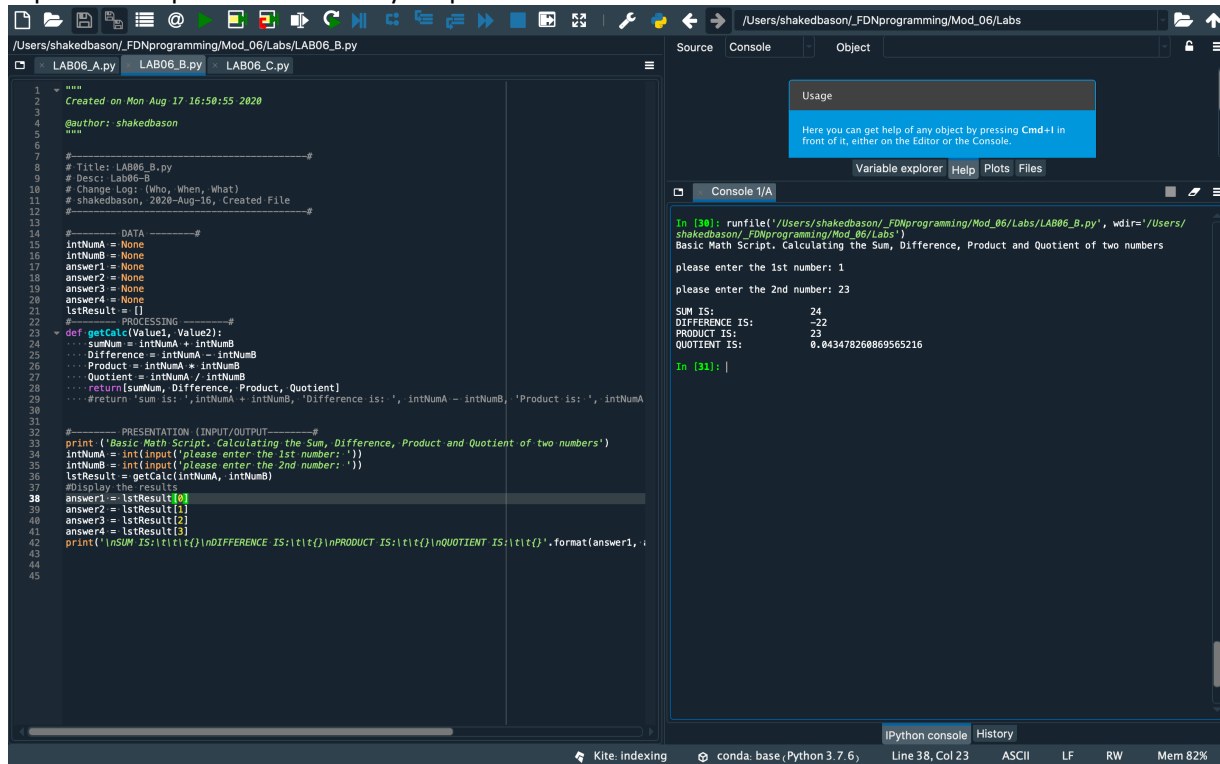
```
1  """  
2  Created on Mon Aug 17 16:50:55 2020  
3  
4  @author: shakedbason  
5  """  
6  
7  # Title: LAB06_A.py  
8  # Desc: Lab06-A  
9  # Change Log: (Who, When, What)  
10 # shakedbason, 2020-Aug-16, Created File  
11  
12 #-----  
13  
14 #----- DATA -----#  
15 intNumA = None  
16 intNumB = None  
17  
18 #----- PROCESSING -----#  
19 def getSum(Value1, Value2):  
20     """return intNumA + intNumB"""  
21     return intNumA + intNumB  
22 def getDiff(Value1, Value2):  
23     """return intNumA - intNumB"""  
24     return intNumA - intNumB  
25 def getPro(Value1, Value2):  
26     """return intNumA * intNumB"""  
27     return intNumA * intNumB  
28 def getQuo(Value1, Value2):  
29     """return intNumA / intNumB"""  
30     return intNumA / intNumB  
31  
32 #----- PRESENTATION (INPUT/OUTPUT) -----#  
33 print('Basic Math Script. Calculating the Sum, Difference, Product and Quotient of two numbers')  
34 intNumA = int(input('please enter the 1st number: '))  
35 intNumB = int(input('please enter the 2nd number: '))  
36  
37 #display the results  
38 print('In this script calculated using the Numbers: ', intNumA, 'and ', intNumB)  
39 print('The results are: ')  
40 print('Sum: ', getSum(intNumA, intNumB), 'Difference: ', getDiff(intNumA, intNumB),  
41       'Product: ', getPro(intNumA, intNumB), 'Quotient: ', getQuo(intNumA, intNumB))
```

Console 1/A

```
In [31]: runfile('/Users/shakedbason/FDNprogramming/Mod_06/Labs/LAB06_A.py', wdir='/Users/shakedbason/FDNprogramming/Mod_06/Labs')  
Basic Math Script. Calculating the Sum, Difference, Product and Quotient of two numbers  
please enter the 1st number: 22  
please enter the 2nd number: 42  
  
This script calculated using the Numbers: 22 and 42  
the results are:  
Sum: 64  
Difference: -20  
Product: 928  
Quotient: 0.5238095238095238  
In [32]: |
```

Figure 1 - LAB06-A

2. In part 2 of this module, I learned more about functions. I learned how to return a single item, as well as multiple items. I was taught how to bundle values into a collection and how to return that collection. Afterwards, I took the Lab06-A script and copied it to a new script. I changed the first lab and modified it to work with a list. Then, I created one function that includes all 4 functions from Lab A. I used this new function on the 2 numbers that I asked the user to provide and then I saved the results under a list of tuples and unpacked them as my output.



```
1 """
2 Created on Mon Aug 17 16:58:55 2020
3
4 @author: shakedbason
5
6
7 #-----#
8 # Title: LAB06-B.py
9 # Desc: Lab06-B
10 # Change Log: (Who, When, What)
11 # shakedbason, 2020-Aug-16, Created File
12 #-----#
13
14 #----- DATA -----#
15 intNumA = None
16 intNumB = None
17 answer1 = None
18 answer2 = None
19 answer3 = None
20 answer4 = None
21 lstResult = []
22
23 #----- PROCESSING -----#
24 def getCalc(Value1, Value2):
25     sumNum = intNumA + intNumB
26     Difference = intNumA - intNumB
27     Product = intNumA * intNumB
28     Quotient = intNumA / intNumB
29     return (sumNum, Difference, Product, Quotient)
30
31 #----- PRESENTATION (INPUT/OUTPUT) -----#
32 print('Basic Math Script. Calculating the Sum, Difference, Product and Quotient of two numbers')
33 intNumA = int(input('please enter the 1st number: '))
34 intNumB = int(input('please enter the 2nd number: '))
35 lstResult = getCalc(intNumA, intNumB)
36
37 #Display the results
38 answer1 = lstResult[0]
39 answer2 = lstResult[1]
40 answer3 = lstResult[2]
41 answer4 = lstResult[3]
42 print('SUM IS: {} DIFFERENCE IS: {} PRODUCT IS: {} QUOTIENT IS: {}'.format(answer1, answer2, answer3, answer4))
43
44
45
```

Usage

Here you can get help of any object by pressing Cmd+I in front of it, either on the Editor or the Console.

Variable explorer Help Plots Files

Console 1/A

```
In [30]: runfile('/Users/shakedbason/_FDNprogramming/Mod_06/Labs/LAB06-B.py', wdir='/Users/shakedbason/_FDNprogramming/Mod_06/Labs')
Basic Math Script. Calculating the Sum, Difference, Product and Quotient of two numbers
please enter the 1st number: 1
please enter the 2nd number: 23
SUM IS: 24
DIFFERENCE IS: -22
PRODUCT IS: 23
QUOTIENT IS: 0.043478260869565216
In [31]:
```

Python console History

Line 38, Col 23 ASCII LF RW Mem 82%

Figure 2 - LAB06-B

3. In part 3 of this module, I continued my exploration of functions. I learned that it is possible to mix positional and named arguments as well as set a default value to parameters. The default value will be used when the user doesn't provide a parameter to the program. I got to test and use "if" statements and conditions inside functions in order to make them more functional. I also got familiar with the "None" keyword and learned how to use and express it in my code and make it even more useful in functions. I also understood that when working with large data amounts, using references to the data is much faster than copying the data. Finally, I learned what "Scope" means and how to use the "Global" keyword.

4. In the last part of this module, I was taught what “classes” are. I learned that classes are a way of grouping functions, variables, and constants under one umbrella. Then, I did Lab06 C. I copied my code from Lab06 A, and modified it to work under class. I took my four functions and put them under one class. Later, when I wanted to show the user his output, I executed my function under my class, so calling the “simpleMath” class I created. Then, I called the specific functions I wanted to use.

The screenshot displays a Jupyter Notebook environment with a dark theme. The main editor window on the left contains a Python script for a class named `simpleMath`. The script includes a docstring, author information, a title, description, and change log. It defines two instance variables, `intNumA` and `intNumB`, both initialized to `None`. The `simpleMath` class has four static methods: `getSum`, `getDiff`, `getPro`, and `getQuo`, each taking two parameters and returning the result of the respective operation. The presentation section prompts the user to enter two numbers, reads the input, and then displays the results of the calculations using the class methods.

```
1 """
2 Created on Mon Aug 17 16:50:55 2020
3
4 @author: shakedbason
5 """
6
7 #-----#
8 # Title: LAB06_C.py
9 # Desc: Lab06-C
10 # Change Log: (Who, When, What)
11 # Shakedbason, 2020-Aug-16, Created File
12 #-----#
13
14 #----- DATA -----#
15 intNumA = None
16 intNumB = None
17
18 #----- PROCESSING -----#
19 class simpleMath():
20     ... @staticmethod
21     ... def getSum(Value1, Value2):
22     ...     return intNumA + intNumB
23
24     ... @staticmethod
25     ... def getDiff(Value1, Value2):
26     ...     return intNumA - intNumB
27
28     ... @staticmethod
29     ... def getPro(Value1, Value2):
30     ...     return intNumA * intNumB
31
32     ... @staticmethod
33     ... def getQuo(Value1, Value2):
34     ...     return intNumA / intNumB
35
36 #----- PRESENTATION (INPUT/OUTPUT) -----#
37 print('Basic Math Script. Calculating the Sum, Difference, Product and Quotient of two numbers')
38 intNumA = int(input('please enter the 1st number: '))
39 intNumB = int(input('please enter the 2nd number: '))
40
41 #Display the results
42 print('\n\nThis script calculated using the Numbers: ', intNumA, 'and ', intNumB)
43 print('the results are: \n')
44 print('Sum:\t\t', simpleMath.getSum(intNumA, intNumB), '\nDifference:\t', simpleMath.getDiff(intNumA, intNumB), '\nProduct:\t', simpleMath.getPro(intNumA, intNumB), '\nQuotient:\t', simpleMath.getQuo(intNumA, intNumB))
```

The right-hand pane shows the console output. It displays the execution of the script, including the prompts for the two numbers (1 and 2) and the resulting calculations:

```
In [29]: runfile('/Users/shakedbason/FDNprogramming/Mod_06/Labs/untitled3.py', wdir='/Users/shakedbason/FDNprogramming/Mod_06/Labs')
Basic Math Script. Calculating the Sum, Difference, Product and Quotient of two numbers
please enter the 1st number: 1
please enter the 2nd number: 2

This script calculated using the Numbers: 1 and 2
the result are:

Sum: 3
Difference: -1
Product: 2
Quotient: 0.5
In [30]:
```

The bottom status bar indicates the environment is 'conda: base (Python 3.7.6)' and shows the current position at 'Line 46, Col 111'.

Figure 3 - LAB06-C

5. Creating a program - I noted what the code does and added it to this document using the planet-b website.

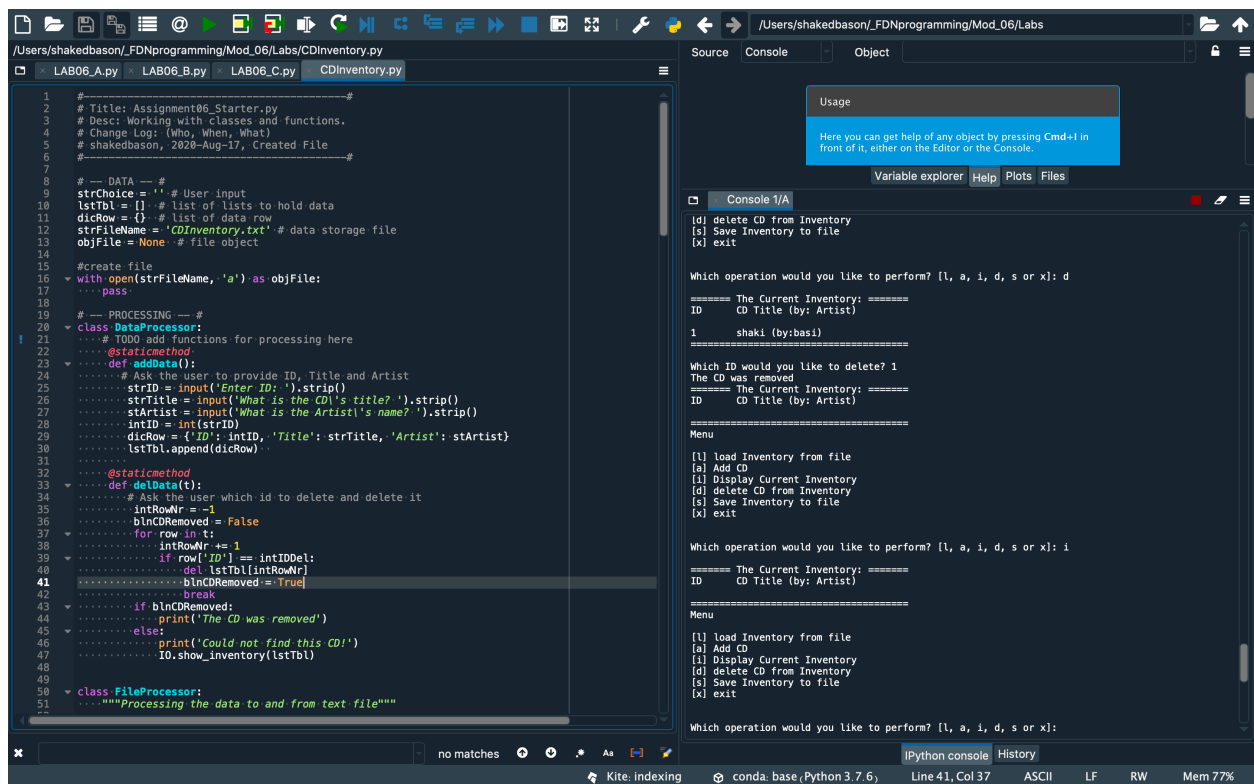
Finally, I created my code for this assignment (Appendix01), which I got from my instructor. The code that I received was incomplete and I had to edit my instructor's code which I found challenging. Since I did not write the code, I needed to go through his code, understand it, and then modify what he wrote to create new code that works with the new classes and functions I added.

Appendix #1

The first part of my code defines new classes. The first class, named `DataProcessor`, includes two functions. One adds data to the `CDInventory` and the second function deletes from it.

The second class, named `FileProcessor`, includes the function of read and write. Lastly, the `IO` class, has the function of print, display, and menu choice.

After I finished defining all of the functions and classes, I used the old familiar `CDInventory` task and Modified it to use and call the functions and the classes.



The screenshot displays a code editor with the file `CDInventory.py` open. The code defines two classes: `DataProcessor` and `FileProcessor`. The `DataProcessor` class includes methods for adding and deleting data from the inventory. The `FileProcessor` class includes methods for reading and writing data to a file. The console output shows the program's execution, including prompts for user input and the current state of the inventory.

```
1 # Title: Assignment06 Starter.py
2 # Desc: Working with classes and functions.
3 # Change Log: (Who, When, What)
4 # shakedbason, 2020-Aug-17, Created File
5
6
7
8 # -- DATA -- #
9 strChoice = '' # User input
10 lstTbl = [] # list of lists to hold data
11 dicRow = {} # list of data row
12 strFileName = 'CDInventory.txt' # data storage file
13 objFile = None # file object
14
15 #create file
16 with open(strFileName, 'a') as objFile:
17     pass
18
19 # -- PROCESSING -- #
20
21 class DataProcessor:
22     """add functions for processing here"""
23     @staticmethod
24     def addData():
25         """Ask the user to provide ID, Title and Artist"""
26         strID = input('Enter ID: ').strip()
27         strTitle = input('What is the CD's title? ').strip()
28         strArtist = input('What is the Artist's name? ').strip()
29         intID = int(strID)
30         dicRow = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}
31         lstTbl.append(dicRow)
32
33     @staticmethod
34     def delData(t):
35         """Ask the user which id to delete and delete it"""
36         intRowNr = -1
37         blnCDRemoved = False
38         for row in t:
39             intRowNr += 1
40             if row['ID'] == intIDDel:
41                 del lstTbl[intRowNr]
42                 blnCDRemoved = True
43                 break
44             if blnCDRemoved:
45                 print('The CD was removed')
46         else:
47             print('Could not find this CD!')
48         IO.show_inventory(lstTbl)
49
50 class FileProcessor:
51     """Processing the data to and from text file"""
```

Console 1/A

```
[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       shaki (by:basi)

Which ID would you like to delete? 1
The CD was removed
===== The Current Inventory: =====
ID      CD Title (by: Artist)
-----

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]: i

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

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 4 - Assignment output

Summary

Module 6 taught me how to use functions and classes, what the difference is between a global and a local variable, how functions help you program using the "Separations of Concerns" pattern, as well as more new concepts.

I continued improving my usage of Loops, Sequences, Strings, and Dictionaries and modified them into functions. I used Spyder as my IDE on this assignment.

Appendix

#1 CDInventory.py

```
• #-----#
• # Title: Assignment06_Starter.py
• # Desc: Working with classes and functions.
• # Change Log: (Who, When, What)
• # shakdbason, 2020-Aug-17, Created File
• #-----#
•
•
• # -- 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
•
• #create file
• with open(strFileName, 'a') as objFile:
•     pass
•
• # -- PROCESSING -- #
• class DataProcessor:
•     # TODO add functions for processing here
•     @staticmethod
•     def addData():
•         # Ask the user to provide ID, Title and Artist
•         strID = input('Enter ID: ').strip()
•         strTitle = input('What is the CD\'s title? ').strip()
•         stArtist = input('What is the Artist\'s name? ').strip()
•         intID = int(strID)
•         dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
•         lstTbl.append(dicRow)
•
•     @staticmethod
•     def delData(t):
•         # Ask the user which id to delete and delete it
•         intRowNr = -1
•         blnCDRemoved = False
•         for row in t:
```

```

    intRowNr += 1
    if row['ID'] == intIDDel:
        del lstTbl[intRowNr]
        blnCDRemoved = True
        break
    if blnCDRemoved:
        print('The CD was removed')
    else:
        print('Could not find this CD!')
    IO.show_inventory(lstTbl)

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):
        # Save data to a file
        objFile = open(strFileName, 'w')
        for row in lstTbl:
            lstValues = list(row.values())
            lstValues[0] = str(lstValues[0])
            objFile.write(','.join(lstValues) + '\n')
        objFile.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('Menu\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 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('==== The Current Inventory: =====')
• print('ID\tCD Title (by: Artist)\n')
• for row in table:
• print('{ }\t{ } (by:{ })'.format(*row.values()))
• print('=====')

• # 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':
• 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
• DataProcessor.addData()
• 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)
•     intIDDel = int(input('Which ID would you like to delete? ').strip())
•     DataProcessor.delData(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 = input('Save this inventory to file? [y/n] ').strip().lower()
•     # 3.6.2 Process choice
•
•     if strYesNo == 'y':
•         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')

```