< Weimin Dang >
< Feb-28-2020>
< IT FDN 100 Winter 2020 >
< Assignment06>

Functions and Classes

Introduction

Assignment 6 aims at getting me to practice the following knowledge points:

1. Practice using functions, classes and docstrings
2. Continue to practice working with others' code, performing peer review, and using github

The Github repo of this assignment is available at https://github.com/wd-uwGH2020/assignment06

Developing the Code

Starter code

Below summarizes the initial assessment of the starter code provided:

1. The starter code is based on an example solution of assignment 05, the basic CD inventory system.
2. The goal of this assignment is to turn some blocks of code into functions, and using classes to organize functions.
3. The starter code has set up the separation of concerns structure, and provided a number of examples of the classes and functions.
4. Three main tasks are to be completed:
    a. Convert the process add a CD code block using functions and classes
    b. Convert the process delete a CD code block using functions and classes
    c. Convert the process save inventory to file using functions and classes

Development of the Modified Code:

1. First, I started the two TODOes in the following code block related to the process of adding a CD:

```
134     # 3.3 process add a CD
135     elif strChoice == 'a':
136         # 3.3.1 Ask user for new ID, CD Title and Artist
137         # TODO move IO code into function
138         strID = input('Enter ID: ').strip()
139         strTitle = input('What is the CD\'s title? ').strip()
140         stArtist = input('What is the Artist\'s name? ').strip()
141 
142         # 3.3.2 Add item to the table
143         # TODO move processing code into function
144         intID = int(strID)
145         dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
146         lstTbl.append(dicRow)
147         IO.show_inventory(lstTbl)
148         continue  # start loop back at top.
```

a. The 1st one on line 137 of the starter code is related to add a new function in the IO class to collect user inputs of a new CD. The following code is converted from the code block to create a new function user_input_to_add_inventory(). No arguments are needed in this function, and the return values are the user inputs.

```
    # TODOne add I/O functions as needed

    @staticmethod
    def user_input_to_add_inventory():
        """Collect user inputs to add new CDs to inventory


        Args:
            None.

        Returns:
            User inputs as string variables.

        """
        str1 = input('Enter ID: ').strip()
        str2 = input('What is the CD\'s title? ').strip()
        str3 = input('What is the Artist\'s name? ').strip()
        return str1, str2, str3
```

b. The 2nd one on line 143 of the starter code is related to add a new function in the DataProcessing class to add the user inputs of a new CD collected to the list of dictionaries, using a table parameter. First, three variables stride, strTitle and strArtist are used to unpack the list of user inputs by calling the function of IO.user_input_to_add_inventory(). A row parameter is then used to convert the unpacked user inputs to a dictionary format.

```
# -- PROCESSING -- #
class DataProcessor:
    # TODOne add functions for processing here
    """Adding or deleting data to be processed"""

    @staticmethod
    def add_inventory(table, row):
        """Add user inputs of new CD to current inventory table


        Args:
            Unpack variables of user inputs collected
            Convert user input CD ID to integer format
            row (dictionary): 2D data strucure (dict) that convert user inputs to the library format

        Returns:
            table (list of dicts): new data are added to the table.

        """
        strID, strTitle, strArtist = IO.user_input_to_add_inventory()
        intID = int(strID)
        row = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}
        table.append(row)
```

The new functions created replaced the original code as the following:

```
218     # 3.3 process add a CD
219     elif strChoice == 'a':
220         # 3.3.1 Ask user for new ID, CD Title and Artist
221         # TODOne move IO code into function
222         # 3.3.1 is embedded in 3.3.2
223
224         # 3.3.2 Add item to the table
225         # TODOne move processing code into function
226         DataProcessor.add_inventory(lstTbl, dicRow)
227         IO.show_inventory(lstTbl)
228         continue  # start loop back at top.
```

2. Next TODO on line 161 of the starter code is to convert the following code block to a function under the DataProcessing class.

```
153     # 3.5 process delete a CD
154     elif strChoice == 'd':
155         # 3.5.1 get Userinput for which CD to delete
156         # 3.5.1.1 display Inventory to user
157         IO.show_inventory(lstTbl)
158         # 3.5.1.2 ask user which ID to remove
159         intIDDel = int(input('Which ID would you like to delete? ').strip())
160         # 3.5.2 search thru table and delete CD
161         # TODO move processing code into function
162         intRowNr = -1
163         blnCDRemoved = False
164         for row in lstTbl:
165             intRowNr += 1
166             if row['ID'] == intIDDel:
167                 del lstTbl[intRowNr]
168                 blnCDRemoved = True
169                 break
170         if blnCDRemoved:
171             print('The CD was removed')
172         else:
173             print('Could not find this CD!')
174         IO.show_inventory(lstTbl)
175         continue  # start loop back at top.
```

One parameter table is needed in this conversion as shown in the following:

```python
    @staticmethod
    def delete_inventory(table):
        """Delete CD inventory from current inventory table based on user input CD ID using a counter intRowNr, and confirm the deletion with user


        Args:
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime.

        Returns:
            table (list of dict): new data are added to the table .

        """
        intRowNr = -1
        blnCDRemoved = False
        for row in lstTbl:
            intRowNr += 1
            if row['ID'] == intIDDel:
                del table[intRowNr]
                blnCDRemoved = True
                break
        if blnCDRemoved:
            print('The CD was removed')
        else:
            print('Could not find this CD!')
```

The new functions created replaced the original code as the following:

```python
    # 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
        # TODOne move processing code into function
        DataProcessor.delete_inventory(lstTbl)
        IO.show_inventory(lstTbl)
        continue  # start loop back at top.
```

3. Next TODO on line 184 of the starter code is to convert the following code:

```python
    # 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
            # TODO move processing code into function
            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.')
        continue  # start loop back at top.
```

In the class of FileProcessor section, defining a write_file function is straight forward, as it is a very similar to the read_file function operation, but pretty much in an inverse way, and this is related to the saving data to file functionality. Similar to the read_file function, the write_file function also needs two parameters file_name and table respectively. All that is required is to replace the specific filename and list table in the code block in the saving data to file section as shown below.

```python
    @staticmethod
    def write_file(file_name, table):
        # TODOne Add code here
        """Function to manage data ingestion from a list of dictionaries to a file

        Overwrite the data from file identified by file_name with the follow:
        Extract the values only of the each row of the list of dicts as a new list, one row a time
        Convert the first value to string format
        Write each of the values comma separated, and start a new line at the end

        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()
```

The new functions created replaced the original code as the following:

```python
    # 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
            # TODOne move processing code into function
            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.
```

This concludes the tasks of the assignment.

Summary

Running the script in Spyder:

Figure 1 on the next page shows a demonstration of the following operations a user performs:

1. Display the existing inventory
2. Add a new CD
3. Delete the entry
4. Save it to the inventory list file
5. Confirm the save
6. Exit the program

Figure 1

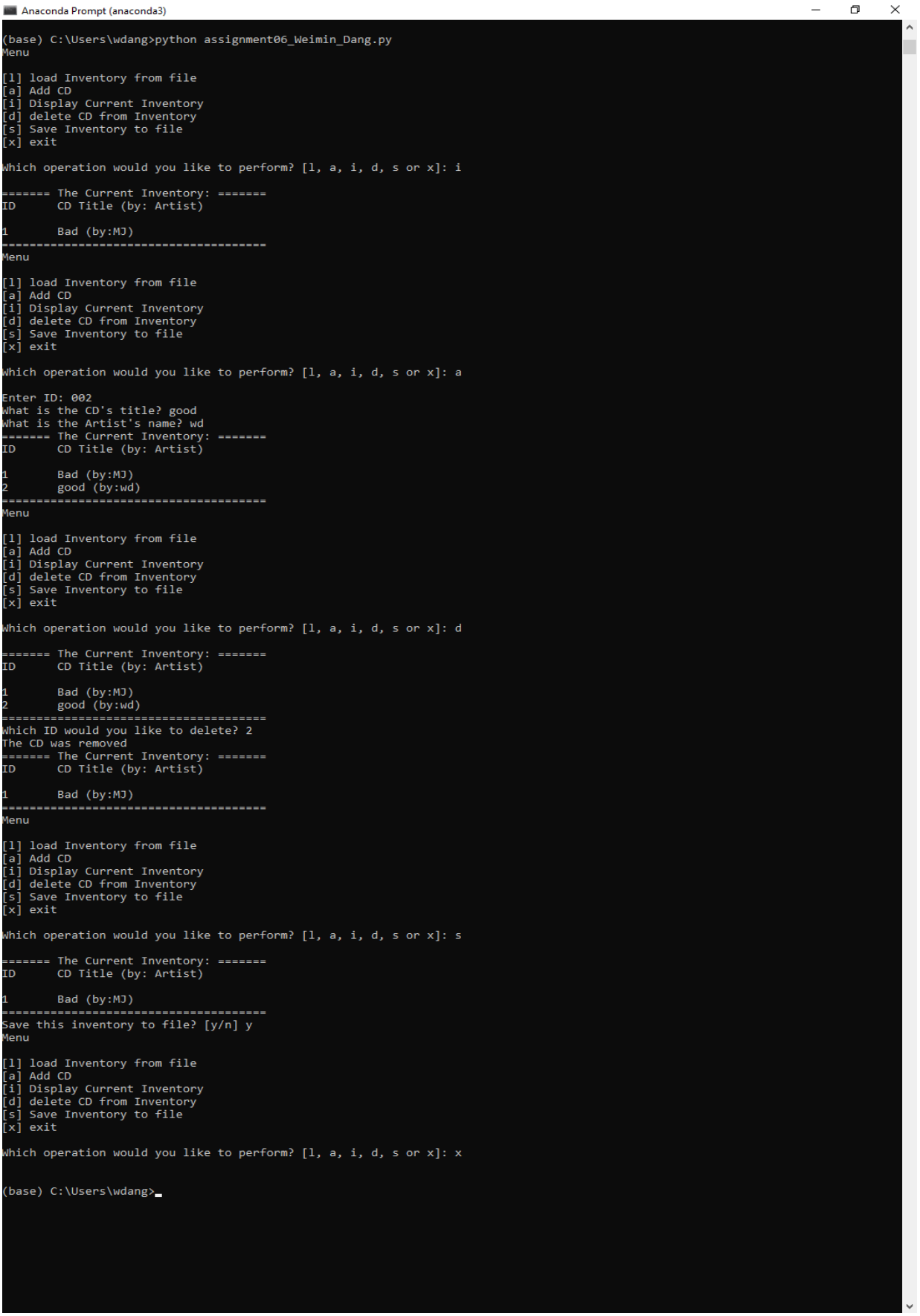
IPython console

Console 1/A

```
In [21]: runfile('C:/Users/wdang/Documents/UW Python Course 2020/UW Python Course/Jan-Mar Intro/wk6/Mod_06/Assignment06/Assignment06_Weimin
Dang.py', wdir='C:/Users/wdang/Documents/UW Python Course 2020/UW Python Course/Jan-Mar Intro/wk6/Mod_06/Assignment06')
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)

1       Bad (by:MJ)
======================================
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: 002

What is the CD's title? Good

What is the Artist's name? WD
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Bad (by:MJ)
2       Good (by:WD)
======================================
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       Bad (by:MJ)
2       Good (by:WD)
======================================

Which ID would you like to delete? 2
The CD was removed
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Bad (by:MJ)
======================================
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       Bad (by:MJ)
======================================

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


In [22]:
```

Figure 2 shows the same code and process running in the terminal.

Figure 2



Lessons learned:

Functions and classes:

Through this exercise in the assignment, it demonstrated how using functions and classes can improve structuring of the code, improve readability, and improve efficiency for repeated scripts.

Appendix – copy of the script

```
#------------------------------------------#
# Title: Assignment06_Starter.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# Wdang, 2020-Feb-28, Modified 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


# -- PROCESSING -- #
class DataProcessor:
    # TODOne add functions for processing here
    """Adding or deleting data to be processed"""

    @staticmethod
    def add_inventory(table, row):
        """Add user inputs of new CD to current inventory table


        Args:
            Unpack variables of user inputs collected
            Convert user input CD ID to integer format
            row (dictionary): 2D data strucure (dict) that convert user inputs to
the library format

        Returns:
            table (list of dicts): new data are added to the table .

        """
        strID, strTitle, strArtist = IO.user_input_to_add_inventory()
        intID = int(strID)
        row = {'ID': intID, 'Title': strTitle, 'Artist': strArtist}
        table.append(row)

    @staticmethod
    def delete_inventory(table):
        """Delete CD inventory from current inventory table based on user input
CD ID using a counter intRowNr, and confirm the deletion with user


        Args:
            table (list of dict): 2D data structure (list of dicts) that holds
the data during runtime.

        Returns:
            table (list of dict): new data are added to the table .

        """
        intRowNr = -1
```

```
        blnCDRemoved = False
        for row in lstTbl:
            intRowNr += 1
            if row['ID'] == intIDDel:
                del table[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):
        # TODOne Add code here
        """Function to manage data ingestion from a list of dictionaries to a
file

        Overwrite the data from file identified by file_name with the follow:
        Extract the values only of the each row of the list of dicts as a new
list, one row a time
        Convert the first value to string format
        Write each of the values comma separated, and start a new line at the end

        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()


# -- 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('=======================================')

    # TODOne add I/O functions as needed

    @staticmethod
    def user_input_to_add_inventory():
        """Collect user inputs to add new CDs to inventory


        Args:
            None.

        Returns:
            User inputs as string variables.

        """
        str1 = input('Enter ID: ').strip()
        str2 = input('What is the CD\'s title? ').strip()
        str3 = input('What is the Artist\'s name? ').strip()
        return str1, str2, str3



# 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
        # TODOne move IO code into function
        # 3.3.1 is embedded in 3.3.2

        # 3.3.2 Add item to the table
        # TODOne move processing code into function
        DataProcessor.add_inventory(lstTbl, dicRow)
        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
        # TODOne move processing code into function
        DataProcessor.delete_inventory(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':
            # 3.6.2.1 save data
            # TODOne move processing code into function
            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 safe:
    else:
        print('General Error')
```