< Weimin Dang >
< Mar-15-2020>
< IT FDN 100 Winter 2020 >
< Assignment08>

Intro to Classes and Objects

Introduction

Assignment 8 aims at getting me to practice the following knowledge points:

1. Review classes and functions
2. Practice classes and objects

The Github repo of this assignment is available at https://github.com/wd-uwGH2020/assignment08

Module 8 starts to introduce the core concept of programming using Python that it is an Object Oriented Programming language. Everything we have learned since the beginning of the course is an object, from a string, a list, a dictionary to the example of a CD in assignment 8. The biggest distinction in this module is to treat a CD as a "concrete" thing, which has attributes or properties such as id, title and artist. In contrast, previously those variables are more just information, which is then constructed to be stored in different data formats such as list, tuple and dictionary.[1]

Developing the Code

Assessment of the starter code:

The overview of the starter code is that, from a functionality perspective, the program is still about the same as what has been done in previous modules. Therefore, a lot of the previous code can be reused. The biggest requirement is to modify and re-write the previous program by treating each piece of CDs as an individual object, and the CD inventory as a list of these CD objects.

By examining the two global variables set up in the starter code as shown below, I got confused and challenged throughout the development of the code. The challenge is that the CD inventory data is "required" to be saved into a txt file, where only string type data can be saved, however, the CDs and the CD inventory list that will be processed in the program are objects. A conversion between the two types of data seems needed in the program somewhere.

```
10    # -- DATA -- #
11    strFileName = 'cdInventory.txt'
12    lstOfCDObjects = []
13
```

Compared to the previous program, delete a CD is not part of the requirement for this assignment. I was not sure if additional or new concepts would be needed to perform this particular function. I left it in to give myself a small challenge.

Development of the code:

As discussed above in the assessment of the starter code, lots of the code from assignment7 can be reused for assignment8.

1. First, I copied over the main menu and the user menu choice sections of the previous code. No changes are needed for the mainloop part and the IO class and methods for these two functions.

[1] Again, all of these are also different types of objects, as Python is an object oriented program.

2. Next, my thinking was: if there is no pre-existing data, there is no existing file to load data from, no existing records to display, and nothing to save. Therefore, the first step of everything else is to add new data or a new CD. This goes to the section of the CD class.

   Since the start code has laid out the three properties of the CD object, the instantiation of an CD object is defined as:

```
27      def __init__(self, cd_id, cd_title, cd_artist):
28          self.cd_id = cd_id
29          self.cd_title = cd_title
30          self.cd_artist = cd_artist
31
```

   Next, based on user inputs of the id, title and artist, a new CD can be created.

```
33      @staticmethod
34      def add_inventory(cd_id, cd_title, cd_artist):
35          """create a new CD object based on user inputs
36
37          Args:
38              each of the three cd attributes
39
40          Returns:
41              the new CD object
42          """
43
44          new_cd = CD(cd_id, cd_title, cd_artist)
45          return new_cd
```

   The user inputs parts are essentially the same as the previous assignment, so I just copied those over to the IO class. It also addresses the same issues such as handling input type error and duplicate ids.

```
219     @staticmethod
220     def user_input_to_add_inventory(table):
221         """Collect user inputs to add new CDs to inventory
232
233         cd_id = IO.get_new_cd_id(table)
234         cd_title = input('What is the CD\'s title? ').strip()
235         cd_artist = input('What is the Artist\'s name? ').strip()
236         return cd_id, cd_title, cd_artist
237
238     @staticmethod
239     def get_new_cd_id(table):
240         """Gets a new, unused CD ID from the user
249         used_ids = []
250         for cd in table:
251             used_ids.append(cd.cd_id)
252
253         while True:
254             cd_id = IO.get_typed_input('Enter a numerical ID: ', 'The entered ID is not an integer
255             if cd_id in used_ids:
256                 print("CD ID '{}' already exsits, use a different ID\n".format(cd_id))
257             else:
258                 return cd_id
259
260
261     @staticmethod
262     def get_typed_input(input_message, error_message):
263         """Prompts the user for input and checks for the correct type.
277
278         while True:
279             try:
280                 typed_value = int(input(input_message).strip())
281                 return typed_value
282             except ValueError:
283                 print(error_message)
```

With these codes, I added the codes on line 332 and 333 as shown below to create a new CD object based on user inputs, then add the CD object to a list of CD objects.

```
330         # 3.3 process add a CD
331         elif strChoice == 'a':
332             # 3.3.1 Ask user for new ID, CD Title and Artist and add to CD inventory table
333             cd_id, cd_title, cd_artist = IO.user_input_to_add_inventory(lstOfCDObjects)
334             lstOfCDObjects.append(CD.add_inventory(cd_id, cd_title, cd_artist))
```

3. Then I modified the show_inventory() method to display the CD objects in the list of CD objects so user can see what is entered and what is in the data.

```
253     @staticmethod
254     def show_inventory(table):
255         """Displays CD objects in memory, only shows the attributes
263
264         print('======= The Current Inventory: =======')
265         print('ID\tCD Title (by: Artist)\n')
266
267         for cd in table:
268             print('{}\t{}\t (by:{})'.format(cd.cd_id, cd.cd_title, cd.cd_artist))
```

4. Now the program can save the data to the designated txt file. Since a txt file can only store string data type, a conversion is made so that only the attributes of the CDs are saved to the file.

```
118     @staticmethod
119     def save_inventory(lst_Inventory, file_name):
120
121         """Function to manage data ingestion from a list to a file
130         
131         with open(file_name, 'w') as objFile:
132             for cd in lst_Inventory:
133                 objFile.write("{},{},{}'\n".format(cd.cd_id, cd.cd_title, cd.cd_artist))
134
```

5. Then it is to read the data from the txt file. First it reads in the list of strings.

```
90      @staticmethod
91      def load_inventory(file_name):
92          """Read data from file identified by file_name into a list.
00
01          try:
02              with open(file_name, 'r') as objFile:
03                  table = []
04                  for line in objFile:
05                      cd = line.strip().split(',')
06                      table.append(cd)
07                  return table
08          # try-except to prevent program crash if specified file can't be found due to such as
09          # filename error, file not created, file removed
10          except FileNotFoundError:
11              print("\n\nFile {} was not found and was not able to be loaded\n".format(file_name))
12              # below is necessary otherwise table = None when except is invoked
13              # this function won't crash but the future code will crash
14              table = []
15              return table
```

Since each line of the strings contains the attributes of one CD, it then creates CD objects based on the attributes of each CD. To do this, a conversion method is created.

```
270     @staticmethod
271     def textlist_to_CDlist(table):
272         """Convert list of CD attributes strings read from saved file to list of CD objects
273 
274         Args:
275             table: list of CD attributes read from saved file.
276 
277         Returns:
278             table: list of CD objects.
279         """
280 
281         cd =[]
282         for line in table:
283             cd_id, cd_title, cd_artist = line
284             cd.append(CD(cd_id, cd_title, cd_artist))
285         return cd
```

The two work together when read the data in.

```
300     # 1. When program starts, read in the currently saved Inventory, convert CD attributes to CD
301     lstOfCDObjects = FileIO.load_inventory(strFileName)
302     # print(lstOfCDObjects) # print list of strings of attributes of CDs in inventory
303     lstOfCDObjects = IO.textlist_to_CDlist(lstOfCDObjects)
304     # print(lstOfCDObjects) # print list of CD object after the conversion on line 303
```

6. Loading is very similar to reading and previous assignment, and not too much modification is needed.

```
318         # 3.2 process load inventory
319         if strChoice == 'l':
320             print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-lo
321             strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will
322             if strYesNo.lower() == 'yes':
323                 print('reloading...')
324                 lstOfCDObjects = FileIO.load_inventory(strFileName)
325                 lstOfCDObjects = IO.textlist_to_CDlist(lstOfCDObjects)
326                 IO.show_inventory(lstOfCDObjects)
327 
328             else:
329                 input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the
330                 IO.show_inventory(lstOfCDObjects)
331             continue  # start loop back at top.
```

7. I added the delete a CD function that is not required per the assignment, in general it works similar to the previous assignment, only some minor modifications are needed on line 64. I tested it and it worked fine, so I hope I didn't miss something big.

```
48      @staticmethod
49      def del_inventory(IDDel, table):
50          """delete a CD object based on user inputs CD ID
51
52          Args:
53              IDDel: user input CD ID for the CD to be deleted
54              table: current list of CD objects in memory
55
56          Returns:
57              the new list of CD objects
58          """
59
60          intRowNr = -1
61          blnCDRemoved = False
62          for cd in table:
63              intRowNr += 1
64              if int(cd.cd_id) == IDDel:
65                  del table[intRowNr]
66                  blnCDRemoved = True
67                  break
68          if blnCDRemoved:
69              print('The CD was removed')
70          else:
71              print('Could not find this CD!')
72
73          return table
```

8. The rest of the program works very similar to the previous assignment so I won't repeat.

This concludes the tasks of the assignment.

Summary

Running the script in Spyder:

Figure 1 on the next few pages show a demonstration of the following operations a user performs. The error handling and dealing with duplicate records work very similar as assignment07 and were tested but not shown as they are not the focus of this assignment:

1. Start the program (a system message returns for non-existence of specified file)
2. Add a new CD
3. Add another new CD
4. Delete the second entry
5. Save it to the inventory list file
6. Add a 3rd CD
7. Load the saved inventory (the 3rd CD added in 6 is overwritten)
8. Exit the program

Figure 1

IPython console

Console 1/A

```
In [9]: runfile('C:/Users/wdang/Documents/UW Python Course 2020/UW Py
UW Python Course/Jan-Mar Intro/wk8/Mod_08/assignment08')


File cdInventory.txt was not found and was not able to be loaded

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 a numerical ID: 1

What is the CD's title? Bad

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

1       Bad      (by:MJ)
Menu
```

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


Enter a numerical ID: 2

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
```

```
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
```

```
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]: a


Enter a numerical ID: 3

What is the CD's title? Better

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

1       Bad      (by:MJ)
3       Better   (by:WD)
Menu
```

```
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       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]: x

Goodbye!

In [10]:
```

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

Figure 2

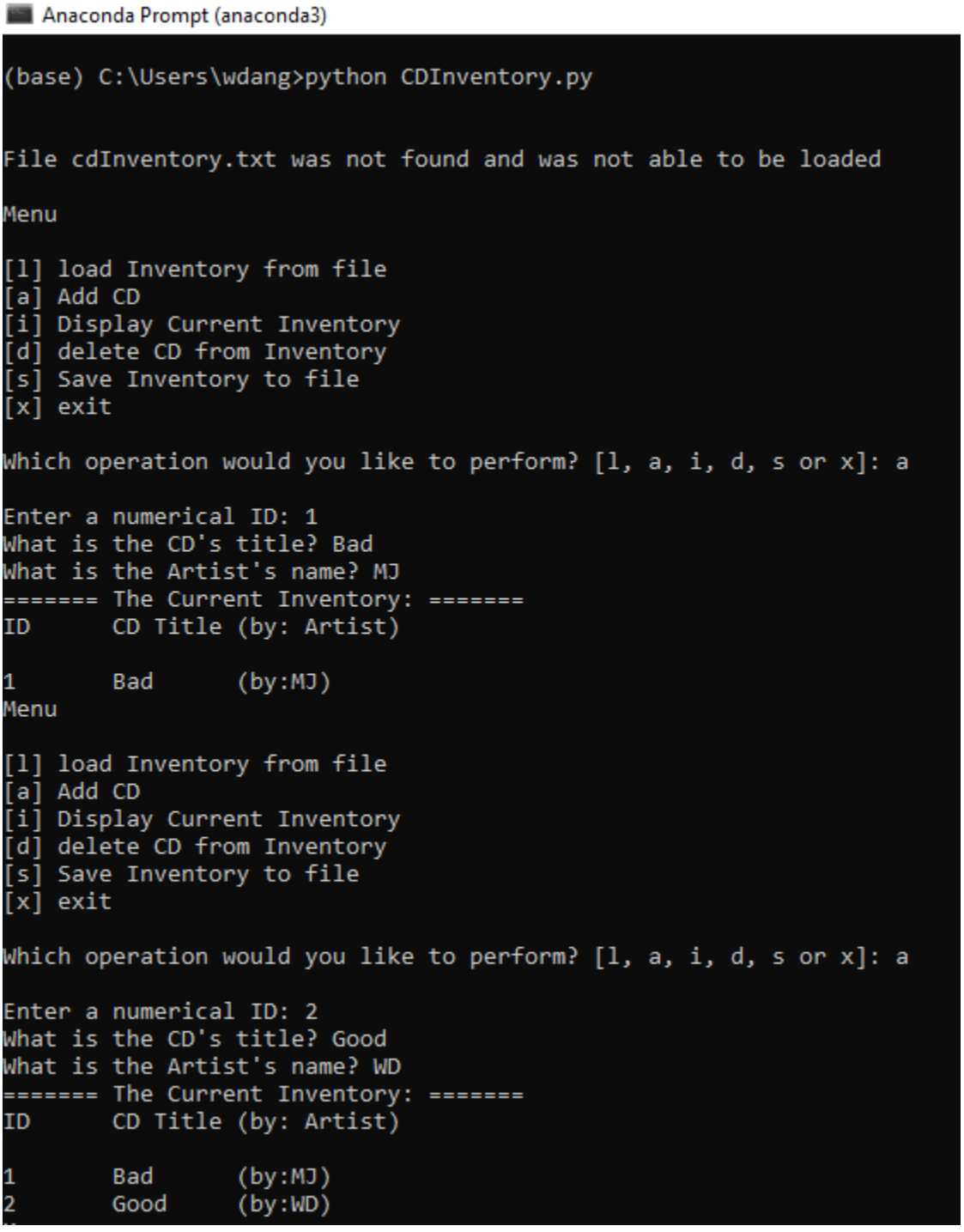

Figure 2 continues:

```
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
```

```
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]: a

Enter a numerical ID: 3
What is the CD's title? Better
What is the Artist's name? WD
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Bad      (by:MJ)
3       Better   (by:WD)
Menu
```

```
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       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]: x

Goodbye!

(base) C:\Users\wdang>
```

Lessons learned:

To be honest, I had no clue at first what I am suppose to do with this assignment. I had one version of the code working but very limited to creating just one new CD object in the add CD functionality. It took me quite a while to realize and understand what it meant the global variable of lstOfCDObjects. Then it took me another while to figure out the distinction between the saved data should be the attributes of the CD objects, while the programs process CD objects, therefore, a conversion of the two object or data type needs to make the program work.

Appendix – copy of the script

```
#------------------------------------------#
# Title: CDInventory.py
# Desc: Assignnment 08 - Working with classes
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, created file
# DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08
# WDang, 2020-Mar-15, modified code
#------------------------------------------#

# -- DATA -- #
strFileName = 'cdInventory.txt'
lstOfCDObjects = []

class CD:
    """Stores data about a CD:

    properties:
        cd_id: (int) with CD ID
        cd_title: (string) with the title of the CD
        cd_artist: (string) with the artist of the CD
    methods:
        add_inventory(cd_id, cd_title, cd_artist, table): -> modified list of CD
objects
        show_inventory(table): -> none
        textList_to_CDlist: -> convert list of CD attributes to list of CD
objects
    """

    def __init__(self, cd_id, cd_title, cd_artist):
        self.cd_id = cd_id
        self.cd_title = cd_title
        self.cd_artist = cd_artist


    @staticmethod
    def add_inventory(cd_id, cd_title, cd_artist):
        """create a new CD object based on user inputs

        Args:
            each of the three cd attributes

        Returns:
            the new CD object
        """

        new_cd = CD(cd_id, cd_title, cd_artist)
        return new_cd


    @staticmethod
    def del_inventory(IDDel, table):
        """delete a CD object based on user inputs CD ID

        Args:
```

```
            IDDel: user input CD ID for the CD to be deleted
            table: current list of CD objects in memory

        Returns:
            the new list of CD objects
        """

        intRowNr = -1
        blnCDRemoved = False
        for cd in table:
            intRowNr += 1
            if int(cd.cd_id) == IDDel:
                del table[intRowNr]
                blnCDRemoved = True
                break
        if blnCDRemoved:
            print('The CD was removed')
        else:
            print('Could not find this CD!')

        return table



# -- PROCESSING -- #
class FileIO:
    """Processes data to and from file:

    properties:

    methods:
        save_inventory(file_name, lst_Inventory): -> None
        load_inventory(file_name): -> (a list of CD objects)

    """

    @staticmethod
    def load_inventory(file_name):
        """Read data from file identified by file_name into a list.

        Args:
            file_name (string): name of file used to read the data from

        Returns:
            a list of CD attributes of each CD.
        """

        try:
            with open(file_name, 'r') as objFile:
                table = []
                for line in objFile:
                    cd = line.strip().split(',')
                    table.append(cd)
                return table
        # try-except to prevent program crash if specified file can't be found
due to such as
```

```
        # filename error, file not created, file removed
        except FileNotFoundError:
            print("\n\nFile {} was not found and was not able to be
loaded\n".format(file_name))
            # below is necessary otherwise table = None when except is invoked
            # this function won't crash but the future code will crash
            table = []
            return table


    @staticmethod
    def save_inventory(lst_Inventory, file_name):

        """Function to manage data ingestion from a list to a file

        Args:
            table: list of CD objects in memory
            file_name: name of file used to save the data to

        Returns:
            None. Create a txt file containing list of CD attributes.
        """

        with open(file_name, 'w') as objFile:
            for cd in lst_Inventory:
                objFile.write("{},{},{}'\n".format(cd.cd_id, cd.cd_title,
cd.cd_artist))



# -- PRESENTATION (Input/Output) -- #
class IO:
    """Data inputs and outputs

    properties:

    methods:
        print_menu(): -> None
        menu_choice(): -> None
        user_input_to_add_inventory(table): -> three CD attritutes for user
create CD
        get_new_cd_id(table): -> new unused CD ID
        get_typed_input(input_message, error_message): -> correct numerical
formatted CD ID input

    """

    @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 = input('Which operation would you like to perform? [l, a, i, d, s
or x]: ').lower().strip()
        while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
            print("'{} is not a valid operation".format(choice))
            choice = input('Which operation would you like to perform? [l, a, i,
d, s or x]: ').lower().strip()
        print()  # Add extra line for layout
        return choice


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

        Args:
            None.

        Returns:
            cd_id (int): ID for the new CD
            cd_title (string): Title for the new CD
            cd_artist (string): Artist of the new CD

        """

        cd_id = IO.get_new_cd_id(table)
        cd_title = input('What is the CD\'s title? ').strip()
        cd_artist = input('What is the Artist\'s name? ').strip()
        return cd_id, cd_title, cd_artist

    @staticmethod
    def get_new_cd_id(table):
        """ Gets a new, unused CD ID from the user

        Args:
            table (list of dict): CDInventory

        Returns:
            cd_id (int): New unused CD ID specified by the user
```

```
        """
        used_ids = []
        for cd in table:
            used_ids.append(cd.cd_id)

        while True:
            cd_id = IO.get_typed_input('Enter a numerical ID: ', 'The entered ID
is not an integer. Please enter a number')
            if cd_id in used_ids:
                print("CD ID '{}' already exsits, use a different
ID\n".format(cd_id))
            else:
                return cd_id


    @staticmethod
    def get_typed_input(input_message, error_message):
        """Prompts the user for input and checks for the correct type.

        Prompts the user for input value of the int type displaying
input_message.
        Checks for correct input type, looping until a proper type is entered
        and displays the passed error_message if bad input is passed.

        Args:
            input_message (string): Message displayed to the user prompting for
input
            error_message (string): Error message displayed to the user if an
incorrect type is entered

        Returns:
            typed_value (int): Int value entered by the user

        """

        while True:
            try:
                typed_value = int(input(input_message).strip())
                return typed_value
            except ValueError:
                print(error_message)


    @staticmethod
    def show_inventory(table):
        """Displays CD objects in memory, only shows the attributes

        Args:
            table: list of CD objects in memory.

        Returns:
            None.
        """

        print('======= The Current Inventory: =======')
        print('ID\tCD Title (by: Artist)\n')
```

```
        for cd in table:
            print('{}\t{}\t (by:{})'.format(cd.cd_id, cd.cd_title, cd.cd_artist))

    @staticmethod
    def textlist_to_CDlist(table):
        """Convert list of CD attributes strings read from saved file to list of
CD objects

        Args:
            table: list of CD attributes read from saved file.

        Returns:
            table: list of CD objects.
        """

        cd =[]
        for line in table:
            cd_id, cd_title, cd_artist = line
            cd.append(CD(cd_id, cd_title, cd_artist))
        return cd


# -- Main Body of Script -- #
# Load data from file into a list of CD objects on script start
# Display menu to user
    # show user current inventory
    # let user add data to the inventory
    # let user save inventory to file
    # let user load inventory from file
    # let user exit program
    # let user delete data from the inventory (added by WDang)

def main():

    # 1. When program starts, read in the currently saved Inventory, convert CD
attributes to CD objects
    lstOfCDObjects = FileIO.load_inventory(strFileName)
    # print(lstOfCDObjects) # print list of strings of attributes of CDs in
inventory
    lstOfCDObjects = IO.textlist_to_CDlist(lstOfCDObjects)
    # print(lstOfCDObjects) # print list of CD object after the conversion on
line 303

    # 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('Goodbye!')
            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...')
                lstOfCDObjects = FileIO.load_inventory(strFileName)
                lstOfCDObjects = IO.textlist_to_CDlist(lstOfCDObjects)
                IO.show_inventory(lstOfCDObjects)

            else:
                input('canceling... Inventory data NOT reloaded. Press [ENTER] to
continue to the menu.')
                IO.show_inventory(lstOfCDObjects)
            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 and add to CD
inventory table
            cd_id, cd_title, cd_artist =
IO.user_input_to_add_inventory(lstOfCDObjects)
            lstOfCDObjects.append(CD.add_inventory(cd_id, cd_title, cd_artist))
            # 3.3.2 Display the updated inventory list
            IO.show_inventory(lstOfCDObjects)
            continue  # start loop back at top.
        # 3.4 process display current inventory
        elif strChoice == 'i':
            IO.show_inventory(lstOfCDObjects)
            continue  # start loop back at top.
        # 3.5 process delete a CD, note not in the scope of this assignment
        elif strChoice == 'd':
            IO.show_inventory(lstOfCDObjects)
        # 3.5.2.1 get user input for which CD to delete
        # 3.5.2 2 search thru table and delete CD if found
            intIDdel = IO.get_typed_input('Which ID would you like to delete? ',
"ID is not valid, please enter an interger value.")
            CD.del_inventory(intIDdel, lstOfCDObjects)
        # 3.5.3 display updated Inventory to user
            IO.show_inventory(lstOfCDObjects)

            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(lstOfCDObjects)
            strYesNo = input('Save this inventory to file? [y/n]
').strip().lower()
            # 3.6.2 Process choice
            if strYesNo == 'y':
                # 3.6.2.1 save CD objects attriutes as strings
                FileIO.save_inventory(lstOfCDObjects, strFileName)
            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')


# start main program

main ()
```