Daisy Pandey
September 1, 2020
Foundations of Programming: Python
Assignment 08

CD Inventory Program Python Script: Object-Oriented Programming

Introduction

Module 8 introduces what is object-oriented programming (OOP). It primarily covers Classes and Objects, Constructors, Fields, Attributes, and Methods. The module also briefly emphasizes on using classes and how to include docstrings for classes.

This document provides the steps I took to create the CD Inventory program using object-oriented programming. I created classes, added code, and used constructor, attributes, methods, and properties.

Creating the CD Inventory Python Script

To create the CD Inventory Python script, I used the Assignment_08 starter script. First, I saved the script and used the Spyder as IDE. Then, I created a header that includes a few comments that list the title of the program, describes what the program is about, and stores the history of changelogs with the name of programmer, date modified, and a brief description of changes (Figure 1).

```
#------------------------------------------#
# Title: CD_Inventory.py
# Desc: Assignnment 08 - Working with classes
# Change Log: (Who, When, What)
# Daisy Pandey, August 31, 2020, created file
# Daisy Pandey, August 31, 2020, added pseudocode
# Daisy Pandey, September 1, 2020, Added code to class CD, class FileIO, and class IO, Created and added code to class DataProcessor
# Daisy Pandey, September 1, 2020, Added code to main body of script, used pickle module to store and load information as binary information
# Daisy Pandey, September 1, 2020, Added structured error handiling to handle errors, updated docstrings
#------------------------------------------#
```

Figure 1. Header with Comments

- **Added code to the CD class:** First, I created a constructor to initialize the component of the CD class. The __init__() method is called the constructor in Python and is always called when an object is created. Second, I added attributes (intId, strTitle, strAtrist), so when an object is created, they are initialized through the constructor method. Then, I created properties. Getters allow to access the private attributes and setters allow to set the value to private attributes. Also, __str__() method returns a string representation of the object (Figure 2).

https://www.datacamp.com/community/tutorials/property-getters-setters?utm_source=adwords_ppc&utm_campaignid=1565261270&utm_adgroupid=67750485268&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=332661264374&utm_targetid=aud-299261629574:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=9033271&gclid=Cj0KCQjwv7L6BRDxARIsAGj-34qRG_8TZtTOpXIz-J_L_8oEOoxPPVZMX8NL0RqRWOShLRJc9Ypozj0aAkFCEALw_wcB [1] (external site)

[1] Retrieved September 1, 2020

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

    properties:
        cdId: (int) with CD ID
        cdTitle: (string) with the title of the CD
        cdArtist: (string) with the artist of the CD

    methods:
        getters: allows to access the private attributes
        setters: allows to set the value to private attributes
        __str__(): returns a string representation of the object
    """
    # Initialize Component of the CD class
    def __init__(self, intId, strTitle, strAtrist):

        # private variables
        self.__cdId = int(intId)
        self.__cdTitle = strTitle.title()       #title() -> Return a version of the string where each word is titlecased.
        self.__cdArtist = strAtrist.title()


    # Getters and Setters Properties Decorators for CD id, title, and artist
    # getter method to get the properties using an object
    @property
    def cdId (self):
        return self.__cdId
    # setter method to change the value 'id' using an object
    @cdId.setter
    def cdId (self, id):
        self.__cdId = int(id)

    # getter method to get the properties using an object
    @property
    def cdTitle (self):
        return self.__cdTitle
    # setter method to change the value 'title' using an object
    @cdTitle.setter
    def cdTitle (self, title):
        self.__cdTitle = title

    # getter method to get the properties using an object
    @property
    def cdArtist (self):
        return self.__cdArtist
    # setter method to change the value 'artist' using an object
    @cdArtist.setter
    def cdArtist (self, artist):
        self.__cdArtist = artist

    def __str__(self):
        return f'{self.cdId}\t{self.cdTitle} (by:{self.cdArtist})'
```

Figure 2. Adding Code to the CD Class

- **Added code to class FileIO and Used pickle module:**
- **FileIO:** This class object processes the data to and from a binary file. It groups two functions: read_file() and write_file().
    - **read_file():** This function loads the current CD data from a binary file into memory. It also uses a parameter (file_name) that allows the function to receive values for processing.
    - **write_file():** The function allows to write the data to a binary file. Like read_file() function, it also uses two parameters (file_name and table) that allow the function to receive values for processing.

First, I imported the pickle module. The pickle module allows to preserve and store more complex data in a binary file.

Next, I added code to the class FileIO to process data to and from file. To read data from a binary file and unpickle it, I used open built-in function with “rb” mode and pickle.load() function (Figure 3). The open () function allows to access a file and the “rb” mode reads data from a binary file. The *r* stands for read mode and the *b* stands for binary mode. The pickle.load() function unpickles and loads the data from the file back into the memory.

```python
            # Load exisitng data from binary file
            with open(file_name, 'rb') as objFile:
                data = pickle.load(objFile)
                return data
```

Figure 3. Reading Data from a File and Unpickling It

In addition, similar to reading data from a binary file, I used open() function with “wb” mode and pickle.dump() function (Figure 4). The open() function opens the file for writing and the “wb” mode writes data to a binary file. The *w* stands for write mode and *b* stands for binary mode. The pickle.dump() function pickles and stores data in the file.

```
155         # Save data to a binary file
156         with open(file_name, 'wb') as objFile:
157             pickle.dump(table, objFile)
```

Figure 4. Pickling Data and Writing to a File

https://www.datacamp.com/community/tutorials/pickle-python-tutorial [2] (external site)

- **Created class DataProcessor and added code:** DataProcessor class object groups add_data() function. It allows to add CD data to the inventory.
    - **add_data():** The function adds data to the 2D table (list of CD objects). It receives its values through four parameters (strID, strTitle, stArtist, table). Then, I instantiated an object called “addCd” of the CD class (Figure 5).

```
103         # Add item to the table
104         cdId = strID
105         cdTitle = strTitle
106         cdArtist = strArtist
107         addCd = CD(cdId, cdTitle, cdArtist)
108         table.append(addCd)
109         return table
```

Figure 5. Adding Item to the Table

- **Added code to IO class:** The IO class object handles inputs and outputs (I/O). It groups four functions: print_menu(), menu_choice(), show_inventory(), and get_userInput().
    - **print_menu():** The function displays a menu of choices to the user. The menu structure includes adding CD data, loading inventory from a file, viewing the current inventory, storing data to a text file, and exiting the program.
    - **menu_choice():** The function gets user input for menu selection. It returns the choice (user-input) as a string type through the return statement.
    - **show_inventory():** The function displays the current inventory data.
    - **get_userInput():** The function gets user input for ID, CD title, and CD artist and returns three values (strID, strTitle, stArtist) through the return statement.

- **Added structured error handling:**

To ensure the program does not crash and handle exceptions (errors), I have used the *try* statement with an *except* clause. The *try* statement generates an exception and an except clause enables you to handle the error with a user-defined response.

https://docs.python.org/3/library/exceptions.html [3] (external file)

The following are the structured error handling that is added to the CD Inventory program.

[2] Retrieved August 23, 2020
[3] Retrieved August 23, 2020

- ValueError exception using try-except block: Handles exception for negative numbers and string values. The program results in ValueError exception if the user input is a string value, zero or a negative number. The print() function only executes messages if a ValueError is raised.

- FileNotFoundError exception using try-except block: Handles exception if a file does not exist in the directory. The FileNotFoundError exception is raised when trying to open a binary file to read if the file does not exist in the directory. If the error occurs, then the program creates a new binary file in the directory and stores data by calling the write_file() function.

I have also added code to the main body of the script. Moreover, I have updated docstrings and included docstrings for classes.

The *while* loop with continue is used to create the program for multiple inputs. The loop repeats a statement or group of statements while the condition is TRUE. Also, the program includes an *if* statement with the use of the *elif* clause.

Running Python Script in Spyder

First, I executed the script in Spyder to ensure all the options in the script were functioning (Figures 6, 7, 8). In addition, I located the cdInventory.dat file, opened it in a text editor, and verified the program was saving the data to the file correctly (Figure 9).

```
In [2]: runfile('C:/Users/daisy/CD_Inventory.py', wdir='C:/Users/daisy')
Menu

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


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


Enter ID: 1

What is the CD's title? Smile

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

1       Smile (by:Katy Perry)
=======================================
Menu

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


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


Enter ID: 2

What is the CD's title? V

What is the Artist's name? Maroon 5
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
=======================================
Menu
```

Figure 6. Running script in Spyder with add (a) option

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

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

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
======================================
Menu

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


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

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

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
======================================

Save this inventory to file? [y/n] y
Data saved to file.

Menu

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


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

Figure 7. Running script in Spyder with display (i) and save (s) options

```
Which operation would you like to perform? [l, a, i, 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       Smile (by:Katy Perry)
2       V (by:Maroon 5)
======================================
Menu

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


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


In [3]:
```

Figure 8. Running script in Spyder with load (l) and exit (x) options

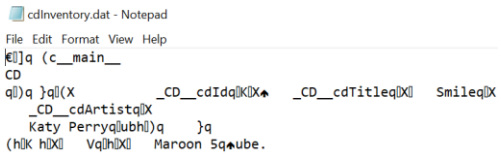

Figure 9. Data Written in a cdInventory.dat File

Running Python Script in Prompt

I reran the script in Anaconda Prompt (Figures 10 and 11) and opened the text editor to ensure the data I had entered in prompt has been written to the file correctly, shown in Figure 12.

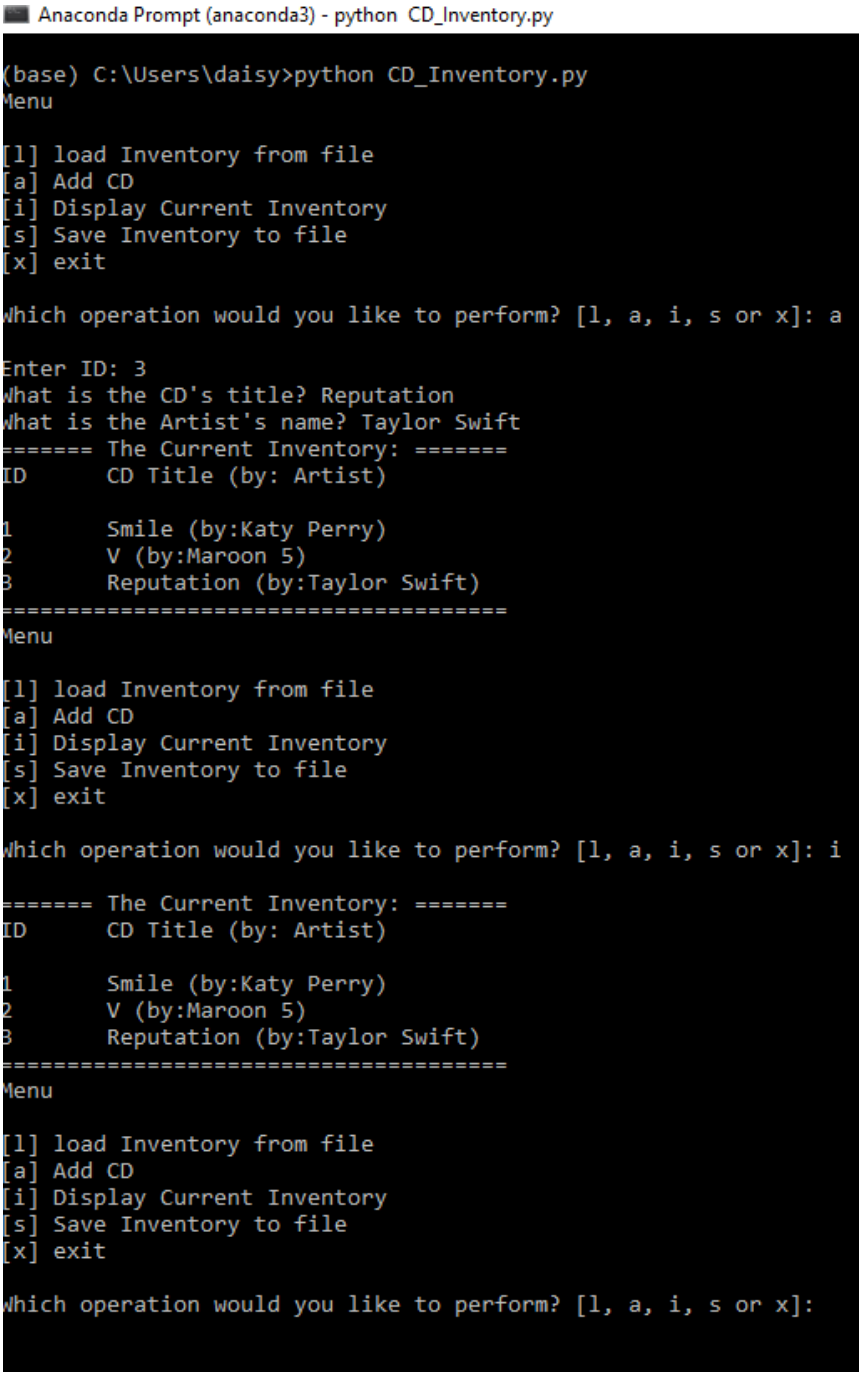

Figure 10. Running script in prompt with add (a) and display (i) options

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

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

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
3       Reputation (by:Taylor Swift)
======================================
Save this inventory to file? [y/n] y
Data saved to file.

Menu

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

Which operation would you like to perform? [l, a, i, 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       Smile (by:Katy Perry)
2       V (by:Maroon 5)
3       Reputation (by:Taylor Swift)
======================================
Menu

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

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


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

Figure 11. Running script in prompt with save (s), load (l), and exit (x) options

cdInventory.dat - Notepad

File Edit Format View Help

```
€]q (c__main__
CD
q)q }q(X          _CD__cdIdq♦K♦X♠   _CD__cdTitleq♦X♦   Smileq♦X
   _CD__cdArtistq♦X
   Katy Perryq♦ubh♦)q    }q
(h♦K h♦X♦   Vq♦h♦X♦   Maroon 5q♠ubh♦)q
}q♦(h♦K♦h♦X
   Reputationq♦h♦X♠   Taylor Swiftq♦ube.
```

Figure 12. Data Written in a cdInventory.dat File

GitHub Account

I have created a repository, “Assignment_08”, where I have uploaded my knowledge document and CD Inventory Python script. The link to my GitHub repository is https://github.com/daisypandey/Assignment_08.

Summary

With Module 8, I learned what is object-oriented programming. After reviewing the Python Programming for the Absolute Beginner Textbook (Chapter 8), FDN_Py_Module_08 pdf document, videos, and few websites, I was able to successfully create the CD Inventory Program Python Script by using object-oriented programming. This assignment demonstrates my knowledge on creating classes to define objects, writing methods and creating attributes for objects, instantiating objects (from classes) and restricting access to an object’s attributes.

Appendix

The following is the final CD Inventory Program Python Script in the Planet B website.
http://www.planetb.ca/syntax-highlight-word [4] (external site)

```
#------------------------------------------#
# Title: CD_Inventory.py
# Desc: Assignnment 08 - Working with classes
# Change Log: (Who, When, What)
# Daisy Pandey, August 31, 2020, created file
# Daisy Pandey, August 31, 2020, added pseudocode
# Daisy Pandey, September 1, 2020, Added code to class CD, class FileIO, and class IO, Created and added code to class DataProcessor
# Daisy Pandey, September 1, 2020, Added code to main body of script, used pickle module to store and load information as binary information
# Daisy Pandey, September 1, 2020, Added structured error handiling to handle errors, updated docstrings
#------------------------------------------#

import pickle

# -- DATA -- #
strFileName = 'cdInventory.dat'
lstOfCDObjects = []     # list of Object to hold data
objFile = None          # file object

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

    properties:
        cdId: (int) with CD ID
        cdTitle: (string) with the title of the CD
        cdArtist: (string) with the artist of the CD

    methods:
        getters: allows to access the private attributes
        setters: allows to set the value to private attributes
        __str__(): returns a string representation of the object
    """
    # Initialize Component of the CD class
    def __init__(self, intId, strTitle, strAtrist):

        # private variables
        self.__cdId = int(intId)
        self.__cdTitle = strTitle.title()       #title() -
> Return a version of the string where each word is titlecased.
        self.__cdArtist = strAtrist.title()


    # Getters and Setters Properties Decorators for CD id, title, and artist
    # getter method to get the properties using an object
    @property
    def cdId (self):
        return self.__cdId
    # setter method to change the value 'id' using an object
    @cdId.setter
    def cdId (self, id):
        self.__cdId = int(id)

    # getter method to get the properties using an object
    @property
    def cdTitle (self):
        return self.__cdTitle
```

[4] Retrieved September 1, 2020

```
    # setter method to change the value 'title' using an object
    @cdTitle.setter
    def cdTitle (self, title):
        self.__cdTitle = title

    # getter method to get the properties using an object
    @property
    def cdArtist (self):
        return self.__cdArtist
    # setter method to change the value 'artist' using an object
    @cdArtist.setter
    def cdArtist (self, artist):
        self.__cdArtist = artist

    def __str__(self):
        return f'{self.cdId}\t{self.cdTitle} (by:{self.cdArtist})'

# -- PROCESSING -- #
class DataProcessor:
    """Adding CD data to the inventory"""

    @staticmethod
    def add_data(strID, strTitle, strArtist, table):
        """Function to add data to the 2D table (list of CD objects)

        Handles ValueError exception type for negative and non-numeric values

        Args:
            StrID (string): Input parameter for CD ID.
            Strtitle (string): Input parameter for CD Title.
            StArtist (string): Input parameter for CD Artist.
            table (list of CD objects): 2D data structure (list of CD objects) that holds the data du
ring runtime

          Returns:
            table (list of CD objects): 2D data structure (list of CD objects) that holds the data du
ring runtime
        """
        # Handling exception for negative numbers and string values
        try:
            intID = int(strID)
            if intID <= 0:
                raise ValueError
        except ValueError:
            print('====Error!!=====')
            print(f'You entered {strID}, which is not a valid entry for ID.')
            print('Please enter a number that is greater than zero.')
                  print()
                  return

               # Add item to the table
               cdId = strID
               cdTitle = strTitle
               cdArtist = strArtist
               addCd = CD(cdId, cdTitle, cdArtist)
               table.append(addCd)
               return table

      class FileIO:
          """Processes data to and from file:

          properties:

          methods:
              write_file(file_name, table): -> None
              read_file(file_name): -> (a list of CD objects)
          """
```

```
121.           @staticmethod
122.           def read_file(file_name):
123.               """Function to manage data ingestion from binary file to a list of CD objects
124.
125.               Reads the data from a binary file identified by file_name into a 2D table (list of CD
   objects)
126.
127.               Args:
128.                   file_name (string): name of file used to read the data from
129.                   table (list of CD objects): 2D data structure (list of CD objects) that holds the
   data during runtime
130.
131.               Returns:
132.                   data
133.               """
134.               # Load exisitng data from binary file
135.               with open(file_name, 'rb') as objFile:
136.                   data = pickle.load(objFile)
137.                   return data
138.
139.           @staticmethod
140.           def write_file(file_name, table):
141.               """Function to save data to a binary file
142.
143.               Writes the data to a binary file identified by file_name into a 2D table (list of CD o
   bjects)
144.
145.               Args:
146.                   file_name(string): name of file used to write the data to
147.                   table(list of CD objects): 2D data structure (list of CD objects) that hold the da
   ta during runtime
148.
149.               Returns:
150.                   None
151.               """
152.               # Save data to a binary file
153.               with open(file_name, 'wb') as objFile:
154.                   pickle.dump(table, objFile)
155.
156.       # -- PRESENTATION (Input/Output) -- #
157.       class IO:
158.           """Handling Input / Output"""
159.
160.           @staticmethod
161.           def print_menu():
162.               """Displays a menu of choices to the user
163.
164.               Args:
165.                   None.
166.
167.               Returns:
168.                   None.
169.               """
170.
171.               print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory
   ')
172.               print('[s] Save Inventory to file\n[x] exit\n')
173.
174.           @staticmethod
175.           def menu_choice():
176.               """Gets user input for menu selection
177.
178.               Args:
179.                   None.
180.
181.               Returns:
182.                   choice (string): a lower case sting of the users input out of the choices l, a, i,
    s or x
```

```
183.                """
184.                choice = ' '
185.                while choice not in ['l', 'a', 'i', 's', 'x']:
186.                    choice = input('Which operation would you like to perform? [l, a, i, s or x]: ').l
    ower().strip()
187.                print()  # Add extra space for layout
188.                return choice
189.
190.            @staticmethod
191.            def show_inventory(table):
192.                """Displays current inventory table
193.
194.                Args:
195.                    table (list of CD objects): 2D data structure (list of CD objects) that holds the
    data during runtime.
196.
197.                Returns:
198.                    None.
199.                """
200.                # Display current inventory
201.                print('======= The Current Inventory: =======')
202.                print('ID\tCD Title (by: Artist)\n')
203.                for row in table:
204.                    print(row)
205.                print('======================================')
206.
207.            @staticmethod
208.            def get_userInput():
209.                """Function to get user input for ID, CD title, and CD artist
210.
211.                Args:
212.                    None.
213.
214.                Returns:
215.                    StrID (string): Input for CD ID.
216.                    Strtitle (string): Input for CD Title.
217.                    StArtist (string): Input for CD Artist.
218.                """
219.                # Ask user for new ID, CD Title and Artist
220.                strID = input('Enter ID: ').strip()
221.                strTitle = input('What is the CD\'s title? ').strip()
222.                strArtist = input('What is the Artist\'s name? ').strip()
223.                return strID, strTitle, strArtist
224.
225.        # -- Main Body of Script -- #
226.        # When program starts, read in the currently saved Inventory if exist, if not create one
227.        # If file does not exist, handle error with FileNotFoundError exception
228.        try:
229.            lstOfCDObjects = FileIO.read_file(strFileName)
230.        except FileNotFoundError:
231.            FileIO.write_file(strFileName, lstOfCDObjects)
232.
233.        # Start main loop
234.        while True:
235.            # Display Menu to user and get choice
236.            IO.print_menu()
237.            strChoice = IO.menu_choice()
238.
239.        # Process menu selection
240.            # Process exit first
241.            if strChoice == 'x':
242.                break
243.
244.            # Process load inventory
245.            if strChoice == 'l':
246.                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.clear()
                    table = FileIO.read_file(strFileName)
                    lstOfCDObjects.extend(table)
                    IO.show_inventory(lstOfCDObjects) # Display Inventory to user
                else:
                    input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
                    IO.show_inventory(lstOfCDObjects)
                continue  # start loop back at top.

            # Process add a CD
            elif strChoice == 'a':
                # Store user inputs
                userInputId, userInputTitle, userInputArtist = IO.get_userInput()

                # Add data to the 2D table (list of CD objects)
                DataProcessor.add_data(userInputId, userInputTitle, userInputArtist, lstOfCDObjects)
                IO.show_inventory(lstOfCDObjects)
                continue  # start loop back at top.

            # Process display current inventory
            elif strChoice == 'i':
                IO.show_inventory(lstOfCDObjects)
                continue  # start loop back at top.

            # Process save inventory to file
            elif strChoice == 's':
                # 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()
                # Process choice
                if strYesNo == 'y':
                    FileIO.write_file(strFileName, lstOfCDObjects) # Calling write_file function
                    print('Data saved to file.')
                    print()
                else:
                    input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')

                continue  # start loop back at top.

            # Catch-all should not be possible, as user choice gets vetted in IO, but to be save:
            else:
                print('General Error')
```