Daisy Pandey
August 26, 2020
Foundations of Programming: Python
Assignment 07

CD Inventory Program Python Script: Binary Files and Structured Error Handling

Introduction

Module 7 introduces binary files and structured error handling. It primarily covers working with binary files, handling exceptions (errors), and the benefits of using structured error handling. The module also briefly emphasizes on creating custom GitHub readme files.

This document provides the steps I took to modify the last week's CD Inventory program. The following are the modifications that are made:

- Modified the permanent data store to use binary data
- Added structured error handling
- Updated docstrings

Modifying the CD Inventory Python Script

To modify the CD Inventory Python script, I used the last week's CD Inventory Program script. First, I saved the script and used the Spyder as IDE. Second, 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: CDInventory.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# Daisy Pandey, August 16, 2020, Modifying CD Inventory Program script
# Daisy Pandey, August 16, 2020, Added code, added new functions, moved existing code to those functions
# Daisy Pandey, August 17, 2020, Added docstrings for add_data, del_data, and write_file functions, modified/added comments
# Daisy Pandey, August 19, 2020, Added get_UserInput function and docstring, added code to check if file exists
# Daisy Pandey, August 21, 2020, Removed dependence on global variables
# Daisy Pandey, August 23, 2020, Modified the code to use pickle module to store and load information as binary information
# Daisy Pandey, August 23, 2020, Added structured error handling to handle errors
# Daisy Pandey, August 24, 2020, Updated docstrings
#------------------------------------------#
```

Figure 1. Header with Comments

Lastly, I made some modifications and organized the code.

- **Modified the permanent data store to use binary data:**

The first thing I did in the program was I imported the pickle module (Figure 2). The pickle module allows to preserve and store more complex data in a binary file.

```
15  import pickle
```

Figure 2. Importing Pickle Module

Next, 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.

```
102         # Load exisitng data from binary file
103         table.clear() # Removes all the existing items from table
104         with open(file_name, 'rb') as objFile:
105             data = pickle.load(objFile)
106             table.extend(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.

```
122         # Save data to a binary file
123         with open(file_name, 'wb') as objFile:
124             pickle.dump(table, objFile)
```

Figure 4. Pickling Data and Writing to a File

https://www.datacamp.com/community/tutorials/pickle-python-tutorial [1] (external site)

- **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 [2] (external file)

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

- ValueError exception using try-except block (Figure 5): 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.

```
42          # Handling exception for negative numbers and string values
43          try:
44              intID = int(strID)
45              if intID <= 0:
46                  raise ValueError
47          except ValueError:
48              print('====Error!!=====')
49              print(f'You entered {strID}, which is not a valid entry for ID.')
50              print('Please enter a number that is greater than zero.')
51              print()
52              return
```

Figure 5. ValueError Exception Using Try-Except Block

- FileNotFoundError exception using try-except block (Figure 6): 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.

```
197 try:
198     FileProcessor.read_file(strFileName, lstTbl)
199 except FileNotFoundError:
200     FileProcessor.write_file(strFileName, lstTbl)
```

Figure 6. FileNotFoundError Exception Using Try-Except Block

- ValueError exception using try-except block (Figure 7): The program results in ValueError exception if the user-input is a non-numeric value in the delete CD menu. The print() function only executes messages if a ValueError is raised.

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

```
244         try:
245             intIDDel = int(input('Which ID would you like to delete? ').strip())
246         except ValueError:
247             print('Oops! That was not a number.')
```

Figure 7. ValueError Exception Using Try-Except Block

- **Updated docstrings:**

I have also updated docstrings for write_file() function and read_file() function to describe what these functions do.

The following are the classes and functions that are defined in the program:

- **DataProcessor:** This class object groups two functions: add_data() and del_data(). It allows to add CD data to the inventory and delete CD data from the inventory.
    - **add_data():** The function adds data to the 2D table (list of dictionaries). It receives its values through four parameters (strID, strTitle, stArtist, table).
    - **del_data():** The function deletes CD data from existing inventory. It accepts the ID to delete and the list table to remove it from.

- **FileProcessor:** 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 parameters (file_name and table) that allow the function to receive values for processing. With .extend() function, it extends the list by appending all the items from the iterable. https://stackoverflow.com/questions/252703/what-is-the-difference-between-pythons-list-methods-append-and-extend [3] (external site)
    - **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.

- **IO:** This 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, deleting an entry from inventory, 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 table returning only the dictionary values.
    - **get_userInput():** The function belongs to the IO class object. It gets user input for ID, CD title, and CD artist and returns three values (strID, strTitle, stArtist) through the return statement.

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.

[3] Retrieved August 23, 2020

Pickling in Python Research

There are plenty of websites and videos that provide information on using Python's pickling feature. The **docs.python.org** gives an in-depth explanation of what is pickling, what is unpickling, and using pickle module to serialize and deserialize objects in Python. Also, it discusses the comparison between the pickle module and other Python modules and explains pickle module functions, pickle module exceptions, and pickle module classes. Moreover, the website highlights what can be pickled and unpickled as well as gives examples.

https://docs.python.org/3.7/library/pickle.html[4] (external site)

I also watched a video, "Python Pickle Module for saving objects (serialization)", that explains how to use Python's pickling feature. However, the video is short, the tutorial gives a great explanation on how to pickle objects in Python.

https://www.youtube.com/watch?v=2Tw39kZIbhs [5] (external site)

Exception Handling in Python Research

There are numerous websites and videos that provide information on how to use Python's exception handling feature. The **datacamp.com** website provides a thorough explanation of exception and error handling in Python as well as its benefits. It also explains errors of various types such as Syntax Error, Out of Memory Error, Recursion Error, and Exceptions. Moreover, the website lists and discusses various built-in exceptions with examples.

https://www.datacamp.com/community/tutorials/exception-handling-python [6] (external site)

The **realpython.org** website is also helpful for explaining Python Exceptions. The article explains what is an exception and the difference between a syntax error and an exception. In addition, it offers an in-depth explanation on handling exceptions with try and except block, raising exceptions, and making assertions. Furthermore, the article includes a few examples.

https://realpython.com/python-exceptions/[7] (external site)

Running Python Script in Spyder

First, I executed the script in Spyder to ensure all the options in the script were functioning (Figures 8, 9, 10, 11, 12). 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 13).

[4] Retrieved August 25, 2020
[5] Retrieved August 25, 2020
[6] Retrieved August 25, 2020
[7] Retrieved August 25, 2020

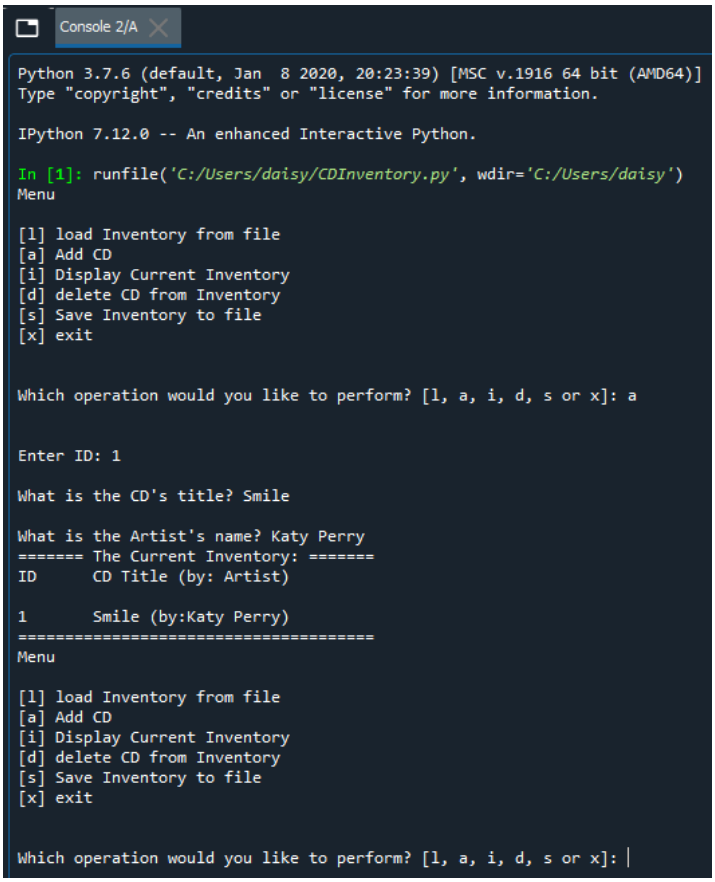

Figure 8. Running script in Spyder with add (a) option

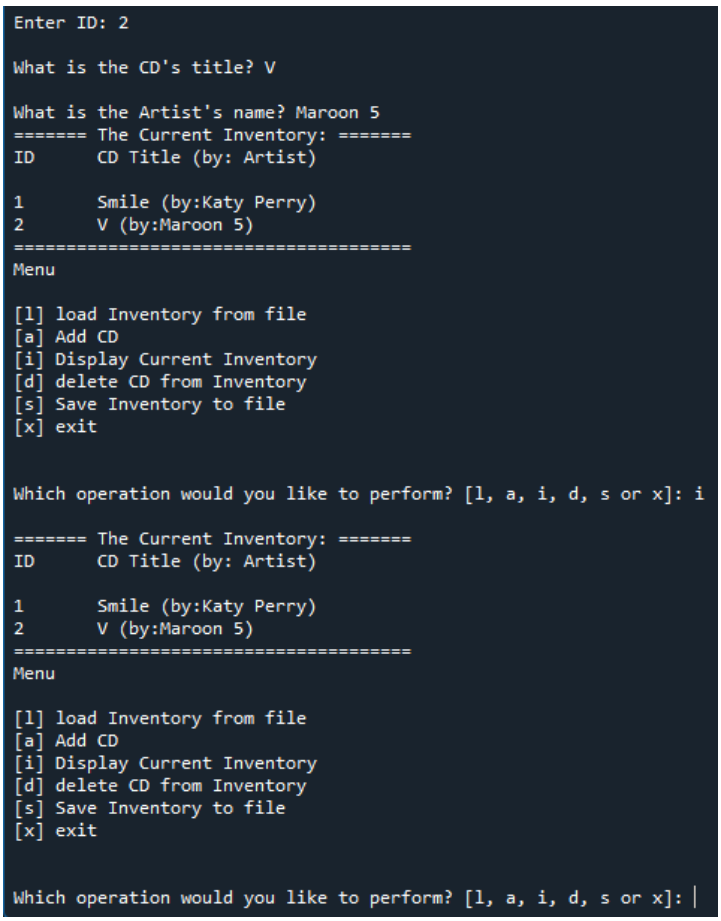

Figure 9. Running script in Spyder with display (i) option

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

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

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
3       A Head Full of Dreams (by:Coldplay)
======================================

Which ID would you like to delete? 3
The CD was removed
======= 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
[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 10. Running script in Spyder with delete (d) option

```
Which operation would you like to perform? [l, a, i, d, 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
[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 11. Running script in Spyder with save (s) option

```
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       Smile (by:Katy Perry)
2       V (by:Maroon 5)
======================================
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 [4]:
```

Figure 12. Running script in Spyder with load (l) and exit (x) options

CDInventory.dat - Notepad

File Edit Format View Help

```
€]q (}q(X    IDq KX   TitleqX   SmileqX   ArtistqX
   Katy Perryqu}q(h K hX   VqhX   Maroon 5q  ue.
```

Figure 13. Data Written in a CDInventory.dat File

Running Python Script in Prompt

I reran the script in Anaconda Prompt (Figures 14-18) and opened the text editor to ensure the data I had entered in prompt has been written to the file correctly, highlighted in Figure 19.

Anaconda Prompt (anaconda3) - python CDInventory.py

```
(base) C:\Users\daisy>python CDInventory.py
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: 3
What is the CD's title? A Head Full of Dreams
What is the Artist's name? Coldplay
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
3       A Head Full of Dreams (by:Coldplay)
======================================
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: 4
What is the CD's title? Reputation
What is the Artist's name? Taylor Swift
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
3       A Head Full of Dreams (by:Coldplay)
4       Reputation (by:Taylor Swift)
======================================
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 14. Running script in prompt with add (a) option

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

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

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
3       A Head Full of Dreams (by:Coldplay)
4       Reputation (by:Taylor Swift)
======================================
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 15. Running script in prompt with display (i) option

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

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

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
3       A Head Full of Dreams (by:Coldplay)
4       Reputation (by:Taylor Swift)
======================================
Which ID would you like to delete? 4
The CD was removed
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
3       A Head Full of Dreams (by:Coldplay)
======================================
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 16. Running script in prompt with delete (d) option

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

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

1       Smile (by:Katy Perry)
2       V (by:Maroon 5)
3       A Head Full of Dreams (by:Coldplay)
======================================
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
[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 17. Running script in prompt with save (s) option

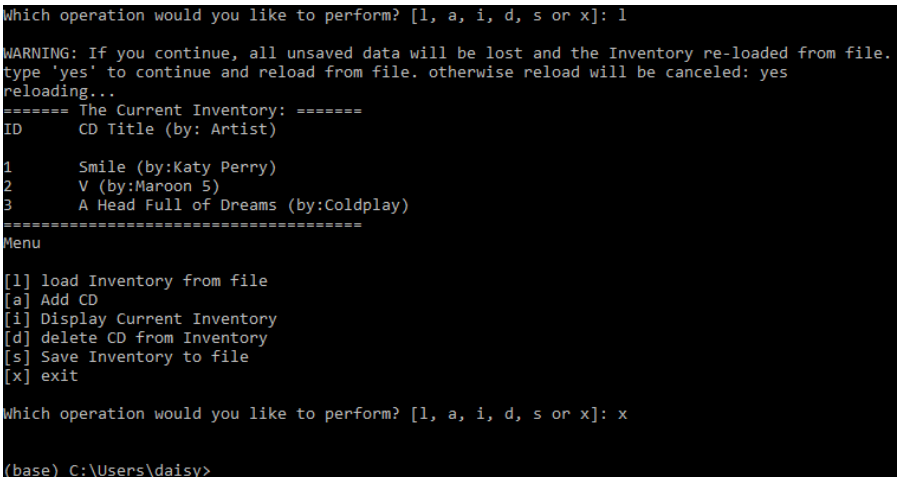

Figure 18. Running script in prompt with load (l) and exit (x) options

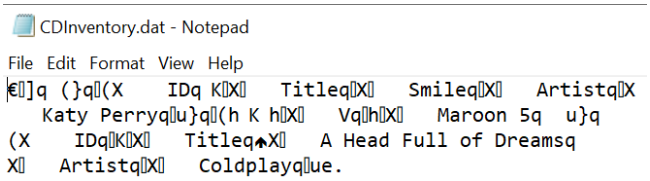

Figure 19. Data Written in a CDInventory.dat File

GitHub Account

I have created a repository, "Assignment_07", where I have uploaded my knowledge document and CD Inventory Python script. The link to my GitHub repository is https://github.com/daisypandey/Assignment_07.

Summary

With Module 7, I learned about binary files and structured error handling. After reviewing the Python Programming for the Absolute Beginner Textbook (Chapter 7), FDN_Py_Module_07 pdf document, videos, and few websites, I was able to successfully modify the CD Inventory Program Python Script by using pickling feature and error handling feature. This assignment demonstrates my knowledge on using structured error handling and reading and writing binary files.

Appendix

The following is the modified and final CD Inventory Program Python Script in the Planet B website. http://www.planetb.ca/syntax-highlight-word [8] (external site)

```
#------------------------------------------#
# Title: CDInventory.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# Daisy Pandey, August 16, 2020, Modifying CD Inventory Program script
# Daisy Pandey, August 16, 2020, Added code, added new functions, moved existing code to those functi
ons
# Daisy Pandey, August 17, 2020, Added docstrings for add_data, del_data, and write_file functions, m
odified/added comments
# Daisy Pandey, August 19, 2020, Added get_UserInput function and docstring, added code to check if f
ile exists
# Daisy Pandey, August 21, 2020, Removed dependence on global variables
```

[8] Retrieved August 25, 2020

```
# Daisy Pandey, August 23, 2020, Modified the code to use pickle module to store and load information as binary information
# Daisy Pandey, August 23, 2020, Added structured error handling to handle errors
# Daisy Pandey, August 24, 2020, Updated docstrings
#------------------------------------------#

import pickle

# -- DATA -- #
strChoice = '' # User input
lstTbl = []  # list of lists to hold data
dicRow = {}  # list of data row
strFileName = 'CDInventory.dat'  # data storage file
objFile = None  # file object

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

    @staticmethod
    def add_data(strID, strTitle, stArtist, table):
        """Function to add data to the 2D table (list of dictionaries)
        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 dict): 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            table (list of dict): 2D data structure (list of dicts) that holds the data during 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
        dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
        table.append(dicRow)
        return table

    @staticmethod
    def del_data(id_to_remove, table):
        """Function to delete CD data from current inventory
        Accepts the ID to delete and the list table to remove it from

        Args:
            id_to_remove (integer): ID to remove CD data
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            None
        """
        # Search thru table and delete CD
        intRowNr = -1
        blnCDRemoved = False
```

```
        for row in table:
            intRowNr += 1
            if row['ID'] == id_to_remove:
                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 binary file"""

    @staticmethod
    def read_file(file_name, table):
        """Function to manage data ingestion from binary file to a list of dictionaries

        Reads the data from a binary 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 runtim
    e

        Returns:
                    None.
                """
                # Load exisitng data from binary file
                table.clear() # Removes all the existing items from table
                with open(file_name, 'rb') as objFile:
                    data = pickle.load(objFile)
                    table.extend(data)

            @staticmethod
            def write_file(file_name, table):
                """Function to save data to a binary file

                Writes the data to a binary 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 write the data to
                    table(list of dict): 2D data structure (list of dicts) that hold the data during r
    untime

                Returns:
                    None
                """
                # Save data to a binary file
                with open(file_name, 'wb') as objFile:
                    pickle.dump(table, objFile)

        # -- PRESENTATION (Input/Output) -- #
        class IO:
            """Handling Input / Output"""

            @staticmethod
            def print_menu():
                """Displays a menu of choices to the user

                Args:
                    None.

                Returns:
                    None.
                """
```

```
140.
141.                 print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory
    ')
142.                 print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')
143.
144.             @staticmethod
145.             def menu_choice():
146.                 """Gets user input for menu selection
147.
148.                 Args:
149.                     None.
150.
151.                 Returns:
152.                     choice (string): a lower case sting of the users input out of the choices l, a, i,
    d, s or x
153.                 """
154.                 choice = ' '
155.                 while choice not in ['l', 'a', 'i', 'd', 's', 'x']:
156.                     choice = input('Which operation would you like to perform? [l, a, i, d, s or x]: '
    ).lower().strip()
157.                 print()  # Add extra space for layout
158.                 return choice
159.
160.             @staticmethod
161.             def show_inventory(table):
162.                 """Displays current inventory table
163.
164.                 Args:
165.                     table (list of dict): 2D data structure (list of dicts) that holds the data during
    runtime.
166.
167.                 Returns:
168.                     None.
169.                 """
170.                 # Display current inventory
171.                 print('======= The Current Inventory: =======')
172.                 print('ID\tCD Title (by: Artist)\n')
173.                 for row in table:
174.                     print('{}\t{} (by:{})'.format(*row.values()))
175.                 print('======================================')
176.
177.             @staticmethod
178.             def get_userInput():
179.                 """Function to get user input for ID, CD title, and CD artist
180.
181.                 Args:
182.                     None.
183.
184.                 Returns:
185.                     StrID (string): Input for CD ID.
186.                     Strtitle (string): Input for CD Title.
187.                     StArtist (string): Input for CD Artist.
188.                 """
189.                 # Ask user for new ID, CD Title and Artist
190.                 strID = input('Enter ID: ').strip()
191.                 strTitle = input('What is the CD\'s title? ').strip()
192.                 stArtist = input('What is the Artist\'s name? ').strip()
193.                 return strID, strTitle, stArtist
194.
195.         # When program starts, read in the currently saved Inventory if exist, if not create one
196.         # If file does not exist, handle error with FileNotFoundError exception
197.         try:
198.             FileProcessor.read_file(strFileName, lstTbl)
199.         except FileNotFoundError:
200.             FileProcessor.write_file(strFileName, lstTbl)
201.
202.         # Start main loop
203.         while True:
```

```
204.           # Display Menu to user and get choice
205.           IO.print_menu()
206.           strChoice = IO.menu_choice()
207.
208.       # Process menu selection
209.           # Process exit first
210.           if strChoice == 'x':
211.               break
212.
213.           # Process load inventory
214.           if strChoice == 'l':
215.               print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-
    loaded from file.')
216.               strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will
     be canceled: ')
217.               if strYesNo.lower() == 'yes':
218.                   print('reloading...')
219.                   FileProcessor.read_file(strFileName, lstTbl)
220.                   IO.show_inventory(lstTbl) # Display Inventory to user
221.               else:
222.                   input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the
    menu.')
223.                   IO.show_inventory(lstTbl)
224.               continue  # start loop back at top.
225.
226.           # Process add a CD
227.           elif strChoice == 'a':
228.               # Store user inputs
229.               userInputId, userInputTitle, userInputArtist = IO.get_userInput()
230.               # Add data to the 2D table (list of dictionaries)
231.               DataProcessor.add_data(userInputId, userInputTitle, userInputArtist, lstTbl)
232.               IO.show_inventory(lstTbl)
233.               continue  # start loop back at top.
234.
235.           # Process display current inventory
236.           elif strChoice == 'i':
237.               IO.show_inventory(lstTbl)
238.               continue  # start loop back at top.
239.
240.           # Process delete a CD
241.           elif strChoice == 'd':
242.               IO.show_inventory(lstTbl)
243.               # Ask user to remove ID
244.               try:
245.                   intIDDel = int(input('Which ID would you like to delete? ').strip())
246.               except ValueError:
247.                   print('Oops! That was not a number.')
248.                   print()
249.                   continue
250.               DataProcessor.del_data(intIDDel, lstTbl) # Deletes data from inventory
251.               IO.show_inventory(lstTbl)
252.               continue  # start loop back at top.
253.
254.           # Process save inventory to file
255.           elif strChoice == 's':
256.               # Display current inventory and ask user for confirmation to save
257.               IO.show_inventory(lstTbl)
258.               strYesNo = input('Save this inventory to file? [y/n] ').strip().lower()
259.               # Process choice
260.               if strYesNo == 'y':
261.                   FileProcessor.write_file(strFileName, lstTbl) # Calling write_file function
262.                   print('Data saved to file.')
263.                   print()
264.               else:
265.                   input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')

266.               continue  # start loop back at top.
267.
```

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