Murat Yokus
August 22nd, 2021
IT FDN 110 B: Introduction to Programming (Python)
Assignment 07

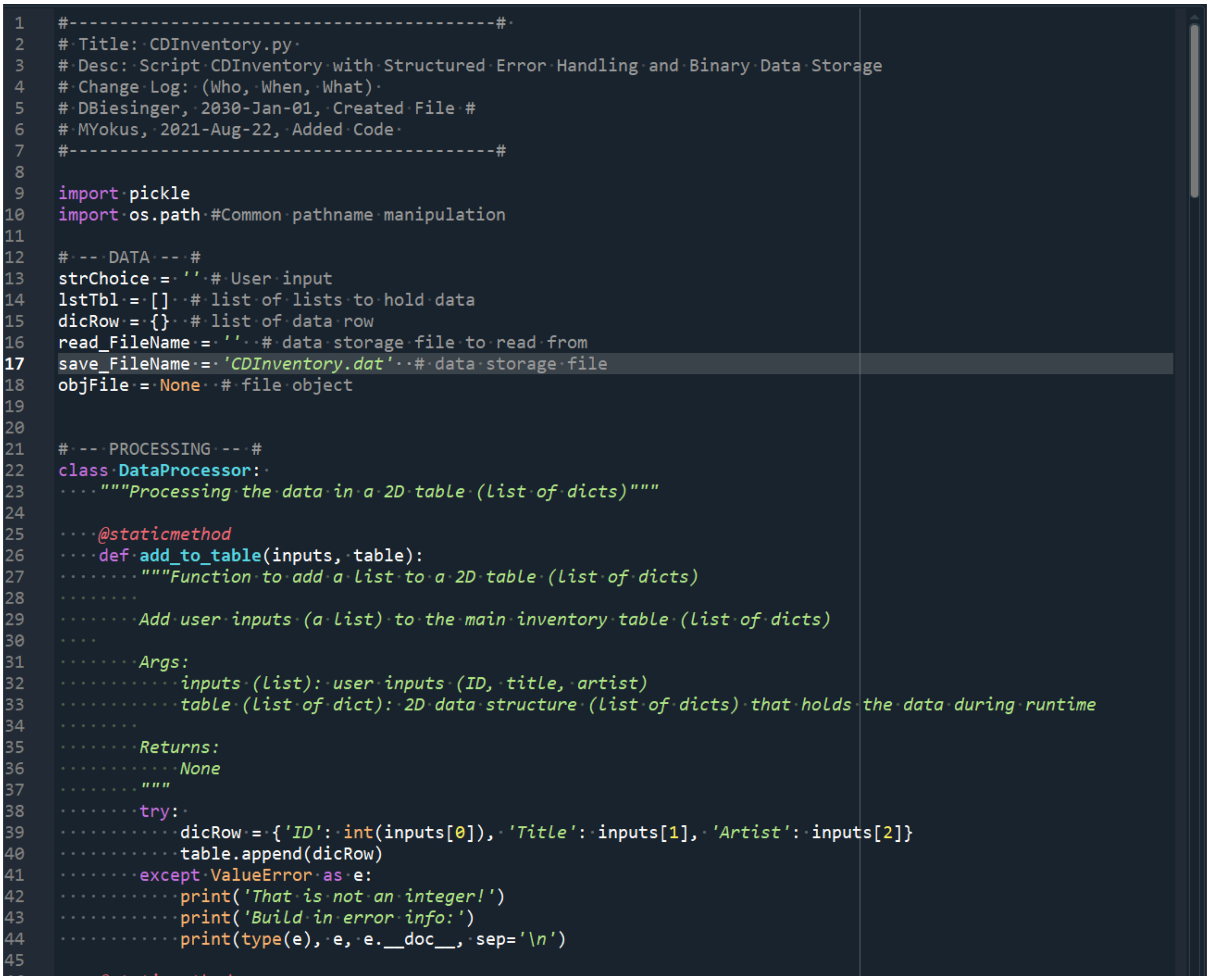
CDInventory.py Script with Error Handling and Binary Data Storage

Introduction

This document outlines the steps required for adding structured error handling around the areas where there is user interaction, type casting (*e.g.,* string to integer), or file access operations. The script is a continuation of Assignment 06 and includes classes and functions to load CD data from CDInventory.dat file, enter CD data, view the current inventory, delete DC data from inventory, save data to CDInventory.dat data file, and exit the program. In contrast to the previous assignment, Assignment 07 uses CDInventory.dat file for both reading and storing the CD inventory binary data. The script was written in the Spyder IDE, and its successful operation was shown in Spyder and anaconda terminal. Finally, the document summarizes my learnings from Module 7.

Steps:

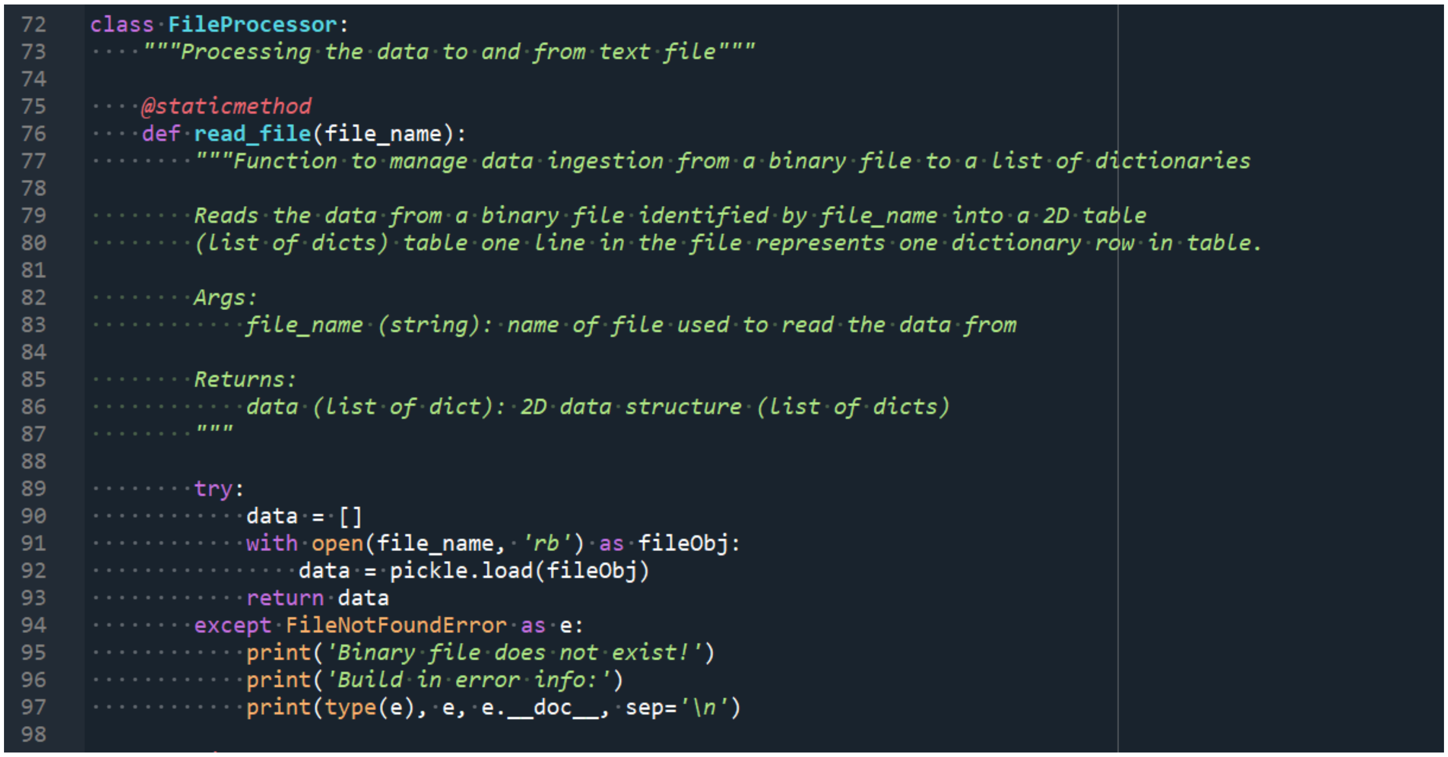
One of the most common errors encountered in this script was the *ValueError a*s shown in **Listing 1**. Whenever user wants to add a new CD entry to the inventory, the CD ID must be an integer. Otherwise, *ValueError* is displayed on the screen. To be able to handle this error type, try-except structured error handling construct was included under the *add_to_table()* function (**Listing 1**, line 39 – 44). When this type of error is encountered, the script warns user to input an integer number.



```
1  #----------------------------------------#
2  # Title: CDInventory.py
3  # Desc: Script CDInventory with Structured Error Handling and Binary Data Storage
4  # Change Log: (Who, When, What)
5  # DBiesinger, 2030-Jan-01, Created File #
6  # MYokus, 2021-Aug-22, Added Code
7  #----------------------------------------#
8
9  import pickle
10 import os.path #Common pathname manipulation
11
12 # -- DATA -- #
13 strChoice = '' # User input
14 lstTbl = []  # list of lists to hold data
15 dicRow = {}  # list of data row
16 read_FileName = ''  # data storage file to read from
17 save_FileName = 'CDInventory.dat'  # data storage file
18 objFile = None  # file object
19
20
21 # -- PROCESSING -- #
22 class DataProcessor:
23     """Processing the data in a 2D table (list of dicts)"""
24
25     @staticmethod
26     def add_to_table(inputs, table):
27         """Function to add a list to a 2D table (list of dicts)
28         
29         Add user inputs (a list) to the main inventory table (list of dicts)
30     
31         Args:
32             inputs (list): user inputs (ID, title, artist)
33             table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
34         
35         Returns:
36             None
37         """
38         try:
39             dicRow = {'ID': int(inputs[0]), 'Title': inputs[1], 'Artist': inputs[2]}
40             table.append(dicRow)
41         except ValueError as e:
42             print('That is not an integer!')
43             print('Build in error info:')
44             print(type(e), e, e.__doc__, sep='\n')
45
```

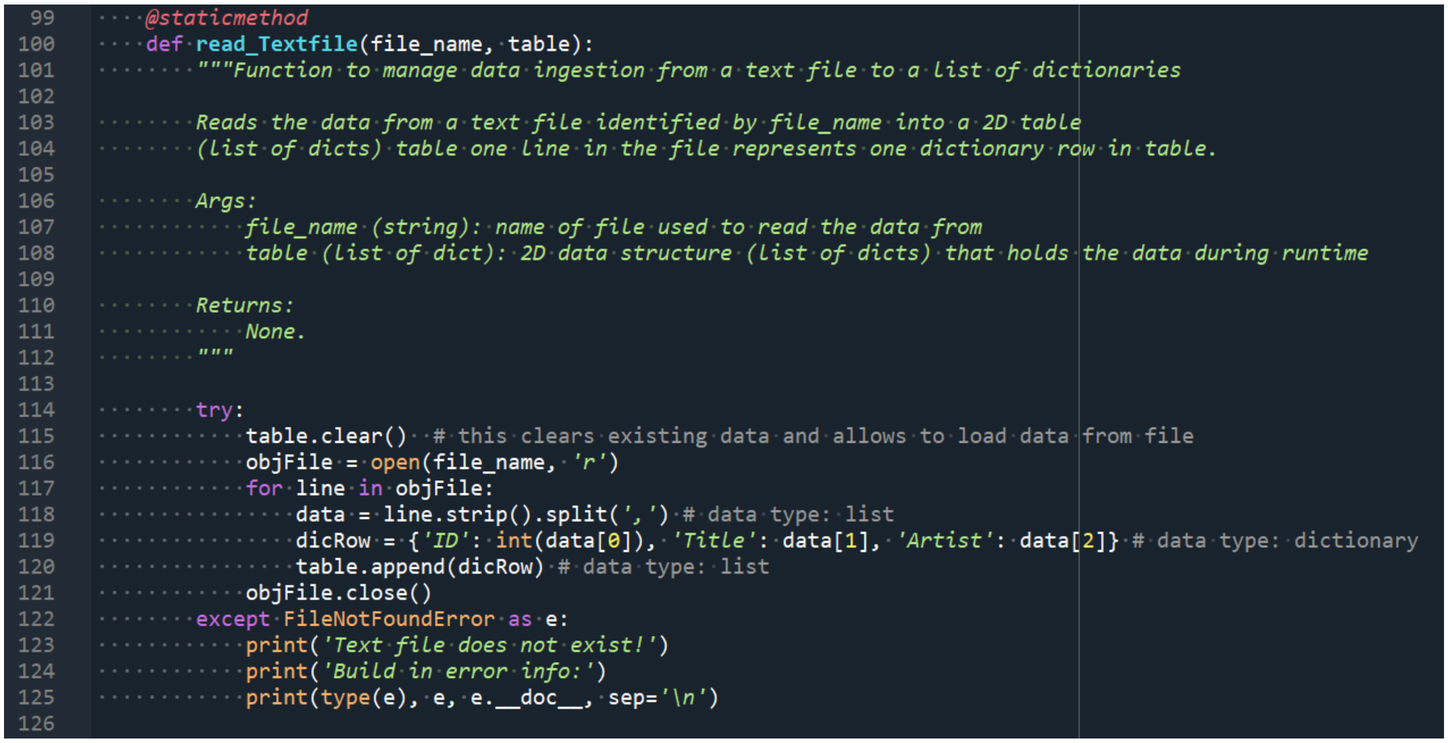
Listing 1- Adding CD entry to the inventory and ValueError handling for CD ID.

Another common type of error encountered in this script was *FileNotFoundError*. The script displays this type of error if the filename does not exist to read data from CDInventory.txt or CDInventory.dat file or to write data to CDInventory.dat file. To be able to handle this error, try-except structured error handling constructs were included under *read_file(),read_Textfile(), and write_file()* functions (**Listing 2**, **3**, and **4**). When this type of error is encountered, the script displays an error to the user similar to the following message "*The file does not exist*".



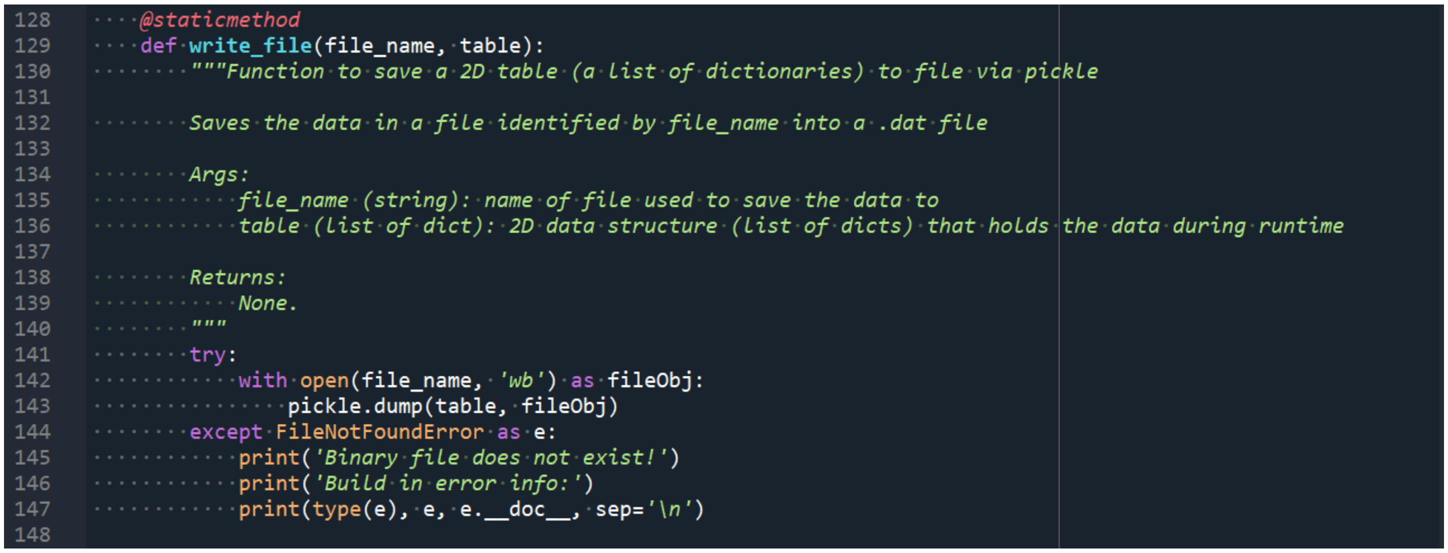
```python
class FileProcessor:
    """Processing the data to and from text file"""

    @staticmethod
    def read_file(file_name):
        """Function to manage data ingestion from a 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

        Returns:
            data (list of dict): 2D data structure (list of dicts)
        """

        try:
            data = []
            with open(file_name, 'rb') as fileObj:
                data = pickle.load(fileObj)
            return data
        except FileNotFoundError as e:
            print('Binary file does not exist!')
            print('Build in error info:')
            print(type(e), e, e.__doc__, sep='\n')

```

Listing 2 – Reading from a binary file and FileNotFoundError handling.



```python
    @staticmethod
    def read_Textfile(file_name, table):
        """Function to manage data ingestion from a text file to a list of dictionaries

        Reads the data from a text 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.
        """

        try:
            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(',') # data type: list
                dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]} # data type: dictionary
                table.append(dicRow) # data type: list
            objFile.close()
        except FileNotFoundError as e:
            print('Text file does not exist!')
            print('Build in error info:')
            print(type(e), e, e.__doc__, sep='\n')

```

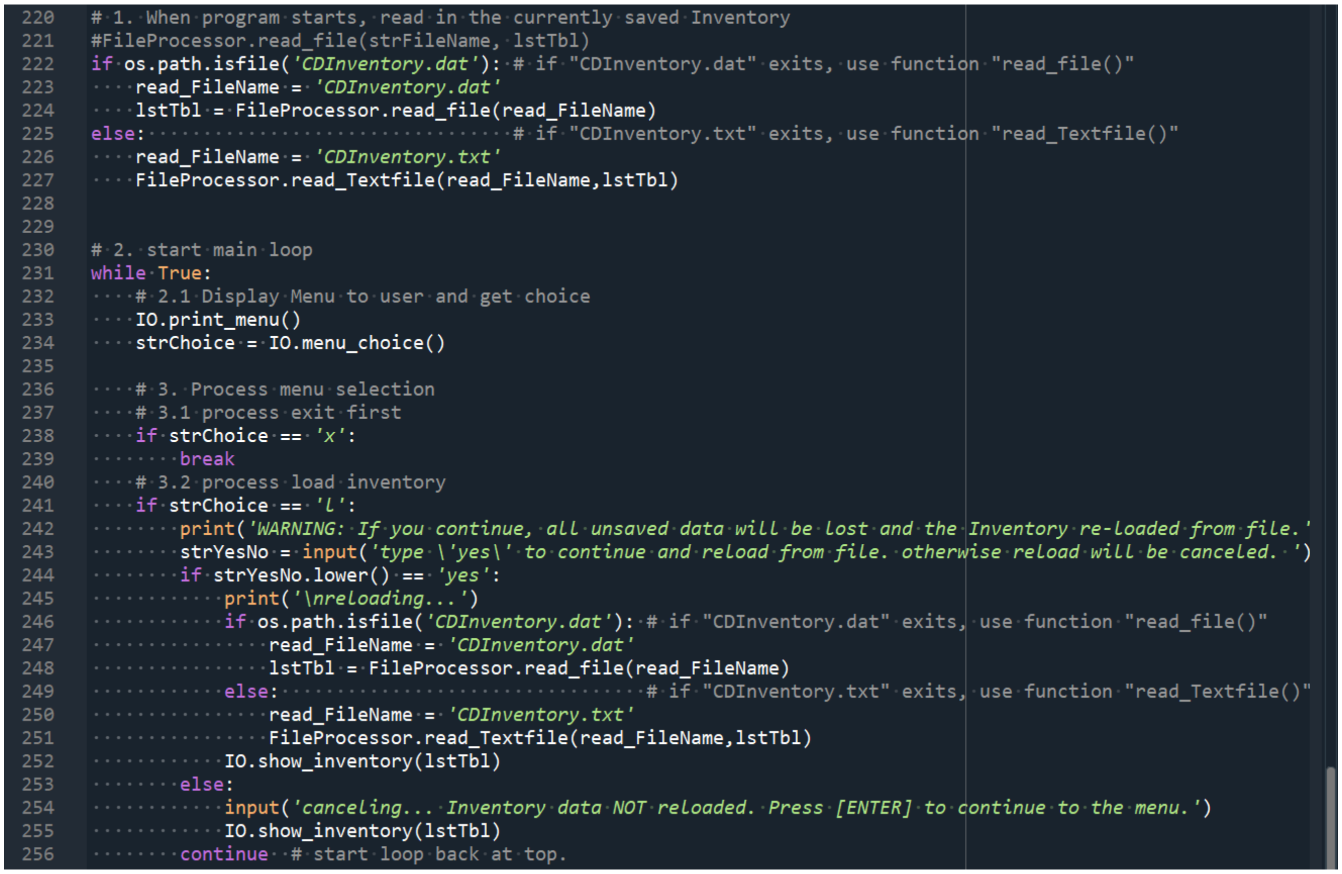
Listing 3 – Reading from a text file and FileNotFoundError handling.



```python
    @staticmethod
    def write_file(file_name, table):
        """Function to save a 2D table (a list of dictionaries) to file via pickle

        Saves the data in a file identified by file_name into a .dat file

        Args:
            file_name (string): name of file used to save the data to
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

        Returns:
            None.
        """
        try:
            with open(file_name, 'wb') as fileObj:
                pickle.dump(table, fileObj)
        except FileNotFoundError as e:
            print('Binary file does not exist!')
            print('Build in error info:')
            print(type(e), e, e.__doc__, sep='\n')

```

Listing 4 – Writing to a binary file and FileNotFoundError handling.

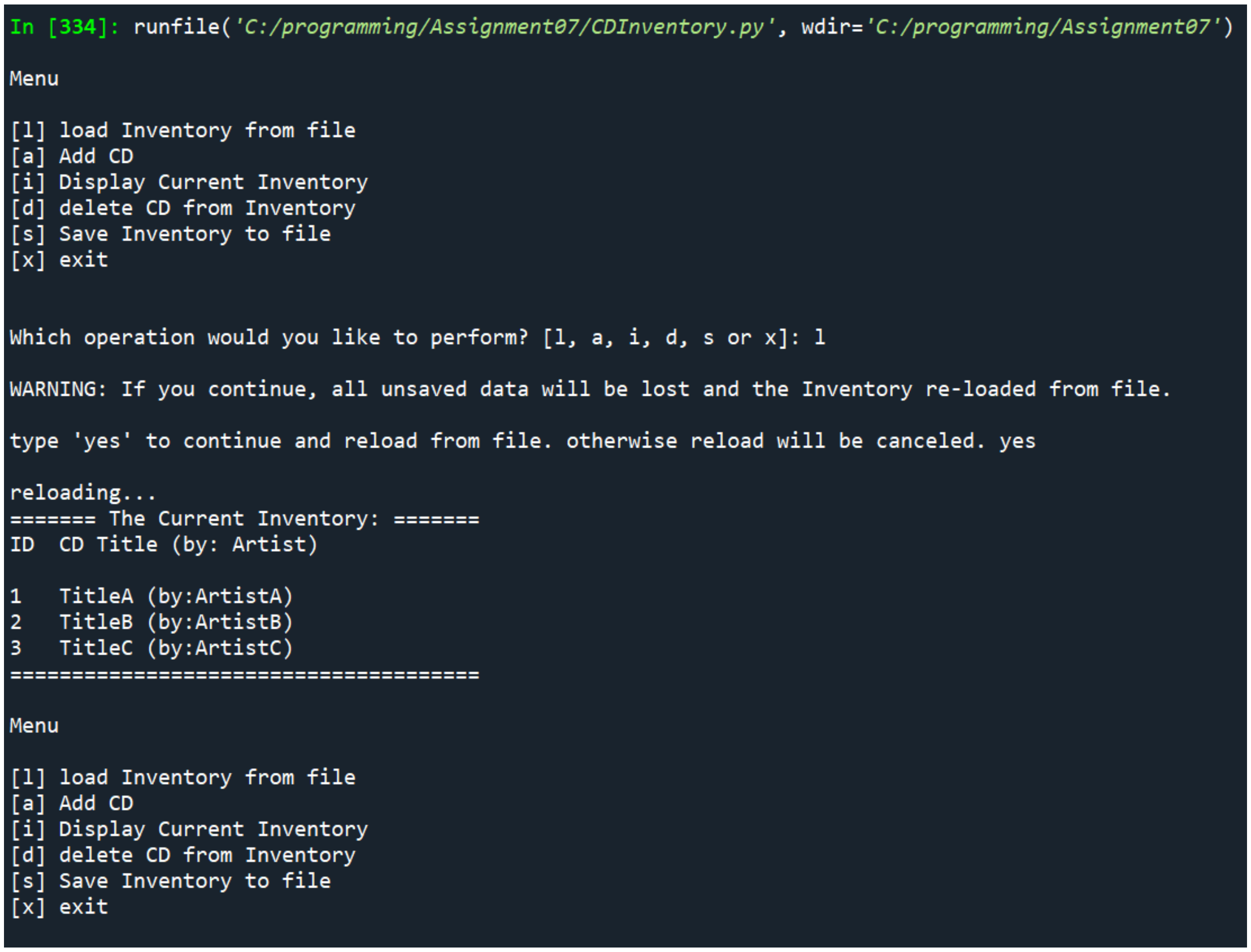
In this assignment, the script uses CDInventory.dat to read and write the binary CD inventory data via pickle module. However, when the script is initialized for the first time, the CDInventory.dat file does not exist in the directory. Therefore, the current CD inventory data needs to be loaded to the memory from the CDInventory.txt file. This was accomplished by importing *os.path* module at the beginning of the script (**Listing 1**, Line 10) and including *os.path.isfile()* function in the CD data loading section (**Listing 5**, line 246 to 251). Briefly, if the CDInventory.dat file does not exist in the directory, the script uses *read_Textfile()* function to read the data from the CDInventory.txt file (**Fig. 7**), which happens on the first run of the script. Once the CD inventory data is saved to the CDInventory.dat file as binary data (**Fig. 8**), the script uses *read_file()* function and CDInventory.dat file to read the binary data to the memory in the subsequent runs of the script.



```python
# 1. When program starts, read in the currently saved Inventory
#FileProcessor.read_file(strFileName, lstTbl)
if os.path.isfile('CDInventory.dat'): # if "CDInventory.dat" exits, use function "read_file()"
    read_FileName = 'CDInventory.dat'
    lstTbl = FileProcessor.read_file(read_FileName)
else:                                 # if "CDInventory.txt" exits, use function "read_Textfile()"
    read_FileName = 'CDInventory.txt'
    FileProcessor.read_Textfile(read_FileName,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('\nreloading...')
            if os.path.isfile('CDInventory.dat'): # if "CDInventory.dat" exits, use function "read_file()"
                read_FileName = 'CDInventory.dat'
                lstTbl = FileProcessor.read_file(read_FileName)
            else:                                 # if "CDInventory.txt" exits, use function "read_Textfile()"
                read_FileName = 'CDInventory.txt'
                FileProcessor.read_Textfile(read_FileName,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.
```

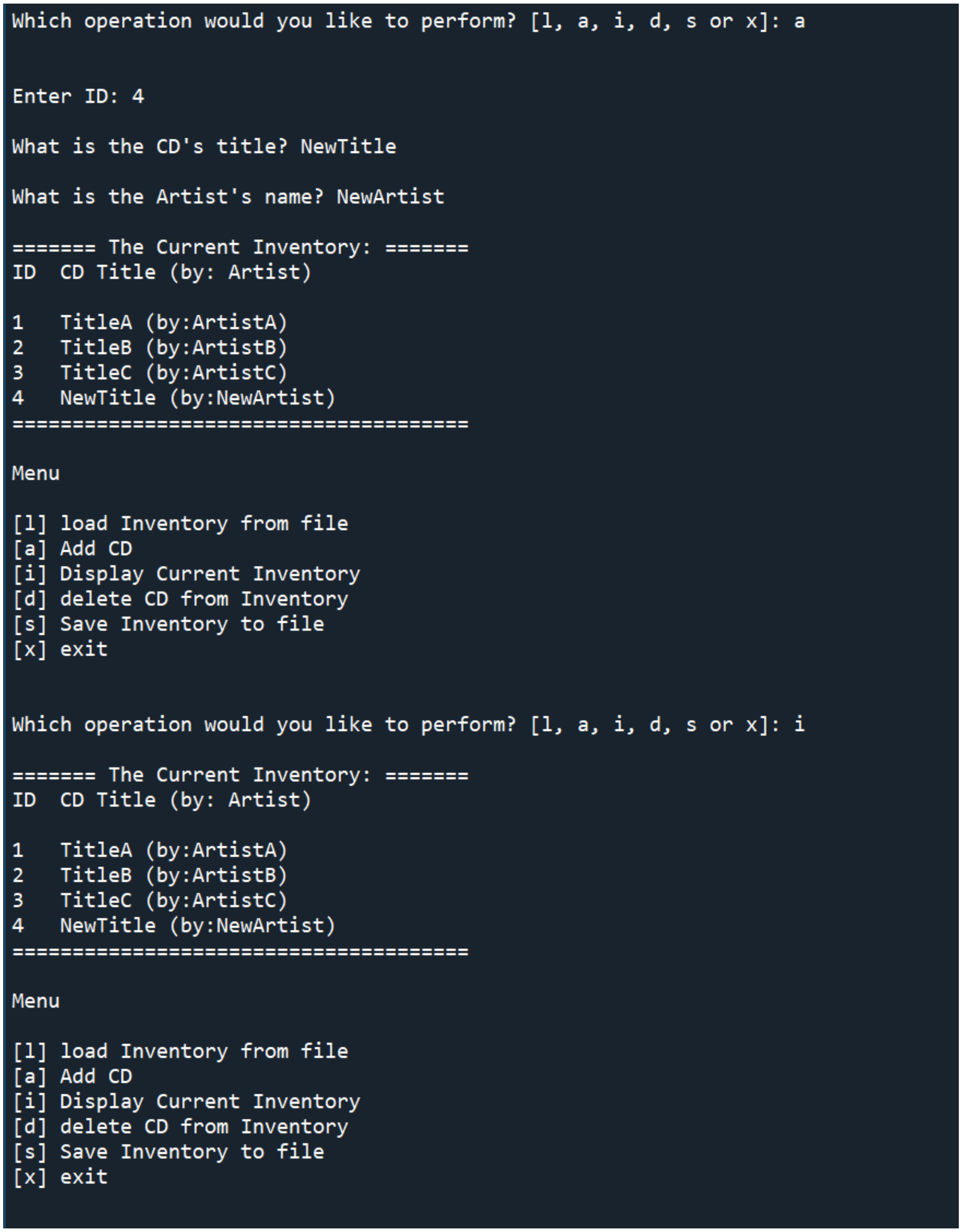
Listing 5 – os.path.isfile() function. Reading from "CDInventory.txt" file if CDInventory.dat file does not exist on the first run in the file directory. For the subsequent runs, the script reads binary data from "CDInventory.dat" file.

Successful operation of the script in Spyder IDE was provided in **Figure 1, 2,** and **3**.



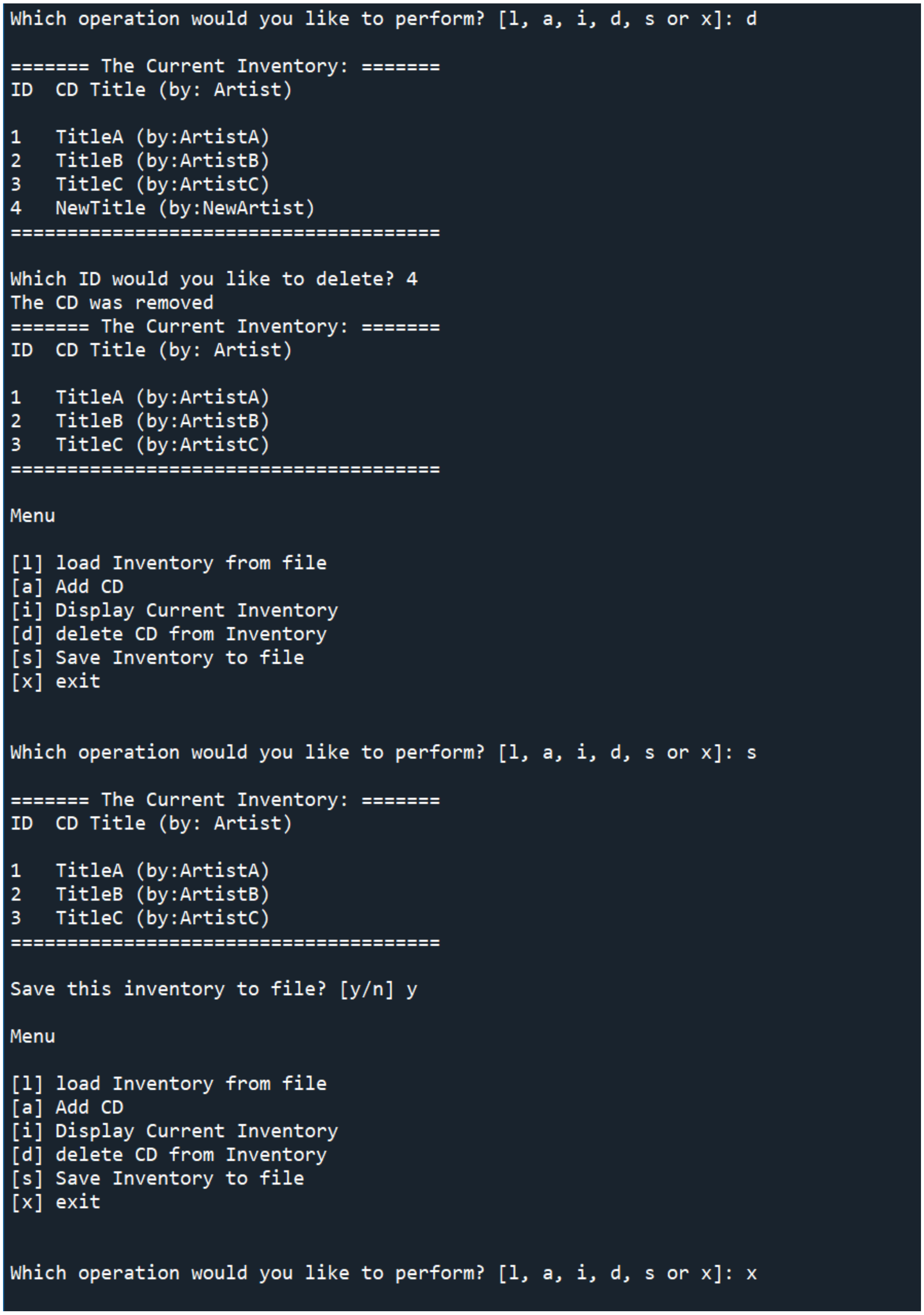
```
In [334]: runfile('C:/programming/Assignment07/CDInventory.py', wdir='C:/programming/Assignment07')

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]: 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   TitleA (by:ArtistA)
2   TitleB (by:ArtistB)
3   TitleC (by:ArtistC)
======================================

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
```

Figure 1 – Successful run of the script in Spyder IDE [1/3].



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


Enter ID: 4

What is the CD's title? NewTitle

What is the Artist's name? NewArtist

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

1   TitleA (by:ArtistA)
2   TitleB (by:ArtistB)
3   TitleC (by:ArtistC)
4   NewTitle (by:NewArtist)
======================================

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   TitleA (by:ArtistA)
2   TitleB (by:ArtistB)
3   TitleC (by:ArtistC)
4   NewTitle (by:NewArtist)
======================================

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
```

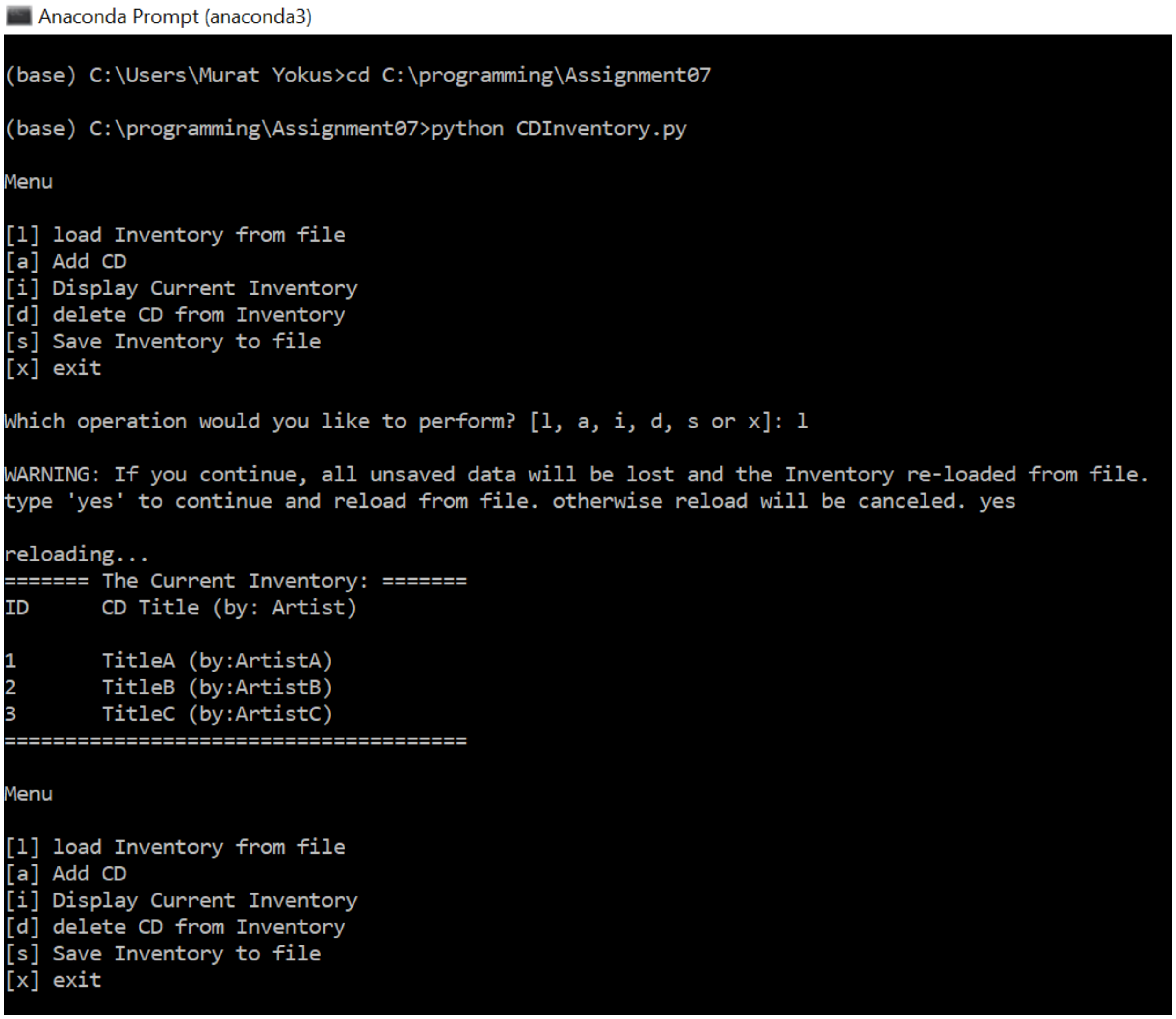
Figure 2– Successful run of the script in Spyder IDE [2/3].



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

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

1   TitleA (by:ArtistA)
2   TitleB (by:ArtistB)
3   TitleC (by:ArtistC)
4   NewTitle (by:NewArtist)
======================================

Which ID would you like to delete? 4
The CD was removed
======= The Current Inventory: =======
ID  CD Title (by: Artist)

1   TitleA (by:ArtistA)
2   TitleB (by:ArtistB)
3   TitleC (by:ArtistC)
======================================

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   TitleA (by:ArtistA)
2   TitleB (by:ArtistB)
3   TitleC (by:ArtistC)
======================================

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
```

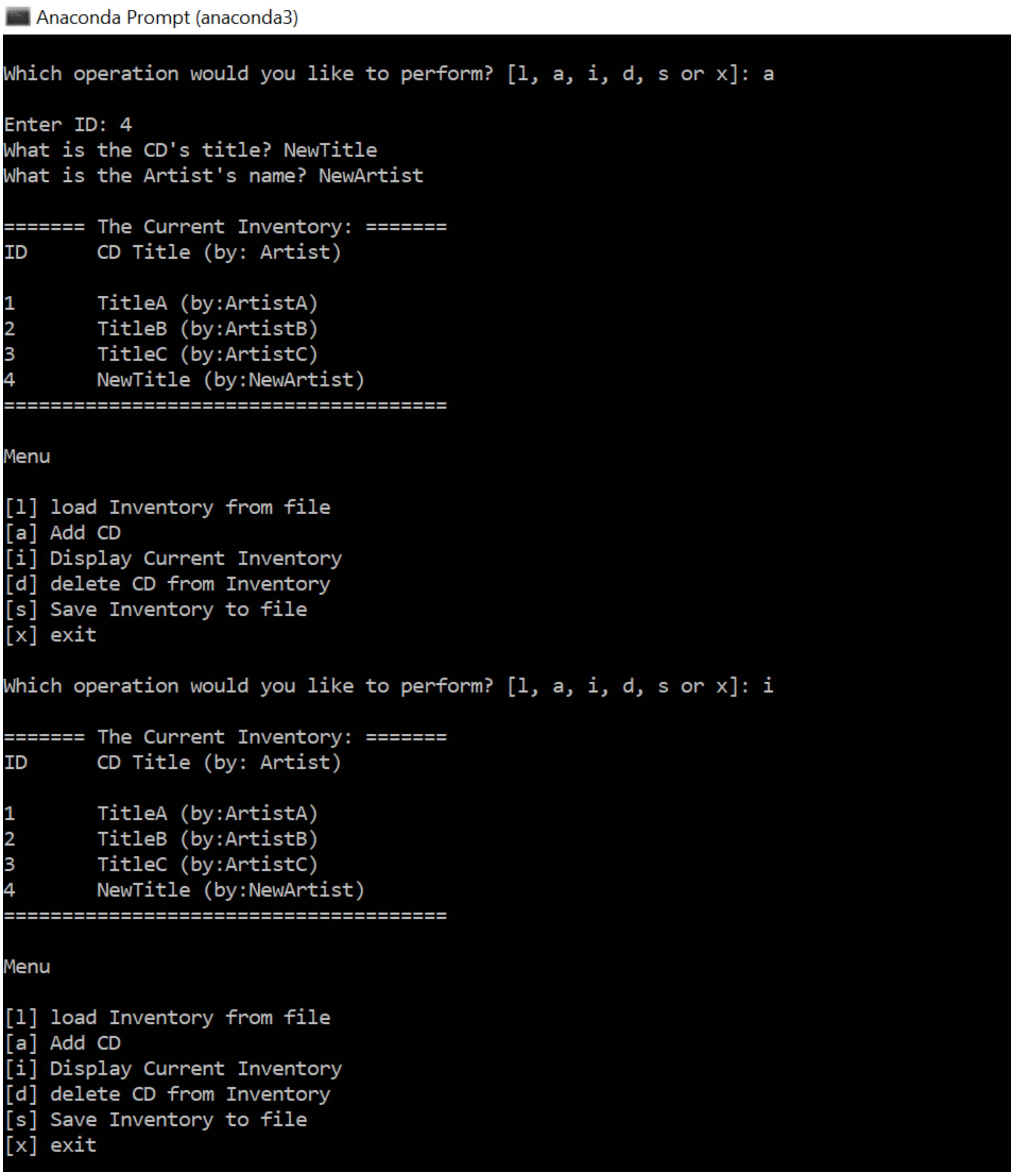
Figure 3– Successful run of the script in Spyder IDE [3/3].

Successful operation of the script in Anaconda Terminal was provided in **Figure 4, 5,** and **6**. The screenshot of the CDInventory.txt and CDInventory.dat files are given in **Figure 7** and **8**.



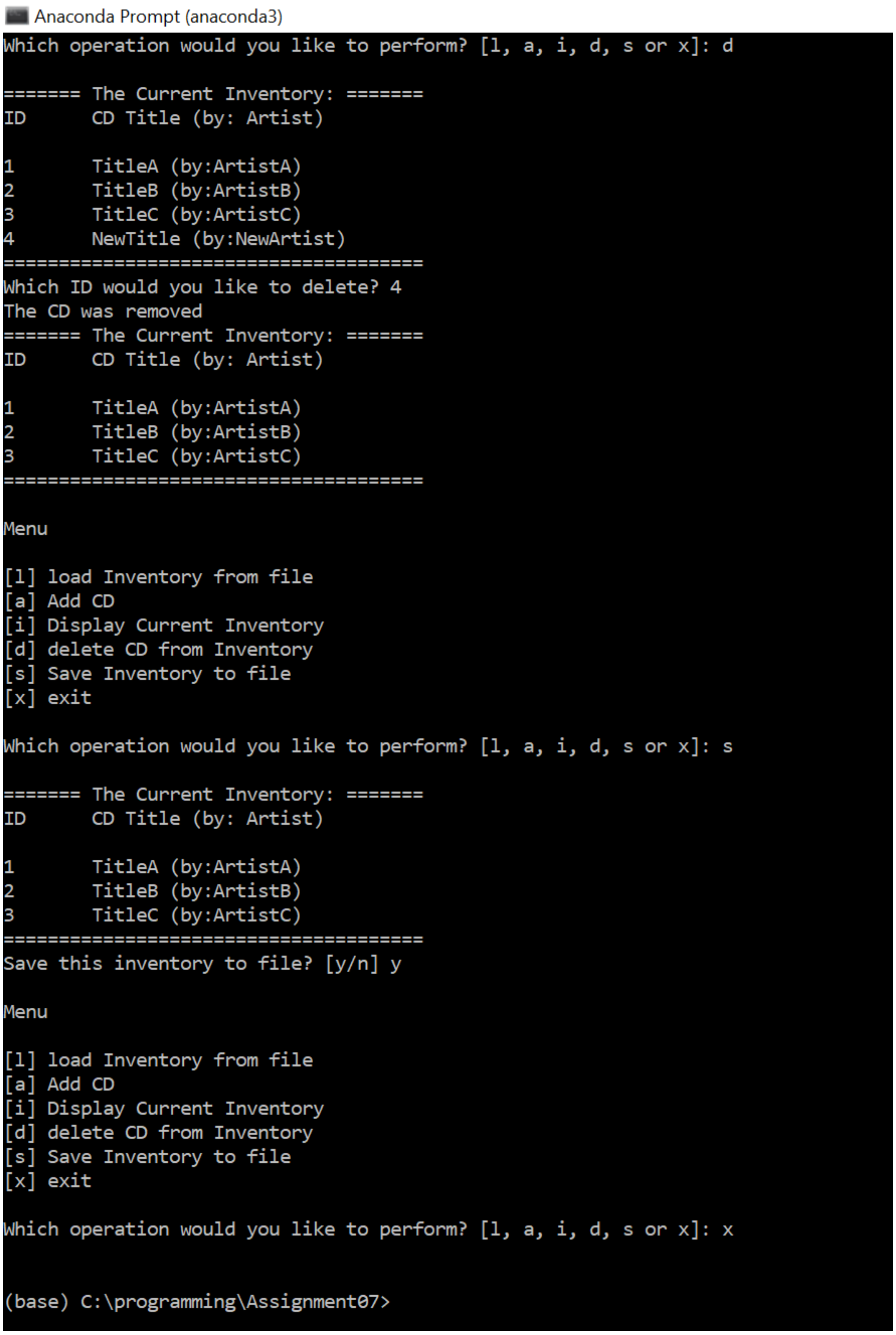
```
Anaconda Prompt (anaconda3)

(base) C:\Users\Murat Yokus>cd C:\programming\Assignment07

(base) C:\programming\Assignment07>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]: 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       TitleA (by:ArtistA)
2       TitleB (by:ArtistB)
3       TitleC (by:ArtistC)
======================================

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
```

Figure 4 – Successful run of the script in terminal [1/3].



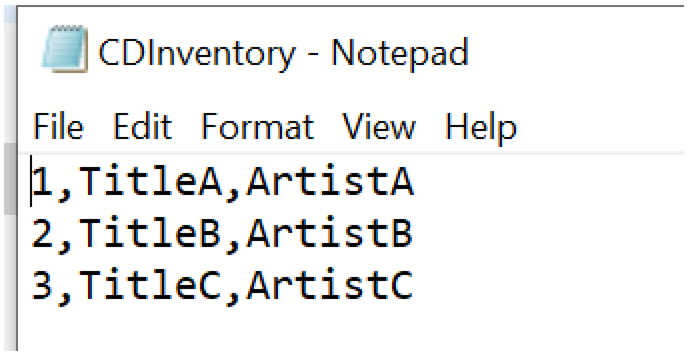
```
Anaconda Prompt (anaconda3)

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

Enter ID: 4
What is the CD's title? NewTitle
What is the Artist's name? NewArtist

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

1       TitleA (by:ArtistA)
2       TitleB (by:ArtistB)
3       TitleC (by:ArtistC)
4       NewTitle (by:NewArtist)
======================================

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       TitleA (by:ArtistA)
2       TitleB (by:ArtistB)
3       TitleC (by:ArtistC)
4       NewTitle (by:NewArtist)
======================================

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
```

Figure 5 – Successful run of the script in terminal [2/3].


Anaconda Prompt (anaconda3)

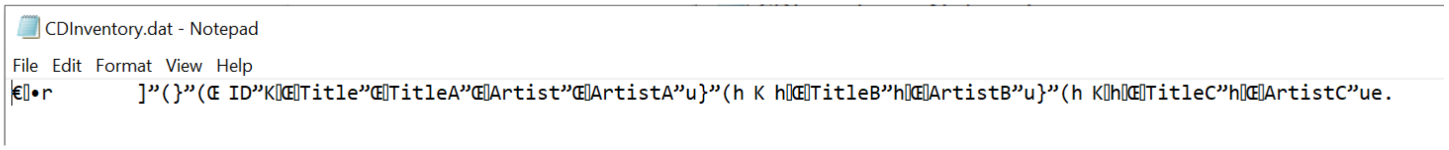
```
Which operation would you like to perform? [l, a, i, d, s or x]: d

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

1       TitleA (by:ArtistA)
2       TitleB (by:ArtistB)
3       TitleC (by:ArtistC)
4       NewTitle (by:NewArtist)
======================================
Which ID would you like to delete? 4
The CD was removed
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       TitleA (by:ArtistA)
2       TitleB (by:ArtistB)
3       TitleC (by:ArtistC)
======================================

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       TitleA (by:ArtistA)
2       TitleB (by:ArtistB)
3       TitleC (by:ArtistC)
======================================
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


(base) C:\programming\Assignment07>
```

Figure 6 – Successful run of the script in terminal [3/3].



```
1,TitleA,ArtistA
2,TitleB,ArtistB
3,TitleC,ArtistC
```

Figure 7– Content of the CDInventory.txt file. This text file was initially used by the script to load the current inventory data to the memory if the CDInventory.dat file does not exist in the file directory.



```
€•r        ]”(}”(Œ ID”KŒTitle”ŒTitleA”ŒArtist”ŒArtistA”u}”(h K hŒTitleB”hŒArtistB”u}”(h KhŒTitleC”hŒArtistC”ue.
```

Figure 8 – Content of the CDInventory.dat file after saving the CD inventory table to a binary file.

Useful Websites for Structured Error Handing in Python:

- https://docs.python.org/3/library/exceptions.html

Useful Websites for Picking in Python

- https://www.bestprog.net/en/2020/04/30/python-binary-files-examples-of-working-with-binary-files/
- https://zetcode.com/python/pickle/

GitHub Link

The knowledge document, the script, and CDInventory.txt file were uploaded to GitHub/Assignment_07 repository. Link: https://github.com/myokus/Assignment_07

Module 7: Learnings

In the Module 7, I learned and practiced the following topics.

- Structured error handling
- Binary data storage and reading

Summary

Overall, the objective of this assignment is to implement structured error handing and data storage/reading using binary files. Potential build-in Python errors (user interaction, type casting (*e.g.,* string to integer), or file access operations) were handled using try-except blocks. Similarly, reading from and writing to binary files were done using *pickle* module.

Appendix

Listing CDInventory.py

```
#------------------------------------------#
# Title: CDInventory.py
# Desc: Script CDInventory with Structured Error Handling and Binary Data Storage
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File #
# MYokus, 2021-Aug-22, Added Code
#------------------------------------------#

import pickle
import os.path #Common pathname manipulation

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


# -- PROCESSING -- #
class DataProcessor:
    """Processing the data in a 2D table (list of dicts)"""

    @staticmethod
    def add_to_table(inputs, table):
        """Function to add a list to a 2D table (list of dicts)

       Add user inputs (a list) to the main inventory table (list of dicts)

       Args:
           inputs (list): user inputs (ID, title, artist)
           table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

       Returns:
           None
       """
        try:
            dicRow = {'ID': int(inputs[0]), 'Title': inputs[1], 'Artist': inputs[2]}
            table.append(dicRow)
        except ValueError as e:
            print('That is not an integer!')
            print('Build in error info:')
            print(type(e), e, e.__doc__, sep='\n')

    @staticmethod
    def del_row(row_del, table):
        """ Function to delete a row in a 2D table

       Deletes an entry from the main inventory table (list of dicts)

       Args:
           row (int): the row number to be deleted
           table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

       Returns:
           None
       """
        intRowNr = -1
        blnCDRemoved = False
        for row in table:
            intRowNr += 1
            if row['ID'] == row_del:
                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):
        """Function to manage data ingestion from a 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

       Returns:
           data (list of dict): 2D data structure (list of dicts)
       """

        try:
            data = []
            with open(file_name, 'rb') as fileObj:
                data = pickle.load(fileObj)
            return data
        except FileNotFoundError as e:
            print('Binary file does not exist!')
            print('Build in error info:')
            print(type(e), e, e.__doc__, sep='\n')

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

       Reads the data from a text 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.
       """

        try:
            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(',') # data type: list
                dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]} # data type: dictionary
                table.append(dicRow) # data type: list
            objFile.close()
        except FileNotFoundError as e:
            print('Text file does not exist!')
            print('Build in error info:')
            print(type(e), e, e.__doc__, sep='\n')
```

```python


    @staticmethod
    def write_file(file_name, table):
        """Function to save a 2D table (a list of dictionaries) to file via pickle

       Saves the data in a file identified by file_name into a .dat file

       Args:
           file_name (string): name of file used to save the data to
           table (list of dict): 2D data structure (list of dicts) that holds the data during runtime

       Returns:
           None.
       """
        try:
            with open(file_name, 'wb') as fileObj:
                pickle.dump(table, fileObj)
        except FileNotFoundError as e:
            print('Binary file does not exist!')
            print('Build in error info:')
            print(type(e), e, e.__doc__, sep='\n')


# -- 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('\nMenu\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']: # 'While not Loop: executes the body of the loop until the condition for loop termination is met'
            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('======================================')

    @staticmethod
    def user_input():
        """ Ask user for new ID, CD Title, and Artist
       Args:
           None.

       Returns:
           a list of user inputs
       """

        strID = input('Enter ID: ').strip()
        strTitle = input('What is the CD\'s title? ').strip()
        stArtist = input('What is the Artist\'s name? ').strip()
        return [strID, strTitle, stArtist]


# 1. When program starts, read in the currently saved Inventory
#FileProcessor.read_file(strFileName, lstTbl)
if os.path.isfile('CDInventory.dat'): # if "CDInventory.dat" exits, use function "read_file()"
    read_FileName = 'CDInventory.dat'
    lstTbl = FileProcessor.read_file(read_FileName)
else:                                 # if "CDInventory.txt" exits, use function "read_Textfile()"
    read_FileName = 'CDInventory.txt'
    FileProcessor.read_Textfile(read_FileName,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('\nreloading...')
            if os.path.isfile('CDInventory.dat'): # if "CDInventory.dat" exits, use function "read_file()"
                read_FileName = 'CDInventory.dat'
                lstTbl = FileProcessor.read_file(read_FileName)
            else:                                  # if "CDInventory.txt" exits, use function "read_Textfile()"
                read_FileName = 'CDInventory.txt'
                FileProcessor.read_Textfile(read_FileName,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
        inputs_list = IO.user_input() # a list of user inputs

        # 3.3.2 Add item to the table
        DataProcessor.add_to_table(inputs_list, lstTbl)
        print()
        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
        DataProcessor.del_row(intIDDel, 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
            FileProcessor.write_file(save_FileName, lstTbl)
        else:
            input('The inventory was NOT saved to file. Press [ENTER] to return to the menu.')
        continue  # start loop back at top.
    # 3.7 catch-all should not be possible, as user choice gets vetted in IO, but to be save:
    else:
        print('General Error')
```