<Sayali Subodh Shinde>

<March 13 2022>

<Foundations of Programming (Python)>

<Assignment 07>

To add Structured Error Handling Capability and write and read the inventory using binary file format in the CDInventory.py

Introduction

Brief overview about this module is I learnt in detail about the read() ,readline(), readlines() to work with text file, where the read is line reading, readline is particular line reading and the readlines function is used for getting a dict of all lines in the file. In this module I also learnt about the Binary files usage using pickle module, whose extension is .dat. and are opened using rb and wb and ab for reading and writing and appending specifically where b is binary The usage of the binary files to have the data in the memory and not save it in a txt file saving memory and processing time. In this module the Labs also emphasized on the sys module using command line run option

The Exception handling is also explained in this module, with standard Exception objects like Value Error, FileNotFoundError, ZeroDivision Error, as the objects of the Class Exception. One can also defined derived class from base Class Exception to create custom Exceptions, and they have some properties from base class. I also learnt the with statement for files usage.

This Module also describes working with Github and the markdowm.pl created in Perl language to standardize the.ReadME file in Github about what is your code about.

Aim of Assignment

The aim of this assignment is to read from and write to the binary file and not text file for CdInventory processing. The second part of the assignment deals with the Error Handling incase of ID not a integer and if the file doesn't exists.

1. Add write functionality change in the class Fileprocessor for functions to read from and write to the binary file
2. Add the Structured Error Handling if the file doesn't exists for reading to add data and write to the file and then reading it.
3. Add Structured Error Handling for the ID of the CDInventory to be int type.

As majority of the code is the same but needs changes I have divided them into 3 bullets as above

GIST:

In the While loop True all time as its condition is true, the menu will be displayed asking for users to choose. Based on what to select from below like **a** to add CD data**, i** to display current CD data**, s** to save Cd data , and **d** to delete the dict in the list that the user wants, and **l** to load the data from a binary file into the memory and **x** to exit with the help of functions and classes. Here the imp point to note is lstbl which is going to hold all individual cd dict is initialized outside while loop to empty list. If this is in the while loop being empty at every iteration of add we would get a extra empty list appended, so it must be out of while.

The important point is here there are 3 different classes serving the purpose of data processing in the memory (adding newly entered data by the user to the 2D table, deleting the data from the memory if the user wishes and has entered the ID). Another class for IO operations like displaying the menu helping chose from the menu, displaying the current inventory, getting inputs about new inventory (like ID, Title, Artist). The last class is for File Processing like reading from the file and writing to the file. The while loop has the options from memory iterated with help of continue and break if x option is chosen.

The major changes with respect to last code are formatting **the functions docstrings**, as we are **reading and writing to binary files** making sure we have (rb, wb) format for file. The file name to be with **<Filename>.dat extension**. As mentioned, I have **written the error handling part near to the code**, which requires error handling, and just print the error.

The functions are called in the respective chosen choices with the arguments as defined in the function calls. The Program is in SOC (separation of concerns), with data, processing, and presentation skills as mentioned.

Change the write function in the class File Processor for binary file

Now we need to save the data to the file '**CdInventory.dat'** if user chose option s, which is what my.strFilename is. To save data I must have the lsttbl with all the list of dictionaries, which we get by doing process_added_inventory function in the class DataProcessor. But before that with the a option we can the add inventory with that add_inventory()function inside the IO class. Once we have added inventory we can write or save the inventory to the binary file, as you can notice the wb for the write binary.I imported pickle module for binary files. A variable called table_final holds values for the list of dictionaries to be written.

```
    @staticmethod
    def write_file(table,file_name):
        '''
        This function is used to write the 2D Table to the file

        Arguemnts/Parameters:

        file_name : The file to which the data must be written.

        table : The data in memory which is in the table.

        Returns:

        None.

        '''

        table_final = ''
        for row in table:

            lstValues = list(row.values())
            lstValues[0] = str(lstValues[0])
            table_final = table_final +','.join(lstValues) + '\n'
        with open(file_name,'wb') as objfile:
            pickle.dump(table_final,objfile)
```

Figure 1 Assignment07 Screenshot of write_function in class_FileProcessor

Change Read function

For reading Binary file in class FileProcessor

Now to read the binary file, we have the pickle module imported. For reading binary values the rb (read binary) is used and the file is now changed to CDinventory.dat. I have also used the with I learnt from Module 7 in this assignment.

```
class DataProcessor:
    "Processing the Data in Memory"
    # DONE add functions for processing here
    @staticmethod
    def Process_added_inventory(intID,strTitle,stArtist,table):

    @staticmethod
    def delete_inventory(intIDDel,table):
class FileProcessor:
    """Processing the data to and from text file"""

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

        Reads the data from file into the data variable.

        Args:
            file_name (string): name of file used to read the data from in this case CDInventory.dat


        Returns:
            data : which is whats is stored in the Cdinventory.dat
        """

        try:
            with open(file_name,'rb') as objfile:
                data = pickle.load(objfile)

        except FileNotFoundError :
           print("File not found, first add data and write to the file and then read it")
           data = 'Error'
        return data
```

Figure 2. Assignment 07 Screenshot of the read function for reading binary files with error handling

Add error handling capability for the read_file function inside the FileProcessor

For error handling if the file doesn't exist it must be created or written before reading. So for this I am using the try and except for exception where FileNotFoundError is the object of class Exception.

```
        try:
            with open(file_name,'rb') as objfile:
                data = pickle.load(objfile)

        except FileNotFoundError :
           print("File not found, first add data and write to the file and then read it")
           data = 'Error'
        return data
```

Figure 3. Assignment 07 Screenshot of error handling for the read_function if file doesn't exist

Add error handling capability for IO ID when not of int type

Add error handling capability for the add_inventory function of the IO class

Now to check if the ID is int I used while type(ID) is not equal to int, get the ID from user with displaying the error message, that Id is not Int type.

```python
    @staticmethod
    def add_inventory():
        '''
        This is used to take the inputs from the user and store to variables which it returns
        as strID, strTitle, str Artist

        Arguemnts/Parameters:
            None

        Returns:

        strID : The ID entered by the user to add.

        strTitle : The String Title entered by the user.

        stArtist : The string Artist entered by the user.

        '''
        intID = None
        strTitle = ''
        stArtist = ''
        while type(intID) != int:
            try:
                intID = int(input('Enter ID: ').strip())
            except ValueError :
                print("This ID is not integer type , please enter integer ")

        strTitle = input('What is the CD\'s title? ').strip()
        stArtist = input('What is the Artist\'s name? ').strip()
        return intID,strTitle,stArtist
```

Figure 4 Assignment07 Screenshot of the error handling for the ID in input not of INT Type

Script

Below is the script from Spyder the consolidated and filled in for Assignemnt06 filled in for the required asks and renamed to CDInventory.py,as the script is huge I couldn't fit all lines.

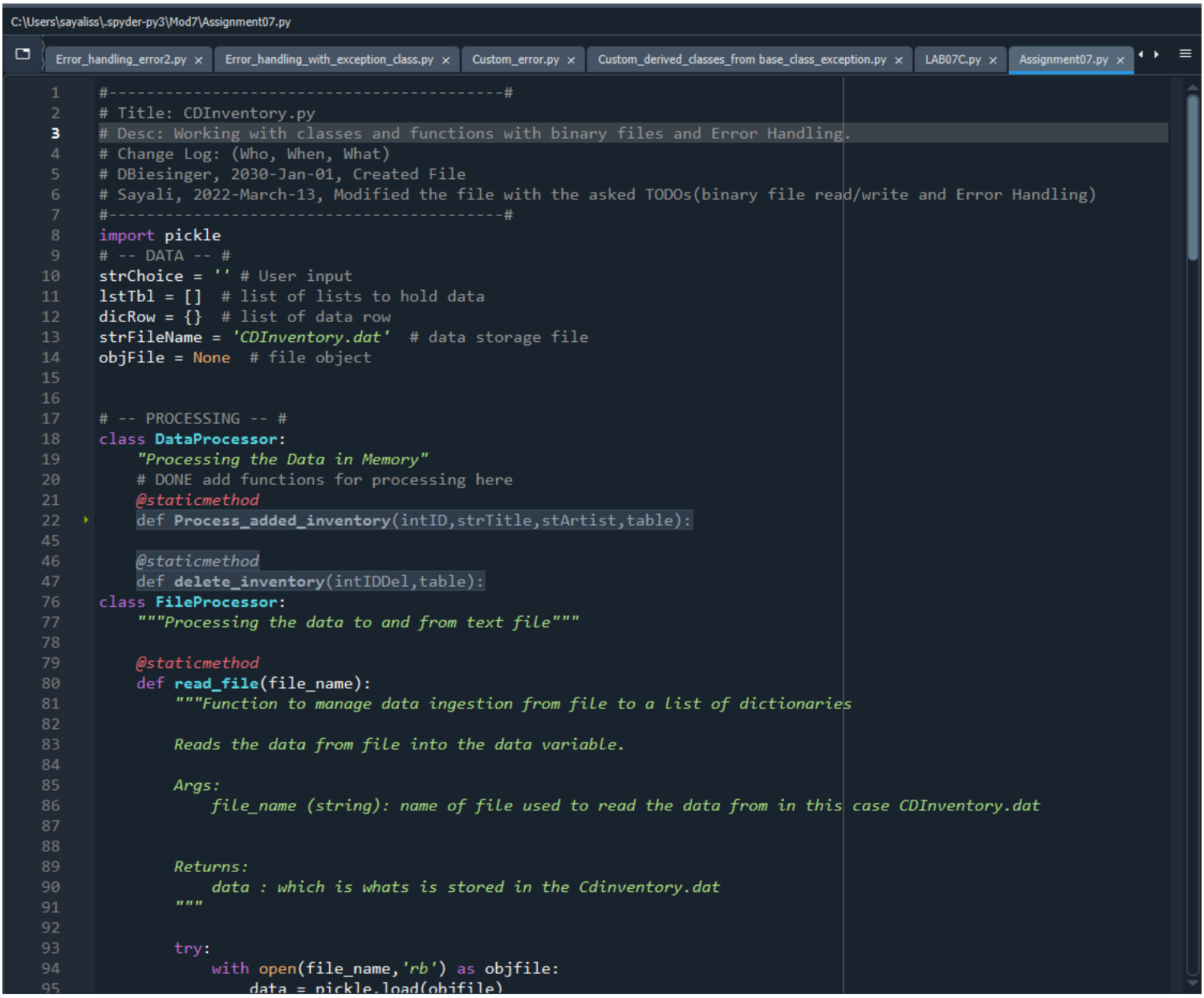

Figure 5 Assignment 07 Screenshot of the Spyder Program

Execution of Program

As requested in the assignemnt07 I have executed the script in Spyder and in Command Prompt. I have also excluded snippets of error if the file not present and if ID is not of type integer.I have also added the **dat** file snippet.

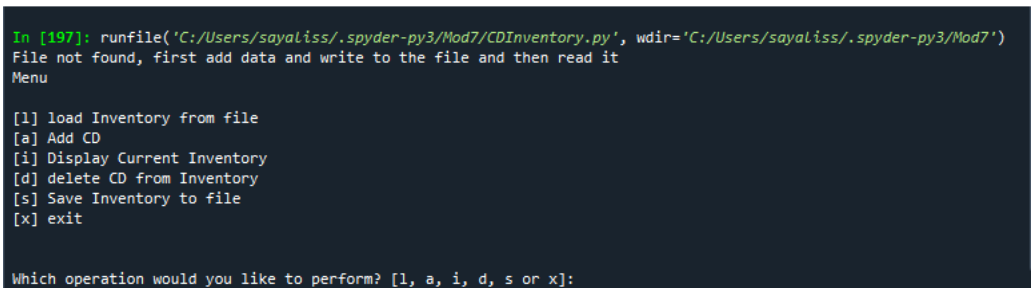

Figure 6 Assignment 07 *Screenshot of Execution with error handling if file not found*

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


Enter ID: 1

What is the CD's title? The big River

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

1   The big River (by:Runrig)
=======================================
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: 2

What is the CD's title? Bad

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

1   The big River (by:Runrig)
2   Bad (by:Michael Jackson)
=======================================
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 7 Assignment 07 *Execution Screenshot of add inventory*

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


Enter ID: ert
This ID is not integer type , please enter integer

Enter ID: 4

What is the CD's title? Sorry

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

1   The big River (by:Runrig)
2   Bad (by:Michael Jackson)
3   Forever (by:Taylor Swift)
4   Sorry (by:Justin Bieber)
=======================================
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 8 Assignment 07 *Execution screenshot of Error handling of valueerror for INT ID*

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

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

1   The big River (by:Runrig)
2   Bad (by:Michael Jackson)
3   Forever (by:Taylor Swift)
4   Sorry (by:Justin Bieber)
======================================

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
```

Figure 9 Assignment07 Execution Screenshot of saving/writing in the binary file

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

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

1   The big River (by:Runrig)
2   Bad (by:Michael Jackson)
3   Forever (by:Taylor Swift)
4   Sorry (by:Justin Bieber)
======================================
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 10 Assignment07 Execution screenshot of loading the current Cd inventory

```
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 canceledyes
reloading...
1,The big River,Runrig
2,Bad,Michael Jackson
3,Forever,Taylor Swift
4,Sorry,Justin Bieber

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

1   The big River (by:Runrig)
2   Bad (by:Michael Jackson)
3   Forever (by:Taylor Swift)
4   Sorry (by:Justin Bieber)
======================================
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 Assignment07 Execution screenshot of loading/reading from binary file

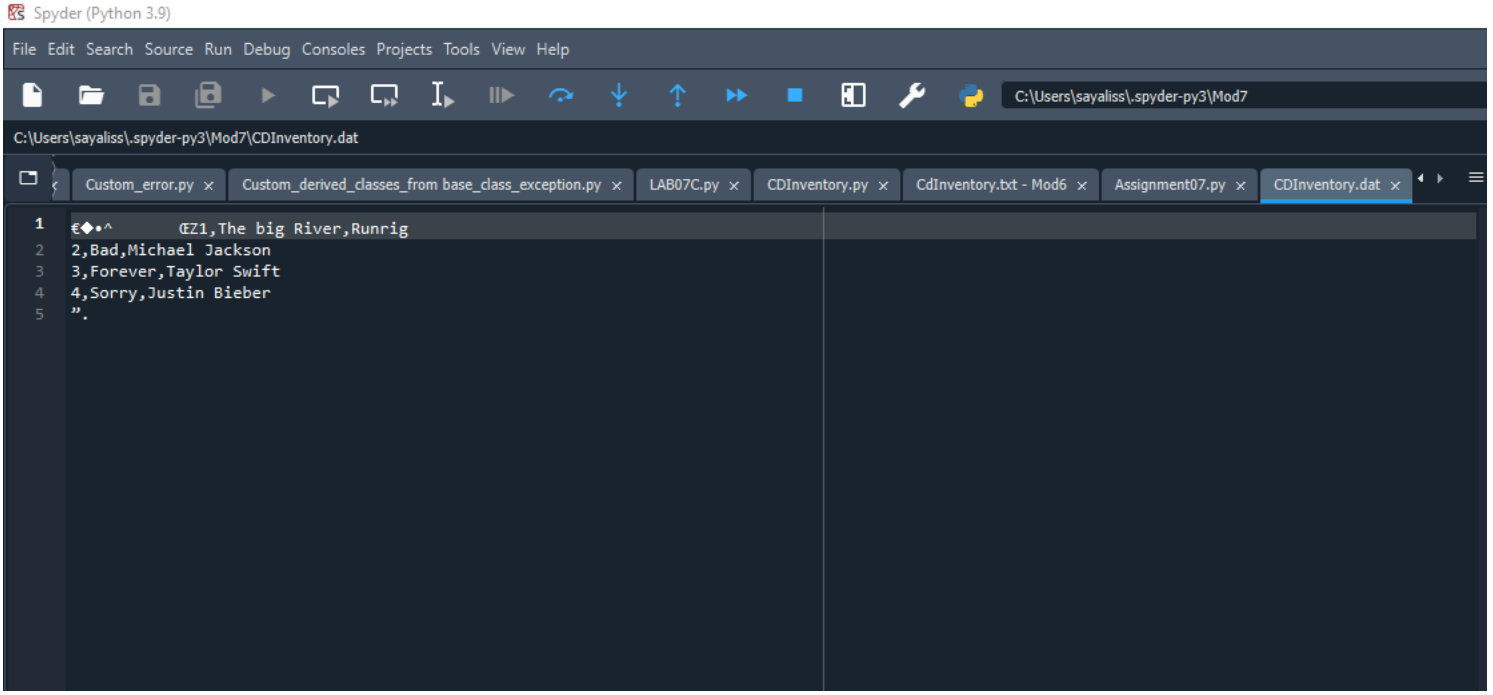

Figure 12 Assignment07 Screenshot of the CDInventory.dat file

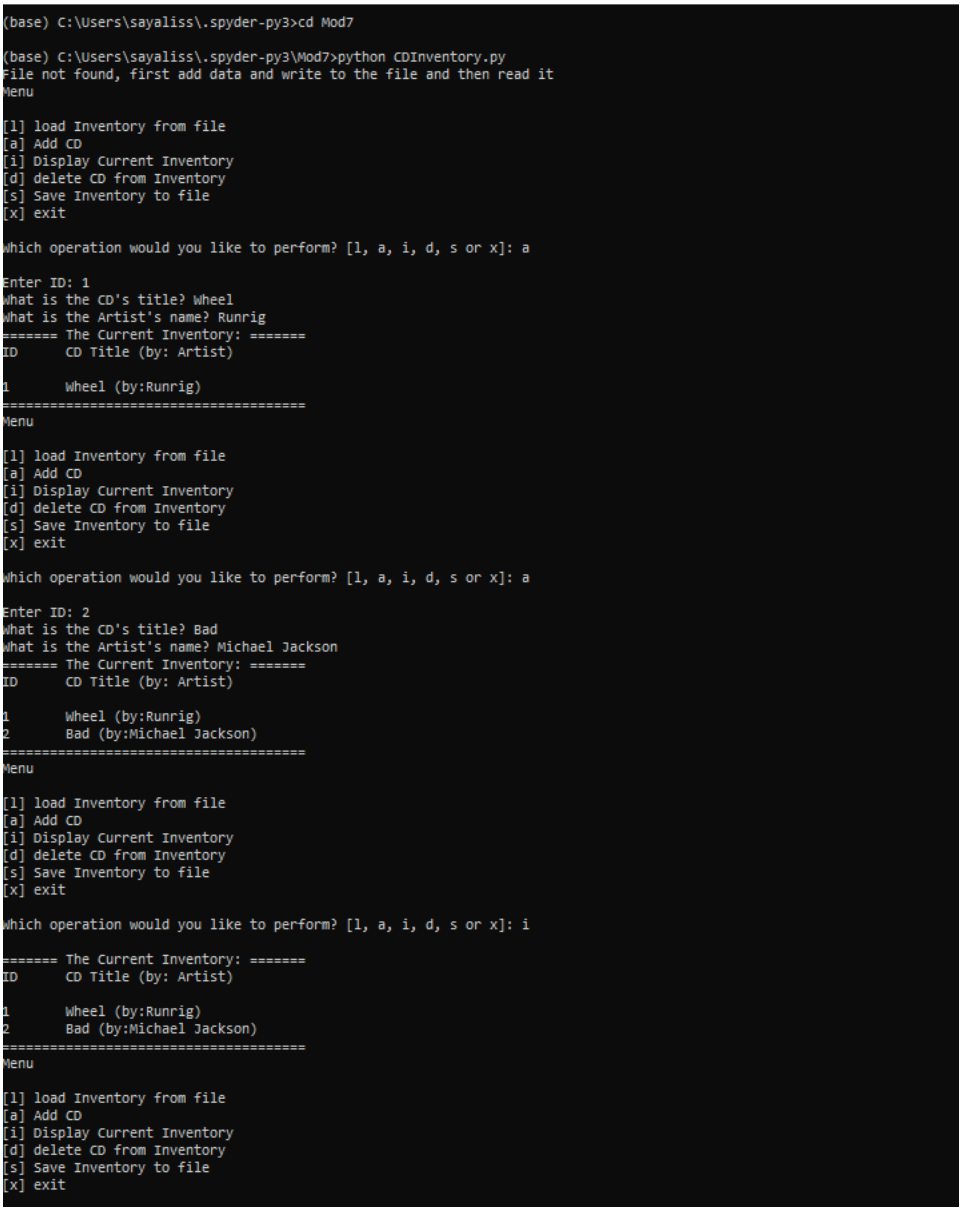

Figure 13 Assignemnt07 Execution of the code on the Command Prompt

I have used Syntax Highlighters (External Reference) 1webpage, to standardize and it displays text, especially script, in different colors and fonts according to the Language.

Summary

I have learnt use of pickle module and binary files usage and I had to understand that if I write or save in the file first or pickle the write_file first it will help with reading the pickled file. I have also learnt the error handling class Exception and its objects, a way to create your own custom error class.

I have uploaded the Gitlab code : https://github.com/sayalisu/Assignment_072

Retrieved 2022-Feb-04

2 Retrieved 2022-Feb-27

Appendix

Script

```
#------------------------------------------#
# Title: CDInventory.py
# Desc: Working with classes and functions with binary files and Error Handling.
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# Sayali, 2022-March-13, Modified the file with the asked TODOs(binary file read/write and Error Handling)
#------------------------------------------#
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:
    "Processing the Data in Memory"
    # DONE add functions for processing here
    @staticmethod
    def Process_added_inventory(intID,strTitle,stArtist,table):
        '''
        TO add the added dictionary to the list we use this Process_added_inventory

        Arguemnts/Parameters:

        strID : This is the INT ID from Added IO Fucntion.

        strTitle : This is the String TITLE from Added IO Fucntion.

        stArtist : This is the String ARTIST from Added IO Fucntion.

        table : The excisting 2D Table.\.

        Returns:

        table : The added row from the IO Function and updates the new 2D List.

        '''

        dicRow = {'ID': intID, 'Title': strTitle, 'Artist': stArtist}
        table.append(dicRow)
        return table

    @staticmethod
    def delete_inventory(intIDDel,table):
        '''
        Deletes the ID selected by the user to delete

        Arguements/ Parameters:

        intIDDel : Its the ID the user has input to delete.

        table : The 2D Table from which we would delete this ID entered row.

```

```
        Returns:

        table : The new 2D table after the deleted entry is removed.

        '''
        intRowNr = -1
        blnCDRemoved = False
        for row in table:
            intRowNr += 1
            if row['ID'] == intIDDel:
                del lstTbl[intRowNr]
                blnCDRemoved = True

                break
        if blnCDRemoved:
            print('The CD was removed')
        else:
            print('Could not find this CD!')
        return table
class FileProcessor:
    """Processing the data to and from text file"""

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

        Reads the data from file into the data variable.

        Args:
            file_name (string): name of file used to read the data from in this case CDInventory.dat


        Returns:
            data : which is whats is stored in the Cdinventory.dat
        """

        try:
            with open(file_name,'rb') as objfile:
                data = pickle.load(objfile)

        except FileNotFoundError :
            print("File not found, first add data and write to the file and then read it")
            data = 'Error'
        return data

    @staticmethod
    def write_file(table,file_name):
        '''
        This function is used to write the 2D Table to the file

        Arguemnts/Parameters:

        file_name : The file to which the data must be written.

        table : The data in memory which is in the table.

        Returns:

        None.
```

```
    '''

    table_final = ''
    for row in table:
        lstValues = list(row.values())
        lstValues[0] = str(lstValues[0])
        table_final = table_final + ','.join(lstValues) + '\n'
    with open(file_name,'wb') as objfile:
        pickle.dump(table_final,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.
        """

        print('Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory')
        print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')

    @staticmethod
    def menu_choice():
        """Gets user input for menu selection

        Args:
            None.

        Returns:
            choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x

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

    @staticmethod
    def show_inventory(table):
        """Displays current inventory table


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

        Returns:
            None.

        """
```

```
        print('======= The Current Inventory: =======')
        print('ID\tCD Title (by: Artist)\n')
        for row in table:
            print('{}\t{} (by:{})'.format(*row.values()))
        print('=======================================')

    # DONE add I/O functions as needed
    @staticmethod
    def add_inventory():
        '''
        This is used to take the inputs from the user and store to variables which it returns
        as strID, strTitle, str Artist

        Arguemnts/Parameters:
            None

        Returns:

        strID : The ID entered by the user to add.

        strTitle : The String Title entered by the user.

        stArtist : The string Artist entered by the user.

        '''
        intID = None
        strTitle = ''
        stArtist = ''
        while type(intID) != int:
            try:
                intID = int(input('Enter ID: ').strip())
            except ValueError :
                print("This ID is not integer type , please enter integer ")

        strTitle = input('What is the CD\'s title? ').strip()
        stArtist = input('What is the Artist\'s name? ').strip()
        return intID,strTitle,stArtist
# 1. When program starts, read in the currently saved Inventory
FileProcessor.read_file(strFileName)


# 2. start main loop
while True:
    # 2.1 Display Menu to user and get choice
    IO.print_menu()
    strChoice = IO.menu_choice()

    # 3. Process menu selection
    # 3.1 process exit first
    if strChoice == 'x':
        break
    # 3.2 process load inventory
    if strChoice == 'l':
        print('WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.')
        strYesNo = input('type \'yes\' to continue and reload from file. otherwise reload will be canceled')
        if strYesNo.lower() == 'yes':
            print('reloading...')
            print (FileProcessor.read_file(strFileName))
            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
        # DONE move IO code into function
        # 3.3.2 Add item to the table
        # DONE move processing code into function
        strID, strTitle, stArtist = IO.add_inventory()
        DataProcessor.Process_added_inventory(strID, strTitle, stArtist, lstTbl)
        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
        # DONE move processing code into function
        DataProcessor.delete_inventory(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':
            FileProcessor.write_file( lstTbl, strFileName)
            # 3.6.2.1 save data
            # DONE move processing code into function
        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')
```