Laura Qin
August 21 2022
IT FDN 110 A
Assignment 07

Binary Files and Structured Error Handling

Introduction

For this assignment we are required to modify our submission from last week, to use binary files for data storage instead of text file, and to handle errors with try-except block.

Working with Binary File through Pickling

The first thing I did was to modify the permanent data storage to use binary data instead of text file. I changed strFileName to ‘CDInventory.dat’, and I updated the two functions in FileProcessor class to read and write the list of dictionaries from and into the pickle file. I took out the code that reads the text file line-by-line and parses each line of text into a dictionary. Instead, since the pickle file can store any python object as-is, including a list of dictionaries, I choose to let the functions read from and write to the pickle file without any modification to the data, to keep the process simple. Here’s the before and after for the two functions:

```
    def read_file(file_name, table):
        '''Function to manage data ingestion from file to a list of dictionaries

        Reads the data from 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.
        '''
        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(',')
            dicRow = {'ID': int(data[0]), 'Title': data[1], 'Artist': data[2]}
            table.append(dicRow)
        objFile.close()
```
Script 1. Starter script - FileProcessor.read_file

```
    def read_file(file_name):
        '''Function to manage data ingestion from file to a list of dictionaries

        Reads the data from the pickle 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:
            table (list of dict): 2D data structure (list of dicts) that holds the data
        '''
        with open(file_name, 'rb') as obj_file:
            table = pickle.load(obj_file)
        return table
```
Script 2. New script - FileProcessor.read_file

```
    def write_file(file_name, table):
        '''Function to save the table in memory into the file

        Go through each row in the table that is a 2D list of dictionaries, extract the
values in each dictionary
        row, combine them into one comma-seperated string and write it to the file

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

        Returns:
            None.
        '''
        obj_file = open(file_name, 'w')
        for row in table:
            lst_values = list(row.values())
            lst_values[0] = str(lst_values[0])
            obj_file.write(','.join(lst_values) + '\n')
        obj_file.close()
```
Script 3. Starter script - FileProcessor.write_file

```
    def write_file(file_name, table):
        '''Function to save the table in memory into the file

        Write the data that is a 2D list of dictionaries into a pickle file

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

        Returns:
            None.
        '''

        with open(file_name, 'wb') as obj_file:
            pickle.dump(table, obj_file)
```
Script 4. New script - FileProcessor.write_file

Learnings about variable scope

Initially I struggled with the previous step because I couldn't figure out why this function is not working. My idea was to pass lstTbl into the function and modify it in the function to take the value of whatever's in the pickle file, instead of having the function return the file. This function doesn't work, it doesn't cause the program to error out but it simply wouldn't change the content of lstTbl.

```
    def read_file(file_name, table):
        with open(file_name, 'rb') as obj_file:
            table = pickle.load(obj_file)
```
Script 5. Initial design for FileProcessor.read_file

I did some Google search and this Stack overflow post helped answer my question. Essentially the function I wrote would take lstTbl as an argument, and then within the function it creates a local variable called table and assigns the data within the pickle file to the local table variable. Since table is a local variable, it's scope is limited to within the function, so its value was changed in the function, but the global variable lstTbl hasn't been changed.

To fix that I removed table from the argument list, and then have the function return the list of dictionaries in the pickle file. That way I was able to circumvent the variable scope issue and have the function return the data that's immediately available for consumption for subsequent steps.

```
def read_file(file_name):
    with open(file_name, 'rb') as obj_file:
        table = pickle.load(obj_file)
    return table
```
Script 6. Final design for FileProcessor.read_file

Error Handling

I added error handling to capture potential file not found error, and data value format error when turning str to int. To strike a balance between the isolation of error and practicality, I wrapped the code in each if/elif block in a try-except structure. For each if/elif block, I consider the possibility of specific types of errors and try to capture those errors with specialized exception classes. For example, for part 3.3 process add a CD, I tried to capture the ValueError exception since an error could easily occur when user enters a non-numeric value for ID.

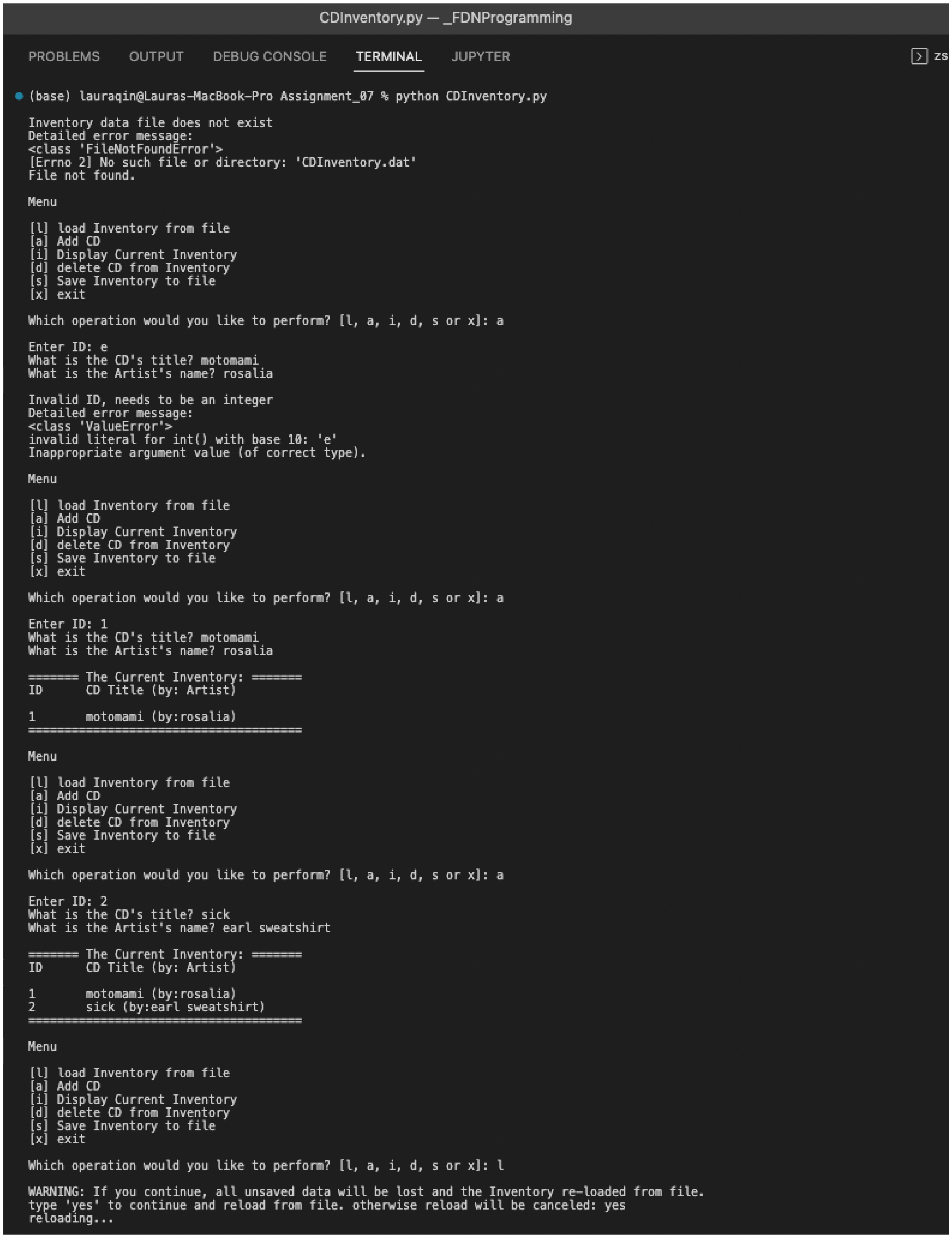
```
# 3.3 process add a CD
elif strChoice == 'a':
    try:
        # 3.3.1 Ask user for new ID, CD Title and Artist
        strID, strTitle, strArtist = IO.get_input_row()
        # 3.3.2 Add item to the table
        DataProcessor.add_row_to_data(strID, strTitle, strArtist, lstTbl)
        IO.show_inventory(lstTbl)
        continue  # start loop back at top.
    except ValueError as e:
        print('\nInvalid ID, needs to be an integer')
        print('Detailed error message: ')
        print(type(e), e, e.__doc__, sep = '\n')
    except Exception as e:
        print('\nThere was a general error')
        print('Detailed error message: ')
        print(type(e), e, e.__doc__, sep = '\n')
```
Script 7. Section 3.3 process add a CD

Another example is 3.2 process load inventory. Since this step reads from a pickle file, I know it's possible for file not found error to occur, so I try to capture it in a FileNotFound exception class.

```
# 3.2 process load inventory
if strChoice == 'l':
    try:
        strYesNo = IO.get_reload_yes_no()
        if strYesNo.lower() == 'yes':
            print('reloading...\n')
            lstTbl = 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.
    except FileNotFoundError as e:
        print('\nInventory data file does not exist')
        print('Detailed error message: ')
        print(type(e), e, e.__doc__, sep = '\n')
    except Exception as e:
        print('\nThere was a general error')
        print('Detailed error message: ')
        print(type(e), e, e.__doc__, sep = '\n')
```
Script 8. Section 3.2 process load inventory

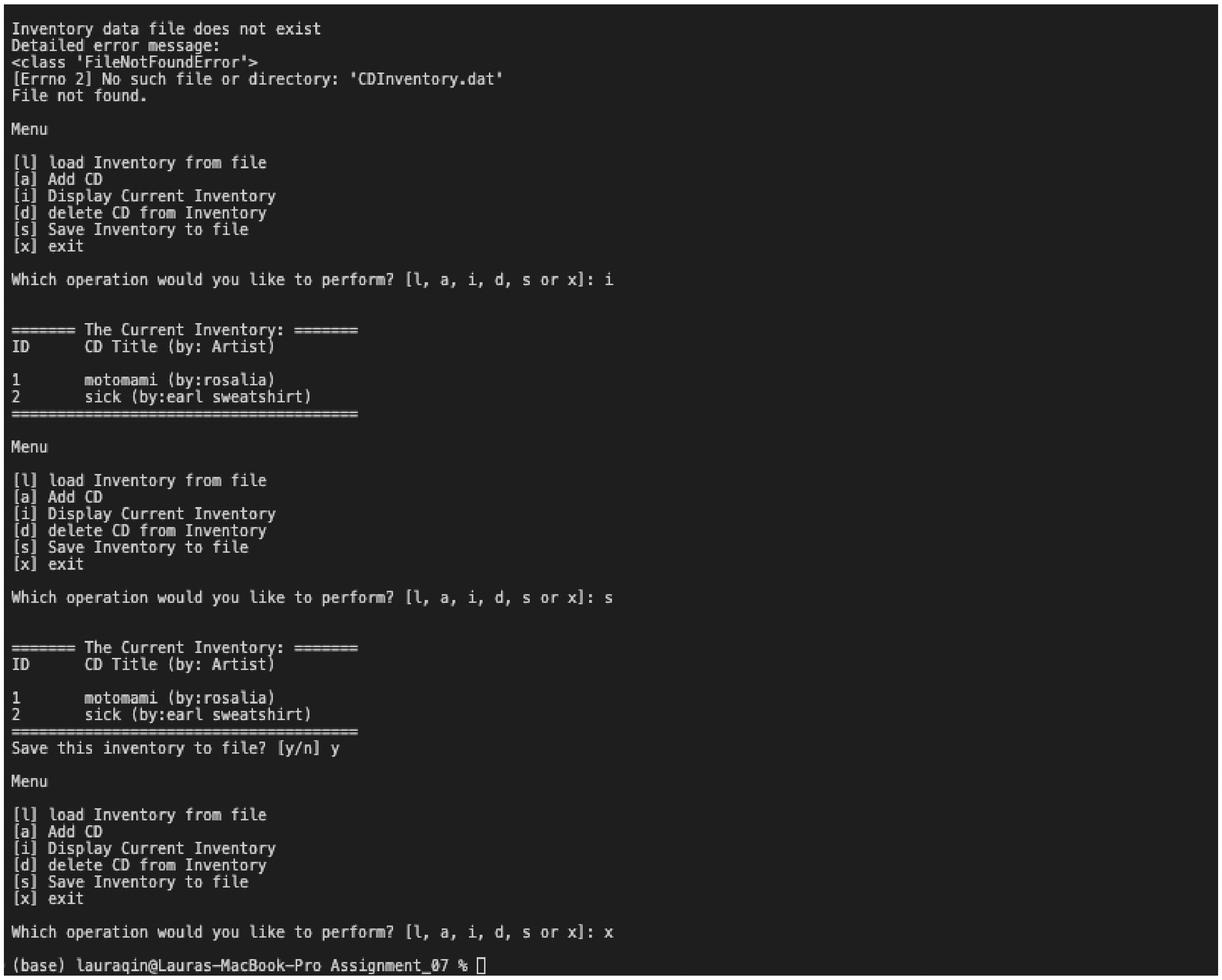
Result

I ran the updated script in VS Code, it's working as expected. It prints out a menu, and asks the user to enter their selection. It was able to execute the user's request, and generate a text file that includes the data entered by the user. It also behaves as expected when run on the Terminal. When the file to read from doesn't exist, and when the user enters a non-numeric ID, instead of crashing it would print out some useful diagnostic information and skip to the beginning menu selection.


CDInventory.py — _FDNProgramming

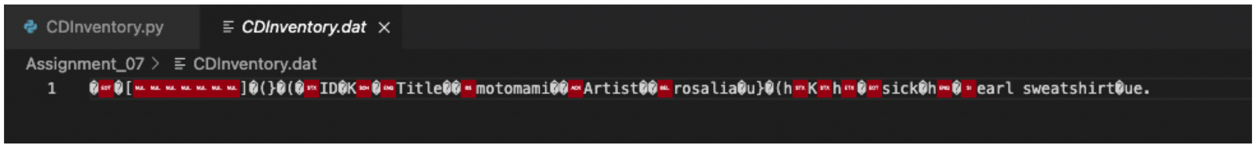
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
(base) lauraqin@Lauras-MacBook-Pro Assignment_07 % python CDInventory.py

Inventory data file does not exist
Detailed error message:
<class 'FileNotFoundError'>
[Errno 2] No such file or directory: 'CDInventory.dat'
File not found.

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: e
What is the CD's title? motomami
What is the Artist's name? rosalia

Invalid ID, needs to be an integer
Detailed error message:
<class 'ValueError'>
invalid literal for int() with base 10: 'e'
Inappropriate argument value (of correct type).

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: 1
What is the CD's title? motomami
What is the Artist's name? rosalia

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

1       motomami (by:rosalia)
======================================

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? sick
What is the Artist's name? earl sweatshirt

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

1       motomami (by:rosalia)
2       sick (by:earl sweatshirt)
======================================

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...
```



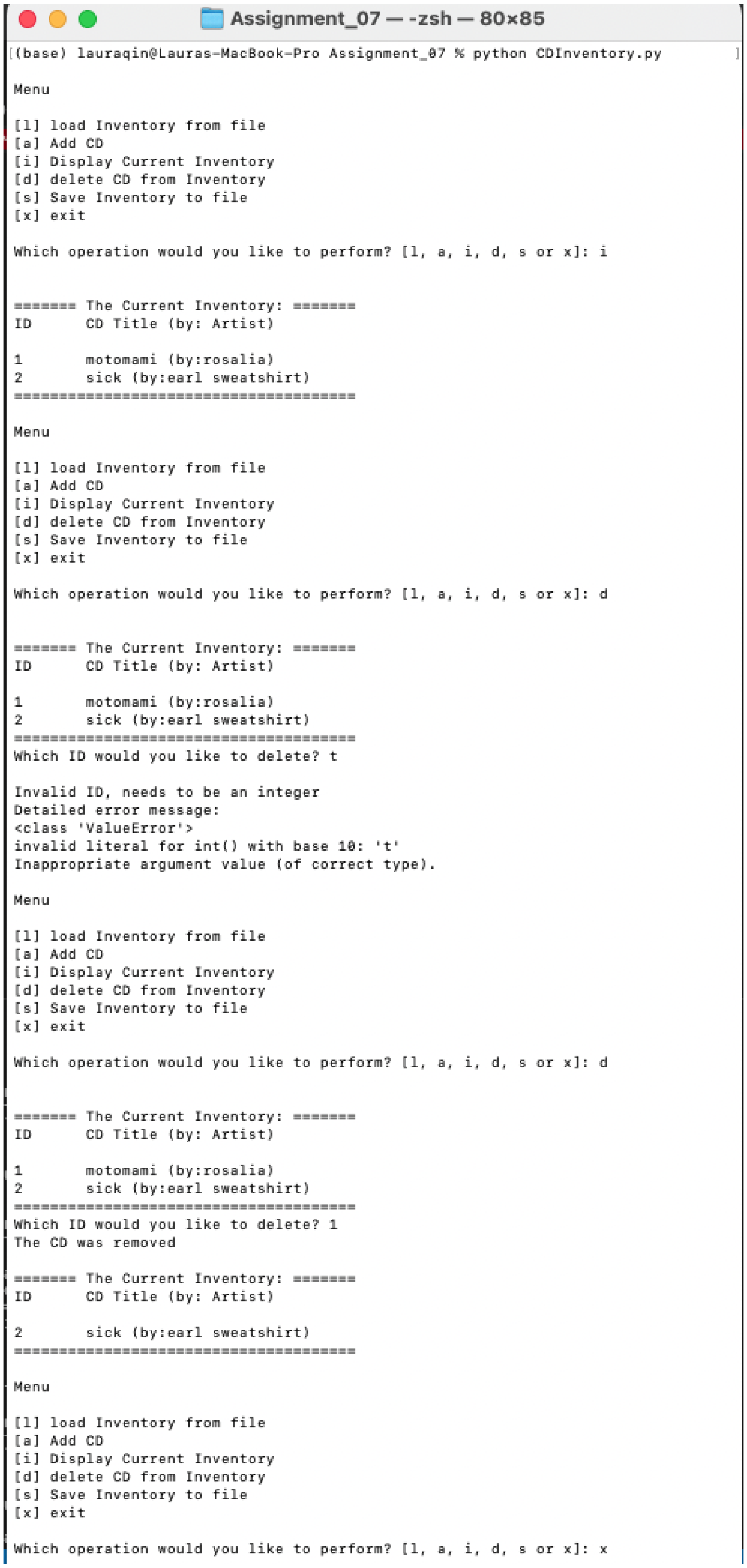
```
Inventory data file does not exist
Detailed error message:
<class 'FileNotFoundError'>
[Errno 2] No such file or directory: 'CDInventory.dat'
File not found.

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       motomami (by:rosalia)
2       sick (by:earl sweatshirt)
======================================

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       motomami (by:rosalia)
2       sick (by:earl sweatshirt)
======================================
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) lauraqin@Lauras-MacBook-Pro Assignment_07 % 
```

Screenshot 1. VSCode Run Result



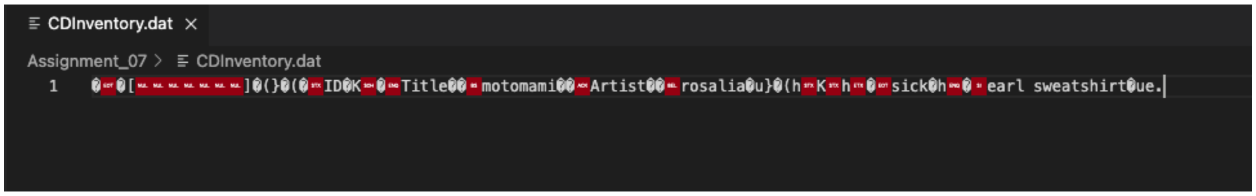
```
�[]�(}�(�ID�K�Title��motomami��Artist��rosalia�u}�(hKh�sick�h�earl sweatshirt�ue.
```

Screenshot 2. CDInventory.dat after VSCode Run


Assignment_07 — -zsh — 80×85

```
(base) lauraqin@Lauras-MacBook-Pro Assignment_07 % 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]: i


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

1       motomami (by:rosalia)
2       sick (by:earl sweatshirt)
======================================

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]: d


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

1       motomami (by:rosalia)
2       sick (by:earl sweatshirt)
======================================
Which ID would you like to delete? t

Invalid ID, needs to be an integer
Detailed error message:
<class 'ValueError'>
invalid literal for int() with base 10: 't'
Inappropriate argument value (of correct type).

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]: d


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

1       motomami (by:rosalia)
2       sick (by:earl sweatshirt)
======================================
Which ID would you like to delete? 1
The CD was removed

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

2       sick (by:earl sweatshirt)
======================================

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
```

Screenshot 3. Terminal Run Result



```
CDInventory.dat
Assignment_07 > CDInventory.dat
�EOT�[NULNULNULNULNULNULNUL]�(}�(�STXID�K�SOH�ENQTitle���motomami���ACKArtist���rosalia�u}�(hSTXKSTXhSTX�EOTsick�hENQ�SIearl sweatshirt�ue.
```

Screenshot 4. CDInventory.txt after Terminal Run

Summary

Through this assignment I was able to learn more about structured error handling, as well as ways to read/write binary files, and I was able to use my new knowledge to update my submission from last week, to use binary files for data storage instead of text file, and to handle errors with try-except block. I also learned more about variable scope.

Appendix

1. CDInventory.py

```
#------------------------------------------------------------------------------
#
# Title: CDInventory.py
# Desc: Working with classes and functions.
# Change Log: (Who, When, What)
# DBiesinger, 2030-Jan-01, Created File
# qinlaura, 2022-Aug-14, Updated File
# qinlaura, 2022-Aug-21, Updated script to use binary file and added 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 stored in memory'''

   @staticmethod
   def add_row_to_data(id, title, artist, table):
       '''Function to add a new row of data to the table in memory

       Taking the id, title and artist entered by the user, turn it into a dictionary
and append it to
       the table in memory.

       Args:
           id (string): the id of the new row
           title (string): the title of the cd
           artist (string): the artist of the cd
           table (list of dict): 2D data structure (list of dicts) that holds the data
during runtime

       Returns:
           None.
       '''
       int_id = int(id)
       dicRow = {'ID': int_id, 'Title': title, 'Artist': artist}
       table.append(dicRow)

   @staticmethod
   def delete_row(delete_id, table):
       '''Function to delete a row from the table in memory

       Taking the id of the row the user wants to delete, search through the table in
memory,
       check each row to see if the id matches the id of the row that the user wants
to delete,
       and remove that row from the table.

       Args:
           delete_id (string): the id of the row that user wants to delete
           table (list of dict): 2D data structure (list of dicts) that holds the data
during runtime

       Returns:
           None.
       '''
       int_delete_id = int(delete_id)
       row_num = -1
       cd_removed = False
       for row in table:
           row_num += 1
           if row['ID'] == int_delete_id:
               del table[row_num]
               cd_removed = True
               break
       if cd_removed:
           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 file to a list of dictionaries

       Reads the data from the pickle 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:
           table (list of dict): 2D data structure (list of dicts) that holds the data
       '''
       with open(file_name, 'rb') as obj_file:
           table = pickle.load(obj_file)
       return table

   @staticmethod
   def write_file(file_name, table):
       '''Function to save the table in memory into the file

       Write the data that is a 2D list of dictionaries into a pickle file
       Args:
           file_name (string): name of file to save data to
           table (list of dict): 2D data structure (list of dicts) that holds the data
during runtime

       Returns:
           None.
       '''

       with open(file_name, 'wb') as obj_file:
           pickle.dump(table, obj_file)

# -- 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']:
           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 get_input_row():
       '''Gets user input for a new row of data

       Prompts the user to enter the id, title and artist for a new row of data, and
then store the entered values
       into three seperate string variables

       Args:
           None.

       Returns:
           id, title, artist (a tuple of strings): the id, title and artist entered by
a user, stored in strings
           inside a tuple
       '''
       id = input('Enter ID: ').strip()
       title = input('What is the CD\'s title? ').strip()
       artist = input('What is the Artist\'s name? ').strip()
       return id, title, artist

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

   @staticmethod
   def get_reload_yes_no():
       '''Ask user to confirm if they want to reload data, save user entry in a string

       Prints out a warning that says if the user procees to reload data from file,
all unsaved changes in
       memory will be discarded. Then it asks user to confirm they want to proceed by
typing in 'yes',
       otherwise abort the reload.

       Args:
           None.

       Returns:
           reload_yes_no (string): whether or not user wants to proceed with reloading
data
       '''
       print('WARNING: If you continue, all unsaved data will be lost and the
Inventory re-loaded from file.')
       reload_yes_no = input('type \'yes\' to continue and reload from file. otherwise
reload will be canceled: ')
       return reload_yes_no

   @staticmethod
   def get_delete_id():
       '''Ask user to enter the id of the row they would like to delete, and store it
in a string

       Args:
           None.

       Returns:
           del_id (string): the id of the row to be deleted from the table
       '''
       del_id = input('Which ID would you like to delete? ').strip()
       return del_id
   @staticmethod
   def get_save_yes_no():
       '''Ask user to whether they want to save the current inventory to file, and
store it in a string

       Args:
           None.

       Returns:
           save_yes_no (string): whether or not the user wants to save current
inventory to file
       '''
       save_yes_no = input('Save this inventory to file? [y/n] ').strip().lower()
       return save_yes_no



# 1. When program starts, read in the currently saved Inventory
try:
   lstTbl = FileProcessor.read_file(strFileName)
except FileNotFoundError as e:
   print('\nInventory data file does not exist')
   print('Detailed error message: ')
   print(type(e), e, e.__doc__, sep = '\n')

# 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':
       try:
           strYesNo = IO.get_reload_yes_no()
           if strYesNo.lower() == 'yes':
               print('reloading...\n')
               lstTbl = 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.
       except FileNotFoundError as e:
           print('\nInventory data file does not exist')
           print('Detailed error message: ')
           print(type(e), e, e.__doc__, sep = '\n')
       except Exception as e:
           print('\nThere was a general error')
           print('Detailed error message: ')
           print(type(e), e, e.__doc__, sep = '\n')
   # 3.3 process add a CD
   elif strChoice == 'a':
       try:
           # 3.3.1 Ask user for new ID, CD Title and Artist
           strID, strTitle, strArtist = IO.get_input_row()
           # 3.3.2 Add item to the table
           DataProcessor.add_row_to_data(strID, strTitle, strArtist, lstTbl)
           IO.show_inventory(lstTbl)
           continue  # start loop back at top.
       except ValueError as e:
           print('\nInvalid ID, needs to be an integer')
           print('Detailed error message: ')
           print(type(e), e, e.__doc__, sep = '\n')
       except Exception as e:
           print('\nThere was a general error')
           print('Detailed error message: ')
           print(type(e), e, e.__doc__, sep = '\n')
   # 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':
       try:
           # 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 = IO.get_delete_id()
           # 3.5.2 search thru table and delete CD
           DataProcessor.delete_row(intIDDel, lstTbl)
           IO.show_inventory(lstTbl)
           continue  # start loop back at top.
       except ValueError as e:
           print('\nInvalid ID, needs to be an integer')
           print('Detailed error message: ')
           print(type(e), e, e.__doc__, sep = '\n')
       except Exception as e:
           print('\nThere was a general error')
           print('Detailed error message: ')
           print(type(e), e, e.__doc__, sep = '\n')
   # 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 = IO.get_save_yes_no()
       # 3.6.2 Process choice
       if strYesNo == 'y':
           # 3.6.2.1 save data
           FileProcessor.write_file(strFileName, 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')
```

2. Github link
   https://github.com/NoraQin/Assignment_07