Jeffrey Shen

3/8/2020

Foundations of Programming (Python)

Assignment 07

Modules and Structured Error Handling with Python

Introduction

In this document, I will provide an overview of using external Python modules and structured error handling. This report will include discussion of the CDInventory_07.py script and what challenges I came across.

CD Inventory Script

Using Pickle Module

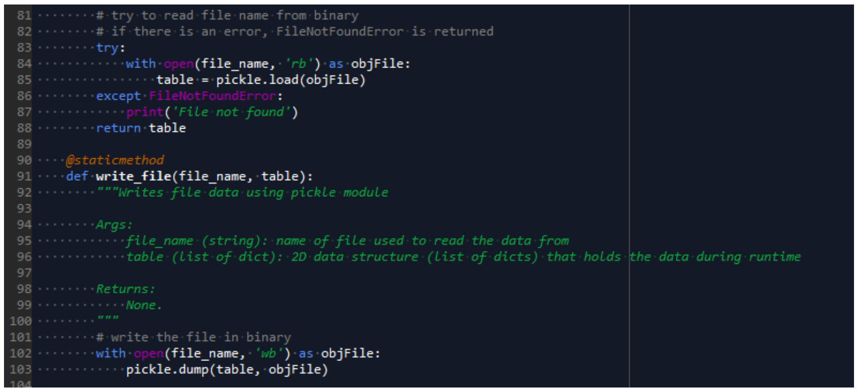
Building upon the last few week's assignments of managing a CD inventory based on user input, I integrated a module and structured error handling. I will only describe the changes included to the script, since majority of the pseudocode and functionality stayed the same.

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

```
if strChoice == 'x'
    do this
elif strChoice == 'l'
    do this
elif strChoice == 'a'
    do this
elif strChoice == 'i'
    do this
elif strChoice == 'd'
    do this
elif strChoice == 's'
    do this
```

I used the pickle module for this assignment. Pickle implements binary protocols for serializing and de-serializing Python object structure.[1] Python is able to interpret and convert this information between byte stream and object hierarchy. To use the pickle module, I simple used the syntax: "import pickle" There is documentation on how to use pickle since it has certain attributes (e.g. pickle.load, pickle.dump)[2] Here is a snippet of my code utilizing pickle for file operations.

[1] https://docs.python.org/3/library/pickle.html
[2] https://wiki.python.org/moin/UsingPickle



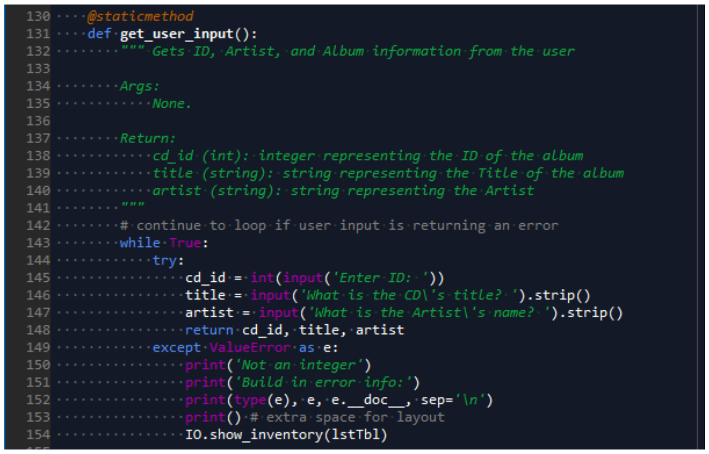
```
81         # try to read file name from binary
82         # if there is an error, FileNotFoundError is returned
83         try:
84             with open(file_name, 'rb') as objFile:
85                 table = pickle.load(objFile)
86         except FileNotFoundError:
87             print('File not found')
88         return table
89
90     @staticmethod
91     def write_file(file_name, table):
92         """Writes file data using pickle module
93
94         Args:
95             file_name (string): name of file used to read the data from
96             table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
97
98         Returns:
99             None.
100        """
101        # write the file in binary
102        with open(file_name, 'wb') as objFile:
103            pickle.dump(table, objFile)
```

Figure 1 Pickle Module

The try-except error handling structure will be discuss later in this report. I will focus on the pickle syntax for now. I used the same "with open…" syntax to call out what file I want to operate on. However, in this case, in line 84 I use 'rb' which stands for read binary. This tells Python that this is not a text file. I had a lot of difficulty understanding and implementing this. I kept on trying to use pickle to read text file. The table is populated with the data inside the file by using "pickle.load(objFile)" in line 85. The same type of syntax is used to save a file but in this case, I use 'wb' which stands for write binary. This tells Python to write in binary format, and I call out "pickle.dump" to use the pickle module. Using pickle simplifies the arguments and parameters we need to pass through as it is inherently part of the Python module.

Structured Error Handling

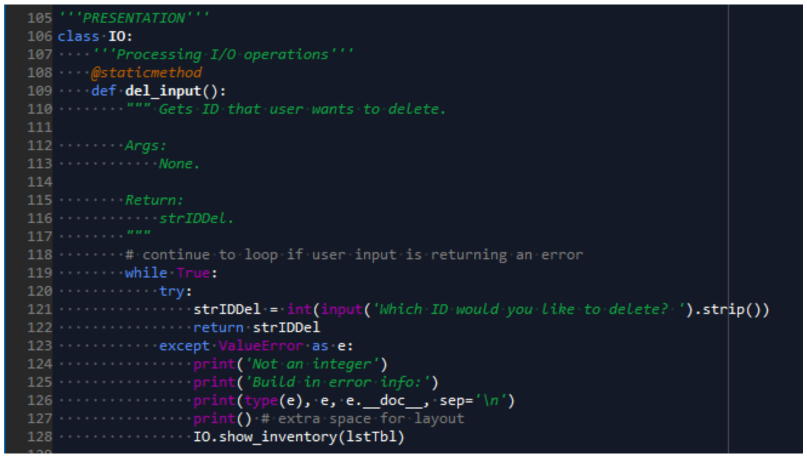
The idea behind this is to learn how to handle errors in my Python code. There can be different types of errors: ZeroDivisionError, TypeError, ValueError, FileNotFoundError, Exception, and more. The general syntax is to use a try-except format.



```
130     @staticmethod
131     def get_user_input():
132         """ Gets ID, Artist, and Album information from the user
133
134         Args:
135             None.
136
137         Return:
138             cd_id (int): integer representing the ID of the album
139             title (string): string representing the Title of the album
140             artist (string): string representing the Artist
141         """
142         # continue to loop if user input is returning an error
143         while True:
144             try:
145                 cd_id = int(input('Enter ID: '))
146                 title = input('What is the CD\'s title? ').strip()
147                 artist = input('What is the Artist\'s name? ').strip()
148                 return cd_id, title, artist
149             except ValueError as e:
150                 print('Not an integer')
151                 print('Build in error info:')
152                 print(type(e), e, e.__doc__, sep='\n')
153                 print() # extra space for layout
154                 IO.show_inventory(lstTbl)
```

Figure 2 Try-Except

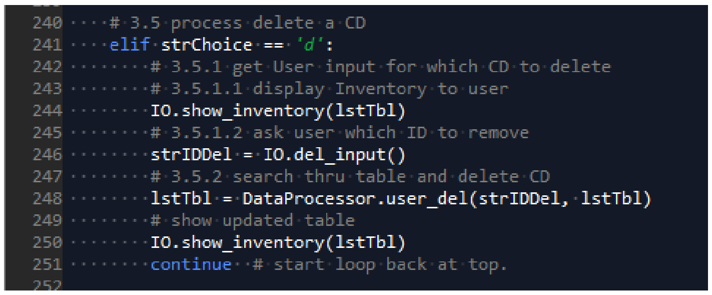
For example, in my get_user_input() function, line 144 calls out "try:" Following in lines 145-148 is the parameters that I would like to receive from the user. However, if there is an error passed (in this case if cd_id is not an integer), the program will proceed to line 149 except clause. Lines 150-153 will output what the error code was and how the user can proceed. I included line 153-154 for formatting and helping remind the user what is in the table. At first, I included multiple try-except structures to cycle through, then realized, the user can indefinitely make wrong inputs so I created a while loop.



```python
105 '''PRESENTATION'''
106 class IO:
107     '''Processing I/O operations'''
108     @staticmethod
109     def del_input():
110         """ Gets ID that user wants to delete.
111 
112         Args:
113             None.
114 
115         Return:
116             strIDDel.
117         """
118         # continue to loop if user input is returning an error
119         while True:
120             try:
121                 strIDDel = int(input('Which ID would you like to delete? ').strip())
122                 return strIDDel
123             except ValueError as e:
124                 print('Not an integer')
125                 print('Build in error info:')
126                 print(type(e), e, e.__doc__, sep='\n')
127                 print() # extra space for layout
128                 IO.show_inventory(lstTbl)
```

Figure 3 del_input() function

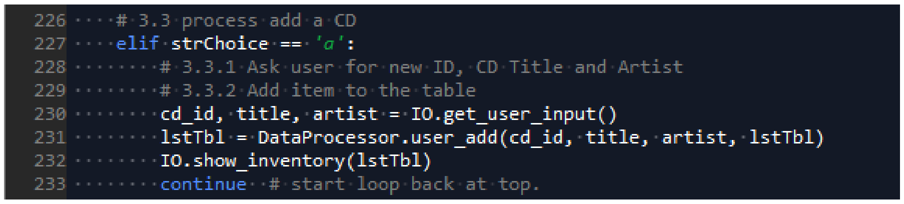
The same idea is implemented here in my function del_input(). The user needs to input an integer for ID or there will be an except clause that outputs error messages. I actually included this del_input() function as I didn't create it from last week. I noticed that it could be included in my IO class. In line 246, the function is called out.



```python
240     # 3.5 process delete a CD
241     elif strChoice == 'd':
242         # 3.5.1 get User input for which CD to delete
243         # 3.5.1.1 display Inventory to user
244         IO.show_inventory(lstTbl)
245         # 3.5.1.2 ask user which ID to remove
246         strIDDel = IO.del_input()
247         # 3.5.2 search thru table and delete CD
248         lstTbl = DataProcessor.user_del(strIDDel, lstTbl)
249         # show updated table
250         IO.show_inventory(lstTbl)
251         continue  # start loop back at top.
```

Figure 4 Calling out del_input function

Another change I included from last week was how the user add inputs from the main while loop. Originally, in line 231 was "lstTbl = DataProcessor.user_add(*IO.get_user_input(), lstTbl)" however, I kept on getting errors on non-iterable values, if the IO.get_user_input() pushed an error to the user. I broke it out to two lines so I could manage that better.



```python
226     # 3.3 process add a CD
227     elif strChoice == 'a':
228         # 3.3.1 Ask user for new ID, CD Title and Artist
229         # 3.3.2 Add item to the table
230         cd_id, title, artist = IO.get_user_input()
231         lstTbl = DataProcessor.user_add(cd_id, title, artist, lstTbl)
232         IO.show_inventory(lstTbl)
233         continue  # start loop back at top.
```

Figure 5 "Adding" function calls

Example runs from Spyder and the terminal are included in the Appendix.

Questions

• What are the benefits of using structured error handling?

- The error messages can be integrated into the program and allow better readability/debugging. The idea of structured error handling helps programmer better understands the error messages the Python retrieves.

• What are the differences between a text file and a binary file?

- A text file is readable to humans (generally, or that is the assumption). Generally including numbers, strings. A binary file is what a sequence of bytes (binary digits, bits). A computer can read binary formats and interpret it back to the user. Using a binary reader is needed to understand.

• How is the Exception class used?

- Exception classes are utilized for error handling. They can be implemented and called out using “raise” which the program will then call out the exception class.

• How do you "derive" a new class from the Exception class?

- Derived classes are inherited from the base class. They can be customized to meet specific needs.

• When might you create a class derived from the Exception class?

- If there are specific exceptions, you want to call out or help the program go back to a “Traceback”

• What is the Markdown language?

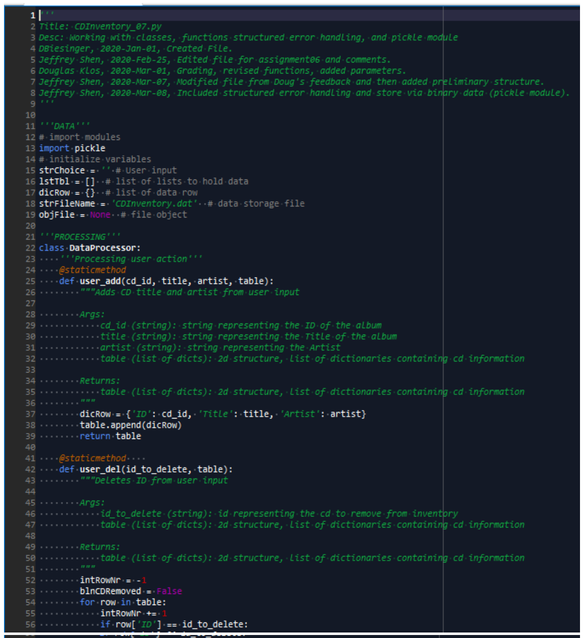
- It’s a way of organizing the readme.md file to allow better understanding and consistency across programmers to view what is included in a repository.

Summary

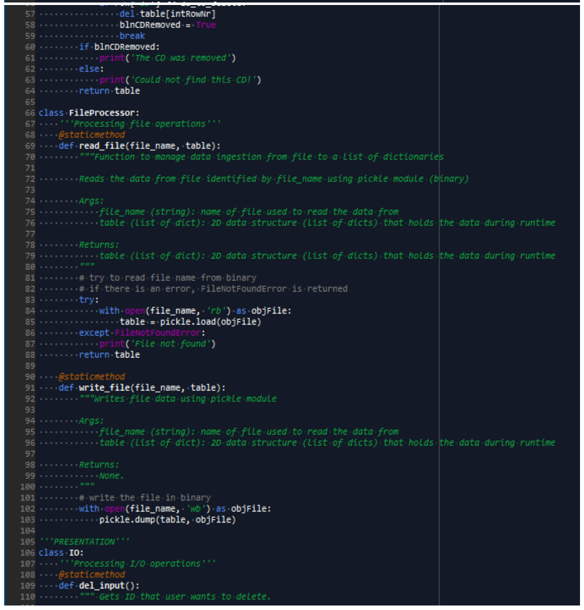
In this lab, I explored using modules and structured error handling to enhance last week’s assignment. The overall functionality stayed the same for managing the CD inventory but included best practices. In the beginning, I struggled to understand and digest how to use the pickle module. I could not integrate saving and loading accurately. Comprehending the syntax and using module attributes proved to be a more challenging task than I thought. Structured error handling was a much easier concept to grasp as it was similar to if-else syntax but instead used try-except clauses.

Appendix

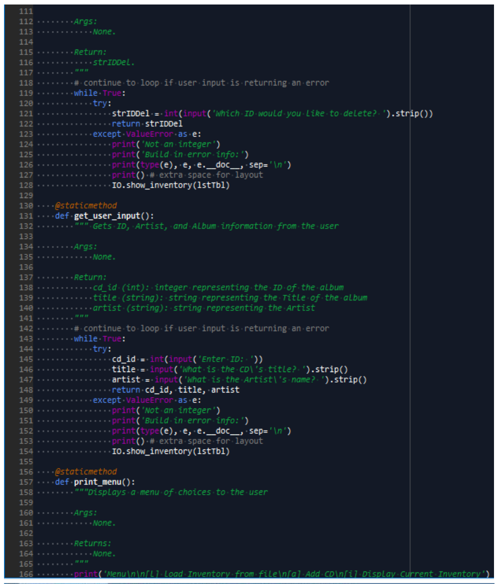
Complete Code for CDInventory_07.py



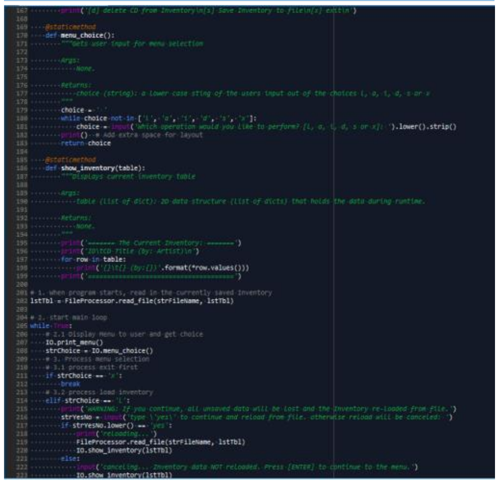
```python
'''
Title: CDInventory_07.py
Desc: Working with classes, functions structured error handling, and pickle module
DBiesinger, 2020-Jan-01, Created File.
Jeffrey Shen, 2020-Feb-25, Edited file for assignment06 and comments.
Douglas Klos, 2020-Mar-01, Grading, revised functions, added parameters.
Jeffrey Shen, 2020-Mar-07, Modified file from Doug's feedback and then added preliminary structure.
Jeffrey Shen, 2020-Mar-08, Included structured error handling and store via binary data (pickle module).
'''

'''DATA'''
# import modules
import pickle
# initialize variables
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 user action'''
    @staticmethod
    def user_add(cd_id, title, artist, table):
        """Adds CD title and artist from user input

        Args:
            cd_id (string): string representing the ID of the album
            title (string): string representing the Title of the album
            artist (string): string representing the Artist
            table (list of dicts): 2d structure, list of dictionaries containing cd information

        Returns:
            table (list of dicts): 2d structure, list of dictionaries containing cd information
        """
        dicRow = {'ID': cd_id, 'Title': title, 'Artist': artist}
        table.append(dicRow)
        return table

    @staticmethod    
    def user_del(id_to_delete, table):
        """Deletes ID from user input

        Args:
            id_to_delete (string): id representing the cd to remove from inventory
            table (list of dicts): 2d structure, list of dictionaries containing cd information

        Returns:
            table (list of dicts): 2d structure, list of dictionaries containing cd information
        """
        intRowNr = -1
        blnCDRemoved = False
        for row in table:
            intRowNr += 1
            if row['ID'] == id_to_delete:
```



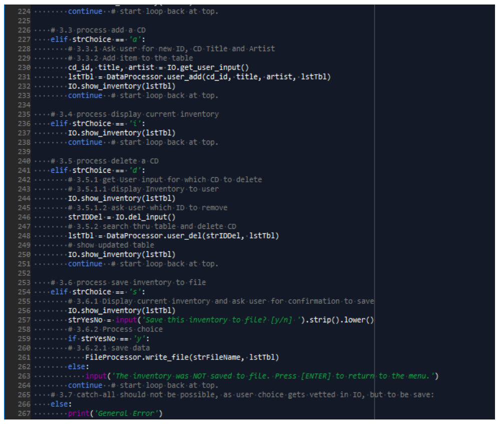
```python
                del table[intRowNr]
                blnCDRemoved = True
                break
        if blnCDRemoved:
            print('The CD was removed')
        else:
            print('Could not find this CD!')
        return table

class FileProcessor:
    '''Processing file operations'''
    @staticmethod
    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 using pickle module (binary)

        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:
            table (list of dict): 2D data structure (list of dicts) that holds the data during runtime
        """
        # try to read file name from binary
        # if there is an error, FileNotFoundError is returned
        try:
            with open(file_name, 'rb') as objFile:
                table = pickle.load(objFile)
        except FileNotFoundError:
            print('File not found')
        return table

    @staticmethod
    def write_file(file_name, table):
        """Writes file data using pickle module

        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.
        """
        # write the file in binary
        with open(file_name, 'wb') as objFile:
            pickle.dump(table, objFile)

'''PRESENTATION'''
class IO:
    '''Processing I/O operations'''
    @staticmethod
    def del_input():
        """ Gets ID that user wants to delete.
```



```python

        Args:
            None.

        Return:
            strIDDel.
        """
        # continue to loop if user input is returning an error
        while True:
            try:
                strIDDel = int(input('Which ID would you like to delete? ').strip())
                return strIDDel
            except ValueError as e:
                print('Not an integer')
                print('Build in error info:')
                print(type(e), e, e.__doc__, sep='\n')
                print() # extra space for layout
                IO.show_inventory(lstTbl)

    @staticmethod
    def get_user_input():
        """ Gets ID, Artist, and Album information from the user

        Args:
            None.

        Return:
            cd_id (int): integer representing the ID of the album
            title (string): string representing the Title of the album
            artist (string): string representing the Artist
        """
        # continue to loop if user input is returning an error
        while True:
            try:
                cd_id = int(input('Enter ID: '))
                title = input('What is the CD\'s title? ').strip()
                artist = input('What is the Artist\'s name? ').strip()
                return cd_id, title, artist
            except ValueError as e:
                print('Not an integer')
                print('Build in error info:')
                print(type(e), e, e.__doc__, sep='\n')
                print() # extra space for layout
                IO.show_inventory(lstTbl)

    @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')
```

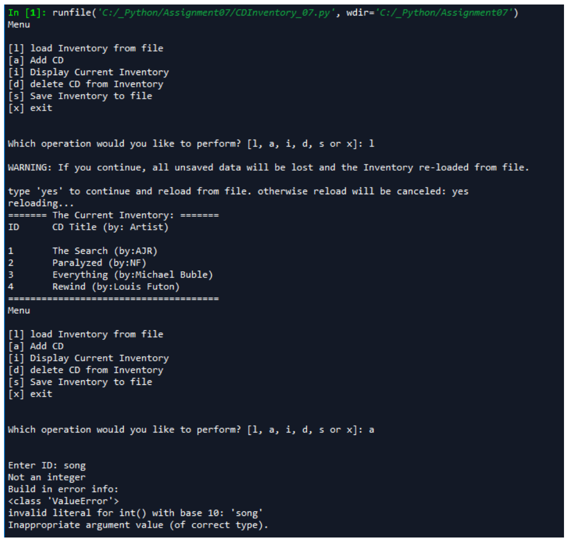


```python
        print('[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n')

    @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('======================================')

# 1. When program starts, read in the currently saved Inventory
lstTbl = FileProcessor.read_file(strFileName, 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
    elif 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...')
            FileProcessor.read_file(strFileName, lstTbl)
            IO.show_inventory(lstTbl)
        else:
            input('canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.')
            IO.show_inventory(lstTbl)
```

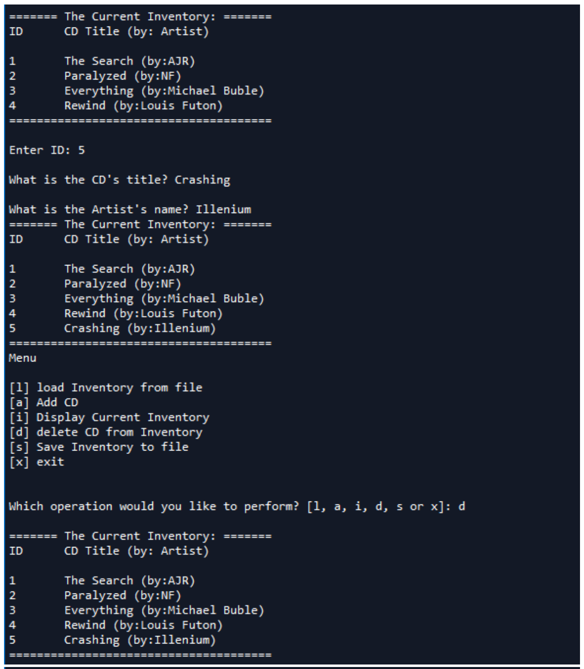


```python
        continue  # start loop back at top.

    # 3.3 process add a CD
    elif strChoice == 'a':
        # 3.3.1 Ask user for new ID, CD Title and Artist
        # 3.3.2 Add item to the table
        cd_id, title, artist = IO.get_user_input()
        lstTbl = DataProcessor.user_add(cd_id, title, artist, 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 User input 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
        strIDDel = IO.del_input()
        # 3.5.2 search thru table and delete CD
        lstTbl = DataProcessor.user_del(strIDDel, lstTbl)
        # show updated table
        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(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')
```

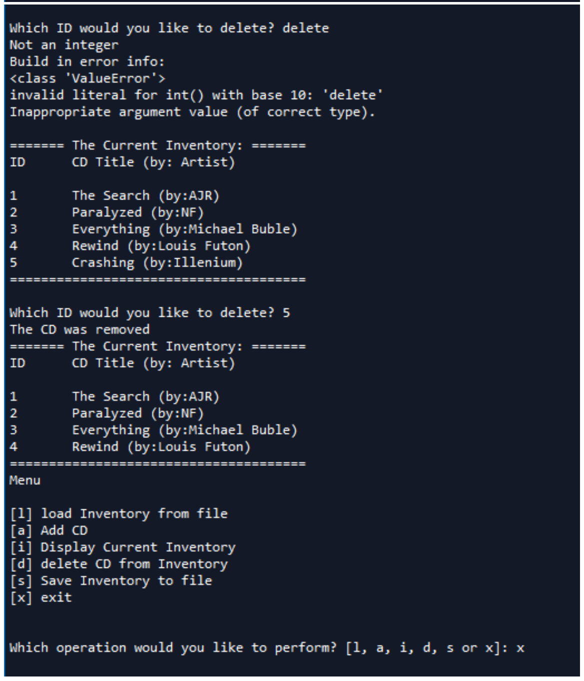
Example Run from Spyder



```
In [1]: runfile('C:/_Python/Assignment07/CDInventory_07.py', wdir='C:/_Python/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       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
======================================
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: song
Not an integer
Build in error info:
<class 'ValueError'>
invalid literal for int() with base 10: 'song'
Inappropriate argument value (of correct type).
```

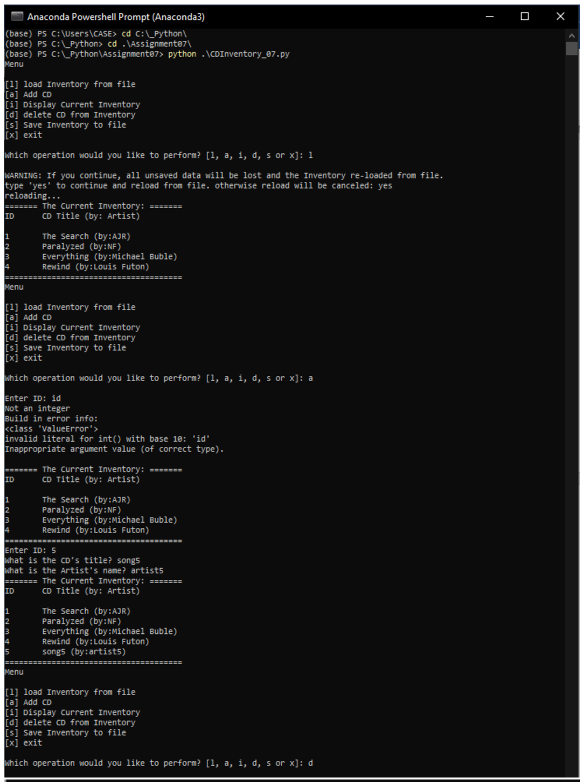


```
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
======================================

Enter ID: 5

What is the CD's title? Crashing

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

1       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
5       Crashing (by:Illenium)
======================================
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       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
5       Crashing (by:Illenium)
======================================
```



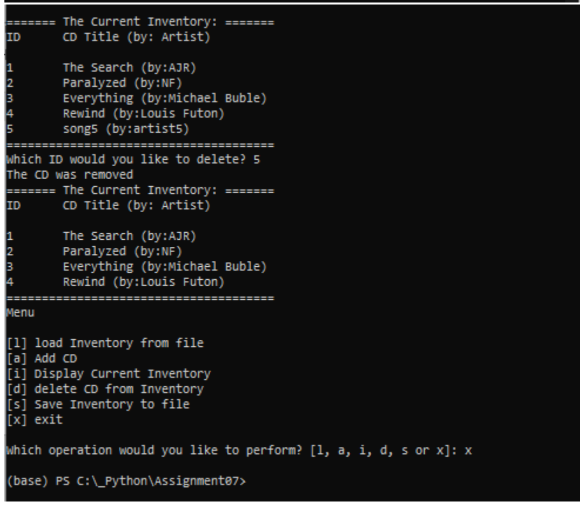
```
Which ID would you like to delete? delete
Not an integer
Build in error info:
<class 'ValueError'>
invalid literal for int() with base 10: 'delete'
Inappropriate argument value (of correct type).

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

1       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
5       Crashing (by:Illenium)
======================================

Which ID would you like to delete? 5
The CD was removed
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
======================================
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
```

Example Run from Terminal



Anaconda Powershell Prompt (Anaconda3)

```
(base) PS C:\Users\CASE> cd C:\_Python\
(base) PS C:\_Python> cd .\Assignment07\
(base) PS C:\_Python\Assignment07> python .\CDInventory_07.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       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
======================================
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: id
Not an integer
Build in error info:
<class 'ValueError'>
invalid literal for int() with base 10: 'id'
Inappropriate argument value (of correct type).

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

1       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
======================================
Enter ID: 5
What is the CD's title? song5
What is the Artist's name? artist5
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
5       song5 (by:artist5)
======================================
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       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
5       song5 (by:artist5)
======================================
Which ID would you like to delete? 5
The CD was removed
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       The Search (by:AJR)
2       Paralyzed (by:NF)
3       Everything (by:Michael Buble)
4       Rewind (by:Louis Futon)
======================================
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) PS C:\_Python\Assignment07>
```