Benjamin Reinhart

March 7, 2021

IT FDN 110 A

Assignment 08

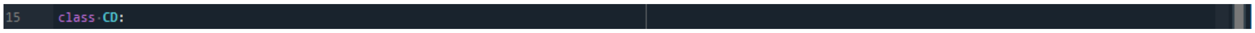
CD Class Creation

Introduction

In this assignment I will cover how to create a script in Python that manages a CD Inventory using object-oriented programming by creating a CD class. I will create the script to handle a variety of errors and read in or save data using text files. I will use the information learned in Module 08 about classes along with the concepts we have previously covered regarding the CD Inventory in order to do this. Here is a link to the GitHub repository where this is saved: https://github.com/reinhartben/Assignment_08.git

Creating the Script

In the previous CD Inventory scripts, we have used a list of dictionaries to keep track of our CDs. In this assignment we are going to create a CD class that will replace the dictionaries in the table. Classes are like a blueprint for the object they create. It allows us to build functionality into the object and manage the data stored there. When we initiate a CD in our script, each instance will have the same functionality, but will be differentiated by different attributes we give upon creation[1]. In order to create the CD class, we use the syntax in Figure 1, below.



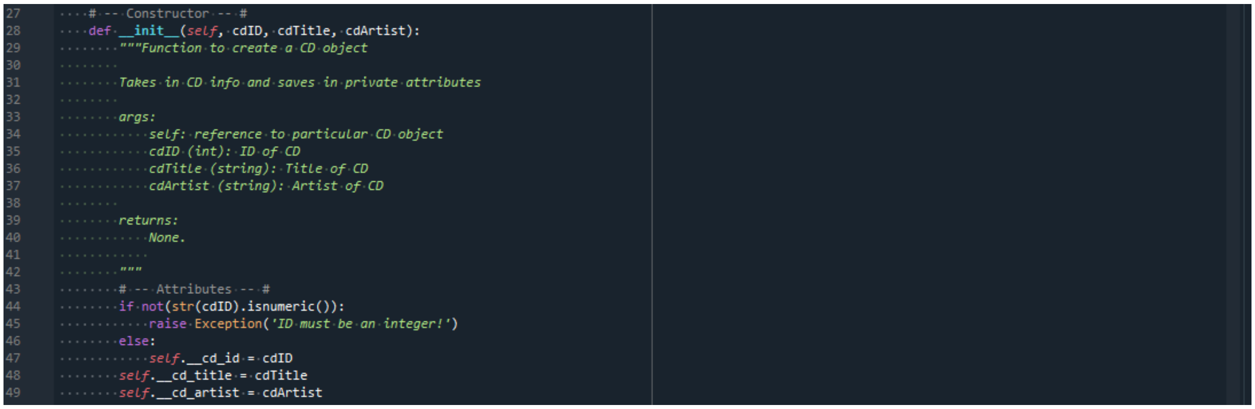
```
class CD:
```

Figure 1: Class creation

This CD class holds information about the CD and allows us to interact with the instance of the CD that we create. When we first create a CD object, the constructor function, __init__() is called. In order to differentiate our CD functionality, we overwrite the inherited, object constructor function[2]. This allows us to store our CD ID, Title, and Artist. You can see this in Figure 2, below.

[1] Dirk Biesinger, Foundations of Programming (Python), Module 08 Page 2

[2] Biesinger, Foundations of Programming (Python), Module 08 Pages 3-4

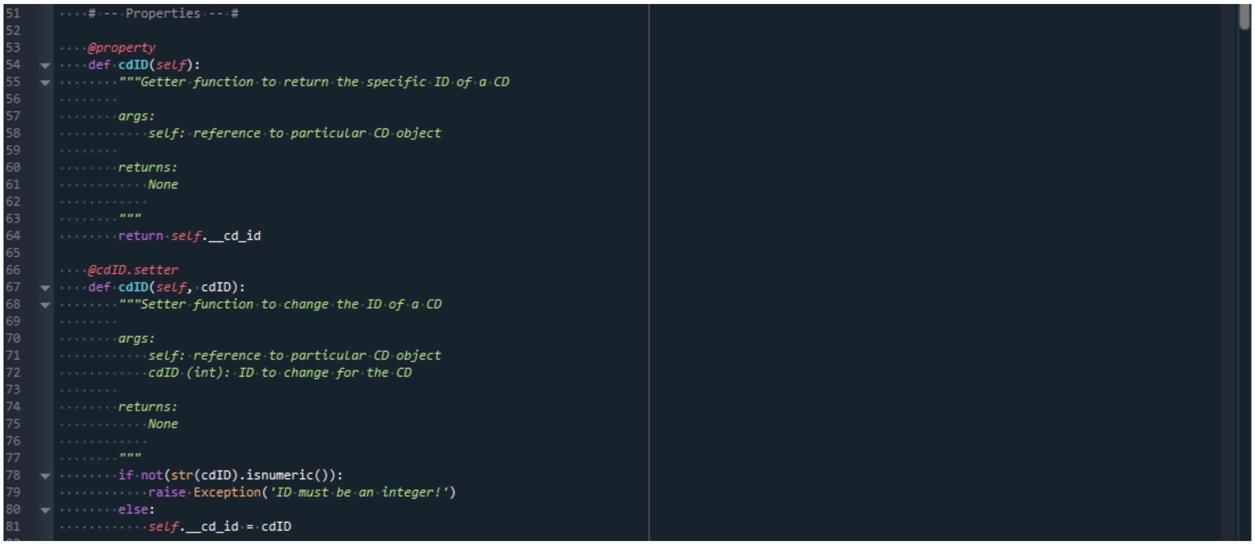


```python
    # -- Constructor -- #
    def __init__(self, cdID, cdTitle, cdArtist):
        """Function to create a CD object

        Takes in CD info and saves in private attributes

        args:
            self: reference to particular CD object
            cdID (int): ID of CD
            cdTitle (string): Title of CD
            cdArtist (string): Artist of CD

        returns:
            None.

        """
        # -- Attributes -- #
        if not(str(cdID).isnumeric()):
            raise Exception('ID must be an integer!')
        else:
            self.__cd_id = cdID
        self.__cd_title = cdTitle
        self.__cd_artist = cdArtist
```

Figure 2: Constructor Function

Notice the ‘self’ keyword in the screenshot. This self makes sure that we are referencing the specific instance of this class[3]. When we create a CD object, we give also pass in attributes that make this instance unique from other CDs[4]. In this case, it’s the CDs ID, Title, and Artist.

In order to control the validity of what gets assigned to these attributes, I made them private with the double underscore before the names then added getter and setter properties for each attribute. You can see an example of these in Figure 3.



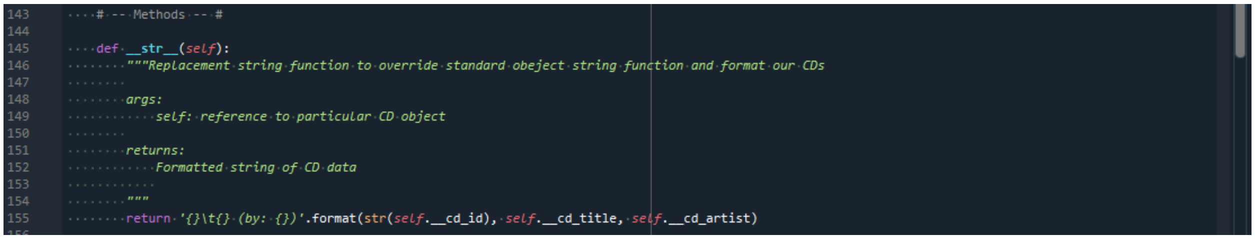
```python
    # -- Properties -- #

    @property
    def cdID(self):
        """Getter function to return the specific ID of a CD

        args:
            self: reference to particular CD object

        returns:
            None

        """
        return self.__cd_id

    @cdID.setter
    def cdID(self, cdID):
        """Setter function to change the ID of a CD

        args:
            self: reference to particular CD object
            cdID (int): ID to change for the CD

        returns:
            None

        """
        if not(str(cdID).isnumeric()):
            raise Exception('ID must be an integer!')
        else:
            self.__cd_id = cdID
```

Figure 3: Properties

The next thing I looked to add to my class was any methods required. These are the main functionality provided by our class[5]. In the Pseudocode provided, there were no methods noted, but I decided to overwrite the standard object __str__() function in order to make

[3] Biesinger, Foundations of Programming (Python), Module 08 Page 5
[4] Biesinger, Foundations of Programming (Python), Module 08 Pages 5-6
[5] Biesinger, Foundations of Programming (Python), Module 08 Page 10

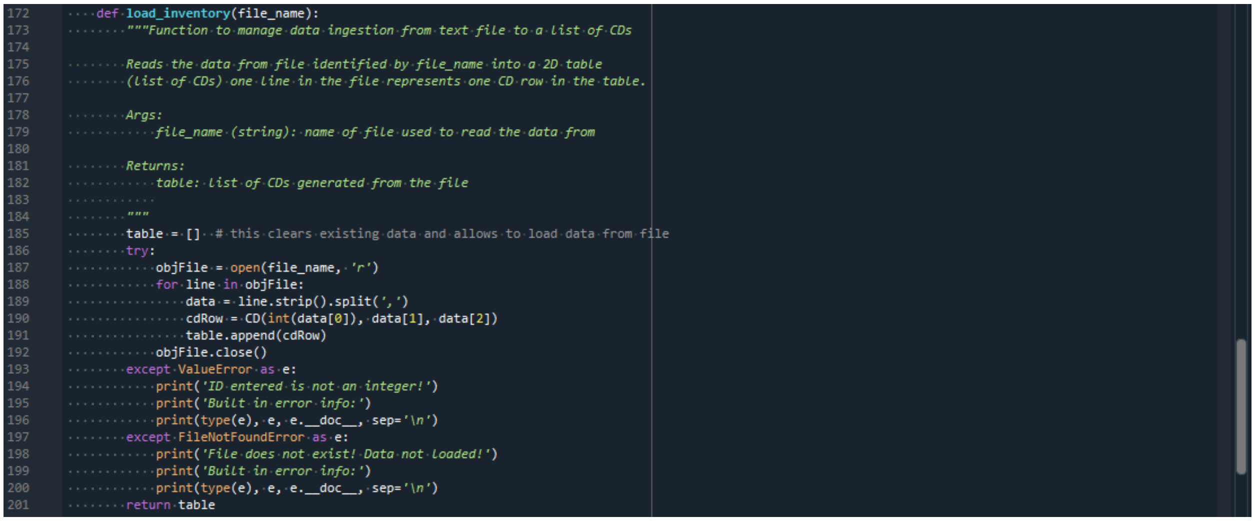
printing out my inventory easier. The __str__() function in Python typically helps return the objects data as a string, so I formatted the CD data to return the formatted string[6].



```python
    # -- Methods -- #

    def __str__(self):
        """Replacement string function to override standard obeject string function and format our CDs
        
        args:
            self: reference to particular CD object
        
        returns:
            Formatted string of CD data
            
        """
        return '{}\t{} (by: {})'.format(str(self.__cd_id), self.__cd_title, self.__cd_artist)
```

Figure 4: __str__()

This finalizes the creation of a simple CD class that helps me manage CD data more easily. For the FileIO and IO classes, I was able to use a lot of the code from previous assignments in order to add the functionality. Per the Pseudocode we were using a txt file instead of binary, so I used the file writing and reading functionality we learned through Assignment 06. Below is an example of loading the inventory from the file.



```python
    def load_inventory(file_name):
        """Function to manage data ingestion from text file to a list of CDs

        Reads the data from file identified by file_name into a 2D table
        (list of CDs) one line in the file represents one CD row in the table.

        Args:
            file_name (string): name of file used to read the data from

        Returns:
            table: list of CDs generated from the file
            
        """
        table = []  # this clears existing data and allows to load data from file
        try:
            objFile = open(file_name, 'r')
            for line in objFile:
                data = line.strip().split(',')
                cdRow = CD(int(data[0]), data[1], data[2])
                table.append(cdRow)
            objFile.close()
        except ValueError as e:
            print('ID entered is not an integer!')
            print('Built in error info:')
            print(type(e), e, e.__doc__, sep='\n')
        except FileNotFoundError as e:
            print('File does not exist! Data not loaded!')
            print('Built in error info:')
            print(type(e), e, e.__doc__, sep='\n')
        return table
```

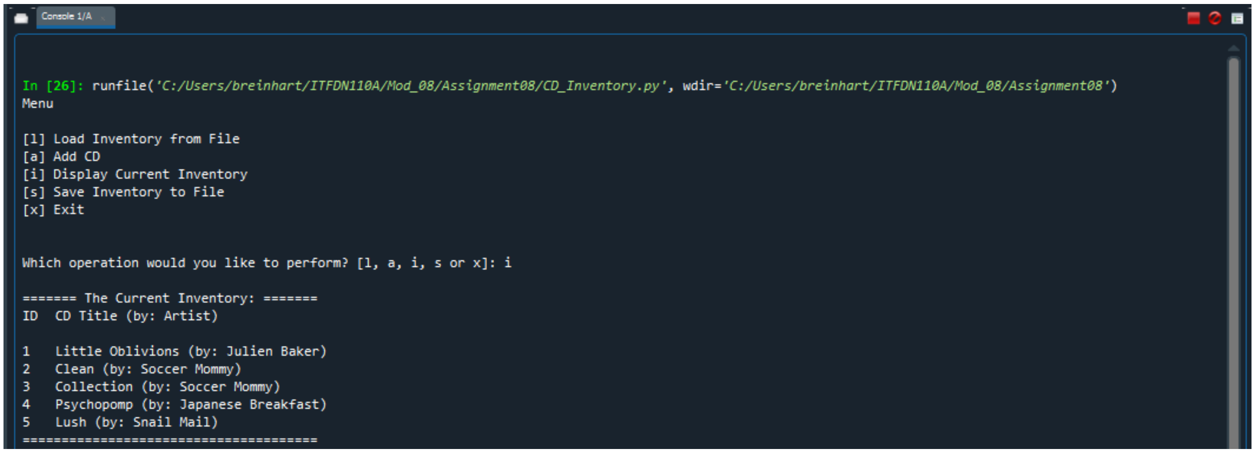
Figure 5: load_inventory

Then we also had to add in most of the main script from Assignment 07 last week, besides the ‘delete’ functionality. I did this by copying over the code from Assignment 07 and updating as needed. Now it was time to test the script.

Running the Script - Spyder

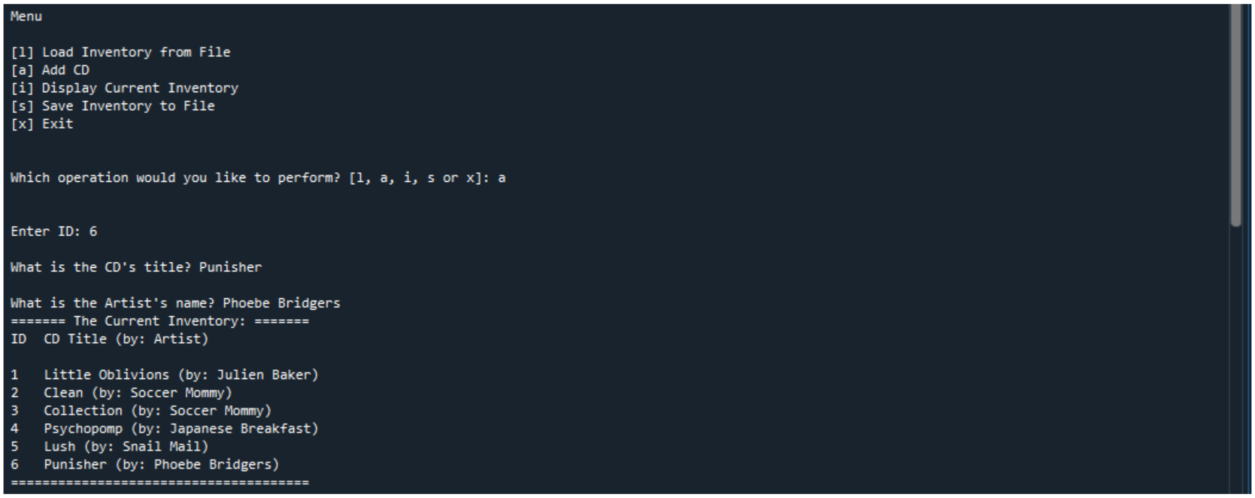
To test the script, I started in Spyder and tried all the functionality. In figures 6-9 you can see the script working in Spyder and the final text file.

[6] Biesinger, Foundations of Programming (Python), Module 08 Page 11



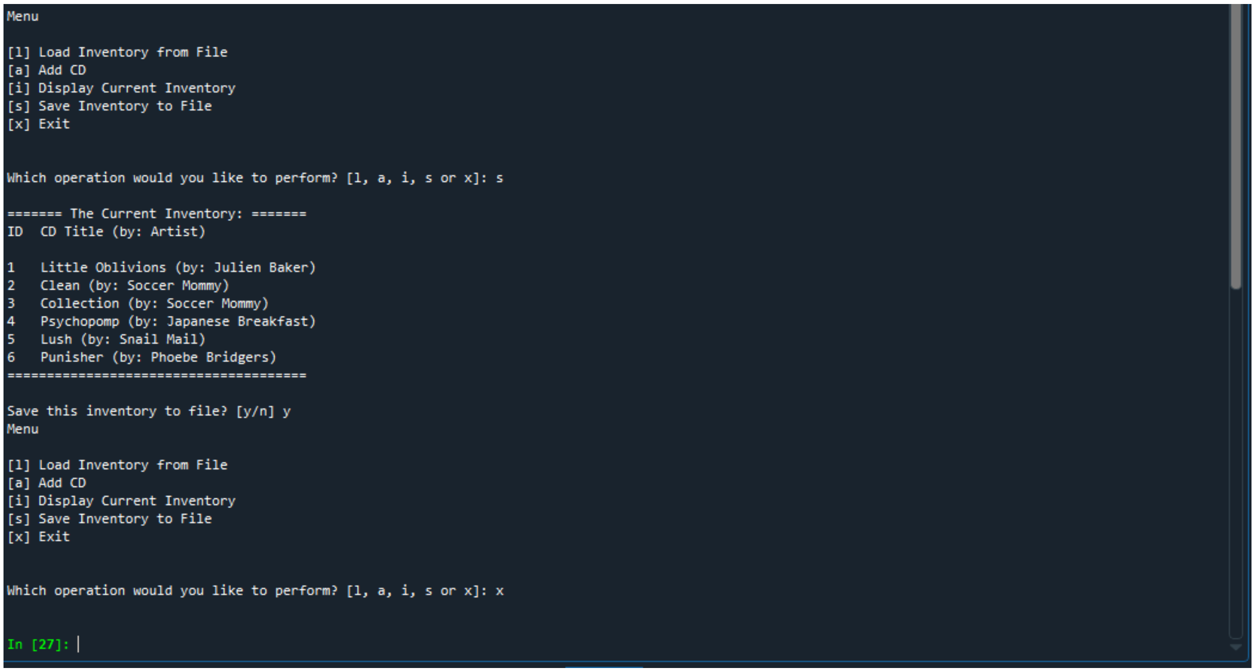
```
In [26]: runfile('C:/Users/breinhart/ITFDN110A/Mod_08/Assignment08/CD_Inventory.py', wdir='C:/Users/breinhart/ITFDN110A/Mod_08/Assignment08')
Menu

[l] Load Inventory from File
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to File
[x] Exit


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

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

1   Little Oblivions (by: Julien Baker)
2   Clean (by: Soccer Mommy)
3   Collection (by: Soccer Mommy)
4   Psychopomp (by: Japanese Breakfast)
5   Lush (by: Snail Mail)
======================================
```

Figure 6: Display Inventory - Spyder



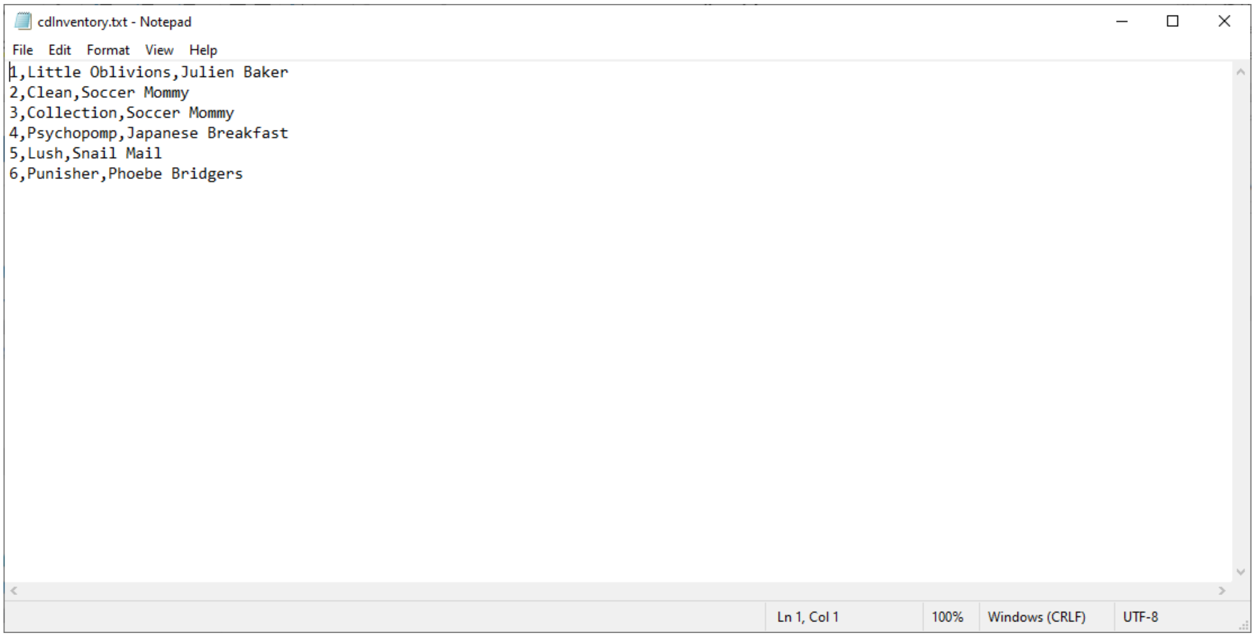
```
Menu

[l] Load Inventory from File
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to File
[x] Exit


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


Enter ID: 6

What is the CD's title? Punisher

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

1   Little Oblivions (by: Julien Baker)
2   Clean (by: Soccer Mommy)
3   Collection (by: Soccer Mommy)
4   Psychopomp (by: Japanese Breakfast)
5   Lush (by: Snail Mail)
6   Punisher (by: Phoebe Bridgers)
======================================
```

Figure 7: Add CD - Spyder



```
Menu

[l] Load Inventory from File
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to File
[x] Exit


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

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

1   Little Oblivions (by: Julien Baker)
2   Clean (by: Soccer Mommy)
3   Collection (by: Soccer Mommy)
4   Psychopomp (by: Japanese Breakfast)
5   Lush (by: Snail Mail)
6   Punisher (by: Phoebe Bridgers)
======================================

Save this inventory to file? [y/n] y
Menu

[l] Load Inventory from File
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to File
[x] Exit


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


In [27]: 
```

Figure 8: Save Inventory and Exit - Spyder

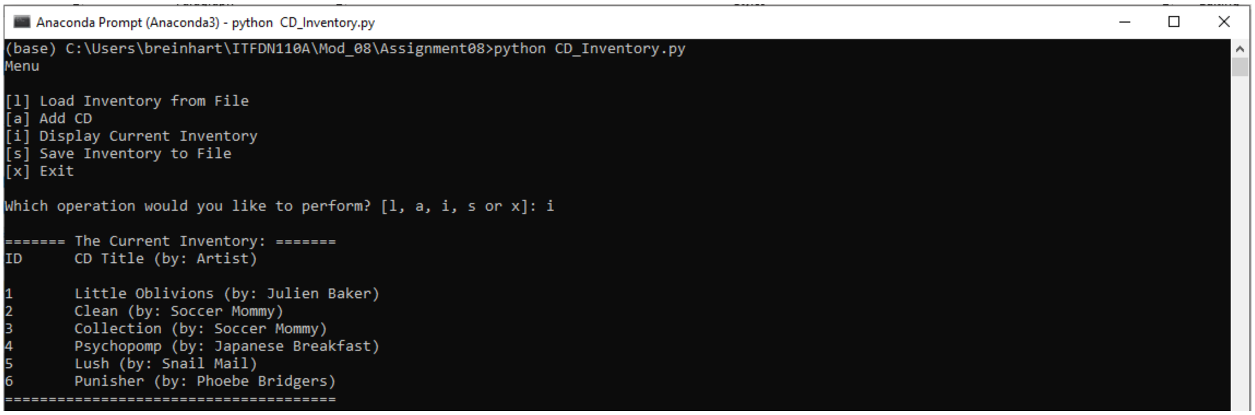


```
1,Little Oblivions,Julien Baker
2,Clean,Soccer Mommy
3,Collection,Soccer Mommy
4,Psychopomp,Japanese Breakfast
5,Lush,Snail Mail
6,Punisher,Phoebe Bridgers
```

Figure 9: Text File After Successful Spyder Run

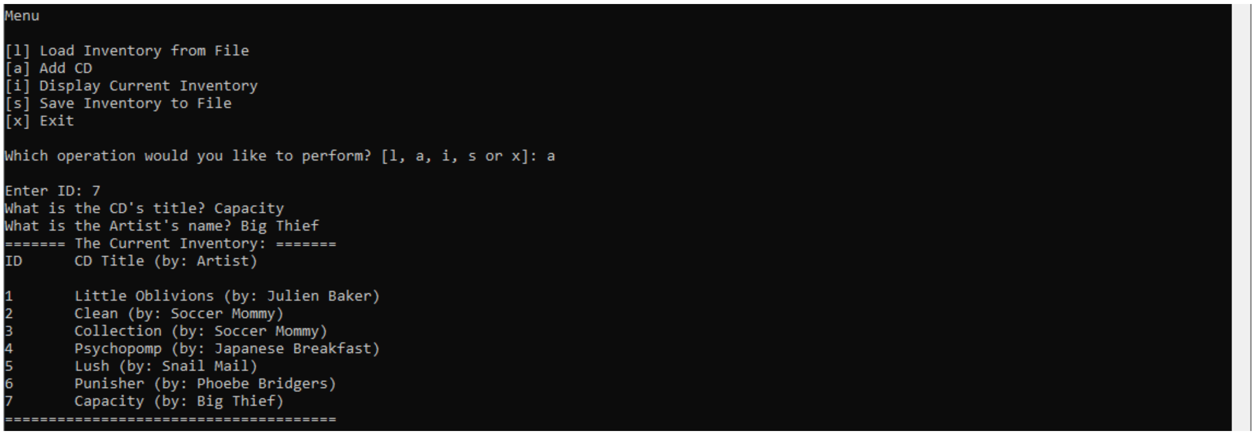
Running the Script - Terminal

We also ran the program through the Terminal to finish testing. Below you can see the screenshots from the successful run along with the resulting text file.



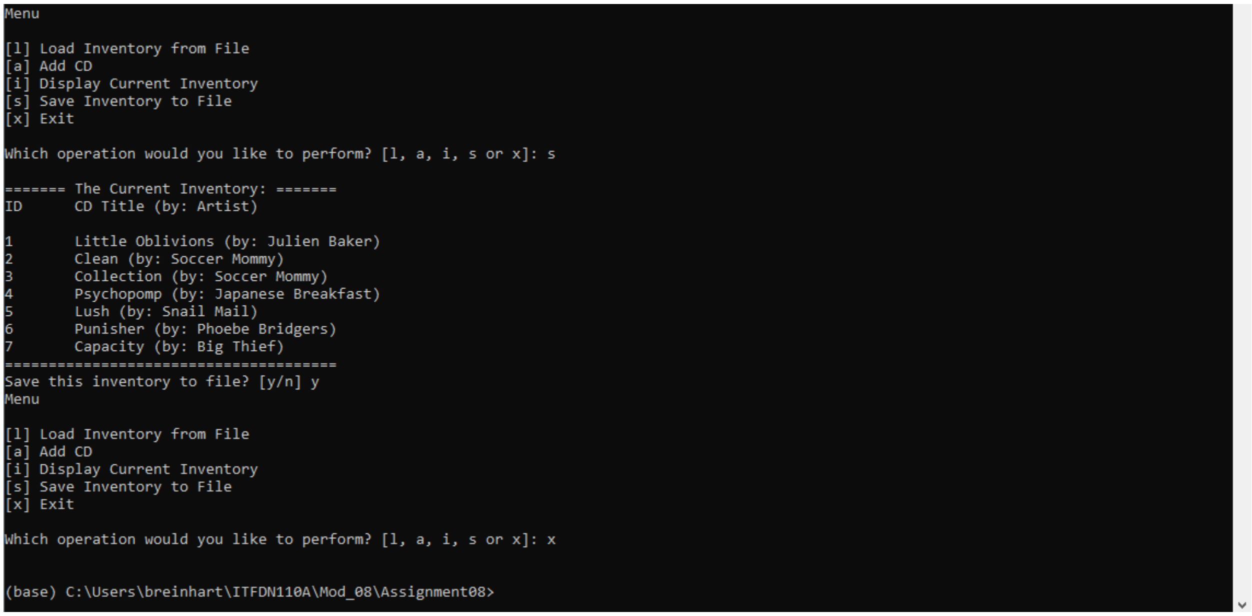
```
(base) C:\Users\breinhart\ITFDN110A\Mod_08\Assignment08>python CD_Inventory.py
Menu

[l] Load Inventory from File
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to File
[x] Exit

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

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

1       Little Oblivions (by: Julien Baker)
2       Clean (by: Soccer Mommy)
3       Collection (by: Soccer Mommy)
4       Psychopomp (by: Japanese Breakfast)
5       Lush (by: Snail Mail)
6       Punisher (by: Phoebe Bridgers)
======================================
```

Figure 10: Display Inventory - Terminal



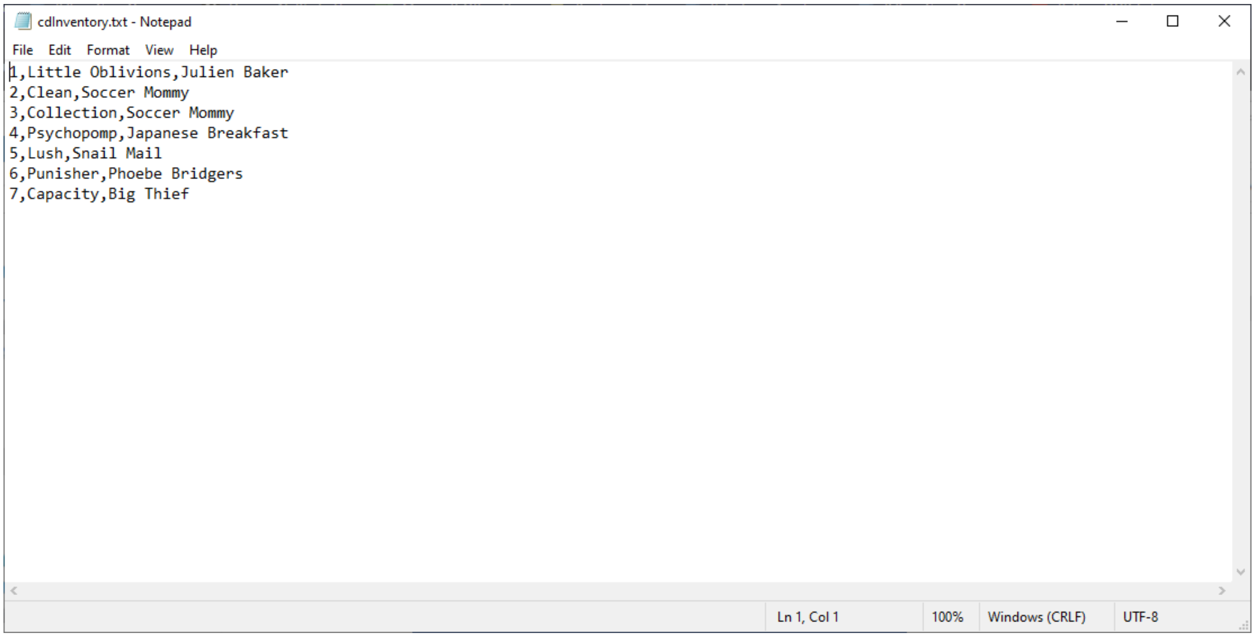
```
Menu

[l] Load Inventory from File
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to File
[x] Exit

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

Enter ID: 7
What is the CD's title? Capacity
What is the Artist's name? Big Thief
======= The Current Inventory: =======
ID      CD Title (by: Artist)

1       Little Oblivions (by: Julien Baker)
2       Clean (by: Soccer Mommy)
3       Collection (by: Soccer Mommy)
4       Psychopomp (by: Japanese Breakfast)
5       Lush (by: Snail Mail)
6       Punisher (by: Phoebe Bridgers)
7       Capacity (by: Big Thief)
======================================
```

Figure 11: Add CD - Terminal



```
Menu

[l] Load Inventory from File
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to File
[x] Exit

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

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

1       Little Oblivions (by: Julien Baker)
2       Clean (by: Soccer Mommy)
3       Collection (by: Soccer Mommy)
4       Psychopomp (by: Japanese Breakfast)
5       Lush (by: Snail Mail)
6       Punisher (by: Phoebe Bridgers)
7       Capacity (by: Big Thief)
======================================
Save this inventory to file? [y/n] y
Menu

[l] Load Inventory from File
[a] Add CD
[i] Display Current Inventory
[s] Save Inventory to File
[x] Exit

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


(base) C:\Users\breinhart\ITFDN110A\Mod_08\Assignment08>
```

Figure 12: Save Inventory and Exit - Terminal



```
1,Little Oblivions,Julien Baker
2,Clean,Soccer Mommy
3,Collection,Soccer Mommy
4,Psychopomp,Japanese Breakfast
5,Lush,Snail Mail
6,Punisher,Phoebe Bridgers
7,Capacity,Big Thief
```

Figure 13: Text File After Successful Terminal Run

Summary

Given the information provided throughout Module 08, I was able to create a script that creates a CD class to help a user manage a CD inventory.