Jesse Kim

11/26/2022

UW PCE – IT FDN 110

Assignment 07

Assignment 7 - Exception Handling and Binary Storage/Access

## Introduction

This week’s lesson and modules covered reading and writing to text files as well as different ways to access and store the data in those files. Accessing and storing data via pickle was a new method we learned to work with binary data. Finally, we looked at adding structured exception handling within our code to allow a user to work around known errors that might come up when running various code blocks.

## Exception Handling Research

So I pulled up this [video](https://www.bing.com/videos/search?q=exception+handline+python&view=detail&mid=B3594AF7164CBD6EF5FEB3594AF7164CBD6EF5FE&FORM=VIRE) given that videos can be much more efficient at explaining a topic than text. Yes, I use Bing not ashamed of it; I use Google suite for work and like to keep my school work in a separate browser altogether so it’s a necessary evil. But I digress, I like the comparison of how exception handling can be like driving to a destination. Especially given I used to deliver flowers so knowing how to get around certain areas of the city could be very useful so that made a lot of sense to me. I also looked at this [site](https://www.geeksforgeeks.org/python-exception-handling/) to see how they tried to get the lesson across but found their examples to be a little too generic.

What’s funny is I started to understand it a lot better when I started adding it into my code. Initially I understood it as a sub-block of code after the input line and kept having to duplicate inputs when running the code and it finally dawned on me that the try/except should be written around the input line to catch the possibility of an exception.

## Pickling Research

I found the concept of pickling straightforward, however, it was difficult to implement into the code and I was never able to find a way to get the read\_file function to work when trying to load a pickle file despite trying multiple examples of code.

Compared to the lesson in Chapter 7 of the book we use for class, I didn’t think any source was better or worse. Some examples were a little too complex for beginners whereas others were approximately the same as what we see in the module lesson and textbook. I did like that this [site](https://www.geeksforgeeks.org/pickle-python-object-serialization/) offered a quick look at how to handle an EOF Error when loading a file using pickle. This is an error I ran into quite a bit while working on my code.

## Assignment 7

This week’s assignment covered the previous code where we updated our script to allow a user to access and edit a list of stored compact discs (CDs) as well as write the data to a file. Here we reviewed different areas where a user might run into a known error and attempted to write an exception for those errors so a user would not have to exit the code and start over.

Exceptions

The first exception I looked to create was within the function that requests the CD data from a user. Specifically, where the code converts a user input to an integer. Here I specified that it should account for a ValueError and instruct the user to continue.

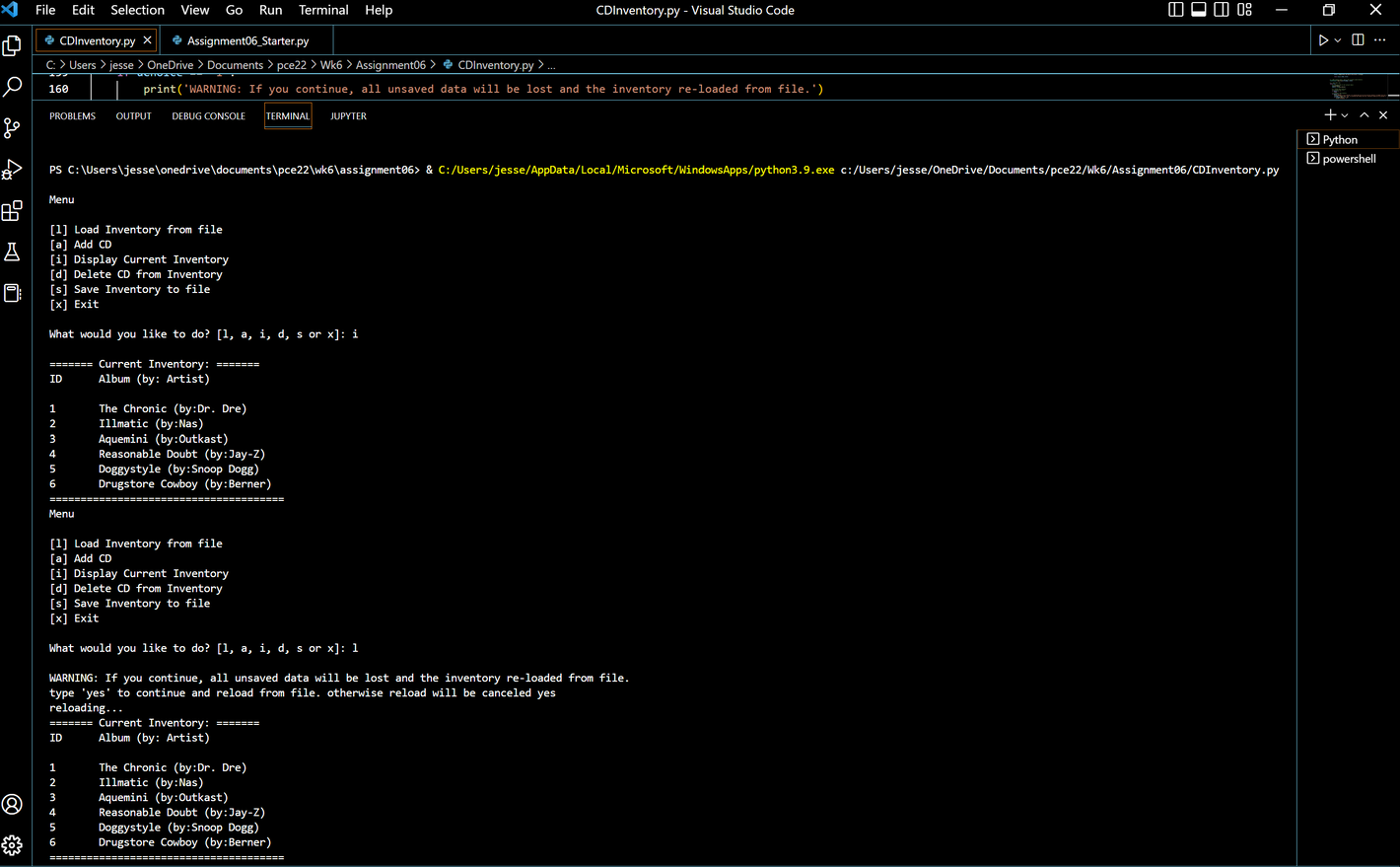
I’ve noticed that the terminal or the IDE window doesn’t always start where the python file is located and while you can run the file, if you have a data file you’re trying to access in another directory you’ll get an error message for File Not Found, so I added an exception handler for this scenario to the step where a user might select to load the data from an existing file.

Lastly, to account for any keystroke errors when deleting a row, I added an exception handler to account for a Value Error where the user inputs the ID of the row they would like to delete.

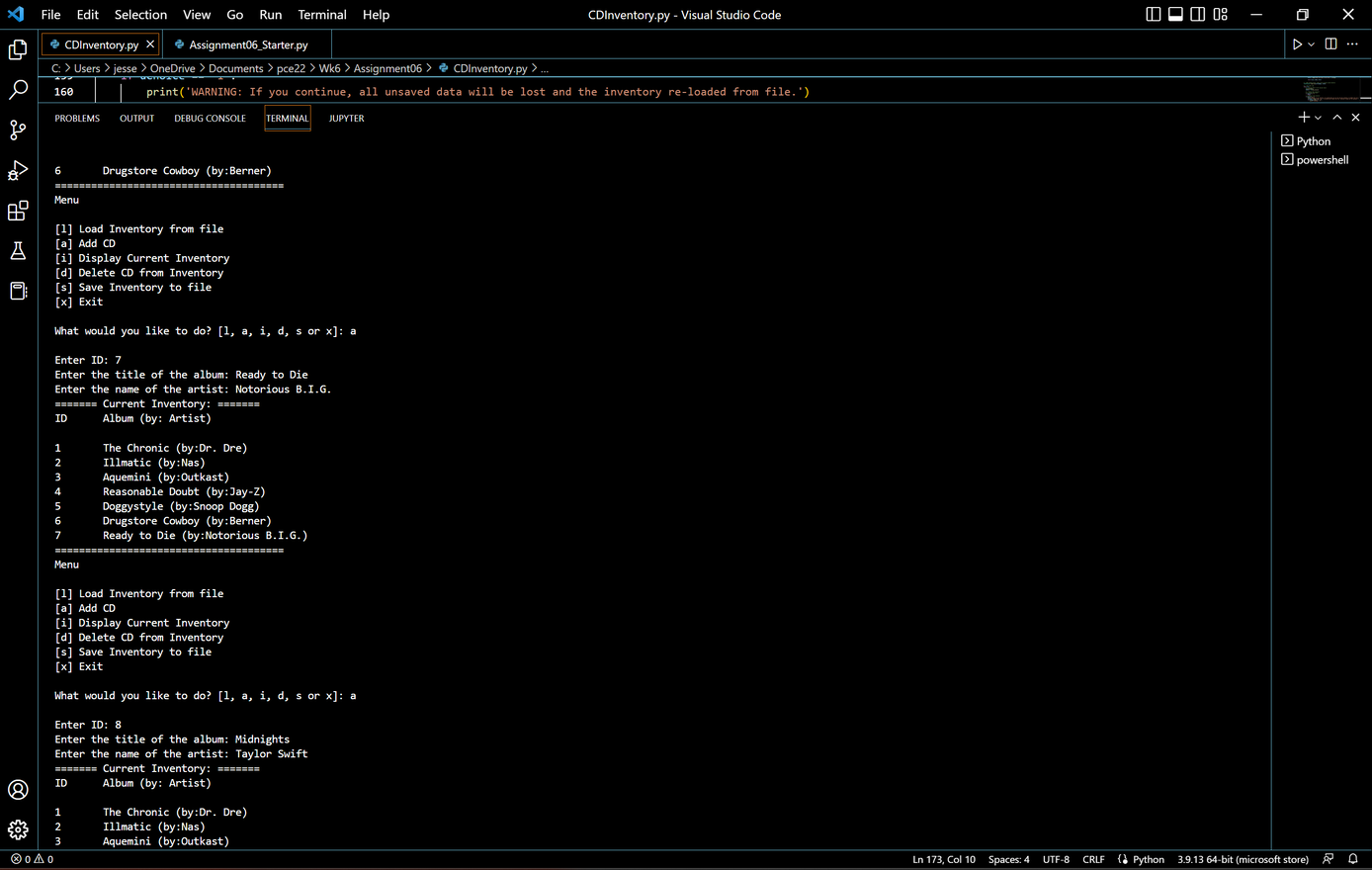
I wasn’t sure if we were supposed to replace any existing code that accounts for potential errors like the else statement or in the menu\_choice function where the code uses a while loop to evaluate the user’s choice against the list of acceptable choices. The existing options seemed more efficient so I left them as-is.

### Testing in the VS Code Terminal

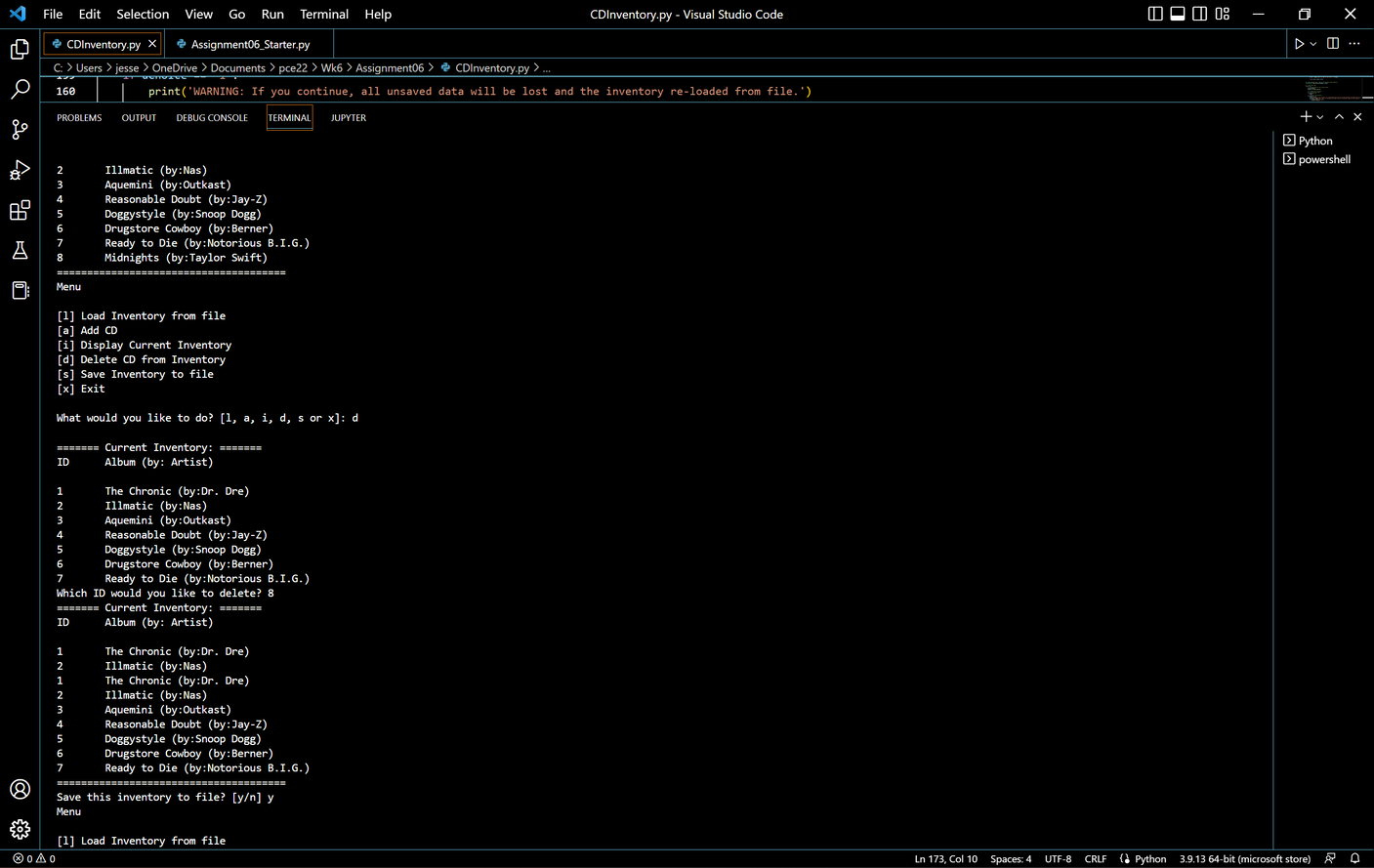
#### *Loading the program checking inventory and loading inventory from the file*



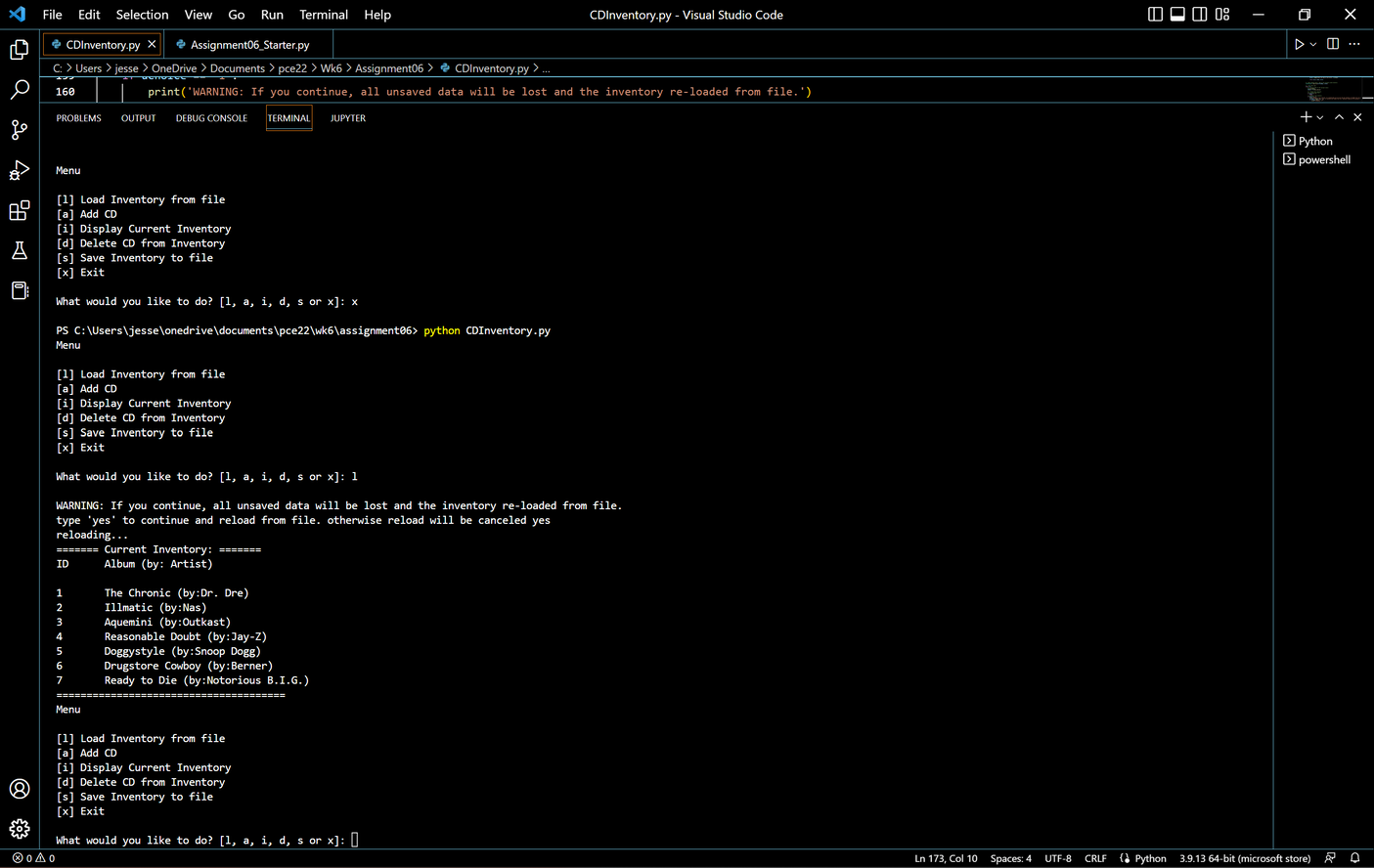
#### *Adding a couple of new albums*



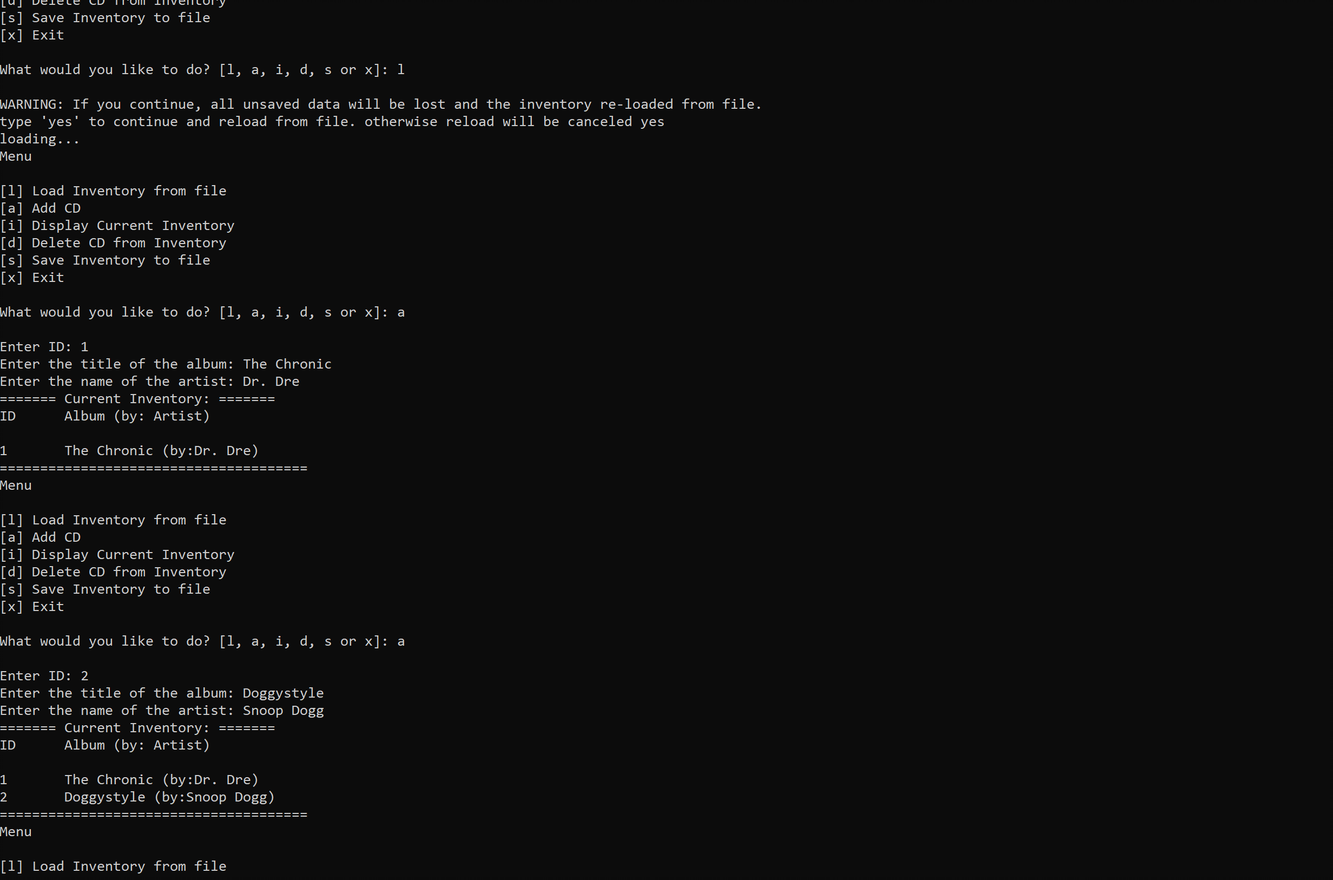
*Whoops, wrong list...delete ID 8 from the list, then save the file*

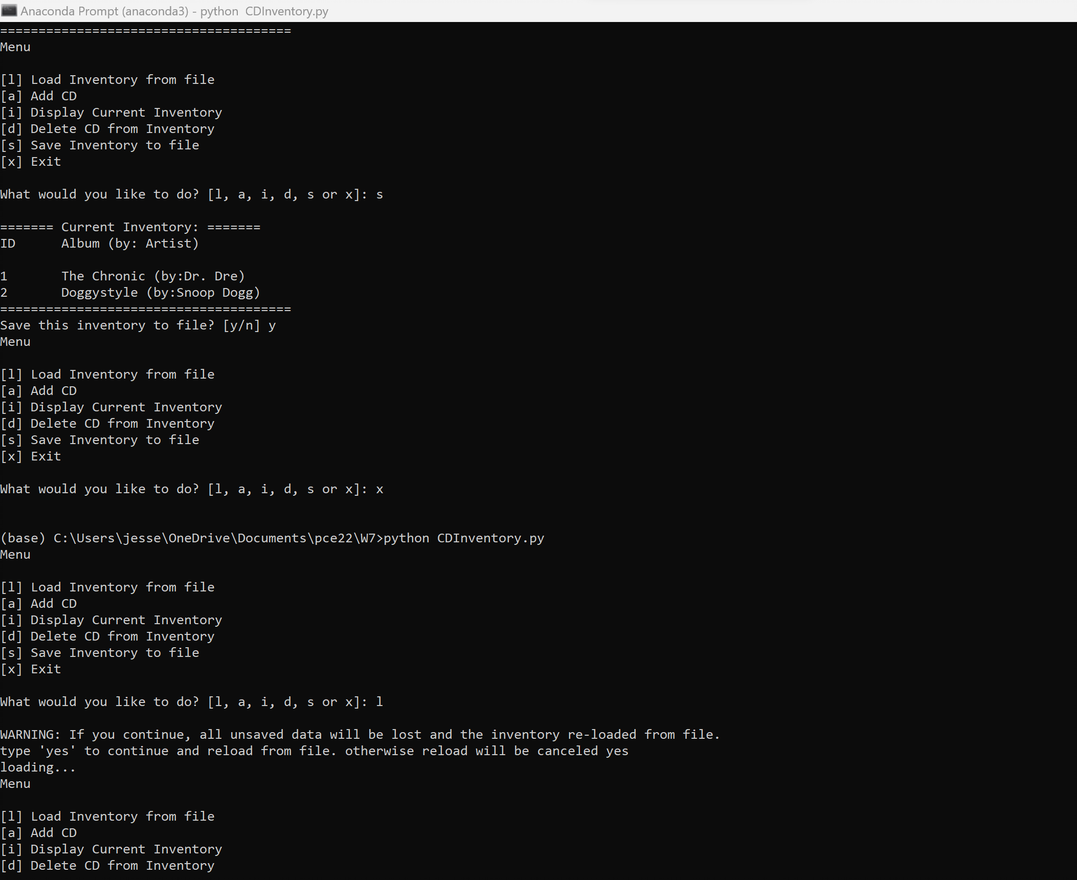


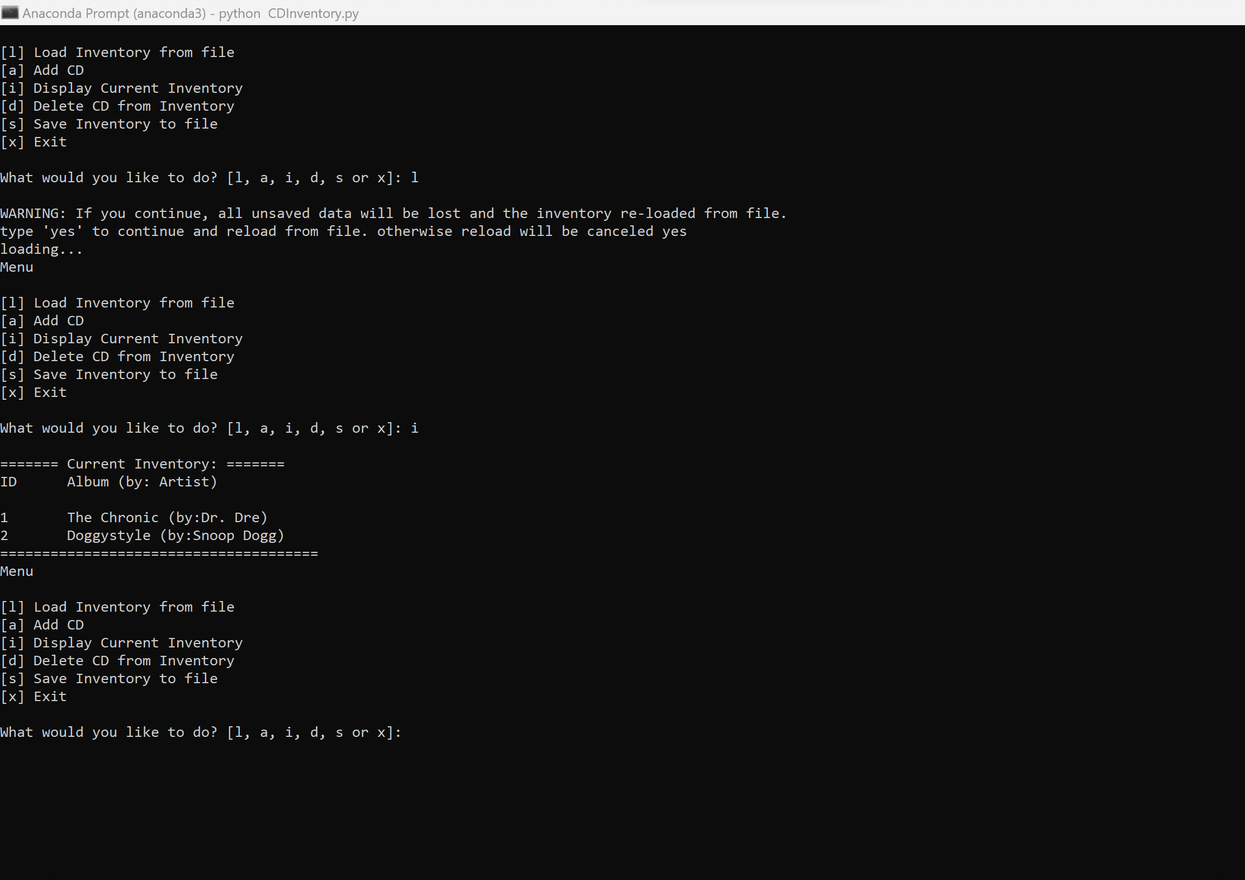
#### *Exit the program then load again and see if the latest changes were saved to the file*



Running in a Terminal Window







## Summary

I liked this week’s exercise a lot better than last week. It was nice to have a bit of time to fix my code from last week. I don’t exactly know why but taking the code elements out of the while loop and embedding them into classes and functions made more sense. Although, relating them back to the arguments took me a little while to understand and fix while debugging the errors in my code. This also made the actual running loop seem a lot less intimidating than last week.