Kimberly Brock

March 7 , 2021

Foundations of Programming (Python) – IT FDN 110 A

Assignment08

GitHub: [kb1981/Assignment\_08 (github.com)](https://github.com/kb1981/Assignment_08)

Introduction

Module 08 is all about Object Oriented Programming (OOP). We discussed creating classes to define the objects within the program. We went through creating attributes and method for individual objects and how to restrict access to those attributes. As always, we went through the class modules, read a chapter in the book, reviewed some additional websites, and put our knowledge to use by creating a script.

Module 08 – OOP, yeah you know me …

Module 08 was all about Object Oriented Programming (OOP). OOP is taking a class (blueprint) and creating an object that can be customized. Fields in a class can hold class level data. Constructors are used when creating an object to ensure they have all the data needed in the correct types. Attributes are variables, and attribute properties can be used to make sure the right data type is assigned to these. Methods allow you to tell the user what is in the object in a clear way. I usually write more on this section, but I feel the information got into my brain and I was able to utilize it for the assignment, and now I may need some more time to regurgitate it into words.

*Module 08 can be found at the link:* [*https://saravji.github.io/saravjis\_hut/FDN\_Prog/Modules.html*](https://saravji.github.io/saravjis_hut/FDN_Prog/Modules.html) *and the Lab08\_A, B, C, D, E are shown in Appendix A.*

Python Programming for the Absolute Beginner, Third Edition

This week in the book *Python Programming for the Absolute Beginner, Third Edition*, By Michael Dawson I read through. “Chapter 8–Software Objects: The Critter Caretaker Program.” This chapter was all about, you guessed it, object-oriented programming. We walked through how to define a class and a method. The class acts as the blueprint. Objects are the items created from the Class that can be customized. A constructor is part of the class that automatically is called when a new object is created to set up the initial attributes. Attributes are what help distinguish your objects apart from others, it is their uniqueness. We learned how to call and assign object attributes, and also make them private, so they are only available to the class. Finally we walked through the cute Critter Caretaker program and learned about the methods involved.

Web Article

* <https://realpython.com/python3-object-oriented-programming/> (external site)

The above site described Object Oriented Programming (OOP). The site walked through how to define a class and gave the differences of classes versus instances. The site showed how to instantiate (create new) an object in Python and assign it attributes. In addition to attributes, methods can also be inside a class as well to return useful information to the user. The last item discussed in this site was inheriting from other classes and examples of a parent/child class.

Additional Video

• Git and GitHub <https://youtu.be/IHaTbJPdB-s> (external site)

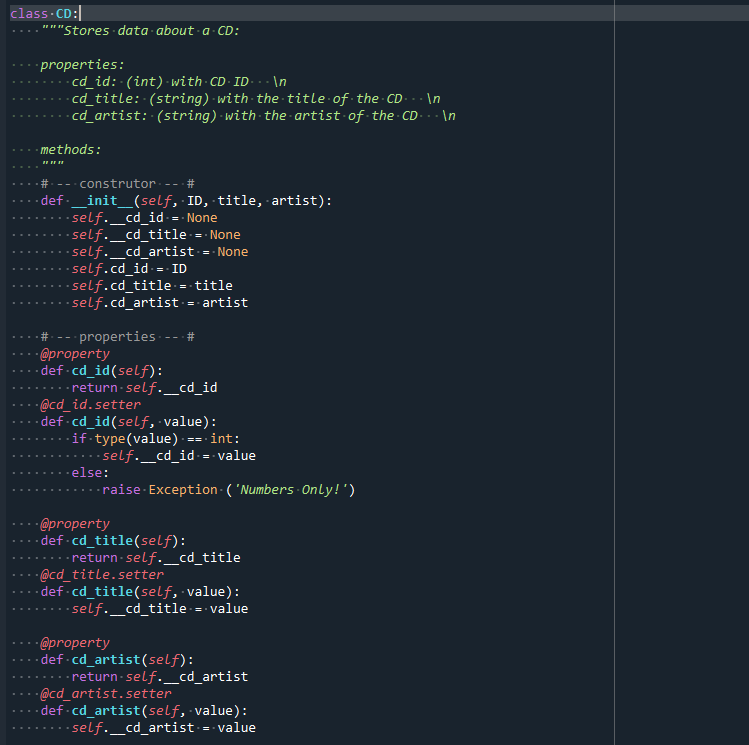
This site explains how Git and GitHub work together. The site explains how you can utilize Git to keep track of revisions to code, and helpful if you need to go back to a previous version. GitHub is great for teams so people working together on sections of code can try out how their section integrates with the main code without messing it up.

Applying Knowledge – CD\_Inventory.py, EVEN **more** *enhanced*

To begin, I believe this was the second assignment in this class that almost had my computer flying out the window.

**# TODO Add Code to the CD class**

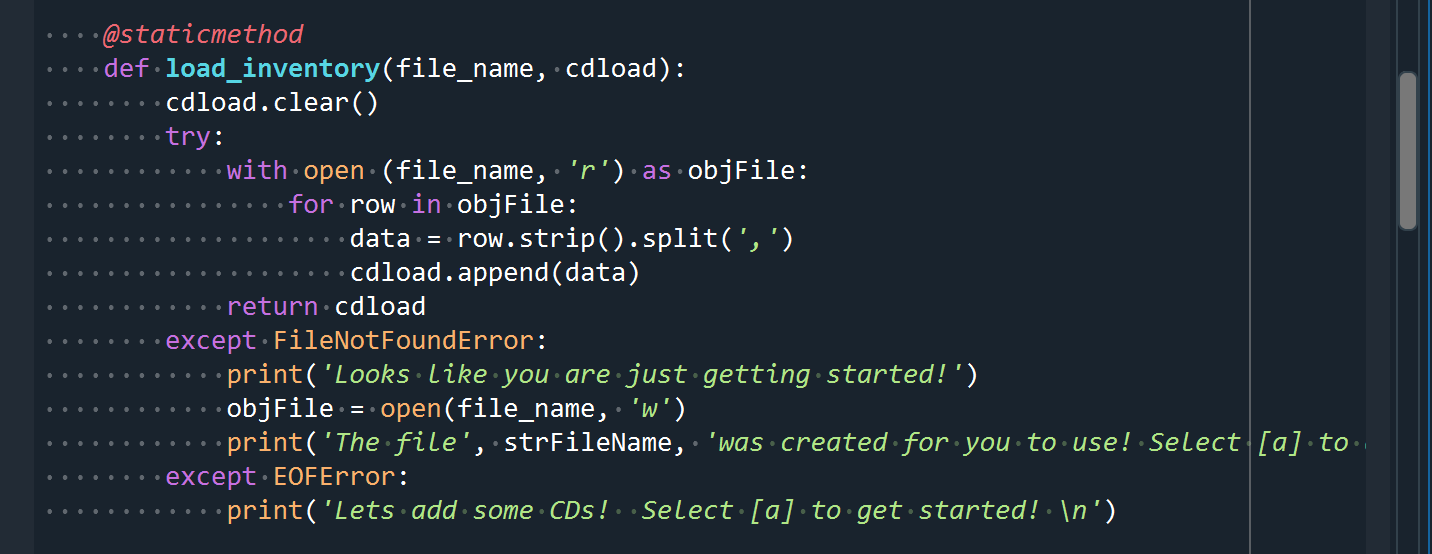
For this section, we were setting up our CD Class, our blueprint for all CDs. I included an ID, Title, and Artist as attributes. For the ID, I included some data handling to ensure it was an integer.



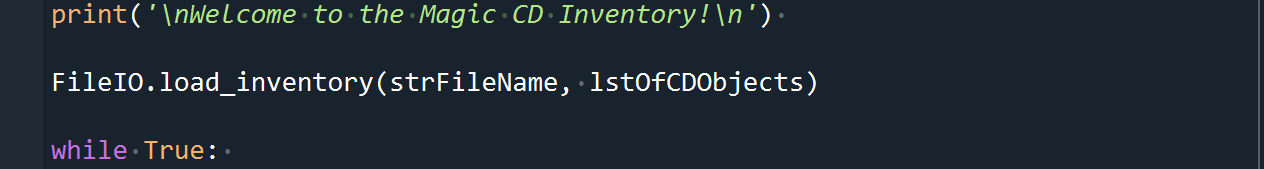
Listing . TODO Add Code to the CD class

**# TODO Add code to process data from a file**

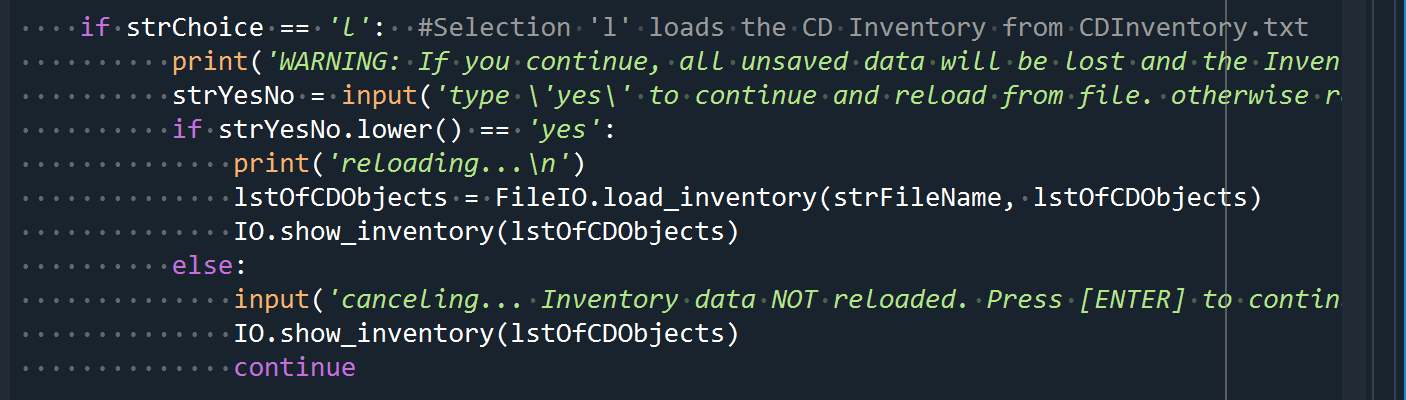
For this section, we added code to load inventory from a file. Included here is error handling if the file is not created yet and if there is nothing in the file. The code calls the file in the main body to initially load anything as a starting point. In the main loop, the user can load the file with the [l] option.



Listing . TODO Add code to process data from a file (load\_inventory)



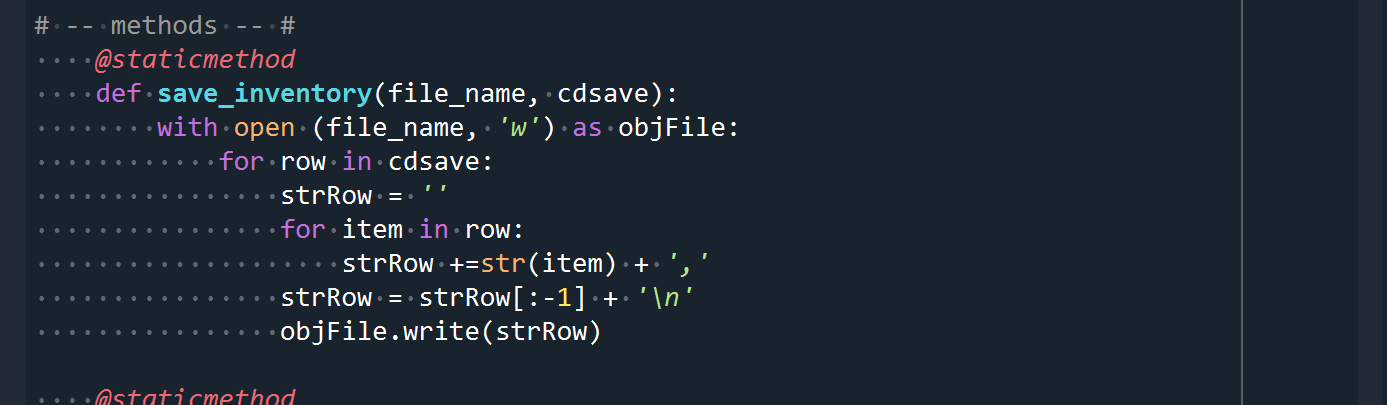
Listing . TODO Add code to process data from a file (Initial load of inventory)



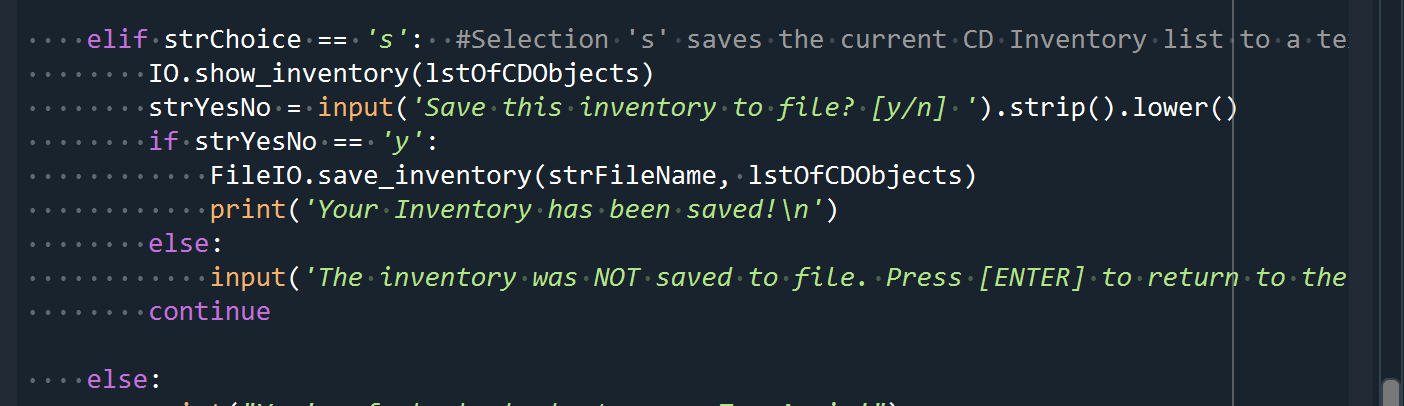
Listing . TODO Add code to process data from a file (option in main loop)

**# TODO Add code to process data to a file**

For this section we added code to save the inventory that is added from the user.

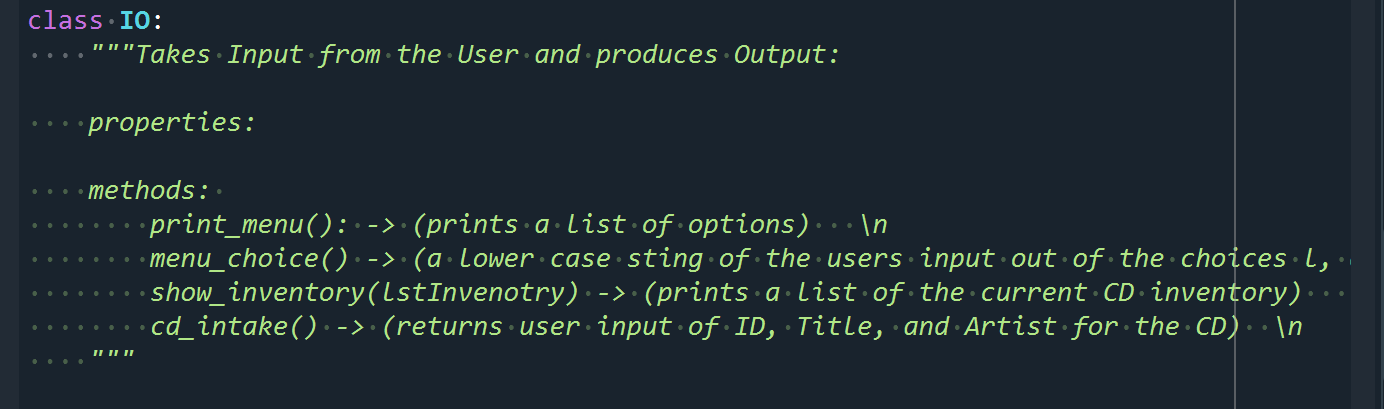


Listing . TODO Add code to process data to a file (save\_inventory)



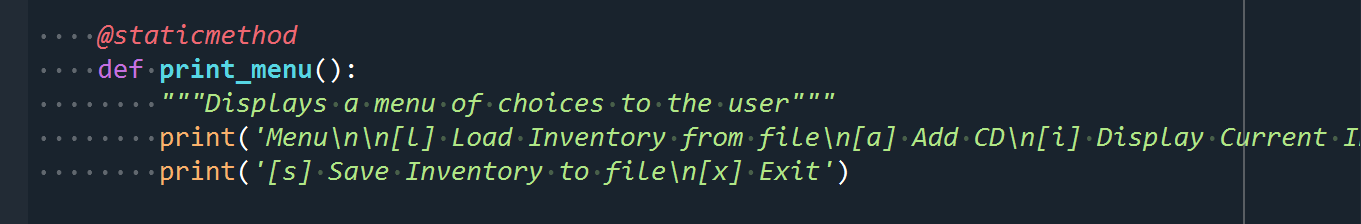
Listing . TODO Add code to process data to a file (option in main loop)

**# TODO add docstring**



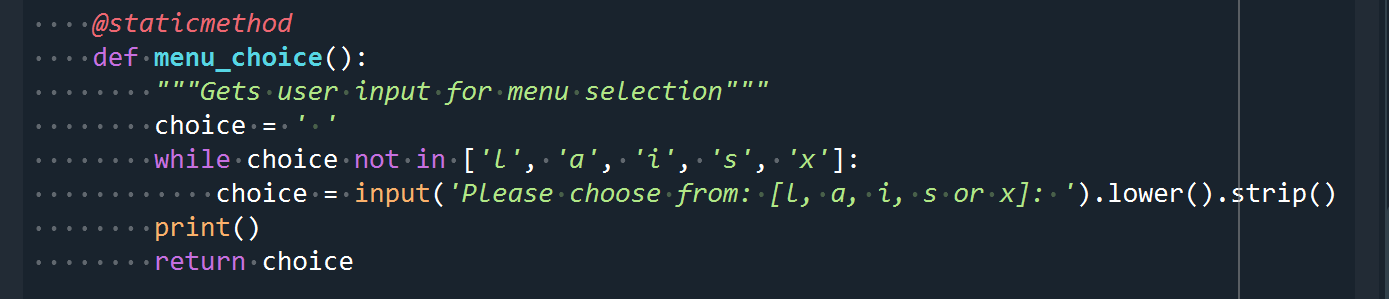
Listing . TODO add docstring to IO class

**# TODO add code to show menu to user**



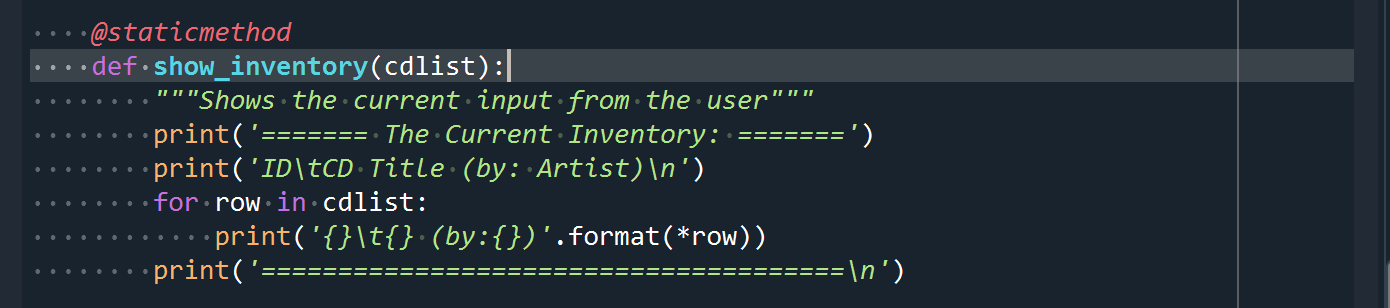
Listing . TODO add code to show menu to user in IO class

**# TODO add code to captures user's choice**



Listing . TODO add code to captures user's choice in IO class

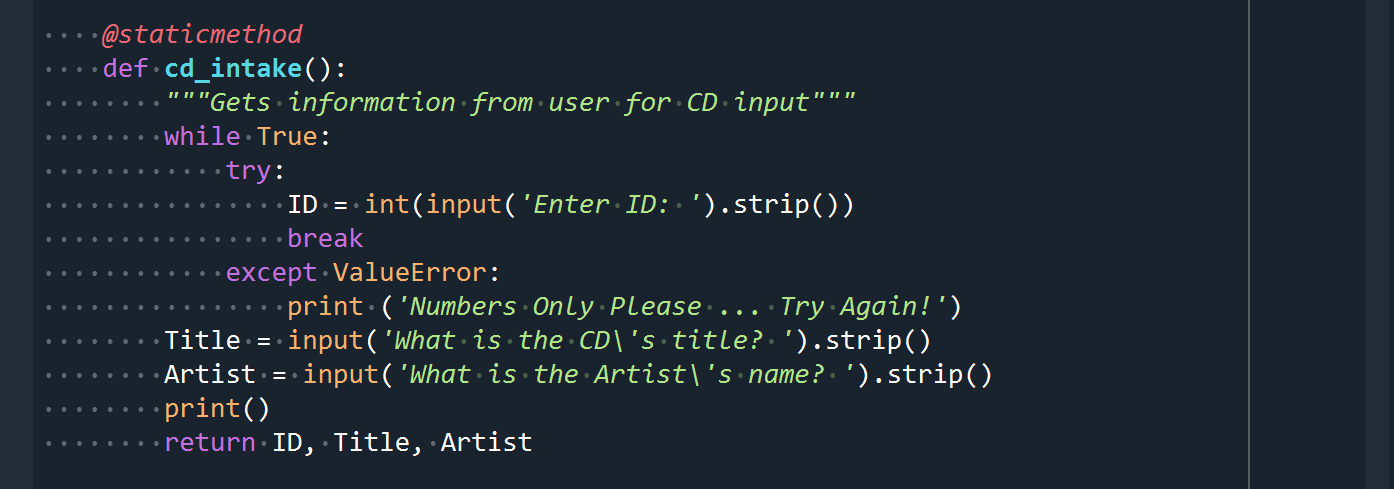
**# TODO add code to display the current data on screen**



Listing . TODO add code to display the current data on screen in IO class

**# TODO add code to get CD data from user**

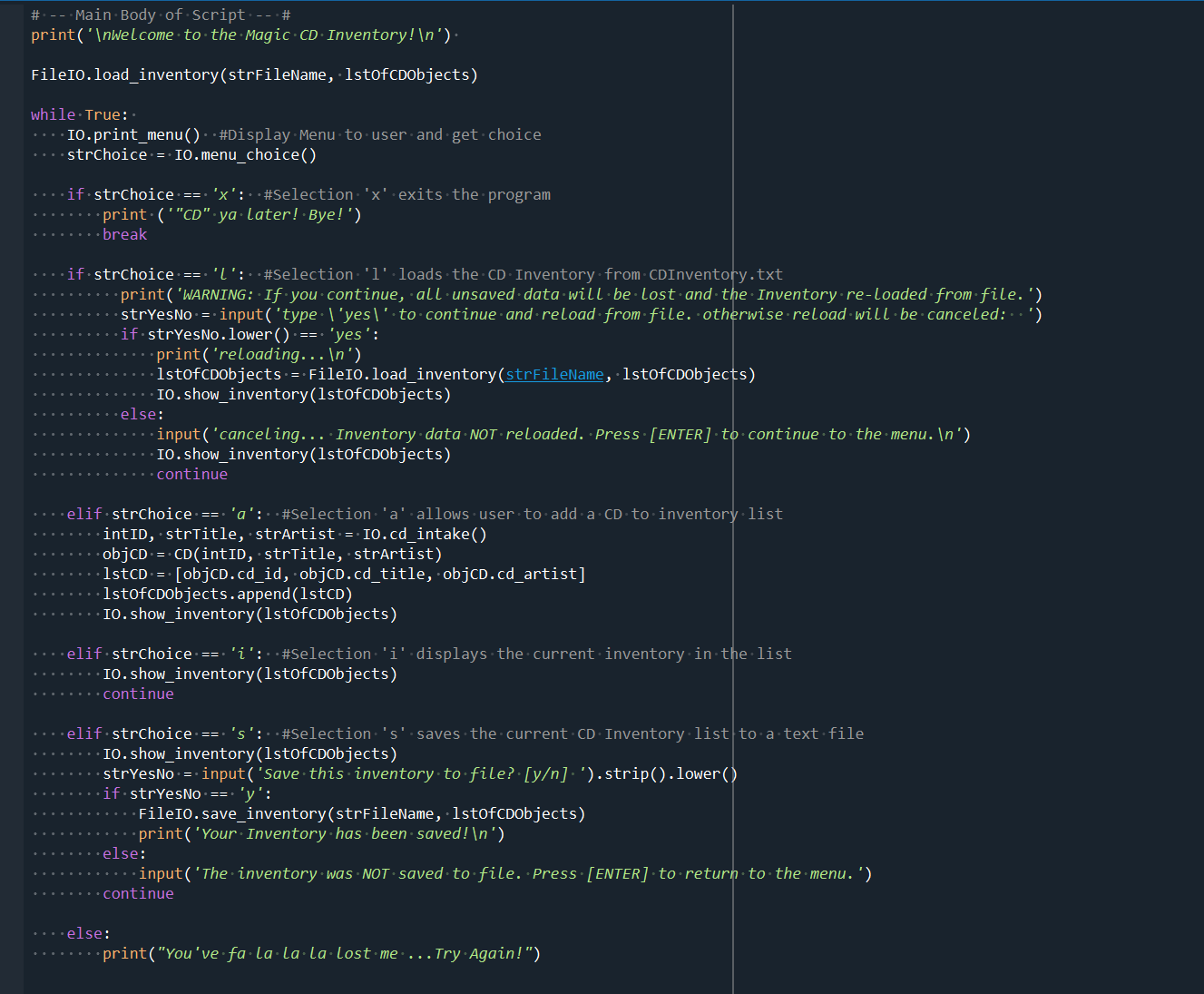
The code included error handling to ensure the user was inputting an integer for the CD ID.



Listing . TODO add code to get CD data from user in IO class

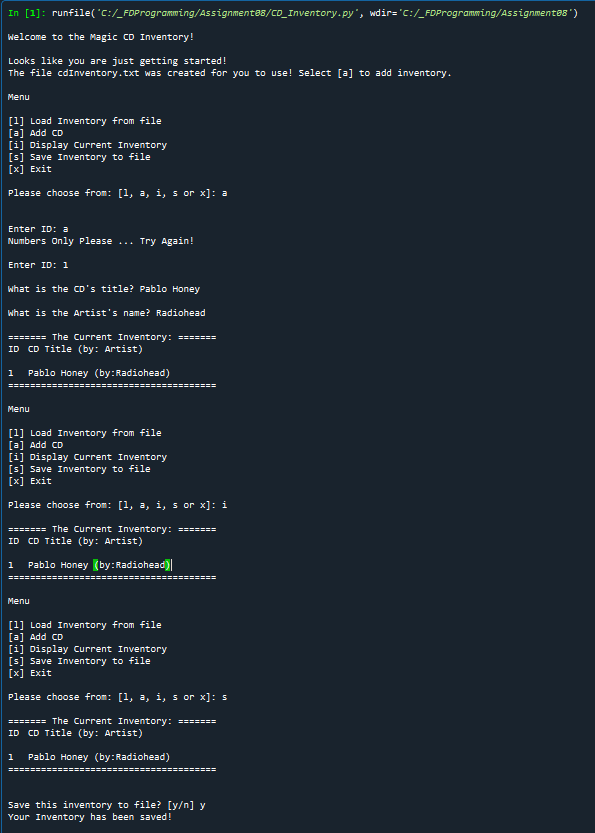
**# TODO Add Code to the main body**

This part included the initial pull / check for inventory that has already been added to the text file. If a file is not there or there is nothing in the file, it is handled by error handling code. The while loop goes through each menu option and has a catch all else at the end.



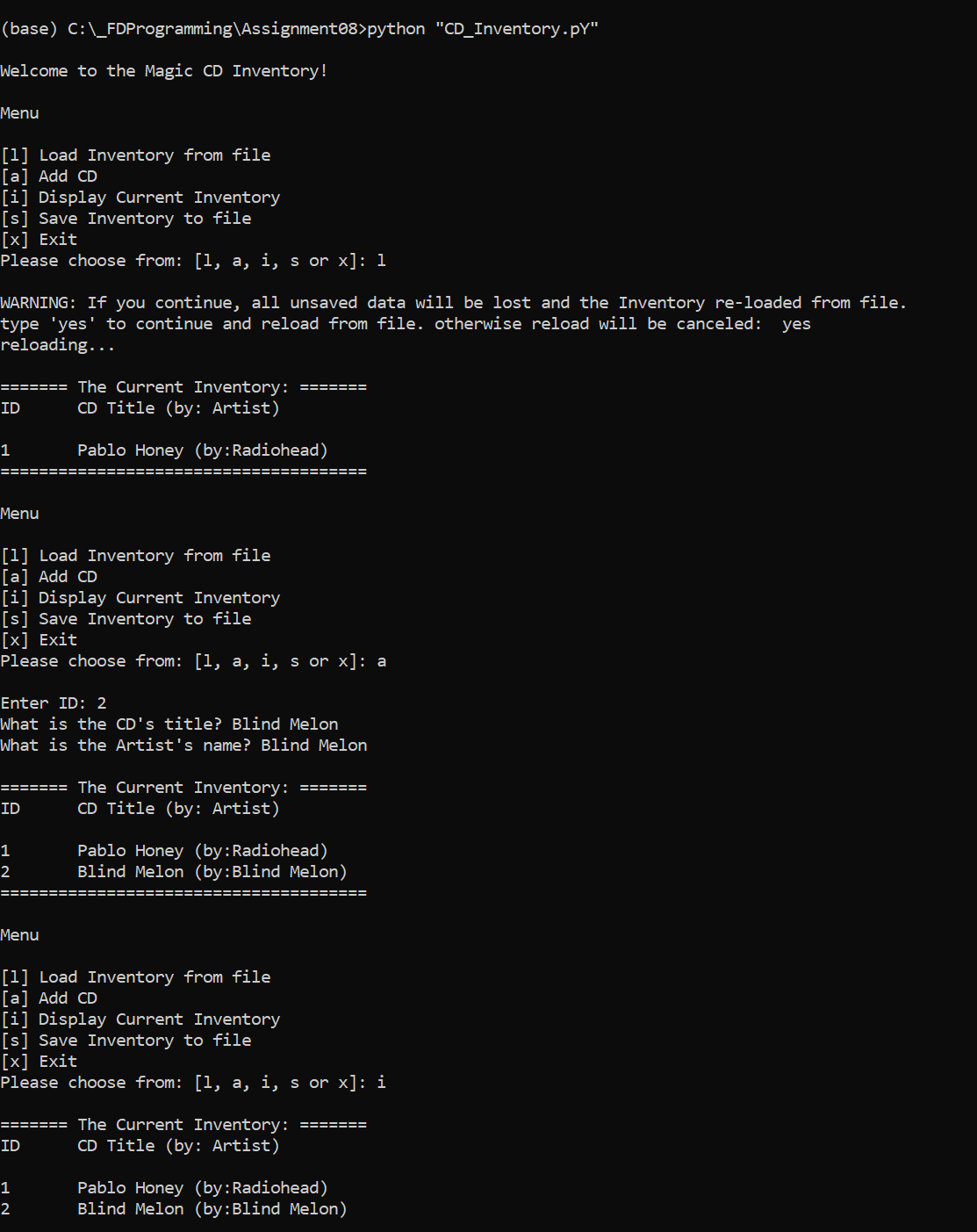
Listing . TODO Add Code to the main body

**Running in Spyder**



Listing . CD\_Inventory running in Spyder

**Running in Terminal**



Listing . CD\_Inventory running in Terminal

**cdInventory.txt file**

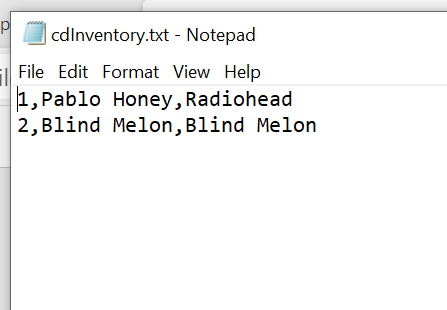


Figure . cdInventory.txt file

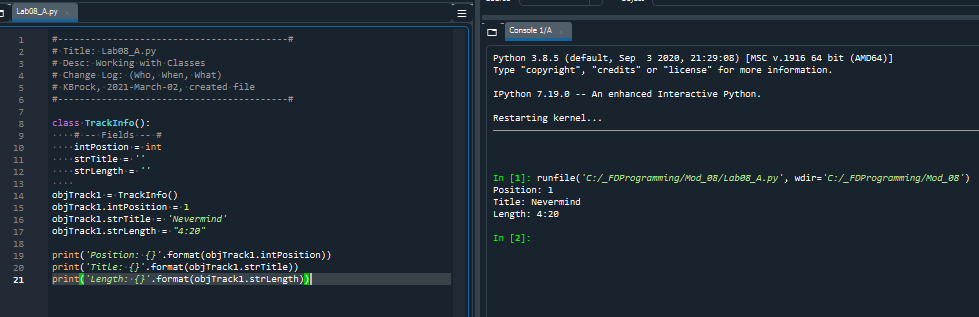
Summary

This week was challenging for me, but I feel like once it kicked in, I got it (at least for this week, I think). We went over OOP and how to use them. We went through the class modules and did some lab associated with classes. We read the class book that walked through working with OOP and gave a cute Critter example. We went to a couple of websites to further learn about OOP and about Git and GitHub. Finally, we did the Assignment08, and it appears as though I have survived, so I will call this a win!

Appendix

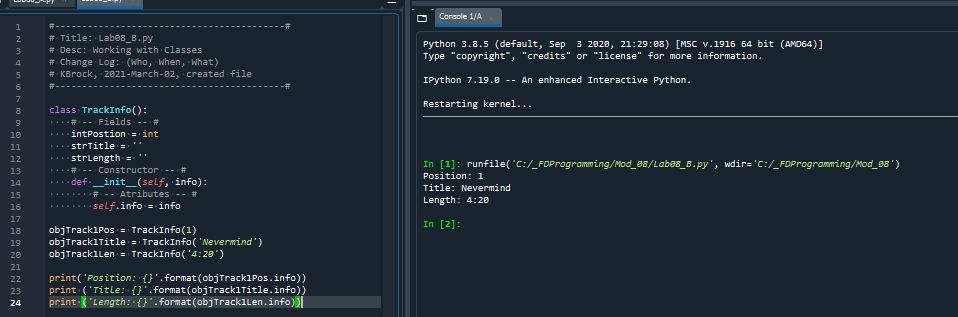
*Appendix A. Module 08 Lab Listings*

Lab08\_A.py works by creating a Class called TrackInfo and including the fields intPosition, strTitle, and strLength. In the main program, I created an object called ObjTrack1 and assigned values to the fields, then printed out the values



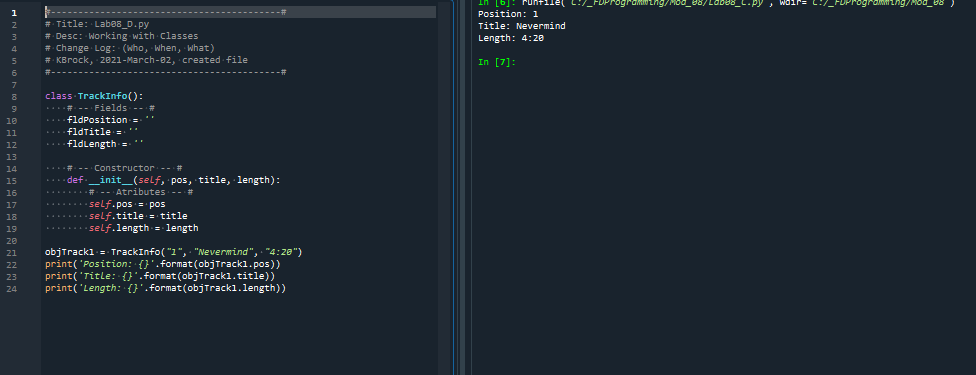
Appendix A Listing 1. Lab08\_A.py

Lab08\_B.py works by using a constructor to take in the information for each created object and store it.



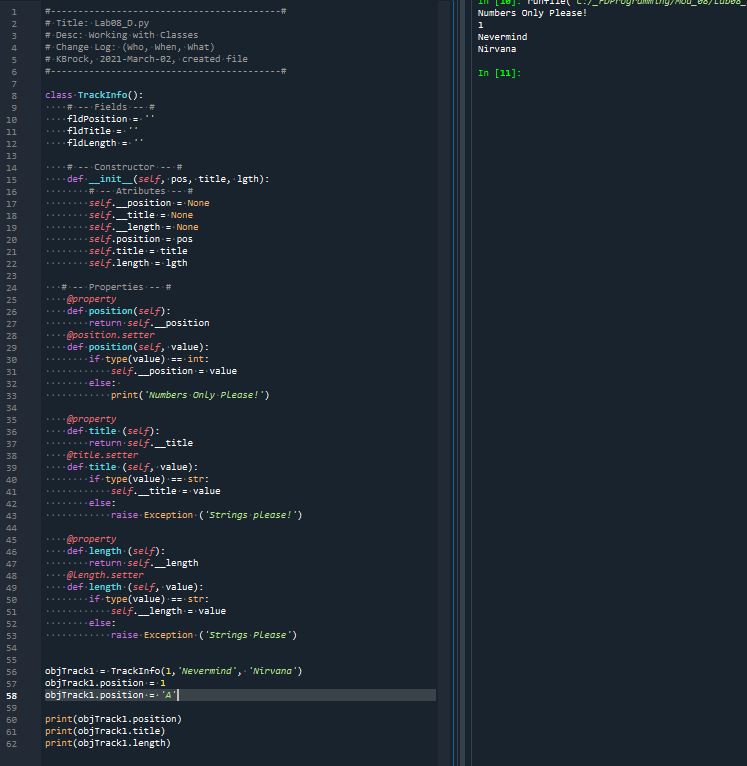
Appendix A Listing . Lab08\_B.py

Lab08\_C.py works a lot like Lab08\_B.py, except I realized I needed to add some more attributes.



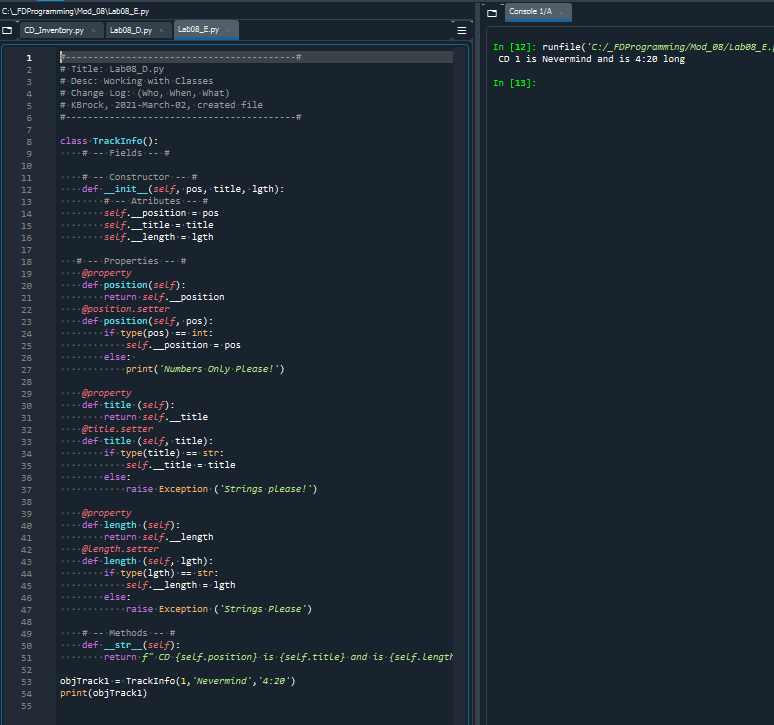
Appendix A Listing . Lab08\_C.py

Lab08\_D.py was working on setting properties for the attributes to control what type of data goes into them. We noticed that the initial creation of the object would not catch the exception, but if you tried to assign the attribute directly, it would catch the exception



Appendix A Listing . Lab08\_D.py

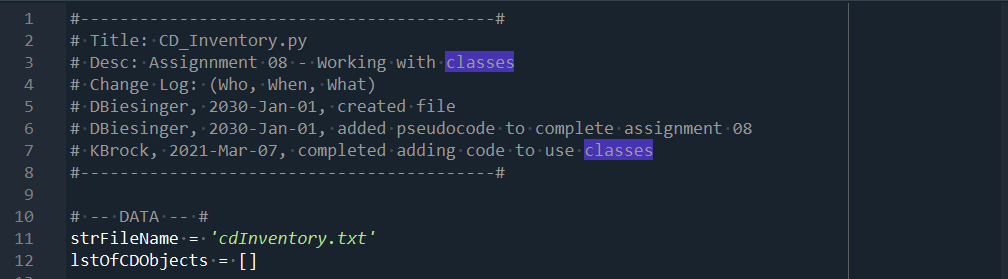
Lab08\_E.py was working with the \_\_str\_\_ method. This was nice because it allowed us to get the information from the object in a clean, understandable format.



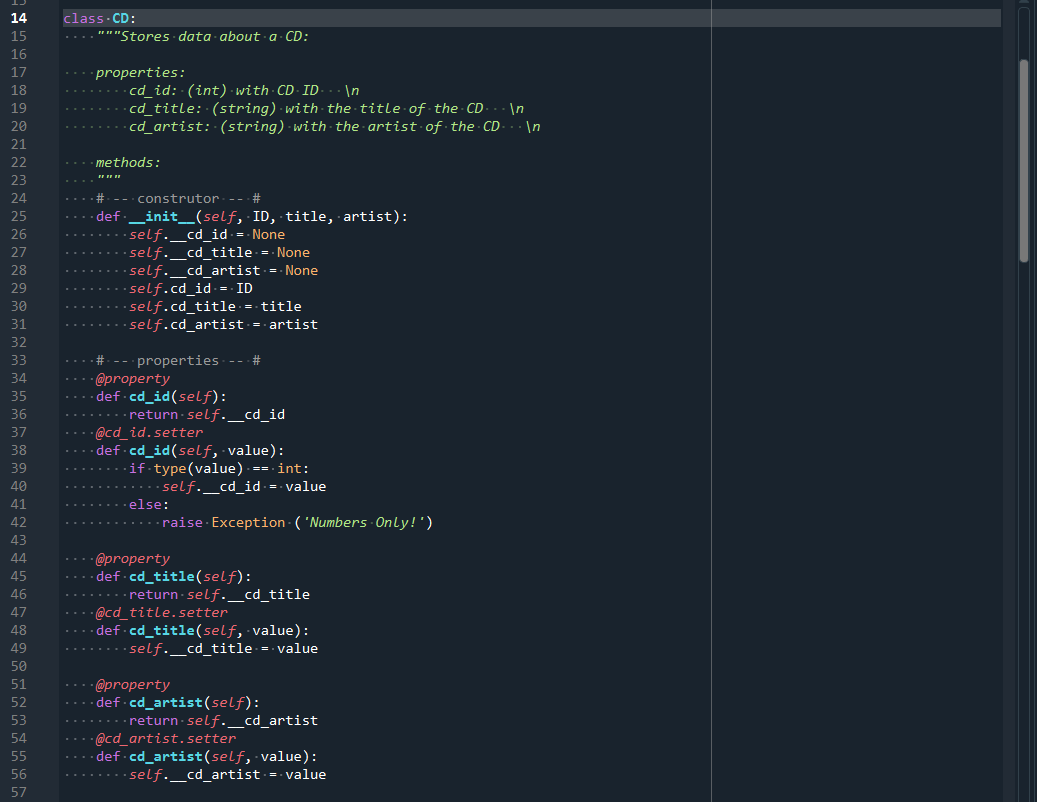
Appendix A Listing . Lab08\_E.py

*Appendix B. Assignment08 – CD\_Inventory.py*

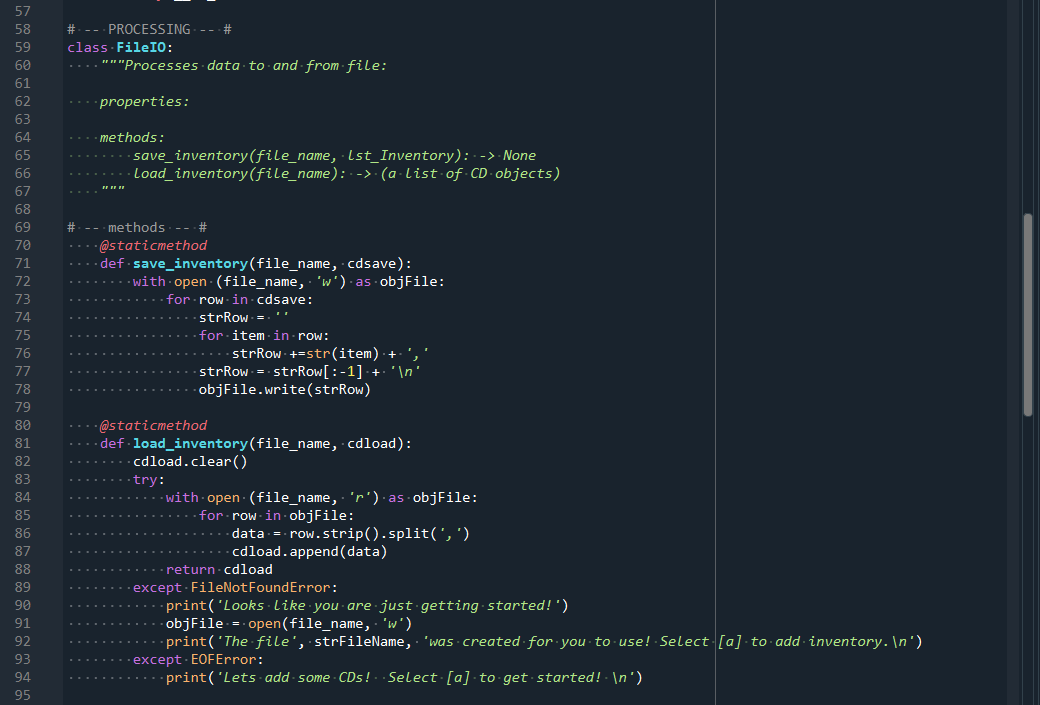
Assignment uploaded to GitHub: [kb1981/Assignment\_08 (github.com)](https://github.com/kb1981/Assignment_08)



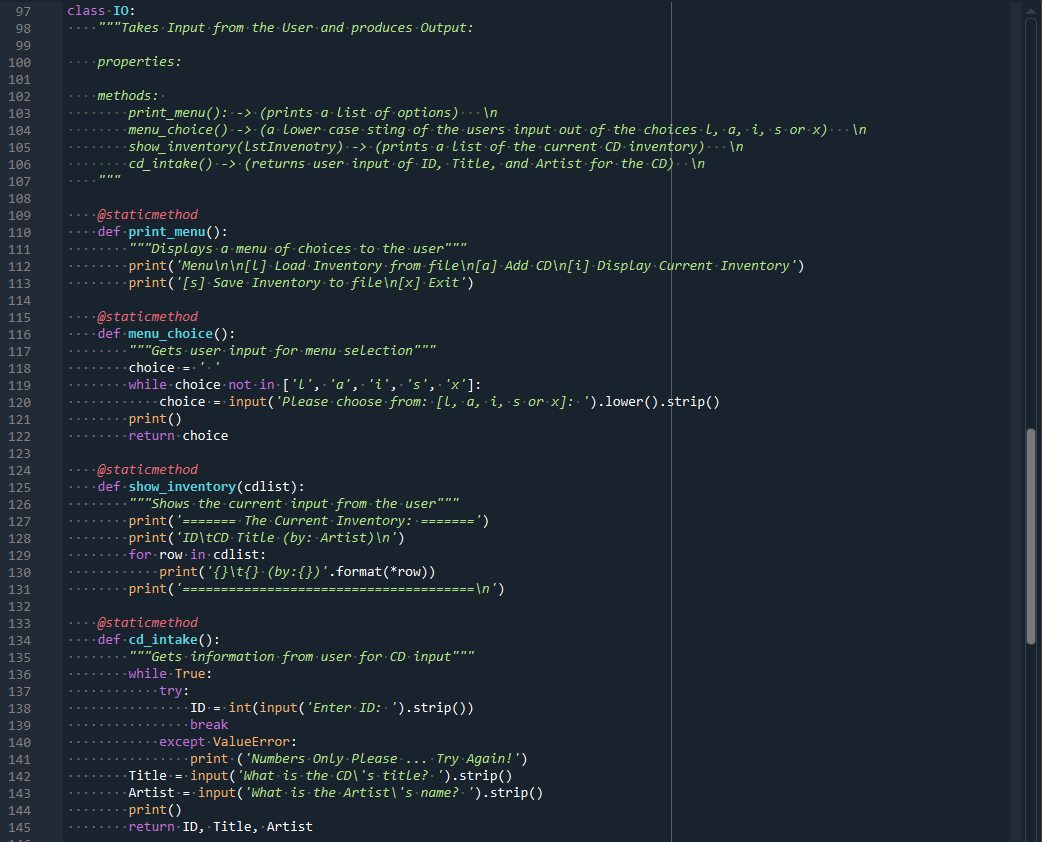
Appendix B Listing . Header and Program Level Data



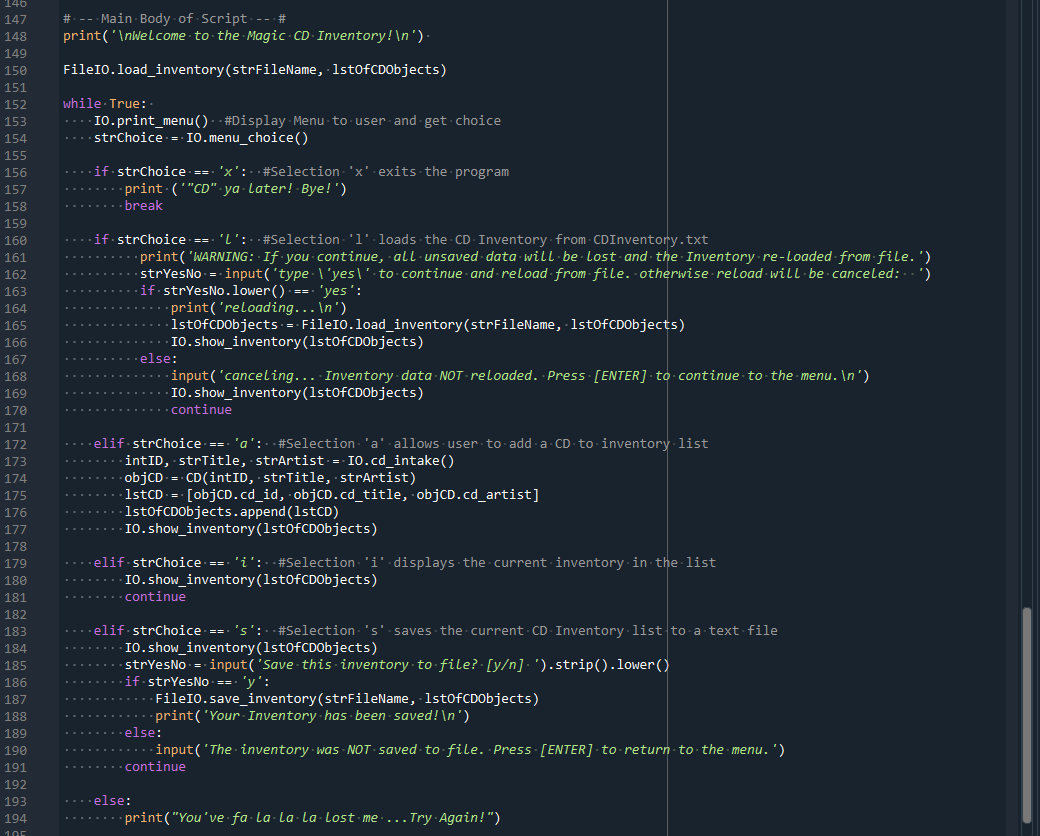
Appendix B Listing . Class CD



Appendix B Listing . Class FileIO



Appendix B Listing . Class IO



Appendix B Listing . Main Body