Rudolph Carvajal

August 22, 2020

Foundations of Programming: Python

Assignment\_06

CD Inventory: Functions and Classes

# Introduction

The focus was on functions in this week’s module. This is an area I am going to have to work on. Although I was able to complete this week’s assignment, functions are a difficult concept to grasp. By doing the reading, it is evident how useful they are -- it was really cool to see how the author put together the tic tac toe game with only 19 lines of code in the main body by leveraging functions. In this document, I will discuss what I learned about functions including arguments and parameters. I also discuss global and local variables and finish by include snapshots of my program running.

# Functions

The biggest advantage of using functions is that they allow you to break up your code into management parts, which make a program easier to read and work with. This reminded me of working in spreadsheets. Writing long code with no logical breaks is a lot like writing one long formula in a cell that controls a financial model. When this occurs, it takes the reviewer an unnecessary long time to understand what the model is doing.

You start a function with *def,* followed by your function name. It is then best practice to use a docstring, which is typically a triple-quoted string and must be the first line of the function (if/when one is used). In Spyder, you can scroll over your function and the information provided in the docstring will be shown which can be really helpful when reviewing code.

# Parameters and Arguments

Parameters are the variable names inside the parentheses of a function that allow the function to receive a value through its arguments. When you include a parameter in a function but then try to call the function without using an argument then you will get an error. By using parameters you are making your code more flexible. In this week’s assignment I was able to complete the program without using parameters in the function but it made the code confined to specific variables used in the code -- this would have made it frustrating when trying to reuse the code. Thus, I went back and added parameters where I thought necessary.

# Global and local variables

Figure 6.7 in the textbook did a great job explaining scopes and showing that variables created within a function can only be used within that function. But if you create global variables then they can be used within your function; however, you can’t change it within the function, only read and use it unless you use the keyword global. Global variables should be used sparingly but it can be a good idea to use a global constant especially if you’re working on programs where key values should stay the same throughout -- the textbook used a tax rate for an example but I can also think of a discount rate if you creating a program that performs a discounted cash flow analysis.

# The CD Inventory Program: with functions added

I began by reading through and running the starter code to get a feel for the program. I then made a note of all the incomplete steps needed to be done which I noticed were called out by the “!” symbol in Spyder (very helpful!). To complete the program I added the following functions:

* Getting a CD from the user: this function was built to be used by the function that added the cd to inventory. It felt a little awkward to write the code this way but I thought doing this would result in fewer lines of code.
* Adding that CD to inventory: this function assigns variables to the output returned by get\_cd() and adds the data to inventory.
* Deleting a CD: this function takes the CD listed by the user and searches the 2D table and if found, deletes it.
* Writing data to a file: this functions writes the current CD inventory to a file indicated.

Below are snapshots of the program running – the first two show different parts of the program running in Spyder and the last one shows the program running in the terminal window.

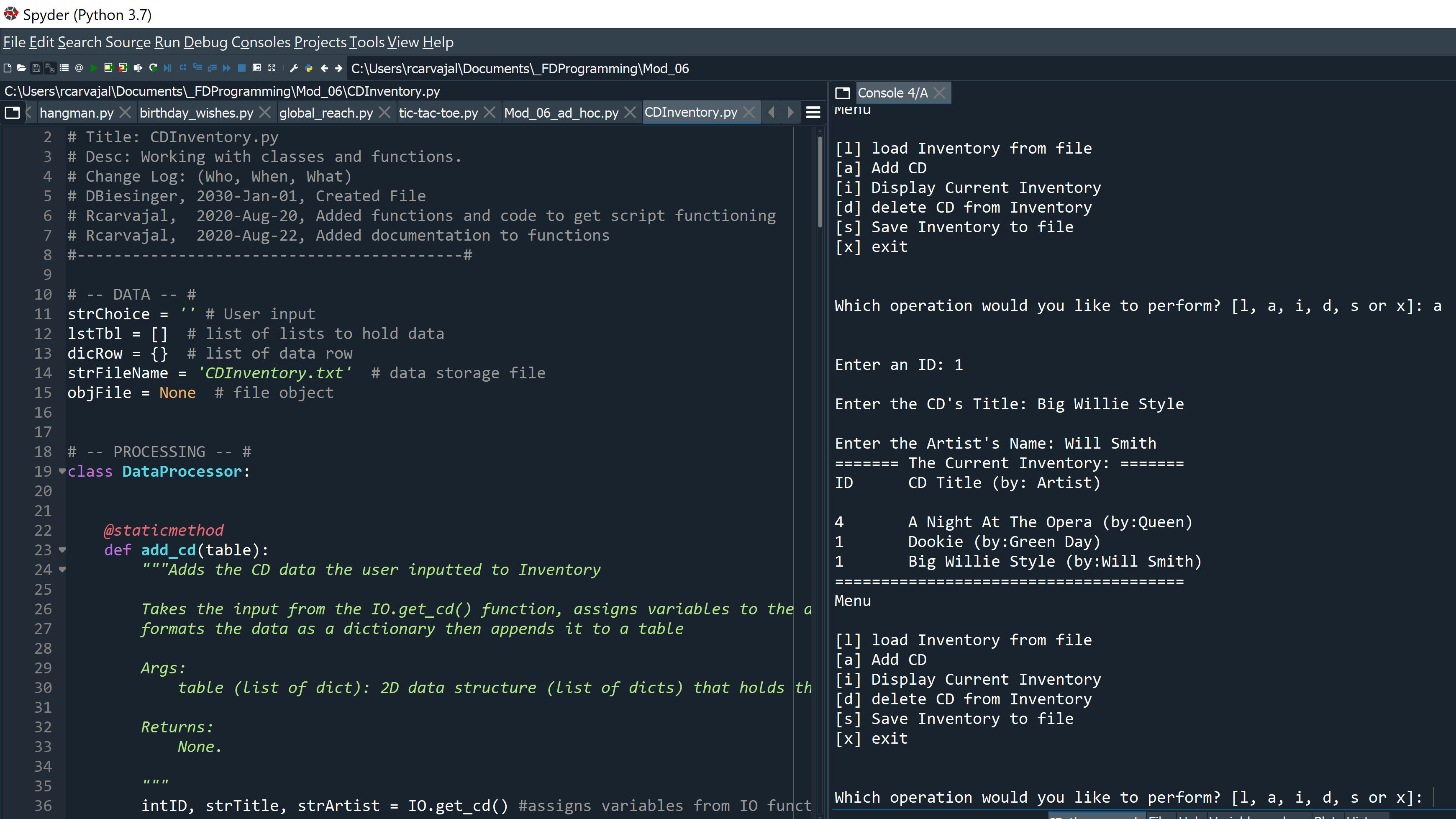


Figure - Adding a CD using add\_cd() function in Spyder

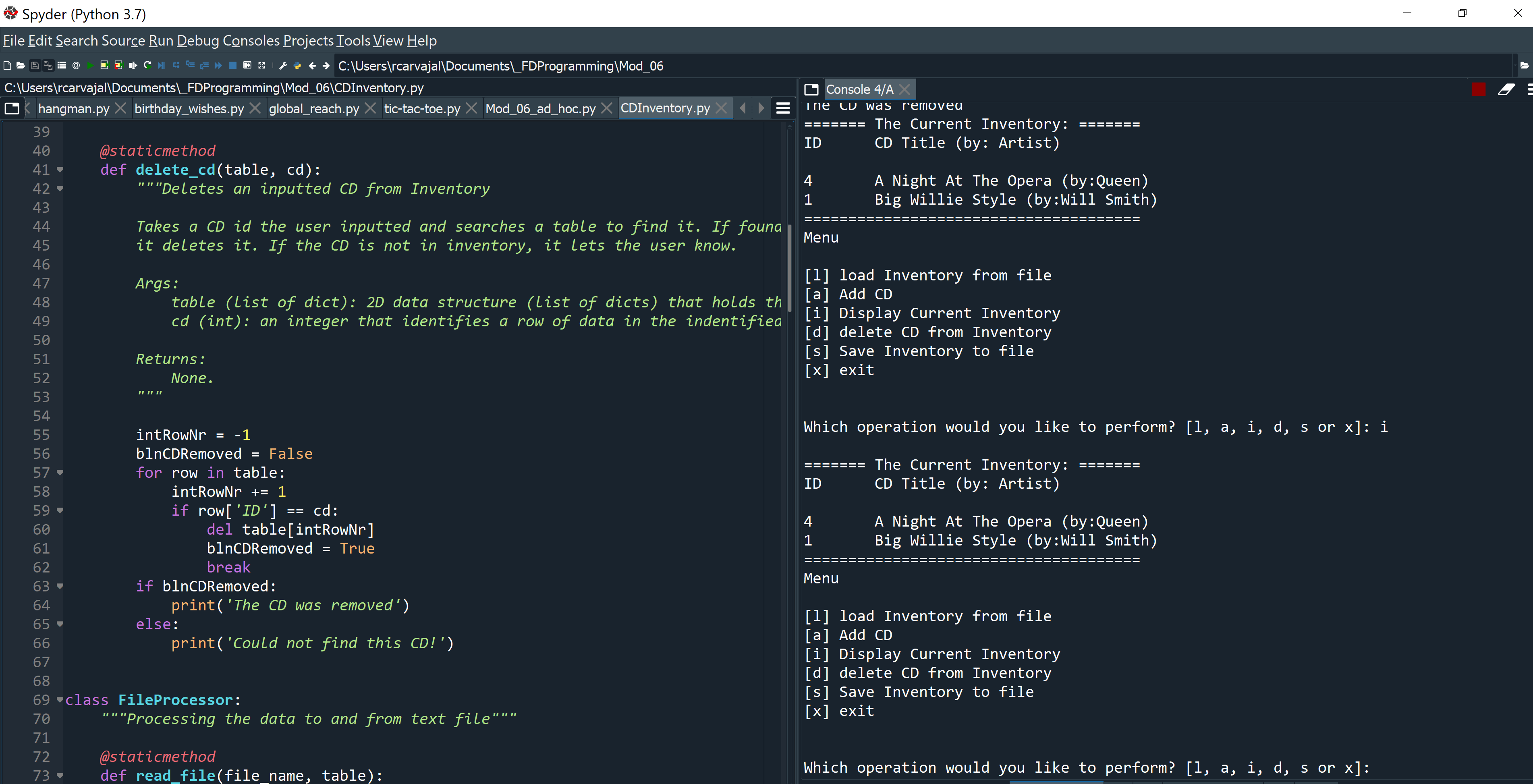


Figure - Deleting a CD from inventory using delete\_cd() function

When running this program. I noticed it allowed me to add two CDs with the same id number. I would consider that a bug that I will fix the next time I work on this program.

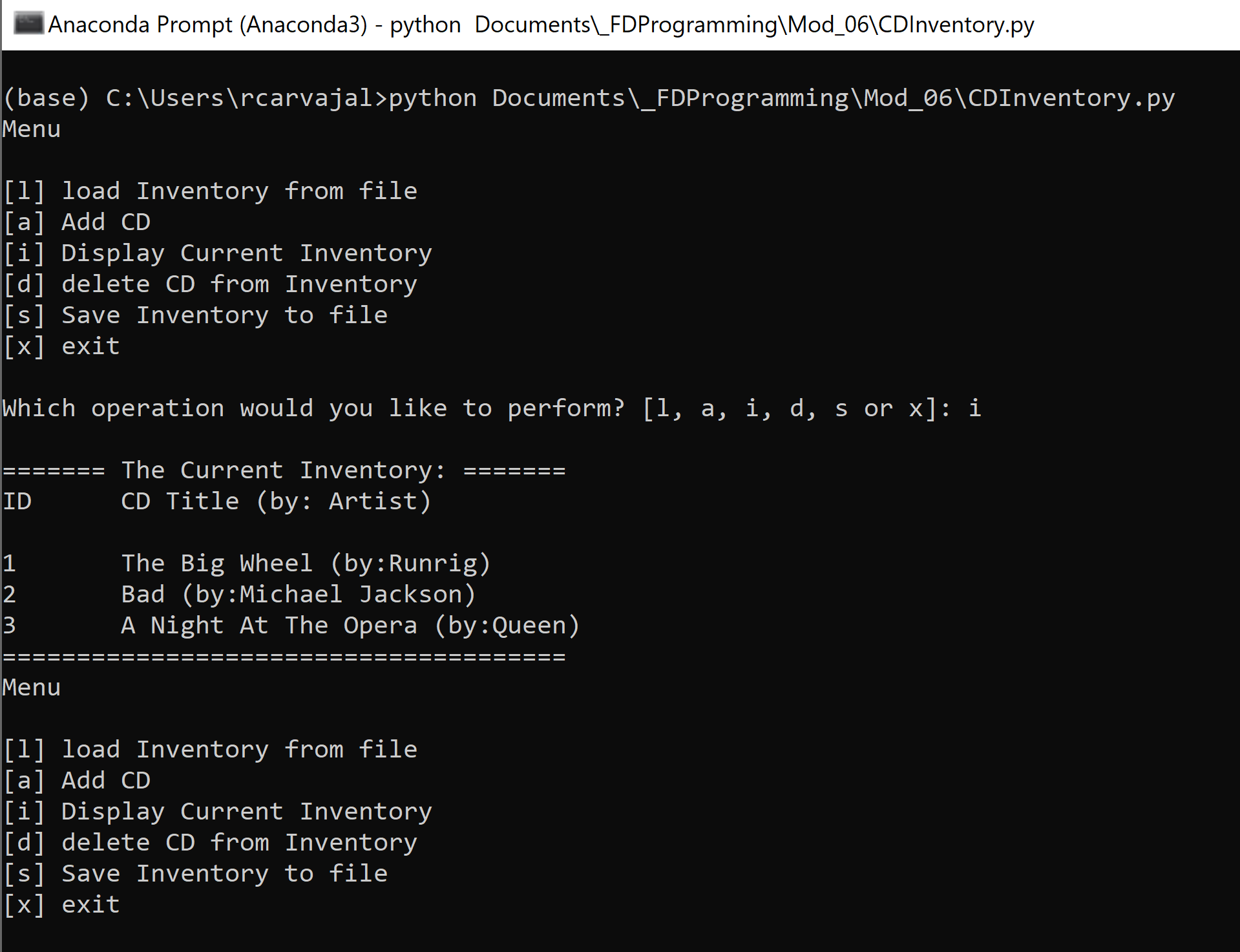


Figure - Running the program in the terminal window

I noticed that the terminal window is picking up a different text file than when I run the program in Spyder. That’s also added to my list of things to look into.

# Summary

This week we learned how to write our own functions. Functions help to keep your code clean and easy to follow. They also can make it easier to reuse aspects of your code and condense the amount of lines in your program. Related to functions we also learned about arguments and parameters and when to use them. I learned that parameters and arguments are slightly different. Parameters are the variables listed inside the parentheses in the function definition where arguments are the values sent to the function when it is called. Furthermore, we learned about global and local variables -- no variable you create in a function, including its parameters, can be directly accessed outside its function. That is because it’s a local variable. The book did a great job explaining this by using a house with tinted windows analogy. Finally, in this module we were introduced to classes. While we didn’t go too in depth with classes so far I can see that they help keep your code more organized.