Maricha Friedman

August 29, 2021

IT FDN 110: Introduction to Programming (Python)

Assignment 8

CD Inventory Script using object class

# Introduction

For this assignment, we had to once again use the CD inventory script, but using a list of objects for storing and loading the CD inventory data. For this assignment, it wasn’t clear if all classes needed to instantiate objects, so this script only uses the CD class to instantiate objects holding individual CD data.

# Creating the CD class

Using the labs from Module 8 as a guide, I added the code to create the CD class, which, when called, will instantiate an object that holds information about a CD with the three attributes: cd\_id, cd\_title, and cd\_artist. I made those attributes private, and then I created a property for each so I could use or call them in the main script or in other functions. I did not actually call them in the main script, and I built in a setter method to ensure the ID number is always an integer, but I could not figure out how to use that within the main script. I also added a counter based on the example from the labs (and the textbook), which I added to the script every time the inventory was displayed. I added a \_\_str\_\_ method to format the object so it would export nicely as comma separated values to the .txt file.

I ended up re-using most of the script from last week’s assignment for the FileIO and IO classes, although I did need to make several adjustments to be able to display the list of objects in the inventory and to read and write from the .txt file.

# Working with a list of CD objects

I was not sure if I needed to also use object-oriented programming in the FileIO and IO classes, and although I tried, I was not really sure how to do that anyway. The examples from the labs were similar enough to the CD inventory context to easily build the CD class based on that, but I wasn’t sure how to do the same with the file functions (what would be the attributes, the file\_name?) or the IO functions to display the menu, process user choice, and add a new CD.

## Add\_cd function

From the previous assignment, I had two separate functions for this, one for the user to input the new CD data (including error handling around the ID) and a separate one to add that input to the list in memory. I kept this structure for this script, putting both in the IO class, and leaving the “inp\_newcd” function as it was, but using that to simplify both the main code and the add\_cd function. In the main code, I simply called the IO.add\_cd function, which now only has 1 argument, “table”, for the inventory list. I called the “inp\_newcd” function as the first line, and then use the 3 values to build my CD object, and append it to the list. This is similar to the way I did it in the previous assignment, using the 3 return values to build the dictionary.

I think there was probably a way to get around this by calling each property of the CD object, but I could not figure that out. I had messier versions of this where I was building the CD object in the main script and using each piece of CD data as arguments in my “add\_cd” function, so my final script I think is a little cleaner and easier to understand.

## Reading and saving the inventory to the file

Saving the inventory was pretty simple, since I built my \_\_str\_\_ method to return a simple line of comma-separated values with a new line character. All I needed to do was loop over the rows in the table (list of objects) and write them to the file. For some reason I got an error that it needed to be string format, so I cast each “row” of the table as string, and that was it:

**def** save\_inventory**(**file\_name**,** table**):**

'''Writes the in-memory inventory to a .txt file

'''

objFile **=** open**(**file\_name**,** 'w'**)**

**for** row **in** table**:**

row **=** str**(**row**)**

objFile**.**write**(**row**)**

objFile**.**close**()**

Loading the inventory required me to revisit some string formatting and manipulation from a few classes ago. While it might not have been the most elegant solution, I just split each line into a list, and then called the CD object and filled it with each item in the list by index. I then appended it to my inventory list:

**def** load\_inventory**(**file\_name**,** table**):**

'''Load inventory from .txt file and add to in-memory list

'''

table**.**clear**()**

objFile **=** open**(**file\_name**,** 'r'**)**

**for** line **in** objFile**:**

data **=** line**.**strip**().**split**(**','**)**

load\_cd **=** CD**(**data**[**0**],** data**[**1**],** data**[**2**])**

table**.**append**(**load\_cd**)**

objFile**.**close**()**

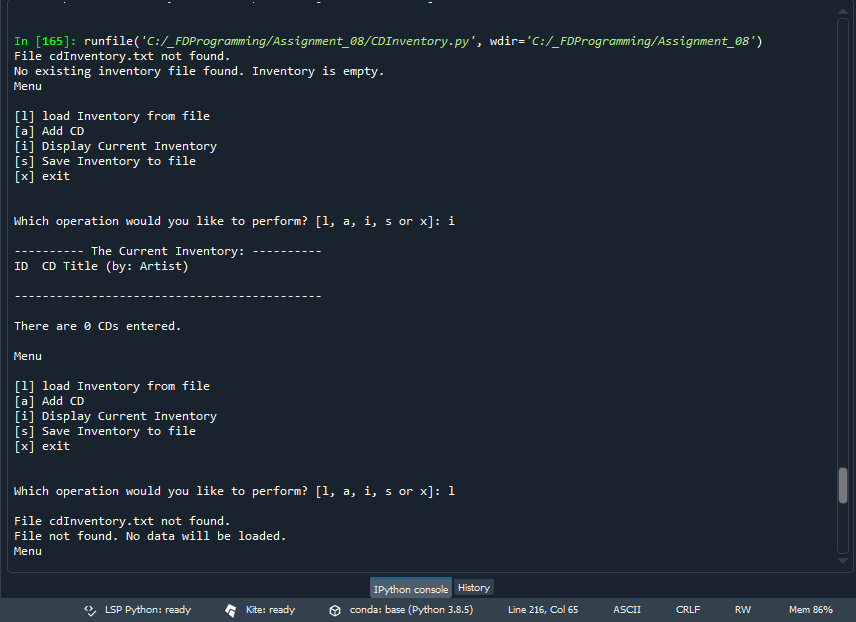
The docstrings in these snippets are truncated to take up less space.

# Summary

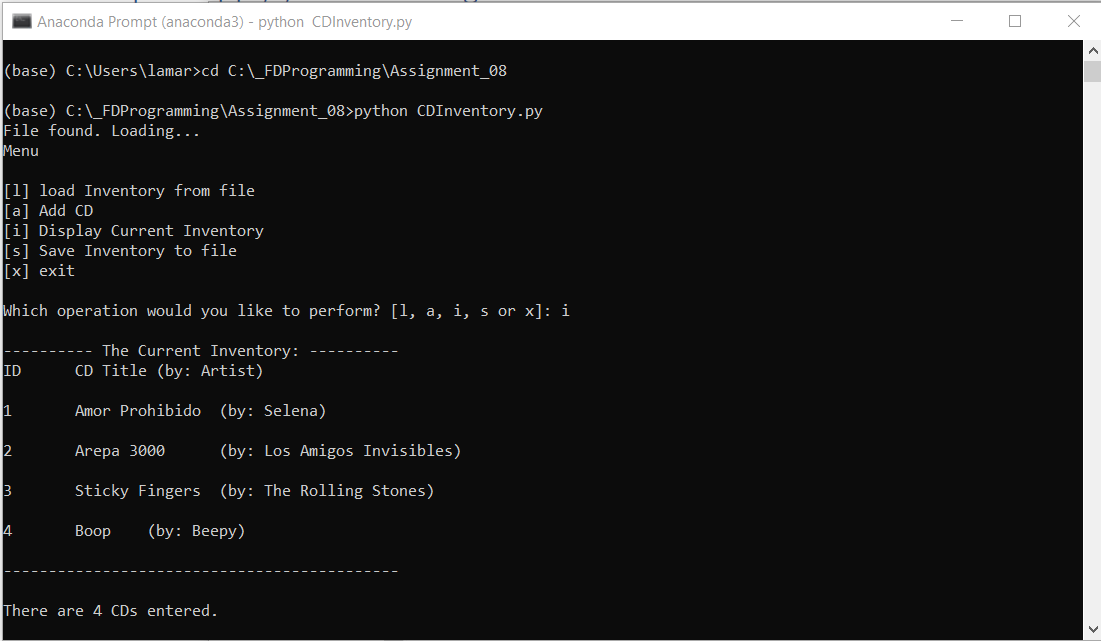
Although I feel like I should have figured out how to incorporate OOP more into the other classes in the script, including using my setter elsewhere, re-doing this script with only the pseudocode forced me to think about ways I could improve on the previous script I was bringing it. It wasn’t a simple copy/paste, as there were new class names and I had to make sure everything made sense now that we were using a list of objects. I also had to update the docstrings. I still don’t fully understand the object class and instantiating objects, but I was able to integrate it into my now-familiar CD Inventory script.

# Appendix

## Script running in Spyder



## Script running in Terminal



## GitHub Repository

Here is the link for my repository for Assignment 8:

<https://github.com/marichaf/Assignment-08>