Andrew Moy

August 14, 2022

IT FDN 110 A Su 22

Assignment 6 (Module 6)

https://github.com/andrewrmoy/Assignment\_06

CDInventory.py program with conversion of running code into Classes and Functions

Introduction

For this week’s assignment, we again did not cover anything new in class directly but through the available course material, we were introduced to classes and functions in general.

Using that new knowledge, we returned to the CDInventory program yet again and this time, we were tasked to convert the previous code we were familiar with into functions within class separators. We were given most of the groundwork already laid out as all the menu and functionalities were already prewritten, so the amount of completely fresh code work required to shape out functions was much lower than the previous weeks where we were asked to create them from scratch.

Creating the Program

Like last week’s module, the basic framework for the entire program was already laid out for us. But instead of doing a modification of the data structures, we were asked to convert all of the previously written framework into easy to call functions under specific classes, which would in theory clean up the code and make it more streamlined and easier to read and understand.

For me personally, there was much confusion as to what exactly was needed upon just opening the assignment file initially. What did processing functions mean when the bottom half of the program already listed the things needed to run the program? As I later figured out after much consternation, the commented out TODOs listed in the code block served as the guidelines for what to do.

Once that was realized the overall difficulty of the assignment became much more manageable. I went back through the menu options available in the original program excluding the exit option since it lacked a TODO. Since the first one on the list after the exit was load inventory, I started with that, defining a load\_inventory function in the DataProcessor class. Because the original code also already included calling from our input/output class, we could simply take it and place it into our DataProcessor function without having to make any major changes to it. Once that was done, I went back to the original code in the main loop and converted it to simply one line calling the new load\_inventory function.

For the next one, adding a new CD to the inventory, the comments specifically separated the code into calling both IO and DataProcessor, so we had to create a function for each. For the IO class, I created a get\_CD function to obtain our three required inputs and place them into identifier variables. I then went into DataProcessor and created a add\_CD function to take the get\_CD function’s data and append it into our primary list table, which was then called again into the main loop.

Showing the inventory was already taken care of with a show\_inventory function in the IO class, so that was also skippable. For deletion and saving, it was more like the load\_inventory function, where most of the code was already present and only required a quick conversion into brand new functions to be called in the main loop.

Summary:

Once I covered converting the TODO guidelines, I felt I was finished with the overall objectives and loaded it to run in both Spyder and the default terminal, but I honestly did not expect to see any issues if misspellings were cleaned up. As seen below, there was no difference in functionality running between last week’s version of the program and this week.



