Patrick Danielson

03/07/2021

Foundations of Programming (Python)

Module 08 – Assignment 08

Classes and Objects

# Introduction

This document summarizes the work performed for Module 08 of the Foundations of Programming course. The purpose of this module was to become familiar with the process of creating custom classes and methods in Python and utilizing these objects to interact with files. The problem addressed in this assignment was to take a starter code that contained only pseudocode and develop a functioning the CD\_Inventory program that utilized custom classes and CD objects. This program would additionally contain structured error handling and a menu that the user could interact with.

# Creating CD Objects

The first major challenge in this assignment was to create a class blueprint for a cd that contained attributes of the ID, Title, and Artist that could be set and accessed for input into the inventory. To accomplish this, I initially create private attributes in the \_\_init\_\_ constructor that received values when the object was instantiated. However, this would allow corrupted data to be instantiated into the object without proper error handling and would produce redundant code when the setter was established for each property. Therefore, I chose to construct the private attributes as None and then called the setter methods for each attribute (cd\_id, cd\_title, and cd\_artist) where the error handling was established to check for data validity. Understanding this distinction in calling methods within the CD class took some time and additional research, including one Datacamp article that I believe explained the interaction well.[[1]](#footnote-1)

# Creating IO Class

After being able to successfully create and interact with the CD objects, the next challenge was to create the Input/Output class that would handle user interaction and the menu. Fortunately, much of this code had already been established in previous assignments, and I utilized my program from Assignment 07 to create most of the static methods within this class.

There were a couple functions that had to be modified in order to work with a list of objects rather than a list of dictionaries. In creating a new entry using the get\_new\_entry() function, the program would still return a list of the data values, however those values would then be used in the main program to instantiate a new CD object that was appending to the table stored in runtime. Additionally, in order to display the inventory to the user, I modified the for loop in the display\_inventory() function to iterate through objects in the list table and print out the attributes for each object to the window.

# File Processing Class

The final class to be established would enable the program to interact with the CDInventory.txt file that would store the CD object attributes in a plain text format. I processed the save\_inventory() method in a similar way to printing the inventory to the window. Once the file had been opened to write data to, I used a for loop to iterate through the objects in the list and write each attribute to a list row of data. This could then be formatted and written to the file.

The next step was to read in data from a plain text file. This was accomplished in the load\_inventory() method, which would iterate through each line in the text file and create a list of data containing 3 items. These items were then used to instantiate a new CD object, which was then appended to the inventory stored in memory.

# Executing the Program

The execution of the program is summarized in this section. The program is first executed in Spyder where I attempted to first add an entry with an invalid ID number to display the error handling. I then created a valid entry and displayed the inventory. This execution is shown in Figure 1 and Figure 2. Figure 3 shows saving the data in Spyder.

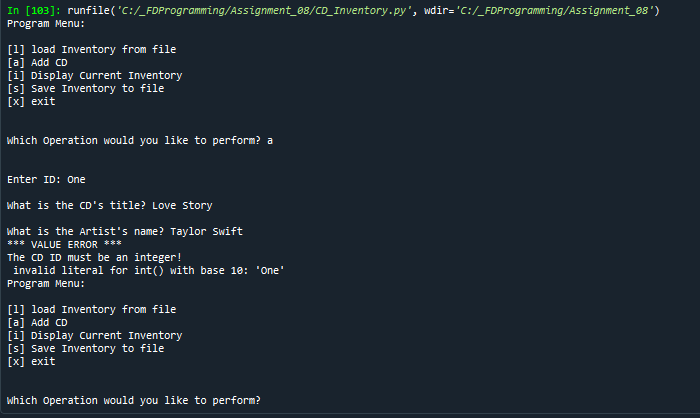


Figure 1: Spyder Execution for Invalid ID Entry

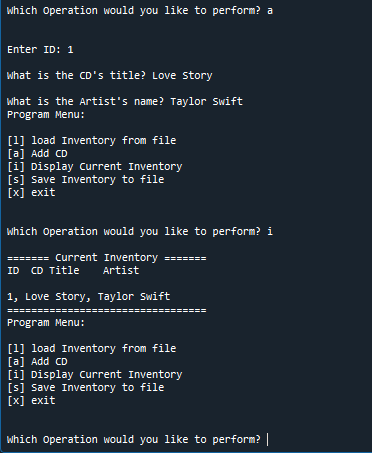


Figure 2: Spyder Execution for Adding and Displaying Entry

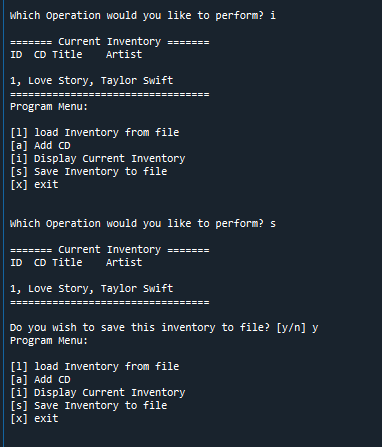


Figure 3: Spyder Execution Saving Data

The terminal execution of the program is shown in Figure 4 for reloading the data file that was previously created in the Spyder execution. Another entry was added and saved within the terminal and shown in Figure 5. The final saved CD\_Inventory.txt file is shown in Figure 6. The files submitted for this assignment do not include the final saved CDInventory.dat file.

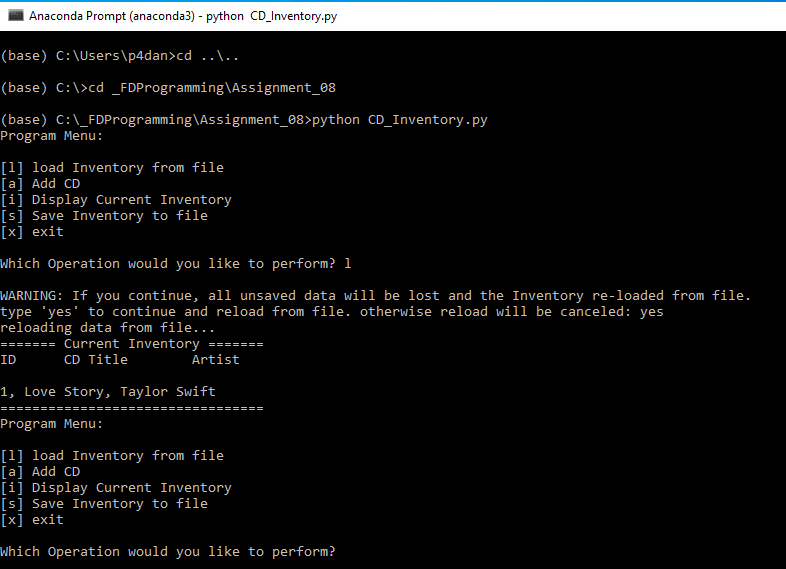


Figure 4: Terminal Execution Reading in Data

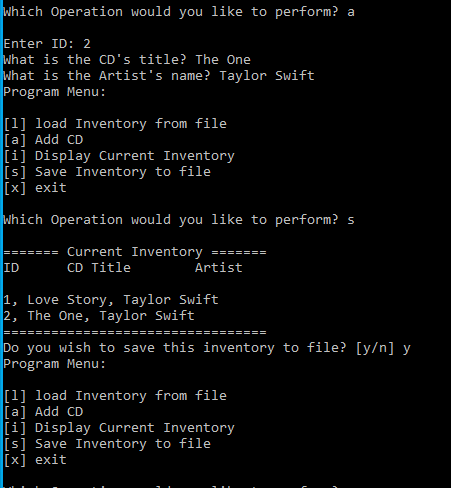


Figure 5: Terminal Execution for Adding and Saving Data

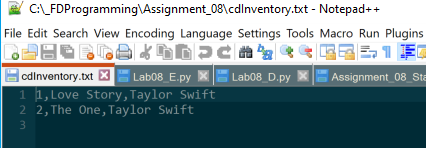


Figure 6: Final CD\_Inventory.txt File

# Summary

In this assignment, I was able to successfully build custom classes for file processing, user input/output, and for creating and operating on custom objects. I was able to efficiently implement structured error handling to ensure that the data instantiated and accessed within the objects were secure and valid. By doing so, I have learned to work with classes and objects in the context of creating various methods to access private attributes of object. The Python script and CDInventory.txt file used in this assignment, along with this document, are posted on GIT[[2]](#footnote-2) under the Assignment\_08 repository.

1. <https://www.datacamp.com/community/tutorials/property-getters-setters?utm_source=adwords_ppc&utm_campaignid=1565261270&utm_adgroupid=67750485268&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=332661264374&utm_targetid=aud-299261629574:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=9033311&gclid=Cj0KCQiA7YyCBhD_ARIsALkj54pufJJJt3KFlh8Cv3XUlYHy56WmwLVwus50qneNDoVJB4zR1136Ol8aAupaEALw_wcB> [↑](#footnote-ref-1)
2. Patrick Danielson GIT Repositories: <https://github.com/pdaniel441?tab=repositories> [↑](#footnote-ref-2)