Arielle Waller

September 2, 2020

Foundations of Programming (Python)

Assignment08

Assignment08

# Introduction

The task for this assignment was to take a starter script with pseudocode and add code so that the program ran as expected.

# Getting Started

The first step I took toward completing this assignment was reading over the pseudocode. I noticed that most of what was asked for had already been coded in last week’s assignment. As a result, I used that as a template, making changes to function names and variables where appropriate. This included changing all references to *lstTbl* to ﻿*lstOfCDObjects*. I also removed the function for deleting an entry in the table since that was not included in this week’s assignment. Finally, I changed the filename to a *.dat* extension so that I would be able to save the list of CD objects directly to the file rather than convert all of the information to a string first.

# Creating the Custom Class

Next, I decided to add code to the CD class. In my constructor, I initialized my CD object with the parameters *self,* *cd\_id, cd\_title,* and *cd\_artist.*

Initially, I used private variables here.

A screenshot of a cell phone

Description automatically generated

Figure - Initializing variables for the CD object.

Next, I created my getter and setter properties for each variable.

A screenshot of a cell phone

Description automatically generated

Figure - Sample getter and setter properties for the CD object.

For my setter property for *cd\_id*, I raised an exception if the user’s *id\_input* was not numeric. I commented out the section of code in the static method *cd\_information* so that I would later be able to test if this exception was raised successfully.

# Working with a List of Custom Objects vs. a List of Dictionaries

After making these changes, I ran the code to get an idea of what else I would need to modify. Since last week’s assignment worked with a list of dictionaries, I saw that I needed to modify the section of code in the static method, *show\_inventory*, in the input/output class that iterated through the dictionary values*.*



Figure - Original code for iterating through the dictionary row values in the table.

Since I would now be working with CD *objects*, I changed *row* to *cd* for clarity. I deleted *\*row.values()* and replaced it with *cd.cd\_id*, *cd.cd\_title*, and *cd.cd\_artist* so that the attributes of each CD object would be accessed instead.

# Getting the Exception to Raise

While the code ran, it was not raising an error when I entered non-numeric input for the CD ID. Through some [research](https://stackoverflow.com/questions/52120764/constraint-in-python-property-setter-not-applied)[[1]](#footnote-1), I [found](https://www.geeksforgeeks.org/getter-and-setter-in-python/)[[2]](#footnote-2) that it was because I was calling the [private](https://www.python-course.eu/python3_properties.php)[[3]](#footnote-3) attribute in my constructor. After modifying the code in the constructor, I found the error was successfully raised for non-numeric inputs.

A screenshot of a cell phone

Description automatically generated

Figure - Calling the cd\_id attribute directly in the constructor.

# Loading Data on Script Start

With the basics covered, I moved to the section of pseudocode instructing us to add code that loaded data from the file into a list of CD objects on script start.

I figured that this was the same chunk of code utilized in loading data from the file from the menu, so I copied and pasted that section of code above the main *while* loop.



Figure - Loading data from the file upon running the script.

Testing the script, I got the following *TypeError*:

A picture containing ball, clock, meter

Description automatically generated

Figure - Error generated when there is no table to iterate through.

Though the method *read\_file* accounted for the possibility that a file did not yet exist, I had no such error handling in *show\_inventory*. I added an *if* statement with *table* as the condition. Thus, if a table existed, the program would iterate through each CD object in the table. Otherwise, a message would display informing the user that they needed to add a CD first.

A close up of a screen

Description automatically generated

Figure - Accounting for the possibility that the table has no data.

# Running the Program

I tested the program in Spyder and the terminal and found it ran successfully.

A screenshot of a cell phone

Description automatically generated

Figure - Running CDInventory.py in Spyder.

A screenshot of a cell phone

Description automatically generated

Figure - Running CDInventory.py in the terminal.

# Summary

I found this week’s assignment challenging in that it was difficult to decide which classes needed to use objects. In the CD class object, I decided not to include any methods since the methods were defined in other classes. I am wondering if there would be any benefit to putting them there directly.

I also found it challenging in that the pseudocode left room for interpretation. I felt like there were so many ways to implement what was asked that it was difficult to decide which approach to take.

Though I was able to get the exception to raise in the *setter* for *cd\_id*, I had trouble getting the exception to raise without stopping the program. I attempted to use a *try-except* block, but when I used this, the exception did not raise at all.

I think I need to become more familiar with *getters*, *setters*, and *properties.* I think I have a basic understanding, but when it comes to manipulating them and using them with exceptions, I am still confused. I do not understand the purpose of privacy with a dunder attribute, field, or method if it is private only by convention.

1. Retrieved 2020-September-02 [↑](#footnote-ref-1)
2. Retrieved 2020-September-02 [↑](#footnote-ref-2)
3. Retrieved 2020-September-02 [↑](#footnote-ref-3)