Andrew Hernandez

March 6, 2021

Foundations of Programming (Python)

Assignment 08

Professor Dirk Biesinger

**Classes, Getters and Setters**

*Keywords: OOP, Classes, Getters, Setters, Methods, Functions, Error Handling, Try-*

*Except, Exception Class, Docstrings, Dictionaries, 2-dimension lists, User input,*

*While-loop, if-elif-else statement*

**Introduction**

Prior to this assignment, we have mainly worked with python as a single file program; however, with this assignment, we have began focusing on Object Oriented Programming (OOP) which will allow us to create applications that are divided into multiple files. OOP is a style of programing where you structure your program by bundling or combining properties and behaviors into individual objects[[1]](#footnote-1). In python everything is an object. This means that all entities have some attributes and associated functionality such as methods. For this assignment, we were tasked with continuing developing our CD inventory program to add a more pythonic way of writing code. When I first saw the pseudocode for this assignment, I was a little caught off guard and confused on how to approach this assignment. After several hours of rereading the chapter and writing and rewriting my code I think that I was able to complete this assignment how it was meant to be done. Furthermore, I realized that we are not breaking up the entire program with getters and setters, but we are only do so with the CD class that was created for us.

**Classes**

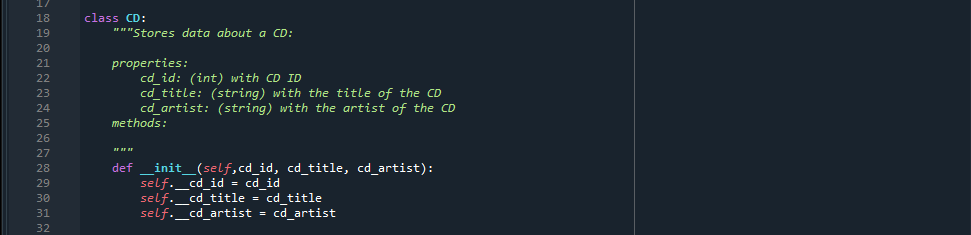
The first part of the assignment was to read the pseudocode and figure out where we need to modify the code with classes. In the pseudocode, there was a new class called CD. Simply put, a class is a template for creating objects. Everything in a class should be related and provide the standard features for an OO program. Below we see the CD class which required use to add parameters to be used in the main of the application. We use the \_\_init\_\_ function to initialize the class and define the class variables. Here we see that we have 3 variables that have been used throughout the CD inventory program. After defining the variables, we can now use those variables when the object is instantiated (Created). In python, we can instantiate an object by simply following this syntax: cd = CD(). With in the object, we can pass the variables that represent the variables that were defined in the \_\_init\_\_ function.

Figure : Creating a class and defining class variables.

**Getters and Setters**

Once the variables are created, we need to use the getters and setters so that we can encapsulate the variables to make that they are restricted from being accessed directly which can result in accidental modification of data[[2]](#footnote-2). For the variables, we first need to create the getters by adding a *@property* decorator which tells python that these functions are the getters for the variables. After we have set up the getters, we can then create the setters. To create the setters for the variable we create a new function with the getter name as the decorator followed by the dot setter word such as *@cd\_title.setter.* Then, we can add some logic to the setter to make sure that the variable conforms with how we want the application to use it. For instance, if the variable is an integer we can use the following code in the setter to make sure that the variable isn’t used outside of its property:

@cd\_id.setter

def cd\_id(self, new\_cd\_id):

if type(new\_cd\_id) == int:

self.\_\_cd\_id = new\_cd\_id

Here we can see that the variable is the cd\_id which needs to be a integer. Within the setter we add an if statement the make sure that they type of the variable that is passed in is of the int type.

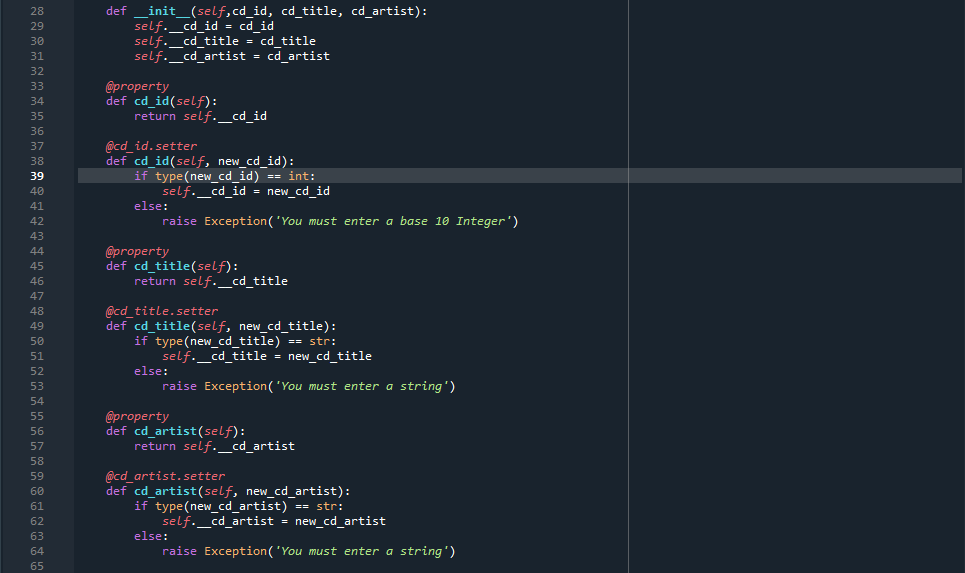
Here are more examples of getters and setters that were used throughout the program. We can see from the image that we can also add some error handling so that the application doesn’t crash in case of a invalid input.

Figure :Getters and Setters

**Appending Data using Classes**

One of the most difficult parts of this assignment was appending new user inputted data into he list of dictionaries that we had. At first I ran into an issue that was causing my IDE to crash and reset the kernel. After the discussion in class, I realized that I wasn’t adding the \_\_ (dunder) to my variables which made them inaccessible in the main of the program. After adding the \_\_ to the variables I was able to pass those variables to the append method that added the data to the list.

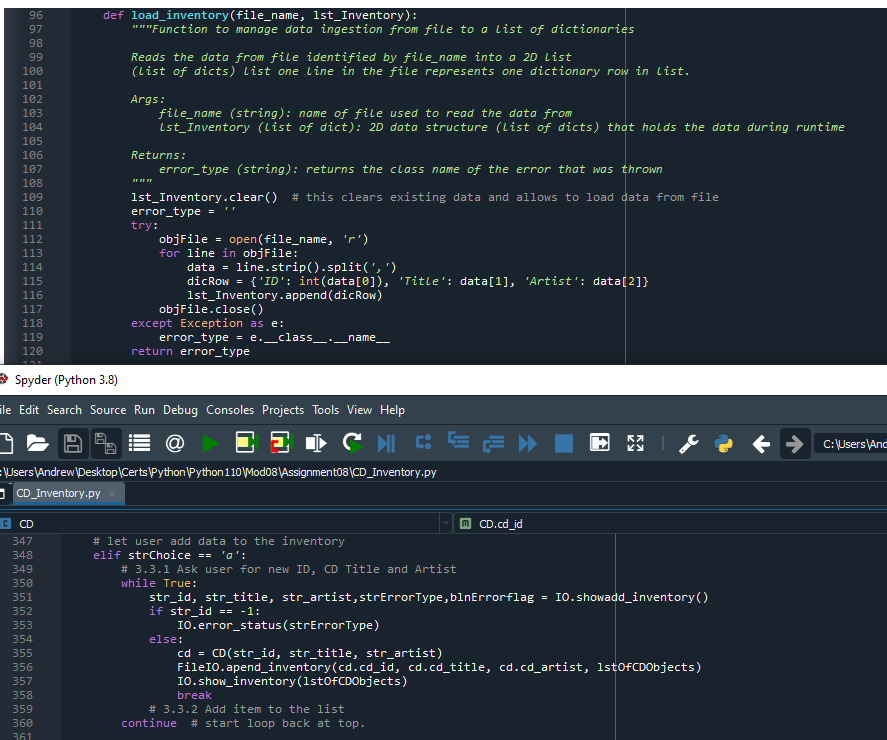
****

Figure :Working with private class variables

**Summary**

The remainder of the assignment was just to organize our code that we wrote in previous assignments. I did have to reformate the code so that it would line up with the pseudocode that was provided. Also, I added some more docstrings to explain what our classes where doing and which methods were in those classes. Adding error handling try-except blocks also took some time because my indention was not lining up but that was an easy fix compared to the logical problems that I faced when trying to append the list.

Here is the link to my repo on GitHub: <https://github.com/Slugdrew/Assignment_08>

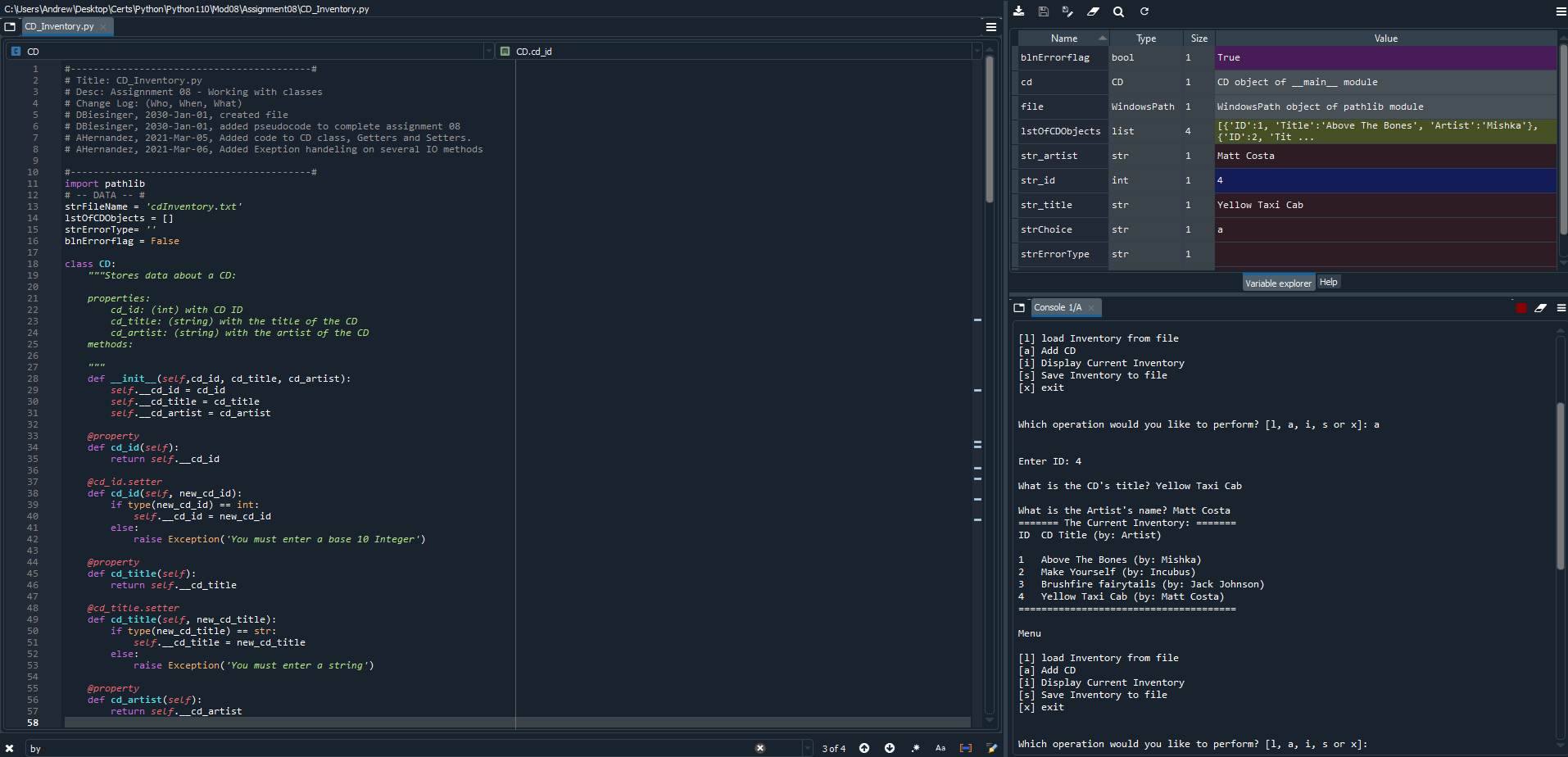
Below are images of the application running in Spyder and in the CLI:

Figure :IDE execution of program

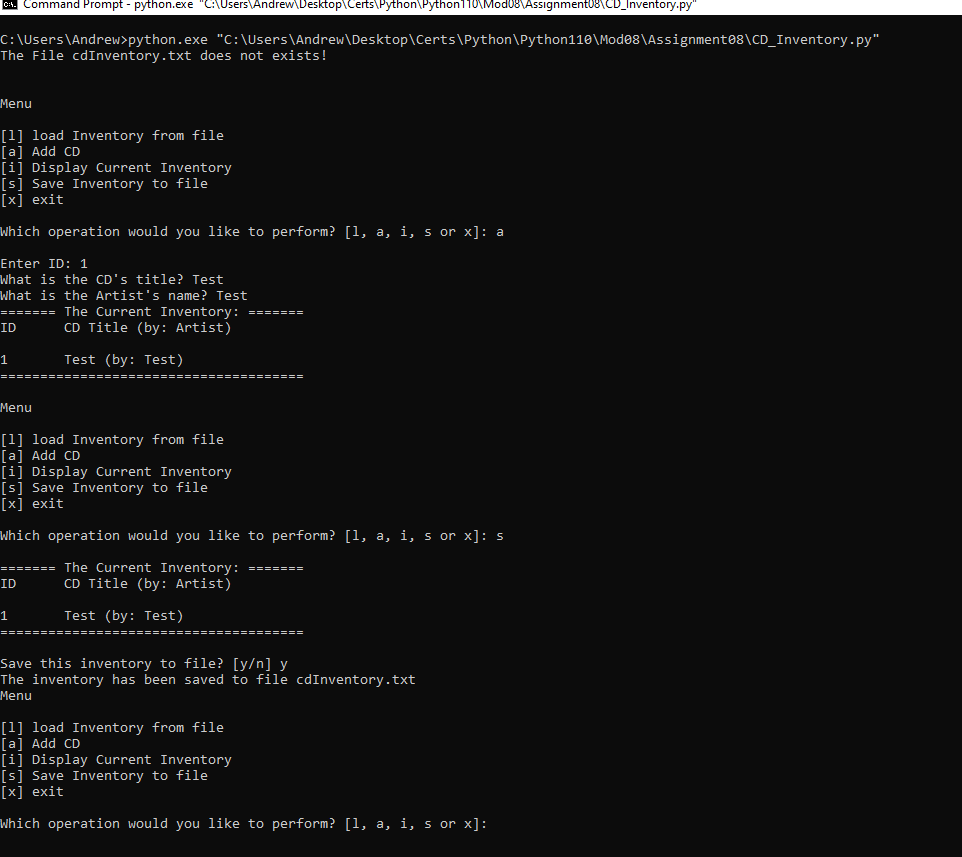


Figure : Program execution CLI

1. <https://realpython.com/python3-object-oriented-programming/#:~:text=Object%2Doriented%20programming%20(OOP)%20is%20a%20method%20of%20structuring,object%2Doriented%20programming%20in%20Python>, accessed 6 March 2021 [↑](#footnote-ref-1)
2. <https://www.geeksforgeeks.org/encapsulation-in-python/> accessed 6 March 7, 2021 [↑](#footnote-ref-2)