Alok Malavia

2022 Mar-21

Foundations of Programming: Python

Assignment\_08

Python – Object Oriented Programming and Classes

# Introduction

In this document we covered Object-Oriented Programming to understand a different way of thinking about programming. We worked with creating Classes and defining objects. We also learned to write methods and create attributes for our objects. We then Instantiated the objects from the classes. In essence we learned how everything is an object, and how a certain object has attributes and methods. We also learned how to create custom functions and use various attributes within those functions. Having said that, this was probably the most challenging topic covered thus far in class and I found the assignment to be equally challenging.

# Classes and their relation to OOP

Classes can be considered as blueprint for an object. A class isn’t an object, however it’s a design for one. Just as a foreman can create many houses from the same blueprint, a programmer can create many objects from the same class. As a result, each object (also called an instance) instantiated from the same class will have a similar structure. So for example, if you have a checking account class, you could use it to create multiple checking account objects and each of those checking account objects would have the same basic structure. And each one might have a balance attribute for example. Classes almost always include methods, basically things that an object can do. In this module we also learned

## Methods aka. Constructors

Methods are like functions in a script. They allow us to organize our statements into blocks that can be invoked by calling the method’s name. The one difference is that a method call also submits a reference to the object it’s invoked on, so the first attribute supplied to a method is the ‘*Self’* reference. (More on the ‘*Self’* reference below). Constructors are a dedicated (special) method that is invoked when creating an object. Constructors make it easy for us to ensure that we have the proper datatypes in the fields. They also allow for the pre-populations (defaults) of values to the fields. In essence it is the same principle as using default values in functions. Python’s constructor method is the dunder init method: \_\_init\_\_(). The instantiate an object you simply called the classes name as if it were a function. We were able to do this numerous times during Assignment\_08. Constructors are a special method. They run once during the creation of the object. They are limited to this one purpose and hence are implicitly called when creating an object with a function call like syntax.

## The ‘Self’ Keyword and Attributes

The self key word is the first parameter in every method. Every method that is called by an object automatically receives a reference to that object. This way the class knows on which object to use the methods. In Python, attributes are internal fields or variables that hold data. One issue with attributes is that they are ‘just’ variables. We have no control over what goes into them or how they change during runtime from external to your class unless you write specific code to validate values before they are assigned. In order to do so, we can use special methods called Properties.

## Properties: Getters and Setters:

One common concept pf controlling validity of values assigned to attributes in our classes is to make the attributes private and enforce interaction with them thru methods that have control mechanism built in. These methods are called properties. Typically, you’ll create two for each attribute. One to set it and one to access it. They are known as Setters and Getters. Getter methods let you add code to format the fields or attributes data. Even when no formatting is necessary, a getter property is required to allow access to the private field or attribute. Getters are created like any other method with the addition of a decorator to identify it as a property getter. The Setter methods let you add validations as well as error handling statements to evaluate parameters passed into the property. If the values are valid, then it is assigned to the field or attribute. We did so in our assignment right from the start, and then we used the fields/attributes through our class. When defining getters and setters in Python, we have to make sure to first define the getter property followed by the setter property.

# Assignment\_08 - The CD Inventory Program - Working with Classes

The goal of Assignment08 was to take the Assignment\_08\_Starter.py script and add code to it by working in classes, methods, attributes, properties, and other topics we learned throughout this past module. In all honesty, I found this assignment to be very difficult. I had to take a trip this past weekend, so I couldn’t devote as much time to it as I would’ve liked. I referred to some of our previous labs in the module to setup my properties (i.e. my getters and setters). I started my script with working on the sections we had worked on in Assignment\_07. Many of the functions were changed, but we had to remember that in assignment\_07 we worked with a list of dictionaries, and in this assignment, we would be working with list of objects. I decided that I would keep the same names as what I would use for the user input and I was used cd\_id, cd\_title, and cd\_artist. Although this seemed like a great idea at first, I soon realized that it got very confusing when I started to work on other parts of the script. In hindsight, I will likely use different names. When I first read the starter script, I couldn’t exactly understand where really to get started, but then I looked at the second line and I realized that we would be creating a list of CD Objects or to put it differently I believe that our inventory list would be an object of our class CD and each line would represent a CD Object.

After I was finally done setting up my functions in my methods portion of the script, I decided that I needed to work on my FileIO class and work on the read\_file function. In the read\_file function I assigned the values to my variables, i.e. cd\_id = data[0], cd\_title = data[1] and cd\_artist = data[2], but then I also had to create a variable for all three of my attributes that I had defined in my CD class so that I could take the inputs I received form the user and then apply them appropriately. Basically, I had to take the inputs and since the inputs are attributes (i.e. attributes are internal fields or variables that hold data), I had to apply them so that they could become part of the CD Object class. I did so by creating a method called add\_cds\_lstobj and I basically appended the data to the lstOfCDObjects. Since we were no longer using picking in this assignment, I had to also modify the section to display the data which was in a .txt format. For that I basically created another method called text and called it in my write\_file function so that each item could be added and written to the text file and displayed back in simple text. What was good about this as well, is that the data in the file would be plain text, but also would reflect each of the CD object attributes as such. Like last week, I added Error handling to certain portions of the code, but I realized that it would work in some sections, but not in others, especially the file handling portion. I need to go back and focus on this area some more to determine how to make it work.

# Script Execution in the Spyder Console:

Text

Description automatically generated

Figure - Script Execution in Spyder -1

Text

Description automatically generated

Figure - Script Exection in Spyder - 2

# Script Execution in the Terminal Window

Text

Description automatically generated

Figure 3 - Script Execution in Terminal Window - 1

Text

Description automatically generated

Figure 4 - Script Execution in Terminal Window - 2

# Summary

This was a very tough assignment. Although I somewhat understand the concepts we covered, it really got tough when I had to define my functions under the methods, but more so then applying and calling those methods in a different class, and then calling additional functions in a class such as FileIO or Class IO. I would like to go back over this script and understand all the various concepts we covered in the module. Object Oriented Programming is something I think will take time for me to get used to, but this assignment was a good starting point, even though it was a tough one. Given that I had very little time to work on it due to a long week and also short weekend, I will be going back to it to understand where I went wrong or what I could do differently. The code and additional details for this Assignment can be found on my [gibthub repository](https://github.com/Amalavia42/Assignment_08).