Dan Oranski

3/16/20

IT FDN 100 B

Assignment 08

Observations of Assignment 08

# Introduction

This document covers the work through of Assignment 08. In this assignment, students applied the concepts of Object Oriented Programming (OOP). The concepts were unified by creating a CD inventory script per the provided pseudocode.

# Object Oriented Programming

Object Oriented Programming (OOP) employs the basic concept that everything is an object. With OOP, programmers define the data type of a data structure, and also the types of operations that can be applied to the data structure. Programming in this manner, the data structure becomes an object that includes both data and functions. Programmers can also create relationships between one object and another. This allows objects to inherit characteristics from other objects.[[1]](#footnote-1)

The advantage of this is that programmers can create modules that do not need to be changed when a new type of object is added. New objects can be created that inherit many of the features from existing objects. This allows object oriented programs to be modified more easily.

LAB 07-A directed students to modify LAB 06-C by adding in functions that read from and wrote to a text file. Below is a screenshot of the results.

# Classes

Classes are the blueprint for an object. They are essentially containers for the data and functionality of an object. The object is simply an instantiation of the class of the object. Within classes, functions are called methods, and variables and constants are called fields. Each instance of a class uses the same blueprint, but exists as it’s own copy in memory. This prevents changes to one object affecting all objects of a class.

The principle construct of a class has the following structure:

class ClassName(InheritFromBaseClass):

# -- Fields -- #

# -- Constructor -- #

# -- Attributes -- #

# -- Properties -- #

# -- Methods -- #

This code creates a class called ‘ClassName’, which inherits it’s characteristics from the class ‘InheritFromBaseClass’. If no class is specified, Python defaults to ‘Object’.

# Fields

Fields are the data stores of a class. Fields are created in the same way in which variables are created.

LAB 08-A directed students to create a class called ‘TrackInfo’ with fields for position, title and length. The results are displayed below.

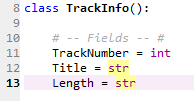


Figure 1 - Creating fields

# Constructors

Constructors are a special type of method that is invoked when creating an object. Constructors allow programmers to ensure that datatypes are enforced within fields. They also allow for pre-populating fields with default values. Constructors run once during the creation of an object. This is there only purpose and they are called implicitly as a result.[[2]](#footnote-2)

Python’s constructor method is the \_\_init\_\_(). To instantiate an object, you call the class’s name as if it were a function:

obj1 = Class(‘message’)

Python implicitly calls the \_\_init\_\_() method and passes any arguments provided when creating an object to the \_\_init\_\_() method.

LAB 08-B directed students to add code to LAB 08-A to create a Constructor class. The results are displayed below.

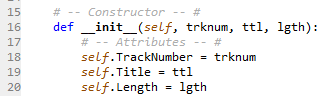


Figure 2 - Constructors

# Destructors

Destructors are methods used when an object gets destroyed or de-allocated. Destructors are generally responsible for freeing up memory used by objects and for cleaning up references.

# The Keyword ‘self’

The word self is not an actual keyword, but it’s so widely accepted as one that it might as well be. The keyword ‘self’ is the first parameter in every method. All methods called by an object automatically receive a reference to that object. This allows methods to get a reference to the object calling the method and allowing access to the attributes of the method. Although it is not required, the first parameter should always be named self.

# Attributes

Attributes are internal fields or variables that hold data. One issue with attributes is that they are ‘just’ variables. The programmer has no control over what goes into them or how they change during runtime from external to the defined class unless specific code is written to handle this.

In LAB 08-C, students were directed to add the code needed to create attributes to the code from LAB 08-B. The results are displayed below.

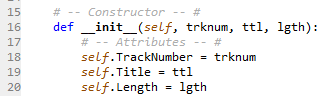


Figure 3 – Attributes

# Properties

A common concept used in controlling the validity of values assigned to attributes in a class is to make the attributes private and enforce the interaction with them through methods that have a control mechanism built in. These special methods are called properties. There are two typical properties for each attribute: one to set it and one to access it. They are commonly referred to as “getter” or “accessor” for reading the attributes and “setter” or “mutator” for writing the attribute. In Python, you make an attribute private by pre-pending a double underscore to its name.

In LAB 08-D students were directed to add code to LAB 08-C for the previously created attributes. The results are displayed below.

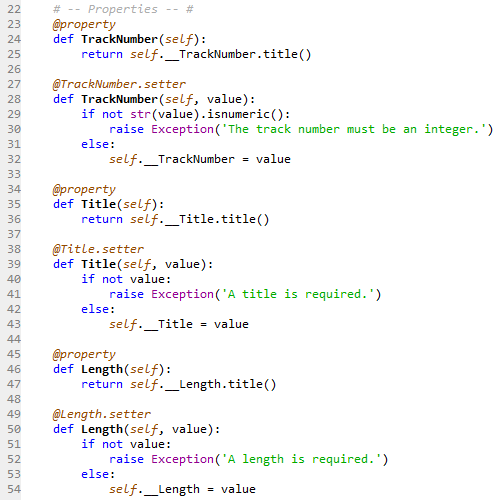


Figure 4 - Properties

# Methods

Methods are like functions in a script. Methods allow programmers to organize statements into blocks of code that can be invoked by calling the method’s name. The main difference is that methods require the ‘self’ parameter as the first attribute.

For LAB 08-E students were directed to add code to LAB 08-D that added methods. The results are shown below.

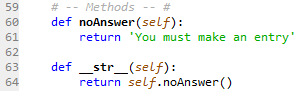


Figure 5 - Methods

# Application of Learning Objectives

Tying everything together, students were asked to create code based on provided pseudocode. This was essentially a recreation of the CDInventory script that has been built on with each module. Error handling, docstrings, functions and classes were all utilized to create the script. The script was created, troubleshot, and run in the Spyder IDE.

The results of Assignment 08 are shown below.

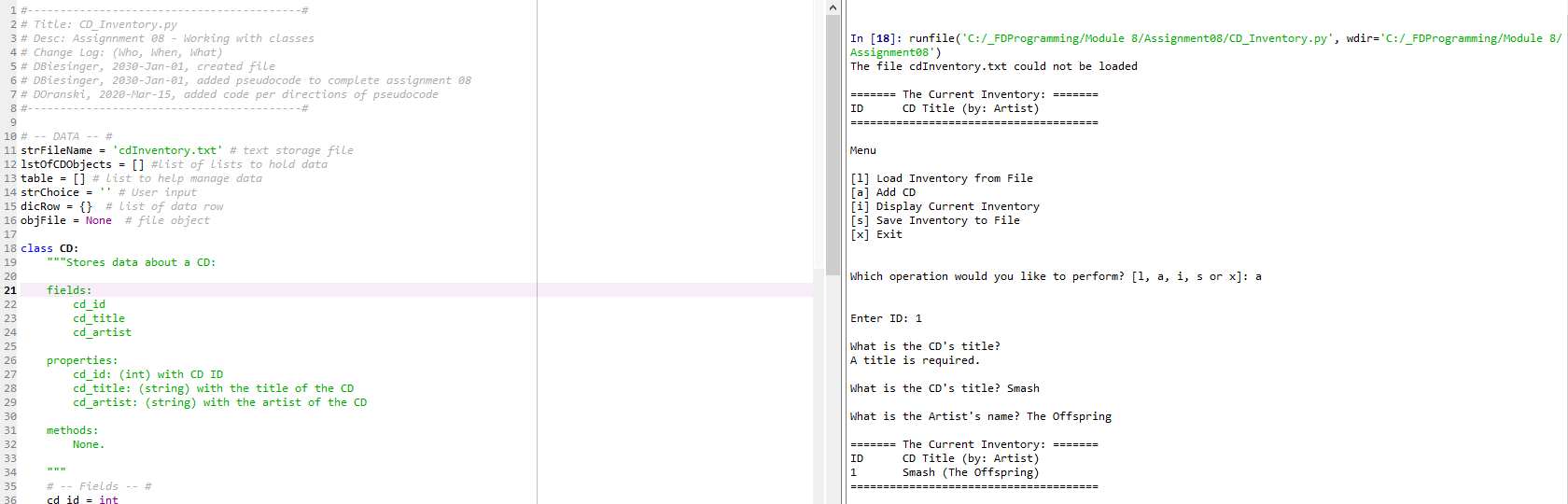


Figure 6 - Assignment08-1

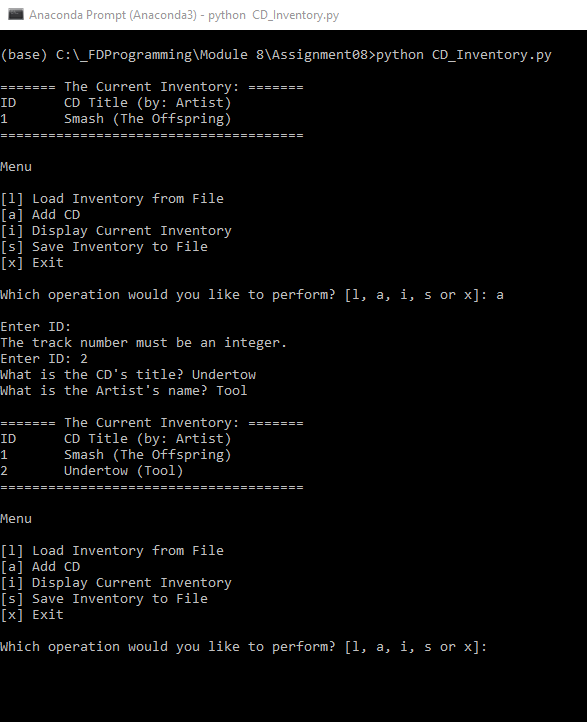


Figure 7 - Assignment08-2

The files for this Assignment were uploaded to GitHub [here](https://github.com/angryeng/Assignment_08).

# Summary

Assignment 08 was less challenging than Assignment 07. Working with classes makes sense and after solidifying the concepts from Modules 06 and 07, this assignment was much more fluid. Being able to create and utilize your own classes is a very useful tool.

1. <https://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html>, retrieved 2020-Mar-15 [↑](#footnote-ref-1)
2. <https://www.geeksforgeeks.org/constructors-in-python/>, retrieved 2020-Mar-14 [↑](#footnote-ref-2)