Sergey Timchenko

3/07/2021

Foundations Of Programming: Python

Assignment 08

Object Oriented Programming

# Introduction

In this assignment the student learns about object oriented programming. Object oriented programming is based on creating the blocks that define and describe the object that can be instantiated several times with the same or similar properties. The student learns how to do the following:

* Create classes to define the object
* Write methods and create attributes for objects
* Instantiate objects from classes
* Restrict access to an object’s attributes

The student modifies the starter program created for Assignment 08 to describe and instantiate the objects. In this assignment that will be the compact disks that have similar properties but every disk has its own ID, Name and Artist.

Classes

Classes are the blueprint or a template of the object. When the class is created it defines the name properties and functionality of the object. Later when the instance of the object is created this instance defines the particular properties of functionality of the object based on the functionality and properties defined in the class.

Example of the class definition from the Assignment 08

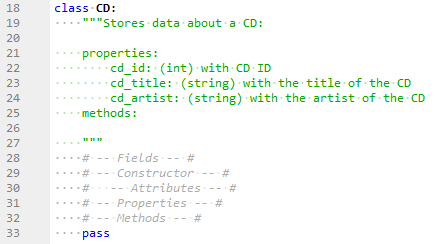


Figure 1 – Class definition

Fields

Classes usually contain fields to store the data. Fields get created similar way as variables

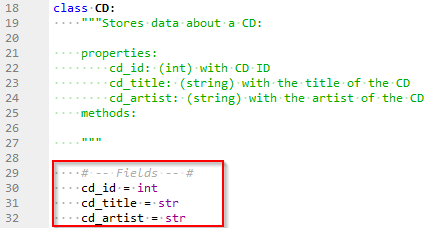


Figure 2 – Example of fields definition from the Assignment 08

# Constructors

Constructors are spatial methods that define the properties of the object when the object is being instantiated.

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.

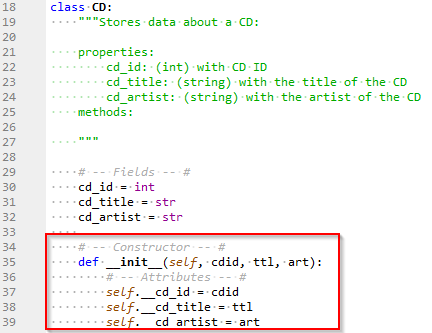


Figure 3 – Example of Constructor definition from the Assignment 08

# The Keyword self

All methods in python including some special methods like initializer have first parameter *self*. This parameter refers to the object which invokes the method. When the programmer creates new object the *self* parameter in the \_\_init\_\_  method is automatically set to reference the object that just been created.

Here is an example of use *self* from <https://realpython.com/> tutorial[[1]](#footnote-1)

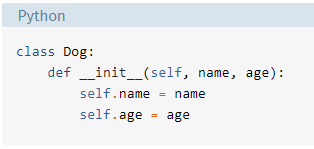


Figure 4 – Example of using self keyword realpython.com

# Attributes

In Python, 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.

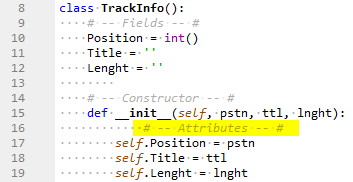


Figure 5 – Example of Attributes from Lab08\_C

# 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.

Here is the example of using properties (setters and getters) from Lab08\_D

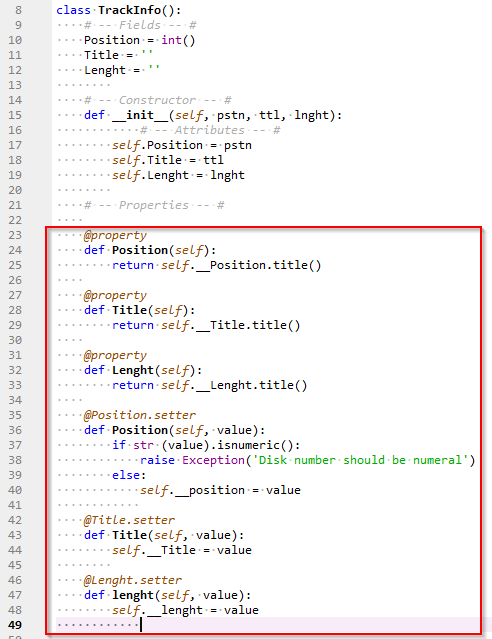


Figure 6 – Example of properties from Lab08\_D

# Methods

Methods are like functions in a script: They allow the programmer to organize the 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.

The student was asked to add methods and use a \_\_str\_\_ method to return a formatted content of the attributes in Lab08\_E. Here are the results:

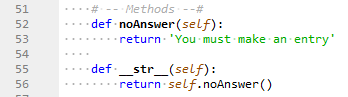


Figure 7 – Example of Methods from Lab08\_E

# Creating a script

In this assignment I was asked to modify the starter file for Assignment 08 by adding TODO tasks.

During modification of the program I’ve added the following features of the object oriented program:

* Fields
* Constructors
* Attributes
* Properties
* Methods

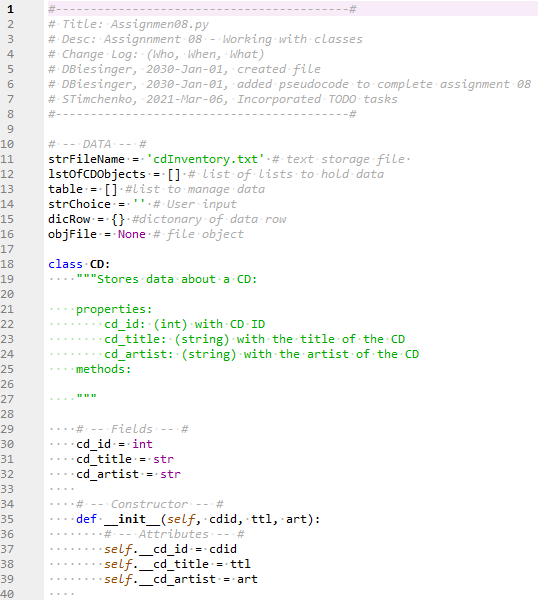


Figure 8 – Beginning of the Assignment08 code

# The result of the program run

After the code is modified we are testing the program in Spider sub window. Per instruction in the starter file

The program does the following:

1. It shows the user the current inventory
2. It lets the user add the new data to the inventory
3. It lets the user to load the inventory from the text file
4. It lets the user to exit the program

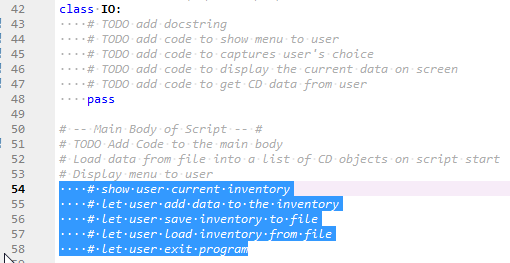


Figure 9 – TODO tasks in the starter file

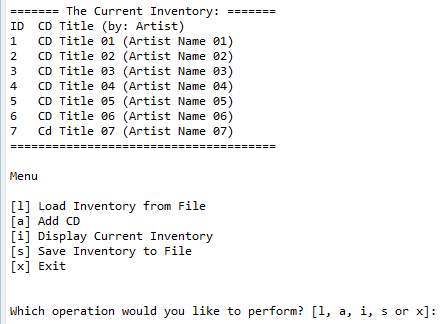


Figure 10 – Running script in Spider IDE

Now we want to check if the program runs correctly in Anaconda command line window.

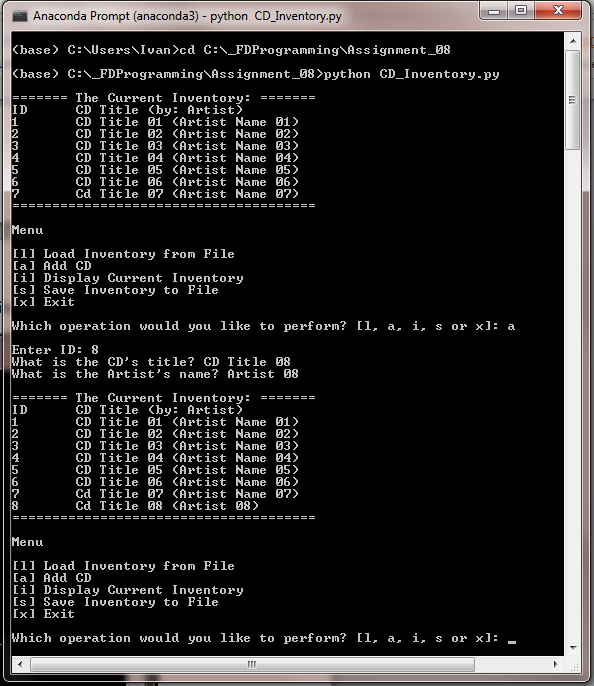


Figure 11 – Running the program in Anaconda Prompt window

The files for this Assignment were uploaded to GitHub [here](https://github.com/snt2177/Assignment_08.git).

# Summary

In this Assignment I used the elements of Object Oriented Programming. I created the classes to define the CD inventory, wrote methods and created attributes of the objects, instantiated the objects and restricted access to the object attributes.

It was difficult to apply the more or less understandable theory of Object Oriented Programing to the particular tasks described in the starter file.

1. <https://realpython.com/python3-object-oriented-programming/>, retrieved 2021-Mar-3 [↑](#footnote-ref-1)