Jonathan Den Haan

12/04/2022

Foundations of Programming: Python

Assignment 08

**Module 8**

**Introduction:**

This module introduces us to object oriented programming. Basically, everything in python is an object with attributes and methods. We have already been using python objects, just not naming them as such. This module introduces us to that way of thinking teaches us about creating classes to define objects, writing methods and creating attributes for objects, instantiating objects and restricting access to object attributes.

**Labs:**

**Lab08A**

Text

Description automatically generated

This script works by defining a class, TrackInfo as an object. This object has 3 fields which are predefined as an integer and two strings. The script then defines an instance of this object as “objTrack1” and defines each field with strings. Then the script calls on objTrack1 to print each field. In this way a unique instance of TrackInfo has been created that contains the structure first defined in the class, with the data defined when the instance was created.

**LAB08\_B**

Text

Description automatically generated

This script uses a constructor. Constructors are only invoked when creating an object, and they allow you to call an object like a function, with arguments passed in at the calling point. This script just converts whatever number is passed in when the object is created to an integer and that becomes the value for the position field. Then the script prints the individual fields like before.

**LAB08\_C**

Text

Description automatically generated

This script add attributes to the the constructor ‘\_\_init\_\_’ in the class TrackInfo. Like when calling a function, the object’s attributes can be assigned when creating a new instance of the object. That’s how this script works, first by defining the attributes (or fields) in the class, and then calling the class with specific data for each attribute. The script can then print the values without them having to be individually assigned in separate lines.

LAB08\_D

Text

Description automatically generated

This script is essentially the same as before, except that the value for position, title, and length are run through what are essentially functions that test their validity. These ‘functions’ in this case are called properties, and they define what are valid entries for the attributes defined in the constructor. If invalid data is entered the .setter properties will raise exceptions.

LAB08\_D

Text

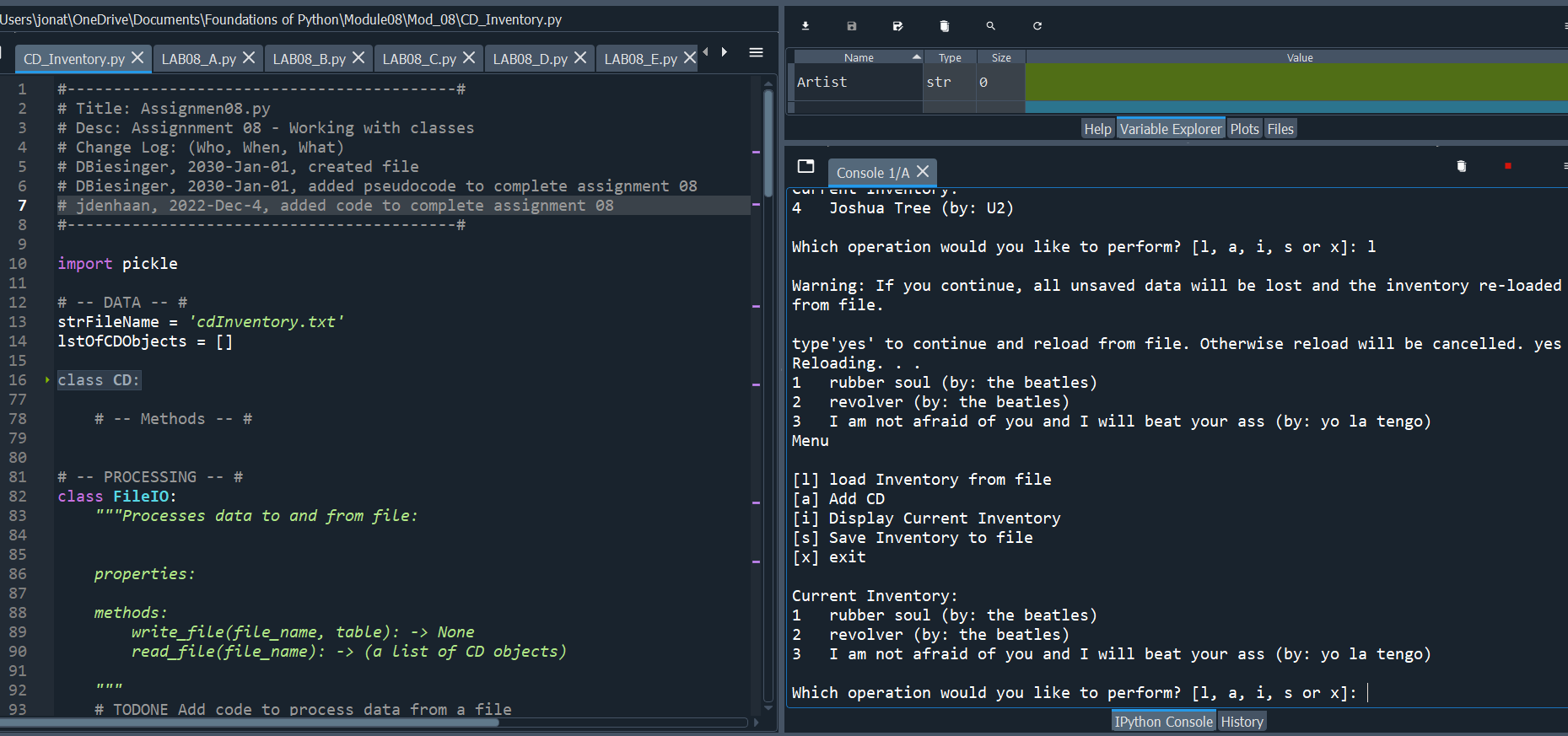
Description automatically generated

This script adds a method to the previous scripts. A method is a function that submits a reference to the object it is invoked on. In this case the normal str() method is replaced by overriding it with a function \_\_str\_\_ that itself calls another method. That method formats the data in such a way that it forms a complete sentence telling the user how long the song is:

Text

Description automatically generated

Assignment08



Text

Description automatically generated

This assignment was hard! I mostly struggled with how to call an object, instead of just relying on previous data structures. I began by creating an object class, and then worked to amend the scrip so that it would add and load data from file or user input into objects. I found that saving to binary file was more efficient since I didn’t have to tell the script to iterate through a list in order to create objects.

The error handling mostly works, but I’m a bit confused that it’s still possible to add blank CD objects, even though the CD class has error handling built in to prevent this. Maybe it’s because the error handling is too far removed from when the data is entered by the user? In my case, I ask the user for input, then pass their input to another variable as a tuple, and then add that tuple to a function that writes the data into an object. So maybe that’s to many steps removed, but in my mind it should still throw an error. I’d be interested to learn more and I’m sure I will!

**Summary:**

In this module we began learning about Object Oriented Programming. We used classes to define objects, wrote methods and created attributes for our objects, and used private attributes to restrict access to an objects attributes and validate entries.