Lesson 9 - OOP

Properties (No more getters and setters!)

Do you find getters and setters to be long to type and annoying to use? Why can’t I just access the attributes directly?

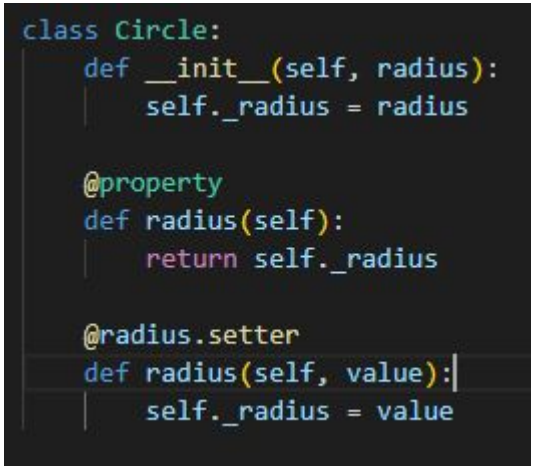
Properties

- Actually, you can. Welcome to properties.
- A property is a way to access your attributes so that it looks like you are simply getting an attribute, but underlying, it is actually using a getter and setter.
- This is how you would access a property from an instance of an object:

```
circle = Circle(42)
circle.radius
```

- That looks just like an attribute, no? Let's see what this is set up to do behind the scenes that makes it act like a getter and setter

- This is how you would add a property to your class:
- Let’s do it together.



The advantage

- Realize now that we can go back to writing all of our attributes like we used to. No need for getters and setters. No need for private (underscores) or for properties until the attributes require added functionality. Why?
- Let's illustrate with an example. If we have a class Person with an attribute ".name" and then 1 year later, there is a requirement to capitalize the first letter of a person's name, we can simply make ".name" into a property which will make sure to capitalize the first letter of name by using an underlying setter. Nobody who was using the ".name" attribute will have code that will break as a result of this change because both the attribute ".name" and the property ".name" look identical.

Practice

- Write a class called Rectangle with a property for length and a property for width. (It will be about 15 lines of code. You have 10 minutes).

Guidelines: When and how to use properties

- Use public attributes whenever appropriate, even if you expect the attribute to require functional behavior in the future. (yes, this contradicts what I told you about getters and setters).
- Avoid defining setter and getter methods for your attributes. You can always turn them into properties if needed.
- Use properties when you need to attach behavior to attributes and keep using them as regular attributes in your code.
- Avoid side effects in properties because no one would expect operations like assignments to cause any side effects.

Source: realpython.com

Be Aware

- Properties do have drawbacks. There are some cases that you will want to use a getter or setter and NOT a property. We will not go into it now, but you should be aware of it.

For a review of today's lesson, see the bottom half of https://realpython.com/python-getter-setter/

It will also deepen your understanding of when to use getters and setters vs when to use properties. Plus some of the drawbacks of properties.