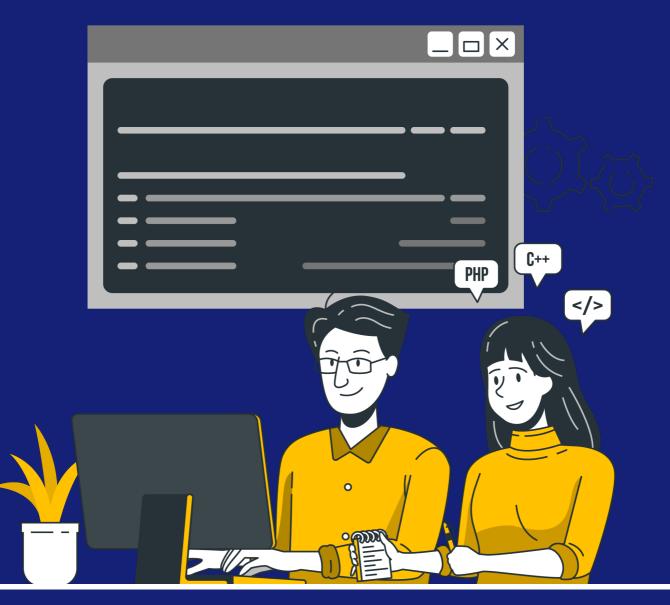


# **Lesson Plan**

# Property Decorators - Getters, Setters, And Deletes



### 1. Getter Method - @property:

Imagine you have a class representing a circle, and you want to calculate its area based on its radius. But you don't want users to directly access the radius attribute; instead, you want them to call a method that computes and returns the area. Here's how you can achieve that using @property:

```
class Circle:
    def __init__(self, radius):
        self._radius = radius

    @property
    def radius(self):
        return self._radius

    def area(self):
        return 3.14 * self._radius * self._radius

# Usage
circle = Circle(5)
print(circle.radius) # Access radius via getter
print(circle.area()) # Calculate area
```

In this example, @property decorator allows you to define a method (radius) that behaves like an attribute. So when you access circle.radius, it internally calls the radius method and returns the value.

# 2. Setter Method - @cproperty\_name.setter:

Now, suppose you also want to allow users to change the radius but with some validation, ensuring the radius is always positive. You can use the @cproperty\_name.setter decorator for that:

```
class Circle:
    def __init__(self, radius):
        self._radius = radius

    @property
    def radius(self):
        return self._radius

    @radius.setter
    def radius(self, value):
        if value \leq 0:
            raise ValueError("Radius must be positive.")
        self._radius = value

# Usage
circle = Circle(5)
circle.radius = 7 # Change radius
print(circle.radius) # Output: 7
```



Here, @radius.setter decorator allows you to define a method (radius) that will be called when you try to assign a value to circle.radius. It ensures that the new value meets certain criteria (in this case, it must be positive).

## 3. Deleter Method - @roperty\_name.deleter:

Lastly, let's say you want to allow users to delete the radius attribute altogether. Maybe you want to reset the circle. You can use @circle.deleter for this purpose:

Here, @radius.deleter decorator allows you to define a method (radius) that will be called when you try to delete circle.radius. In this example, it deletes the radius attribute.