# Getters and Setters

In this lesson, we will learn about getters and setters in OOP.

**We'll cover the following**


- Get and Set
- Example

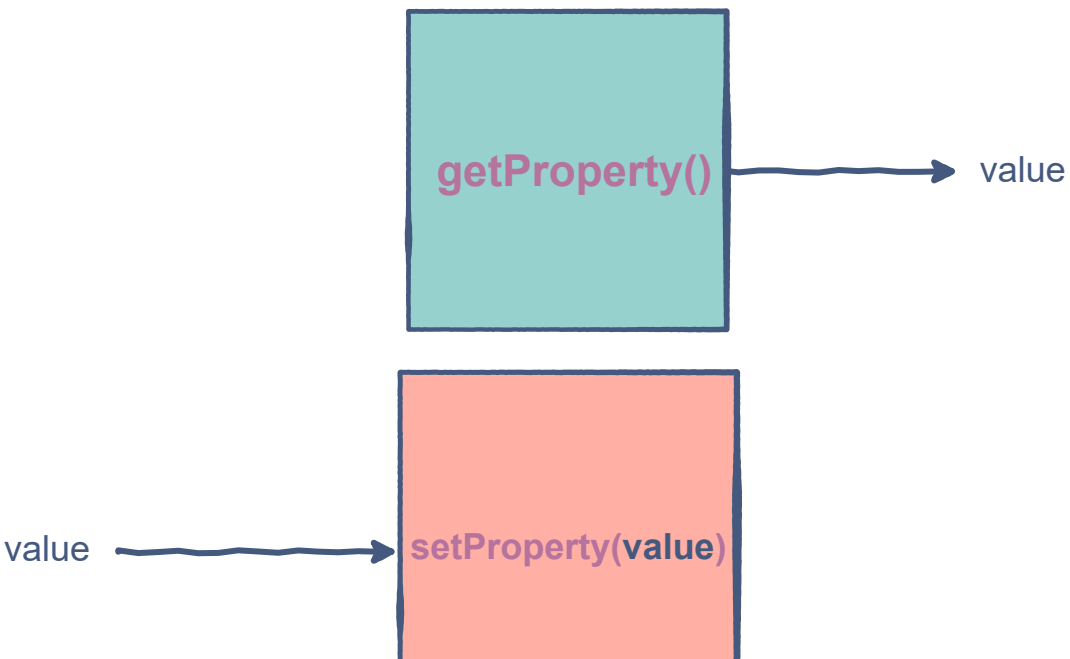

## Get and Set

In order to allow controlled access to properties from outside the class, getter and setter methods are used.

> A **getter** method allows reading a property's value.

> A **setter** method allows modifying a property's value.

It is a common convention to write the name of the corresponding member fields with the get or set command.



## Example #

Let's write get and set methods for `__username` in our `User` class:

```
class User():
    def __init__(self, username=None):  # defining initializer
        self.__username = username

    def setUsername(self, x):
        self.__username = x

    def getUsername(self):
        return (self.__username)


Steve = User('steve1')
print('Before setting:', Steve.getUsername())
Steve.setUsername('steve2')
print('After setting:', Steve.getUsername())
```

In the above class, `User`, we have defined a private property, named `__username`, which the main code cannot access. Also, note that we have started the name of this private property with `__`.

For this property to interact with any external environment, we have to use the get and set functions. The get function, `getUsername()`, returns the value of `__username` and the `setUsername(x)` sets the value of `__username` equal to the parameter `x` passed.

---

Now let's understand encapsulation using examples in our next lesson.