What is self?

'self' has a special place while working with classes in python

Python classes need a way to refer to themselves. This isn't some kind of narcissistic navel-gazing on the part of the class. Instead, it's a way to tell one instance from another. The word **self** is a way to describe itself, literally. Let's take a look at an example as I always find that helpful when I'm learning something new and strange:

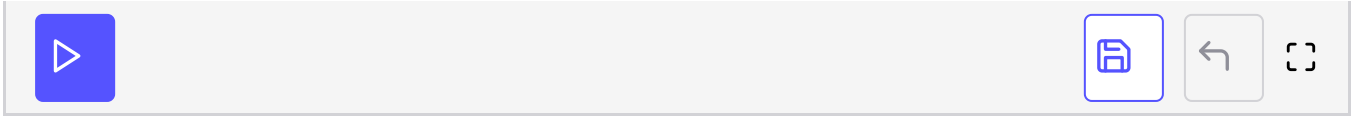
If you add dd the following code to the end of that class you wrote to create the Vehicle class:

```
# Assume that Vehicle classes is already defined

if __name__ == "__main__":
    car = Vehicle("blue", 5, 4)
    print(car.color)

    truck = Vehicle("red", 3, 6)
    print(truck.color)
```

The conditional statement above is a common way of telling Python that you only want to run the following code if this code is executed as a standalone file. If you had imported your module into another script, then the code underneath the conditional would not run. Anyway, if you run this code, you will create two instances of the Vehicle class: a car instance and a truck instance. Each instance will have its own attributes and methods. This is why when we print out the color of each instance, they are different. The reason is that the class is using that **self** argument to tell itself which is which. Let's change the class a bit to make the methods more unique:

```
class Vehicle(object):
    """docstring"""
```

```
    def __init__(self, color, doors, tires, vtype):
        """Constructor"""

        self.color = color
        self.doors = doors
        self.tires = tires
        self.vtype = vtype

    def brake(self):
        """
        Stop the car
        """
        return "%s braking" % self.vtype

    def drive(self):
        """
        Drive the car
        """
        return "I'm driving a %s %s!" % (self.color, self.vtype)

if __name__ == "__main__":
    car = Vehicle("blue", 5, 4, "car")
    print(car.brake())
    print(car.drive())

    truck = Vehicle("red", 3, 6, "truck")
    print(truck.drive())
    print(truck.brake())
```



In this example, we pass in another parameter to tell the class which vehicle type we're creating. Then we call each method for each instance. If you run this code, you should see the following output:

```
car braking
I'm driving a blue car!
I'm driving a red truck!
truck braking
```

This demonstrates how the instance keeps track of its "self". You will also notice that we are able to get the values of the attributes from the **__init__** method into the other methods. The reason is because all those attributes are prepended with **self.**. If we hadn't done that, the variables would have gone out of scope at the end of the **__init__** method.