Creating a Class

How to create a class in python?

Creating a class in Python is very easy. Here is a very simple example:

```
# Python 2.x syntax
class Vehicle(object):
    """docstring"""

    def __init__(self):
        """Constructor"""
        pass
```

This class doesn't do anything in particular, however it is a very good learning tool. For example, to create a class, we need to use Python's **class** keyword, followed by the name of the class. In Python, convention says that the class name should have the first letter capitalized. Next we have an open parentheses followed by the word **object** and a closed parentheses. The **object** is what the class is based on or inheriting from. This is known as the base class or parent class. Most classes in Python are based on **object**. Classes have a special method called **__init__** (for initialization). This method is called whenever you create (or instantiate) an object based on this class. The **__init__** method is only called once and is not to be called again inside the program. Another term for **__init__** is **constructor**, although this term isn't used that much in Python.

You may be wondering why I keep saying **method** instead of **function**. A function changes its name to "method" when it is within a class. You will also notice that every method has to have at least one argument (i.e. self), which is not true with a regular function.

In Python 3, we don't need to explicitly say we're inheriting from **object**. Instead, we could have written the above like this:

```
# Python 3.x syntax
```

```
class Vehicle:
    """docstring"""


    def __init__(self):
        """Constructor"""
        pass
```

You will notice that the only difference is that we no longer need the parentheses if we're basing our class on **object**. Let's expand our class definition a bit and give it some attributes and methods.

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

    def __init__(self, color, doors, tires):
        """Constructor"""
        self.color = color
        self.doors = doors
        self.tires = tires

    def brake(self):
        """
        Stop the car
        """
        return "Braking"

    def drive(self):
        """
        Drive the car
        """
        return "I'm driving!"
```

The code above added three attributes and two methods. The three attributes are:

```
self.color = color
self.doors = doors
self.tires = tires
```

Attributes describe the vehicle. So the vehicle has a color, some number of doors and some number of tires. It also has two methods. A method describes what a class does. So in this case, a vehicle can **brake** and **drive**. You may have noticed that all of the methods, including the first one have a funny argument called **self**. Let's talk about that!