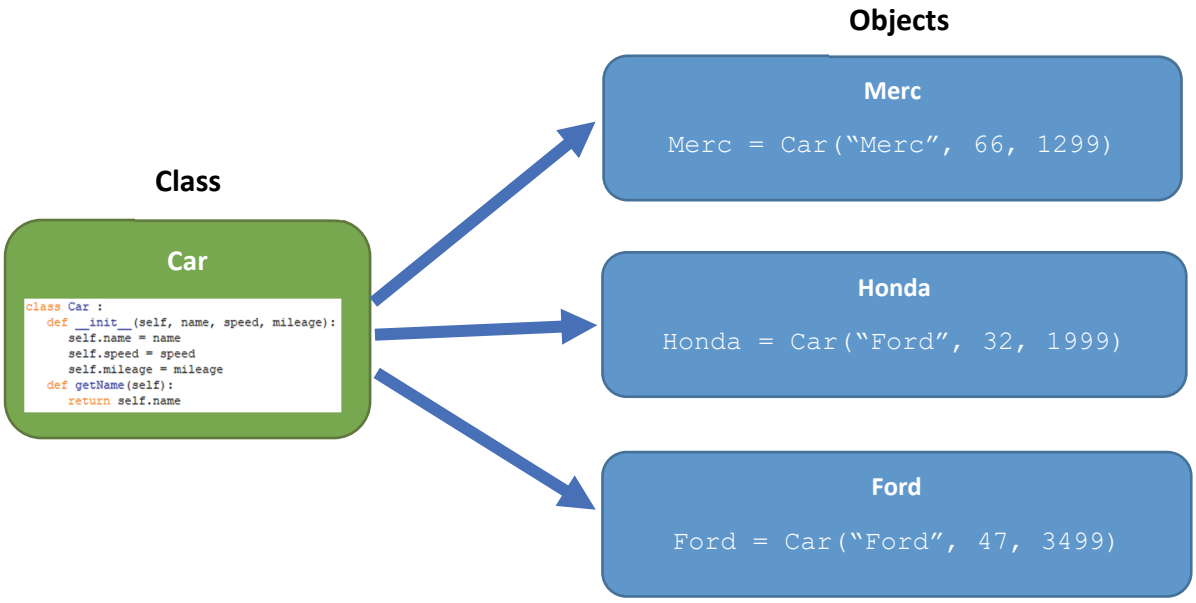
Classes and Objects

A class is a user-defined blueprint or template that defines the attributes (variables) and methods (functions), and contains them all in a single unit called a class. For example, `Car`.

```
class Car :
    def __init__(self, name, speed, mileage):
        self.name = name
        self.speed = speed
        self.mileage = mileage
    def getName(self):
        return self.name
```

An object is an instance of a class. So, for example you could use the `Car` class to create some objects such as `Merc`, `Honda`, and `Ford`.

`Merc` `= Car("Merc", 66, 1299)`



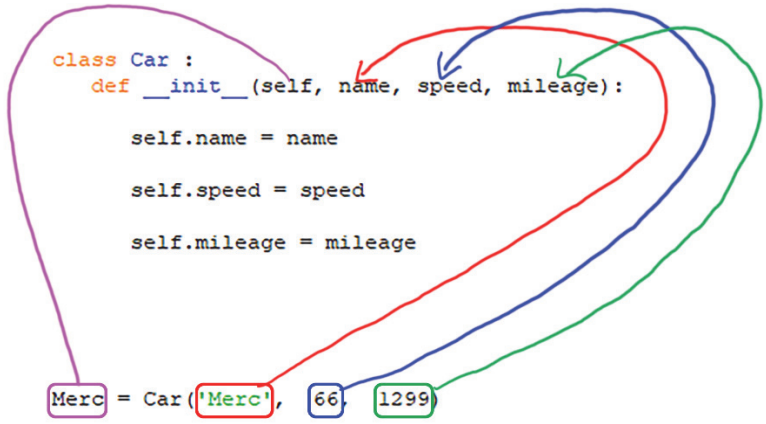
When an object is created, the Python interpreter automatically calls the `__init__()` method. This method is called a constructor and is used to initialise objects created with the class.

`def __init__(self, name, speed, mileage):`

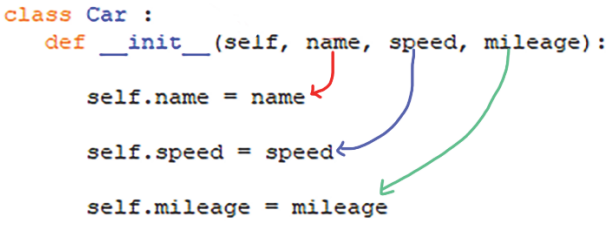
Within the `__init__()` method, we initialise the attributes.

```
self.name = name
self.speed = speed
self.mileage = mileage
```

Let's take a look at what is happening. In the `__init__()` method, `self` refers to the instance of the object itself and is automatically passed. Here at the bottom of the diagram below, when we create an object (Merc) from the Car class, we pass some data `name, speed, mileage`.



Next we assign the attributes passed to the `__init__()` method to the attributes of the object.



We can use the object of a class to perform actions. For example:

`Ford.`**`getName()`**

or

`Ford.`**`speed`** `= 66`