

CODE CAMP

Day 006: Classes and Objects

Dashboard

CPU Usage: 0% used – 0.00s of 100s. Resets in 23 hours, 15 minutes [More](#)

File storage: 0% full – 124.0 MB of 512.0 MB quota

Recent Consoles

+ 5 -

 [Bash console 15154103](#)

 [Bash console 15153805](#)

New console:

You can have up to 2 consoles. To get more, [upgrade your account!](#)

Recent Files

You have no recently edited files

[+ Open another file](#)



.001 Log In to Python Anywhere

Visit [PythonAnywhere.com](https://pythonanywhere.com) and log in to your account. You can click on one of the bash consoles you logged in to previously. They are listed under “Recent Consoles.”

If your command line is cluttered from your previous work you can type the **clear** command to clear the console visual history.



.001 Log In to Python Anywhere

If you've done everything correctly, your screen should appear similar to the image above. At your command prompt, type the command **nano** to open your text editor.

```
class Vehicle:
    def __init__(self, color, weight, numDoors,
topSpeed, currentSpeed):
        self.color = color
        self.weight = weight
        self.numDoors = numDoors
        self.topSpeed = topSpeed
        self.currentSpeed = currentSpeed
```

.002 Classes and Objects

Python is an object-oriented language.

Object orientation is a way of thinking about problems that breaks problem components in to classes and objects. Classes are, essentially, blueprints for objects that are part of the problem domain. Objects are instances of those classes, based on the blue prints.

The code on the left is a vehicle class, representing (simplistically) a vehicle blueprint in our code.

Here, we represent a vehicle by establishing several vehicle properties-- it's color, weight, top speed, etc.

```
class Vehicle:
    def __init__(self, color, weight, numDoors,
topSpeed, currentSpeed):
        self.color = color
        self.weight = weight
        self.numDoors = numDoors
        self.topSpeed = topSpeed
        self.currentSpeed = currentSpeed
```

.002 Classes and Objects

Key in the code on the left. The first line defines the class called **Vehicle**.


The function that appears in the **Vehicle** class is called the constructor. It runs each time a new instance of this vehicle class is created using this blueprint. (We haven't created any objects yet.)

The constructor, quite literally, constructs the object. In Python the constructor is always named `__init__`.

Let's create an object, which will clarify how the constructor works.

```
class Vehicle:
    def __init__(self, make, model, color,
weight, numDoors, topSpeed, currentSpeed):
        self.make = make
        self.model = model
        self.color = color
        self.weight = weight
        self.numDoors = numDoors
        self.topSpeed = topSpeed
        self.currentSpeed = currentSpeed

markCar = Vehicle("Nissan", "Rogue", "Silver",
4200, 5, 110, 0)
```



.002 Classes and Objects

Note that we've added two properties to our **Vehicle** class. **make** and **model** are now two properties that are tracked in this class.

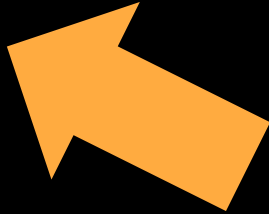
It is the final line of code that creates an instance of the **Vehicle** class. This process is known as instantiation.

You can see that both string values and numerical values are passed in to the class constructor.

```
class Vehicle:
    def __init__(self, make, model, color,
weight, numDoors, topSpeed, currentSpeed):
        self.make = make
        self.model = model
        self.color = color
        self.weight = weight
        self.numDoors = numDoors
        self.topSpeed = topSpeed
        self.currentSpeed = currentSpeed

markCar = Vehicle("Nissan", "Rogue", "Silver" ,
4200, 5, 110, 0)

print(markCar.make + " " + markCar.model)
```



.002 Classes and Objects

We can access all of the properties of the **markCar** object using the dot notation as demonstrated here.

We can use the same notation to change the value of a property that is part of the **markCar** object as in:

markCar.currentSpeed = 50 or
markCar.color = "black"

.002 Classes and Objects

```
class Vehicle:
    def __init__(self, make, model, color,
weight, numDoors, topSpeed, currentSpeed):
        self.make = make
        self.model = model
        self.color = color
        self.weight = weight
        self.numDoors = numDoors
        self.topSpeed = topSpeed
        self.currentSpeed = currentSpeed
```

```
markCar = Vehicle("Nissan", "Rogue", "Silver" ,
4200, 5, 110, 0)
```

```
rint(markCar.make + " " + markCar.model)
markCar.currentSpeed = 50
markCar.color = "black"
print("Speed: " + str(markCar.currentSpeed))
```



Bash console 15153805

```
20:14 ~ $ python Vehicle.py
Nissan Rogue
Speed: 50
20:14 ~ $
```



```
class Vehicle:
    def __init__(self, make, model, color,
weight, numDoors, topSpeed, currentSpeed):
        self.make = make
        self.model = model
        self.color = color
        self.weight = weight
        self.numDoors = numDoors
        self.topSpeed = topSpeed
        self.currentSpeed = currentSpeed

markCar = Vehicle("Nissan", "Rogue", "Silver" ,
4200, 5, 110, 0)
momCar = Vehicle("Hyundai" , "Tucson" , "Aqua" ,
3900, 5, 120, 0)

print(markCar.make + " " + markCar.model)
print(momCar.make + " " + momCar.model)

markCar.currentSpeed = 50
markCar.color = "black"
print("Speed: " + str(markCar.currentSpeed))
```

.002 Classes and Objects

The code has been altered so that a second object based on the **Vehicle** class has been created.

Remember that a class is merely a blueprint for a real object. You can have many instances of a class active in a program at once.





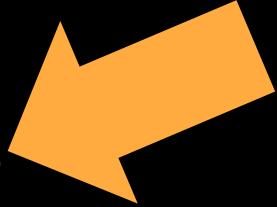
.003 Activity: The Object Oriented Coke Machine

Vending machines are something we take for granted. Let's say you needed to code the functions of a soda vending machine in Python. What properties would be part of the class representing the machine?

Make a list of the classes you can think of and post your list to the code camp discussion.

```
class Vehicle:
    def __init__(self, make, model, color, weight, numDoors, topSpeed, currentSpeed):
        self.make = make
        self.model = model
        self.color = color
        self.weight = weight
        self.numDoors = numDoors
        self.topSpeed = topSpeed
        self.currentSpeed = currentSpeed

    def accelerate(self):
        if self.currentSpeed < self.topSpeed:
            self.currentSpeed = self.currentSpeed + 5
        return self.currentSpeed
```



.004 Object Methods

Object methods are functions that are part of your class design. In the **Vehicle** class we've added a class method called **accelerate**. This method checks to see if the **self.currentSpeed** is less than **self.topSeed**, and, if so, increases **self.currentSpeed**. Note that **self** is self-referential and it means "in this instance of the class."

Often a class has multiple methods associated with it.

```
class Vehicle:
    def __init__(self, make, model, color, weight, numDoors, topSpeed, currentSpeed):
        self.make = make
        self.model = model
        self.color = color
        self.weight = weight
        self.numDoors = numDoors
        self.topSpeed = topSpeed
        self.currentSpeed = currentSpeed
    def accelerate(self):
        if self.currentSpeed < self.topSpeed:
            self.currentSpeed = self.currentSpeed + 5
        return self.currentSpeed

    def brake(self):
        if self.currentSpeed > 0:
            self.currentSpeed = self.currentSpeed - 5
        if self.currentSpeed < 0:
            self.currentSpeed = 0
        return self.currentSpeed

markCar = Vehicle("Nissan", "Rogue", "Silver", 4200, 5, 110, 0)
markCar.accelerate()
markCar.accelerate()
print("The vehicle is now moving at " + str(markCar.currentSpeed) + " MPH.")
```



.004 Object Methods

Key in the code for your class with the added methods and object declarations. You might notice that object methods appear to expect a parameter called **self**. This is because Python implicitly passes the object to method calls, but you need to explicitly declare the parameter for it.

How would you modify the code just completed to add a new object called **golfCart**. It has the following properties:

```
make = Melex      currentSpeed = 0  
model = EC21  
color = white  
weight = 800  
numDoors = 0  
topSpeed = 25
```




.004 Object Methods

Write the necessary code to create your **golfCart** instance of the **Vehicle** class.

Use the **accelerate** method until you've reached to top speed of **golfCart**.

Modify the **accelerate()** method so that when the top speed is reached, it prints out "Can't go faster. Top speed reached."



.005 Activity: The Object Oriented Coke Machine

Think about your object oriented soda vending machine again.

Which methods you would add to your class in order for a user to operate the machine.

Do the methods you are adding change the properties list that you wrote earlier?