# Procedural Programming

*Sometimes called Imperative Programming*

There is data. (state)

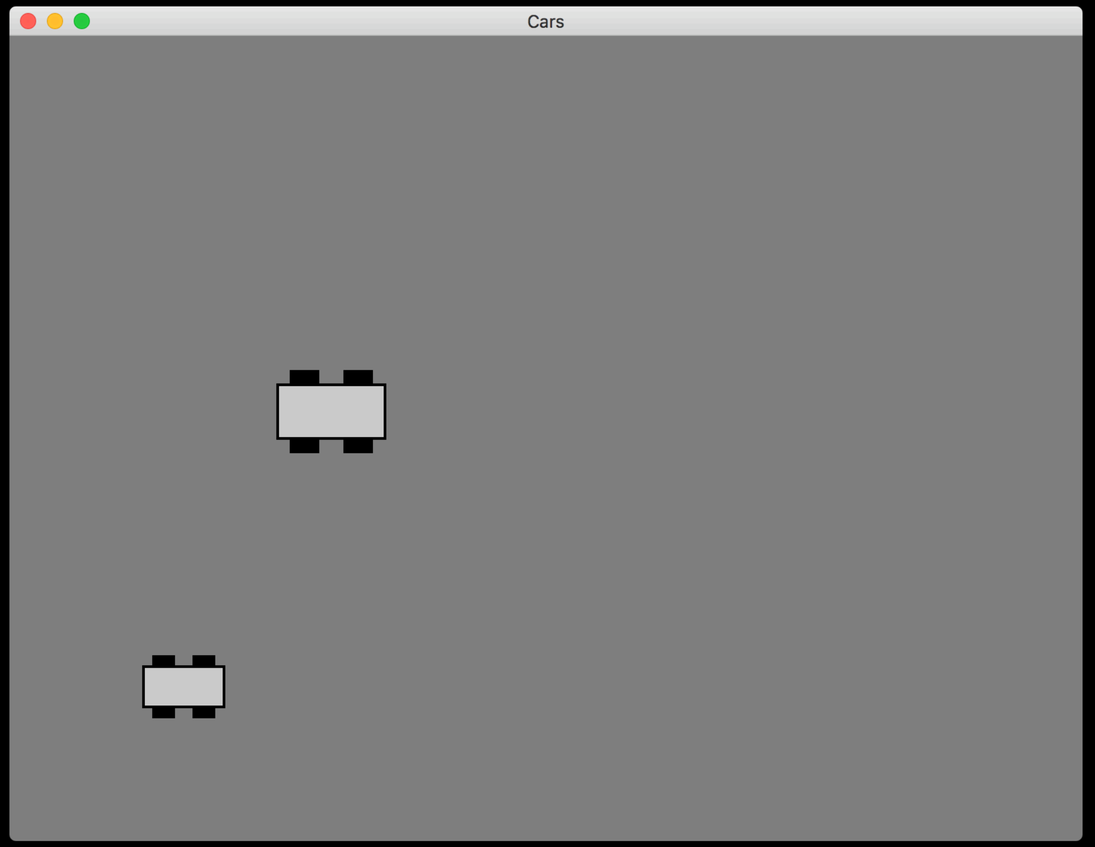
There are instructions that change the program's state.

Instructions can be organized into functions.

# Object Oriented Programming

A programming paradigm that organizes a relationship between certain functions and data.

Consider the 2d car demo program from our discussion about functions.




## Functions

There is a function, with parameters, that can render a car. It uses the input parameters to locate where to draw the car, and how large it should be.

```
func drawCar(x: Double, y: Double, size: Double) {
   // drawing instructions..
}
```

## Data

There are a set of variables for each car storing position, size, and velocity.

```
var car1x = 200.0
var car1y = 300.0
var car1size = 80.0

var car2x = 100.0
var car2y = 60.0
var car2size = 60.0
```

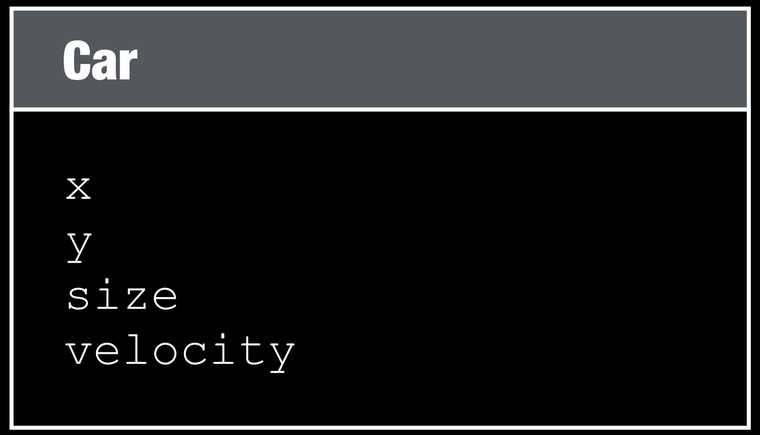
All of the variables that we use to represent a car are actually independent.

```
var car1x
var car1y
var car1size
var car1velocity
```

All of the variables that we use to represent a car are actually independent.

```
var car1x
var car1y
var car1size
var car1velocity
```

It would be good to work with all the pieces of data for one car together, as one unit.



# Swift and Object Oriented Programming

Classes and structures are general-purpose, flexible constructs that become the building blocks of your program's code. Using classes and structures you can create new Types to use in in your program.

You define properties and methods to add functionality to your classes and structures by using exactly the same syntax as for constants, variables, and functions.

# Swift and Object Oriented Programming

A class or structure is a template that allows you to create multiple instances of a type.

An instance is a specific occurrence of the type in a running program.

The terms object and instance are often used interchangeably.

# Swift and Object Oriented Programming

Define properties to store values. Same syntax as constants and variables.

Define methods to provide functionality. Same syntax as functions.

# Class

The basic syntax for defining a class. The keyword class followed by the name, then a block of code.

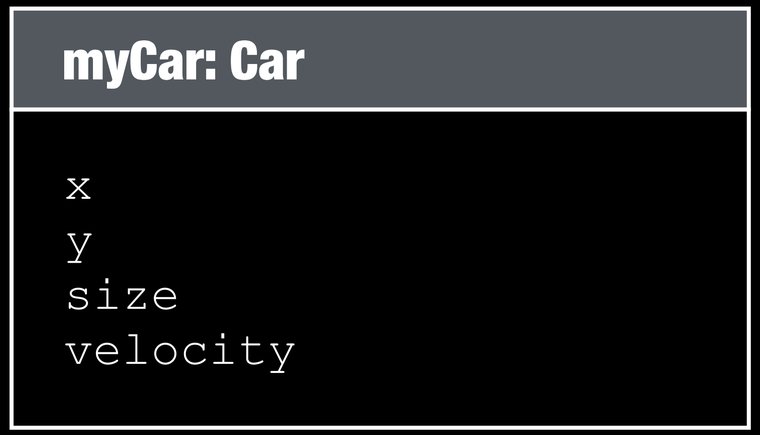
```
class ClassName {

  // The components of a class are declared inside the
  // body of the class.

}
```

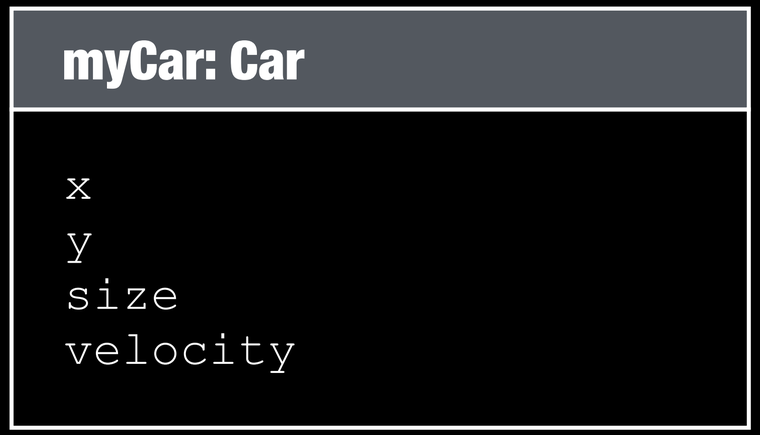
# Creating an instance

```
var myCar: Car = Car()
```

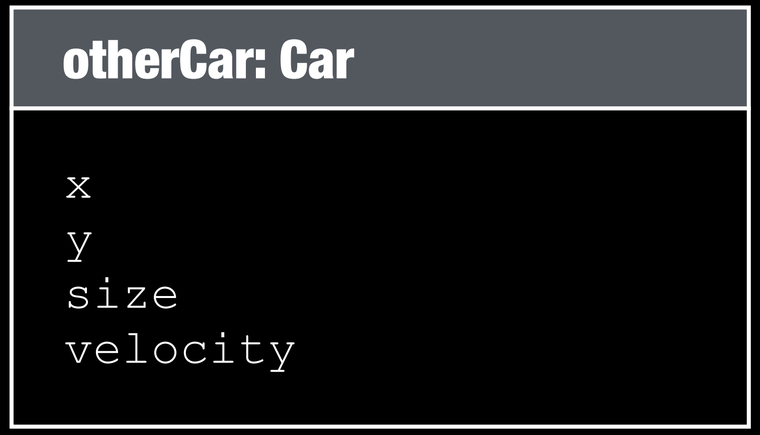



myCar is a new variable - of type Car. Here we call the Car initializer to make a new Car instance (or object). This new car object has its own set of properties.

```
var myCar: Car = Car()
```




```
var otherCar: Car = Car()
```




Making two Car objects - each object has its own properties for storing values.

# Properties

```
class Car {
   var x = 100.0
   var y = 100.0
   var size = 64.0
   var velocity = 2.0
}
```

# Properties

```
class Car {
   var x = 100.0
   var y = 100.0
   var size = 64.0
   var velocity = 2.0
}
```

Properties (and methods) can be accessed using “dot syntax.” Given an instance, the property is accessed by adding a dot, then the name of the property.

```
// Set the value of a property
myCar.x = 231.0

// Get the value of a property
myCar.x
```

# Methods

```
class Car {
   var x = 100.0
   var y = 100.0
   var size = 64.0
   var velocity = 2.0

   func move() {
      x = x + velocity
   }

   func render() {
      // drawing instructions here..
   }
}
```

Calling a method. To call an instance method of a class, you need an instance (ie. an object) thru which to call the method. For example, if you have a variable myCar, which references an instance of a Car, you can call the move or render methods.

```
myCar.move()
myCar.render()
```

It isn't possible to call the move or render method without an instance of a Car object.

# Methods

```
class Car {
   var x = 100.0
   var y = 100.0
   var size = 64.0
   var velocity = 2.0

   func move() {
      x = x + velocity
   }

   func render() {
      // drawing instructions here..
   }
}
```

Notice the instruction in the body of the move method. Methods can access property values of the instance that called them. The move method uses the x and velocity properties.