A Tour of Go

There are two reasons to use a pointer receiver.

The first is so that the method can modify the value that its receiver points to.

The second is to avoid copying the value on each method call. This can be more efficient if the receiver is a large struct, for example.

In this example, both `Scale` and `Abs` are with receiver type `*Vertex`, even though the `Abs` method needn't modify its receiver.

In general, all methods on a given type should have either value or pointer receivers, but not a mixture of both. (We'll see why over the next few pages.)

methods-with-pointer-receivers.go

Imports off Syntax off

```
1  package main
2
3  import (
4      "fmt"
5      "math"
6  )
7
8  type Vertex struct {
9      X, Y float64
10 }
11
12 func (v *Vertex) Scale(f float64) {
13     v.X = v.X * f
14     v.Y = v.Y * f
15 }
16
17 func (v *Vertex) Abs() float64 {
18     return math.Sqrt(v.X*v.X + v.Y*v.Y)
19 }
20
21 func main() {
22     v := &Vertex{3, 4}
23     fmt.Printf("Before scaling: %+v, Abs: %v\n", v, v.Abs())
24     v.Scale(5)
25     fmt.Printf("After scaling: %+v, Abs: %v\n", v, v.Abs())
26 }
```

A Tour of Go