# More on Go, Stocks

## Goroutines

Go routes allow you to create parallel running code.

```
 1: package main
 2:
 3: import (
 4:     "fmt"
 5:     "sync"
 6: )
 7:
 8: var wg sync.WaitGroup
 9:
10: func f(from string) {
11:     wg.Add(1)
12:     defer wg.Done()
13:     for i := 0; i < 3; i++ {
14:         fmt.Printf("%s: %v\n", from, i)
15:     }
16: }
17:
18: func main() {
19:
20:     f("direct")
21:
22:     go f("goroutine")
23:
24:     for i := 0; i < 10; i++ {
25:         wg.Add(1)
26:         go func(msg string) {
27:             defer wg.Done()
28:             fmt.Printf("%s\n", msg)
29:         }(fmt.Sprintf(" I am %d ", i))
30:     }
31:
32:     wg.Wait()
33:     fmt.Printf("All Done\n")
34: }
```

## Go Interfaces

Two uses for interfaces (Actually more than 2 but 2 primary uses).

1. Variable parameter list functions.
2. Interfaces to sets of functions.

## Variable parameter list functions.

Also an example of reflection.

```
 1: package main
 2:
 3: import "fmt"
 4:
 5: func vexample(a int, b ...interface{}) {
 6:     for pos, bVal := range b {
 7:         switch v := bVal.(type) {
 8:         case int:
 9:             fmt.Printf("It's an int, %d at %d\n", v, pos)
10:         case []int:
11:             fmt.Printf("It's a slice of int\n")
12:         default:
13:             fmt.Printf("It's a something else\n")
14:         }
15:     }
16: }
```

## Interfaces to sets of functions.

```
 1: package main
 2:
 3: type InterfaceSpecType interface {
 4:     DoFirstThing(p1 int, p2 int) error
 5:     DoSomethingElse() error
 6: }
 7:
 8: type InterfaceOtherType interface {
 9:     DoSomethingElse() error
10:     DoSomethingSpecial(in int) error
11: }
12:
13: type ImplementationType struct {
14:     AA int
15:     BB int
16: }
17:
18: // Verify at compile time that the implementation type
19: // is a valid implementation of the interface.
20: var _ InterfaceSpecType = (*ImplementationType)(nil)
21:
22: // Validate 2nd interface spec.
23: var _ InterfaceOtherType = (*ImplementationType)(nil)
24:
25: func NewImplementationType() InterfaceSpecType {
26:     return &ImplementationType{
27:         AA: 1,
28:         BB: 2,
29:     }
30: }
31:
32: func (xy *ImplementationType) DoFirstThing(p1 int, p2 int) error {
33:     // ... do something ...
34:     return nil
35: }
36:
37: func (xy *ImplementationType) DoSomethingElse() error {
38:     // ... do something ...
39:     return nil
40: }
41:
```

```
42: func (xy *ImplementationType) DoSomethingSpecial(in int) error {
43:     // ... do something ...
44:     return nil
45: }
46:
47: func Demo() {
48:     var dd InterfaceSpecType
49:     dd = NewImplementationType()
50:     _ = dd.DoSomethingElse()
51: }
52:
53: func main() {
54:     Demo()
55: }
```

## Go Channels

We will come back to this later.

## Go Weaknesses

What are the limitations of using Go

1. No objects - Use interfaces instead. No inheritance.
2. Executables are big

# Stock Stuff

What is a Stock?

What is a Dividend?

Wyoming Laws on Stocks.

What is a Bond? What is a Fixed Coupon v.s. a Variable Capon?

What is Yield?

How are dividends payed?

Other Investments (Gold, Diamonds, Houses, Apartments)