Week 3

Common Mistakes in Golang

Agenda

1. Function literal
2. Nil interface
3. Slice
4. Float
5. Regexp

1. Function literal

```go
func main() {
	var functions []func()

	for i := 0; i < 10; i++ {
		functions = append(functions, func() {
			fmt.Println(i)
		})
	}

	for _, f := range functions {
		f()
	} // what & why?
}
```

```go
func main() {
	var wg sync.WaitGroup

	for _, i := range []int{-1,2,3,4} {
		wg.Add(1)
		go func() {
			defer wg.Done()
			fmt.Println(i)
		}()
	}

	wg.Wait() // what & why?
}
```

1. Function literal (ctn)

How to fix

```go
func main() {
	var functions []func()

	for i := 0; i < 10; i++ {
		j := i
		functions = append(functions, func() {
			fmt.Println(j)
		})
	}

	for _, f := range functions {
		f()
	} // what & why?
}
```

```go
func main() {
	var wg sync.WaitGroup

	for _, i := range []int{-1,2,3,4} {
		i := i
		wg.Add(1)
		go func() {
			defer wg.Done()
			fmt.Println(i)
		}()
	}

	wg.Wait() // what & why?
}
```

2. Nil interface

```
type NonZeroError struct {}

func(e *NonZeroError) Error() string {
        return "This is NOT Zero"
}

func checkZero(i int) error {
        var err *NonZeroError
        if i!=0 {
                err = &NonZeroError{}
        }
         return err
    }
}

func main() {
    if err := checkZero(0); err == nil {
         fmt.Println("This is Zero")
    } else {
        fmt.Println(err.Error())
    } // what & why?
}
```

3. Nil interface (ctn)

How interface variable is stored

```go
func main() {
	var i interface{}
	describe(i) // (<nil>, <nil>)

	i = 42
	describe(i) // (42, int)

	i = "hello"
	describe(i) // (hello, string)

	var tmpInt *int = nil
	i = tmpInt
	describe(i) // (<nil>, *int)
}

func describe(i interface{}) {
	fmt.Printf("(%v, %T)\n", i, i)
}
```

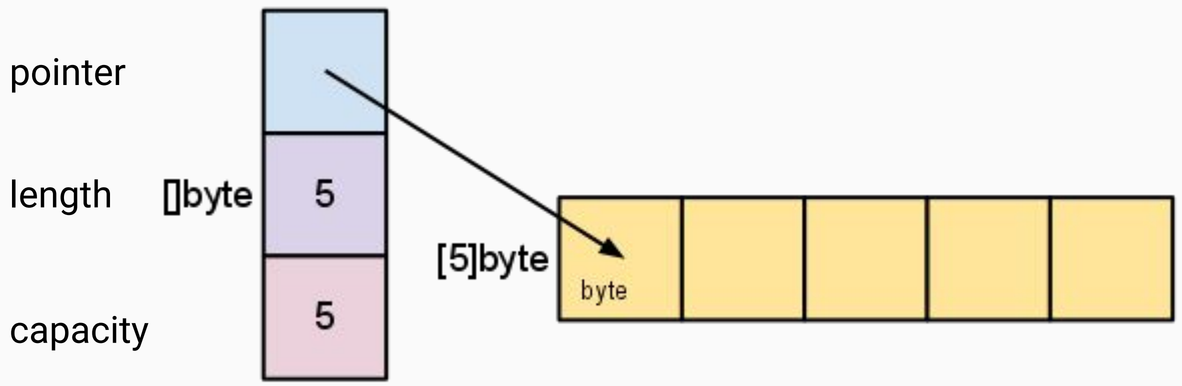
3. Slice

Example:

```
func main() {
        s1 := make([]string, 1, 2)
        s1[0] = "a"
        _ = append(s1, "b")
        s2 := s1[:2]

        s3 := make([]string, 1, 1)
        s3[0] = "a"
        _ = append(s3, "b")
        s4 := s3[:2]

        fmt.Println(s1) //  what and why?
        fmt.Println(s2) //
        fmt.Println(s4) //
}
```

3. Slice (ctn)

How slice variable is stored



4. Float

```go
package main

import (
	"fmt"
)

func main() {
	fmt.Println(float64(4.1)*100)
	fmt.Println(float64(0.7) + float64(0.21) == float64(0.91)) // what & why?
}
```

4. Float (ctn)

How to fix?

```
package main

import (
	"Fmt"
	"github.com/shopspring/decimal"
)

type MyNumber struct {
	Value int64
	Exp int64
}
```

5. Regexp

```
var data = []byte(`Jan 18 06:41:30 corecompute sshd[42327]: Failed keyboard-interactive/pam`)

var hits int

func main() {
	var localReSSHD = regexp.MustCompile(`sshd\[\d{5}\]:\s*Failed`)

	if localReSSHD.Match(data) {
		hits++
	} // what is the problem and why?
}
```

5. Regexp (ctn)

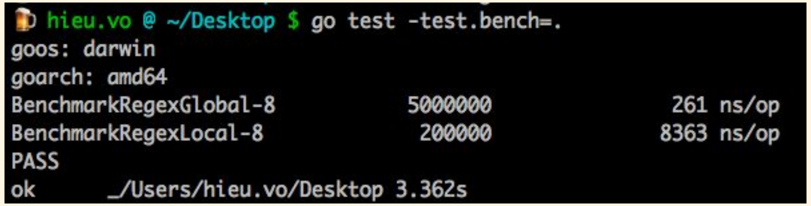
How to fix:

```
var data = []byte(`Jan 18 06:41:30 corecompute sshd[42327]: Failed keyboard-interactive/pam`)

var hits int
var localReSSHD = regexp.MustCompile(`sshd\[\d{5}\]:\s*Failed`)

func main() {
	if localReSSHD.Match(data) {
		hits++
	}
}
```

5. Regexp (ctn)

Try to benchmark

```
package main

import (
    "regexp"
    "testing"
)

var data = []byte(`Jan 18 06:41:30 corecompute sshd[42327]: Failed keyboard-interactive/pam`)

var hits int
var reSSHD = regexp.MustCompile(`sshd\[\d{5}\]:\s*Failed`)

func BenchmarkRegexGlobal(b *testing.B) {
    for i := 0; i < b.N; i++ {
        if reSSHD.Match(data) {
            hits++
        }
    }
}

func BenchmarkRegexLocal(b *testing.B) {
    for i := 0; i < b.N; i++ {
        var localReSSHD = regexp.MustCompile(`sshd\[\d{5}\]:\s*Failed`)
        if localReSSHD.Match(data) {
            hits++
        }
    }
}
```



```
hieu.vo @ ~/Desktop $ go test -test.bench=.
goos: darwin
goarch: amd64
BenchmarkRegexGlobal-8          5000000               261 ns/op
BenchmarkRegexLocal-8            200000              8363 ns/op
PASS
ok      _/Users/hieu.vo/Desktop 3.362s
```

Thanks