Common gotchas

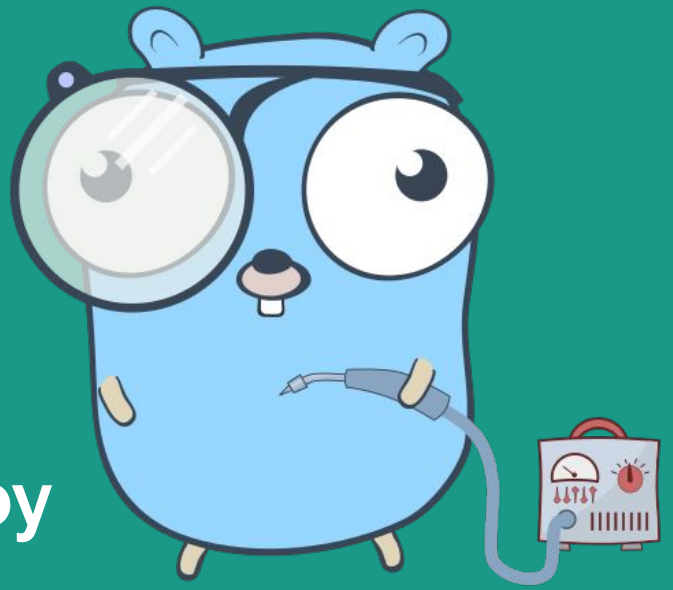
Ilija Matoski

 https://matoski.com

 https://linkedin.com/ilijamt

 https://github.com/ilijamt

- When nil is not nil
- Arrays won't change
- Copy?
- Map iterations
- Loop variable captured by func literal



When nil is not nil

```
type obj struct {
        Err  error
}

func (o *obj) Error() string {
        return o.Err.Error()
}

func a() error {
        var err *obj = nil
        return err
}

func b() (err error) {
        return err
}
```

```
err := a()
fmt.Println(err)                    // nil
fmt.Println(err == nil)             // false
fmt.Println(err == (*obj)(nil))     // true


err := b()
fmt.Println( err)                   // nil
fmt.Println(err == nil)             // true
```

Arrays won't change

```
func change_array(a [2]int) {
        a[0] = 8
}

func change_slice(a []int) {
        if len(a) > 0 {
                a[0] = 8
        }
}
```

```
a := [2]int{1, 2}
change_array(a)
fmt.Println( a)                 // [1 2]

a := [2]int{1, 2}
change_slice(a)
fmt.Println( a)                 // [8 2]
```

Copy?

```
func main() {
        var src, dst []int
        src = []int{1, 2, 3}
        copy(dst, src)
        fmt.Println(dst)      // []
}
```

```
func main() {
        var src, dst []int
        src = []int{1, 2, 3}
        dst = make([]int, len(src))
        copy(dst, src)
        fmt.Println(dst)      // [1 2 3]
}
```

```
func main() {
        var src, dst []int
        src = []int{1, 2, 3}
        copy(dst, src)
        fmt.Println(dst)      // [1 2 3]
}
```

Map iterations are not deterministic

```
m := map[string]int{"one": 1, "two": 2, "three": 3, "four": 4}

for k, v := range m {
        fmt.Println(k, v)
}
```

```
three 3
four 4
one 1
two 2
```

```
two 2
three 3
four 4
one 1
```

```
one 1
two 2
three 3
four 4
```

```
three 3
four 4
two 2
one 1
```

Loop variable captured by func literal

```
func main() {
	wg := sync.WaitGroup{}

	wg.Add(10)
	for i := 0; i < 10; i++ {
		go func() {
			defer wg.Done()
			fmt.Println(i)
		}()
	}

	wg.Wait()
}

// go build or go run don't complain.
// go vet says:
// ./prog.go:15:16: loop variable i captured by func literal
```

```
$ go run main.go

10
10
10
8
10
10
10
10
10
10
```

Loop variable captured by func literal

```
func main() {
	wg := sync.WaitGroup{}

	wg.Add(10)
	for i := 0; i < 10; i++ {
		go func(i int) {
			defer wg.Done()
			fmt.Println(i)
		}(i)
	}

	wg.Wait()
}
```

```
$ go run main.go

9
4
0
1
2
3
6
5
7
8
```

Questions