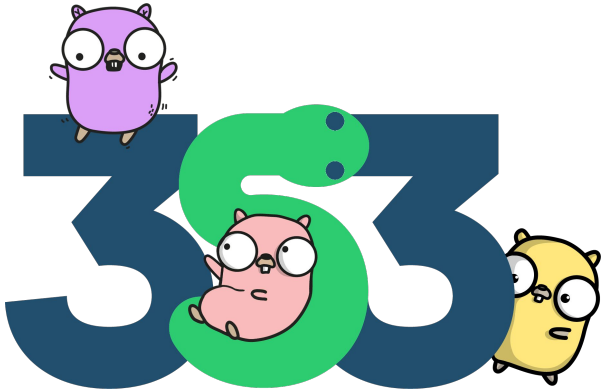
In the Loop

miki @tebeka
miki@353solutions.com

CEO, CTO, UFO ...



the boring one

```
total := 0
for i := 0; i < 1000; i++ {
    if i%3 == 0 || i%5 == 0 {
        total += i
    }
}
fmt.Println(total)
```

two variables

```
func isPalindrome(s string) bool {
    for i, j := 0, len(s)-1; i < j; i, j = i+1, j-1 {
        if s[i] != s[j] {
            return false
        }
    }
    return true

}
```

while

```
total, a, b := 0, 1, 1
 for a <= 4_000_000 {
    if a%2 == 0 {
        total += a
    }
    a, b = b, a+b
}
fmt.Println(total)
```

forever

```
func handler(p Provider) {
    for {
        msg := p.Next()
        if msg == nil {
            break
        }
        fmt.Printf("%+v\n", msg)
    }
}
```

range ∴ slice

```go
cart := []string{"bread", "butter", "beer"}
// indices
for i := range cart {
    fmt.Println(i)
}
// index + value
for i, v := range cart {
    fmt.Println(i, v)
}
// values
for _, v := range cart {
    fmt.Println(v)
}
```

range ∴ values

```
var players = []struct {
    name   string
    points int
}{
    {"Rick", 1_000_000},
    {"Morty", 13},
}
➤ for _, player := range players {
    player.points += 353
}
fmt.Printf("%v\n", players)
// [{Rick 1000000} {Morty 13}]
```

range ∴ reference

```
var players = []struct {
    name   string
    points int
}{
    {"Rick", 1_000_000},
    {"Morty", 13},
}
for i := range players {
    players[i].points += 353
}
fmt.Printf("%v\n", players)
// [{Rick 1000353} {Morty 366}]
```

range ∴ map

Same as slices ☺️

range ∴ channel

```
ch := make(chan int)
go func() {
    for i := 0; i < 3; i++ {
        ch <- i
    }
    close(ch)
}()

➤   for v := range ch {
        fmt.Println(v)
    }
```

range ∴ nothing

```
// fan out
ch := make(chan reply)
for _, s := range vs {
    s := s
    go func() {
        ch <- reply{s, isPalindrome(s)}
    }()
}
// collect
for range vs {
    r := <-ch
    fmt.Printf("%-5s -> %v\n", r.s, r.isP)
}
```

closure

```
// fan out
ch := make(chan reply)
for _, s := range vs {
    s := s
    go func() {
        ch <- reply{s, isPalindrome(s)}
    }()
}
```

See redefining for loop variable semantics by Russ Cox.

nested

```
hasEven := false
for r := range mat {
    for c := range mat[0] {
        if mat[r][c]%2 == 0 {
            hasEven = true
            fmt.Println("found")
            break
        }
    }
}
fmt.Println("has even:", hasEven)
```

nested ∴ fix

```
    hasEven := false
loop:
    for r := range mat {
        for c := range mat[0] {
            if mat[r][c]%2 == 0 {
                hasEven = true
                fmt.Println("found")
➤                break loop
            }
        }
    }
    fmt.Println("has even:", hasEven)
```

goto

https://xkcd.com/292/

standard library

```
$ ag --vimgrep -s --go 'goto\s+' ~/sdk/go1.19.5/src | \
    grep -v testdata  | \
    grep -v _test.go | \
    wc -l
610
```

nested ∴ goto

```
    hasEven := false
    for r := range mat {
        for c := range mat[0] {
            if mat[r][c]%2 == 0 {
                hasEven = true
                fmt.Println("found")
➤                goto found
            }
        }
    }
found:
    fmt.Println("has even:", hasEven)
```

iteration ∴ scanner

```
lnum := 0
s := bufio.NewScanner(r)
for s.Scan() {
    lnum++
    if strings.Contains(s.Text(), term) {
        fmt.Printf("%d: %s\n", lnum, s.Text())
    }
}
if err := s.Err(); err != nil {
    log.Fatalf("error: %s", err)
}
```

See discussion: standard iterator interface by Ian Lance Taylor

Thank You

miki @tebeka
miki@353solutions.com