

Go Control Structures

WASA: Web and Software Architecture

Prof. Emanuele Panizzi

```
package main
import "fmt"
func main() {
    for i := 0; i <= 5; i++ {
        fmt.Println(i)
    }
    sum := 1
    for sum < 8 { // this is like a while
        sum += sum
    }
    fmt.Println(sum)
}
```

- can you make an infinite loop?

```
package main
import "fmt"

func fact(n int) int {
    if n <= 1 {
        return 1
    } else {
        return n * fact(n-1)
    }
}

func main() {
    fmt.Println(fact(5))
}
```

If (contd)

```
// ...  
func pow(x, n, lim float64) float64 {  
    if v := math.Pow(x, n); v < lim {  
        return v  
    } else if v > 2 * lim {  
        return 99999 - v  
    }  
    // can't use v here  
    return lim  
}  
func main() {  
    fmt.Println(pow(3,2,10), pow(3,3,20), pow(3,4,20))  
    // 9 20 99918  
}
```

Switch

```
package main
import ("fmt"; "time")
func main() {
    fmt.Println("When's Saturday?")
    today := time.Now().Weekday()
    switch time.Saturday {
    case today + 0:
        fmt.Println("Today.")
    case today + 1:
        fmt.Println("Tomorrow.")
    default:
        fmt.Println("Too far away.")
    }
}
```

Switch (contd)

Switch evaluates cases from top to bottom and stops when a case succeeds

```
i := 0  
switch i {  
case 0:  
case f():  
}
```

Switch (contd)

- with no condition: like a sequence of *ifs*

```
switch {  
  case t.Hour() < 12:  
    fmt.Println("Good morning!")  
  case t.Hour() < 17:  
    fmt.Println("Good afternoon.")  
  default:  
    fmt.Println("Good evening.")  
}
```

- execution deferred to the end of the surrounding function

```
func main() {  
    defer fmt.Println("world")  
  
    fmt.Println("hello")  
}
```

- deferred instructions execution: LIFO