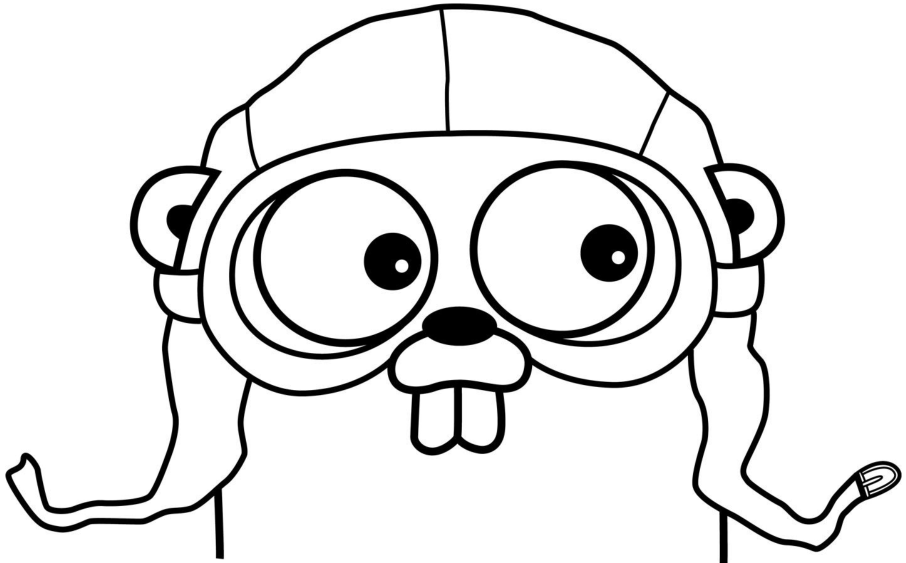
GO(LANG) by EXAMPLE

A short introduction by
Christophe Hesters & Okke van 't Verlaat



WHY GO?

FAST

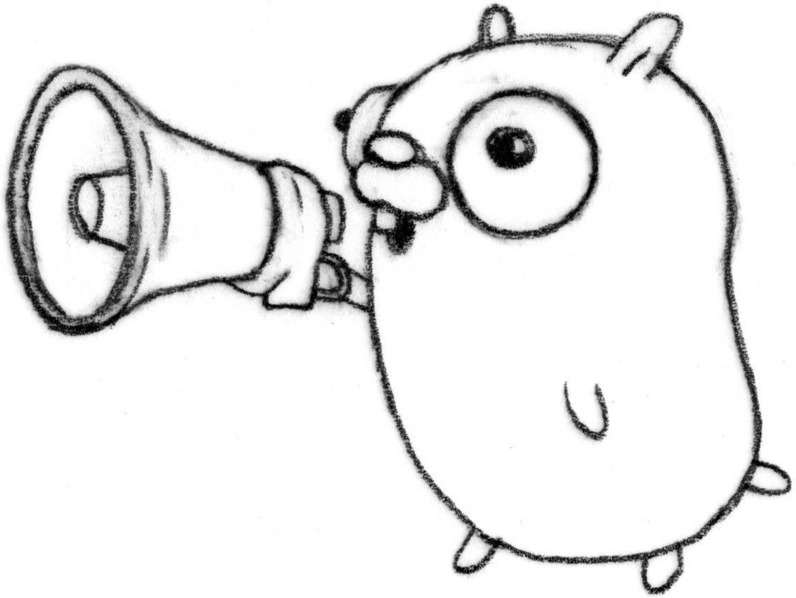
CROSS PLATFORM

SIMPLE

CONCURRENT

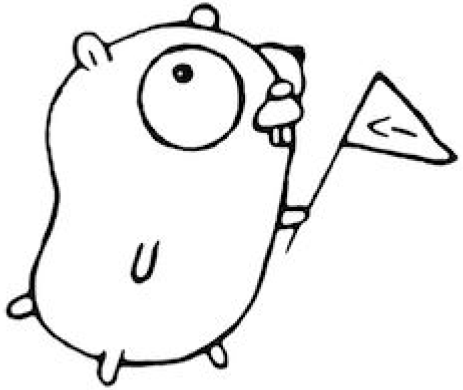
HELLO

```go
package main

import "fmt"

func main() {
    fmt.Printf("hello, world\n")
}
```



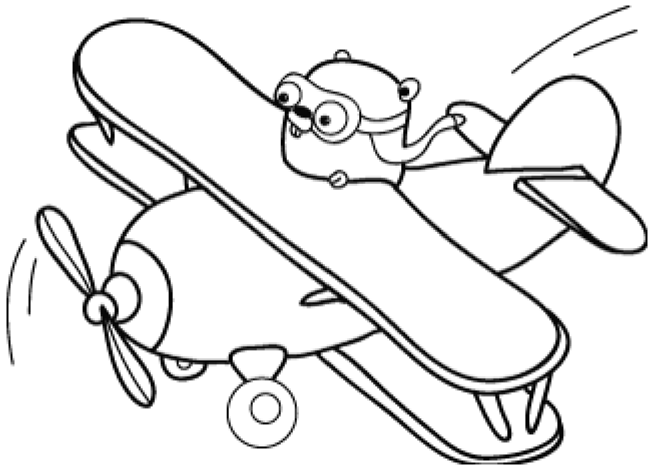
Functions

```
package main

import "fmt"

func add(x int, y int) int {
   return x + y
}

func main() {
   fmt.Println(add(42, 13))
}
```



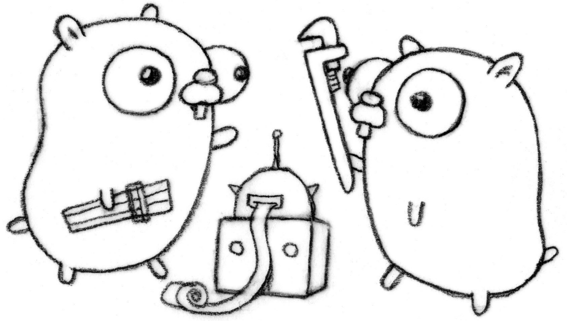
Functions are first class citizens

```go
package main

import "fmt"

func main() {
   anonymous := func(name string) string {
        return "Yo, " + name
   }

  fmt.Println(anonymous("me"))
}
```



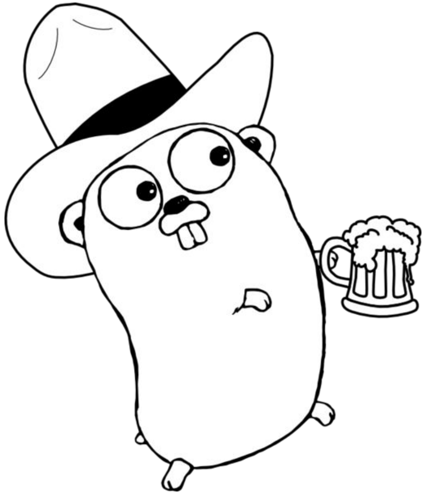
Functions are first class citizens II

```
package main

import "fmt"

func print(name string, formatter func(string) string) {
  fmt.Println(formatter(name))
}

func defaultFormatter() func(string) string {
  return func(name string) string {
      return "Yo, " + name
  }
}

func main() {
  print("me", defaultFormatter())
}
```



Functions can return multiple values

```
// make life easier by using types
//
type formatter func(string) string

func messageWithFormatter() (string, formatter) {
  return "me", defaultFormatter()
}

func main() {
  msg, formatter := messageWithFormatter()
  print(msg, formatter)
}
```

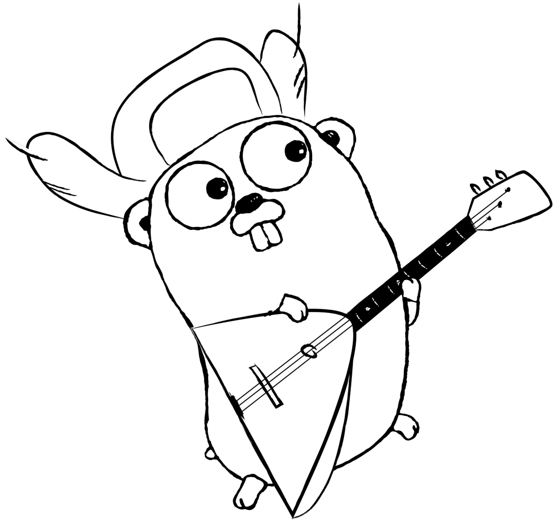


structures

```
package main

import "fmt"

type message struct {
    from string
    to   string
}

func print(msg *message) {
    fmt.Println(msg.from)
    fmt.Println(msg.to)
}

func main() {
    msg := &message{from: "okke", to: "christophe"}
    print(msg)
}
```

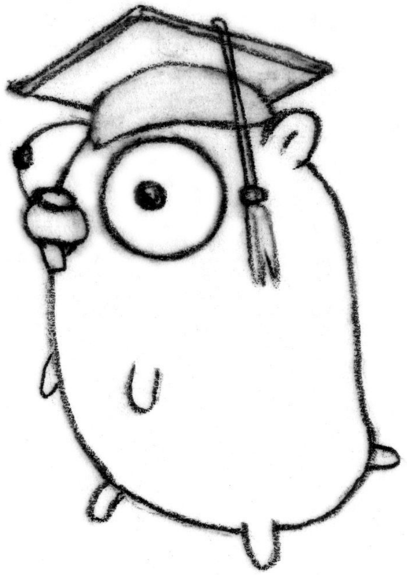
functions on structures

```go
package main

import "fmt"

type message struct {
    from string
    to   string
}

func (msg *message) print() {
    fmt.Println(msg.from)
    fmt.Println(msg.to)
}

func main() {
    msg := &message{from: "okke", to: "christophe"}
    msg.print()
}
```



interfaces

```
type message struct {
   from string
   to   string
}
type Message interface {
   Print()
}
func (msg *message) Print() {
   fmt.Println(msg.from)
   fmt.Println(msg.to)
}
func accept(msg Message) {
   msg.Print()
}

func main() {
   accept(&message{from: "okke", to: "christophe"})
}
```



there are no type hierarchies

```go
type message struct {
	from string
	to   string
}

type verboseMessage struct {
	message
	text string
}

type Message interface {
	Print()
}

type VerboseMessage interface {
	Message
	Send()
}
```

```go
func (msg *message) Print() {
	fmt.Println(msg.from)
	fmt.Println(msg.to)
}

func (verbose *verboseMessage) Print() {
	verbose.message.Print()
	fmt.Println(verbose.text)
}

func (verbose *verboseMessage) Send() {
	fmt.Printf("just send %v\n", verbose)
}

func printAndSend(msg VerboseMessage) {
	msg.Print()
	msg.Send()
}
```

```go
msg := &verboseMessage{message: message{from: "me", to: "you"}, text:"ave"}
```

Let's Try

https://github.com/toefel18/golangworkshop

```
go get github.com/toefel18/golangworkshop
```

https://github.com/a8m/go-lang-cheat-sheet/blob/master/golang_refcard.pdf

testing

in functions.go:

```go
package main

type Num int

func (n Num) addSquare() Num {
   return Num(n + (n * n))
}
```

in functions_test.go:

```go
package main

import "testing"

func TestAddSquareToNumber(t *testing.T) {
   // 4 + 4*4 = 20
   //
   if n := Num(4).addSquare(); n != 20 {
        t.Errorf("expected 20, not %v", n)
   }
}
```

run

```
> go test
```

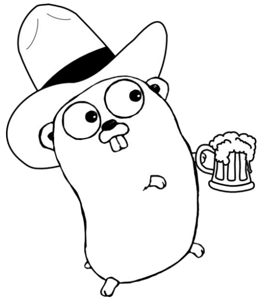
maps

```
func main() {
   m := make(map[string]int)
   m["java"] = 6
   m["ruby"] = 9
   m["go"] = 8
   m["python"] = 5

   for k, _ := range m {
        printScore(m, k)
   }
}

func printScore(m map[string]int, language string) {
   if score, exists := m[language]; exists {
        fmt.Printf("%s:%d", language, score)
   }
}
```

maps

```
func main() {
   var nullmap map[string]int // Map is null
   fmt.Println(len(nullmap))  // 0
   fmt.Println(nullmap)       // map[]
   fmt.Println(nullmap["hi"]) // 0
   nullmap["hi"] = 1          // panic: assignment to entry in nil map
}
```

slices (arrays)



```
var a [5]int
a[1] = 1
fmt.Printf("slice a:%v has len %d and capacity %d\n", a, len(a), cap(a))

b := []int{1, 2, 3, 4, 5}
fmt.Printf("slice b:%v has len %d and capacity %d\n", b, len(b), cap(b))

c := make([]int, 5)
fmt.Printf("slice c:%v has len %d and capacity %d\n", c, len(c), cap(c))

d := c[:0]
fmt.Printf("slice d:%v has len %d and capacity %d\n", d, len(d), cap(d))

d = d[1:3]
fmt.Printf("slice d:%v has len %d and capacity %d\n", d, len(d), cap(d))

d = append(d, 6)
fmt.Printf("slice d:%v has len %d and capacity %d\n", d, len(d), cap(d))
fmt.Printf("slice c:%v has len %d and capacity %d\n", c, len(c), cap(c))
```

error handling

```
package main

import "fmt"

func DivideBySquare(n int64) (float64, error) {
  if n == 0 {
      return 0, error("could not divide by zero")
  }
  return float64(n) / (float64(n) * float64(n)), nil
}

func main() {
  if n, err := DivideBySquare(3); err == nil {
      fmt.Printf("number: %v\n", n)
  }
}
```

panic, defer & recover

```
func willFail() {
   panic("oops")
}

func wontFail() {

   defer func() {

        if r := recover(); r != nil {
             fmt.Printf("Recovered from %v\n", r)
        }
   }()

   willFail()

}
```

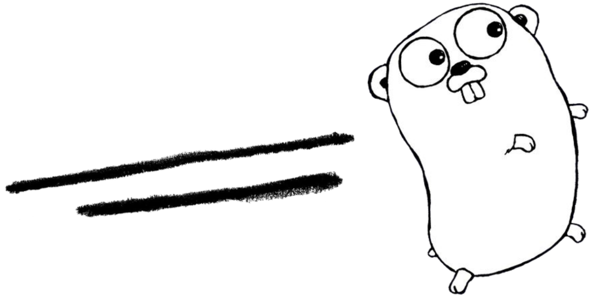
goroutines

```
func printAll(messages []string) {
   for i, v := range messages {
        fmt.Printf("%d:%s\n", i, v)
   }
}

func main() {

   printAll([]string{"foreground", "1", "2", "3"})

   go printAll([]string{"background", "a", "b", "c", "d", "e", "f"})
   go printAll([]string{"background", "z", "x", "y"})

   var input string
   fmt.Scanln(&input)
   fmt.Println("done")
}
```

channels

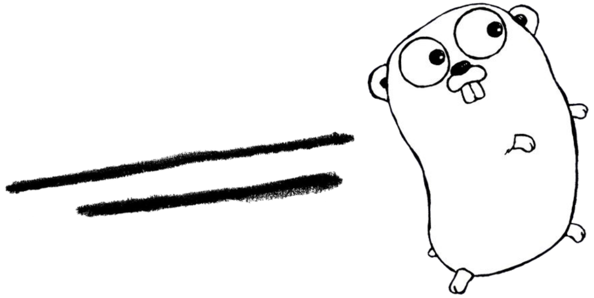
```
func producer(s string, c chan string) {
   for i := 0; ; i++ {
        c <- fmt.Sprintf("%s %d", s, i)
        runtime.Gosched()
   }
}

func consumer(c chan string) {
   for {
        fmt.Println(<-c)
   }
}
```

```
func main() {
   c := make(chan string)

   go producer("uno", c)
   go producer("dos", c)
   go consumer(c)

   time.Sleep(time.Second * 2)
}
```



Go in Docker

```
CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o main .


--- [Dockerfile] ---

FROM centurylink/ca-certs                # (FROM scratch + certificates)
COPY ./main /main
ENTRYPOINT ["/main"]
```



Let's Try some more