Go

Perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away.

-- Antoine de Saint-Exupéry

Stephan Behnke
Small Improvements

Why another language?

Software is
too slow.

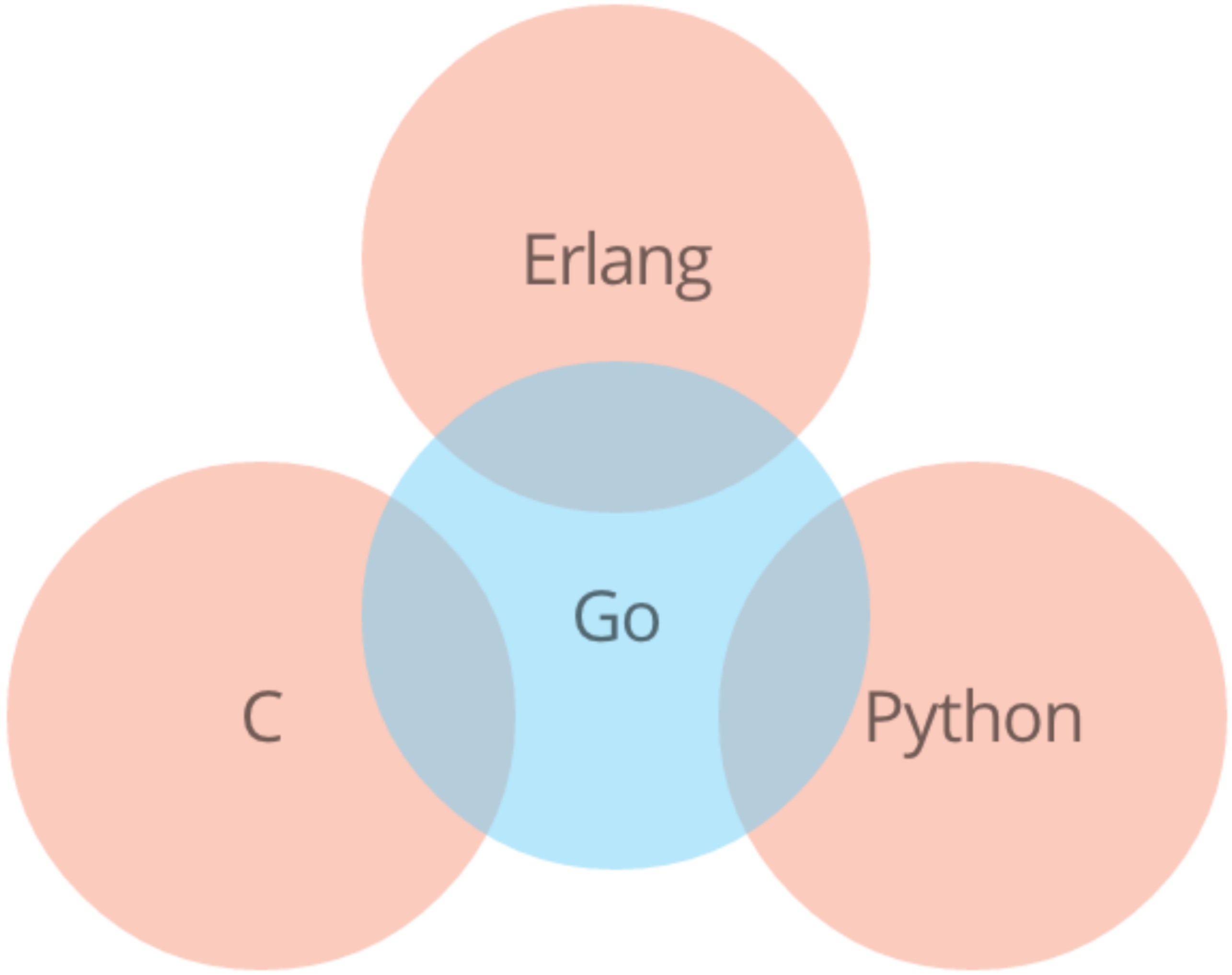
runtime
start up
build
tests

Software is
complicated.

architecture
deployment
build tools
syntax

Software is
hard to scale.

dependencies
code base
CPU cores
dev team


Erlang
Go
C
Python

Hello Go!

```go
// main.go
package main

import "fmt"

func main() {
    fmt.Println("Hello Go!")
}
```

```
> go build main.go

> ./main

  Hello Go!

> du -h ./main

  1.9M
```

```
> ./go-crosscompile-build-all
```

Imported packages are accessed by using the name as a **prefix**.

Everything starting with a **capital letter** is public. The rest is private.

Supported Platforms:

OS	Architecture
• Linux	• 32-bit
• OS X	• 64-bit
• Windows	• ARM

Hello Go!

```go
// main.go
package main

import "fmt"

func main() {
    fmt.Println("Hello Go!")
}
```

```
> go build main.go
> ./main
  Hello Go!
```

```
> du -h ./main
  1.9M
```

```
> ./go-crosscompile-build-all
```

Imported packages are accessed by using the name as a **prefix**.

Everything starting with a **capital letter** is public. The rest is private.

S

O

Compiler Speed

project	LOC	build time
groupcache	1.421	0.4 s
docker	55.007	2.6 s
stdlib	244.453	6.3 s

Example

Example

Problem

Which movie should I see?

Solution

rottentomatoes.com
each movie has a score
between 0 and 100

Requirements

- initialize application
- on request
    - load list of current movies *
    - load each movie's score
    - display results

** skipped in this example*

Main

```go
package main

import "flag"

var port int

func main() {
    // initialize API client
    config := getConfig()
    initRottenTomatoes(config)

    // initialize HTTP server
    var defaultPort int = 8080
    port = *flag.Int("port", defaultPort, "port")

    startHttpServer(port)
}
```

A variable has a **type and a value**.

Without initialization the variable has **zero value**.

Inside a function the **type can be omitted**.

Data types I

```go
type API struct {
    Url string
    Key string
}
```

A **struct** is a collection of fields.

```go
type RotTomAPI struct {
    API
}
```

A struct can **embed** other structs.

```go
func NewRotTomAPI(url string, key string) RotTomAPI {
    return RotTomAPI{API{url, key}}
}
```

Any function can be a **constructor**.

```go
func (rt RotTomAPI) String() string {
    return "RotTomAPI[" + rt.Url + "]"
}
```

A struct can have **methods**.

Parse Configuration

```go
import "strings"

func getConfig() map[string]string {

    var text string = loadConfigFile("config.txt")
    var lines []string = strings.Split(text, "\n")

    var conf = map[string]string{}
    for i := 0; i < len(lines); i++ {
        var line string = lines[i]

        keyval := strings.Split(line, "=")
        key, val := keyval[0], keyval[1]

        conf[key] = val
    }

    return conf
}
```

A **slice** is a sequence of data.

A **map** maps keys to values.

Data types II

```go
type Movies struct {
    Total  int
    Movies □Movie
}
```

```go
type Movie struct {
    Title string
    Ratings struct {
        CriticsScore  int `json:"critics_score"`
        AudienceScore int `json:"audience_score"`
    }
    // ...
```

Fields can have **type annotations**.

```go
func (m Movie) Scores() (int, int) {
    critics := m.Ratings.CriticsScore
    audience := m.Ratings.AudienceScore
    return critics, audience
}
```

Functions can **return multiple values**.

Parse JSON

```go
import "encoding/json"

func (m RotTomAPI) parse(js []byte) (*Movie, error) {
    movs := Movies{}

    err := json.Unmarshal(js, &movs)
    if err != nil {
        return nil, err
    }

    return &movs.Movies[0], nil
}
```

Pointers allow you to pass references of values.

An empty pointer is **nil**.

Errors must be handled explicitly.

API Batch Request

```go
func (m RotTomAPI) LoadAll(titles []string) []*Movie {
    var res []*Movie

    for _, title := range titles {
        mov, err := m.loadMovie(title)
        if err == nil {
            res = append(res, mov)
        }
    }

    return res
}
```

Functions often return an additional **error value**.

`range` allows to iterate over slices and maps.

API Batch Request

Loops

```
for {
    ...
}
```

```
for i < 10 {
    ...
}
```

```
for i := 0; i < 10; i++ {
    ...
}
```

```
for i, val := range some_slice {
    ...
}
```

```
for key, val := range some_map {
    ...
}
```

```
□*Movie {
```

Functions often return an additional **error value**.

range allows to iterate over slices and maps.

API Batch Request

```go
func (m RotTomAPI) LoadAll(titles []string) []*Movie {
    var res []*Movie

    for _, title := range titles {
        mov, err := m.loadMovie(title)
        if err == nil {
            res = append(res, mov)
        }
    }

    return res
}
```

Functions often return

Go is strict

- **no** unused variables

```
var declared and not used
```

- **no** unused imports

```
imported and not used: "..."
```

- **no** cyclic dependencies

```
import cycle not allowed
```

- **no** method overloading

```
func redeclared in this block
```

Standard library

Batteries included.

- HTTP server + client
- HTML templates
- cryptography
- XML and JSON
- reflection
- hashing
- testing
- UTF-8
- RPC
- SQL

HTTP Server

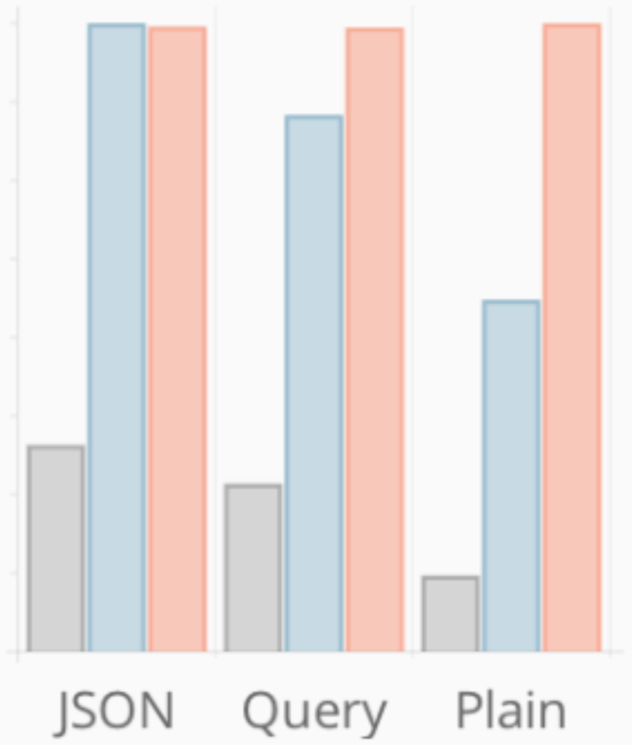
```go
import "fmt"
import "net/http"
import "html/template"

func startHttpServer(port int) {
    http.HandleFunc("/movies", movies)

    addr := fmt.Sprintf("localhost:%d", port)
    http.ListenAndServe(addr, nil)
}

func movies(w http.ResponseWriter, r *http.Request) {
    tmpl, _ := template.ParseFiles("movies.html")
    tmpl.Execute(w, loadMovies())
}
```

Functions can use other **functions as arguments**.

HTTP S

Runtime Speed

Node <> Go <> Java



http://techempower.com/benchmarks

```
t)

Request) {
html")
```

Functions can use other **functions as arguments**.

API Batch Request: async

```go
func (m RotTomAPI) LoadAll(titles □string) □*Movie {
    var res □*Movie
    ch := make(chan *Movie) // init

    // initiate requests
    for _, title := range titles {
        go func(qry string) {
            mov, _ := m.loadMovie(qry)
            ch <- mov // send
        }(title)
    }

    // collect results
    for _ = range titles {
        mov := <-ch // receive
        if mov != nil {
            res = append(res, mov)
        }
    }
    return res
}
```

A goroutine is like a **lightweight thread**.

It works like the **&** in Unix.

API Batch Request: async

```go
func (m RotTomAPI) LoadAll(titles []string) []*Movie {
    var res []*Movie
    ch := make(chan *Movie) // init

    // initiate requests
    for _, title := range titles {
        go func(qry string) {
            mov, _ := m.loadMovie(qry)
            ch <- mov // send
        }(title)
    }

    // collect results
    for _ = range titles {
        mov := <-ch // receive
        if mov != nil {
            res = append(res, mov)
        }
    }
    return res
}
```

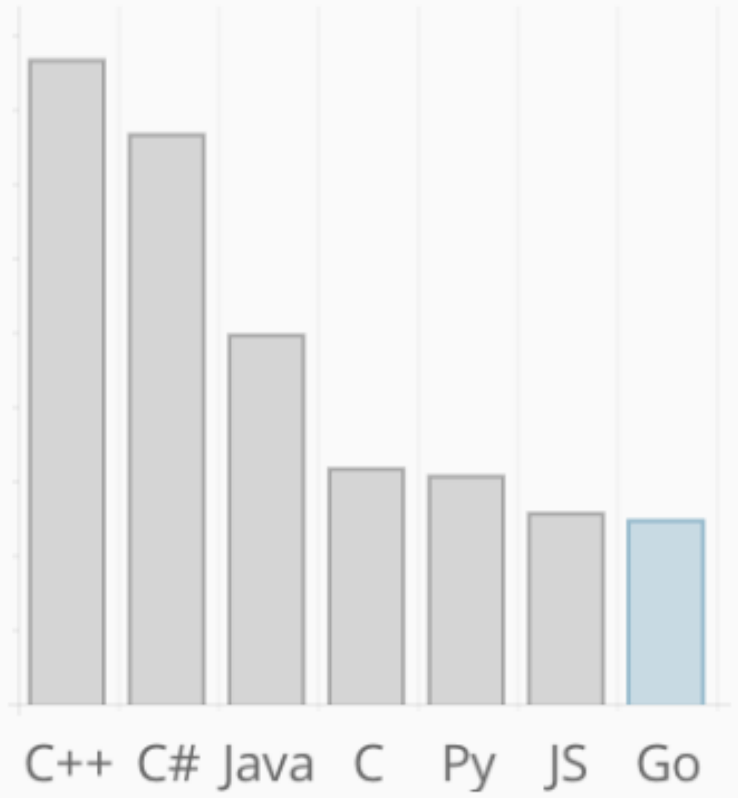
A goroutine is like a **lightweight thread**.

It works like the **&** in Unix.

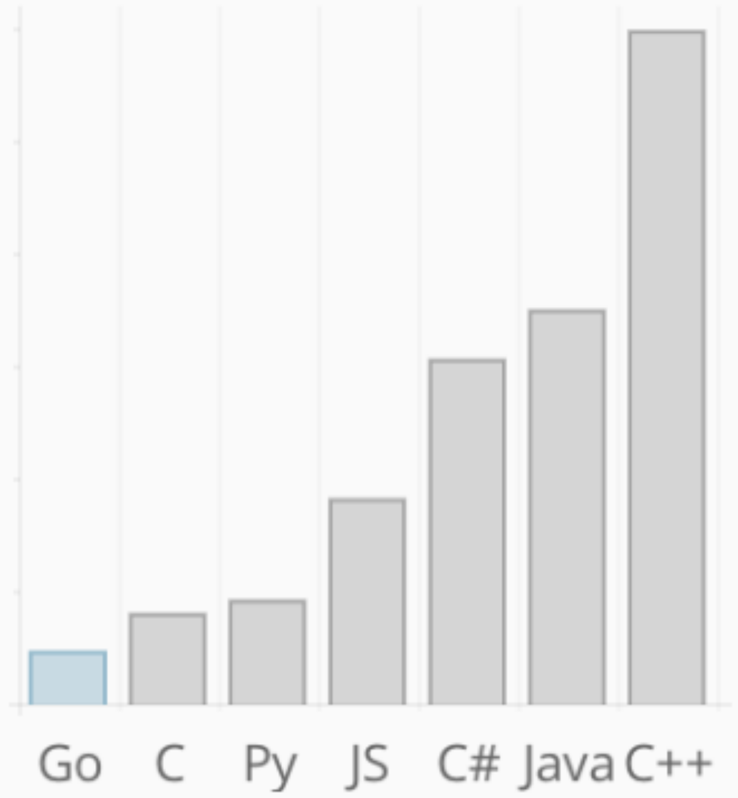
A channel is a **pipe** to connect goroutines.

It waits until sender and receiver are ready.

Reserved Keywords



Words in Spec



Going on ...

Polymorphism

```
type Swimmer interface {
    Swim()
}

type Quacker interface {
    Quack()
}

type Walker interface {
    Walk()
}

type Duck interface {
    Swimmer
    Quacker
    Walker
}
```

An **interface** is set of methods. They can be **composed**.

```
type DarkwingDuck struct {}

func (d DarkwingDuck) Swim() {
    fmt.Printf("I'm swimming!")
}

func (d DarkwingDuck) Walk() {
    fmt.Printf("I'm walking!")
}

func (d DarkwingDuck) Quack() {
    fmt.Printf("Quack!")
}
```

DarkwingDuck implements all methods of type **Duck**.

It **is** a Duck.

Polymorphism

```go
type Swimmer interface {
    Swim()
}

type Quacker interface {
    Quack()
}

type Walker interface {
    Walk()
}

type Duck interface {
    Swimmer
    Quacker
    Walker
}
```

```go
type DarkwingDuck struct {}

func (d DarkwingDuck) Swim() {
    fmt.Printf("I'm swimming!")
}
```

An **interface** is set of methods.
They can be **composed**.

Tools

```
> go test
```

*run test files ('*_test.go')*

```
> go fmt
```

formats source code

```
> gdb <executable>
```

starts debugger

```
> go vet
```

shows code warnings

```
> go tool pprof
```

CPU profiling

```
> golint
```

show style warnings

Use cases

Who uses Go?

CLOUDFLARE

DISQUS

YouTube

docker

Iron.io

Google

GitHub

SendGrid

SOUNDCLOUD

10gen

Where to use Go?

Good fit

- networking app
- high-scalability
- cloud servers
- command-line app
- agent

It depends

- classic web application

Bad idea

- pluggable application
- GUI application
- real-time system

Should I stay or should I Go?

The Bad

- **GC**
  stop-the-world garbage collector
- **package management**
  no versioning
- **external libraries**
  quantity and maturity
- **advanced tooling**
  missing good IDE support

The Ugly (controversial)

- **error handling**
  every error is handled explicitly
- **no generics**
  only build-in data structures are generic

Yeah, another language!

Go
is fast.

runtime
start up
build
tests

Go
is simple.

architecture
deployment
build tools
syntax

Go
is scalable.

dependencies
code base
CPU cores
dev team

Yeah, another language!

Go
is fast.

runtime

start up

build

tests

Go
is simple.

architect

deploym

build to

syntax

Go
is scalable.

Plus: Cute mascot.

-- the Gopher

Thank you.

Now is the time for questions.

Stephan Behnke

@stebehn

http://small-improvements.com

We're hiring smart people.

http://small-improvements.com/careers

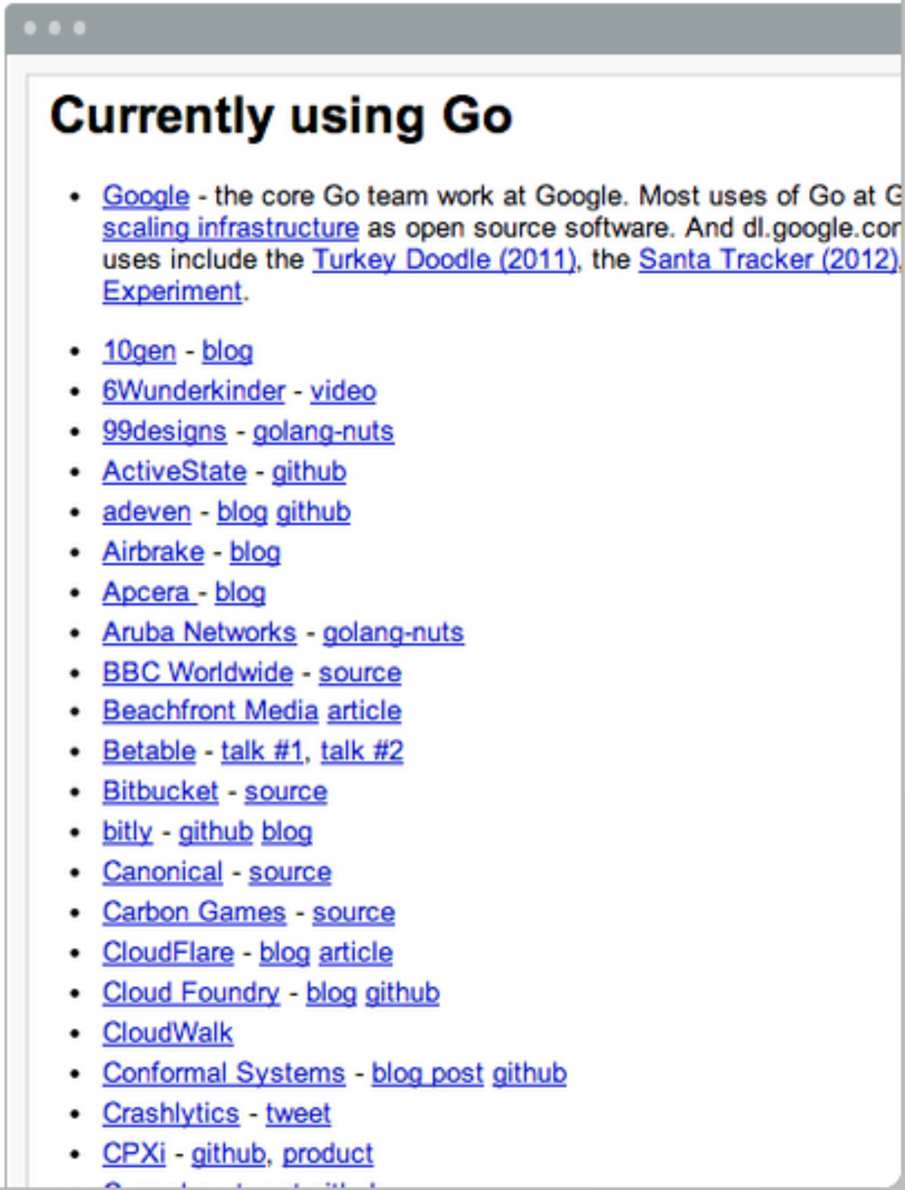
Credits

- **Image: Gopher**
  [http://dave.cheney.net/resources-for-new-go-programmers]
- **Image: Cat**
  [http://memecrunch.com/meme/2WIPI/quack]
- **Image: Rocket**
  [http://de.freepik.com/freie-ikonen/rakete_694699.htm]
- **Image: Company logos**
  [each company's website]

- **Consulting**
  David Gerstl, Timur Çelikel and Dennis Dillert

Appendix: Where to Go from here?

- GDG Berlin Golang
  http://bit.ly/go-berlin-meetup
- Video: Get Started with Go
  http://bit.ly/go-intro-video
- A Tour of Go
  http://tour.golang.org
- Effective Go
  http://bit.ly/effective_go
- Organizations that use Go
  http://bit.ly/go_orgs



Currently using Go

- Google - the core Go team work at Google. Most uses of Go at G
  scaling infrastructure as open source software. And dl.google.cor
  uses include the Turkey Doodle (2011), the Santa Tracker (2012),
  Experiment.
- 10gen - blog
- 6Wunderkinder - video
- 99designs - golang-nuts
- ActiveState - github
- adeven - blog github
- Airbrake - blog
- Apcera - blog
- Aruba Networks - golang-nuts
- BBC Worldwide - source
- Beachfront Media article
- Betable - talk #1, talk #2
- Bitbucket - source
- bitly - github blog
- Canonical - source
- Carbon Games - source
- CloudFlare - blog article
- Cloud Foundry - blog github
- CloudWalk
- Conformal Systems - blog post github
- Crashlytics - tweet
- CPXi - github, product

Appendix: Advanced Concurrency

```go
ch := make(chan *Movie)
throttle := time.Tick(200 * time.Millisecond)

for _, title := range titles {
    <-throttle
    go func(qry string) {
        mov, _ := m.loadMovie(qry)
        ch <- mov
    }(title)
}

for {
    select {
    case mov := <-ch:
        res = append(res, mov)
        if len(res) == len(titles) {
            return
        }
    case <-time.After(5 * time.Second):
        return
    }
}
```

Appendix: Formatting

```go
package main

import ("net/http"
)


type User struct {
    mail string // lowercase
    lastSeen time.Time // last login
}

func
(u *User) String() string {
    if (u.mail=="") {
        return   "[unkown]"
    }
    return "["+u.mail+"]"
}
```

```go
package main

import (
    "time"
)

type User struct {
    mail     string    // lowercase
    lastSeen time.Time // last login
}

func (u *User) String() string {
    if u.mail == "" {
        return "[unkown]"
    }
    return "[" + u.mail + "]"
}
```

```
> gofmt -w main.go

> goimports -w main.go
```

Appendix: Testing

```go
// math_test.go
import "math"
import "testing"

func TestFloor(t *testing.T) {
    v := math.Floor(1.5)
    if v != 1.0 {
        t.Error("Expected 1.0, got ", v)
    }
}
```

```
> go test

  PASS

  ok    main    0.010s
```

```go
import . "github.com/onsi/gomega"

Describe("Math", func() {
    It("should return floor", func() {
        v := math.Floor(1.5)
        Expect(v, Equals, 1.0)
    })
})
```

```
> go test

  Running Suite: Math

  =====================

  Ran in 0.010 seconds

  SUCCESS!
```

Appendix: Sorting

```go
package main

import "sort"
import "fmt"

type ByLength □string

func (s ByLength) Len() int {
    return len(s)
}
func (s ByLength) Swap(i, j int) {
    s[i], s[j] = s[j], s[i]
}
func (s ByLength) Less(i, j int) bool {
    return len(s[i]) < len(s[j])
}

func main() {
    fruits := □string{"peach", "banana", "kiwi"}
    sort.Sort(ByLength(fruits))
    fmt.Println(fruits)
}
```