Documentation & Variadic

resources, help, variadic, BNF, documentation

Preview

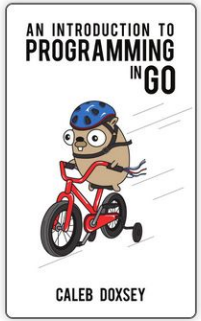
- learning golang from golang.org
  - http://golang.org/doc/
    - https://golang.org/doc/install
    - https://golang.org/ref/spec
    - https://tour.golang.org/welcome/1
    - https://golang.org/doc/code.html
    - https://golang.org/doc/effective_go.html
    - https://blog.golang.org/

- getting help
  - **go help**
    - **go help [command]**
    - **go help [topic]**
    - go help examples:
      - go help env
        - **variadic**
          - **...**
        - BNF / EBNF
          - **Backus Naur Form**
          - Extended Backus Naur Form
      - go help packages
      - go help build
      - go help install
  - **documentation**
    - **godoc.org**
      - **“imported by” example code**
    - golang.org
    - **godoc**
      - **google: golang godoc**
        - golang.org/x/tools/cmd/godoc
      - godoc -src fmt Println
      - godoc -src fmt
      - godoc -http=:6060
      - godoc -html fmt
      - godoc -html fmt Println
      - godoc -q Reader
      - godoc -q Writer
      - godoc -q rand
    - close server
      - **ctrl + c**
    - **http://go-search.org/**

resources

Go Resources

Books



An Introduction to Programming in Go

A short, concise introduction to computer programming using the language Go. Designed by Google, Go is a general purpose programming language with modern features, clean syntax and a robust well-documented common library, making it an ideal language to learn as your first programming language.

Guides

1. **Machine Setup**
   - Windows
   - OSX
2. **Bootcamp**
   - A 4-week instructional series that covers the material in *An Introduction to Programming in Go* as well as the basics of server-side web development and Google App Engine.

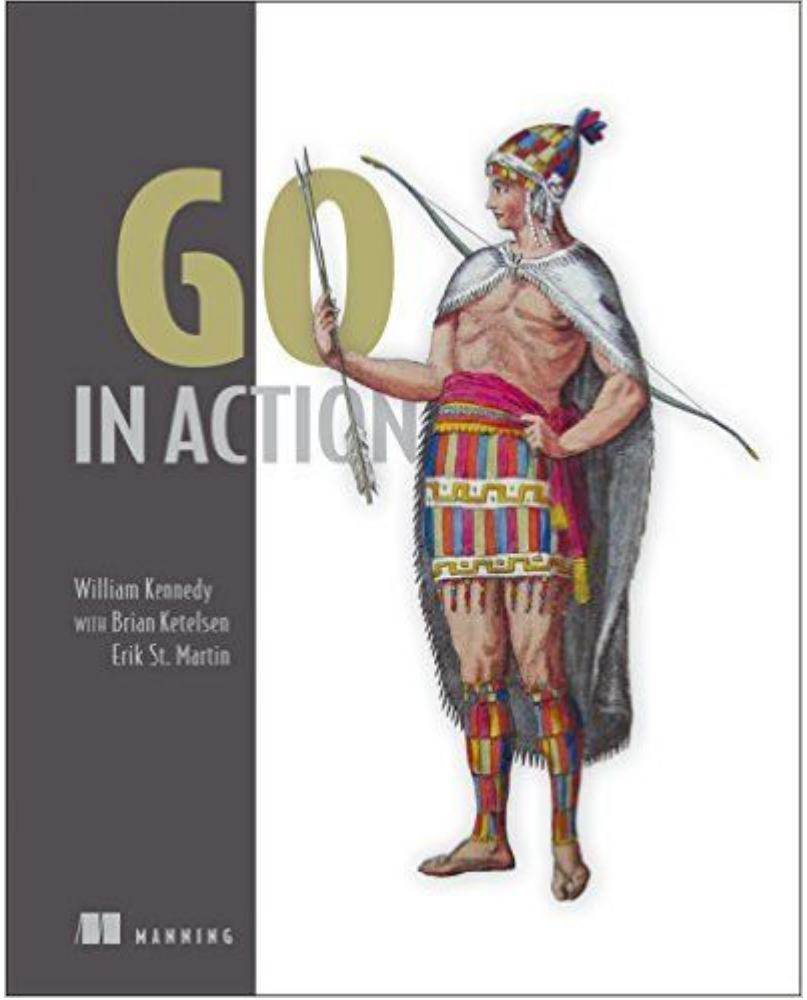
https://www.golang-book.com/

Go by Example

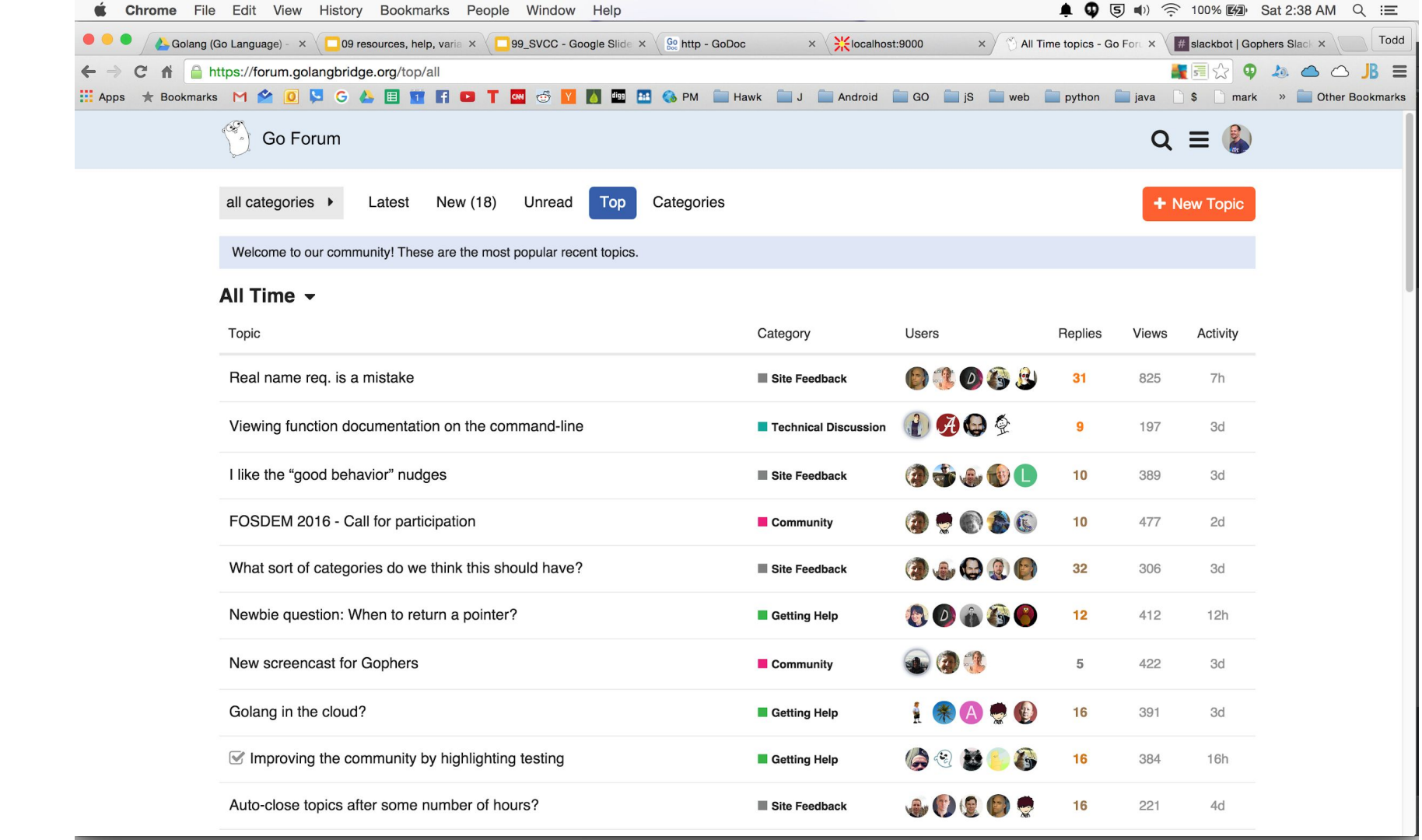
Go is an open source programming language designed for building simple, fast, and reliable software.

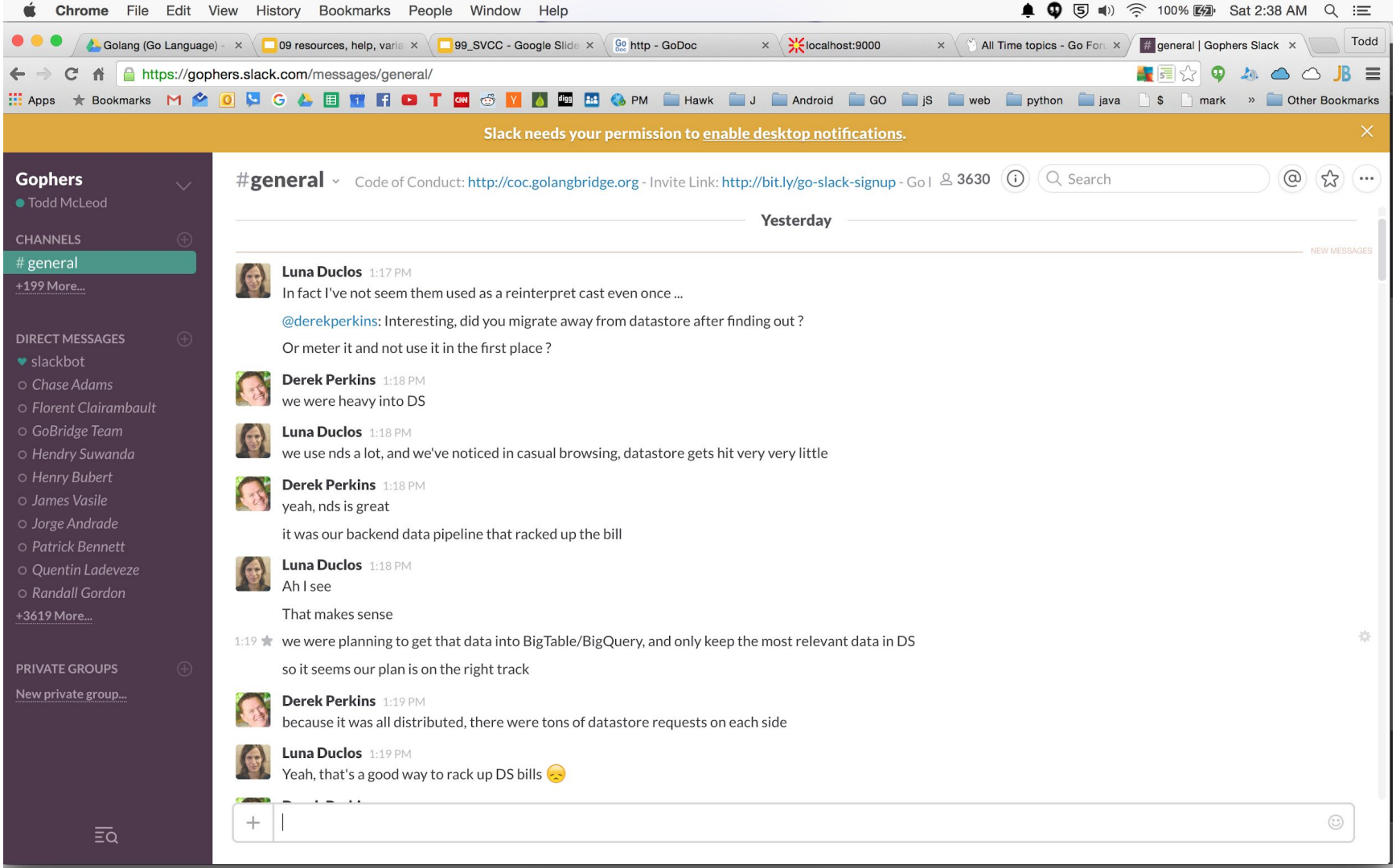
Go by Example is a hands-on introduction to Go using annotated example programs. Check out the first example or browse the full list below.

https://gobyexample.com/



https://www.manning.com/books/go-in-action


Chrome File Edit View History Bookmarks People Window Help
100% Sat 2:38 AM
Golang (Go Language) -
09 resources, help, varia
99_SVCC - Google Slide
http - GoDoc
localhost:9000
All Time topics - Go For
slackbot | Gophers Slac
Todd
https://forum.golangbridge.org/top/all
Apps Bookmarks PM Hawk J Android GO jS web python java $ mark Other Bookmarks
Go Forum
all categories
Latest
New (18)
Unread
Top
Categories
+ New Topic
Welcome to our community! These are the most popular recent topics.
All Time
Topic
Category
Users
Replies
Views
Activity
Real name req. is a mistake
Site Feedback
31
825
7h
Viewing function documentation on the command-line
Technical Discussion
9
197
3d
I like the “good behavior” nudges
Site Feedback
10
389
3d
FOSDEM 2016 - Call for participation
Community
10
477
2d
What sort of categories do we think this should have?
Site Feedback
32
306
3d
Newbie question: When to return a pointer?
Getting Help
12
412
12h
New screencast for Gophers
Community
5
422
3d
Golang in the cloud?
Getting Help
16
391
3d
Improving the community by highlighting testing
Getting Help
16
384
16h
Auto-close topics after some number of hours?
Site Feedback
16
221
4d


Chrome File Edit View History Bookmarks People Window Help
100% Sat 2:38 AM
Golang (Go Language) -
09 resources, help, varia
99_SVCC - Google Slide
http - GoDoc
localhost:9000
All Time topics - Go For
general | Gophers Slack
Todd
https://gophers.slack.com/messages/general/
Apps Bookmarks PM Hawk J Android GO jS web python java $ mark Other Bookmarks
Slack needs your permission to enable desktop notifications.
Gophers
Todd McLeod
CHANNELS
general
+199 More...
DIRECT MESSAGES
slackbot
Chase Adams
Florent Clairambault
GoBridge Team
Hendry Suwanda
Henry Bubert
James Vasile
Jorge Andrade
Patrick Bennett
Quentin Ladeveze
Randall Gordon
+3619 More...
PRIVATE GROUPS
New private group...
#general
Code of Conduct: http://coc.golangbridge.org - Invite Link: http://bit.ly/go-slack-signup - Go I
3630
Search
Yesterday
NEW MESSAGES
Luna Duclos 1:17 PM
In fact I've not seem them used as a reinterpret cast even once ...
@derekperkins: Interesting, did you migrate away from datastore after finding out ?
Or meter it and not use it in the first place ?
Derek Perkins 1:18 PM
we were heavy into DS
Luna Duclos 1:18 PM
we use nds a lot, and we've noticed in casual browsing, datastore gets hit very very little
Derek Perkins 1:18 PM
yeah, nds is great
it was our backend data pipeline that racked up the bill
Luna Duclos 1:18 PM
Ah I see
That makes sense
1:19 we were planning to get that data into BigTable/BigQuery, and only keep the most relevant data in DS
so it seems our plan is on the right track
Derek Perkins 1:19 PM
because it was all distributed, there were tons of datastore requests on each side
Luna Duclos 1:19 PM
Yeah, that's a good way to rack up DS bills

The Go Programming Language

Documents | Packages | The Project | Help | Blog | Search

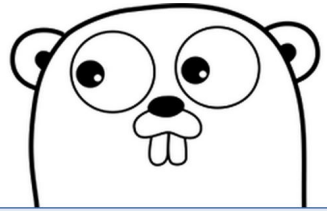
Try Go

Pop-out

```
// You can edit this code!
// Click here and start typing.
package main

import "fmt"

func main() {
        fmt.Println("Hello, 世界")
}
```

Hello, World! | Run | Share | Tour

Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.



Download Go

Binary distributions available for
Linux, Mac OS X, Windows, and more.

Featured video

Featured articles

GopherCon 2015 Roundup

A few weeks ago, Go programmers from around the world descended on Denver, Colorado for GopherCon 2015. The two-day, single-track conference attracted more than 1,250 attendees—nearly double last year's number—and featured 22 talks presented by Go community members.

Published 28 July 2015

Go, Open Source, Community

[This is the text of my opening keynote at Gophercon 2015. The video is available here.]

Published 8 July 2015

Read more

https://golang.org/

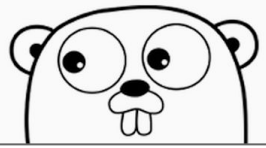
A Tour of Go

hello.go

Syntax off

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Println("Hello, 世界")
7  }
8
```

Reset Format Run



Hello, 世界

Welcome to a tour of the Go programming language.

The tour is divided into a list of modules that you can access by clicking on A Tour of Go on the top left of the page.

You can also view the table of contents at any time by clicking on the menu on the top right of the page.

Throughout the tour you will find a series of slides and exercises for you to complete.

You can navigate through them using

- "previous" or `PageUp` to go to the previous page,
- "next" or `PageDown` to go to the next page.

The tour is interactive. Click the Run button now (or type `shift-enter`) to compile and run the program on a remote server. The result is displayed below the code.

These example programs demonstrate different aspects of Go. The programs in the tour are meant to be starting points for your own experimentation.

Edit the program and run it again.

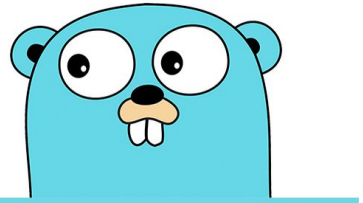
Note that when you click on Format or `ctrl-enter` the text in the editor is formatted using the gofmt tool. You can switch on and off syntax highlighting the clicking on syntax button.

When you're ready to move on, click the right arrow below or type the `PageDown` key.

< 1/4 >

https://tour.golang.org/welcome/1

ARCHIVES | SEARCH | LATEST ISSUE | RSS



Go Newsletter

A weekly newsletter about the Go programming language

Enter your e-mail address | Subscribe now »

Check out our latest issue (71) for a sample of what we're all about.

ONE e-mail each Thursday. Easy to unsubscribe.
No spam — your e-mail address is safe
Published by Cooper Press and curated by Peter Cooper and Matt Cottingham

Go
@golang

Follow

If you're a #golang programmer you should really be subscribed to golangweekly.com. Even I learn new stuff with each issue! :-)
5:02 PM - 15 Sep 2013

http://golangweekly.com/

learning golang
from golang.org

Documentation

The Go programming language is an open source project to make programmers more productive.

Go is expressive, concise, clean, and efficient. Its concurrency mechanisms make it easy to write programs that get the most out of multicore and networked machines, while its novel type system enables flexible and modular program construction. Go compiles quickly to machine code yet has the convenience of garbage collection and the power of run-time reflection. It's a fast, statically typed, compiled language that feels like a dynamically typed, interpreted language.

Installing Go

Getting Started

Instructions for downloading and installing the Go compilers, tools, and libraries.

Learning Go

A Tour of Go

An interactive introduction to Go in three sections. The first section covers basic syntax and data structures; the second discusses methods and interfaces; and the third introduces Go's concurrency primitives. Each section concludes with a few exercises so you can practice what you've learned. You can take the tour online or install it locally.

How to write Go code

Also available as a screencast, this doc explains how to use the go command to fetch, build, and install packages, commands, and run tests.

Effective Go

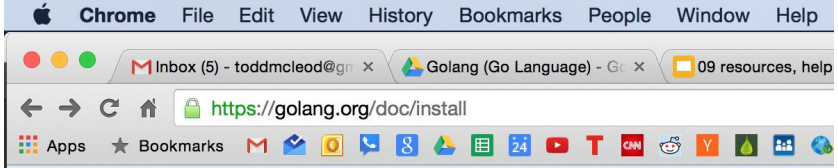
A document that gives tips for writing clear, idiomatic Go code. A must read for any new Go programmer. It augments the tour and the language specification, both of which should be read first.

Frequently Asked Questions (FAQ)

Answers to common questions about Go.

The Go Wiki

install



The Go Programming Language

Getting Started

Download the Go distribution

Download Go
Click here to visit the downloads page

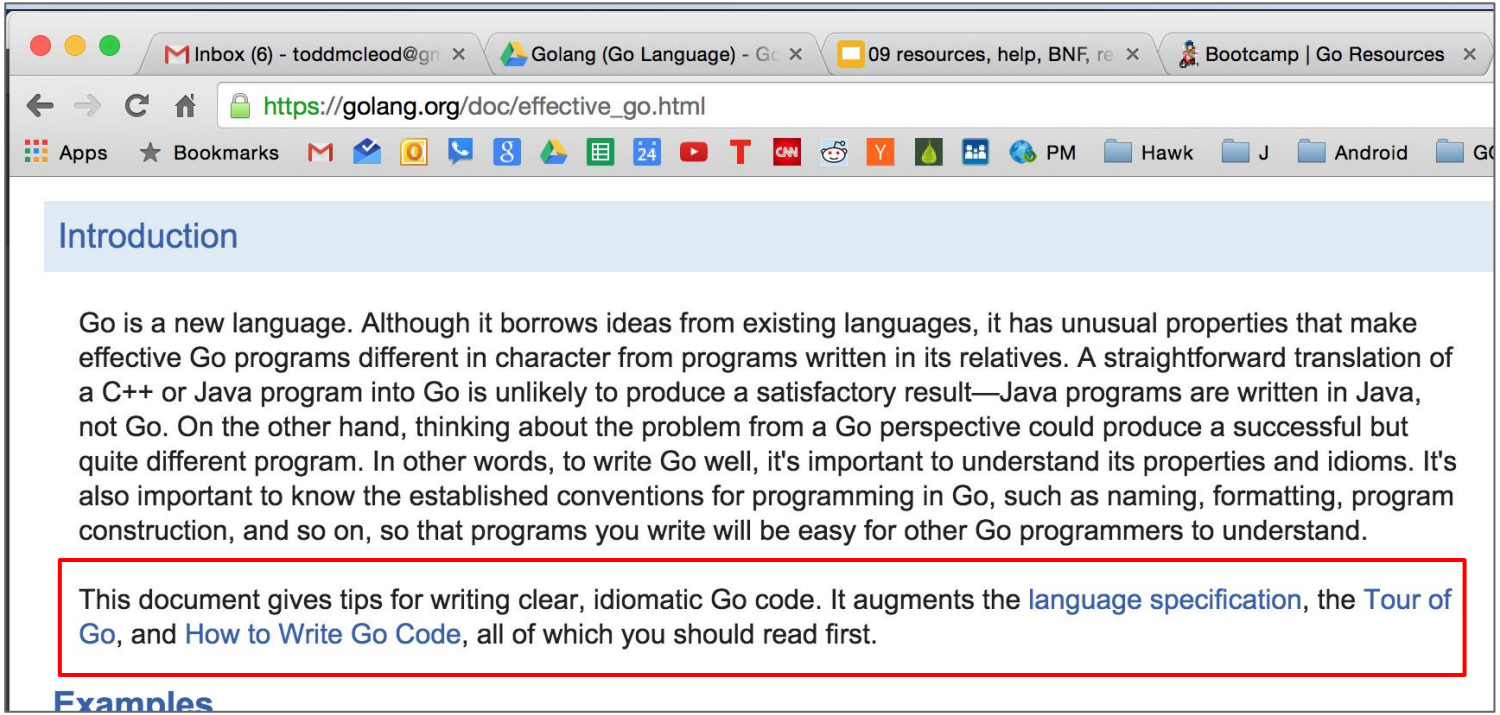
Official binary distributions are available for the FreeBSD (release 8-STABLE a
and above), and Windows operating systems and the 32-bit (386) and 64-bit (a
architectures.

If a binary distribution is not available for your combination of operating system
from source or installing gccgo instead of gc.

System requirements

Go binary distributions are available for these supported operating systems an

effective go
but not just yet ...



language spec

The Go Programming Language Specification

Version of November 11, 2014

tour of go
we already saw this

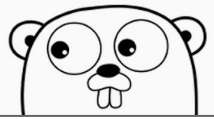
A Tour of Go

hello.go

Syntax off

```
1  package main
2
3  import "fmt"
4
5  func main() {
6      fmt.Println("Hello, 世界")
7  }
8
```

Reset Format Run



Hello, 世界

Welcome to a tour of the Go programming language.

The tour is divided into a list of modules that you can access by clicking on A Tour of Go on the top left of the page.

You can also view the table of contents at any time by clicking on the menu on the top right of the page.

Throughout the tour you will find a series of slides and exercises for you to complete.

You can navigate through them using

- "previous" or `PageUp` to go to the previous page,
- "next" or `PageDown` to go to the next page.

The tour is interactive. Click the Run button now (or type `shift-enter`) to compile and run the program on a remote server. The result is displayed below the code.

These example programs demonstrate different aspects of Go. The programs in the tour are meant to be starting points for your own experimentation.

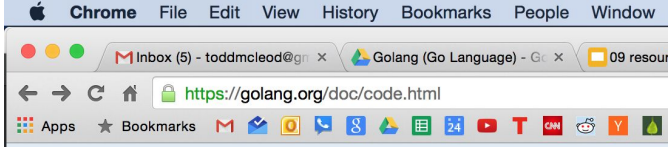
Edit the program and run it again.

Note that when you click on Format or `ctrl-enter` the text in the editor is formatted using the gofmt tool. You can switch on and off syntax highlighting the clicking on syntax button.

When you're ready to move on, click the right arrow below or type the `PageDown` key.

< 1/4 >

how to write go code



The Go Programming Language

How to Write Go Code

Introduction

This document demonstrates the development of a simple Go package
way to fetch, build, and install Go packages and commands.

The go tool requires you to organize your code in a specific way. Please
the simplest way to get up and running with your Go installation.

A similar explanation is available as a screencast.

Code organization

Workspaces

The go tool is designed to work with open source code maintained in pu

Now …
effective go

The Go Programming Language

Effective Go

https://golang.org/doc/effective_go.html

the go blog

Articles

The Go Blog

The official blog of the Go project, featuring news and in-depth articles by the Go team and guests.

Codewalks

Guided tours of Go programs.

- First-Class Functions in Go
- Generating arbitrary text: a Markov chain algorithm
- Share Memory by Communicating
- Writing Web Applications - building a simple web application.

Language

- JSON-RPC: a tale of interfaces
- Go's Declaration Syntax
- Defer, Panic, and Recover
- Go Concurrency Patterns: Timing out, moving on
- Go Slices: usage and internals
- A GIF decoder: an exercise in Go interfaces
- Error Handling and Go
- Organizing Go code

Packages

- JSON and Go - using the json package.
- Gobs of data - the design and use of the gob package.
- The Laws of Reflection - the fundamentals of the reflect package.
- The Go image package - the fundamentals of the image package.
- The Go image/draw package - the fundamentals of the image/draw package.

Tools

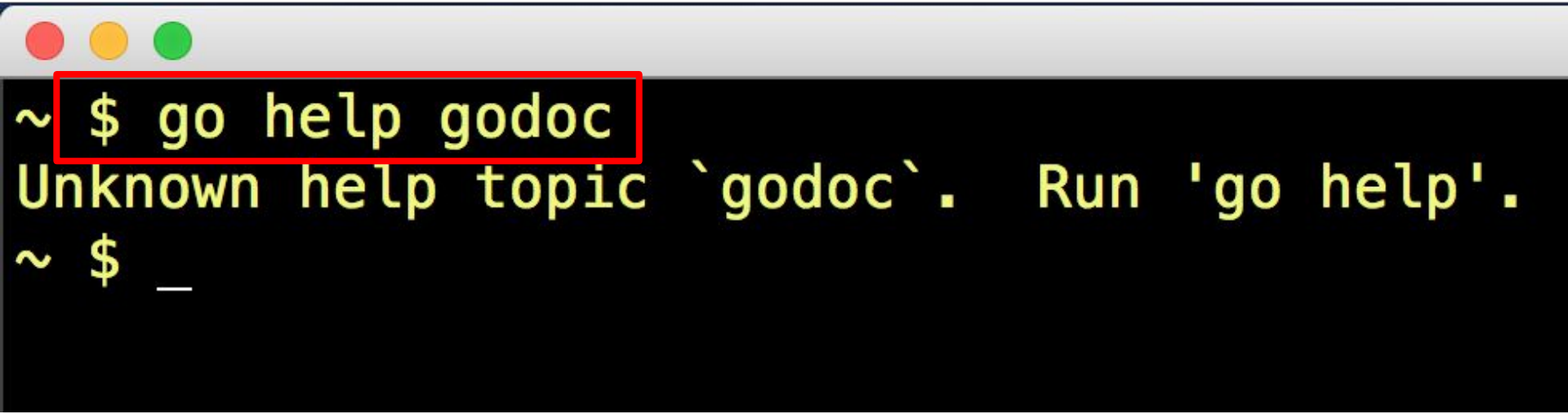
- About the Go command - why we wrote it, what it is, what it's not, and how to use it

https://golang.org/doc/#articles

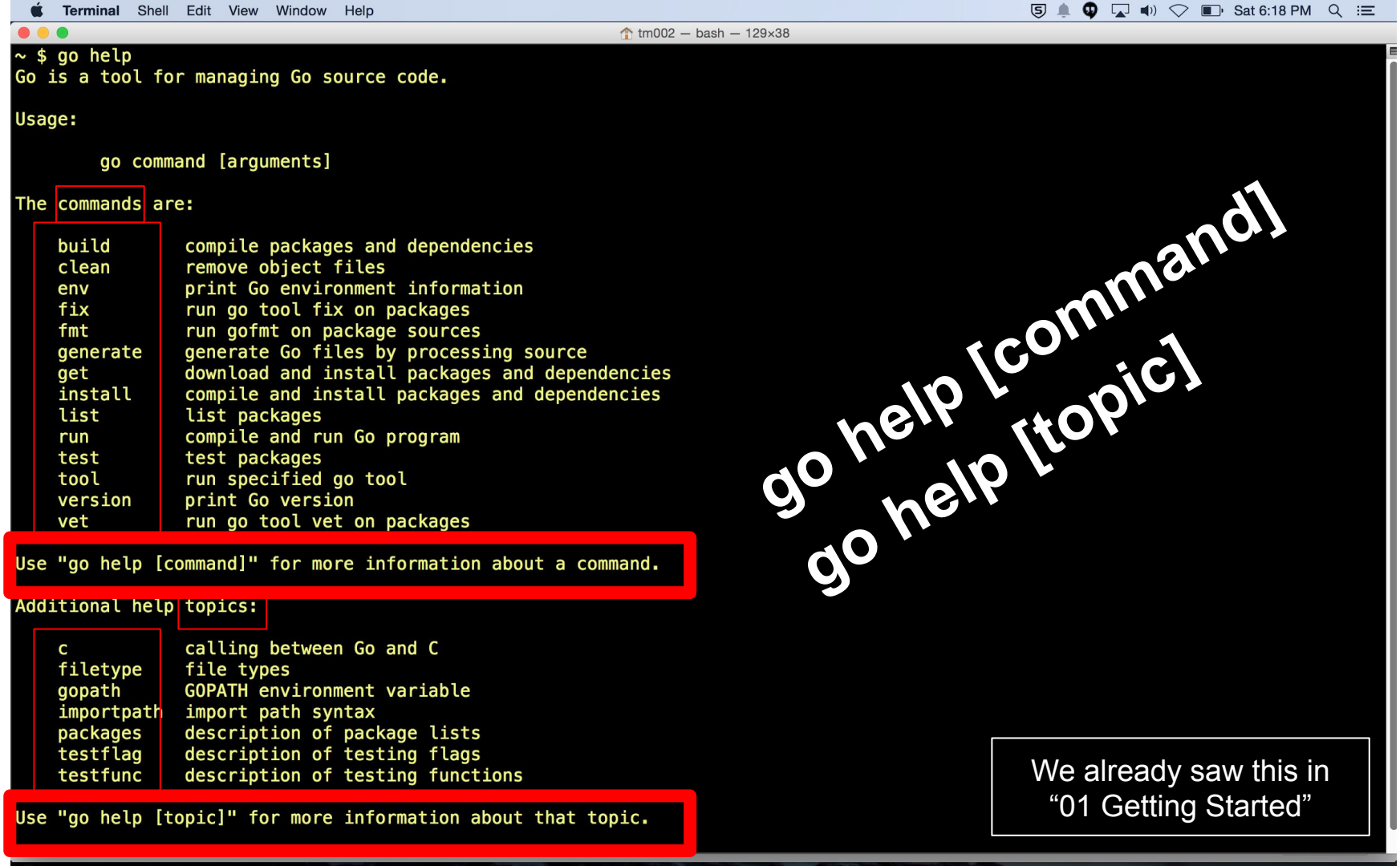
getting help

go help

review

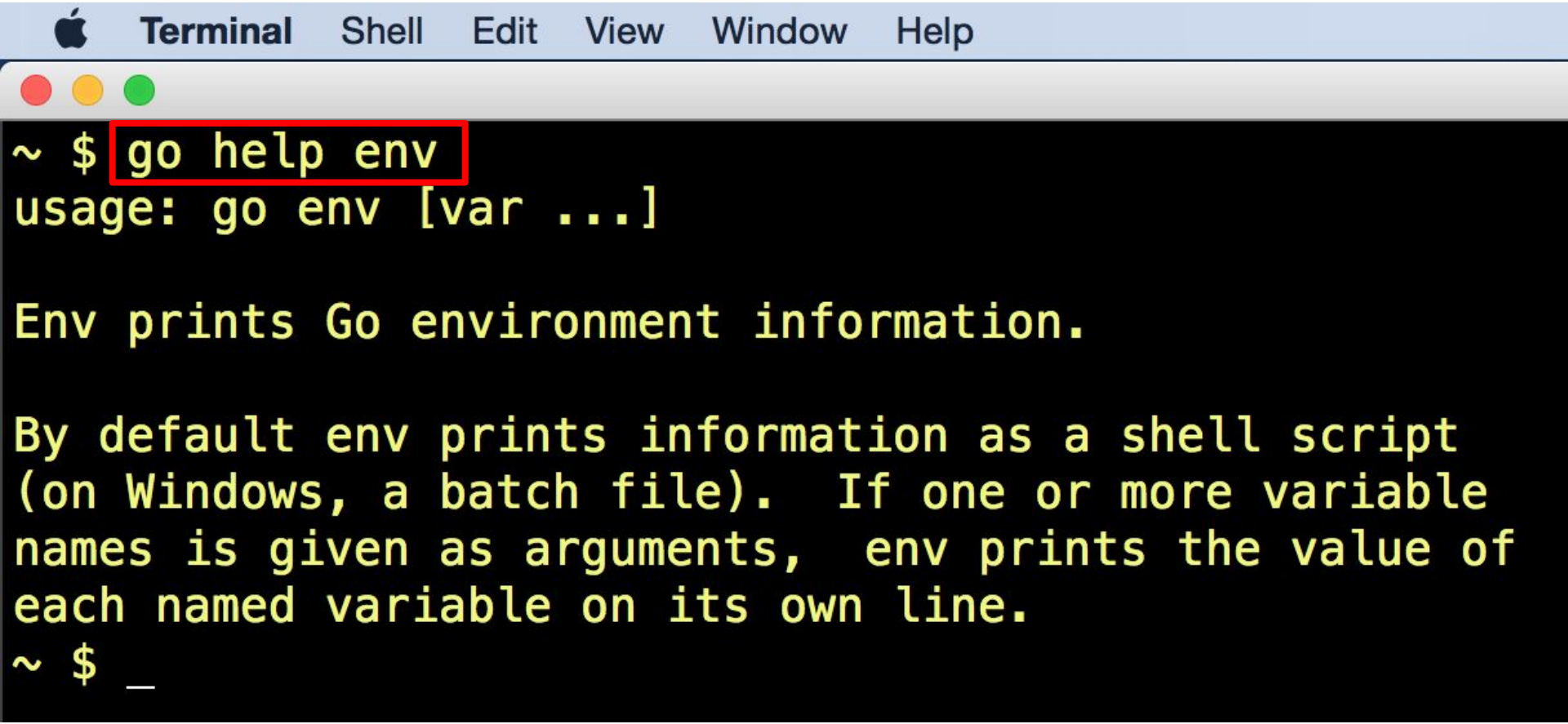


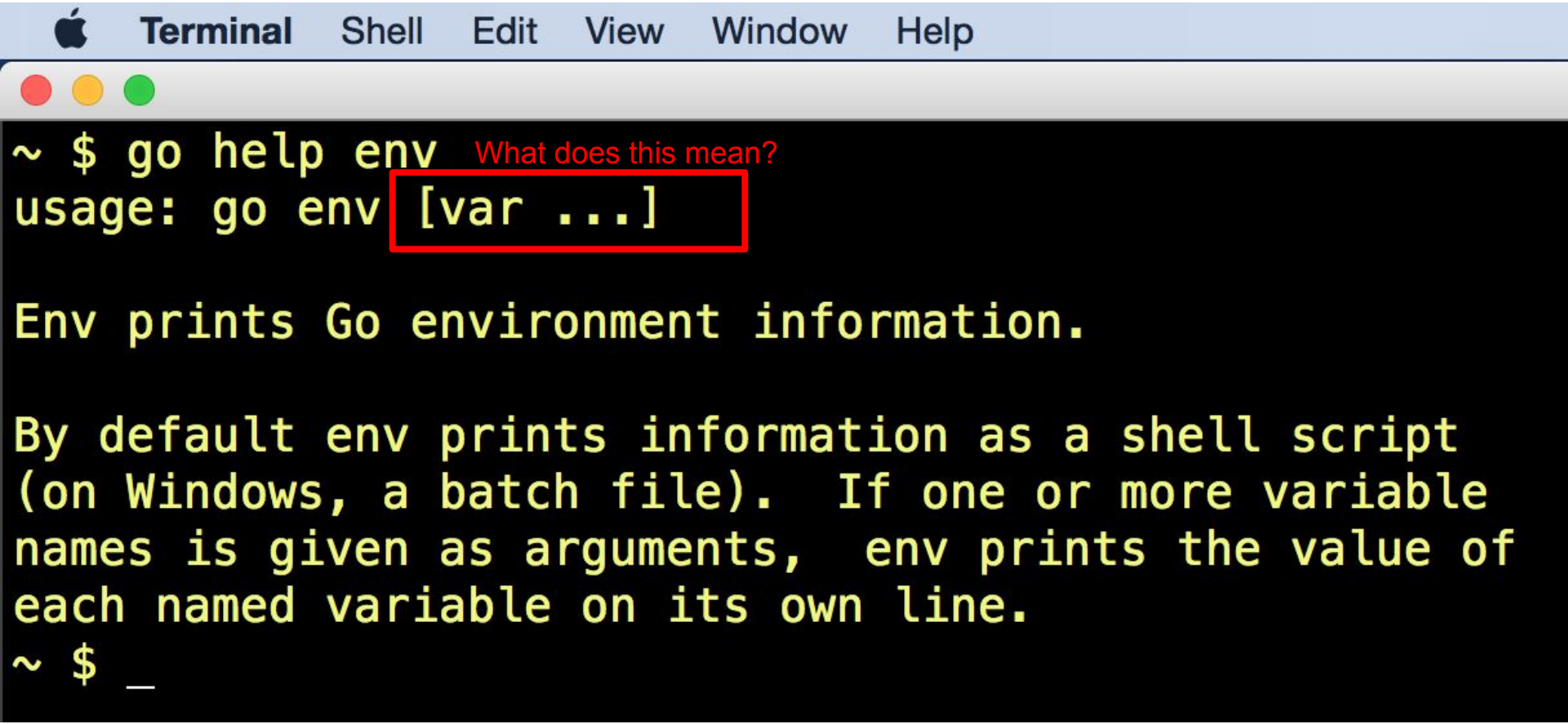
Why doesn’t this work?


Terminal Shell Edit View Window Help
Sat 6:18 PM
tm002 — bash — 129×38
~ $ go help
Go is a tool for managing Go source code.
Usage:
go command [arguments]
The commands are:
build compile packages and dependencies
clean remove object files
env print Go environment information
fix run go tool fix on packages
fmt run gofmt on package sources
generate generate Go files by processing source
get download and install packages and dependencies
install compile and install packages and dependencies
list list packages
run compile and run Go program
test test packages
tool run specified go tool
version print Go version
vet run go tool vet on packages
Use "go help [command]" for more information about a command.
Additional help topics:
c calling between Go and C
filetype file types
gopath GOPATH environment variable
importpath import path syntax
packages description of package lists
testflag description of testing flags
testfunc description of testing functions
Use "go help [topic]" for more information about that topic.
go help [command]
go help [topic]
We already saw this in
“01 Getting Started”

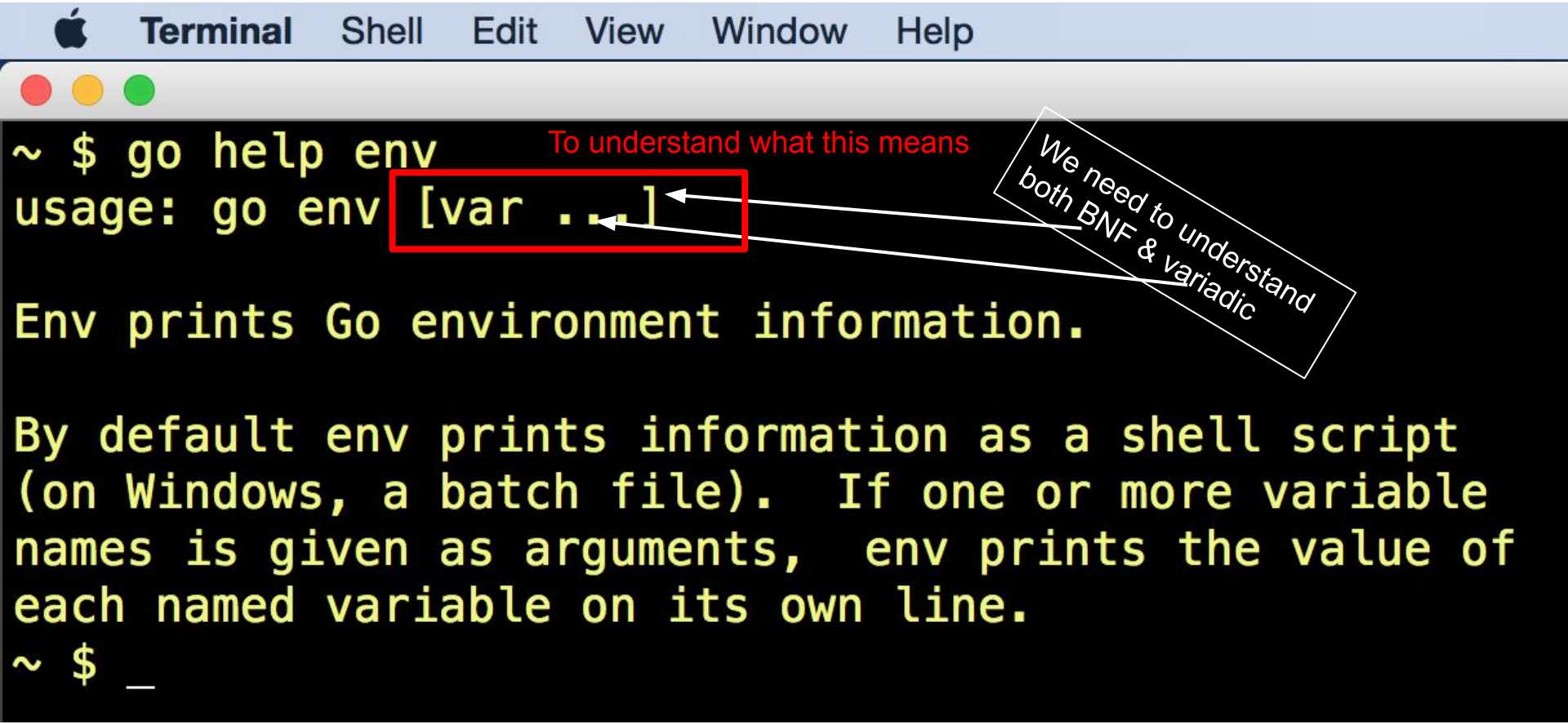
review

use “**go help [command]**” at the terminal
to learn more about the **env** command


Terminal Shell Edit View Window Help
~ $ go help env
usage: go env [var ...]
Env prints Go environment information.
By default env prints information as a shell script
(on Windows, a batch file). If one or more variable
names is given as arguments, env prints the value of
each named variable on its own line.
~ $ _


Terminal Shell Edit View Window Help
~ $ go help env
What does this mean?
usage: go env [var ...]
Env prints Go environment information.
By default env prints information as a shell script
(on Windows, a batch file). If one or more variable
names is given as arguments, env prints the value of
each named variable on its own line.
~ $ _

Go Variadic


Terminal Shell Edit View Window Help
~ $ go help env
usage: go env [var ...]
Env prints Go environment information.
By default env prints information as a shell script
(on Windows, a batch file). If one or more variable
names is given as arguments, env prints the value of
each named variable on its own line.
~ $ _
To understand what this means
We need to understand both BNF & variadic

Go Variadic

bnf

Backus-Naur Form

Backus Naur Form

- Udacity Grammar
- Udacity BNF
  - and on udacity:
    - grammar
    - BNF
    - BNF Quiz
- 0612 TV BNF
- Wikipedia BNF
- Wikipedia EBNF

Backus Naur Form

- Udacity Grammar
- Udacity BNF
  - and on udacity:
    - grammar
    - BNF
    - BNF Quiz
- 0612 TV BNF
- Wikipedia BNF
- Wikipedia EBNF

Usage	Notation
definition	=
concatenation	,
termination	;
termination	. [1]
alternation	|
option	[...]
repetition	{ ... }
grouping	(...)
terminal string	" ... "
terminal string	' ... '
comment	(* ... *)
special sequence	? ... ?
exception	-

variadic

...params
args...

Go Variadic

variadic

Multiple parameters

```
func Greeting(prefix string, who ...string)
Greeting("nobody")
Greeting("hello:", "Joe", "Anna", "Eileen")
```

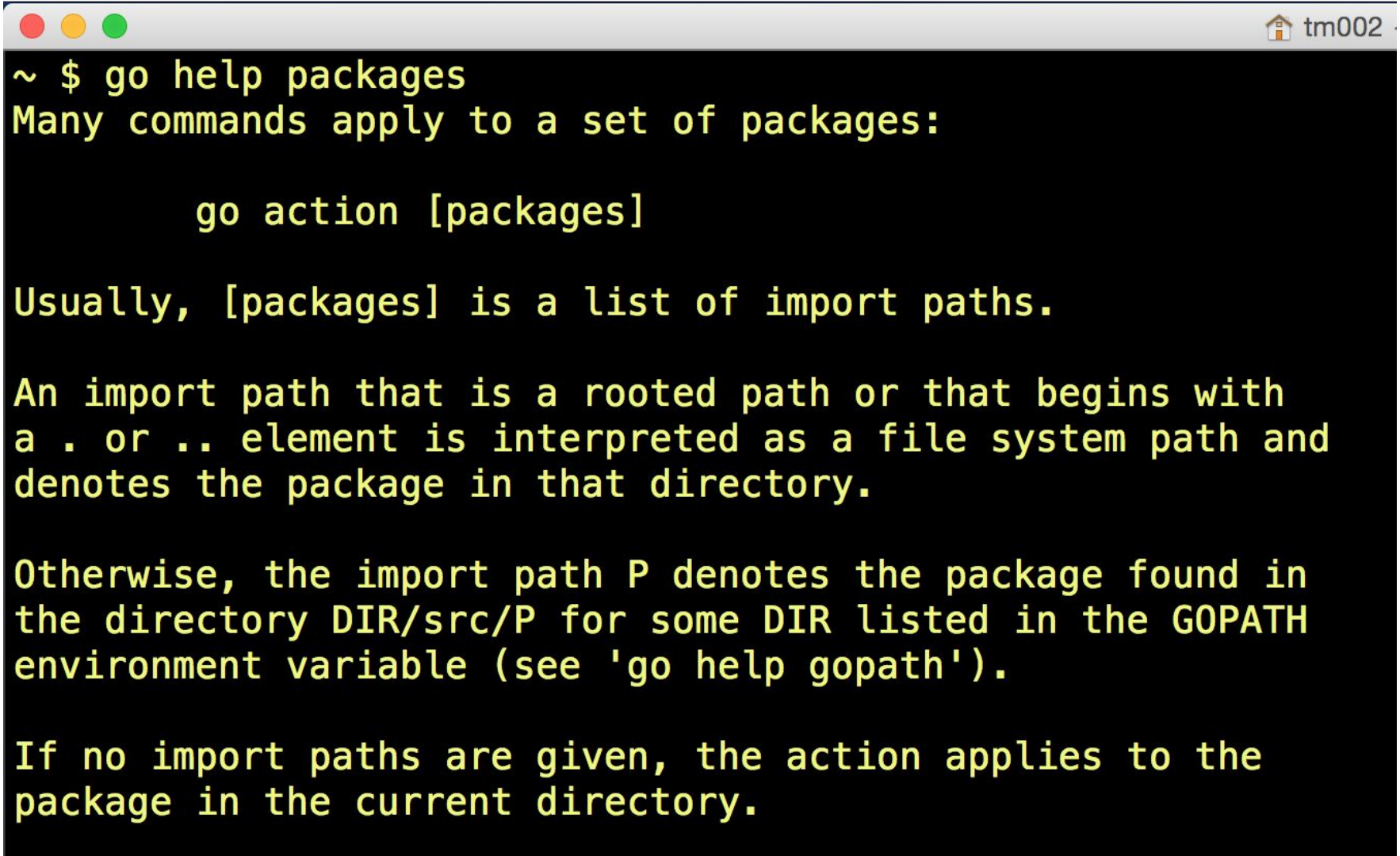
Multiple arguments

```
s := []string{"James", "Jasmine"}
Greeting("goodbye:", s...)
```

Go Variadic

exercise

use “**go help [topic]**” at the terminal
to learn about **packages**


tm002
~ $ go help packages
Many commands apply to a set of packages:
go action [packages]
Usually, [packages] is a list of import paths.
An import path that is a rooted path or that begins with
a . or .. element is interpreted as a file system path and
denotes the package in that directory.
Otherwise, the import path P denotes the package found in
the directory DIR/src/P for some DIR listed in the GOPATH
environment variable (see 'go help gopath').
If no import paths are given, the action applies to the
package in the current directory.

```
There are three reserved names for paths that should not be used
for packages to be built with the go tool:

- "main" denotes the top-level package in a stand-alone executable.

- "all" expands to all package directories found in all the GOPATH
trees. For example, 'go list all' lists all the packages on the local
system.

- "std" is like all but expands to just the packages in the standard
Go library.
```

```
An import path is a pattern if it includes one or more "..." wildcards,
each of which can match any string, including the empty string and
strings containing slashes.  Such a pattern expands to all package
directories found in the GOPATH trees with names matching the
patterns.  As a special case, x/... matches x as well as x's subdirectories.
For example, net/... expands to net and packages in its subdirectories.

An import path can also name a package to be downloaded from
a remote repository.  Run 'go help importpath' for details.

Every package in a program must have a unique import path.
By convention, this is arranged by starting each path with a
unique prefix that belongs to you.  For example, paths used
internally at Google all begin with 'google', and paths
denoting remote repositories begin with the path to the code,
such as 'code.google.com/p/project'.

As a special case, if the package list is a list of .go files from a
single directory, the command is applied to a single synthesized
package made up of exactly those files, ignoring any build constraints
in those files and ignoring any other files in the directory.

Directory and file names that begin with "." or "_" are ignored
by the go tool, as are directories named "testdata".
~ $ _
```

exercise

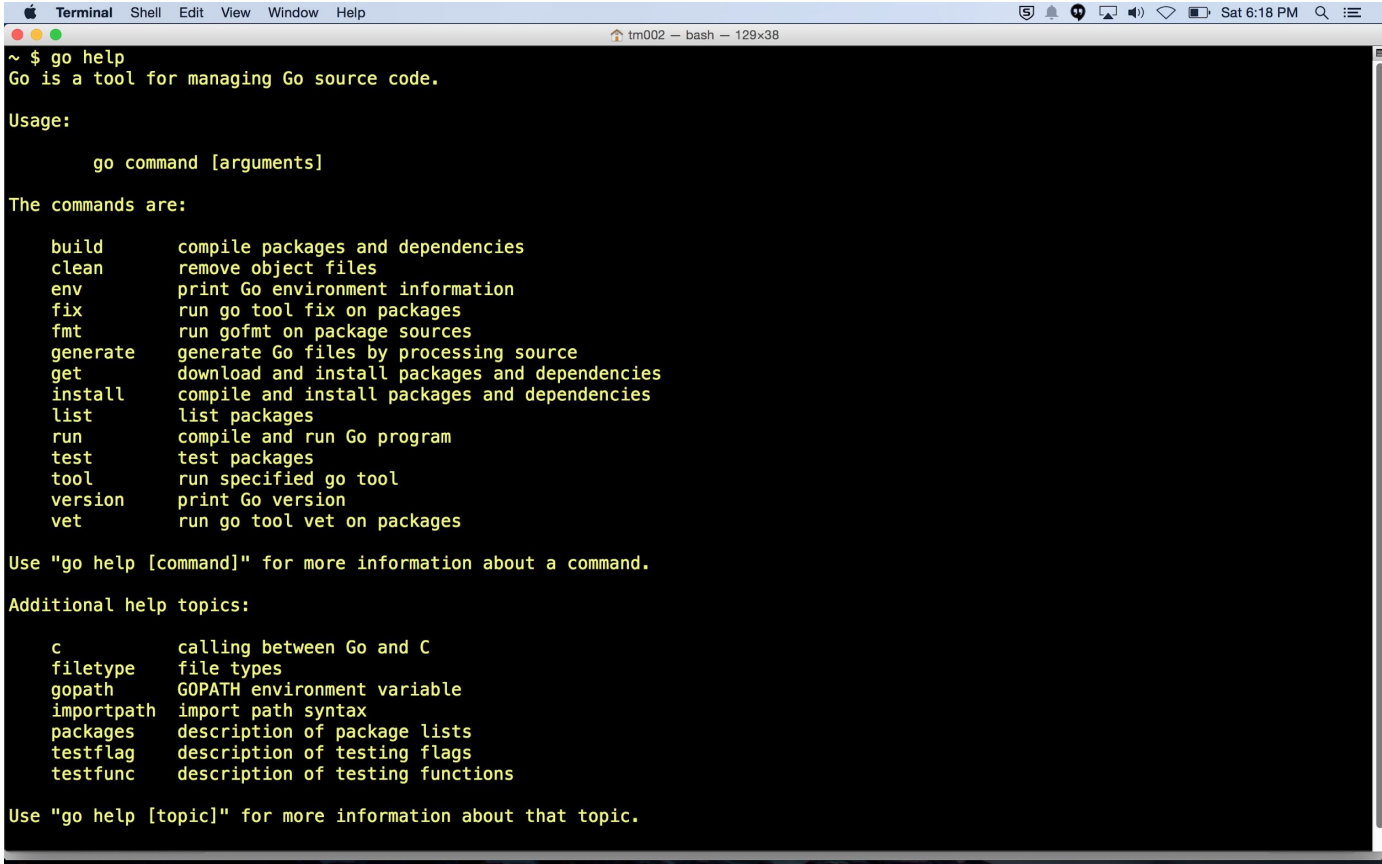
can you find online
the same information we found
using “**go help [topic]**” at the terminal
to learn about **packages**?

exercise

can you find online
the same information we found
using “**go help [topic]**” at the terminal
to learn about **packages**?

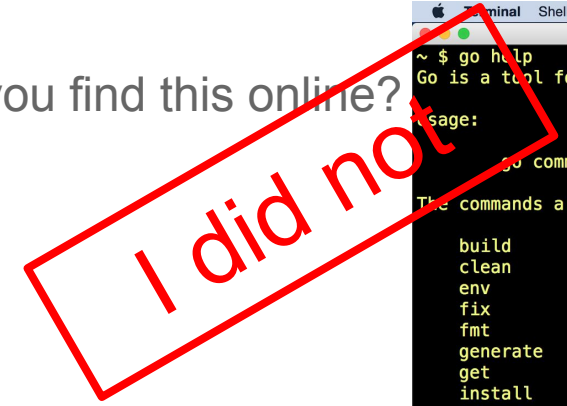
exercise

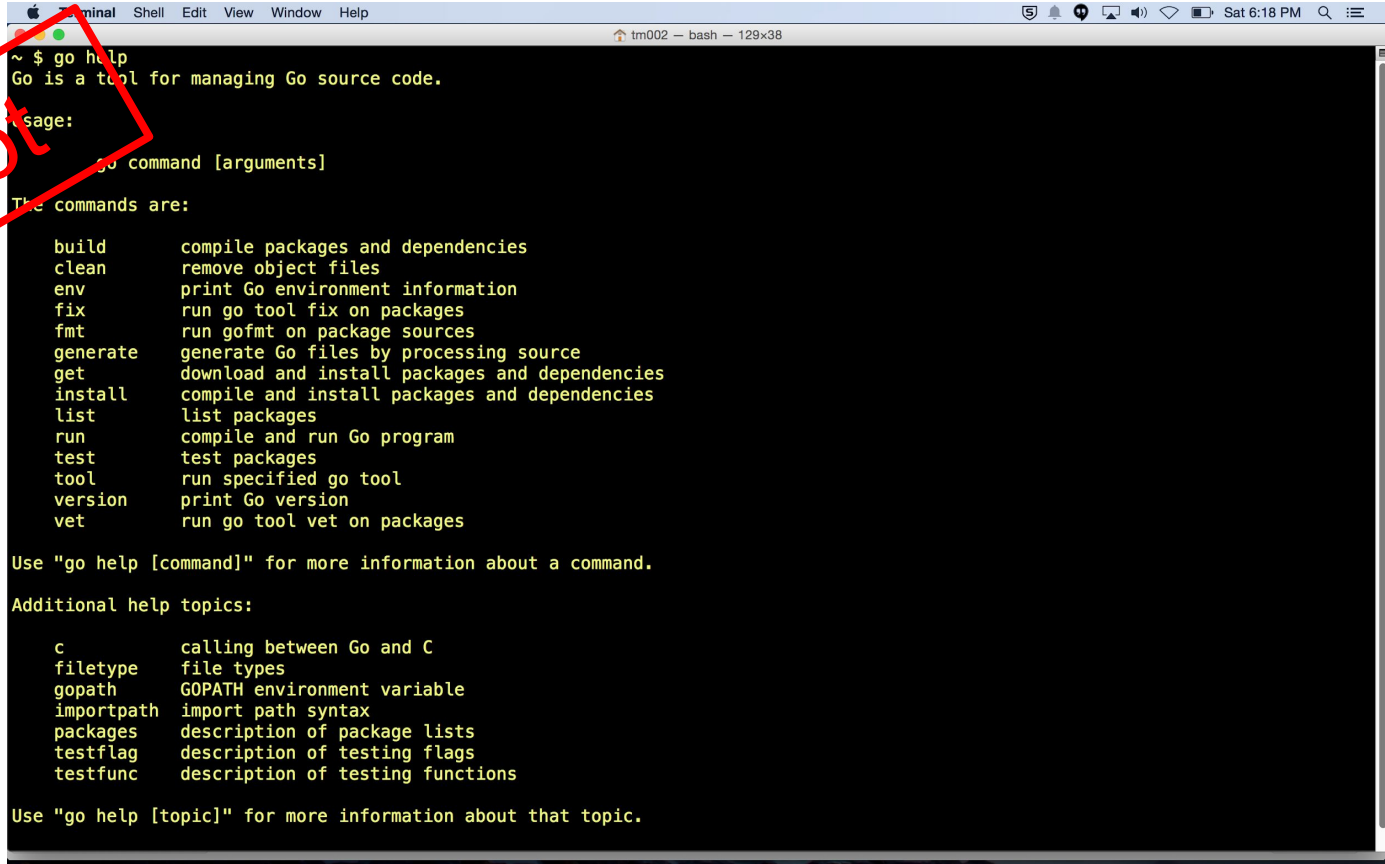
can you find this online?



exercise

can you find this online?



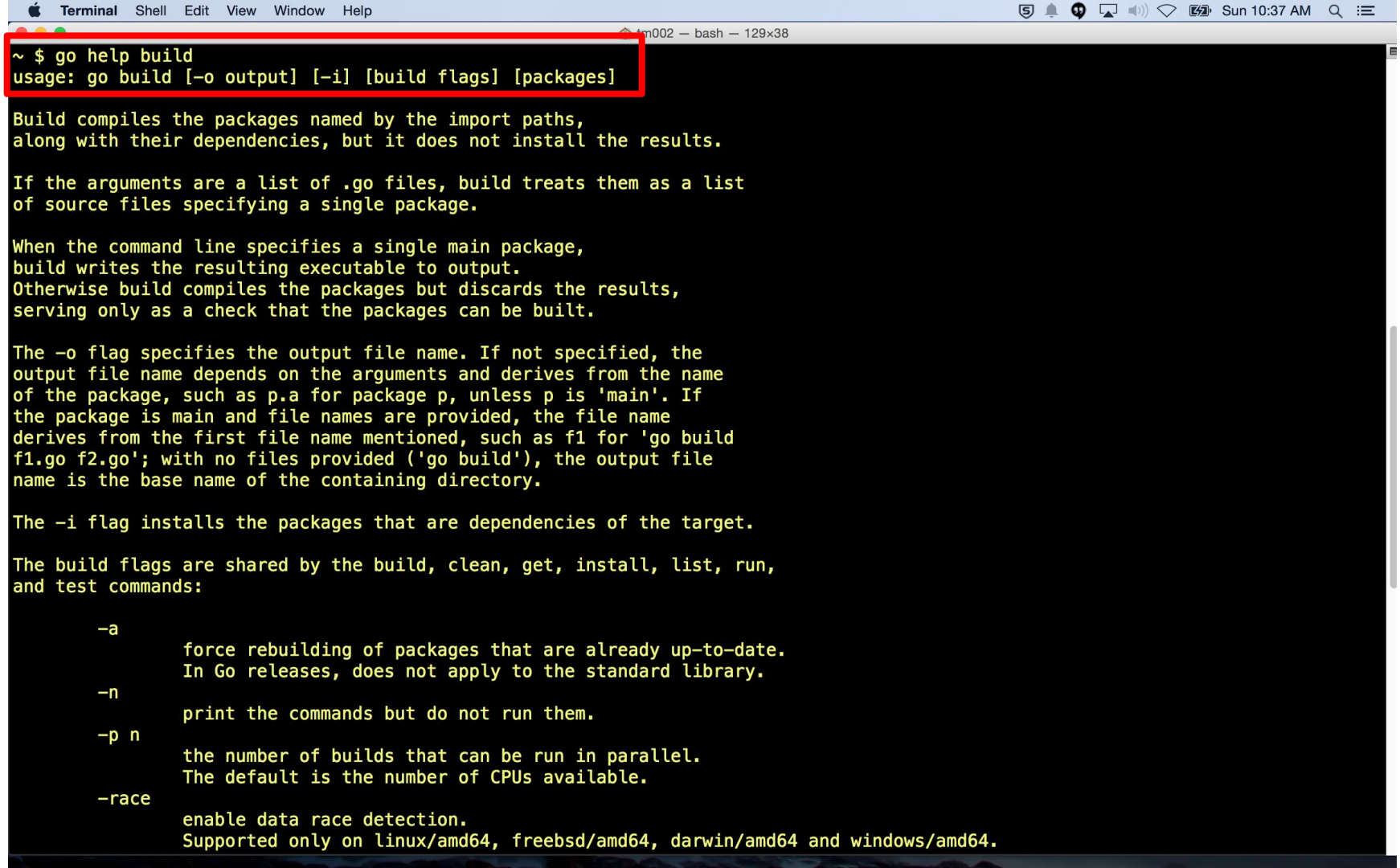


some information is only available
at the terminal, it seems

finding information sometimes requires digging around

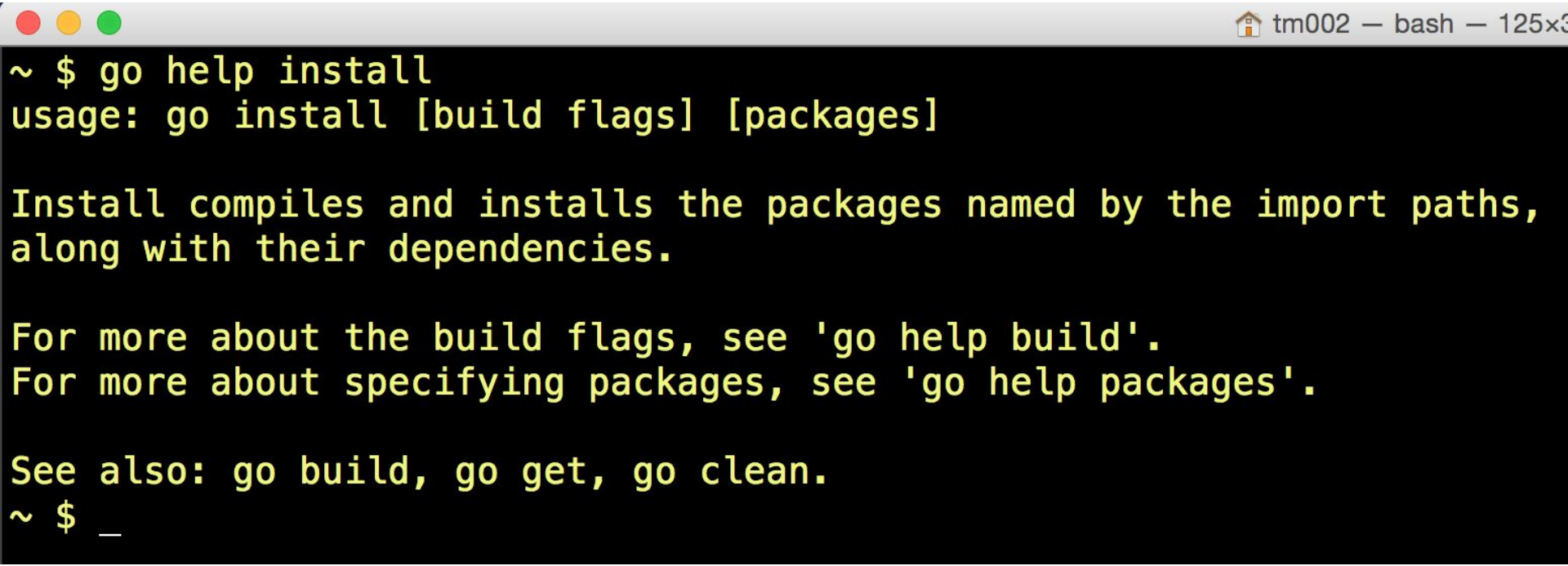
exercise

use “**go help [command]**” at the terminal
to learn more about the **build** command


Terminal Shell Edit View Window Help
Sun 10:37 AM
tm002 — bash — 129×38
~ $ go help build
usage: go build [-o output] [-i] [build flags] [packages]
Build compiles the packages named by the import paths,
along with their dependencies, but it does not install the results.
If the arguments are a list of .go files, build treats them as a list
of source files specifying a single package.
When the command line specifies a single main package,
build writes the resulting executable to output.
Otherwise build compiles the packages but discards the results,
serving only as a check that the packages can be built.
The -o flag specifies the output file name. If not specified, the
output file name depends on the arguments and derives from the name
of the package, such as p.a for package p, unless p is 'main'. If
the package is main and file names are provided, the file name
derives from the first file name mentioned, such as f1 for 'go build
f1.go f2.go'; with no files provided ('go build'), the output file
name is the base name of the containing directory.
The -i flag installs the packages that are dependencies of the target.
The build flags are shared by the build, clean, get, install, list, run,
and test commands:
-a
force rebuilding of packages that are already up-to-date.
In Go releases, does not apply to the standard library.
-n
print the commands but do not run them.
-p n
the number of builds that can be run in parallel.
The default is the number of CPUs available.
-race
enable data race detection.
Supported only on linux/amd64, freebsd/amd64, darwin/amd64 and windows/amd64.

exercise

use “**go help [command]**” at the terminal
to learn more about the **install** command


tm002 — bash — 125×3
~ $ go help install
usage: go install [build flags] [packages]
Install compiles and installs the packages named by the import paths,
along with their dependencies.
For more about the build flags, see 'go help build'.
For more about specifying packages, see 'go help packages'.
See also: go build, go get, go clean.
~ $ _

documentation

godoc.org
vs
golang.org
vs
godoc

what’s the difference?

godoc.org

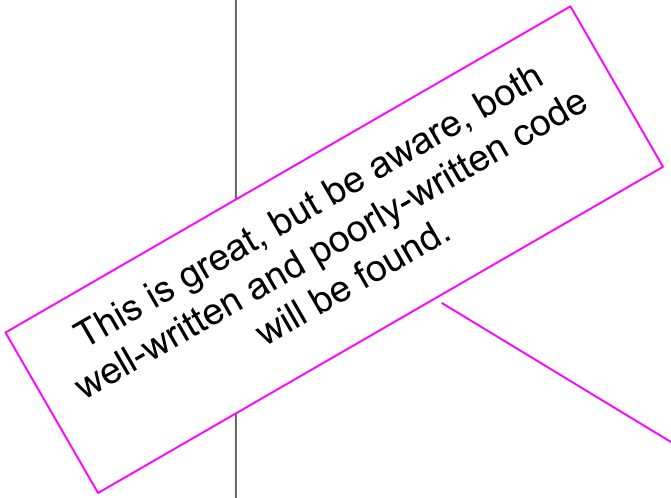
documentation for packages written by anyone, including documentation for official SDK

godoc

documentation at TERMINAL for packages written by anyone, including documentation for official SDK

golang.org

documentation for official SDK only



godoc.org/encoding/csv

The exported fields can be changed to customize the details before the first call to Write or WriteAll.

Comma is the field delimiter.

If UseCRLF is true, the Writer ends each record with \r\n instead of \n.

Example

func NewWriter

```
func NewWriter(w io.Writer) *Writer
```

NewWriter returns a new Writer that writes to w.

func (*Writer) Error

```
func (w *Writer) Error() error
```

Error reports any error that has occurred during a previous Write or Flush.

func (*Writer) Flush

```
func (w *Writer) Flush()
```

Flush writes any buffered data to the underlying io.Writer. To check if an error occurred during the Flush, call Error.

func (*Writer) Write

```
func (w *Writer) Write(record []string) (err error)
```

Writer writes a single CSV record to w along with any necessary quoting. A record is a slice of strings with each string being one field.

func (*Writer) WriteAll

```
func (w *Writer) WriteAll(records [][]string) (err error)
```

WriteAll writes multiple CSV records to w using Write and then calls Flush.

Example

Package csv imports 8 packages (graph) and is imported by 779 packages. Updated 17 days ago. Refresh now. Tools for package owners.

Website Issues | Go Language

Back to top

documentation

godoc

more about godoc
http://godoc.org/golang.org/x/tools/cmd/godoc

tools: golang.org/x/tools/cmd/godoc

Command godoc

I found this by googling: golang godoc

Godoc extracts and generates documentation for Go programs.

It has two modes.

Without the -http flag, it runs in command-line mode and prints plain text documentation to standard output and exits. If both a library package and a command with the same name exists, using the prefix cmd/ will force documentation on the command rather than the library package. If the -src flag is specified, godoc prints the exported interface of a package in Go source form, or the implementation of a specific exported language entity:

```
godoc fmt               # documentation for package fmt
godoc fmt Printf        # documentation for fmt.Printf
godoc cmd/go            # force documentation for the go command
godoc -src fmt          # fmt package interface in Go source form
godoc -src fmt Printf   # implementation of fmt.Printf
```

In command-line mode, the -q flag enables search queries against a godoc running as a webserver. If no

Things sometimes migrate from tools to the standard SDK library

numbers. An improvement to the DNS resolver on Linux and BSD systems has removed the cgo requirement for programs that do name lookups. The go/types package has been moved to the standard library from the golang.org/x/tools repository. (The new go/constant and go/importer packages are also a result of this move.) The reflect package has added the ArrayOf and FuncOf functions, analogous to the existing SliceOf function.

https://blog.golang.org/go1.5

The flags are:

```
-v
```

exercise

can you find this at the terminal?

tools: golang.org/x/tools/cmd/godoc

Command godoc

Godoc extracts and generates documentation for Go programs.

It has two modes.

Without the -http flag, it runs in command-line mode and prints plain text documentation to standard output and exits. If both a library package and a command with the same name exists, using the prefix cmd/ will force documentation on the command rather than the library package. If the -src flag is specified, godoc prints the exported interface of a package in Go source form, or the implementation of a specific exported language entity:

```
godoc fmt               # documentation for package fmt
godoc fmt Printf        # documentation for fmt.Printf
godoc cmd/go            # force documentation for the go command
godoc -src fmt          # fmt package interface in Go source form
godoc -src fmt Printf   # implementation of fmt.Printf
```

In command-line mode, the -q flag enables search queries against a godoc running as a webserver. If no explicit server address is specified with the -server flag, godoc first tries localhost:6060 and then http://golang.org.

```
godoc -q Reader
godoc -q math.Sin
godoc -server=:6060 -q sin
```

With the -http flag, it runs as a web server and presents the documentation as a web page.

```
godoc -http=:6060
```

Usage:

```
godoc [flag] package [name ...]
```

The flags are:

```
        -v
```

exercise

can you find this at the terminal?

I did not

tools: golang.org/x/tools/cmd/godoc

Command godoc

Godoc extracts and generates documentation for Go programs.

It has two modes.

Without the -http flag, it runs in command-line mode and prints plain text documentation to standard output and exits. If both a library package and a command with the same name exists, using the prefix cmd/ will force documentation on the command rather than the library package. If the -src flag is specified, godoc prints the exported interface of a package in Go source form, or the implementation of a specific exported language entity.

```
godoc fmt                # documentation for package fmt
godoc fmt Printf         # documentation for fmt.Printf
godoc cmd/go             # force documentation for the go command
godoc -src fmt           # fmt package interface in Go source form
godoc -src fmt Printf    # implementation of fmt.Printf
```

In command-line mode, the -q flag enables search queries against a godoc running as a webserver. If no explicit server address is specified with the -server flag, godoc first tries localhost:6060 and then http://golang.org.

```
godoc -q Reader
godoc -q math.Sin
godoc -server=:6060 -q sin
```

With the -http flag, it runs as a web server and presents the documentation as a web page.

```
godoc -http=:6060
```

Usage:

```
godoc [flag] package [name ...]
```

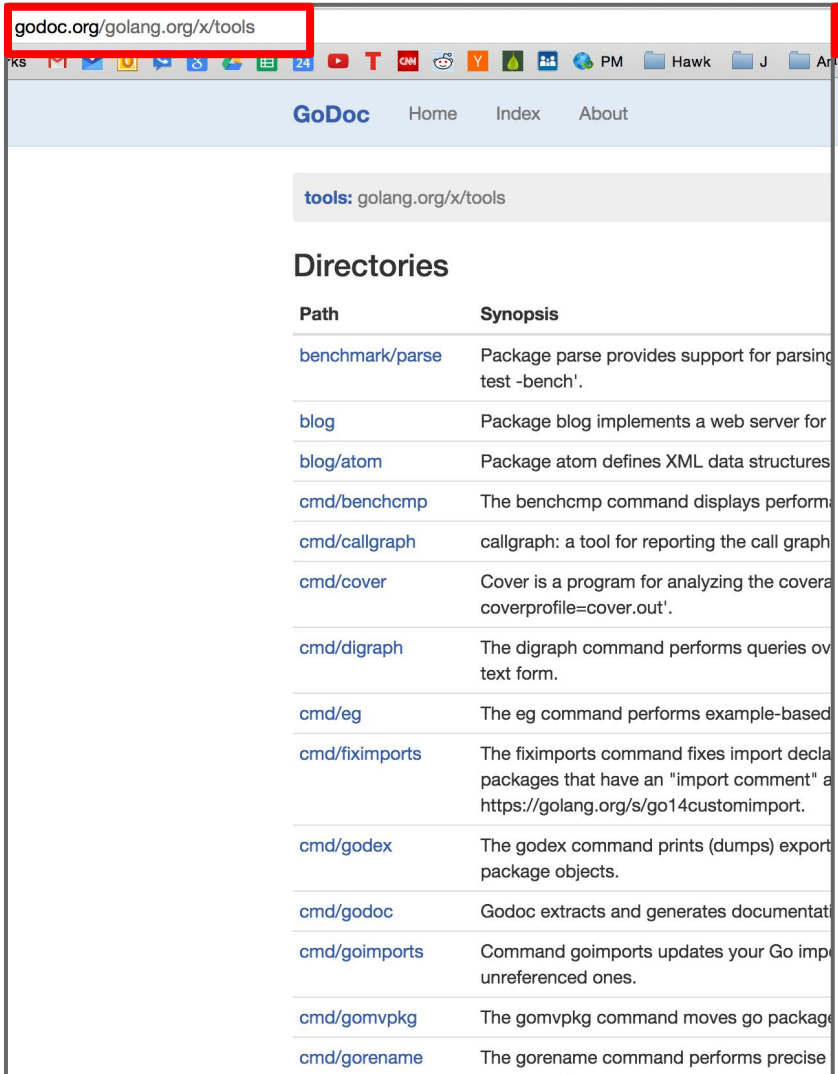
The flags are:

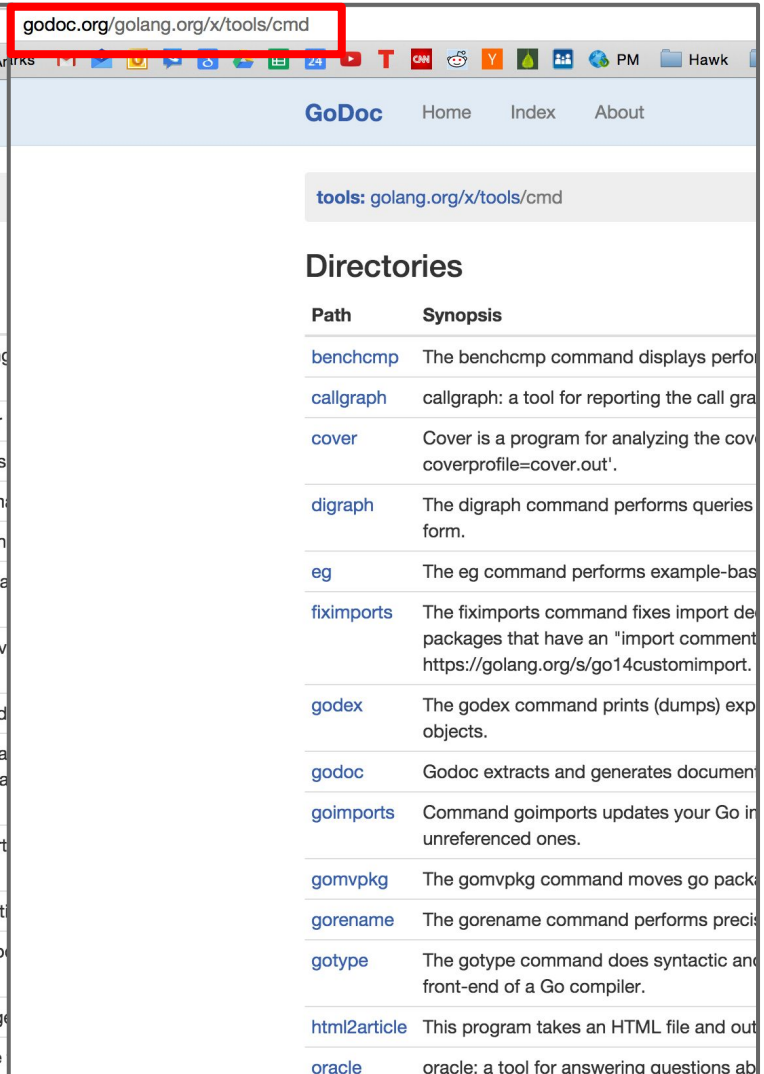
```
        -v
```

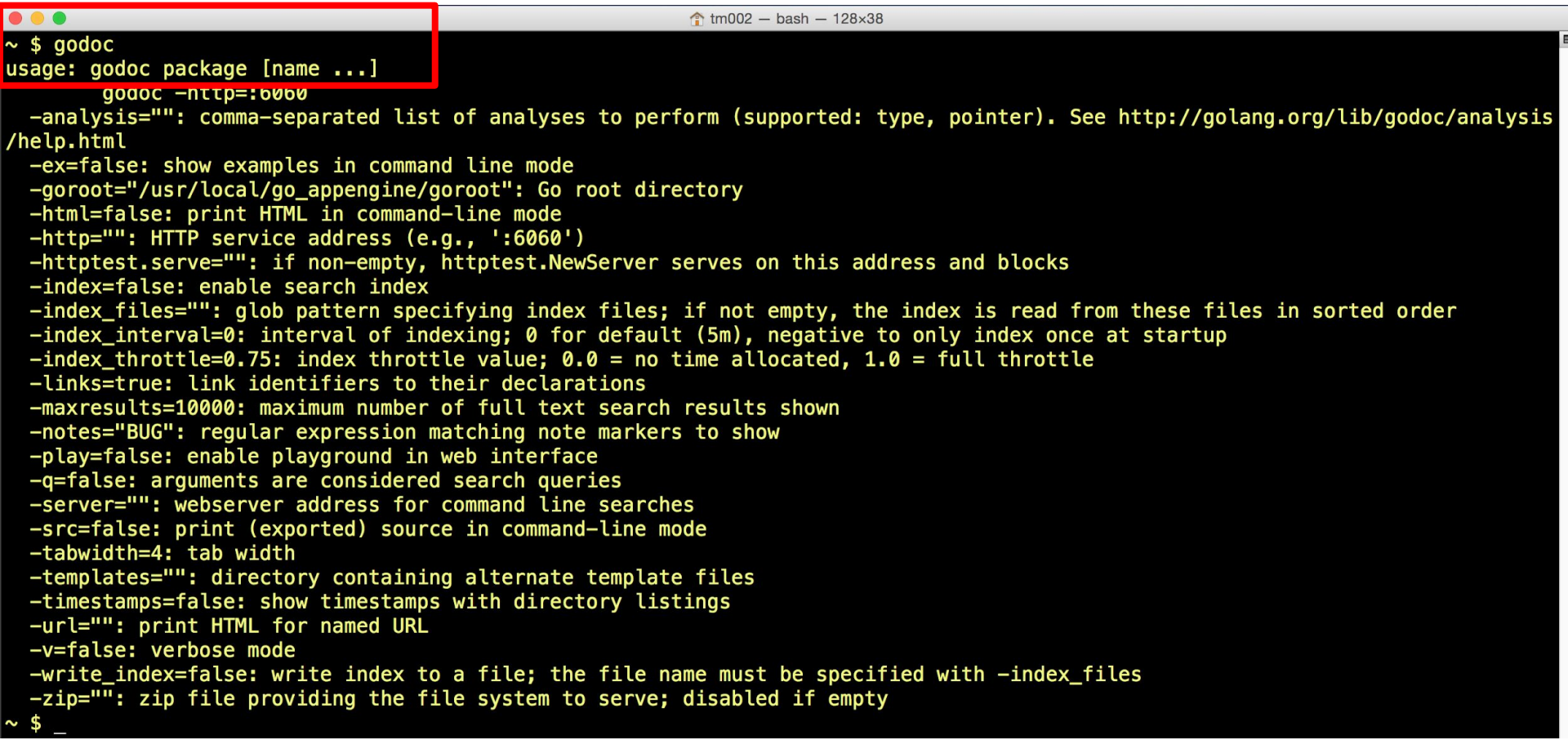
exercise

just out of curiosity,
can you find more tools starting from this URL?

http://godoc.org/golang.org/x/tools/cmd/godoc

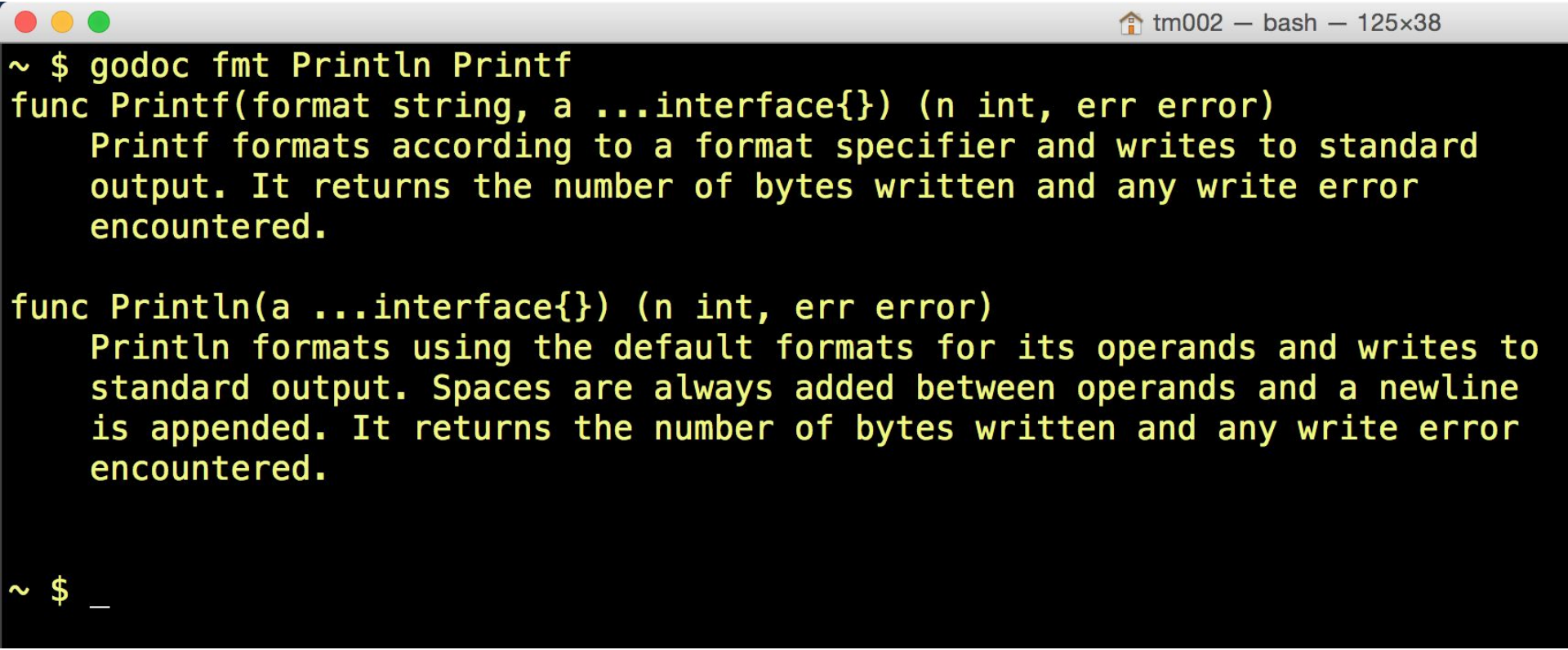

godoc.org/golang.org/x/tools
PM
Hawk
J
GoDoc
Home
Index
About
tools: golang.org/x/tools
Directories
Path
Synopsis
benchmark/parse
Package parse provides support for parsing
test -bench'.
blog
Package blog implements a web server for
blog/atom
Package atom defines XML data structures
cmd/benchcmp
The benchcmp command displays perform
cmd/callgraph
callgraph: a tool for reporting the call graph
cmd/cover
Cover is a program for analyzing the covera
coverprofile=cover.out'.
cmd/digraph
The digraph command performs queries ov
text form.
cmd/eg
The eg command performs example-based
cmd/fiximports
The fiximports command fixes import decla
packages that have an "import comment" a
https://golang.org/s/go14customimport.
cmd/godex
The godex command prints (dumps) export
package objects.
cmd/godoc
Godoc extracts and generates documentati
cmd/goimports
Command goimports updates your Go imp
unreferenced ones.
cmd/gomvpkg
The gomvpkg command moves go package
cmd/gorename
The gorename command performs precise


godoc.org/golang.org/x/tools/cmd
PM
Hawk
GoDoc
Home
Index
About
tools: golang.org/x/tools/cmd
Directories
Path
Synopsis
benchcmp
The benchcmp command displays perfo
callgraph
callgraph: a tool for reporting the call gra
cover
Cover is a program for analyzing the cov
coverprofile=cover.out'.
digraph
The digraph command performs queries
form.
eg
The eg command performs example-bas
fiximports
The fiximports command fixes import de
packages that have an "import comment
https://golang.org/s/go14customimport.
godex
The godex command prints (dumps) exp
objects.
godoc
Godoc extracts and generates document
goimports
Command goimports updates your Go in
unreferenced ones.
gomvpkg
The gomvpkg command moves go packa
gorename
The gorename command performs preci
gotype
The gotype command does syntactic an
front-end of a Go compiler.
html2article
This program takes an HTML file and out
oracle
oracle: a tool for answering questions ab


tm002 — bash — 128×38
~ $ godoc
usage: godoc package [name ...]
godoc -http=:6060
-analysis="": comma-separated list of analyses to perform (supported: type, pointer). See http://golang.org/lib/godoc/analysis/help.html
-ex=false: show examples in command line mode
-goroot="/usr/local/go_appengine/goroot": Go root directory
-html=false: print HTML in command-line mode
-http="": HTTP service address (e.g., ':6060')
-httptest.serve="": if non-empty, httptest.NewServer serves on this address and blocks
-index=false: enable search index
-index_files="": glob pattern specifying index files; if not empty, the index is read from these files in sorted order
-index_interval=0: interval of indexing; 0 for default (5m), negative to only index once at startup
-index_throttle=0.75: index throttle value; 0.0 = no time allocated, 1.0 = full throttle
-links=true: link identifiers to their declarations
-maxresults=10000: maximum number of full text search results shown
-notes="BUG": regular expression matching note markers to show
-play=false: enable playground in web interface
-q=false: arguments are considered search queries
-server="": webserver address for command line searches
-src=false: print (exported) source in command-line mode
-tabwidth=4: tab width
-templates="": directory containing alternate template files
-timestamps=false: show timestamps with directory listings
-url="": print HTML for named URL
-v=false: verbose mode
-write_index=false: write index to a file; the file name must be specified with -index_files
-zip="": zip file providing the file system to serve; disabled if empty
~ $ _

exercise

use “**godoc package [name ...]**”
with multiple names


tm002 — bash — 125×38
~ $ godoc fmt Println Printf
func Printf(format string, a ...interface{}) (n int, err error)
Printf formats according to a format specifier and writes to standard output. It returns the number of bytes written and any write error encountered.
func Println(a ...interface{}) (n int, err error)
Println formats using the default formats for its operands and writes to standard output. Spaces are always added between operands and a newline is appended. It returns the number of bytes written and any write error encountered.
~ $ _

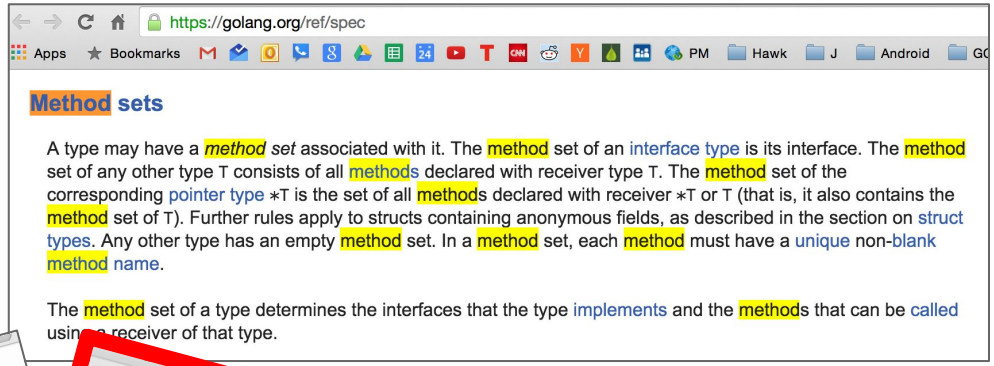
exercise

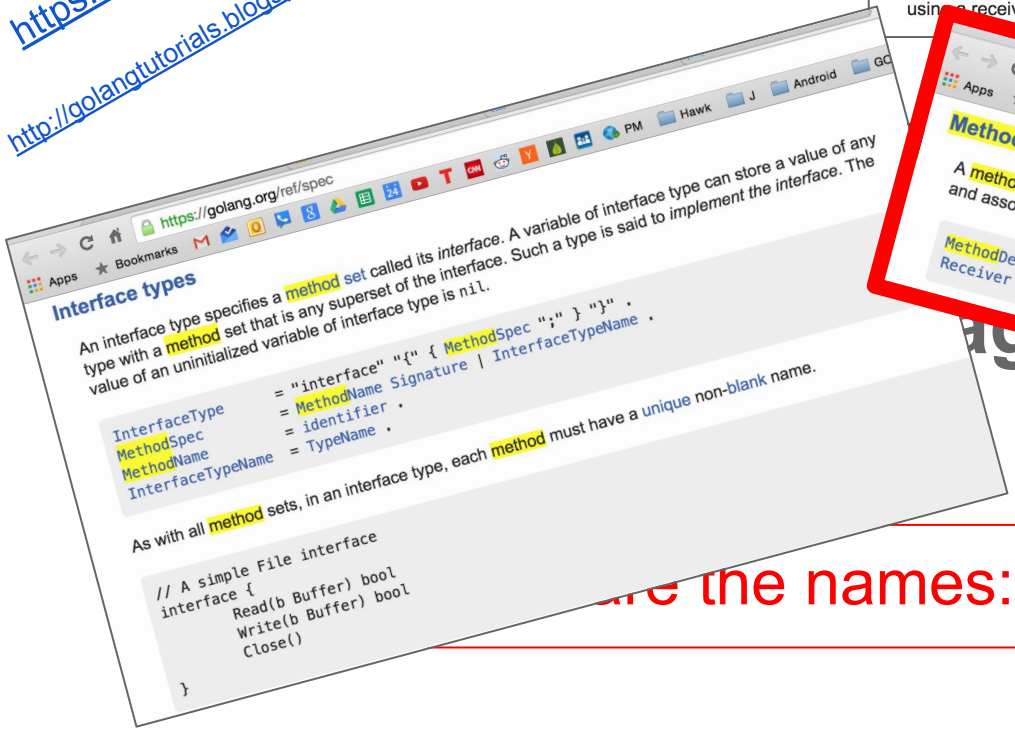
use “**godoc package [name ...]**”
with multiple names

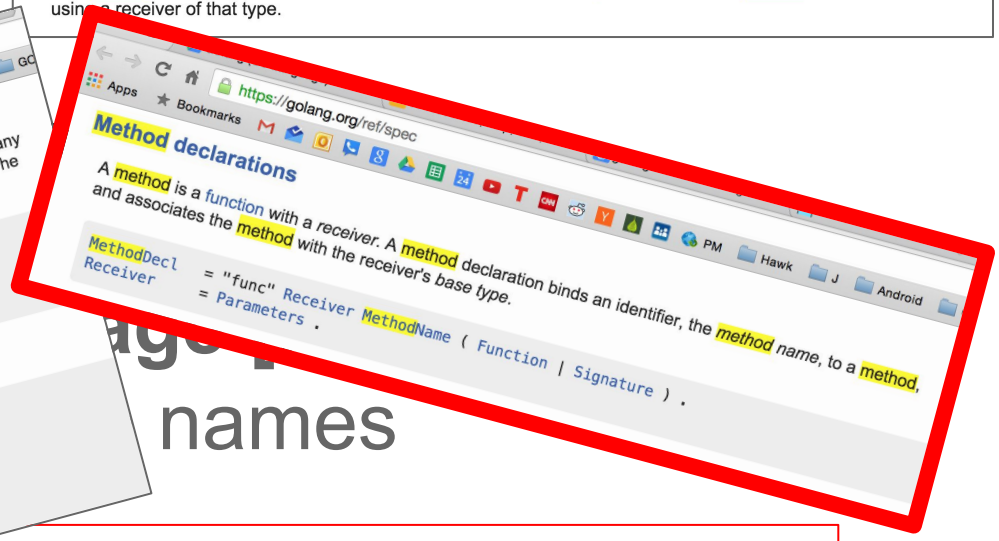
What are the names: functions or methods?

https://gobyexample.com/methods

http://golangtutorials.blogspot.com/2011/06/methods-on-structs.html





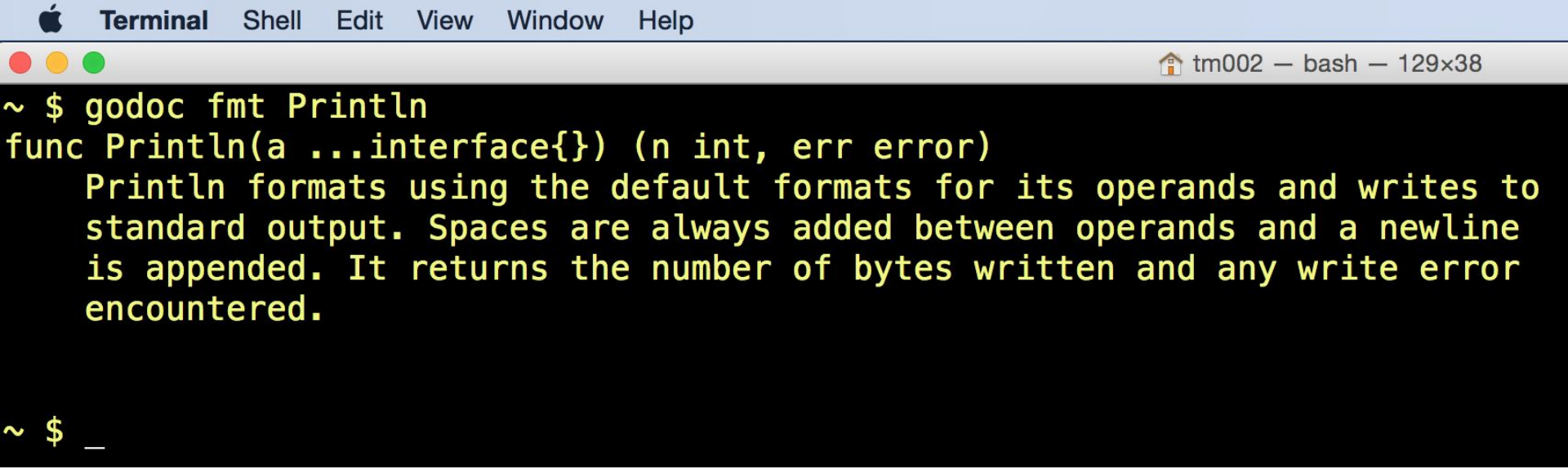


names

the names: functions or methods?

exercise

use “**go doc package [name ...]**”
with one name


Terminal Shell Edit View Window Help
tm002 — bash — 129×38
~ $ godoc fmt Println
func Println(a ...interface{}) (n int, err error)
Println formats using the default formats for its operands and writes to
standard output. Spaces are always added between operands and a newline
is appended. It returns the number of bytes written and any write error
encountered.
~ $ _

Terminal Shell Edit View Window Help

tm002 — bash — 129×38

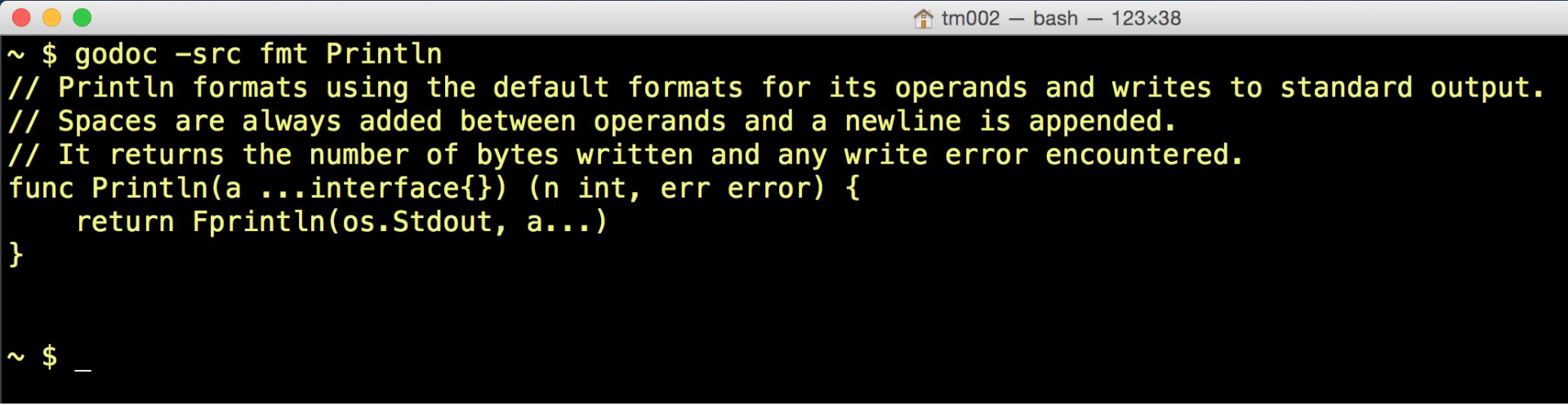
```
~ $ godoc fmt Println
func Println(a ...interface{}) (n int, err error)
    Println formats using the default formats for its operands and writes to
    standard output. Spaces are always added between operands and a newline
    is appended. It returns the number of bytes written and any write error
    encountered.


~ $ _
```

reminder: same as finding it at godoc.org (or golang.org)

but this gives us some code to look at:

exercise

use “**godoc [flag] package [name ...]**” with a flag
to show you at the terminal an example of the code used for
fmt Println


tm002 — bash — 123×38
~ $ godoc -src fmt Println
// Println formats using the default formats for its operands and writes to standard output.
// Spaces are always added between operands and a newline is appended.
// It returns the number of bytes written and any write error encountered.
func Println(a ...interface{}) (n int, err error) {
    return Fprintln(os.Stdout, a...)
}
~ $ _

```
~ $ godoc -src fmt Println
// Println formats using the default formats for its operands and writes to standard output.
// Spaces are always added between operands and a newline is appended.
// It returns the number of bytes written and any write error encountered.
func Println(a ...interface{}) (n int, err error) {
    return Fprintln(os.Stdout, a...)
}


~ $ _
```

godoc vs godoc.org

godoc.org/fmt#Println

https://drive.google.com/

PM Hawk J Android GO jS web python

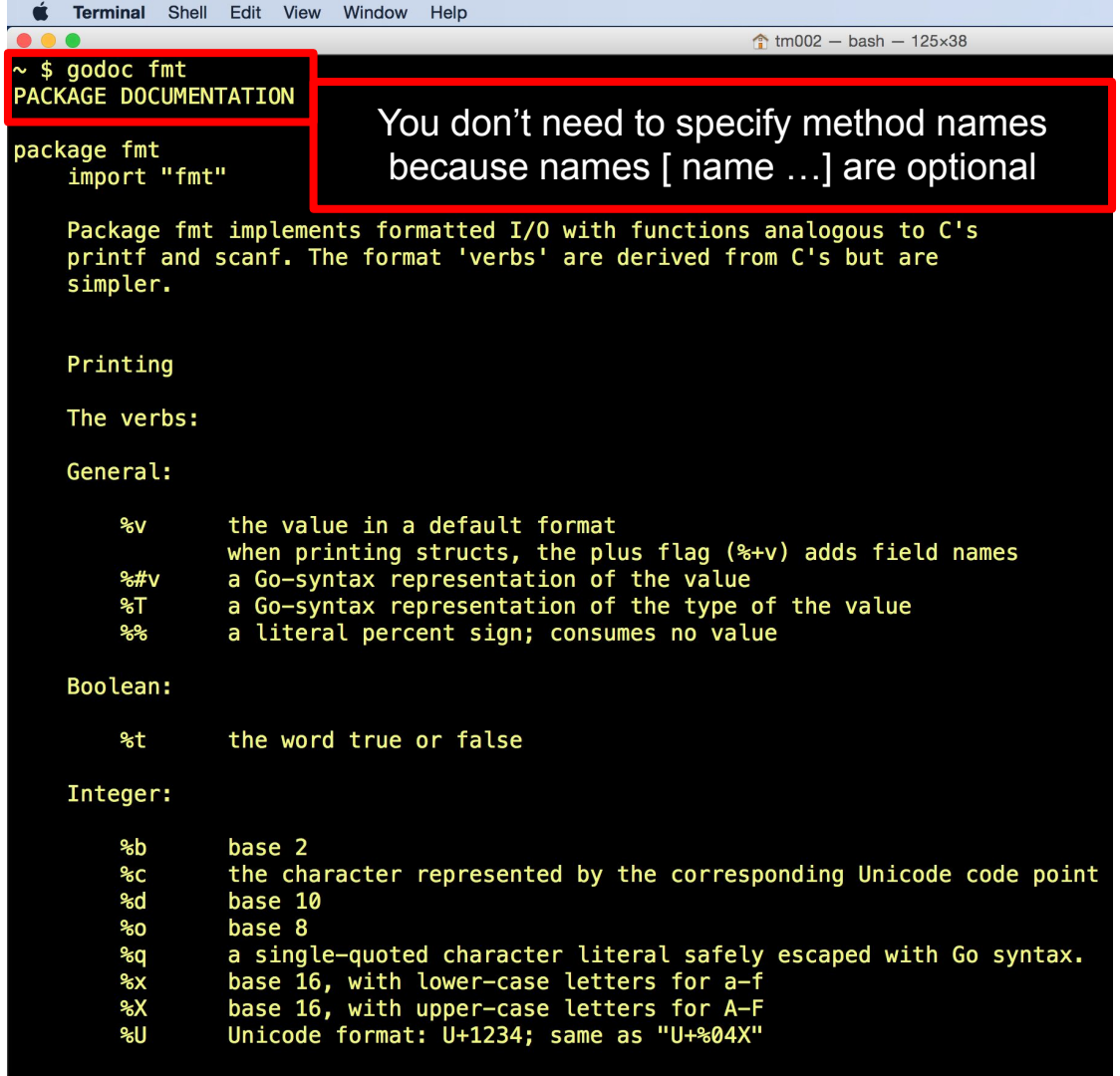
func Println

```
func Println(a ...interface{}) (n int, err error)
```

Println formats using the default formats for its operands and writes to standard output. Spaces are always added between operands and a newline is appended. It returns the number of bytes written and any write error encountered.

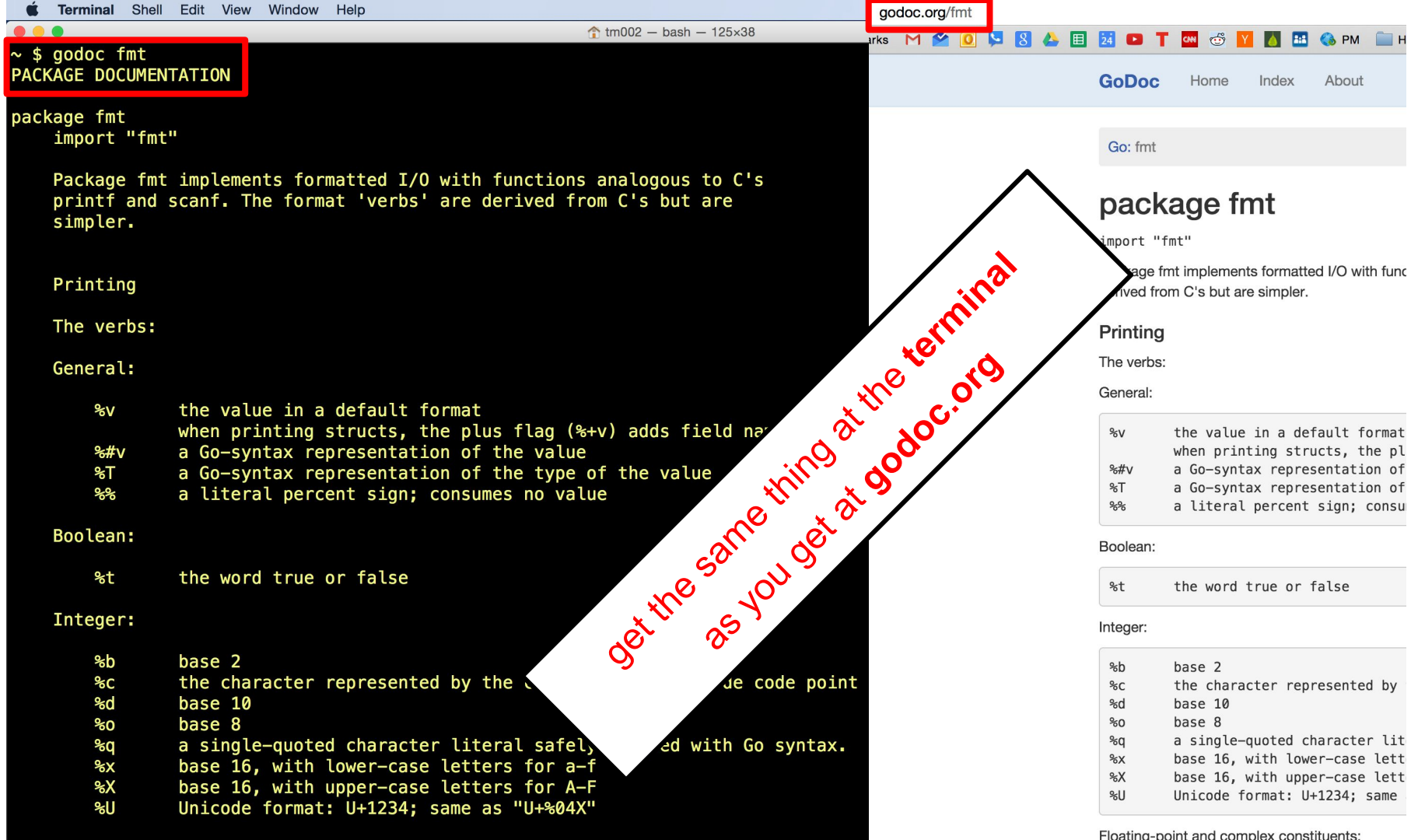
exercise

use “**godoc [flag] package [name ...]**”
to show you at the terminal the package
fmt


Terminal Shell Edit View Window Help
tm002 — bash — 125×38
~ $ godoc fmt
PACKAGE DOCUMENTATION
You don’t need to specify method names because names [name …] are optional
package fmt
import "fmt"
Package fmt implements formatted I/O with functions analogous to C's printf and scanf. The format 'verbs' are derived from C's but are simpler.
Printing
The verbs:
General:
%v the value in a default format
when printing structs, the plus flag (%+v) adds field names
%#v a Go-syntax representation of the value
%T a Go-syntax representation of the type of the value
%% a literal percent sign; consumes no value
Boolean:
%t the word true or false
Integer:
%b base 2
%c the character represented by the corresponding Unicode code point
%d base 10
%o base 8
%q a single-quoted character literal safely escaped with Go syntax.
%x base 16, with lower-case letters for a-f
%X base 16, with upper-case letters for A-F
%U Unicode format: U+1234; same as "U+%04X"

exercise

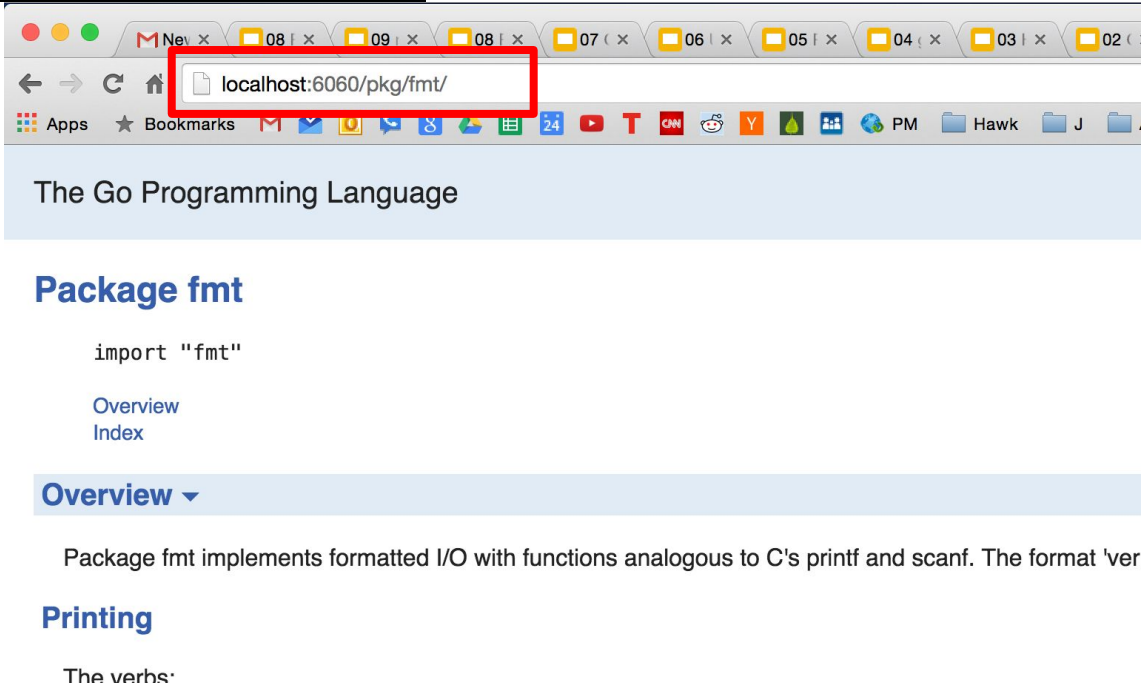
find on godoc.org
the package
fmt


Terminal Shell Edit View Window Help
tm002 — bash — 125×38
~ $ godoc fmt
PACKAGE DOCUMENTATION
package fmt
import "fmt"
Package fmt implements formatted I/O with functions analogous to C's
printf and scanf. The format 'verbs' are derived from C's but are
simpler.
Printing
The verbs:
General:
%v the value in a default format
%#v a Go-syntax representation of the value
%T a Go-syntax representation of the type of the value
%% a literal percent sign; consumes no value
Boolean:
%t the word true or false
Integer:
%b base 2
%c the character represented by the
%d base 10
%o base 8
%q a single-quoted character literal safely
%x base 16, with lower-case letters for a-f
%X base 16, with upper-case letters for A-F
%U Unicode format: U+1234; same as "U+%04X"
get the same thing at the terminal
as you get at godoc.org
godoc.org/fmt
GoDoc
Home
Index
About
Go: fmt
package fmt
import "fmt"
Printing
The verbs:
General:
Boolean:
Integer:

exercise

use “**godoc [flag] package [name ...]**” with a flag *then* lookup fmt package documentation in a browser at localhost:6060


~ $ godoc -http=:6060
_


localhost:6060/pkg/fmt/
Apps
Bookmarks
PM
Hawk
The Go Programming Language
Package fmt
import "fmt"
Overview
Index
Overview
Package fmt implements formatted I/O with functions analogous to C's printf and scanf. The format 'ver
Printing
The verbs:

close server

at terminal: **ctrl + c**

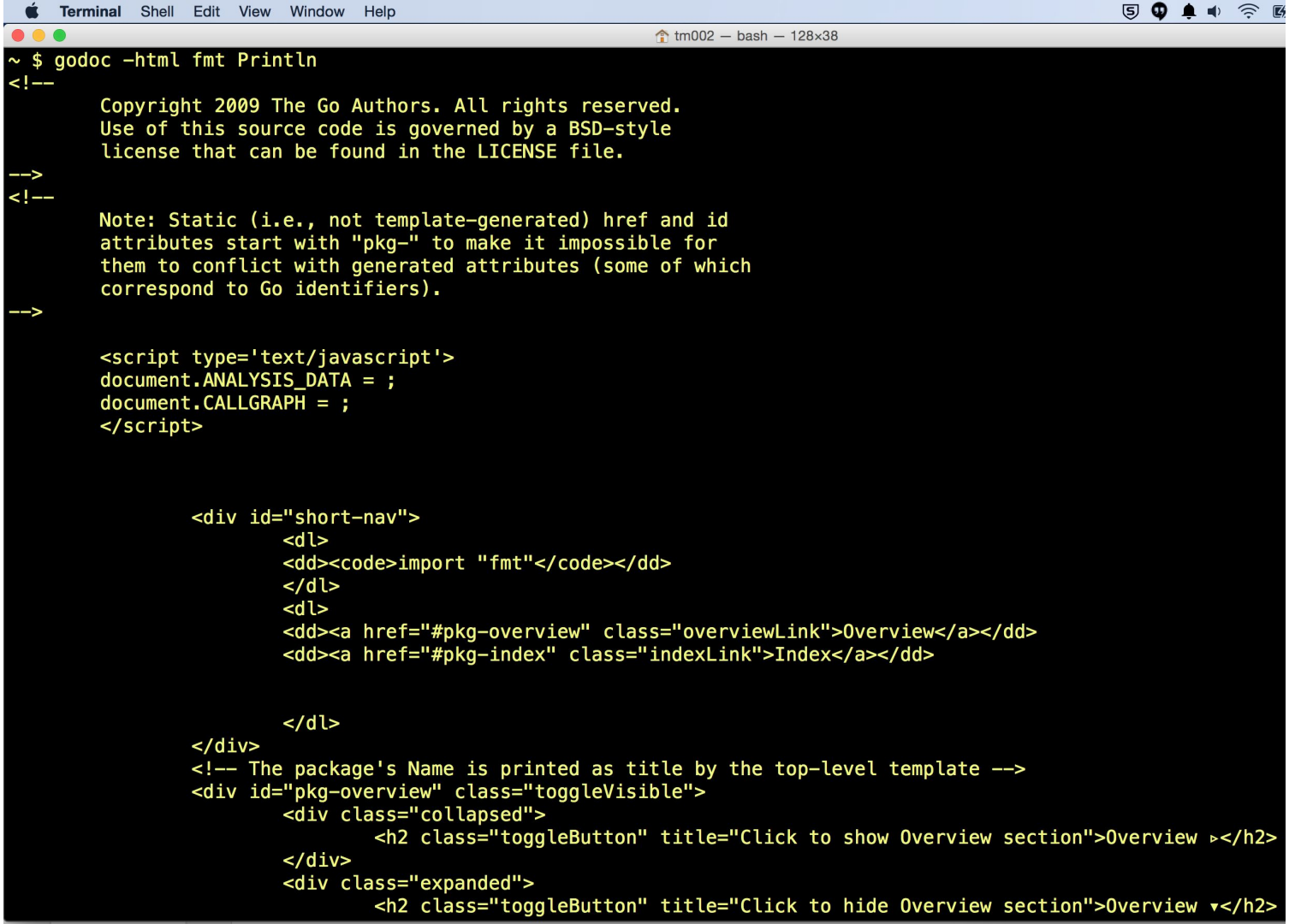
exercise

use “**godoc [flag] package [name ...]**” with a flag
to print html for the **fmt package** in the terminal

```
~ $ godoc -html fmt
<!--
        Copyright 2009 The Go Authors. All rights reserved.
        Use of this source code is governed by a BSD-style
        license that can be found in the LICENSE file.
-->
<!--
        Note: Static (i.e., not template-generated) href and id
        attributes start with "pkg-" to make it impossible for
        them to conflict with generated attributes (some of which
        correspond to Go identifiers).
-->

        <script type='text/javascript'>
        document.ANALYSIS_DATA = ;
        document.CALLGRAPH = ;
        </script>



                <div id="short-nav">
                        <dl>
                        <dd><code>import "fmt"</code></dd>
                        </dl>
                        <dl>
                        <dd><a href="#pkg-overview" class="overviewLink">Overview</a></dd>
                        <dd><a href="#pkg-index" class="indexLink">Index</a></dd>


                        </dl>
                </div>
                <!-- The package's Name is printed as title by the top-level template -->
                <div id="pkg-overview" class="toggleVisible">
                        <div class="collapsed">
                                <h2 class="toggleButton" title="Click to show Overview section">Overview ▹</h2>
                        </div>
                        <div class="expanded">
                                <h2 class="toggleButton" title="Click to hide Overview section">Overview ▾</h2>
```

exercise

use “**godoc [flag] package [name ...]**” with a flag
to print html for the **fmt Println method** in the terminal


Terminal Shell Edit View Window Help
tm002 — bash — 128×38
~ $ godoc -html fmt Println
<!--
Copyright 2009 The Go Authors. All rights reserved.
Use of this source code is governed by a BSD-style
license that can be found in the LICENSE file.
-->
<!--
Note: Static (i.e., not template-generated) href and id
attributes start with "pkg-" to make it impossible for
them to conflict with generated attributes (some of which
correspond to Go identifiers).
-->
<script type='text/javascript'>
document.ANALYSIS_DATA = ;
document.CALLGRAPH = ;
</script>

<dl>
<dd><code>import "fmt"</code></dd>
</dl>
<dl>
<dd>Overview</dd>
<dd>Index</dd>
</dl>

<!-- The package's Name is printed as title by the top-level template -->

<h2 class="toggleButton" title="Click to show Overview section">Overview ▹</h2>

<h2 class="toggleButton" title="Click to hide Overview section">Overview ▾</h2>

exercise

use “**godoc [flag] package [name ...]**” with a flag
to find every “**Reader**” in the documentation

```
~ $ godoc -q Reader
QUERY
        Reader

DID YOU MEAN

        reader

Types
    strings.Reader
    io.Reader
    bytes.Reader
    bufio.Reader
    net/textproto.Reader
    compress/gzip.Reader
    compress/flate.Reader
    debug/dwarf.Reader
    mime/multipart.Reader
    archive/zip.Reader
    archive/tar.Reader
    image/jpeg.Reader
    encoding/csv.Reader

Variables
    crypto/rand.Reader

PACKAGE-LEVEL DECLARATIONS

package bufio
        /src/bufio/bufio.go:31

package bytes
        /src/bytes/reader.go:17

package csv
        /src/encoding/csv/reader.go:102
```

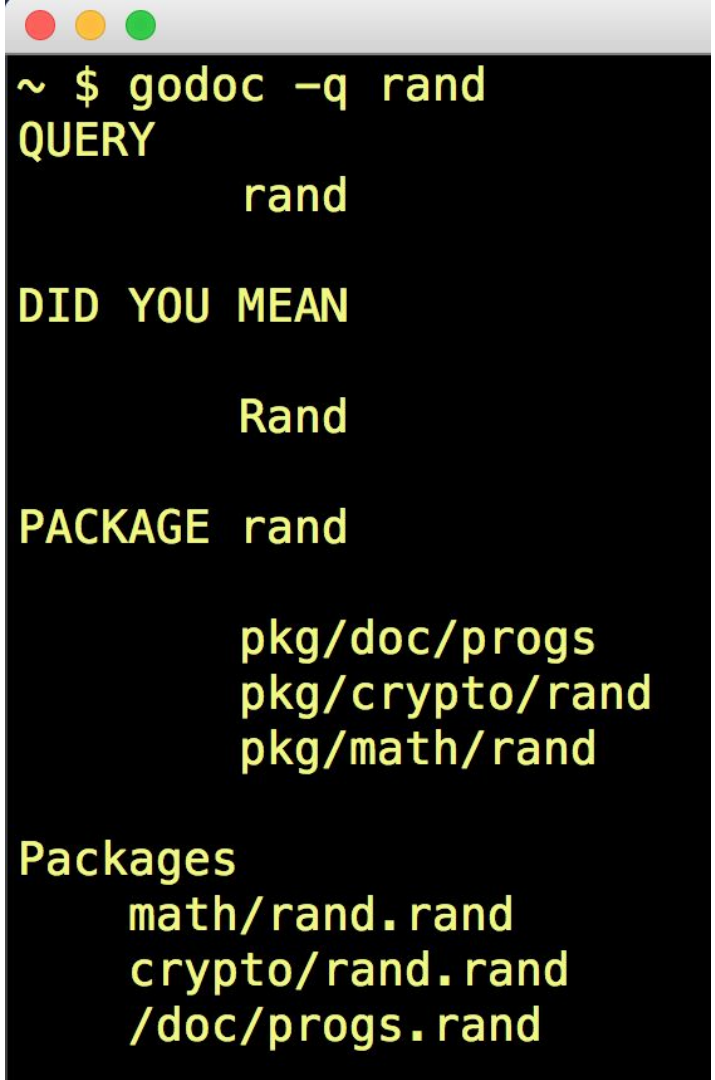
exercise

use “**godoc [flag] package [name ...]**” with a flag
to find every “**Writer**” in the documentation

```
~ $ godoc -q Writer
QUERY
        Writer

DID YOU MEAN

        writer

Types
    io.Writer
    bufio.Writer
    net/textproto.Writer
    compress/gzip.Writer
    compress/flate.Writer
    text/tabwriter.Writer
    mime/multipart.Writer
    archive/zip.Writer
    compress/zlib.Writer
    archive/tar.Writer
    encoding/csv.Writer
    log/syslog.Writer

PACKAGE-LEVEL DECLARATIONS

package bufio
        /src/bufio/bufio.go:479

package csv
        /src/encoding/csv/writer.go:24

package flate
        /src/compress/flate/deflate.go:526

package gzip
        /src/compress/gzip/gzip.go:27

package io
        /src/io/io.go:83
```

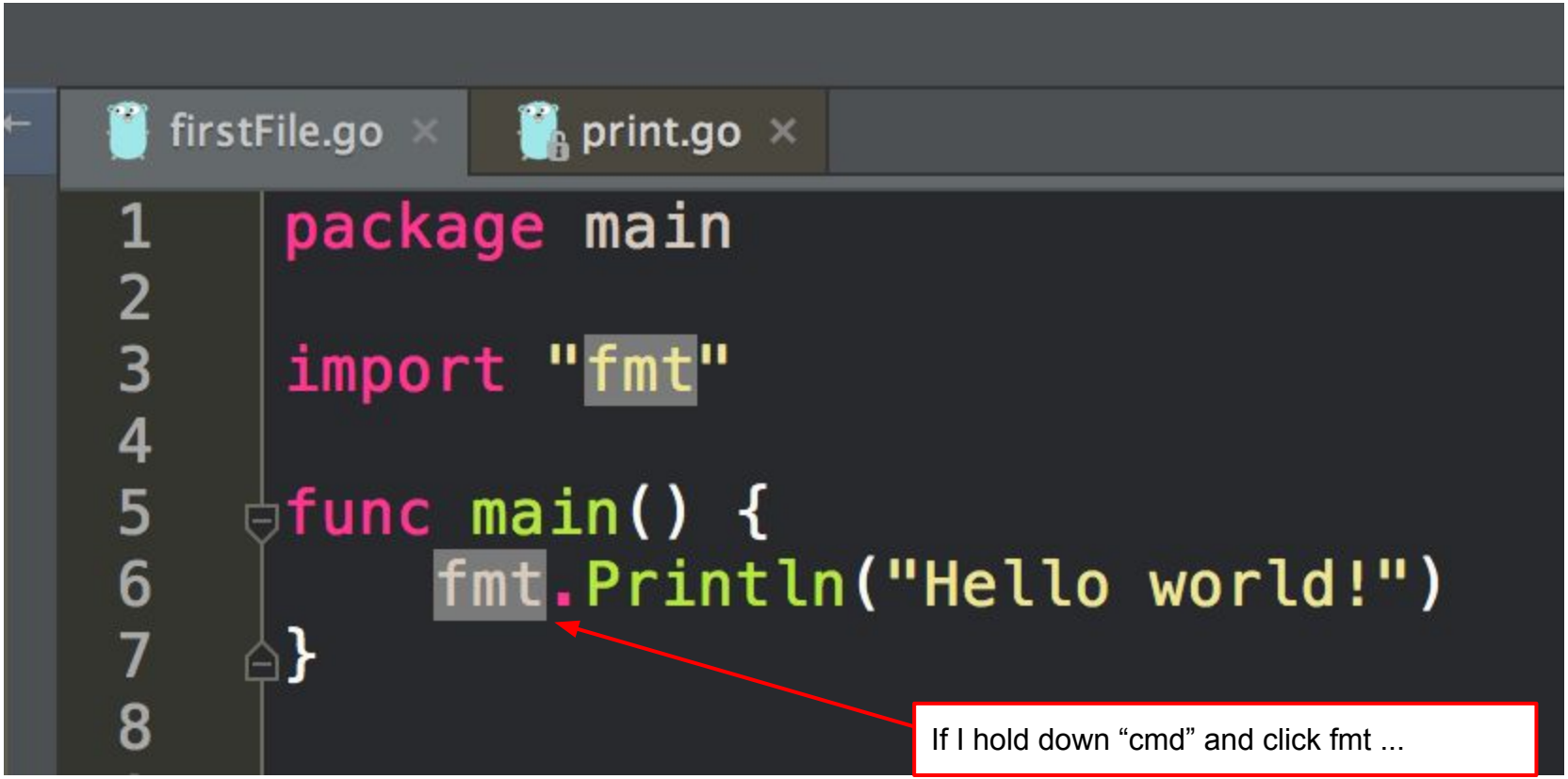
exercise

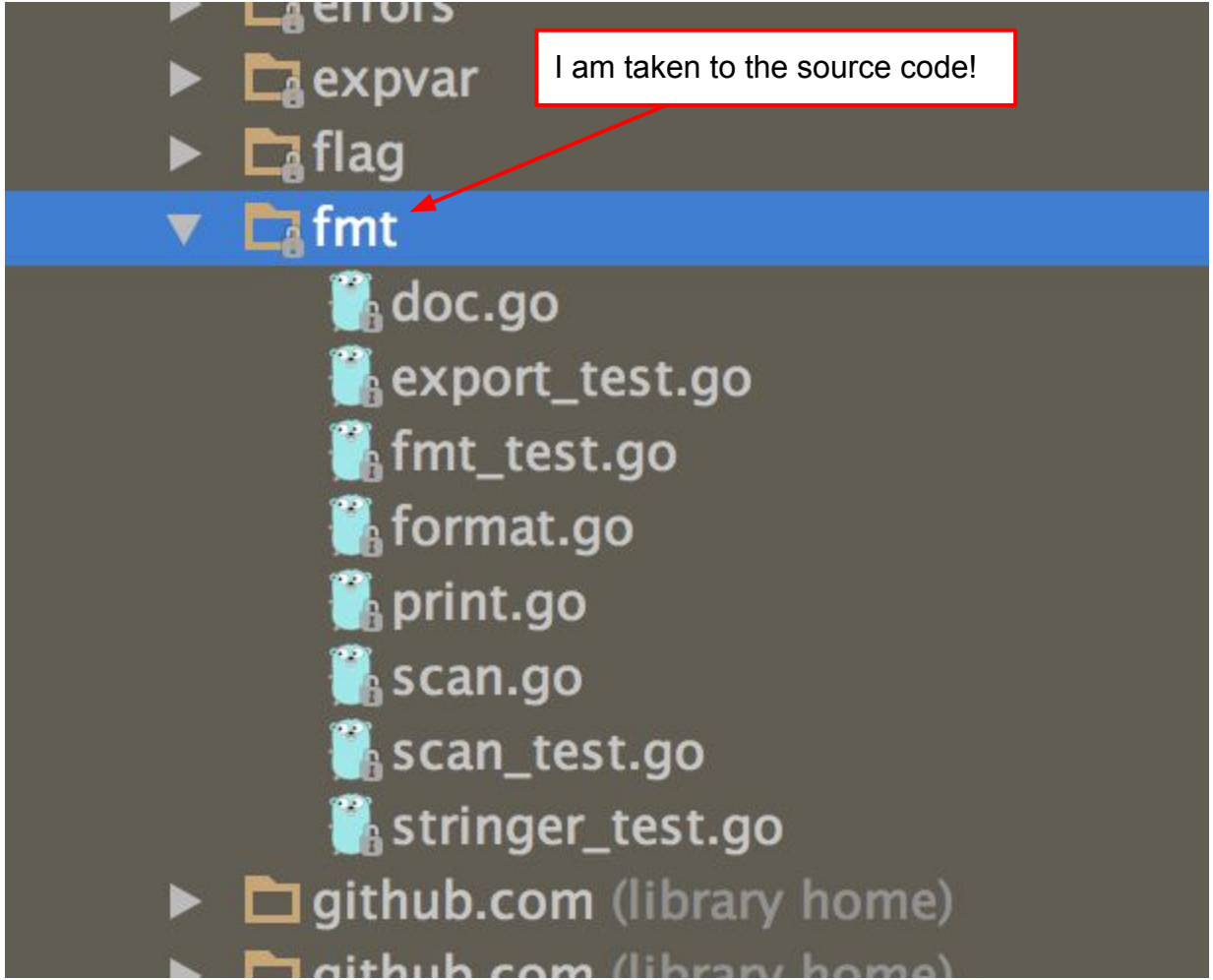
use “**godoc [flag] package [name ...]**” with a flag to find package that gives a random number (search for “**rand**”)


~ $ godoc -q rand
QUERY
rand
DID YOU MEAN
Rand
PACKAGE rand
pkg/doc/progs
pkg/crypto/rand
pkg/math/rand
Packages
math/rand.rand
crypto/rand.rand
/doc/progs.rand

webstorm cmd + click

reminder:
cmd+click in webstorm
allows you to click on a package / method
to view the source code of that package / method


firstFile.go
print.go
1 package main
2
3 import "fmt"
4
5 func main() {
6 fmt.Println("Hello world!")
7 }
8
If I hold down “cmd” and click fmt ...


I am taken to the source code!
expvar
flag
fmt
doc.go
export_test.go
fmt_test.go
format.go
print.go
scan.go
scan_test.go
stringer_test.go
github.com (library home)

Do you know the answer to this question now?

What's the difference between

godoc.org

golang.org

godoc at terminal

go help [command]

go help [topic]

Review Questions

variadic

Define variadic.

What is the difference between code like this:

```
func Greeting(prefix string, who ...string)
Greeting("nobody")
Greeting("hello:", "Joe", "Anna", "Eileen")
```

And like this:

```
s := []string{"James", "Jasmine"}
Greeting("goodbye:", s...)
```

BNF

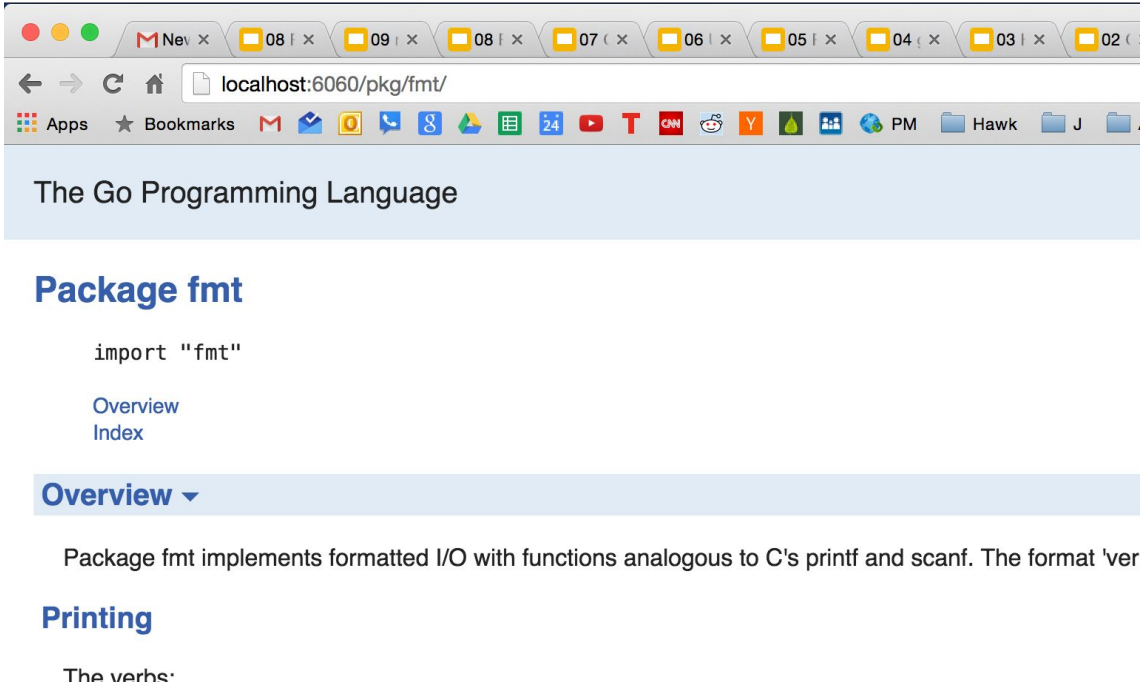
Define BNF

EBNF

Create your own example using EBNF

godoc

- Launch a server showing docs at localhost



godoc

- Using the terminal, find every “**Reader**” in the documentation