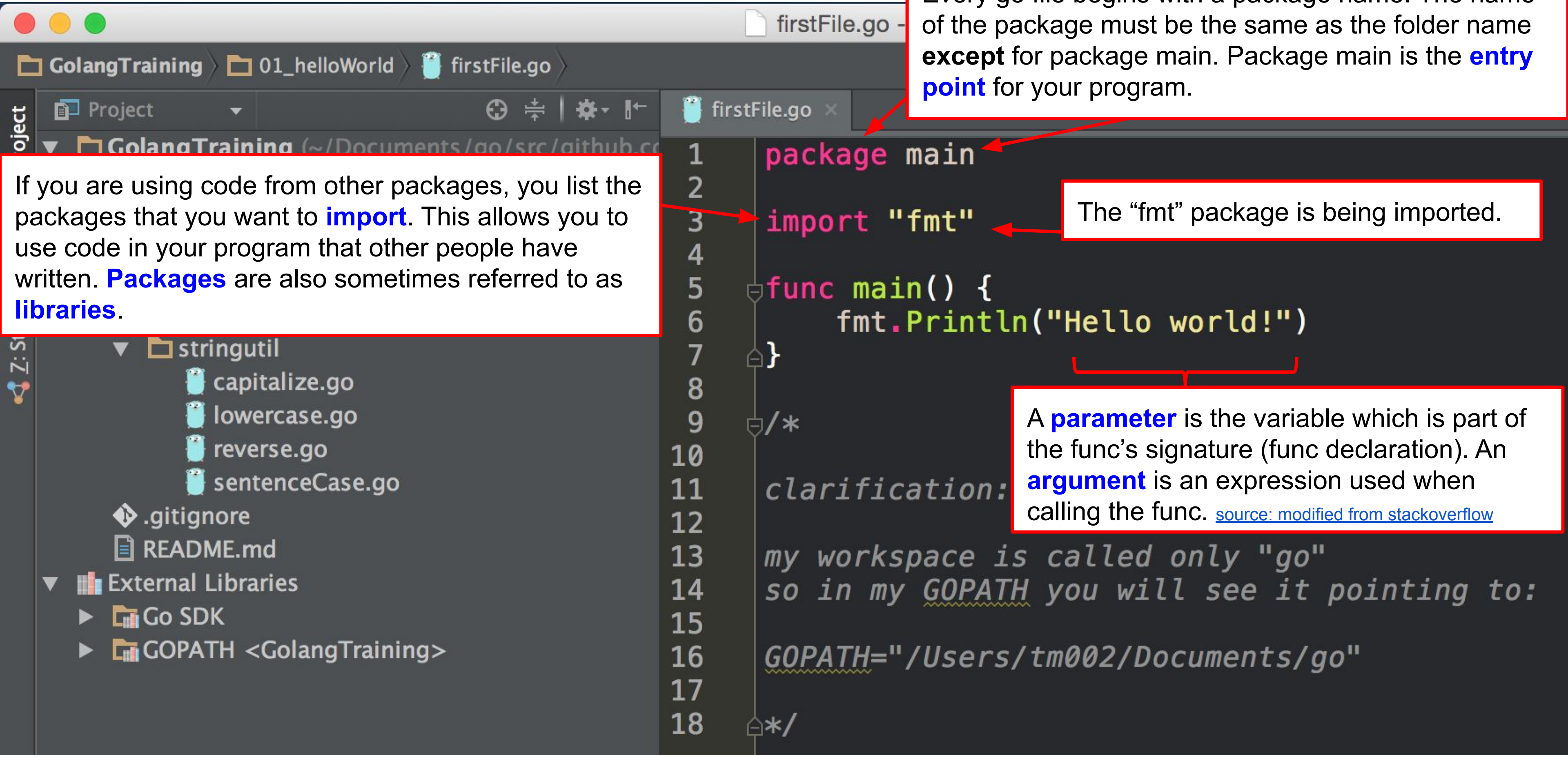
Hello World

code used in training

you can find all of the code shown in this training here:
https://github.com/GoesToEleven/GolangTraining



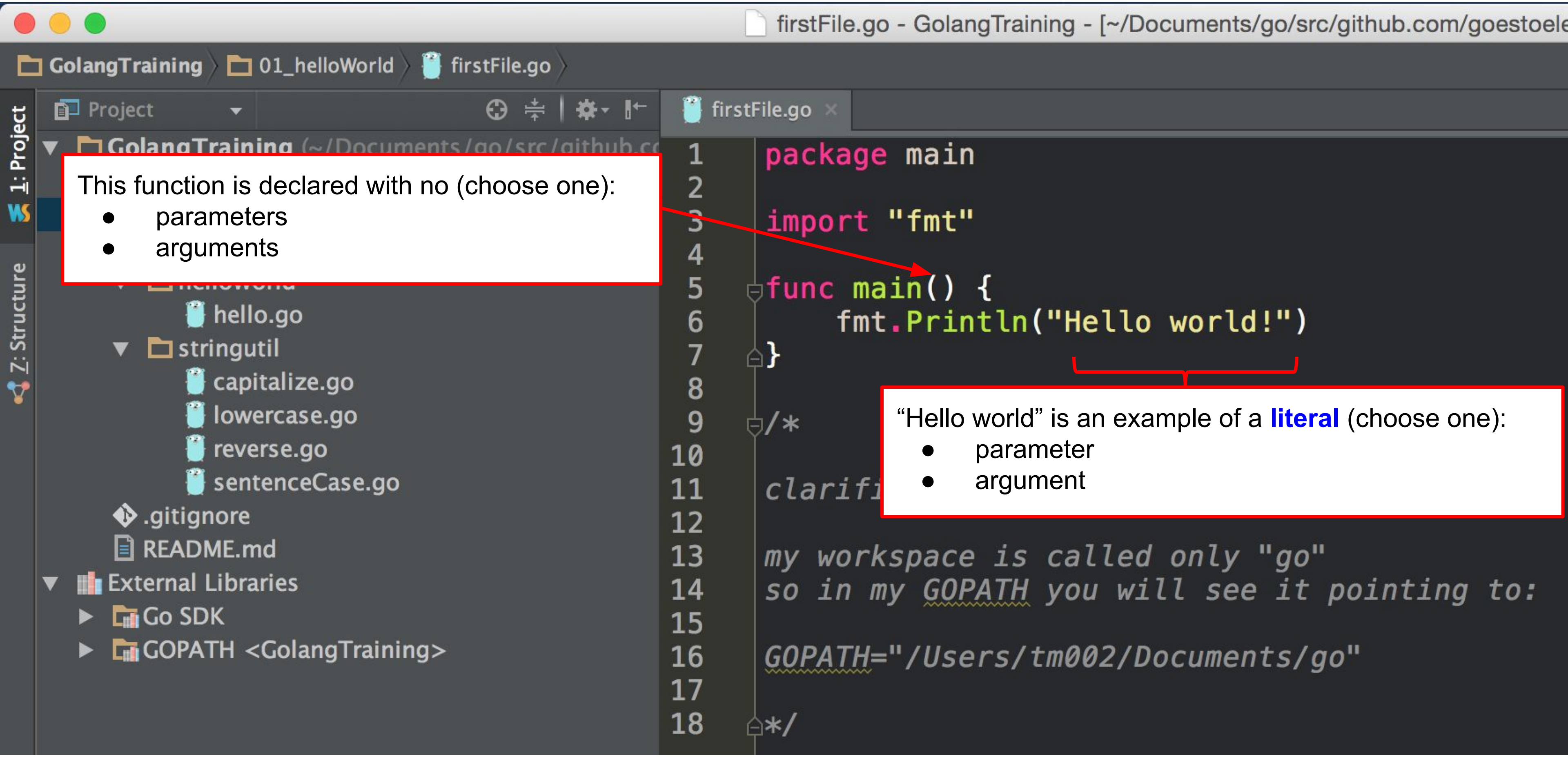
```go
package main

import "fmt"

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

/*

clarification:

my workspace is called only "go"
so in my GOPATH you will see it pointing to:

GOPATH="/Users/tm002/Documents/go"

*/
```

Every go file begins with a package name. The name of the package must be the same as the folder name **except** for package main. Package main is the **entry point** for your program.

If you are using code from other packages, you list the packages that you want to **import**. This allows you to use code in your program that other people have written. **Packages** are also sometimes referred to as **libraries**.

The “fmt” package is being imported.

A **parameter** is the variable which is part of the func’s signature (func declaration). An **argument** is an expression used when calling the func. source: modified from stackoverflow



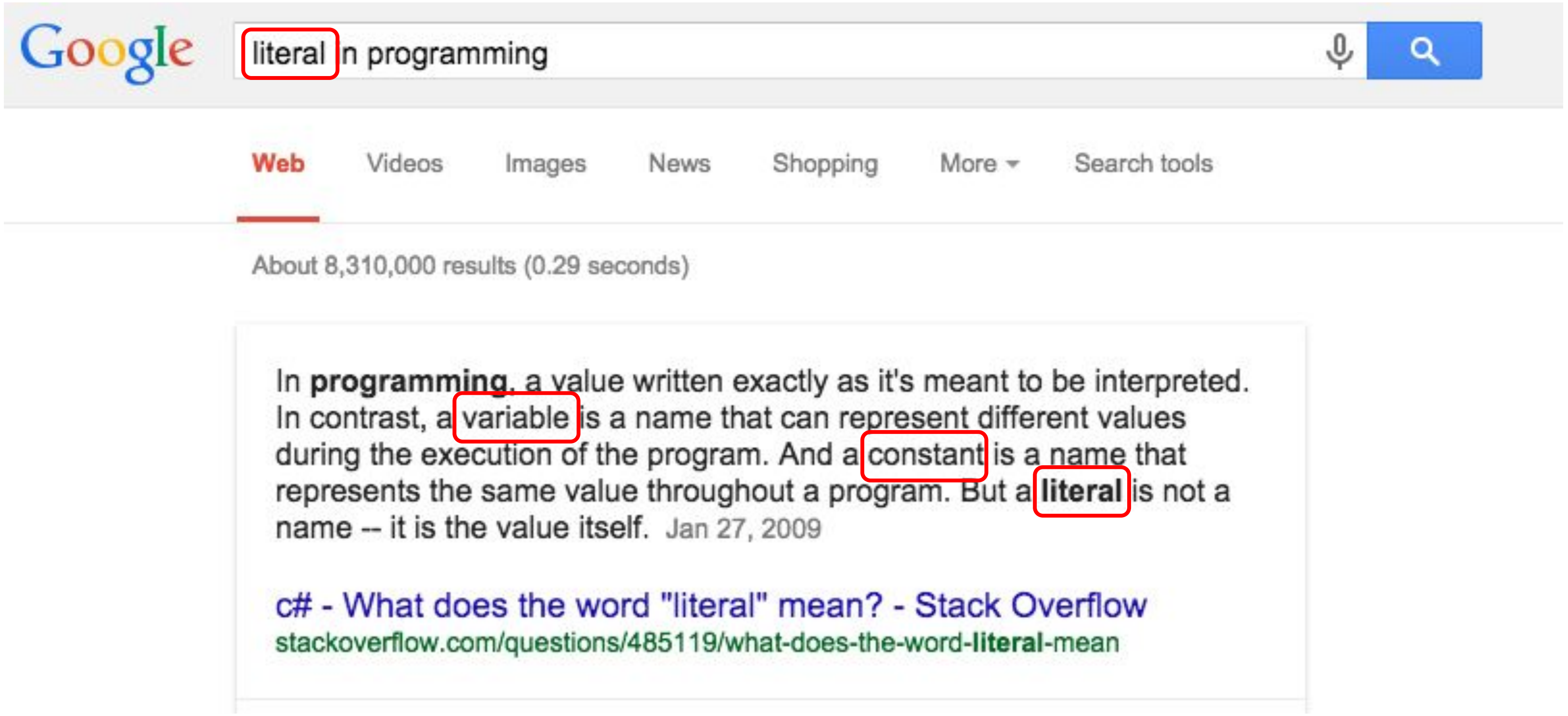
```
package main

import "fmt"

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

/*

clarifi

my workspace is called only "go"
so in my GOPATH you will see it pointing to:

GOPATH="/Users/tm002/Documents/go"

*/
```

This function is declared with no (choose one):

- parameters
- arguments

“Hello world” is an example of a **literal** (choose one):

- parameter
- argument

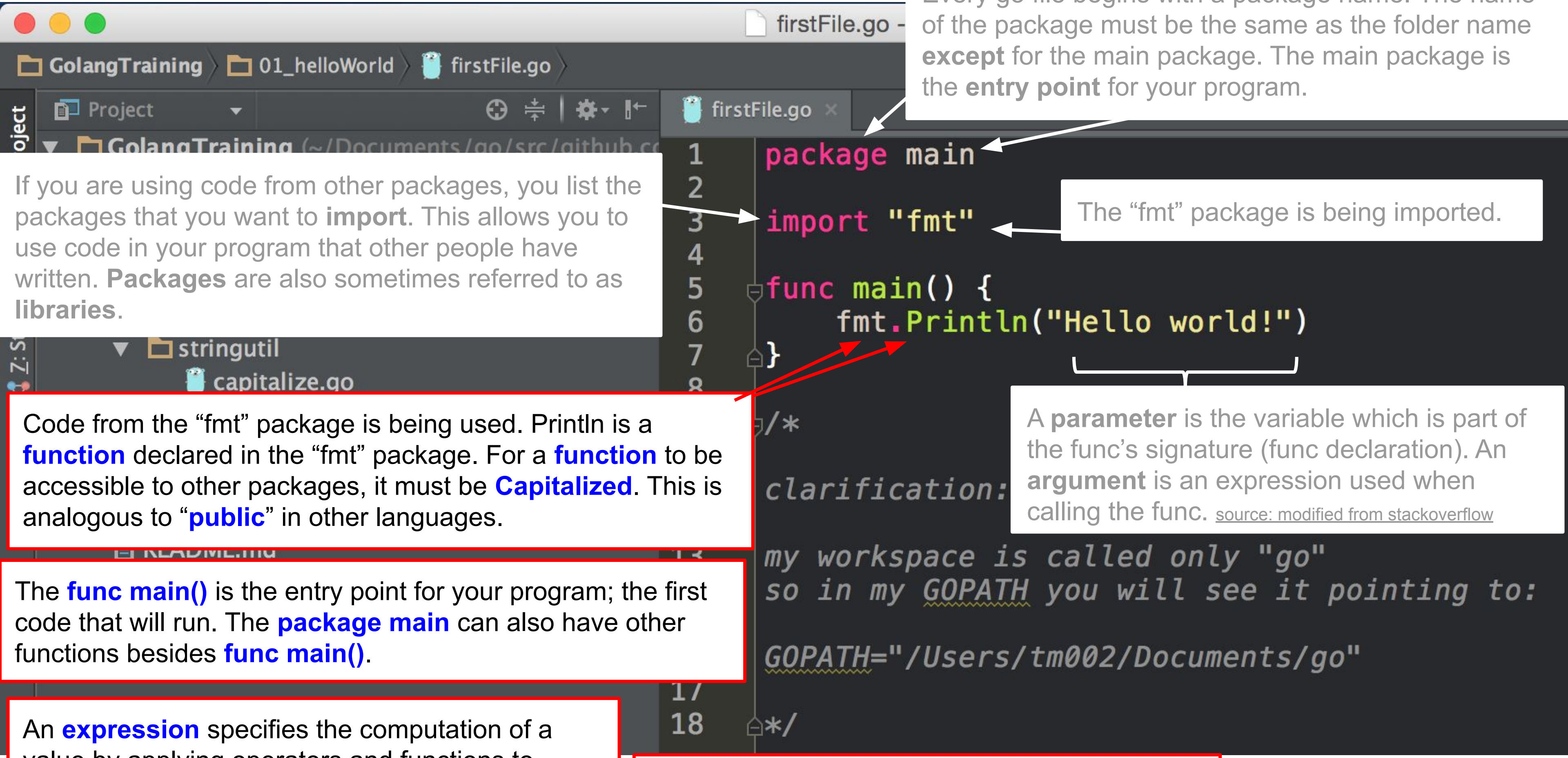

Google

literal in programming

About 8,310,000 results (0.29 seconds)

In **programming**, a value written exactly as it's meant to be interpreted. In contrast, a variable is a name that can represent different values during the execution of the program. And a constant is a name that represents the same value throughout a program. But a **literal** is not a name -- it is the value itself. Jan 27, 2009

c# - What does the word "literal" mean? - Stack Overflow
stackoverflow.com/questions/485119/what-does-the-word-**literal**-mean



```go
package main

import "fmt"

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

/*
clarification:
my workspace is called only "go"
so in my GOPATH you will see it pointing to:
GOPATH="/Users/tm002/Documents/go"
*/
```

Every go file begins with a package name. The name of the package must be the same as the folder name **except** for the main package. The main package is the **entry point** for your program.

If you are using code from other packages, you list the packages that you want to **import**. This allows you to use code in your program that other people have written. **Packages** are also sometimes referred to as **libraries**.

The "fmt" package is being imported.

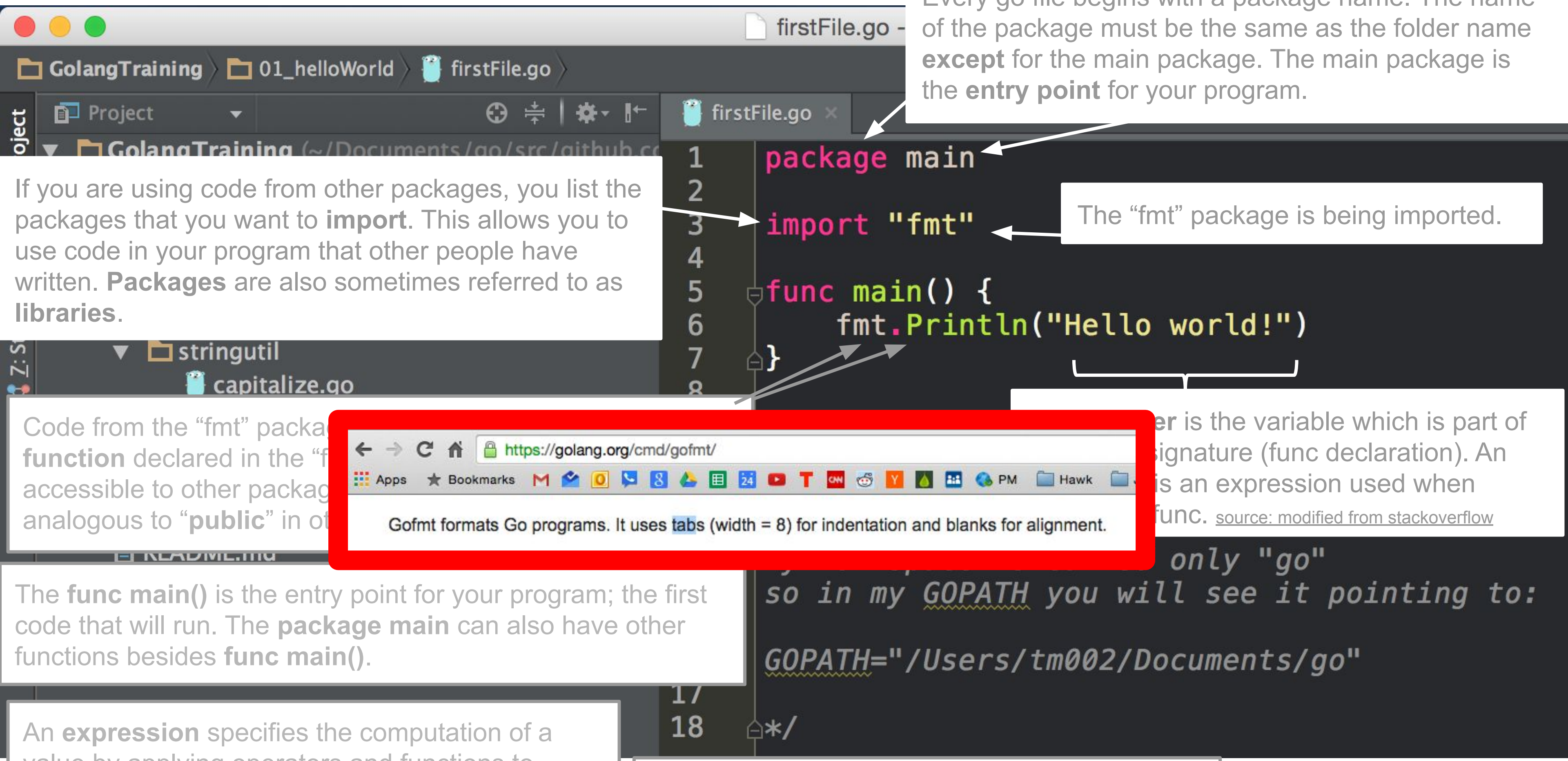
Code from the "fmt" package is being used. Println is a **function** declared in the "fmt" package. For a **function** to be accessible to other packages, it must be **Capitalized**. This is analogous to "**public**" in other languages.

A **parameter** is the variable which is part of the func's signature (func declaration). An **argument** is an expression used when calling the func. source: modified from stackoverflow

The **func main()** is the entry point for your program; the first code that will run. The **package main** can also have other functions besides **func main()**.

An **expression** specifies the computation of a value by applying operators and functions to operands. source: effective go

Statements control execution. source: effective go



```
package main

import "fmt"

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

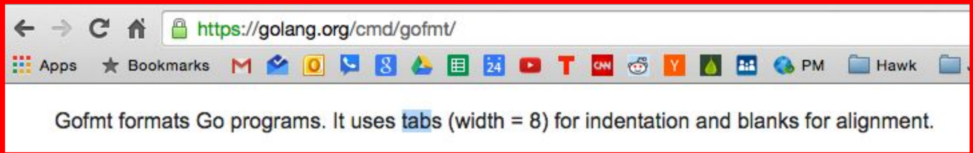
```
                              only "go"
so in my GOPATH you will see it pointing to:

GOPATH="/Users/tm002/Documents/go"

*/
```

Every go file begins with a package name. The name of the package must be the same as the folder name **except** for the main package. The main package is the **entry point** for your program.

If you are using code from other packages, you list the packages that you want to **import**. This allows you to use code in your program that other people have written. **Packages** are also sometimes referred to as **libraries**.

The “fmt” package is being imported.

Code from the “fmt” packa
function declared in the “f
accessible to other packag
analogous to “**public**” in ot



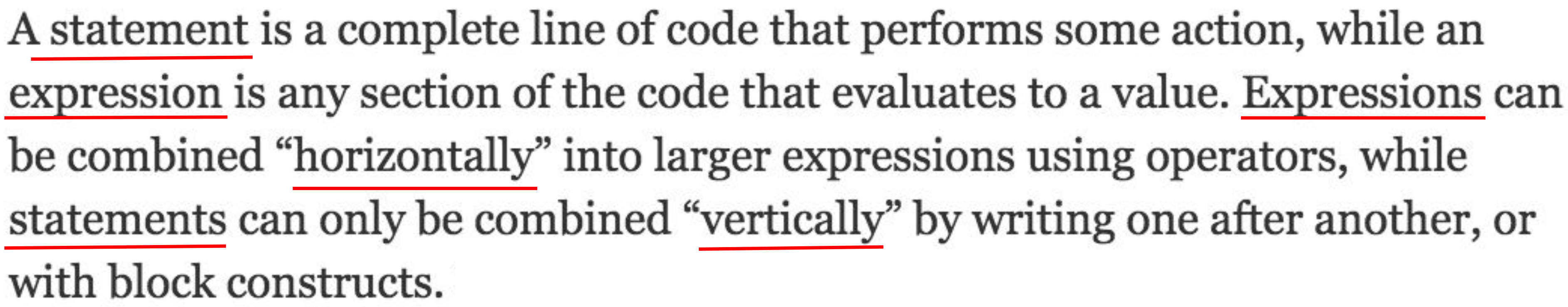
Gofmt formats Go programs. It uses tabs (width = 8) for indentation and blanks for alignment.

er is the variable which is part of
signature (func declaration). An
is an expression used when
func. source: modified from stackoverflow

The **func main()** is the entry point for your program; the first code that will run. The **package main** can also have other functions besides **func main()**.

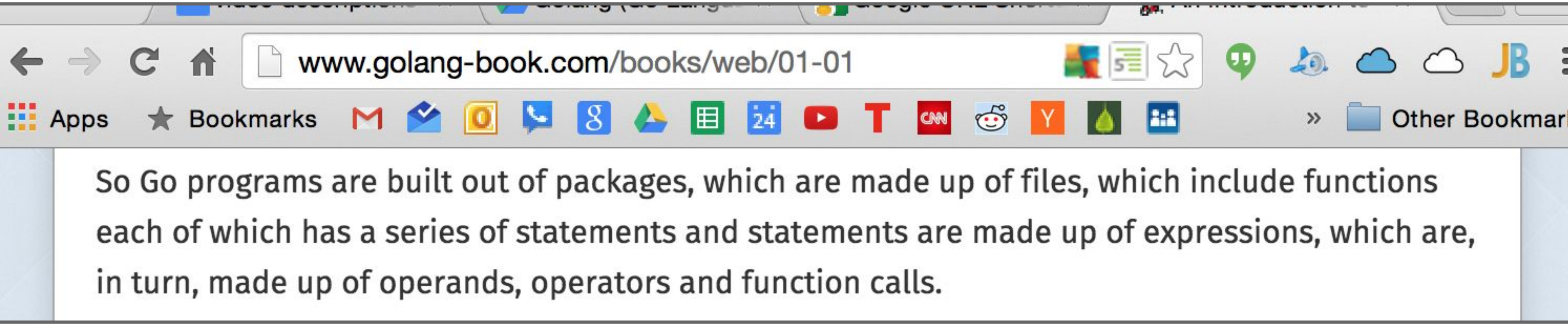
An **expression** specifies the computation of a value by applying operators and functions to operands. source: effective go

Statements control execution. source: effective go

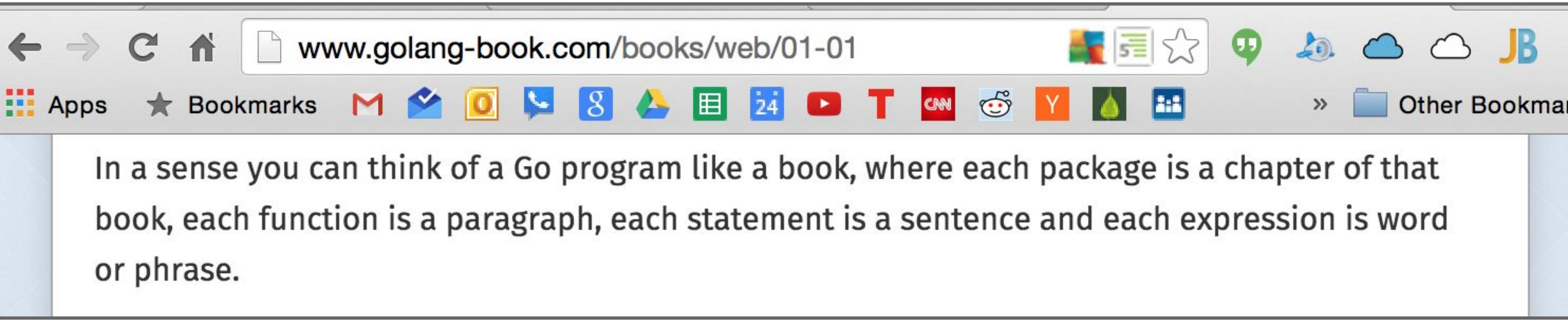


A statement is a complete line of code that performs some action, while an expression is any section of the code that evaluates to a value. Expressions can be combined "horizontally" into larger expressions using operators, while statements can only be combined "vertically" by writing one after another, or with block constructs.

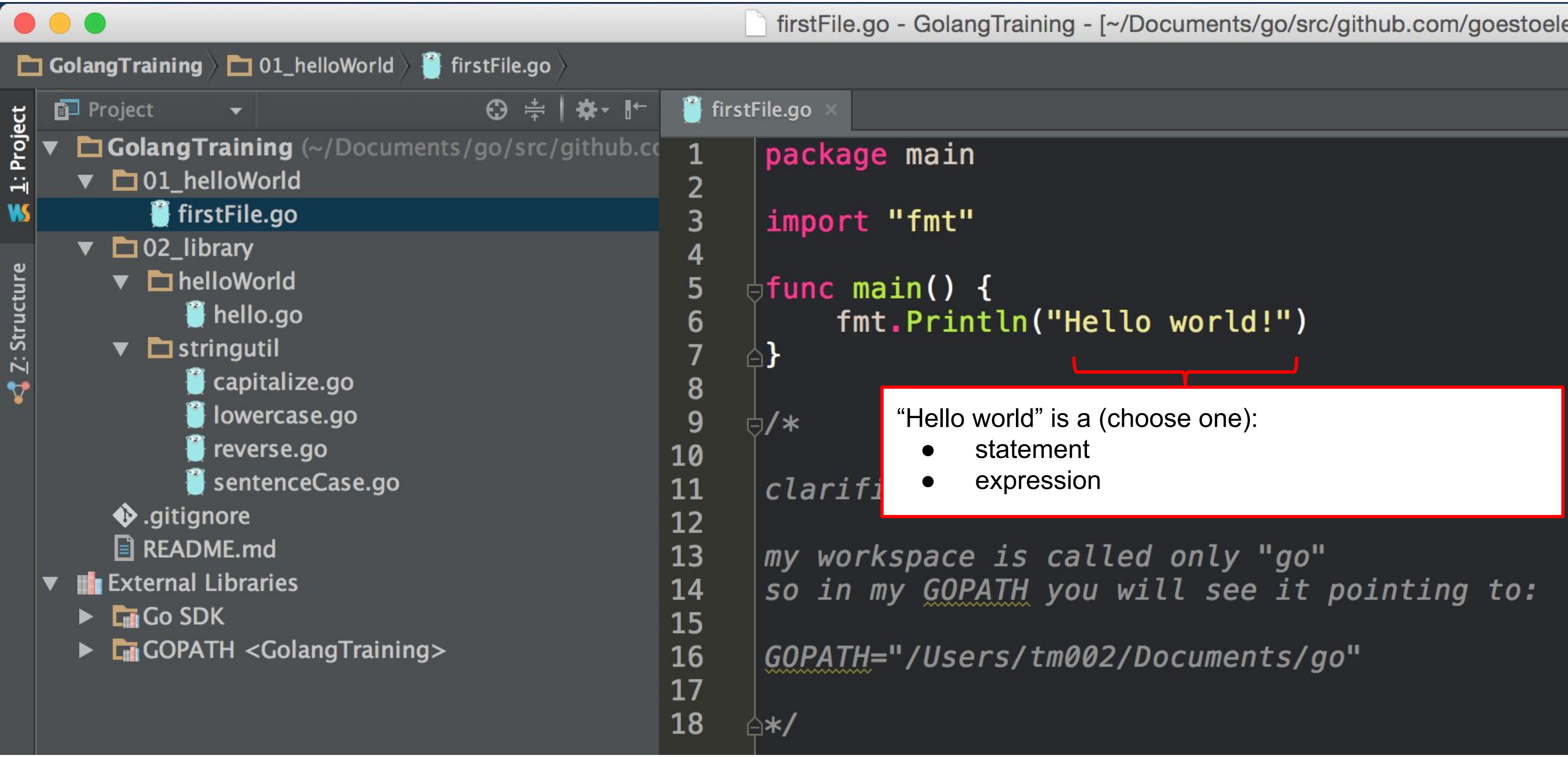
http://www.quora.com/Whats-the-difference-between-a-statement-and-an-expression-in-Python



So Go programs are built out of packages, which are made up of files, which include functions each of which has a series of statements and statements are made up of expressions, which are, in turn, made up of operands, operators and function calls.



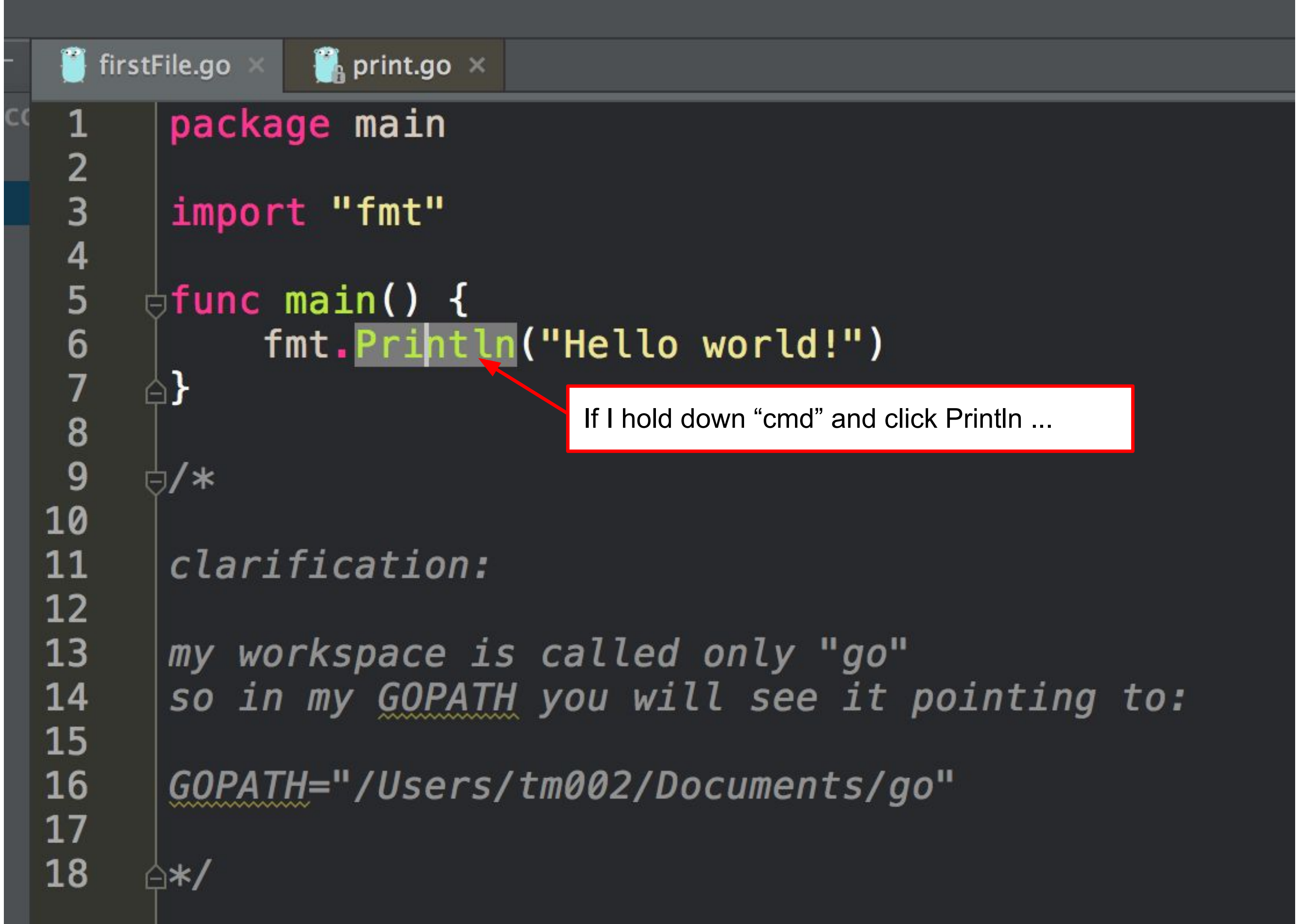
In a sense you can think of a Go program like a book, where each package is a chapter of that book, each function is a paragraph, each statement is a sentence and each expression is word or phrase.



```go
package main

import "fmt"

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

/*

clarifi

my workspace is called only "go"
so in my GOPATH you will see it pointing to:

GOPATH="/Users/tm002/Documents/go"

*/
```

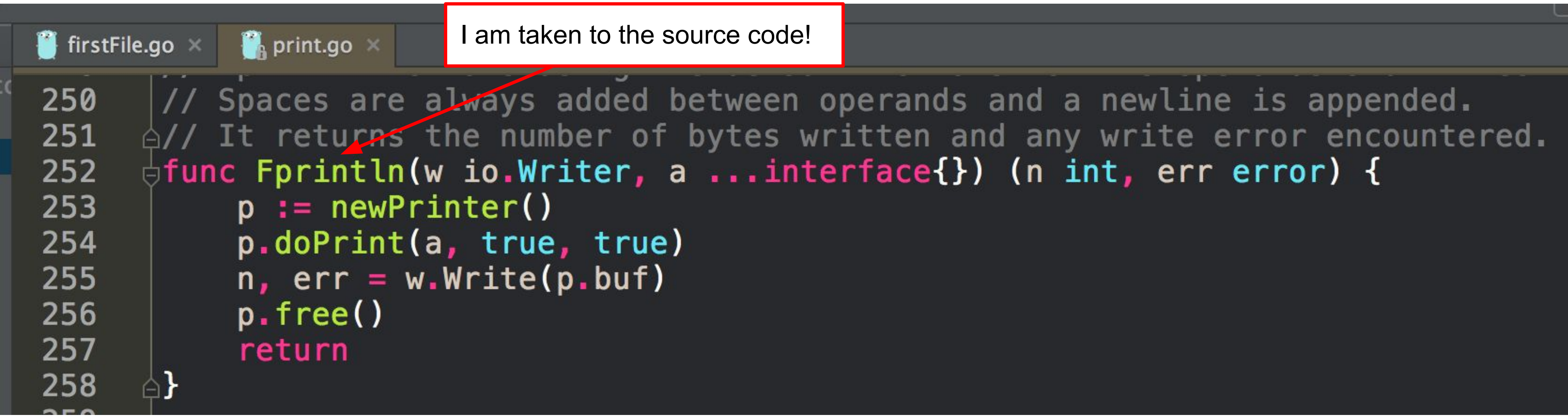
"Hello world" is a (choose one):

- statement
- expression



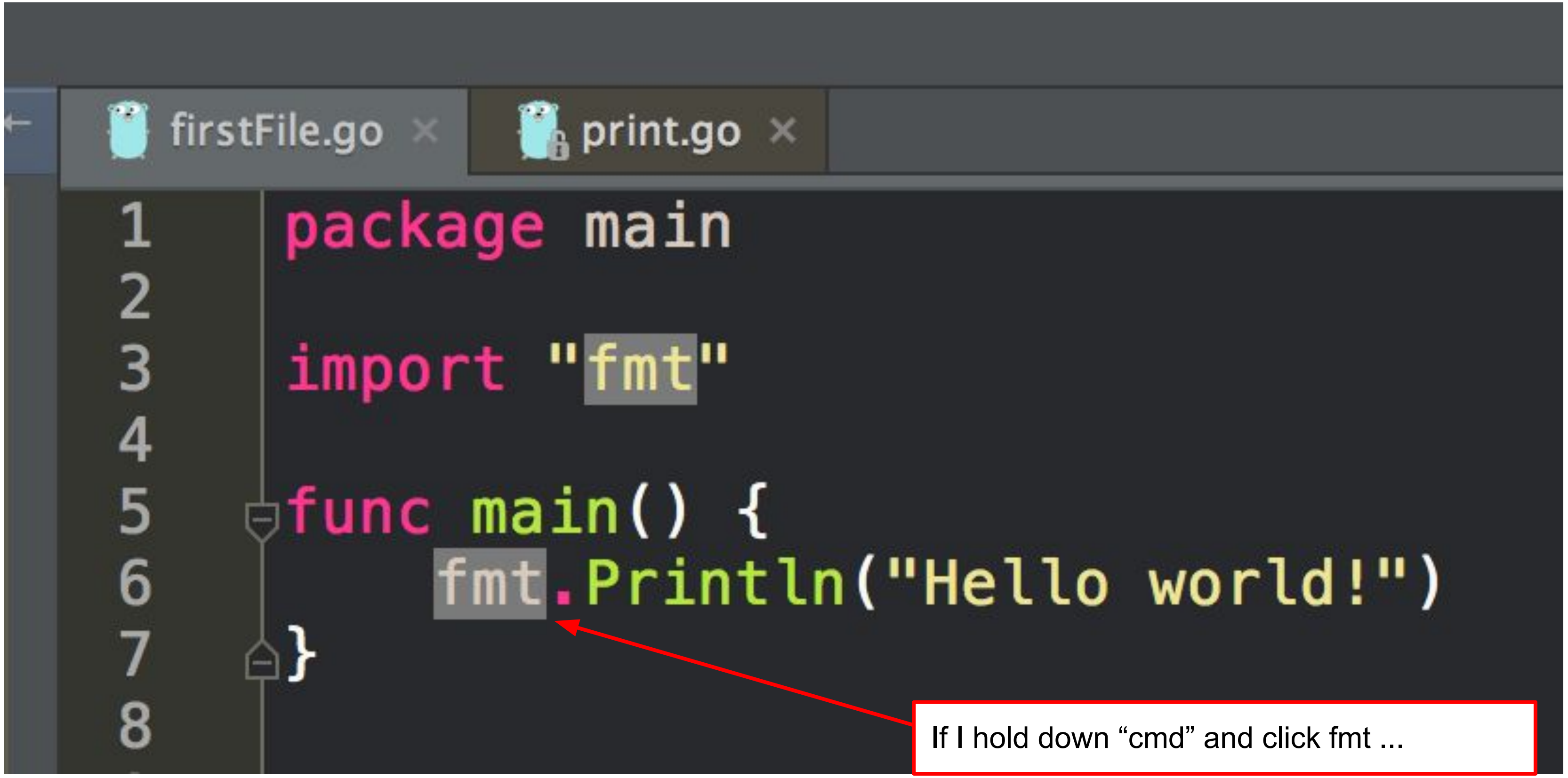
```
package main

import "fmt"

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

/*

clarification:

my workspace is called only "go"
so in my GOPATH you will see it pointing to:

GOPATH="/Users/tm002/Documents/go"

*/
```

If I hold down “cmd” and click Println ...


I am taken to the source code!

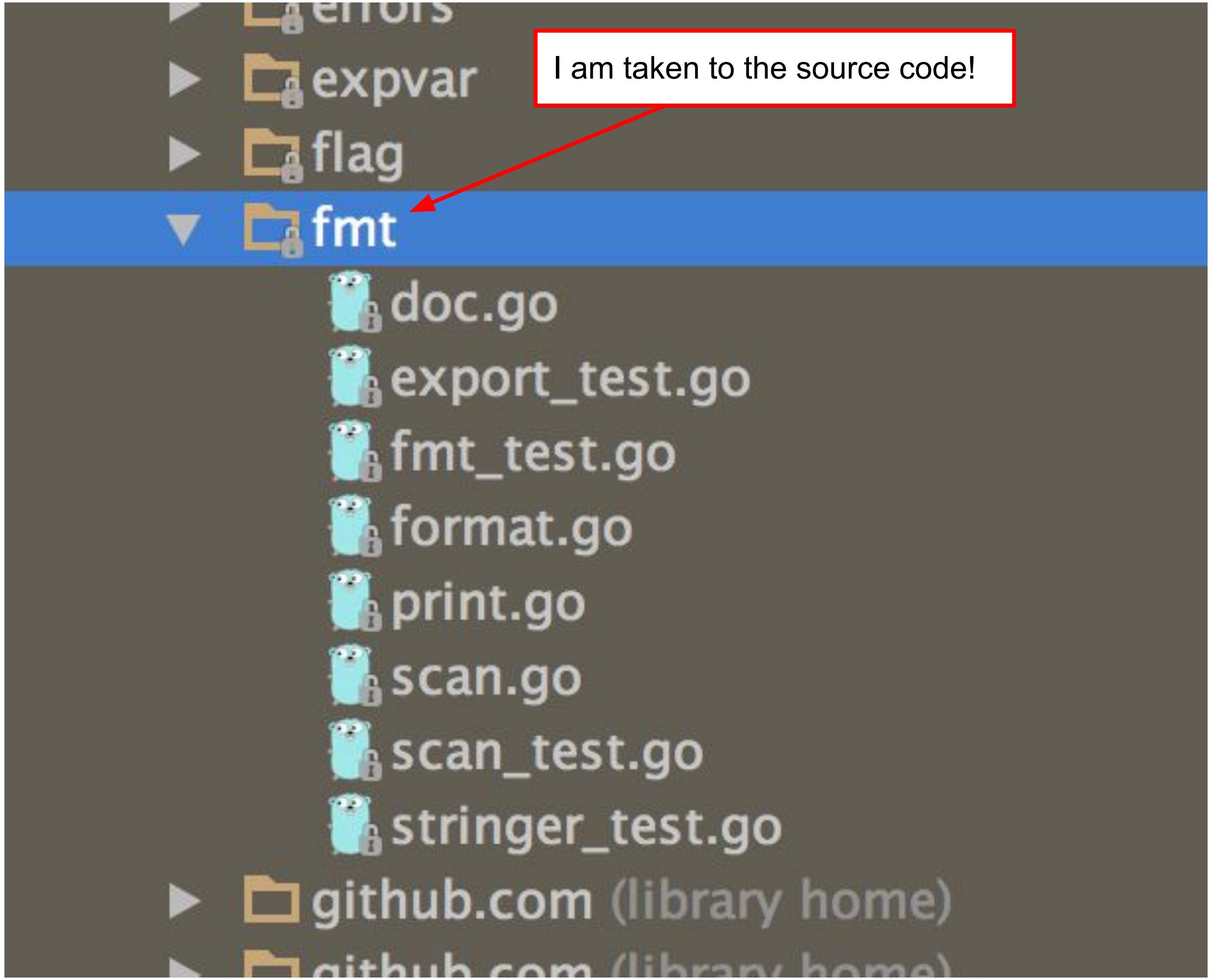
firstFile.go × print.go ×

```go
// Spaces are always added between operands and a newline is appended.
// It returns the number of bytes written and any write error encountered.
func Fprintln(w io.Writer, a ...interface{}) (n int, err error) {
	p := newPrinter()
	p.doPrint(a, true, true)
	n, err = w.Write(p.buf)
	p.free()
	return
}
```

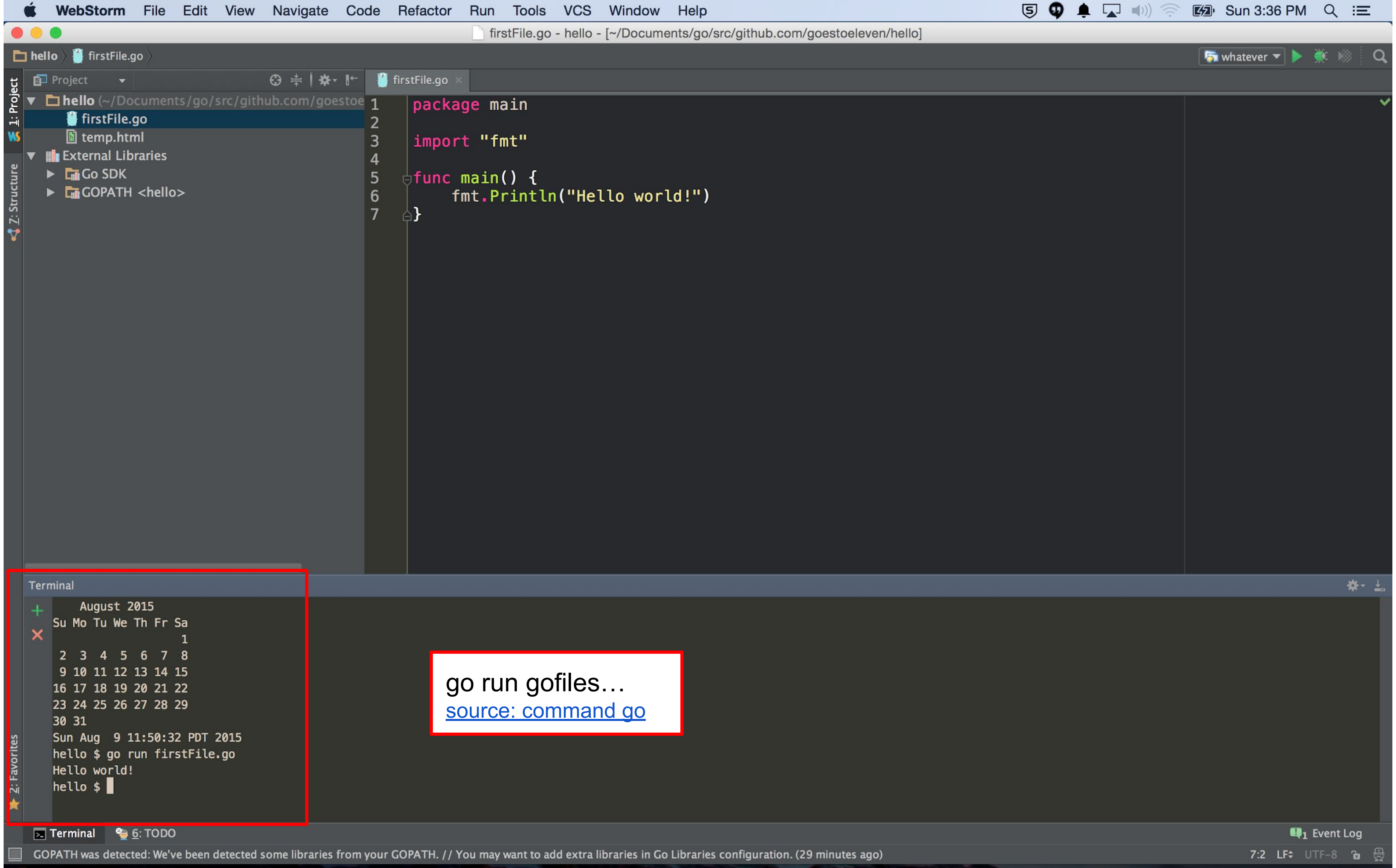


```go
package main

import "fmt"

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

If I hold down “cmd” and click fmt ...


I am taken to the source code!

```
▶ errors
▶ 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)
▶ github.com (library home)
```

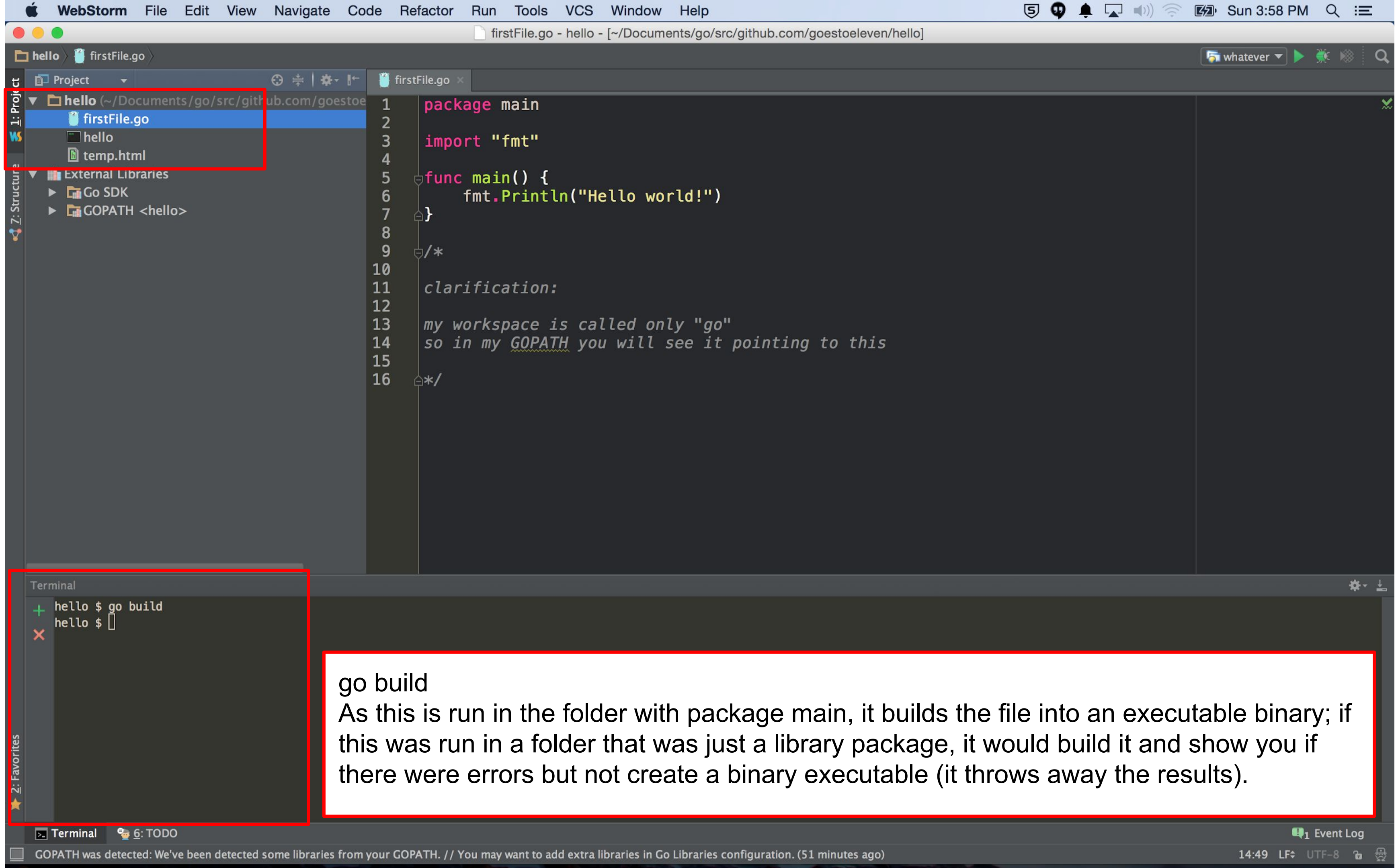

WebStorm File Edit View Navigate Code Refactor Run Tools VCS Window Help
Sun 3:36 PM
firstFile.go - hello - [~/Documents/go/src/github.com/goestoeleven/hello]
hello
firstFile.go
whatever
Project
hello (~/Documents/go/src/github.com/goestoe
temp.html
External Libraries
Go SDK
GOPATH <hello>
Terminal
6: TODO
Event Log
7:2 LF UTF-8

```go
package main

import "fmt"

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

```
    August 2015
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
Sun Aug  9 11:50:32 PDT 2015
hello $ go run firstFile.go
Hello world!
hello $
```

GOPATH was detected: We've been detected some libraries from your GOPATH. // You may want to add extra libraries in Go Libraries configuration. (29 minutes ago)

go run gofiles…
source: command go

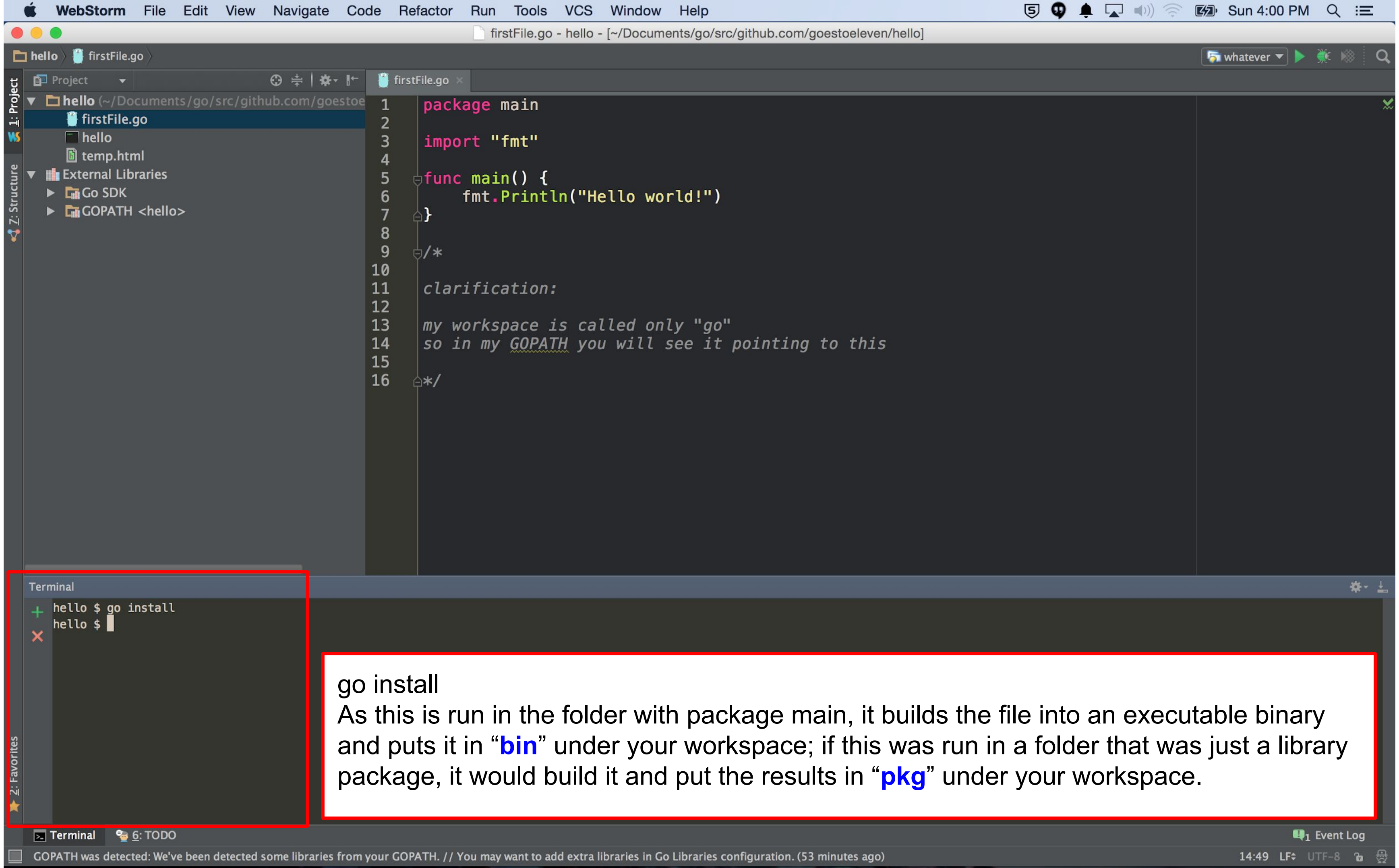

WebStorm File Edit View Navigate Code Refactor Run Tools VCS Window Help
firstFile.go - hello - [~/Documents/go/src/github.com/goestoeleven/hello]
hello
firstFile.go
Project
hello (~/Documents/go/src/github.com/goestoe
firstFile.go
hello
temp.html
External Libraries
Go SDK
GOPATH <hello>
whatever
Terminal
6: TODO
Event Log

```go
package main

import "fmt"

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

/*

clarification:

my workspace is called only "go"
so in my GOPATH you will see it pointing to this

*/
```

```
hello $ go build
hello $ 
```

GOPATH was detected: We've been detected some libraries from your GOPATH. // You may want to add extra libraries in Go Libraries configuration. (51 minutes ago)

go build
As this is run in the folder with package main, it builds the file into an executable binary; if this was run in a folder that was just a library package, it would build it and show you if there were errors but not create a binary executable (it throws away the results).



firstFile.go - hello - [~/Documents/go/src/github.com/goestoeleven/hello]

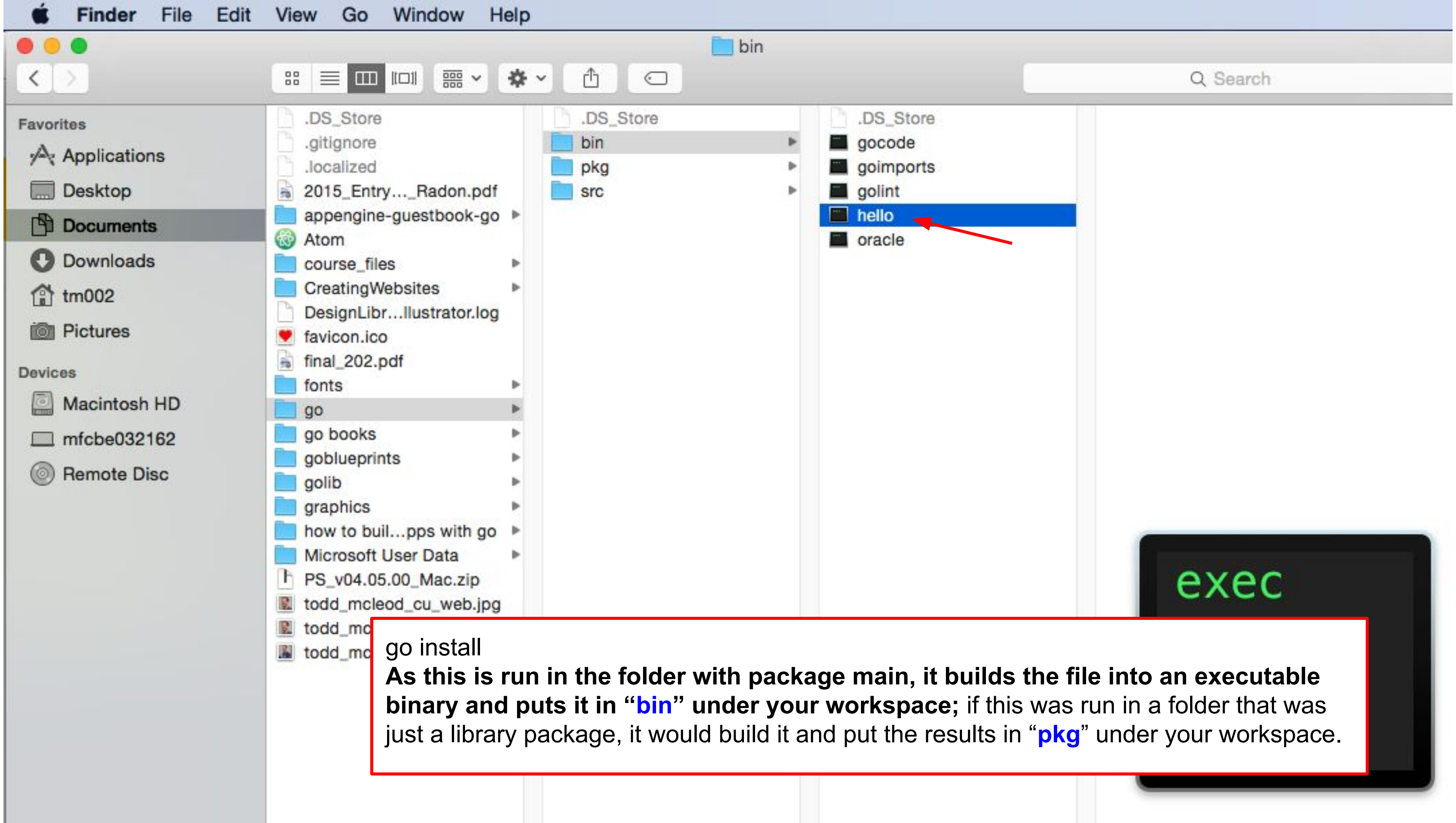
```go
package main

import "fmt"

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

/*

clarification:

my workspace is called only "go"
so in my GOPATH you will see it pointing to this

*/
```

Terminal

```
hello $ go install
hello $ 
```

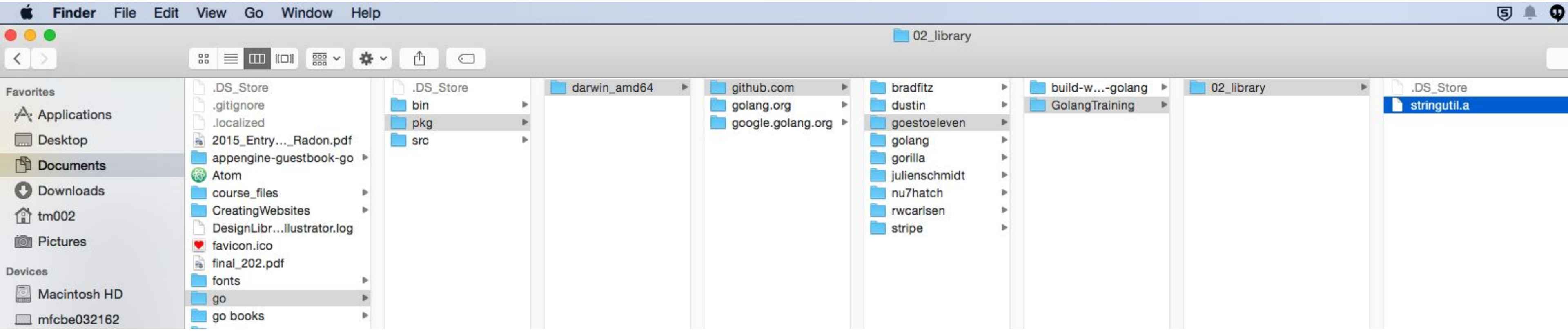
go install
As this is run in the folder with package main, it builds the file into an executable binary and puts it in "**bin**" under your workspace; if this was run in a folder that was just a library package, it would build it and put the results in "**pkg**" under your workspace.

GOPATH was detected: We've been detected some libraries from your GOPATH. // You may want to add extra libraries in Go Libraries configuration. (53 minutes ago)

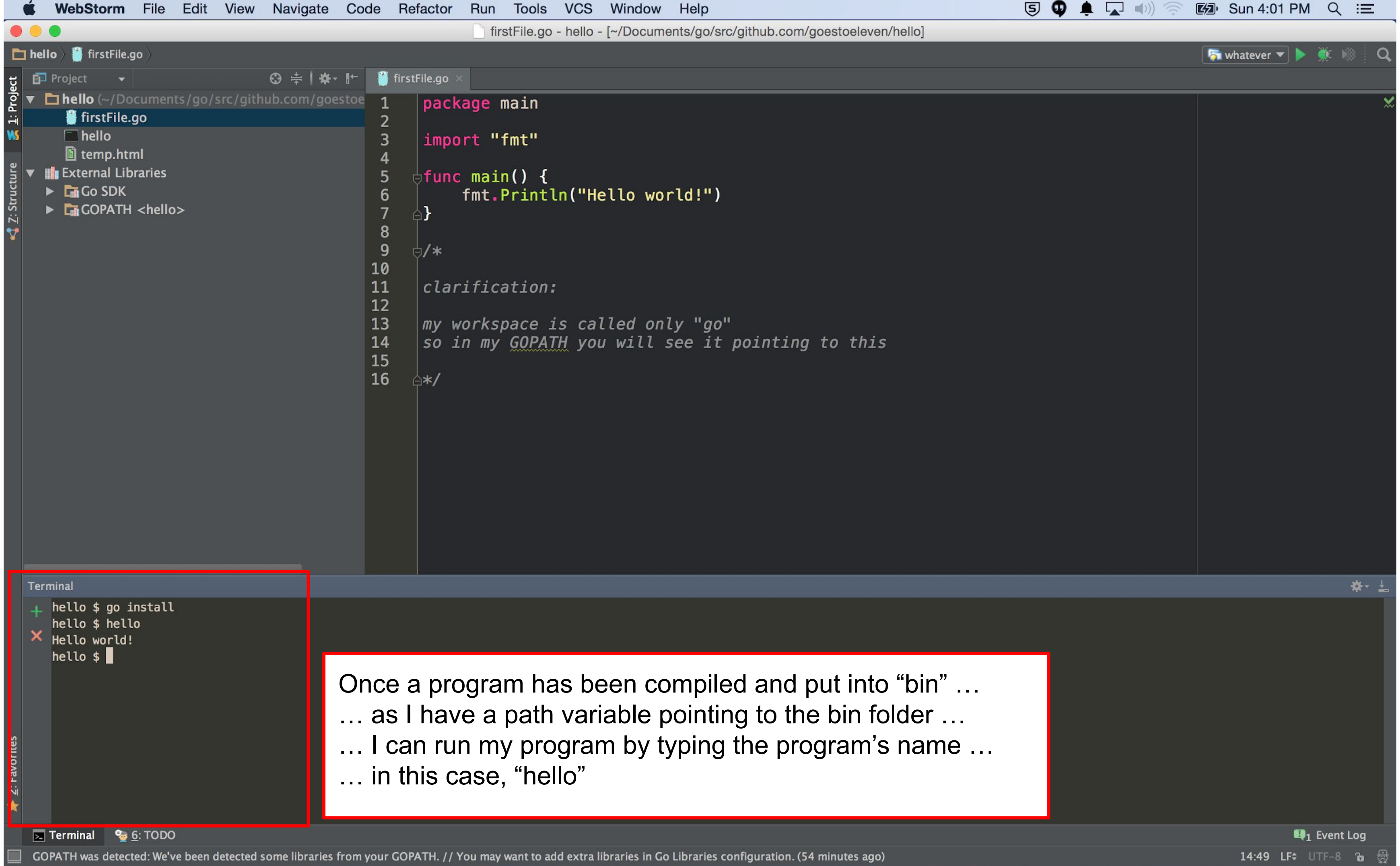


go install

As this is run in the folder with package main, it builds the file into an executable binary and puts it in "bin" under your workspace; if this was run in a folder that was just a library package, it would build it and put the results in "pkg" under your workspace.



go install
As this is run in the folder with package main, it builds the file into an executable binary and puts it in “bin” under your workspace; **if this was run in a folder that was just a library package, it would build it and put the results in “pkg” under your workspace.**



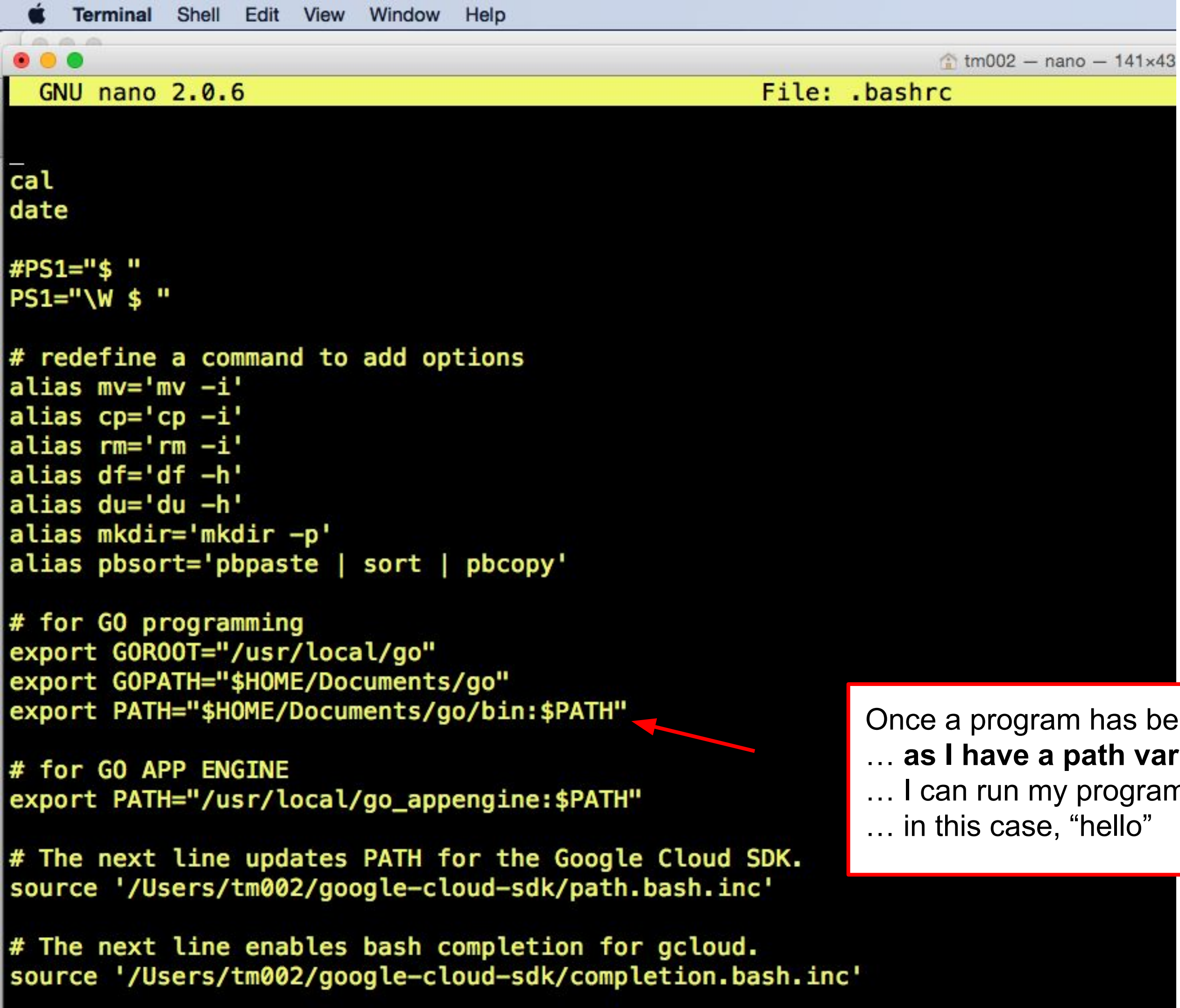
WebStorm File Edit View Navigate Code Refactor Run Tools VCS Window Help
Sun 4:01 PM
firstFile.go - hello - [~/Documents/go/src/github.com/goestoeleven/hello]
hello
firstFile.go
whatever
Project
hello (~/Documents/go/src/github.com/goestoe
firstFile.go
hello
temp.html
External Libraries
Go SDK
GOPATH <hello>
Terminal
6: TODO
Event Log
14:49 LF: UTF-8

```go
package main

import "fmt"

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

/*

clarification:

my workspace is called only "go"
so in my GOPATH you will see it pointing to this

*/
```

```
hello $ go install
hello $ hello
Hello world!
hello $
```

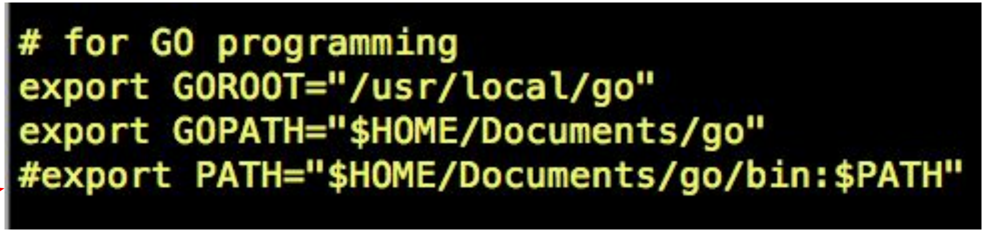
Once a program has been compiled and put into “bin” …
… as I have a path variable pointing to the bin folder …
… I can run my program by typing the program’s name …
… in this case, “hello”

GOPATH was detected: We've been detected some libraries from your GOPATH. // You may want to add extra libraries in Go Libraries configuration. (54 minutes ago)



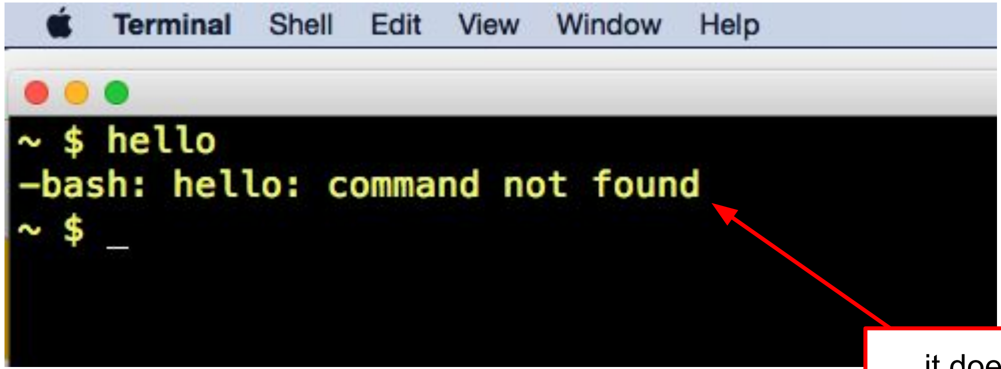
```
  GNU nano 2.0.6                                   File: .bashrc


_
cal
date

#PS1="$ "
PS1="\W $ "

# redefine a command to add options
alias mv='mv -i'
alias cp='cp -i'
alias rm='rm -i'
alias df='df -h'
alias du='du -h'
alias mkdir='mkdir -p'
alias pbsort='pbpaste | sort | pbcopy'

# for GO programming
export GOROOT="/usr/local/go"
export GOPATH="$HOME/Documents/go"
export PATH="$HOME/Documents/go/bin:$PATH"

# for GO APP ENGINE
export PATH="/usr/local/go_appengine:$PATH"

# The next line updates PATH for the Google Cloud SDK.
source '/Users/tm002/google-cloud-sdk/path.bash.inc'

# The next line enables bash completion for gcloud.
source '/Users/tm002/google-cloud-sdk/completion.bash.inc'
```

Once a program has been compiled and put into "bin" …
… **as I have a path variable pointing to the bin folder** …
… I can run my program by typing the program's name …
… in this case, "hello"



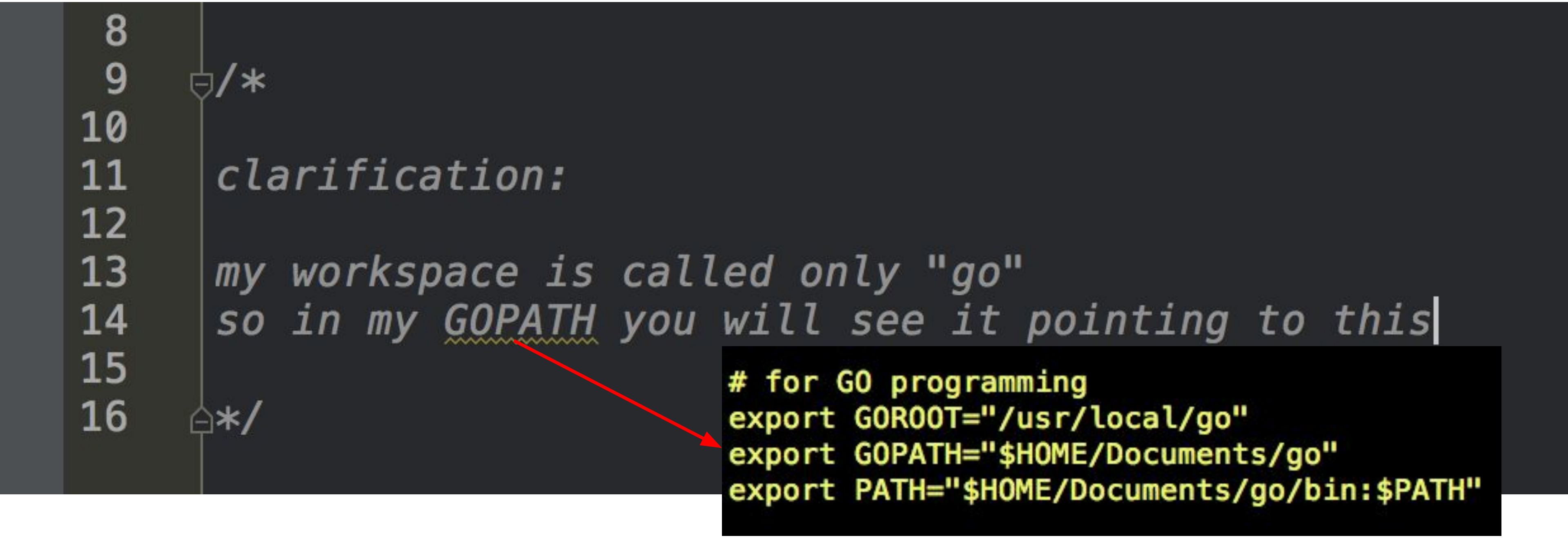
```
# for GO programming
export GOROOT="/usr/local/go"
export GOPATH="$HOME/Documents/go"
#export PATH="$HOME/Documents/go/bin:$PATH"
```

If I comment it out ...



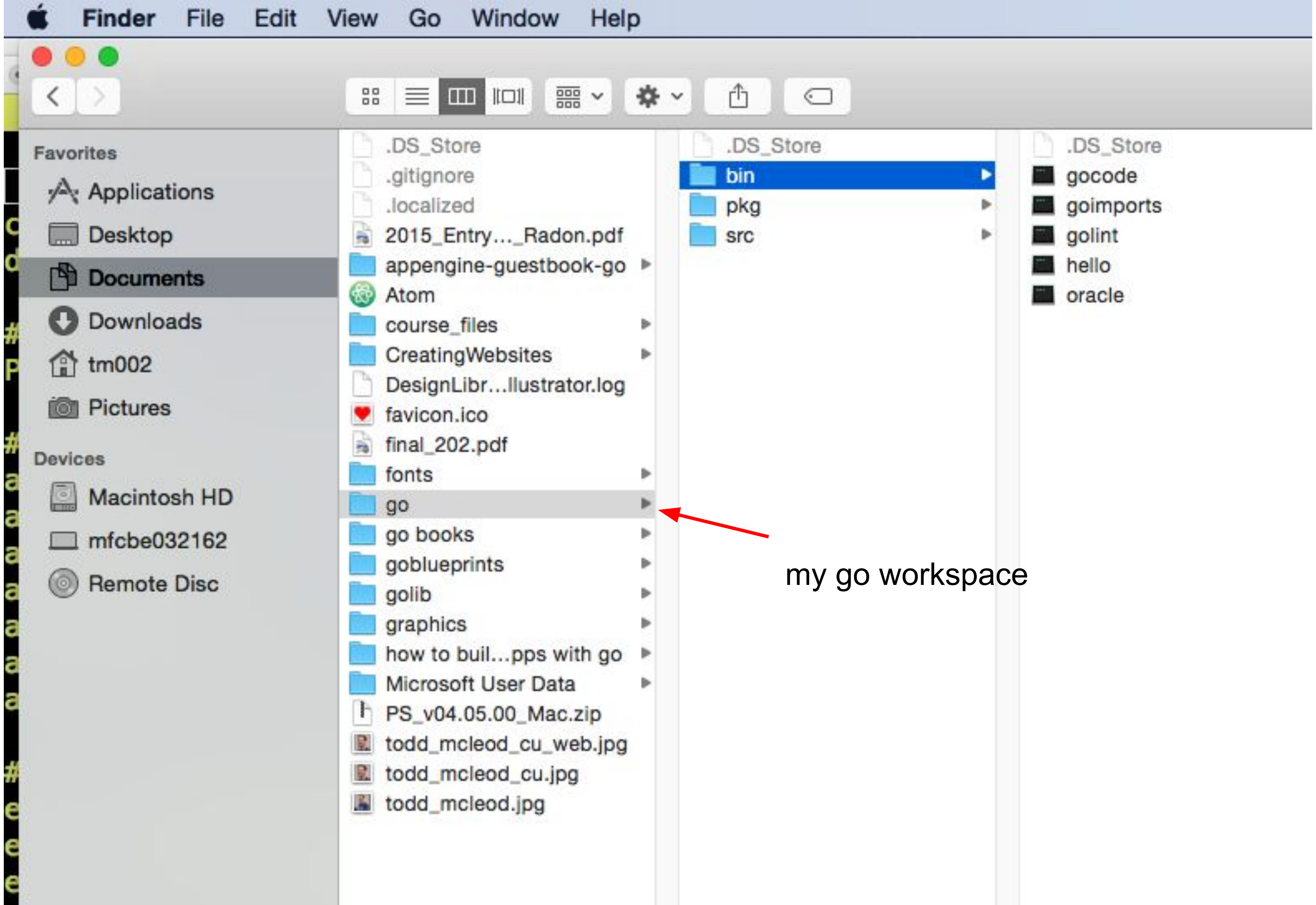
```
~ $ hello
-bash: hello: command not found
~ $ _
```

… it doesn’t run.



```
/*

clarification:

my workspace is called only "go"
so in my GOPATH you will see it pointing to this

*/
```

```
# for GO programming
export GOROOT="/usr/local/go"
export GOPATH="$HOME/Documents/go"
export PATH="$HOME/Documents/go/bin:$PATH"
```



```
.DS_Store                      .DS_Store    .DS_Store
.gitignore                     bin          gocode
.localized                     pkg          goimports
2015_Entry..._Radon.pdf        src          golint
appengine-guestbook-go                      hello
Atom                                        oracle
course_files
CreatingWebsites
DesignLibr...llustrator.log
favicon.ico
final_202.pdf
fonts
go
go books
goblueprints
golib
graphics
how to buil...pps with go
Microsoft User Data
PS_v04.05.00_Mac.zip
todd_mcleod_cu_web.jpg
todd_mcleod_cu.jpg
todd_mcleod.jpg
```

my go workspace

Review

- code completion
- package main
    - **func main()**
        - entry point for your program
- **packages**, aka, libraries
- import
- **functions**
    - What makes a function accessible outside a package
        - **capitalization**
            - makes the function “public”
        - lowercase
            - function restricted to package
- **parameters vs arguments**
- **expressions vs statements**
- **variable, constant, literal**
- **cmd + click** → takes you to source code

- **go run**
    - go run gofiles...
- **go build**
- **go install**

Review Questions

Package Main

- What is the purpose of **package main** in a go program?
- What function must **package main** contain?
- Can **package main** contain a function called **func blueSky()** ?

funcs

What makes a func accessible outside a package?

Parameters vs Arguments

- What is the difference between the two?

Expressions vs Statements

- What is the difference between the two?

Variable, Constant, Literal

- Define the three concepts above.
- Give an example of a literal from the “hello world” example.

go run

- Build “hello go” in your editor
- use **go run** from the command line to make your “hello go” program execute

go build

Go build does what when run on a folder containing package main?

go build

Go build does what when run on a folder containing a library package?

go install

Go install does what when run on a folder containing package main?

go install

Go install does what when run on a folder containing a library package?