

Using Libraries

(aka, using other people's code)

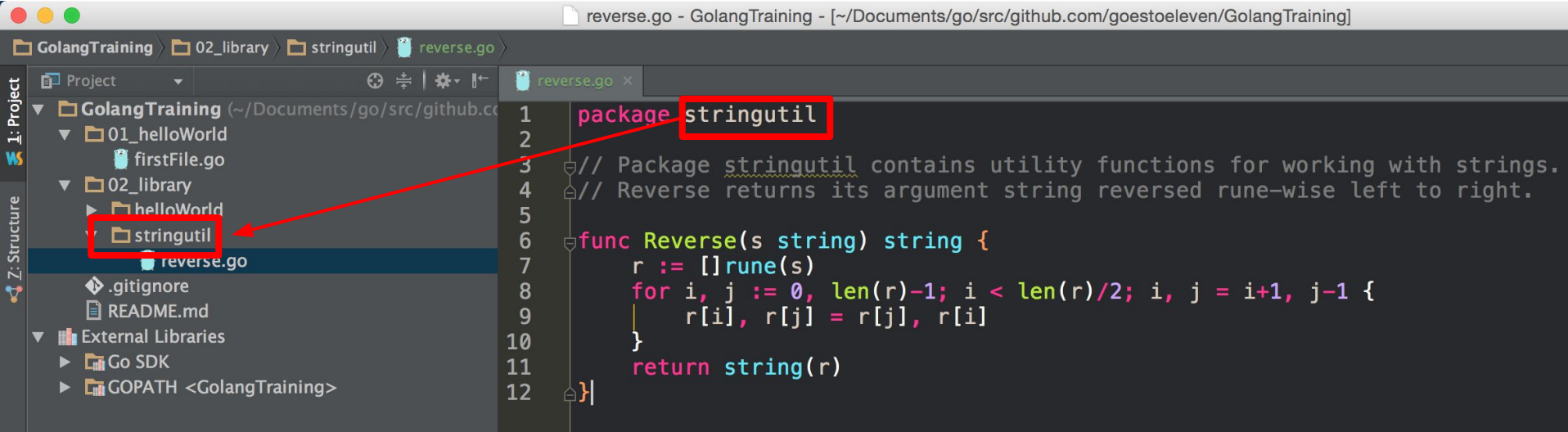
(aka, using code in other **packages**)

(aka, using **packages**)

naming & organizing

files and packages

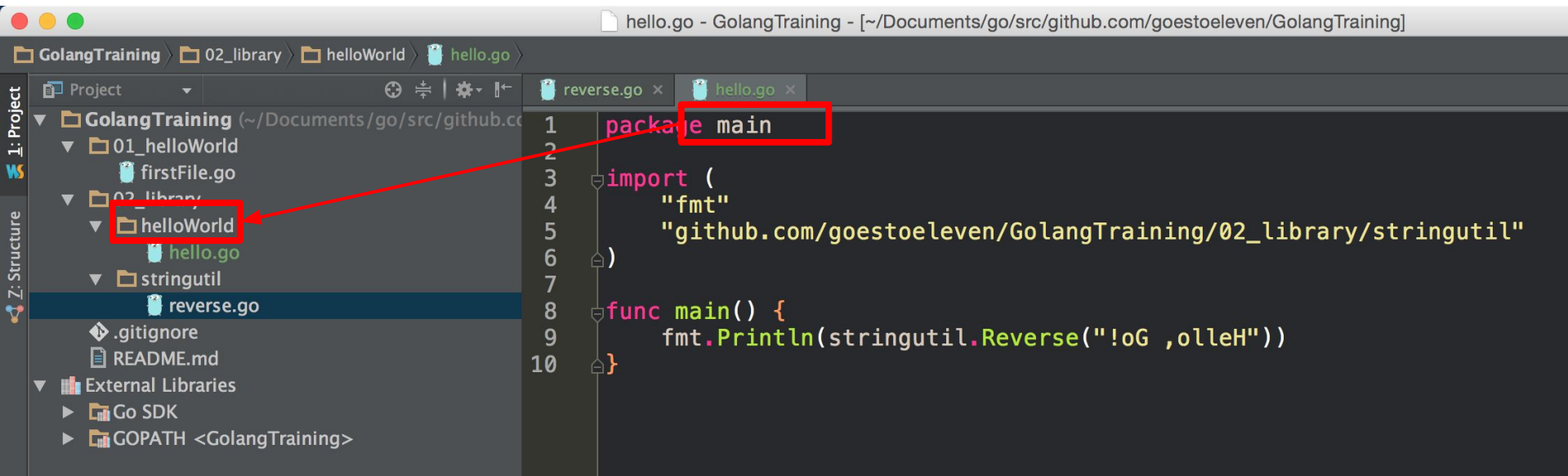
Package name must match containing folder name ...



The screenshot shows an IDE window with a Go project named 'GolangTraining'. The project structure is visible on the left, showing a folder named 'stringutil' under '02_library'. The code editor on the right shows the contents of 'reverse.go', which starts with the package declaration 'package stringutil'. A red box highlights the 'stringutil' folder in the project tree, and another red box highlights the 'package stringutil' declaration in the code. A red arrow points from the folder to the package declaration, illustrating the requirement that the package name must match the containing folder name.

```
1 package stringutil
2
3 // Package stringutil contains utility functions for working with strings.
4 // Reverse returns its argument string reversed rune-wise left to right.
5
6 func Reverse(s string) string {
7     r := []rune(s)
8     for i, j := 0, len(r)-1; i < len(r)/2; i, j = i+1, j-1 {
9         r[i], r[j] = r[j], r[i]
10    }
11    return string(r)
12 }
```

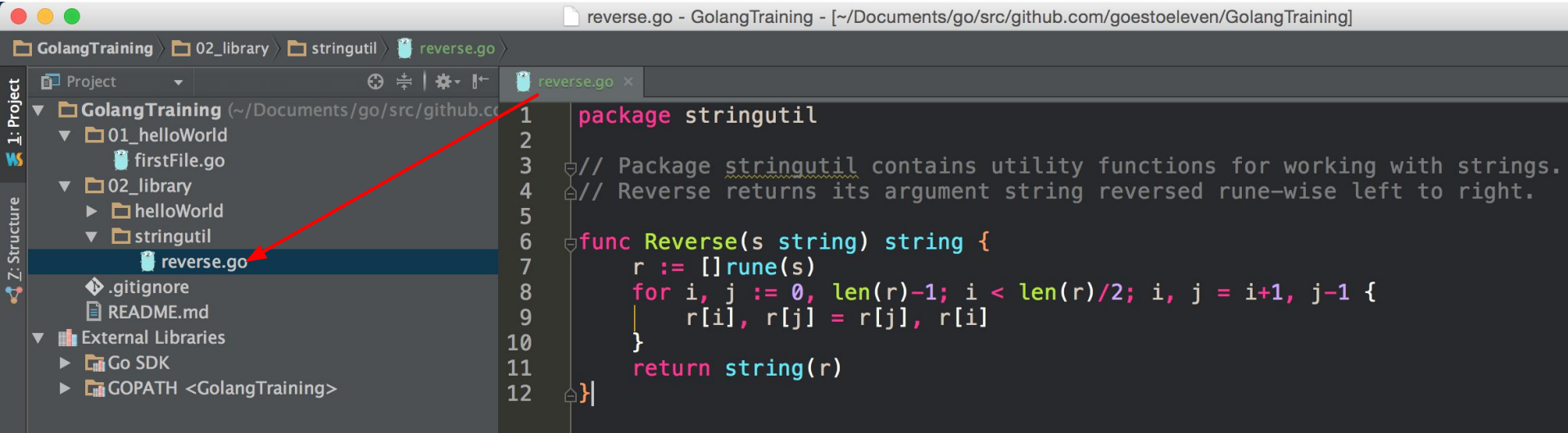
... unless the package is main



Reminder:

There can only be one main function per executable program.
You can have other functions in the main package.

File names: short, evocative, concise, error on the side of brevity.



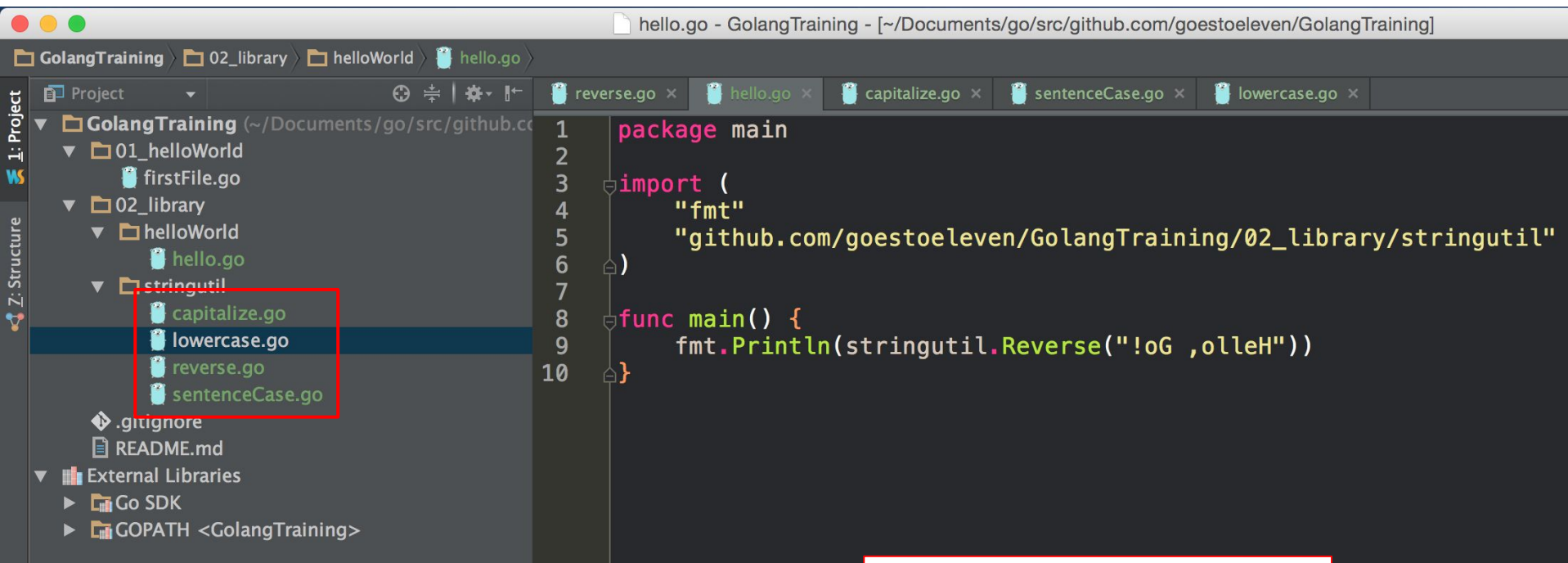
The screenshot shows an IDE window with a project structure on the left and a code editor on the right. The project structure is as follows:

- GolangTraining (~/Documents/go/src/github.com/goestoeleven/GolangTraining)
 - 01_helloWorld
 - firstFile.go
 - 02_library
 - helloWorld
 - stringutil
 - reverse.go
 - External Libraries
 - Go SDK
 - GOPATH <GolangTraining>

The code editor shows the content of `reverse.go`:

```
1 package stringutil
2
3 // Package stringutil contains utility functions for working with strings.
4 // Reverse returns its argument string reversed rune-wise left to right.
5
6 func Reverse(s string) string {
7     r := []rune(s)
8     for i, j := 0, len(r)-1; i < len(r)/2; i, j = i+1, j-1 {
9         r[i], r[j] = r[j], r[i]
10    }
11    return string(r)
12 }
```

A red arrow points from the `reverse.go` file in the project structure to the code editor.



You can have as many files
as you'd like in a package

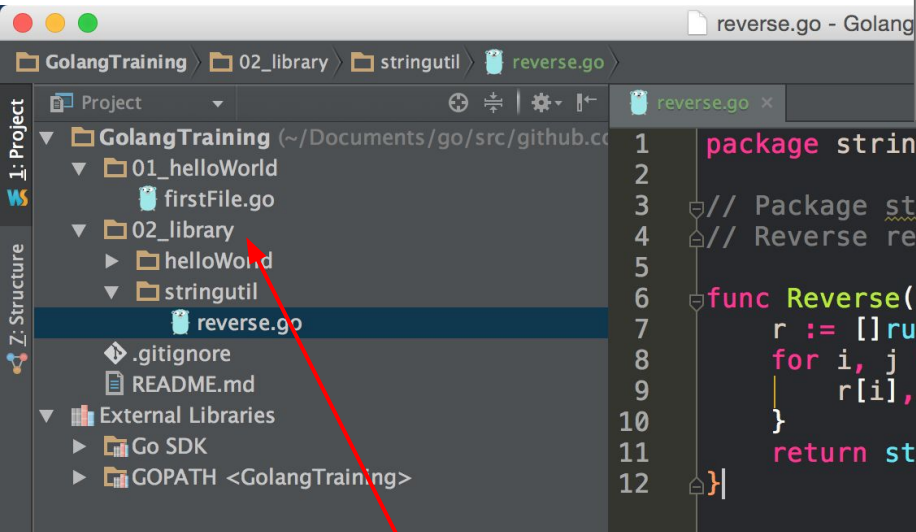
By the way, FYI

id·i·o·mat·ic

/ˌɪdēəˈmɑdɪk/

adjective

1. using, containing, or denoting expressions that are natural to a native speaker.
"distinctive idiomatic dialogue"
synonyms: vernacular, colloquial, everyday, conversational; [More](#)
2. appropriate to the style of art or music associated with a particular period, individual, or group.
"a short Bach piece containing lots of idiomatic motifs"



Notice the folder structure:

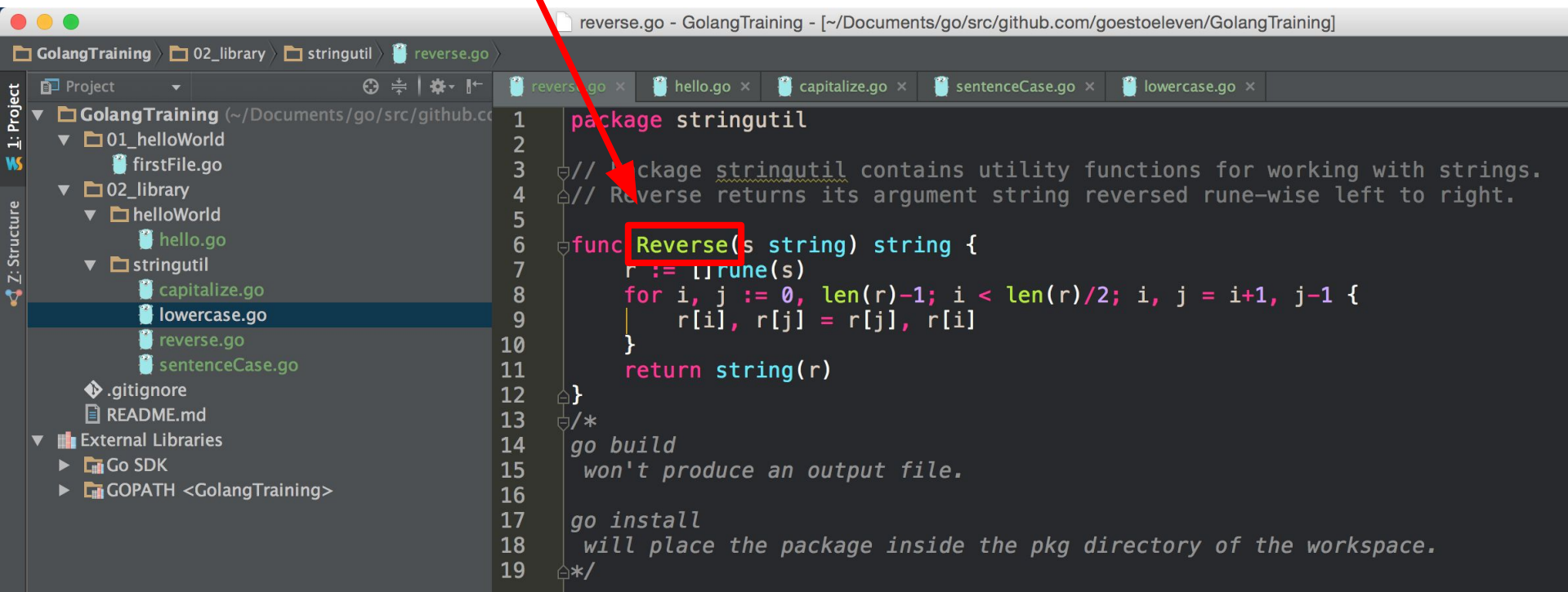
- 02_library is not idiomatic go
- I'm using this non-idiomatic structure to keep these files in sequence

capitalization

Public / private
(capitalized) / (camelCase)

applies to functions, variables, and types

Reminder:
For a function to be available outside a package,
the function name must be uppercase



```
reverse.go - GolangTraining - [~/Documents/go/src/github.com/goestoeleven/GolangTraining]

GolangTraining > 02_library > stringutil > reverse.go

Project
└─ GolangTraining (~/Documents/go/src/github.com/goestoeleven/GolangTraining)
   └─ 01_helloWorld
      └─ firstFile.go
   └─ 02_library
      └─ helloWorld
         └─ hello.go
      └─ stringutil
         └─ capitalize.go
         └─ lowercase.go
         └─ reverse.go
         └─ sentenceCase.go
   └─ .gitignore
   └─ README.md
└─ External Libraries
   └─ Go SDK
   └─ GOPATH <GolangTraining>

1 package stringutil
2
3 // Package stringutil contains utility functions for working with strings.
4 // Reverse returns its argument string reversed rune-wise left to right.
5
6 func Reverse(s string) string {
7     r := []rune(s)
8     for i, j := 0, len(r)-1; i < len(r)/2; i, j = i+1, j-1 {
9         r[i], r[j] = r[j], r[i]
10    }
11    return string(r)
12 }
13 /*
14 go build
15     won't produce an output file.
16
17 go install
18     will place the package inside the pkg directory of the workspace.
19 */
```

Reminder:

For a function to be available outside a package,
the function name must be uppercase

```
1 package stringutil
2
3 // Package stringutil contains utility functions for working with strings.
4 // Reverse returns its argument string reversed rune-wise left to right.
5
6 func Reverse(s string) string {
7     return reverseTwo(s)
8 }
9
10 func reverseTwo(s string) string {
11     r := []rune(s)
12     for i, j := 0, len(r)-1; i < len(r)/2; i, j = i+1, j-1 {
13         r[i], r[j] = r[j], r[i]
14     }
15     return string(r)
16 }
17
```

reverseTwo is available inside the package ...

Reminder:
For a function to be available outside a package,
the function name must be uppercase

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

GolangTraining > 02_library > helloWorld > hello.go

Project
└─ GolangTraining (~/Documents/go/src/github.com/goestoeleven/GolangTraining)
   └─ 01_helloWorld
      └─ firstFile.go
   └─ 02_library
      └─ helloWorld
         └─ hello.go
      └─ stringutil
         └─ capitalize.go
         └─ lowercase.go
         └─ reverse.go
         └─ sentenceCase.go
      └─ .gitignore
      └─ README.md
   └─ External Libraries
      └─ Go SDK
      └─ GOPATH <GolangTraining>

reverse.go x hello.go x capitalize.go x sentenceCase.go x lowercase.go x

1 package main
2
3 import (
4     "fmt"
5     "github.com/goestoeleven/GolangTraining/02_library/stringutil"
6 )
7
8 func main() {
9     fmt.Println(stringutil.Reverse("!oG ,olleH"))
10 }
```

code runs

```
Terminal
+ helloWorld $ go run hello.go
Hello, Go!
x helloWorld $
```

Reminder:
For a function to be available outside a package,
the function name must be uppercase

The screenshot shows an IDE window titled "hello.go - GolangTraining - [~/Documents/go/src/github.com/goestoeleven/GolangTraining]". The left sidebar displays the project structure, with "reverse.go" selected under the "stringutil" package. The main editor shows the code in "reverse.go":

```
1 package main
2
3 import (
4     "fmt"
5     "github.com/goestoeleven/GolangTraining/02_library/stringutil"
6 )
7
8 func main() {
9     fmt.Println(stringutil.reverseTwo("!oG ,olleH"))
10 }
```

A pink line connects the "reverse.go" file in the sidebar to the "reverseTwo" function call in the code. A white box with a black border contains the text: "reverseTwo is available inside the stringutil package ... but not outside the package." The bottom terminal shows the command "go run hello.go" and the resulting error: "cannot refer to unexported name stringutil.reverseTwo". A red diagonal banner in the bottom right corner reads "code doesn't run".

reverseTwo is available inside the stringutil package ...
but not outside the package.

code doesn't run

```
helloWorld $ go run hello.go
# command-line-arguments
./hello.go:9: cannot refer to unexported name stringutil.reverseTwo
./hello.go:9: undefined: stringutil.reverseTwo
helloWorld $
```

GolangTraining > 02_library > helloWorld > hello.go

Project

1: Project

2: Structure

3: Favorites

4: GolangTraining (~/Documents/go/src/github.com/goestoeleven/GolangTraining)

- 01_helloWorld
 - firstFile.go
- 02_library
 - helloWorld
 - hello.go
 - stringutil
 - capitalize.go
 - lowercase.go
 - reverse.go
 - sentenceCase.go
- .gitignore
- README.md
- External Libraries
 - Go SDK
 - GOPATH <GolangTraining>

```
1 package main
2
3 import (
4     "fmt"
5     "github.com/goestoeleven/GolangTraining/02_library/stringutil"
6 )
7
8 func main() {
9     fmt.Println(stringutil.Reverse("!oG ,olleH"))
10 }
```

```
1 package stringutil
2
3 // Package stringutil contains utility functions for working with strings.
4 // Reverse returns its argument string reversed rune-wise left to right.
5
6 func Reverse(s string) string {
7     return reverseTwo(s)
8 }
9
10
```

```
1 package stringutil
2
3 func reverseTwo (s string) string {
4     r := []rune(s)
5     for i, j := 0, len(r)-1; i < len(r)/2; i, j = i+1, j-1 {
6         r[i], r[j] = r[j], r[i]
7     }
8     return string(r)
9 }
10
```

Terminal

```
+ helloWorld $ go run hello.go
Hello, Go!
X helloWorld $
```

what is this demonstrating?

GolangTraining > 02_library > helloWorld > hello.go

Project

GolangTraining (~/Documents/go/src/github.com/goestoeleven/GolangTraining)

01_helloWorld

firstFile.go

02_library

helloWorld

hello.go

stringutil

capitalize.go

lowercase.go

reverse.go

sentenceCase.go

.gitignore

README.md

External Libraries

Go SDK

GOPATH <Go>

Public function

not public function

```
1 package main
2
3 import (
4     "fmt"
5     "github.com/goestoeleven/GolangTraining/02_library/stringutil"
6 )
7
8 func main() {
9     fmt.Println(stringutil.Reverse("!oG ,olleH"))
10 }
```

```
1 package stringutil
2
3 // Package stringutil contains utility functions for working with strings.
4 // Reverse returns its argument string reversed rune-wise left to right.
5
6 func Reverse(s string) string {
7     return reverseTwo(s)
8 }
9
10
```

```
1 package stringutil
2
3 func reverseTwo(s string) string {
4     r := []rune(s)
5     for i, j := 0, len(r)-1; i < len(r)/2; i, j = i+1, j-1 {
6         r[i], r[j] = r[j], r[i]
7     }
8     return string(r)
9 }
10
```

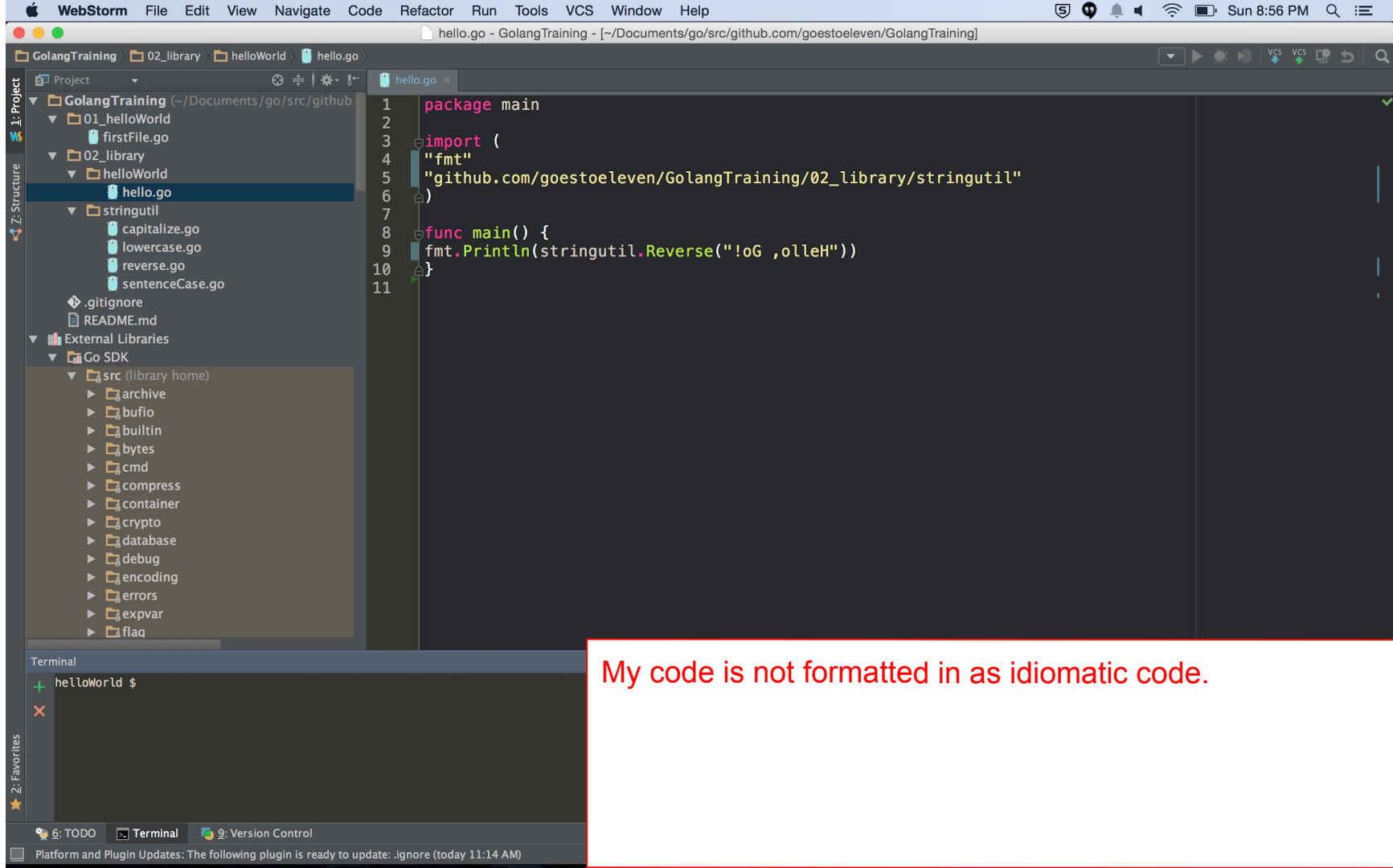
what is this demonstrating?

Terminal

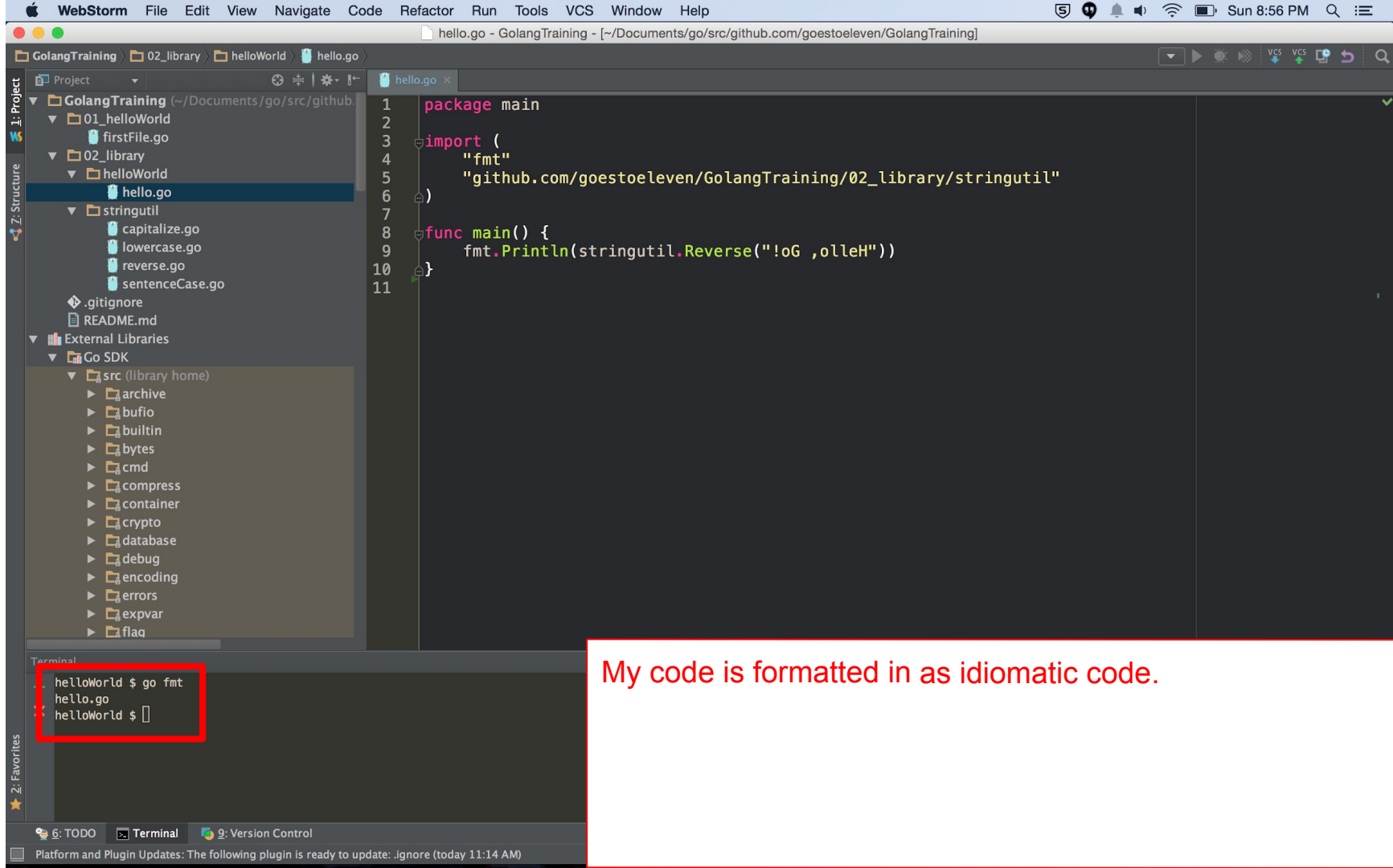
```
+ helloWorld $ go run hello.go
Hello, Go!
X helloWorld $
```

go fmt

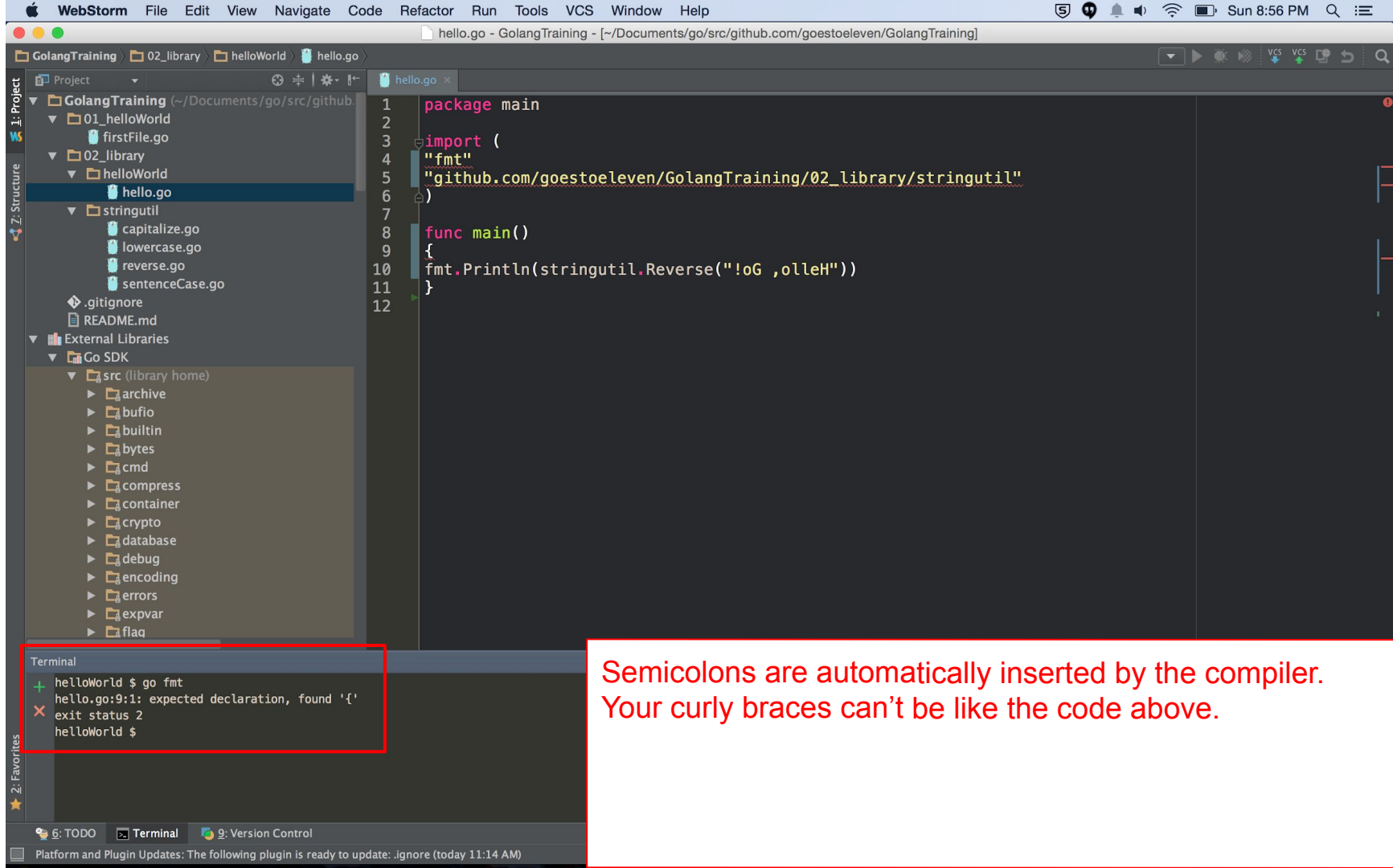
formats your code



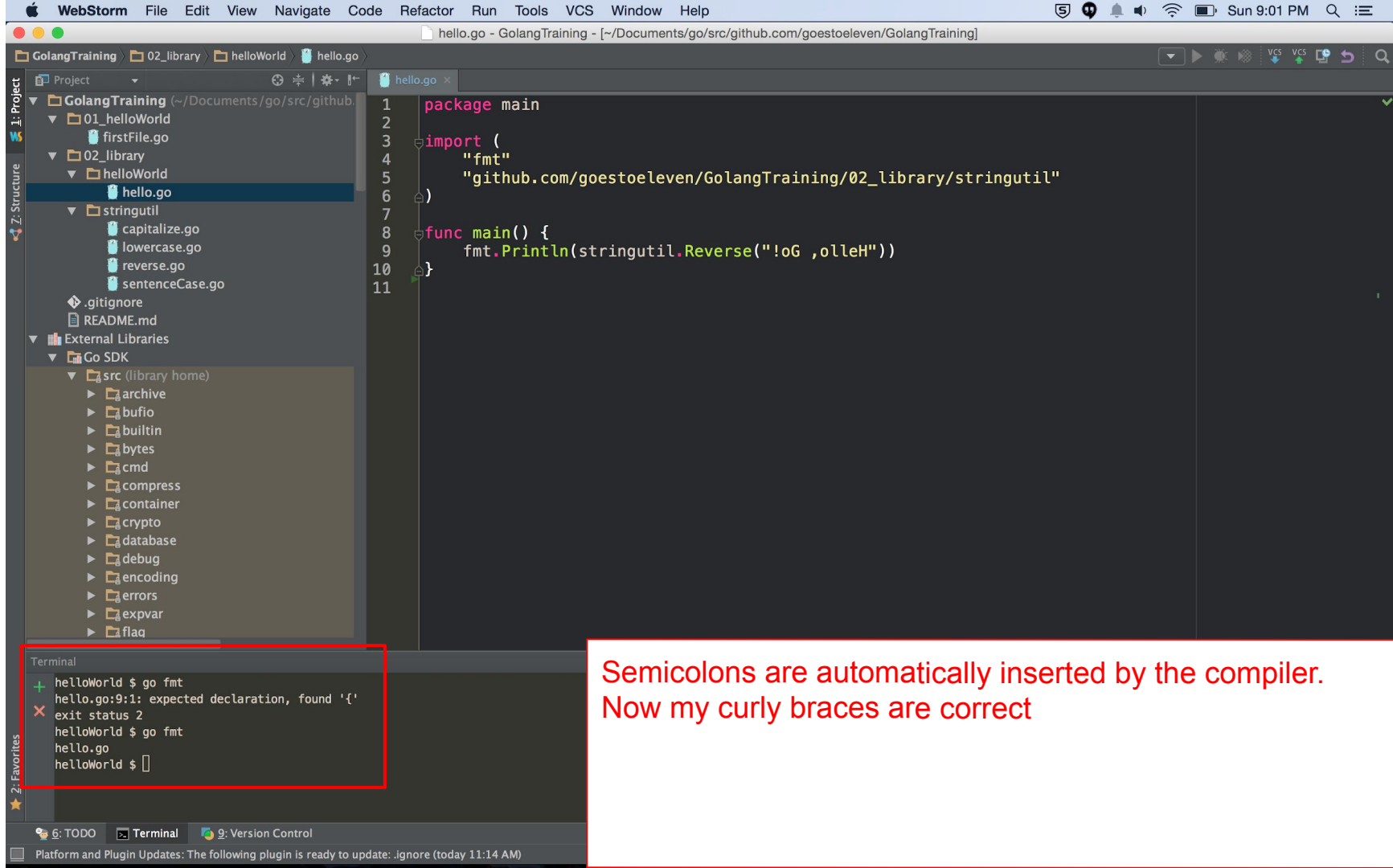
My code is not formatted in as idiomatic code.



My code is formatted in as idiomatic code.



Semicolons are automatically inserted by the compiler.
Your curly braces can't be like the code above.



Review

- package name must match folder name
 - except for **package main**
- package main
 - must have **func main() {}**
 - can contain other functions also
- packages can be organized into any number of files
- capitalization
 - Public
 - private
- camelCase
- **go fmt**

Review Questions

package main

Does **package main** have to be in a folder called **main**?

package main

Does **package main** have to have a func called **main**?

camelCase

True or False:

idiomatic go uses camelCase.

go fmt

- Write “hello go” again, indenting your code all of the way to the left
 - take a screenshot of this
- run **go fmt** at the terminal
 - take a screenshot of this