Variables & Scope

variables, zero value, scope, capitalization

“There are only two hard things in Computer Science:
cache invalidation and naming things.”

~ Phil Karlton

variables

```go
package main

import "fmt"

func main() {
    var message string
    message = "Hello World."
    fmt.Print(message)
}
```

```go
package main

import "fmt"

func main() {
    var message string
    message = "Hello World."
    fmt.Print(message)
    fmt.Println(message)
    fmt.Println(message)
}
```

```go
package main

import "fmt"

func main() {
    var message string
    var a, b, c int
    a = 1

    message = "Hello World!"

    fmt.Println(message, a, b, c)
}
```

```go
package main

import "fmt"

func main() {

    var message string = "Hello World!"
    var a, b, c int = 1, 2, 3

    fmt.Println(message, a, b, c)
}
```

```go
package main

import "fmt"

func main() {

    var message = "Hello World!"
    var a, b, c = 1, 2, 3

    fmt.Println(message, a, b, c)
}
```

```go
package main

import "fmt"

func main() {

    var message = "Hello World!"
    var a, b, c = 1, false, 3

    fmt.Println(message, a, b, c)
}
```

```go
package main

import "fmt"

func main() {

    // you can only do this inside a func
    message := "Hello World!"
    a, b, c := 1, false, 3

    fmt.Println(message, a, b, c)
}
```

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

GolangTraining 〉 03_variables 〉 01_variables 〉 variables.go

Project

- GolangTraining (~/Documents/go/src/github.c
  - 01_helloWorld
  - 02_library
  - 03_variables
    - 01_variables
      - variables.go
    - 02_typeOf
    - 03_constants
    - 04_priv_pub
  - .gitignore
  - README.md
- External Libraries
  - Go SDK
  - GOPATH <GolangTraining>

variables.go

```go
package main

import "fmt"

var a string = "this is stored in the variable a" // package scope
var b, c string = "stored in b", "stored in c" // package scope
var d string // package scope

func main() {

    d = "stored in d" // declaration above; assignment here; package scope
    var e int = 42 // function scope - subsequent variables have same package scope:
    f := 43
    g := "stored in g"
    h, i := "stored in h", "stored in i"
    j, k, l, m := 44.7, true, false, 'm' // single quotes
    n := "n" // double quotes
    o := `o` // back ticks

    fmt.Println("a - ", a)
    fmt.Println("b - ", b)
    fmt.Println("c - ", c)
    fmt.Println("d - ", d)
    fmt.Println("e - ", e)
    fmt.Println("f - ", f)
    fmt.Println("g - ", g)
    fmt.Println("h - ", h)
    fmt.Println("i - ", i)
    fmt.Println("j - ", j)
    fmt.Println("k - ", k)
    fmt.Println("l - ", l)
    fmt.Println("m - ", m)
    fmt.Println("n - ", n)
    fmt.Println("o - ", o)
}
```

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

```go
package main

import "fmt"

var a string = "this is stored in the variable a" // package sc
var b, c string = "stored in b", "stored in c" // package scope
var d string // package scope

func main() {

    d = "stored in d" // declaration above; assignment here; pa
    var e int = 42 // function scope - subsequent variables hav
    f := 43
    g := "stored in g"
    h, i := "stored in h", "stored in i"
    j, k, l, m := 44.7, true, false, 'm' // single quotes
    n := "n" // double quotes
    o := `o` // back ticks

    fmt.Println("a - ", a)
    fmt.Println("b - ", b)
```

Terminal

```
03_variables $ cd 01_variables/
01_variables $ go run variables.go
a -  this is stored in the variable a
b -  stored in b
c -  stored in c
d -  stored in d
e -  42
f -  43
g -  stored in g
h -  stored in h
i -  stored in i
j -  44.7
k -  true
l -  false
m -  109
n -  n
o -  o
01_variables $
```

[Lexical Elements]

lexical

lex·i·cal

/ˈleksək(ə)l/

adjective

of or relating to the words or vocabulary of a language.
"lexical analysis"

- relating to or of the nature of a lexicon or dictionary.
  "a lexical entry"

Lexical Elements

declare, assign, initialize

declare + assign = initialize

exercise

declare a variable of type string
assign it the value of your name
use the variable in an statement

zero value

false for booleans, 0 for integers, 0.0 for floats, "" for strings
nil for pointers, functions, interfaces, slices, channels, and maps

https://golang.org/ref/spec#The_zero_value

scope

universe → package → file → function → curly braces

https://golang.org/ref/spec#Declarations_and_scope

scope.go

```go
package main

import "fmt"

var x int

func main() {
    fmt.Println(x)
}
```

package level scope

WebStorm File Edit View Navigate Code Refactor Run Tools VCS Window Help

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

GolangTraining > 05_scope > 02_package-scope > dukes > main.go

Project

- GolangTraining (~/Documents/go/src/github.com/go
  - 01_helloWorld
  - 02_library
  - 03_variadic
  - 04_variables
  - 05_scope
    - 01_package-scope
    - 02_package-scope
      - characters
      - dukes
  - 06_constants
  - 07_memory-address
  - a06_remainder
  - a07_for-loop
  - a08_switch-statements
  - a09_conditionals
  - a10_exercises
  - xx06_fmt-package
  - xx09_typeOf
  - .gitignore
  - README.md
- External Libraries

model.go

```go
package characters

var sherrif string = "Boss Hogg"
```

controller.go

```go
package characters

func Sherrif() string {
    return sherrif
}
```

main.go

```go
package main

import (
    "fmt"
    "github.com/goestoeleven/GolangTraining/05_scope/02_package-scope/characters"
)

func main() {
    fmt.Println(characters.Sherrif())
}
```

Terminal

```
dukes $ go run main.go
Boss Hogg
dukes $
```

package level scope

model.go

```go
package characters

var sherrif string = "Boss Hogg"

```

controller.go

```go
package characters

func Sherrif() string {
    return sherrif
}
```

main.go

```go
package main

import (
    "fmt"
    "github.com/goestoeleven/GolangTraining/05_scope/02_package-scope/characters"
)

func main() {
    fmt.Println(characters.Sherrif())
    fmt.Print(characters.sherrif) // not valid - can't access the variable
}
```

invalid

variables declared at the top level (outside a function) have a package scope

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

GolangTraining 〉 05_scope 〉 02_package-scope 〉 dukes 〉 main.go

Project

- GolangTraining (~/Documents/go/src/github.com/
  - 01_helloWorld
  - 02_library
  - 03_variadic
  - 04_variables
  - 05_scope
    - 01_package-scope
    - 02_package-scope
      - characters
        - controller.go
        - model.go
      - dukes
        - main.go
    - 03_pacakge-scope
  - 06_constants
  - 07_memory-address
  - a06_remainder
  - a07_for-loop
  - a08_switch-statements
  - a09_conditionals
  - a10_exercises
  - xx06_fmt-package
  - xx09_typeOf
  - .gitignore
  - README.md

model.go

```go
package characters

var SherrifName string = "Boss Hogg"
```

Capitalized variables and functions are accessible outside the package; like “public” in some languages

controller.go

```go
package characters

func Sherrif() string {
    return SherrifName
}
```

main.go

```go
package main

import (
    "fmt"
    "github.com/goestoeleven/GolangTraining/05_scope/02_package-scope/characters"
)

func main() {
    fmt.Println(characters.Sherrif())
    fmt.Println(characters.SherrifName) // valid - can access the variable
}
```

Terminal

```
Boss Hoggdukes $ go run main.go
Boss Hogg
Boss Hogg
dukes $
```

scope.go

```go
package main

import "fmt"

var x string = "Boss Hogg"

func oneFunc() {
    fmt.Println(x)
}

func twoFunc() {
    fmt.Println(x)
}

func main() {
    oneFunc()
    twoFunc()
}
```

package level scope

scope.go

```go
package main

import "fmt"

func oneFunc() {
    var x string = "Boss Hogg"
    fmt.Println(x)
}

func twoFunc() {
    fmt.Println(x)
}

func main() {
    oneFunc()
    twoFunc()
}
```

invalid

scope of variable x is within the function in which it is declared

WebStorm File Edit View Navigate Code Refactor Run Tools VCS Window Help

one.go - GolangTr

GolangTraining > 05_scope > 07_invalid-scope > one.go

Project

- 01_helloWorld
- 02_library
- 03_variadic
- 04_variables
- 05_scope
  - 01_package-scope
  - 02_package-scope
  - 03_invalid-scope
  - 04_package-scope
  - 05_pacakge-scope
  - 06_invalid-scope
  - 07_invalid-scope
    - another.go
    - one.go
- 06_constants
- 07_memory-address
- a06_remainder
- a07_for-loop
- a08_switch-statements
- a09_conditionals

one.go

```
1  package main
2
3  import "fmt"
```

another.go

```
1  package main
2
3  func main() {
4      fmt.Println("Hello")
5  }
6
```

invalid
imports have a file scope

scope.go

```go
package main

import "fmt"

func main() {
    x := 42
    fmt.Println(x)
}
```

func level scope

scope.go

```go
package main

import "fmt"

func main() {
    fmt.Println(x)
    x := 42
}
```

invalid

you can’t use a variable before it is initialized; there is no hoisting like in javascript

scope.go

```go
package main

import "fmt"

func main() {
	x := 42
	another()
}

func another() {
	fmt.Println(x)
}
```

invalid

scope of variable x is within the function in which it is declared

scope.go

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello")
    {
        x := 42
        fmt.Println(x)
    }
}
```

curly-braces level scope

scope.go

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello")
    {
        x := 42
        fmt.Println(x)
    }
        fmt.Println(x)
}
```

invalid

scope of variable x is within the curly-braces in which it is declared

resource

learn more about scope

capitalization

Public, private

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

GolangTraining 05_scope 02_package-scope dukes main.go

Project

GolangTraining (~/Documents/go/src/github.com/g

- 01_helloWorld
- 02_library
- 03_variadic
- 04_variables
- 05_scope
  - 01_package-scope
  - 02_package-scope
    - characters
      - controller.go
      - model.go
    - dukes
      - main
  - 03_pacal
- 06_const
- 07_m
- a0
- a07
- a08_sw -statements
- a09_conditionals
- a10_exercises
- xx06_fmt-package
- xx09_typeOf
- .gitignore
- README.md

We already saw this

model.go

```go
package characters

var SherrifName string = "Boss Hogg"
```

Capitalized variables and functions are accessible outside the package; like “public” in some languages

controller.go

```go
package characters

func Sherrif() string {
    return SherrifName
}
```

main.go

```go
package main

import (
    "fmt"
    "github.com/goestoeleven/GolangTraining/05_scope/02_package-scope/characters"
)

func main() {
    fmt.Println(characters.Sherrif())
    fmt.Println(characters.SherrifName) // valid - can access the variable
}
```

Terminal

```
Boss Hoggdukes $ go run main.go
Boss Hogg
Boss Hogg
dukes $
```

Capitalized top-level variables, and functions, are accessible outside the package.

WebStorm File Edit View Navigate Code Refactor Run Tools VCS Window Help

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

GolangTraining > 05_scope > 14_capitalization > dukes > main.go

Project

- GolangTraining (~/Documents/go/src/github.com/go
  - 01_helloWorld
  - 02_library
  - 03_variadic
  - 04_variables
  - 05_scope
    - 01_package-scope
    - 02_package-scope
    - 03_invalid-scope
    - 04_package-scope
    - 05_pacakge-scope
    - 06_invalid-scope
    - 07_invalid-scope
    - 08_func-scope
    - 09_invalid-scope
    - 10_invalid-scope
    - 11_braces-scope
    - 12_invalid-scope
    - 13_capitalization
    - 14_capitalization
      - characters
        - controller.go
        - model.go
      - dukes
        - main.go
  - 06_constants

model.go

```go
package characters

var SherrifName string = "Boss Hogg"

```

controller.go

```go
package characters

func Sherrif() string {
    return SherrifName
    SideKick := "Roscoe"
}
```

main.go

```go
package main

import (
    "fmt"
    "github.com/goestoeleven/GolangTraining/05_scope/14_capitalization/characters"
)

func main() {
    fmt.Println(characters.Sherrif())
    fmt.Println(characters.SherrifName) // valid - can access the variable
    fmt.Println(characters.SideKick) // invalid - cannot access the variable
}
```

invalid

scope on SideKick is func scope

Capitalized top-level variables, and functions, are accessible outside the package.

WebStorm File Edit View Navigate Code Refactor Run Tools VCS Window Help

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

GolangTraining 05_scope 15_capitalization characters controller.go

Project

- GolangTraining (~/Documents/go/src/github.com/
  - 01_helloWorld
  - 02_library
  - 03_variadic
  - 04_variables
  - 05_scope
    - 01_package-scope
    - 02_package-scope
    - 03_invalid-scope
    - 04_package-scope
    - 05_pacakge-scope
    - 06_invalid-scope
    - 07_invalid-scope
    - 08_func-scope
    - 09_invalid-scope
    - 10_invalid-scope
    - 11_braces-scope
    - 12_invalid-scope
    - 13_capitalization
    - 14_capitalization
    - 15_capitalization
      - characters
        - controller.go
        - model.go
      - dukes
        - main.go
  - 06_constants
  - 07_memory-address
  - a06_remainder
  - a07_for-loop
  - a08_switch-statements

model.go

```go
package characters

var SherrifName string = "Boss Hogg"
var sideKick = "Roscoe"
```

controller.go

```go
package characters

func Sherrif() string {
    return SherrifName
}

func SideKick() string {
    return sideKick
}
```

main.go

```go
package main

import (
    "fmt"
    "github.com/goestoeleven/GolangTraining/05_scope/15_capitalization/characters"
)

func main() {
    fmt.Println(characters.Sherrif())
    fmt.Println(characters.SherrifName)
    fmt.Println(characters.SideKick())
}
```

Terminal

```
dukes $ go run main.go
Boss Hogg
Boss Hogg
Roscoe
dukes $
```

model.go

```go
package characters

var SherrifName string = "Boss Hogg"
var sideKick = "Roscoe"
```

controller.go

```go
package characters

func Sherrif() string {
	return SherrifName
}

func sideK() string {
	return sideKick
}
```

main.go

```go
package main

import (
	"fmt"
	"github.com/goestoeleven/GolangTraining/05_scope/16_invalid-capitalization/characters"
)

func main() {
	fmt.Println(characters.Sherrif())
	fmt.Println(characters.SherrifName)
	fmt.Println(characters.sideK())
}
```

invalid

scope on **sideK()** is package scope

exercise

- create a package main
- create a package names
  - create the variable **MyName** and assign it a name
    - use MyName in package main
      - take a screenshot
  - create the variable **yourName** and assign it a name
    - what happens when you use yourName in package main?
      - take a screenshot

_ blank identifier

for a variable you don’t want to use

variables.go

```go
package main

import "fmt"

var a string = "this is stored in the variable a"
var b, c string = "stored in b", "stored in c"

func main() {

    fmt.Println("a - ", a)
    fmt.Println("b - ", b)
    fmt.Println("c is not being used - ") // invalid code
}
```

variables.go

```go
package main

import "fmt"

var a string = "this is stored in the variable a"
var b, _ string = "stored in b", "stored then thrown away"

func main() {

    fmt.Println("a - ", a)
    fmt.Println("b - ", b)
    fmt.Println("c is not being used -  and this is no problem")
}
```

```go
for key, value := range oldMap {
    newMap[key] = value
}
```

```go
sum := 0
for _, value := range array {
    sum += value
}
```

Review

- can occur anywhere in your code
  - var a string = "this is stored in the variable a"
  - var b, c string = "stored in b", "stored in c"
  - var d string
  - const p string = "death & taxes"
  - const q = 42
- this lexical element **:=** can only occur inside a function
  - f **:=** 43
  - g **:=** "stored in g"
  - h, i **:=** "stored in h", "stored in i"
  - j, k, l, m **:=** 44.7, true, false, 'm' // single quotes
  - n **:=** "n"
  - o **:=** `o` // backticks
- fmt.Println("a - ", a)
- declare + assign = initialize
- capitalization
  - Public
  - private
- blank identifier

Review Questions

lexical elements

What are lexical elements? Give a few examples of lexical elements in golang.

golang spec

- Open the **golang spec**
- Under lexical elements, find the golang keywords
  - take a screenshot of this

golang spec

- Open the **golang spec**
- What are the two types of comments which may be used in golang code?

golang spec

- Open the **golang spec**
- What is an identifier?

golang spec

- Open the **golang spec**
- According to the golang spec, the first character in an identifier must be what?

golang spec

- Open the **golang spec**
- Find the operators which you can use in golang.
  - take a screenshot of this

zero value

- Describe zero value and how it occurs

zero value

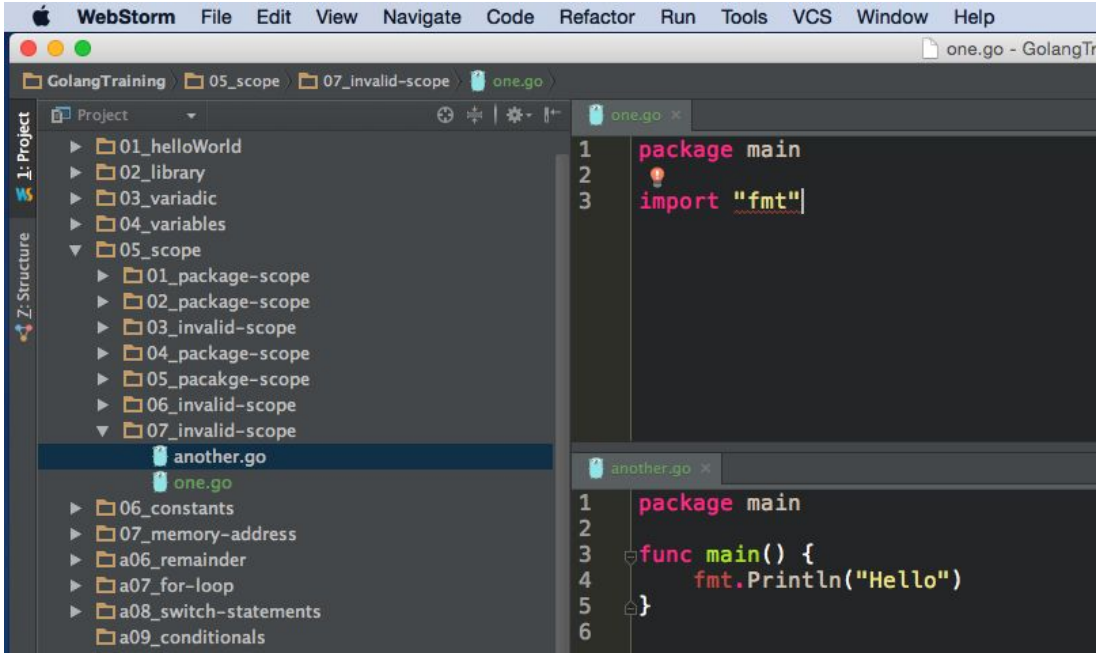
- What are the zero values for each of the following:
  - int
  - float
  - string
  - pointer
  - boolean

scope

- What are the levels of scope within golang?

scope

- When you import a package in a file, what is that package's scope? For example, is "fmt" available in file f1.go?



scope

- When you declare a variable at the top-level (not in a function), and when you make that declaration in a file that is in a package, what is that variable's scope? For example, is the code below valid?

model.go

```go
package characters

var SherrifName string = "Boss Hogg"
```

controller.go

```go
package characters

func Sherrif() string {
    return SherrifName
}
```

scope

- Is this code valid? Why or why not?

```go
func oneFunc() {
    var x string = "Boss Hogg"
    fmt.Println(x)
}

func twoFunc() {
    fmt.Println(x)
}
```

scope

- Is this code valid? Why or why not?

one.go

```go
package main

import "fmt"
```

another.go

```go
package main

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

scope

- Is this code valid? Why or why not?

```go
func main() {
    fmt.Println(x)
    x := 42
}
```

scope

- Is this code valid? Why or why not?

```go
func main() {
    fmt.Println("Hello")
    {
        x := 42
        fmt.Println(x)
    }
        fmt.Println(x)
}
```

scope

- How is the scope of a function changed when you **capitalize** the first letter of a function?

scope

- How is the scope of a variable changed when you **capitalize** the first letter of a variable?

scope

- Can you **capitalize** func main? Would you want to?