Introduction to Go

WomenWhoGo March’18

Speakers: Rounaq Jhunjhunu Wala, Ambar Pal

Programming

Making the computer do what you want

Overview

1. Comments
   a. Single Line
   b. Multi Line
2. Starting Declaration to let Go know we are writing a executable, instead of a package
3. Function Declarations
   a. The main Function
   b. Multiple Parameters
   c. Multiple Return Values
   d. Implicit Returns
4. Variables
   a. Type
   b. Declaration
   c. Assignment
   d. Unused Variables in Go

Overview 2

5. Arrays and Slices
    a. Declaration
    b. Slicing
6. Maps
7. Pointers
8. Automatic Garbage Collection
9. Flow Control
    a. If Else
    b. Switch

Printing

```go
package main

import "fmt"

func main() {
    fmt.Println("HelloWorld")
    fmt.Print("HelloWorld\n")
    fmt.Printf("A %drd %s\n", 3, "HelloWorld")
}
```

Functions - Single Return Value

```
package main
import "fmt"

func add(a int, b int) int {
    return a + b
}

func main() {
    fmt.Println(add(a, b))
}
```

Functions - Multiple Return Values

```
package main
import "fmt"

func addsub(a int, b int) (int, int) {
    return a + b, a - b
}

func main() {
    fmt.Println(addsub(3, 4))
}
```

Functions - Implicit Return Values

```go
package main

import "fmt"

func mul(a int, b int) (res int) {
    res = a * b
    return
}

func main() {
    fmt.Println(mul(3, 4))
}
```

Variables

```
package main
import "fmt"

//global var
var c int = 2 * 3

func main() {
    var a int = 4
    b := 3
    fmt.Println(a + b + c)
}
```

Variables - 2 and Data Types - 1

```
package main
import "fmt"

//global var
var a,b int = 2,3

func main() {
    var c float32 = 4.0
    fmt.Println(float32(a + b) + c)
}
```

Data Types - 2

```
package main
import "fmt"

func main() {
    var a int = 4
    var b float32 = 3.0
    var c string = "str"
    fmt.Printf("%d + %f != %s\n", a, b, c);
}
```

Data Types - Generic Printing, Complex

```
package main
import "fmt"

func main() {
    var a int = 4
    var b float32 = 3.0
    var c string = "str"
    fmt.Printf("%v + %v != %v\n", a, b, c);
    fmt.Printf("%v", 3 + 4i)
}
```

Arrays

```
package main
import "fmt"
//global var
var c int = 2

func main() {
    var a = [4]int{1,2,3,4}
    var b = [2]float32{2.3, 4.3}
    fmt.Printf("%d %f\n", a[2], b[0]);
    fmt.Printf("%v %v", a, b);
}
```

Slices

```
package main
import "fmt"

func main() {
    var a = []int{1,2,3,4};
    var b = [...]int{1,2,3,4};
    var c = [4]int{1,2,3,4};

    fmt.Printf("Slice:%v \nFixed Size Array:%v, %v", a, b,
c);
}
```

Slices - 2

```
package main
import "fmt"
func main() {
    var a = []int{1,2,3,4};
    var b = [...]int{1,2,3,4};
    fmt.Printf("Slice:%v \nArray:%v\n", a, b);

    // c = append(c, 'a', 'b', 'c')
    // fmt.Printf("Append to Array?: %v", c)

    a = append(a, 'a', 'b', 'c')
    fmt.Printf("Append to Slice?: %v", a)
}
```

If - Else

```
package main
import "fmt"

func main() {
    if a := 3;a > 3 {
        fmt.Println("It is more than three")
    } else {
        fmt.Println("It's time to leave now")
    }
}
```

Switch-Case

```
package main
import “fmt”
func main() {
    a := 3
    switch a {
        case 3:
            fmt.Println("It is indeed three")
            break
        default:
            fmt.Println("I don't know what it is")
    }
}
```

For Loop

```
package main
import "fmt"

func main() {
    for i := 4; i >= 0; i -- {
        fmt.Printf("%d minutes left\n", i)
    }
}
```

For Loop - 2

```
package main
import "fmt"

func main() {
    for {
        fmt.Printf("May the force be with you")
    }
}
```

Taking Input

```
package main
import (
    "bufio"
    "fmt"
    "os"
)
func main() {
    reader := bufio.NewReader(os.Stdin)
    name, _ := reader.ReadString('\n')
    fmt.Println("You entered " + name)
}
```

Taking Input - 2

```
package main
import "fmt"

func main() {
    var fname string
    fmt.Scanf("%s",&fname)
    fmt.Printf("You entered %s\n",fname)
}
```

Pointers

```go
package main
import "fmt"

func increment(xptr *int) {
    *xptr ++;
}

func main() {
    x := 0
    increment(&x)
    fmt.Println(x)
}
```

More Pointers

```
package main
import "fmt"

func swap(xptr, yptr *int) {
    xptr, yptr = yptr, xptr
}

func main() {
    x,y := 0,1
    swap(&x, &y)
    fmt.Println(x, y)
}
```

New

```
package main
import "fmt"

func main() {
    xptr = new(int)
    *xptr = 3
    fmt.Println(*xptr)
}
```

Structs and Methods

```
package main
import (
    "fmt"
    m "math"
)

type Point struct {
    x, y float64
}
```

```
func (p *Point) norm()
float64 {
    return m.Sqrt(p.x*p.x +
p.y*p.y)
}

func main() {
    p := Point{x:3,y:4}
    fmt.Println(p.norm())
}
```