Part 8: if else statement

06 APRIL 2017

This is tutorial number 8 in Golang tutorial series.

if is a conditional statement. The syntax of the if statement is

```
if condition {
}
```

If the `condition` is true, the lines of code between `{` and `}` is executed.

Unlike in other languages like C, the `( )` are mandatory even if there is only one statement between the `( )`.

The if statement also has optional `else if` and `else` components.

```
if condition {
} else if condition {
} else {
}
```

There can be any number of `else if`s. The condition is evaluated for truth from the top to bottom. Which ever `if` or `else if`'s condition evaluates to true, the corresponding block of code is executed. If none of the conditions are true then `else` block is executed.

Lets write a simple program to find if a number is odd or even.

```
package main

import (
    "fmt"
)

func main() {
    num := 10
    if num % 2 == 0 { //checks if number is even
        fmt.Println("the number is even")
    }  else {
        fmt.Println("the number is odd")
    }
}
```

The `if num % 2 == 0` statement checks whether the reminder of dividing a number by 2 is zero. It it is, then "the number is even" is printed else "the number is odd" is printed. In the above program `the number is even` is printed.

There is one more variant of `if` which includes a optional `statement` component which is executed before the condition is evaluated. Its syntax is

```
if statement; condition {
}
```

Lets rewrite the program which finds whether the number is even or odd using the above syntax.

```
package main

import (
    "fmt"
)

func main() {
    if num := 10; num % 2 == 0 { //checks if number is e
        fmt.Println(num,"is even")
    }  else {
        fmt.Println(num,"is odd")
    }
}
```

In the above program `num` is initialised in the `if` statement. One thing to be noted is that `num` is available only for access from inside the `if` and `else`. i.e. the scope of `num` is limited to the `if else` blocks. If we try to access `num` from outside the `if` or `else`, the compiler will complain.

Lets write one more program which uses `else if`.

```
package main

import (
    "fmt"
)

func main() {
    num := 99
    if num <= 50 {
        fmt.Println("number is less than or equal to 50")
    } else if num >= 51 && num <= 100 {
        fmt.Println("number is between 51 and 100")
    } else {
        fmt.Println("number is greater than 100")
    }

}
```

In the above program `else if num >= 51 && num <= 100` is true and hence the program will output `number is between 51 and 100`

Thats it for if statement. Hope you enjoyed reading. Please leave your valuable comments and feedback.

Get the free Golang tools cheat sheet

Next tutorial - Loops

Naveen Ramanathan

iOS developer at dietco.de and Golang enthusiast.