IF Statement

This lesson explains how to use if statements in Go using examples

WE'LL COVER THE FOLLOWING

Comparison With Other Languages

The `if` statement looks as it does in C or Java, except that the `( )` are gone and the `{ }` are required. Like `for`, the `if` statement can start with a short statement to execute before the condition. Variables declared by the statement are only in scope until the end of the `if`. Variables declared inside an `if` short statement are also available inside any of the `else` blocks.

- Example of an `if` statement:

```
if answer != 42 {
    return "Wrong answer"
}
```

Here's a complete example to help you better understand the concept:

```
package main

import (
        "fmt"
        "math"
)

func sqrt(x float64) string {
        if x < 0 {
                return sqrt(-x) + "i"
        }
        return fmt.Sprint(math.Sqrt(x))
}

func main() {
```

```
func main() {
        fmt.Println(sqrt(2), sqrt(-4))
}
```

- Example of an `if` short statement with a variable declared within the statement:

```
if err := foo(); err != nil {
    panic(err)
}
```

Here's the complete example using this concept:

```
package main

import (
        "fmt"
        "math"
)

func pow(x, n, lim float64) float64 {
        if v := math.Pow(x, n); v < lim {
                return v
        }
        return lim
}

func main() {
        fmt.Println(
                pow(3, 2, 10),
                pow(3, 3, 20),
        )
}
```

Example

- Example of an `if` and `else` block:

Environment Variables

Key:	Value:
GOPATH	/go

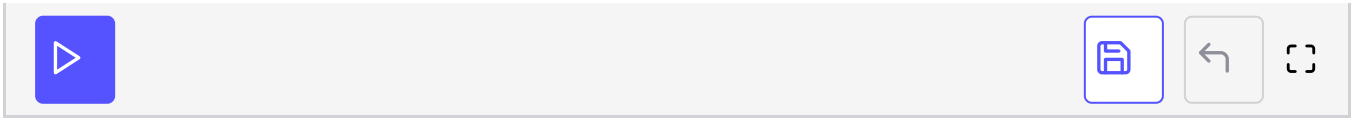
```
package main
```

```
import (
        "fmt"

        "math"
)

func pow(x, n, lim float64) float64 {
        if v := math.Pow(x, n); v < lim {
                return v
        } else {
                fmt.Printf("%g >= %g\n", v, lim)
        }
        // can't use v here, though
        return lim
}

func main() {
        fmt.Println(
                pow(3, 2, 10),
                pow(3, 3, 20),
        )
}
```



Now that we've seen the usage of `if` statements in Go, we will move on to discuss `for` loops in Go in the next lesson.