

# FOR Loop

This lesson discusses for loops, including, for loop without a statement using examples and their use as an alternative to the while loop in Go

## WE'LL COVER THE FOLLOWING



- Comparison with Other Languages
- **for** Loop Syntax
- For Loops Without Statements
- For loop as an Alternative to While
- Infinite Loops

## Comparison with Other Languages #

Go has only one looping construct, the for loop. The basic for loop looks as it does in C or Java, except that the `( )` are gone (they are not even optional) and the `{ }` are required. As in C or Java, you can leave the pre and post statements empty.

### **for** Loop Syntax #

- Example of a **for** loop:

```
package main


import "fmt"

func main() {
    sum := 0
    for i := 0; i < 10; i++ {
        sum += i
    }
    fmt.Println(sum)
}
```



## For Loops Without Statements #

- Example of a `for` loop without pre/post statements:





Environment Variables 

Key:	Value:
GOPATH	/go

```
package main

import "fmt"

func main() {
    sum := 1
    for ; sum < 1000; { //iterate as long as sum<1000
        sum += sum
    }
    fmt.Println(sum)
}
```







## For loop as an Alternative to While #

- Using a `for` loop like a `while` loop is used:

```
package main

import "fmt"

func main() {
    sum := 1
    for sum < 1000 {
        sum += sum
    }
    fmt.Println(sum)
}
```





## Infinite Loops #

- Infinite `for` loop:

```
package main

func main() {
```



```
    for {  
        // do something in a loop forever  
  
    }  
}
```

Now that we have covered the usage of `for` loop in Go, we will discuss another popular control flow feature, switch statements.