Go For Loop

For statement is used for repeating a set of statements number of times.

It is the Only loop in go language.

There are two types of loop in Go language :-

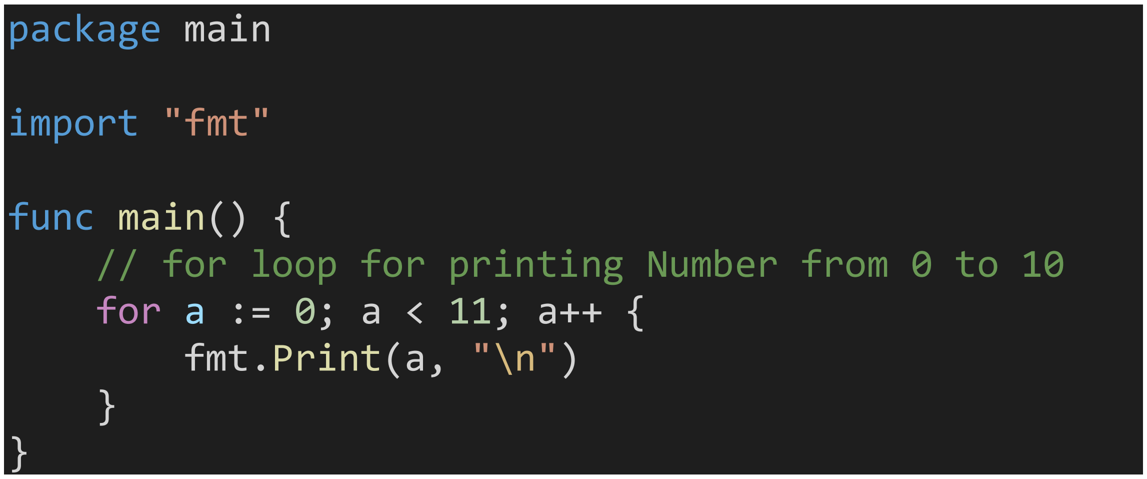
1. Counter-controlled iteration
2. Condition-controlled iteration.

When the execution of the loop is over, the objects created inside the loop gets destroyed.

- **Counter-controlled iteration:**

Example:-

# Simple for Loop

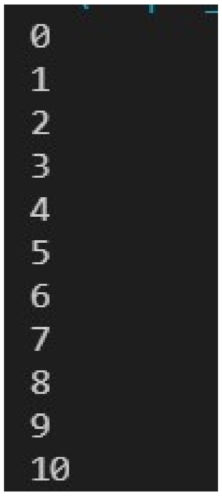


```go
package main

import "fmt"

func main() {
    // for loop for printing Number from 0 to 10
    for a := 0; a < 11; a++ {
        fmt.Print(a, "\n")
    }
}
```

Output:-



```
0
1
2
3
4
5
6
7
8
9
10
```

# Nested For Loop :- For loop inside a for loop is called Nested For Loop.

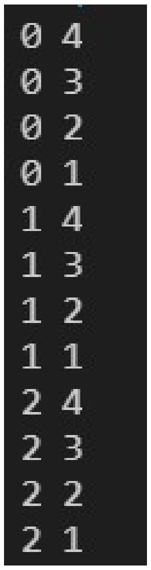
Example:-

```go
package main

import "fmt"

func main() {
    // nested for loop :- loop inside a loop
    for a := 0; a < 3; a++ {
        for b := 4; b > 0; b-- {
            fmt.Print(a, " ", b, "\n")
        }
    }
}
```

Output :-



```
0 4
0 3
0 2
0 1
1 4
1 3
1 2
1 1
2 4
2 3
2 2
2 1
```

# Infinite For loop :- Loop executing forever is known as Infinite for loop.

In infinite for loop, the conditional statement is absent like:-

```go
for i:=0; ; i++{

    }
```

OR

```
for{

    }
```

OR

```
for true {

    }
```

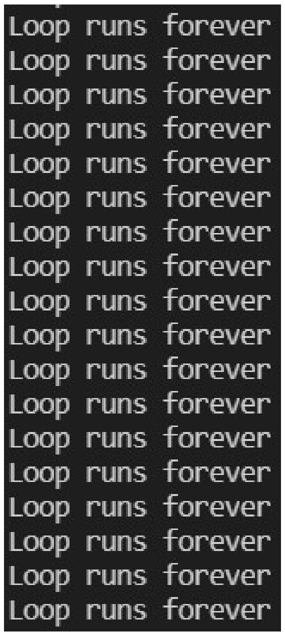
Example:-

```
package main

import "fmt"

func main() {
    for true {
        fmt.Println("Loop runs forever")
    }
}
```

Output:-



```
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
Loop runs forever
```

- **Condition-controlled iteration**

The for loop which has no header is used for condition-controlled iteration. It is similar to while-loop in other languages.

Syntax:-

```
for condition{

    }
```

Example :-

```
package main

import "fmt"

func main() {
    var sum = 1
    for sum < 100 {
        sum += sum
        fmt.Println(sum)
    }
}
```

Output :-

```
 2
 4
 8
 16
 32
 64
 128
```