Loopings

Loopings (first way of looping)

Looping atau perulangan merupakan suatu proses berulang yang dimana proses tersebut akan berhenti jika memenuhi suatu kondisi. Bahasa Go hanya memiliki satu looping yaitu looping dengan menggunakan keyword *for* atau yang kita kenal dengan sebutan *for loop*.

Contohnya seperti pada gambar dibawah ini.

```go
package main

import "fmt"

func main() {
  for i := 0; i < 3; i++ {
    fmt.Println("Angka", i)
  }
}
```

```
Angka 0
Angka 1
Angka 2
```

Loopings (first way of looping)

Jika kita lihat pada gambar pertama dibawah, ini merupakan cara pertama agar kita dapat melakukan looping pada bahasa Go. Looping tersebut akan bekerja selama variable *i* memiliki nilai kurang dari angka 3.

Jika kita jalankan maka hasilnya akan seperti pada gambar kedua dibawah. Perlu diingat disini bahwa kita tidak boleh lupa untuk menambahkan variable *i* seperti pada gambar pertama yang dimana i++ sama saja dengan i = i + 1. Jika variable *i* tidak ditambahkan maka akan menimbulkan i*nfinite loop* atau proses perulangan yang tidak akan berhenti.

```go
package main

import "fmt"

func main() {
  for i := 0; i < 3; i++ {
    fmt.Println("Angka", i)
  }
}
```

```
Angka 0
Angka 1
Angka 2
```

Loopings (second way of looping)

Cara kedua dalam melakukan looping pada bahasa Go adalah dengan menyelipkan kondisional seperti pada looping dengan menggunakan keyword *while*.

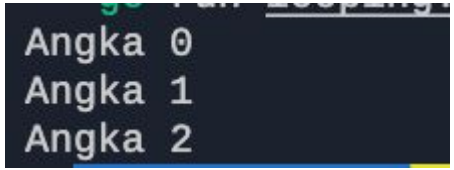
Contohnya seperti pada gambar pertama dibawah. Looping pada gambar pertama dibawah menggunakan kondisi yang dimana jika nilai yang dimiliki oleh variable *i* masih kurang dari angka 3, maka proses looping tersebut akan terus berlanjut. Namun jika sudah melebihi dari angka 3, maka proses looping tersebut akan berhenti. Jika kita menjalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua dibawah.

```go
package main

import "fmt"

func main() {
  var i = 0

  for i < 3 {
    fmt.Println("Angka", i)
    i++
  }
}
```



Loopings (third way of looping)

Cara ketiga dalam melakukan looping pada bahasa Go adalah dengan melakukan looping tanpa memberikan kondisi apapun. Contohnya seperti pada gambar pertama dibawah. Looping pada gambar dibawah memakai bantuan keyword *break* yang dimana dengan menggunakan keyword ini, maka dapat menghentikan suatu proses looping.

```go
package main

import "fmt"

func main() {
  var i = 0

  for {
    fmt.Println("Angka", i)

    i++
    if i == 3 {
      break
    }
  }
}
```

```
Angka 0
Angka 1
Angka 2
```

Loopings (third way of looping)

Pada looping tersebut, terdapat suatu kondisi yang dimana jika nilai pada variable *i* sudah memiliki nilai dengan angka sama dengan 3, maka looping keyword *break* akan terpanggil dan proses looping akan berhenti. Jika kita jalankan pada terminal, maka hasilnya akan seperti pada gambar kedua dibawah.

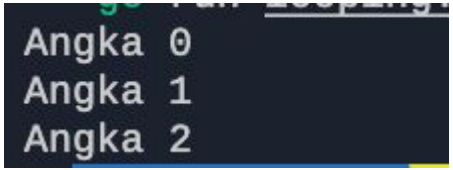
```go
package main

import "fmt"

func main() {
  var i = 0

  for {
    fmt.Println("Angka", i)

    i++
    if i == 3 {
      break
    }
  }
}
```



Loopings (break and continue keywords)

Sebelumnya kita telah menerapkan keyword *break* untuk menghentikan sebuah proses looping, dan sekarang kita juga akan memakai keyword *continue* yang digunakan untuk melanjutkan suatu proses looping. Contohnya seperti pada gambar pertama dibawah.

Pada looping di bawah, terdapat dua kondisional yang dimana pada kondisional pertama digunakan untuk memeriksa jika variable *i* memiliki nilai ganjil, maka proses loopingnya dipaksa berlanjut dan akan mengacuhkan syntax yang ada dibawah keyword continue (pada kasus kita sekarang berarti kondisional kedua dan fungsi *fmt.Println* pada looping dibawah).

```go
package main

import "fmt"

func main() {
  for i := 1; i <= 10; i++ {
    if i%2 == 1 {
      continue
    }

    if i > 8 {
      break
    }

    fmt.Println("Angka", i)
  }
}
```

```
Angka 2
Angka 4
Angka 6
Angka 8
```

Loopings (break and continue keywords)

Lalu pada kondisional kedua digunakan untuk memeriksa jika variable *i* sudah memiliki nilai diatas angka 8, maka keyword *break* akan terpanggil dan proses looping akan berhenti. Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua di bawah.

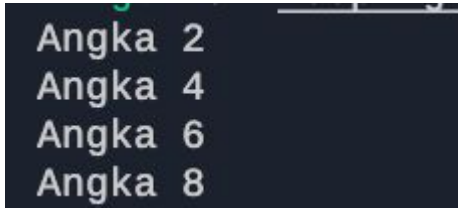
```go
package main

import "fmt"

func main() {
  for i := 1; i <= 10; i++ {
    if i%2 == 1 {
      continue
    }

    if i > 8 {
      break
    }

    fmt.Println("Angka", i)
  }
}
```



Loopings (Nested Looping)

Nested looping atau looping bersarang adalah suatu proses looping yang memiliki suatu proses looping di dalamnya. Contohnya seperti pada gambar pertama dibawah. Jika kita perhatikan, kita menggunakan fungsi *fmt.Println* dan tidak memberikan argumen apapun kedalamnya. Ini bisa dilakukan ketika kita hanya ingin membuat baris baru. Hasilnya sama saja ketika kita memakai fungsi fmt.Print dengan argumen “\n” atau sama dengan (fmt.Print("\n")). Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua dibawah.

```go
package main

import "fmt"

func main() {
  for i := 0; i < 5; i++ {
    for j := i; j < 5; j++ {
      fmt.Print(j, " ")
    }

    fmt.Println()
  }
}
```

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

Loopings (Label)

Pada looping bersarang, keyword break dan continue akan berlaku pada blok perulangan dimana ia digunakan saja.Ada cara agar kedua keyword ini bisa tertuju pada perulangan terluar atau perulangan tertentu, yaitu dengan memanfaatkan teknik pemberian label. Contohnya seperti pada gambar pertama dibawah.

```go
package main

import "fmt"

func main() {
outerLoop:
  for i := 0; i < 3; i++ {
    fmt.Println("Perulangan ke - ", i+1)
    for j := 0; j < 3; j++ {
      if i == 2 {
        break outerLoop
      }
      fmt.Print(j, " ")
    }
    fmt.Print("\n")
  }
}
```

```
Perulangan ke -  1
0 1 2
Perulangan ke -  2
0 1 2
Perulangan ke -  3
```

Loopings (Label)

Jika kita perhatikan hasil dari looping pertama pada gambar kedua, seluruh looping pada perulangan ketiga terhenti karena terdapat sebuah kondisional pada proses looping kedua yang dimana jika variable *i* sudah memiliki nilai dengan angka sama dengan 2, maka looping pertama atau looping terluar akan dihentikan atau sama saja dengan seluruh proses looping terhenti.

```go
package main

import "fmt"

func main() {
outerLoop:
  for i := 0; i < 3; i++ {
    fmt.Println("Perulangan ke - ", i+1)
    for j := 0; j < 3; j++ {
      if i == 2 {
        break outerLoop
      }
      fmt.Print(j, " ")
    }
    fmt.Print("\n")
  }
}
```

```
Perulangan ke -  1
0 1 2
Perulangan ke -  2
0 1 2
Perulangan ke -  3
```