Variables

Naming Conventions Variable

Dalam membuat variable di golang perlu gunakan penulisan yang tepat untuk memudahkan tiap-tiap komponen penulisan yang nantinya memiliki bagian atau fungsinya yang berbeda-beda. Berikut adalah tipe penulisan yang paling efesien dan banyak digunakan dalam pemrograman menggunakan bahasa GO..

camelCase — camelCase adalah konvensi penamaan di mana huruf pertama dari setiap kata dalam kata majemuk akan dikapitalisasi KECUALI untuk huruf pertama di awal kata

Contoh :

```
var (
        favoriteColor   = “green”
        favoriteShape = “rectangle”
)
```

Perhatikan huruf “f” diawal ditulis dengan huruf kecil, sedangkan untuk huruf “C” dan “S” ditulis huruf besar

Kapitalisasi akronim — misal : HTTP, DNS, URL

Variable with data type

Pada bahasa Golang sendiri, terdapat 2 cara penulisan untuk menulis variable. Yang pertama adalah yang dituliskan tipe datanya dan ada juga yang tidak ditulis tipe datanya atau biasa.

Declaring Variable

-Variable with data type

Sekarang mari coba untuk membuat variable yang melibatkan tipe data. Jika kita lihat pada gambar dibawah ini, kita telah membuat suatu 2 variable dengan tipe data yang berbeda. Variable pertama menggunakan tipe data *string* sedangkan variable kedua memakai tipe data *int.* Kedua contoh dibawah ini merupakan contoh dari membuat variable dengan tipe data.

```go
package main

import "fmt"

func main() {
  var name string = "Airell"
  var age int = 23

  fmt.Println("Ini adalah namanya ==>", name)
  fmt.Println("Ini adalah umurnya nya ==>", age)
}
```

Variable with data type

```go
package main

import "fmt"

func main() {
  var name string = "Airell"
  var age int = 23

  fmt.Println("Ini adalah namanya ==>", name)
  fmt.Println("Ini adalah umurnya nya ==>", age)
}
```

Jika gambar disebelah kanan kita jalankan pada terminal, maka hasilnya akan seperti pada gambar dibawah ini.

```
└─ go run variabel.go
Ini adalah namanya ==> Airell
Ini adalah umurnya nya ==> 23
```

Variable with data type

Kita juga bisa me-reassign suatu variable dengan suatu nilai asalkan kita memberikan nilai dengan tipe data yang sama dengan tipe data yang sudah dimiliki oleh variable tersebut. Contoh seperti pada gambar sebelah kanan.

Jika kita perhatikan pada gambar pertama, kita tidak memberikan nilai apapun pada variable *name* pada pertama kali kita membuat variable tersebut. Hal ini dapat dilakukan asalkan kita sudah memberikan tipe datanya. Lalu kita me-reassign variable *name* dan *age* dengan suatu nilai baru dan tidak ada error yang terjadi. Hal ini dikarenakan kita masih memberikan nilai dengan tipe data yang sama seperti pertama kali variable *name* dan *age* dibuat.

```go
package main

import "fmt"

func main() {
  var name string
  var age int = 23

  name = "Airell"
  age = 25

  fmt.Println("Ini adalah namanya ==>", name)
  fmt.Println("Ini adalah umurnya nya ==>", age)
}
```

Variable with data type

Namun jika kita tidak memberikan nilai dengan tipe data yang sama pada variable *name* dan *age,* maka akan terjadi error seperti pada gambar disebelah kanan yang dimana variable *name* kita re-assign dengan sebuah nilai dengan tipe *int* sedangkan variable *age* kita re-assign dengan sebuah nilai dengan tipe data string.

```go
package main

import "fmt"

func main() {
  var name string
  var age int = 23

  name = 30
  age = "Airell"

  fmt.Println("Ini adalah namanya ==>", name)
  fmt.Println("Ini adalah umurnya nya ==>", age)
}
```

Variable without data type

-Variable without data type/type inference

Kita juga dapat membuat suatu variable dengan tidak memberikan tipe datanya secara eksplisit. Hal ini dikenal dengan nama *type inference* yang dimana sistem dari golang yang akan menentukan sendiri tipe datanya secara otomatis tergantung dari nilai yang kita berikan pada variable tersebut. Contohnya seperti pada gambar disebelah kanan.

```go
package main

import "fmt"

func main() {
  var name = "Airell"
  var age = 23

  fmt.Printf("%T, %T", name, age)
}
```

Jika kita perhatikan gambar pertama di sebelah kanan, kita sekarang menggunakan fungsi *fmt.Printf*. Fungsi *fmt.Printf* mempunyai kegunaan yang sama dengan fungsi *fmt.Println*. Hanya saja struktur output dari fungsi *fmt.Printf* ditentukan dari *flag* yang kita berikan. Hal ini akan kita bahas nanti.

```
└─ go run variabel.go
string, int%
```

Jika kita coba menjalankan gambar pertama, maka kita akan mendapatkan keterangan tipe data kedua variable yang kita buat. Variable *name* akan secara otomatis memiliki tipe data *string* dan variable *age* akan mempunyai tipe data *int*.

Variable without data type

-Variable without data type/type inference

Kita juga dapat membuat suatu variable dengan tidak memberikan tipe datanya secara eksplisit. Hal ini dikenal dengan nama *type inference* yang dimana sistem dari golang yang akan menentukan sendiri tipe datanya secara otomatis tergantung dari nilai yang kita berikan pada variable tersebut. Contohnya seperti pada gambar disebelah kanan.

```go
package main

import "fmt"

func main() {
  var name = "Airell"
  var age = 23

  fmt.Printf("%T, %T", name, age)
}
```

Jika kita perhatikan gambar disebelah kanan, kita sekarang menggunakan fungsi fmt.Printf. Fungsi fmt.Printf mempunyai kegunaan yang sama dengan fungsi fmt.Println. Hanya saja struktur output dari fungsi fmt.Printf ditentukan dari *flag* yang kita berikan. Hal ini akan kita bahas nanti.

```
└─ go run variabel.go
string, int%
```

Jika kita coba menjalankan gambar pertama, maka kita akan mendapatkan tipe data kedua variable yang kita buat. Variable *name* akan secara otomatis memiliki tipe data *string* dan variable *age* akan mempunyai tipe data *int*.

Variable without data type(Short Declaration)

Kita juga dapat membuat variable dengan menerapkan teknik **short declaration** yang dimana kita tidak perlu menggunakan keyword *var* lagi. Kita bisa dengan mudah nya membuat suatu variable tanpa tipe data seperti pada gambar di sebelah kanan.

Jika kita perhatikan, kita menggunakan tanda titik dua (:) sebelum tanda sama dengan (=). Dengan seperti ini maka kita telah menerapkan teknik **short declaration** yang dimana teknik ini cukup mempermudah kita dalam membuat suatu variable. Tapi perlu diingat bahwa, ketika kita memakai teknik **short declaration**, maka kita harus langsung memberikan nilai kepada variable tersebut. Jika tidak maka akan terjadi error.

```go
package main

import "fmt"

func main() {
  name := "Airell"
  age := 23

  fmt.Printf("%T, %T", name, age)
}
```

Variable - Sesi 1

Multiple variable declarations

Pada bahasa Golang, kita dapat membuat lebih dari satu variable pada satu baris/line yang sama. Cara nya seperti gambar pertama. Jika kita perhatikan, kita telah membuat 3 variable pada baris pertama dan membuat 3 variable pada baris kedua. Pada baris pertama, ketiga variable tersebut memiliki tipe data string dan kita langsung memberikan nilai kepadanya. Sedangkan pada baris kedua, ketiga variable tersebut memiliki tipe data *int* dan kita memberikan nilai kepadanya dengan cara me-reassign nilai-nilai kepada ketiga variable pada baris kedua tersebut.

Dan ketika kita jalankan pada terminal kita, maka hasil nya akan seperti pada gambar kedua.

```go
package main

import "fmt"

func main() {

  var student1, student2, student3 string = "satu", "dua", "tiga"

  var first, second, third int

  first, second, third = 1, 2, 3

  fmt.Println(student1, student2, student3)

  fmt.Println(first, second, third)

}
```

```
satu dua tiga
1 2 3
```

Multiple variable declarations

Jika kita ingin memberikan nilai dengan tipe data yang berbeda-beda, kita dapat menerapkan *type inference* dengan keyword *var* maupun dengan *short declaration*. Contohnya seperti pada gambar pertama di sebelah kanan.

Jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua.

```go
package main

import "fmt"

func main() {

  var name, age, address = "Airell", 23, "Jalan sudirman"

  first, second, third := "1", 2, "3"

  fmt.Println(name, age, address)

  fmt.Println(first, second, third)

}
```

```
satu dua tiga
1 2 3
```

Variable - Sesi 1

Underscore variable

Bahasa Golang memiliki satu aturan yang cukup strict/ketat, yang dimana tidak ada variable yang boleh menganggur ketika sudah kita buat. Contohnya pada gambar pertama di sebelah kanan.

Jika kita perhatikan pada gambar pertama, terjadi error pada saat compile time yang ditandai dengan garis merah. Ini terjadi karena kita telah membuat dan mendeklarasikan variable tetapi variable-variable tersebut kita biarkan menganggur atau tidak dipakai. Maka dari itu cara menanggulanginya adalah dengan memakai variable *underscore* seperti pada gambar kedua.

Dengan variable *underscore* maka kita dapat menghindari error yang akan terjadi jika kita mempunyai suatu variable yang menganggur atau tidak dipakai.

```go
package main

func main() {

  var firstVariable string

  var name, age, address = "Airell", 23, "Jalan sudirman"

}
```

```go
package main

func main() {

  var firstVariable string

  var name, age, address = "Airell", 23, "Jalan sudirman"

  _, _, _, _ = firstVariable, name, age, address

}
```

Variable - Sesi 1

fmt.Printf Usage

Sebelumnya kita telah menggunakan fungsi fmt.Printf untuk mengetahui tipe data suatu variable dan kali ini akan kita bahas sedikit mengenai fungsi fmt.Printf tersebut. Struktur output yang dihasilkan oleh fungsi fmt.Printf akan bergantung dari flag yang kita pakai. Seperti contohnya jika kita ingin mengetahui tipe data dari suatu variable, maka kita dapat menggunakan *verb*% T. Contohnya seperti pada gambar pertama di sebelah kanan. Jika kita jalankan maka hasilnya akan seperti pada gambar kedua.

Jika kita perhatikan pada gambar pertama, kita juga memakai simbol \n yang dimana simbol ini dapat kita pakai untuk membuat baris baru. Karena pada dasarnya, fmt.Printf tidak akan membuat baris baru tidak seperti fmt.Println yang menambah baris baru secara otomatis.

```go
package main

import "fmt"

func main() {

  first, second := 1, "2"

  fmt.Printf("Tipe data variable first adalah %T \n", first)
  fmt.Printf("Tipe data variable second adalah %T \n", second)

}
```

```
Tipe data variable first adalah int
Tipe data variable second adalah string
```

HACKTIV8

Variable - Sesi 1

fmt.Printf Usage

Kita juga dapat memakai lebih dari satu *verb,* misalkan kita ingin memasukkan berbagai macam variable dengan tipe data berbeda-beda ke dalam satu kalimat. Maka kita bisa melakukannya seperti pada gambar pertama disebelah kanan. *Verb* %s digunakan untuk memformat suatu nilai dengan tipe data string sedangkan *verb* %d digunakan untuk memformat tipe data int dengan base 10 atau angka 0 sampai 9. Jika kita jalankan maka akan terlihat seperti pada gambar dibawah ini. Jika teman-teman ingin mengetahui tentang *verb* yang lainnya, maka bisa mengunjungi link berikut https://pkg.go.dev/fmt.

```go
package main

import "fmt"

func main() {

  var nama = "Airell"

  var age = 23

  var address = "Jalan sudirman"

  fmt.Printf("Halo nama ku %s, umuru aku adalah %d, dan aku tinggal di %s", nama, age, address)

}
```

```
└─ go run variabel.go
Halo nama ku Airell, umuru aku adalah 23, dan aku tinggal di Jalan sudirman
```

HACKTIV8