Data Types

Data Types - Sesi 1

Number (integer)

Tipe data pada bahasa Go terbagi menjadi 4 kategori dengan detail seperti berikut ini:

1. Basic Type: number, string, boolean.
2. Aggregate Type: array dan struct.
3. Reference Type: slice, pointer, map, function, channel
4. Interface Type: interface

Dan pada kali ini, kita akan membahas tentang tipe data pada kategori Basic Type.

Data Types - Sesi 1

Number (integer)

Tipe data number pada bahasa Go terbagi menjadi 2 sub-kategori yaitu integer(non desimal/non floating-point) dan float (desimal/floating point).

-integer

Integer merupakan tipe data numerik non desimal pada bahasa Go. Secara umum integer dibagi menjadi sub-kategori yaitu *uint* dan *int*.

- int: Bilangan cacah (Bilangan positif)
- uint: Bilangan bulat (bilangan positif maupun negatif)

Kemudian int dan uint terbagi menjadi beberapa jenis kembali. Bisa dilihat keterangannya gambar disebelah kanan. Dan jika kita menggunakan *int* pada suatu variable, maka *int* dapat merepresentasikan *int32* atau *int64* tergantung dari besaran nilainya.

Sr.No.	Types and Description
1	**uint8** Unsigned 8-bit integers (0 to 255)
2	**uint16** Unsigned 16-bit integers (0 to 65535)
3	**uint32** Unsigned 32-bit integers (0 to 4294967295)
4	**uint64** Unsigned 64-bit integers (0 to 18446744073709551615)
5	**int8** Signed 8-bit integers (-128 to 127)
6	**int16** Signed 16-bit integers (-32768 to 32767)
7	**int32** Signed 32-bit integers (-2147483648 to 2147483647)
8	**int64** Signed 64-bit integers (-9223372036854775808 to 9223372036854775807)

https://www.tutorialspoint.com/go/go_data_types.htm

Number (integer)

Setelah penjelasan yang kita baca pada halaman sebelumnya,tipe data *int* merupakan representasi atau sama saja dengan *int32* atau *int64* tergantung dari nilainya.

Mari kita perhatikan pada gambar pertama disebelah kanan. Kita membuat 2 variable yang dimana variable *first* mempunyai nilai dengan angka positif dan variable *second* mempunyai nilai dengan angka negatif. Jika kita jalankan pada terminal kita maka kita bisa melihat bahwa tipe data kedua variable tersebut ada *int* seperti pada gambar kedua.

Ada baiknya jika kita menentukan tipe data *integer* dengan jenis apa yang ingin kita pakai untuk menghindari boros memori. Seperti pada variable *first* kita bisa memberikan tipe data *uint8* dan pada variable *second* kita bisa memberikan tipe data *int8*. Hal ini kita lakukan karena *uint8* merepresentasikan angka dengan skala 0 - 255 sedangkan *uint8* merepresentasikan angka dengan skala -128 - 127.

```go
package main

import "fmt"

func main() {
  var first = 89
  var second = -12

  fmt.Printf("tipe data first : %T\n", first)
  fmt.Printf("bilangan second: %T\n", second)
}
```

```
tipe data first : int
bilangan second: int
```

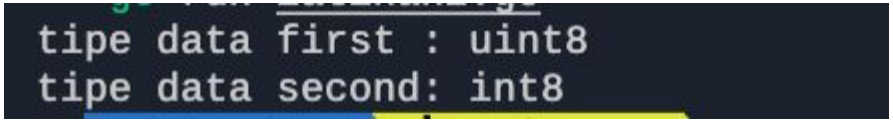
Number (integer)

Jika kita jalankan gambar pertama di bawah, maka hasilnya akan seperti pada gambar kedua

```go
func main() {

  var first uint8 = 89
  var second int8 = -12

  fmt.Printf("tipe data first : %T\n", first)
  fmt.Printf("tipe data second: %T\n", second)
}
```



Number (float)

-float

Float merupakan tipe data numerik desimal pada bahasa Go. Secara umum float dibagi menjadi 2 jenis yaitu *float32* dan *float34*.

Perbedaan kedua jenis float tersebut berada di lebar cakupan nilai desimal yang bisa ditampung. Untuk lebih jelasnya bisa merujuk ke spesifikasi IEEE-754 32-bit floating-point numbers

Data Type	Description
float32	32-bit IEEE 754 floating-point number
float64	64-bit IEEE 754 floating-point number

https://www.geeksforgeeks.org/data-types-in-go/

Number (float)

Perhatikan pada gambar pertama di bawah, kita membuat satu buah variable dengan tipe data *float32* yang mengandung nilai dengan angka *3.63*.

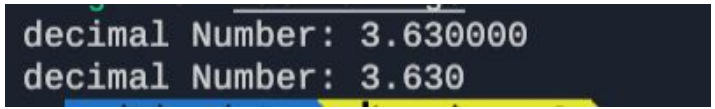
Lalu menggunakan verb %f pada fungsi *fmt.Printf* untuk memformat nilai float tersebut.

```go
package main

import "fmt"

func main() {
  var decimalNumber float32 = 3.63

  fmt.Printf("decimal Number: %f\n", decimalNumber)
  fmt.Printf("decimal Number: %.3f\n", decimalNumber)
}
```



Number (float)

Jika kita lihat pada gambar kedua, format angka desimal yang dicetak di terminal menghasil format angka yang berbeda walaupun masih berasal dari satu variable yang sama.

Pada dasarnya verb %f digunakan untuk memformat angka desimal atau tipe data float menjadi 6 digit angka desimal.

Jika kita ingin mengecilkan digit desimalnya, maka kita dapat menggunakan format verb seperti (%.nf), yang dimana huruf n dapat kita ganti dengan jumlah digit yang ingin kita hasilnya. Seperti contohnya pada fungsi *fmt.Printf* kedua.

```go
package main

import "fmt"

func main() {
  var decimalNumber float32 = 3.63

  fmt.Printf("decimal Number: %f\n", decimalNumber)
  fmt.Printf("decimal Number: %.3f\n", decimalNumber)
}
```

```
decimal Number: 3.630000
decimal Number: 3.630
```

Bool

Tipe data *bool* pada bahasa Go hanya terdiri dari 2 nilai yaitu *false*, dan *true*.

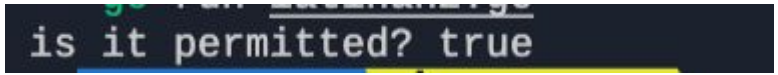
Pada umumnya tipe data ini digunakan dalam perkondisian atau perulangan.

Jika kita melihat pada gambar pertama dibawah, terdapat sebuah variable yang memiliki tipe data *bool* dengan nilai *true*. Lalu kita mencoba untuk mencetak nilai dengan tipe data *bool* tersebut menggunakan fungsi *fmt.Printf* dengan verb %t. Verb %t digunakan untuk memformat tipe data bool menjadi tipe data *string*. Dan jika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua dibawah.

```go
package main

import "fmt"

func main() {
  var condition bool = true
  fmt.Printf("is it permitted? %t \n", condition)
}
```



String

Ciri khas dari tipe data *string* adalah nilainya di apit oleh tanda quote atau petik dua (“”). Contoh penerapannya:

```go
package main

import "fmt"

func main() {
  var message string = "Halo"
  fmt.Print(message)
}
```

String

Selain menggunakan tanda quote, deklarasi string juga bisa dengan tanda grave accent/backticks (``), tanda ini terletak di sebelah kiri tombol 1.

Keistimewaan string yang dideklarasikan menggunakan backtics adalah membuat semua karakter didalamnya tidak di escape, termasuk \n, tanda petik dua dan tanda petik satu, baris baru, dan lainnya. Semua akan terdeteksi sebagai string.

```
package main

import "fmt"

func main() {
  var message = `Selamat datang di "Hacktiv8".
Salam kenal :).
Mari belajar "Scalable Web Service With Go".`

  fmt.Println(message)
}
```

nil & Zero Value

nil bukan merupakan tipe data, melainkan sebuah nilai. Variabel yang isi nilainya nil berarti memiliki nilai kosong.

Semua tipe data yang sudah dibahas sebelumnya memiliki zero value (nilai default tipe data). Artinya meskipun variabel dideklarasikan dengan tanpa nilai awal, tetap akan ada nilai default-nya.

- Zero value dari *string* adalah "" (string kosong).
- Zero value dari *bool* adalah false.
- Zero value dari tipe *numerik non-desimal* adalah 0.
- Zero value dari tipe *numerik desimal* adalah 0.0.

Zero value berbeda dengan *nil*. *Nil* adalah nilai kosong, benar-benar kosong. *Nil* tidak bisa digunakan pada tipe data yang sudah dibahas sebelumnya. Ada beberapa tipe data yang bisa di-set nilainya dengan *nil*, diantaranya:

- pointer
- tipe data function
- slice
- map
- channel
- interface kosong atau interface{}