Slice

Slice

Slice merupakan suatu tipe data yang mirip dengan tipe data *array*, yang juga memiliki kegunaan untuk menyimpan satu atau lebih data. Namun tipe data *slice* dan *array* memiliki sifat yang berbeda. *Slice* tidak memiliki sifat *fixed-length* yang berarti panjang dari slice tidak tetap sehingga kita bisa dengan leluasa menentukan panjang dari *slice* nya. Slice termasuk dalam kategori *reference type* yang dimana jika kita melakukan copy terhadap suatu *slice*, dan kita ubah element dari yang kita copy tersebut, maka *slice* semulanya juga akan ikut terubah.

Cara membuat slice cukup mudah hampir mirip dengan jika kita membuat suatu *array.* Yang menjadi perbedaan adalah kita tidak perlu menuliskan panjang dari *slice* nya tidak seperti *array*. Contohnya seperti pada gambar dibawah ini.

```go
package main

func main() {

  var fruits = []string{"apple", "banana", "mango"}

  _ = fruits

}
```

Slice (make function)

Kita juga bisa membuat sebuah *slice* dengan menggunakan fungsi *make*. Argumen pertama yang diberikan pada gambar pertama di sebelah kanan adalah tipe dari *slice* nya, dan argumen keduanya adalah panjang dari *slice* nya.

Jika kita jalankan maka hasilnya akan seperti pada gambar kedua di sebelah kanan. Bisa dilihat pada gambar kedua, *slice* pada variable *fruits* belum berisi nilai apapun dan variable *fruits* memiliki tipe data slice of string atau slice dengan tipe data string.

Maka dari ikut ketika di print ke terminal, maka hasilnya adalah tiga string kosong.

```go
package main

import "fmt"

func main() {

  var fruits = make([]string, 3)

  _ = fruits

  fmt.Printf("%#v", fruits)

}
```

```
└─ go run array.go
[]string{"", "", ""}
```

Slice (append function)

Jika kita ingin menambahkan element pada *slice* dari variable *fruits* pada halaman sebelumnya, kita bisa melakukannya dengan cara mengakses indexnya seperti pada gambar pertama di sebelah kanan.

Jika ingin lebih mudah, maka kita dapat memanfaatkan fungsi *append*. Fungsi *append* akan mengembalikan nilai dari *slice* yang ditambahkannya, maka dari itu kita harus menyimpan fungsi *append* ke dalam suatu variable.

Karena kita ingin agar variable *fruits* yang bertambah elementnya, maka dari itu kita me-reassign variable *fruits* dengan fungsi *append*. Parameter pertama yang diberikan pada fungsi *append* adalah *slice* yang ingin ditambahkan, lalu parameter setelahnya adalah element-element yang ingin ditambahkan dan jangan lupa dipisahkan dengan koma.

```go
package main

import "fmt"

func main() {

  var fruits = make([]string, 3)
  _ = fruits

  fruits[0] = "apple"
  fruits[1] = "banana"
  fruits[3] = "mango"

  fmt.Printf("%#v", fruits)
}
```

```go
package main

import "fmt"

func main() {

  var fruits = make([]string, 3)

  fruits = append(fruits, "apple", "banana", "mango")

  fmt.Printf("%#v", fruits)
}
```

Slice (append function with ellipsis)

Jika kita ingin memasukkan seluruh element-element pada suatu array ke dalam array lainnya, maka kita dapat menggunakan tanda *ellipsis* (...) atau tanda titik tiga berurut. Contohnya seperti pada gambar pertama di sebelah kanan.

Terdapat 2 buah variable bernama *fruits1* dan *fruits2* yang masing-masing memiliki tipe data slice of string dan menyimpan nama-nama buah.

Lalu variable *fruits1* mencoba untuk menambahkan seluruh element yang terdapat pada variable *fruits2,* dan memakai tanda *ellipsis* untuk mengambil seluruh elementnya.

Jika dijalankan pada terminal, maka hasilnya akan seperti pada gambar kedua disebelah kanan.

```go
package main

import "fmt"

func main() {

  var fruits1 = []string{"apple", "banana", "mango"}

  var fruits2 = []string{"durian", "pineapple", "starfruit"}

  fruits1 = append(fruits1, fruits2...)

  fmt.Printf("%#v", fruits1)
}
```

```
[]string{"apple", "banana", "mango", "durian", "pineapple", "starfruit"}
```

Slice - Sesi 2

Slice (copy function)

Kita juga bisa menggunakan fungsi *copy* untuk meng-copy seluruh element pada sebuah slice ke dalam *slice* lainnya. Perlu diingat disini bahwa ketika kita mencoba untuk meng-copy sebuah *slice* kedalam *slice* lainnya, maka seluruh element pada *slice* lainnya tersebut akan ter-replace oleh element-element yang di copy kannya.

Contohnya seperti pada gambar pertama di sebelah kanan.

```go
package main

import "fmt"

func main() {

  var fruits1 = []string{"apple", "banana", "mango"}

  var fruits2 = []string{"durian", "pineapple", "starfruit"}

  nn := copy(fruits1, fruits2)

  fmt.Println("Fruits1 =>", fruits1)
  fmt.Println("Fruits2 =>", fruits2)
  fmt.Println("Copied elements =>", nn)
}
```

```
Fruits1 => [durian pineapple starfruit]
Fruits2 => [durian pineapple starfruit]
Copied elements => 3
```

Slice (copy function)

Pada kasus kita kali ini, variable *fruits1* ingin meng-copy seluruh element yang ada pada variable *fruits2*.

Argumen pertama yang diterima oleh fungsi *copy* adalah destinasi atau *slice* yang ingin meng-copy, lalu argument *kedua* adalah source/sumber dari *slice* yang ingin di copy.

Fungsi copy juga akan mengembalikan jumlah element yang berhasil ter-copy.

Ketika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua disebelah kanan. Element pada *fruits1* sudah ter-replace oleh *fruits2*, dan terdapat 3 element yang berhasil ter-copy.

```go
package main

import "fmt"

func main() {

  var fruits1 = []string{"apple", "banana", "mango"}

  var fruits2 = []string{"durian", "pineapple", "starfruit"}

  nn := copy(fruits1, fruits2)

  fmt.Println("Fruits1 =>", fruits1)
  fmt.Println("Fruits2 =>", fruits2)
  fmt.Println("Copied elements =>", nn)
}
```

```
Fruits1 => [durian pineapple starfruit]
Fruits2 => [durian pineapple starfruit]
Copied elements => 3
```

Slice (Slicing)

Ada cara lain lagi agar kita dapat mendapatkan element-element dari sebuah *slice* dan kita juga bisa menentukan element dari index ke berapa yang ingin kita dapatkan. Caranya adalah dengan menggunakan *slicing*.

Contohnya seperti pada gambar disebelah kanan. Cara penulisan dalam melakukan slicing adalah sama dengan [start:stop]. *Start* sama dengan awal index yang ingin kita akses dan *stop* berarti index akhirnya. Perhatikan hasil dari syntax pada gambar pertama di gambar kedua.

```go
package main

import "fmt"

func main() {

  var fruits1 = []string{"apple", "banana", "mango", "durian", "pineapple"}

  var fruits2 = fruits1[1:4]
  fmt.Printf("%#v\n", fruits2)

  var fruits3 = fruits1[0:]
  fmt.Printf("%#v\n", fruits3)

  var fruits4 = fruits1[:3]
  fmt.Printf("%#v\n", fruits4)

  var fruits5 = fruits1[:] // sama dengan fruits1[:len(fruits1)]
  fmt.Printf("%#v\n", fruits5)
}
```

```
[]string{"banana", "mango", "durian"}
[]string{"apple", "banana", "mango", "durian", "pineapple"}
[]string{"apple", "banana", "mango"}
[]string{"apple", "banana", "mango", "durian", "pineapple"}
```

Slice (Slicing)

-Variable *fruits2* ingin mendapatkan element dari *fruits1* dari index ke *satu* hingga index ke *tiga,* maka dari itu cara penulisannya adalah fruits[1:4].

-Variable *fruits3* ingin mendapatkan element dari *fruits1* dari index ke *nol* hingga index terakhirnya, maka dari itu cara penulisannya adalah fruits[0:] yang dimana keterangan *stop* boleh dihilangkan jika ingin mendapatkan hingga index terakhir.

- Variable *fruits4* ingin mendapatkan element dari *fruits1* dari index ke *nol* hingga index *kedua*, maka dari itu cara penulisannya adalah fruits[:3] yang dimana keterangan *start* nya boleh dihilangkan ketika ingin mendapatkan element dari index ke *nol.*

```go
package main

import "fmt"

func main() {

  var fruits1 = []string{"apple", "banana", "mango", "durian", "pineapple"}

  var fruits2 = fruits1[1:4]
  fmt.Printf("%#v\n", fruits2)

  var fruits3 = fruits1[0:]
  fmt.Printf("%#v\n", fruits3)

  var fruits4 = fruits1[:3]
  fmt.Printf("%#v\n", fruits4)

  var fruits5 = fruits1[:] // sama dengan fruits1[:len(fruits1)]
  fmt.Printf("%#v\n", fruits5)
}
```

```
[]string{"banana", "mango", "durian"}
[]string{"apple", "banana", "mango", "durian", "pineapple"}
[]string{"apple", "banana", "mango"}
[]string{"apple", "banana", "mango", "durian", "pineapple"}
```

Slice (Slicing)

- Variable *fruits5* ingin mendapatkan element dari *fruits1* dari index 0 hingga index terakhir, maka cara penulisannya adalah fruits[:] yang dimana jika ingin mendapatkan element seluruhnya, berarti tidak perlu memberi keterangan *start* dan *stop* nya, cukup memberikan tanda titik dua saja [:].

```go
package main

import "fmt"

func main() {

  var fruits1 = []string{"apple", "banana", "mango", "durian", "pineapple"}

  var fruits2 = fruits1[1:4]
  fmt.Printf("%#v\n", fruits2)

  var fruits3 = fruits1[0:]
  fmt.Printf("%#v\n", fruits3)

  var fruits4 = fruits1[:3]
  fmt.Printf("%#v\n", fruits4)

  var fruits5 = fruits1[:] // sama dengan fruits1[:len(fruits1)]
  fmt.Printf("%#v\n", fruits5)
}
```

```
[]string{"banana", "mango", "durian"}
[]string{"apple", "banana", "mango", "durian", "pineapple"}
[]string{"apple", "banana", "mango"}
[]string{"apple", "banana", "mango", "durian", "pineapple"}
```

Slice (Combining slicing and append)

Kita juga dapat mengkombinasikan fungsi *append* dengan *slicing*. Contohnya seperti pada gambar pertama di sebelah kanan. Jika kita perhatikan, variable *fruits1* ingin me-replace index *ketiga* hingga seterusnya dengan hanya buah "rambutan" saja. Jika kita jalankan pada terminal maka hasilnya akan seperti pada gambar kedua di sebelah kanan.

```go
package main

import "fmt"

func main() {

  var fruits1 = []string{"apple", "banana", "mango", "durian", "pineapple"}

  fruits1 = append(fruits1[:3], "rambutan")

  fmt.Printf("%#v\n", fruits1)

}
```

Slice (Backing array)

Setiap kita membuat suatu *slice* pada bahasa Go , secara otomatis Go akan membuat suatu array tersembunyi yang disebut dengan *Backing array*. *Backing array* akan bertugas untuk menyimpan element pada *slice,* bukan *slice* nya sendiri. Bahasa Go mengimplementasikan *slice* sebagai sebuah struktur data yang disebut dengan *slice header*. *Slice header* terdiri dari:

- Alamat memori/address dari *backing array*.
- Panjang dari *slice* yang bisa didapatkan dari fungsi *len.*
- Kapasitas dari *slice* yang bisa didapatkan dari fungsi *cap*.

Slice (Backing array)

Ketika kita mencoba untuk mendapatkan beberapa element dari sebuah *slice* yang sudah ada dengan cara melakukan *slicing,* maka *Go* tidak akan membuat suatu *backing array* baru melainkan *slice* tersebut akan berbagi *backing array* yang sama dengan *slice* yang sudah ada.

Contohnya seperti pada gambar pertama di sebelah kanan. Variable *fruits2* melakukan slicing terhadap *fruits1* untuk mendapatkan element dari index ke-2 sampai ke-3 dari *fruits1* atau yang berarti buah durian dan banana.

```go
package main

import "fmt"

func main() {

  var fruits1 = []string{"apple", "mango", "durian", "banana", "starfruit"}

  var fruits2 = fruits1[2:4]

  fruits2[0] = "rambutan"

  fmt.Println("fruits1 => ", fruits1)
  fmt.Println("fruits2 => ", fruits2)

}
```

```
fruits1 =>  [apple mango rambutan banana starfruit]
fruits2 =>  [rambutan banana]
```

Slice (Backing array)

Setelah itu *fruits2* mencoba untuk mengganti buah yang berada pada index ke-0 (buah **durian**) menjadi buah **rambutan**.

Kemudian ketika kita jalankan pada terminal maka kita dapat menyadari bahwa element *fruits1* pada index ke-2 yang ikut berganti menjadi buah **rambutan**.

Ini terjadi karena variable *fruits1* dan *fruits2* masih dalam satu **backing array** yang sama.

Hal ini yang menyebabkan penggunaan *slice* lebih hemat memori jika dibandingkan dengan tipe data *array*.

```go
package main

import "fmt"

func main() {

  var fruits1 = []string{"apple", "mango", "durian", "banana", "starfruit"}

  var fruits2 = fruits1[2:4]

  fruits2[0] = "rambutan"

  fmt.Println("fruits1 => ", fruits1)
  fmt.Println("fruits2 => ", fruits2)

}
```

```
fruits1 =>  [apple mango rambutan banana starfruit]
fruits2 =>  [rambutan banana]
```

Slice (Cap function)

Fungsi *cap* dapat kita gunakan untuk mengetahui kapasitas dari sebuah *array* maupun *slice*.

Ketika pertama kali kita membuat suatu *slice*, panjang dan kapasitasnya dipastikan sama, namun dapat berubah seiring dengan *slicing* yang kita lakukan.

Contohnya seperti pada gambar disebelah kanan. Variable *fruits1* telah disiapkan di awal dengan panjang 4 dan kapasitas 4.

Lalu terdapat 2 variable baru bernama *fruits2* dan *fruits3* yang melakukan *slicing* terhadap *fruits1* untuk mendapatkan 3 element dari *fruits1*.

Lalu kenapa *fruits2* memiliki panjang dan kapasitas yang berbeda sedangkan *fruits3* panjang dan kapasitasnya sama tetapi kapasitasnya berkurang? Mari kita bedah di halaman selanjutnya.

```go
package main

import (
  "fmt"
  "strings"
)

func main() {

  var fruits1 = []string{"apple", "mango", "durian", "banana"}

  fmt.Println("Fruits1 cap:", cap(fruits1)) //4
  fmt.Println("Fruits1 len:", len(fruits1)) //4

  fmt.Println(strings.Repeat("#", 20))

  var fruits2 = fruits1[0:3]

  fmt.Println("Fruits2 cap:", cap(fruits2)) //4
  fmt.Println("Fruits2 len:", len(fruits2)) //3

  fmt.Println(strings.Repeat("#", 20))

  var fruits3 = fruits1[1:]

  fmt.Println("Fruits3 cap:", cap(fruits3)) //3
  fmt.Println("Fruits3 len:", len(fruits3)) //3
}
```

```
Fruits1 cap: 4
Fruits1 len: 4
####################
Fruits2 cap: 4
Fruits2 len: 3
####################
Fruits3 cap: 3
Fruits3 len: 3
```

Slice (Cap function)

variable	slicing	result	len()	cap()
fruits1	-	[fruit, fruit, fruit, fruit]	4	4
fruits2	fruits[0:3]	[fruit, fruit, fruit, ...]	3	4
fruits3	fruits[1:]	[.... , fruit, fruit, fruit]	3	3

```
fruits[x:y]
```

Agar lebih mudah untuk dipahami, kita akan menggunakan analogi x dan y dalam melakukan *slicing*.

Ketika kita melakukan *slicing* yang dimulai dari index ke-0 hingga index ke-y, maka kita akan mendapatkan element dari index ke-0 hingga element sebelum index ke-y dengan panjang yang sesuai dengan element yang kita dapatkan dan kapasitas yang sama dengan kapasitas *slice* aslinya.

Slice (Cap function)

variable	slicing	result	len()	cap()
fruits1	-	[fruit, fruit, fruit, fruit]	4	4
fruits2	fruits[0:3]	[fruit, fruit, fruit, ...]	3	4
fruits3	fruits[1:]	[.... , fruit, fruit, fruit]	3	3

```
fruits[x:y]
```

Sedangkan *slicing* yang dimulai dari index ke-x, yang dimana nilai x adalah lebih dari 0, membuat elemen ke-x slice yang diambil menjadi elemen ke-0 slice baru.

Hal inilah yang membuat kapasitas slice berubah, namun panjang akan tetap sesuai dengan jumlah element yang di dapatkan.

Slice (Creating a new backing array)

Ketika kita ingin mendapatkan element-element dari *slice* yang sudah ada, namun kita juga ingin membuat *backing array* yang baru, maka kita dapat menggunakan fungsi *append* untuk melakukannya.

Contohnya seperti pada gambar pertama di sebelah kanan.

Variable *newCars* ingin mendapatkan element-element dimulai dari index ke-0 hingga index ke-1 dari variable *cars*.

Namun *newCars* mendapatkan element-elementnya dengan menggunakan fungsi *append* walaupun masih menggunakan *slicing* di dalam fungsi *append* nya.

Ketika index ke-0 dari *cars* dirubah, maka *newCars* tidak ikut terubah dikarenakan mereka tidak memiliki *backing array* yang sama.

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

```go
package main

import "fmt"

func main() {
  cars := []string{"Ford", "Honda", "Audi", "Range Rover"}
  newCars := []string{}

  newCars = append(newCars, cars[0:2]...)

  cars[0] = "Nissan"
  fmt.Println("cars:", cars)
  fmt.Println("newCars:", newCars)
}
```

```
cars: [Nissan Honda Audi Range Rover]
newCars: [Ford Honda]
```