Function

Function

Bahasa Go juga memiliki function atau fungsi nya sendiri. Cara menulis sebuah *function* pada Go cukup mudah. Caranya adalah dengan menggunakan keyword *func* lalu diikuti dengan nama *function* dan parameter yang digunakan. Contohnya seperti pada gambar pertama dibawah. Kita telah membuat *function* bernama *greet* yang menerima 2 parameter dengan tipe data *string* dan *int8*. Function *greet* merupakan sebuah function yang tidak mengembalikan/me-return nilai apapun, melainkan *function* ini hanya bertugas untuk menampilkan *text* pada terminal kita menggunakan *fmt.Printf*. Ketika kita jalankan pada terminal kita, maka hasilnya akan seperti pada gambar kedua dibawah.

Perlu diingat disini bahwa, kita harus menjelaskan jenis dari tipe data nya pada parameter *function* jika *function* yang kita gunakan menerima sebuah parameter.

```go
package main

import "fmt"

func main() {
  greet("Airell", 23)
}

func greet(name string, age int8) {
  fmt.Printf("Hello there! My name is %s and I'm %d years old", name, age)
}
```

```
Hello there! My name is Airell and I'm 23 years old
```

Function

Ketika *function* yang kita buat menerima lebih dari satu parameter namun semua tipe datanya sama, maka kita tidak perlu menuliskan seluruh tipe datanya secara satu per satu. Contohnya seperti pada gambar pertama di sebelah kanan. Function *greet* yang kita buat menerima 2 parameter yang sama-sama memiliki tipe data yang sama yaitu tipe data *string*. Maka dari itu kita tidak perlu menuliskan satu per satu tipe datanya.

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

```go
package main

import "fmt"

func main() {
  greet("Airell", "jalan sudirman")
}

func greet(name, address string) {
  fmt.Println("Hello there! My name is", name)
  fmt.Println("I live in", address)
}
```

```
Hello there! My name is Airell
I live in jalan sudirman
```

Function (Return)

Ketika *function* yang kita buat mengembalikan/me-return sebuah nilai, maka kita perlu menuliskan tipe data dari nilai yang di return. Contohnya seperti pada gambar di sebelah kanan.

Kita membuat function bernama *greet* yang me-return sebuah nilai dengan tipe data *string*. Function *greet* menerima 2 parameter dengan tipe data *string* dan *slice of string* atau *slice* dengan tipe data *string*.

Lalu jika kita perhatikan pada line 42, kita menggunakan bantuan dari package *strings* dengan menggunakan function Join. Function *Join* berguna untuk menggabungkan nilai-nilai dengan tipe data *string* yang terdapat pada sebuah *slice* ataupun *array*.

```
27  package main
28
29  import (
30    "fmt"
31    "strings"
32  )
33
34  func main() {
35    var names = []string{"Airell", "Jordan"}
36    var printMsg = greet("Heii", names)
37
38    fmt.Println(printMsg)
39  }
40
41  func greet(msg string, names []string) string {
42    var joinStr = strings.Join(names, " ")
43
44    var result string = fmt.Sprintf("%s %s", msg, joinStr)
45
46    return result
47  }
```

```
Heii Airell Jordan
```

Function (Return)

Kita juga memberikan separator atau pemisah berupa spasi " " yang kita berikan pada argumen kedua pada function *Join*. Lalu setelah itu pada line 44, kita menggunakan function *Sprintf* dari package *fmt*.

Function *Sprintf* memiliki fungsi yang sama seperti function *Printf*, namun bedanya adalah function *Sprintf* akan me-return sebuah nilai sedang function *Printf* tidak.

Kemudian nilai return dari function *greet* kita simpan dalam sebuah variable bernama *printMsg*.

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

```go
package main

import (
  "fmt"
  "strings"
)

func main() {
  var names = []string{"Airell", "Jordan"}
  var printMsg = greet("Heii", names)

  fmt.Println(printMsg)
}

func greet(msg string, names []string) string {
  var joinStr = strings.Join(names, " ")

  var result string = fmt.Sprintf("%s %s", msg, joinStr)

  return result
}
```

```
Heii Airell Jordan
```

Function (Returning multiple values)

Function pada bahasa Go dapat me-return lebih dari satu nilai. Contohnya seperti function bernama *calculate* pada gambar pertama di sebelah kanan. Function *calculate* tersebut berfungsi untuk menghitung luas dan keliling lingkaran. Lalu function *calculate* akan me-return 2 nilai yaitu hasil dari kalkulasi luas dan keliling berdasarkan diameter yang diberikan pada parameter function *calculate*.

Untuk menghitung luas nya, kita memakai bantuan dari package *math* dengan memanfaatkan konstanta *Pi*. Konstanta *math.Pi* adalah konstanta bawaan package math yang merepresentasikan Pi atau 22/7.

Kemudian untuk menghitung kelilingnya, kita memakai fungsi *math.Pow* yang digunakan untuk memangkat nilai. math.Pow(2, 3) berarti 2 pangkat 3, hasilnya 8. Fungsi *math.Pow* juga berasal dari package *math*.

```go
package main

import (
  "fmt"
  "math"
)

func main() {
  var diameter float64 = 15

  var area, circumference = calculate(diameter)

  fmt.Println("Area:", area)
  fmt.Println("Circumference:", circumference)
}

func calculate(d float64) (float64, float64) {
  //Menghitung luas
  var area float64 = math.Pi * math.Pow(d/2, 2)
  //Menghitung keliling

  var circumference = math.Pi * d

  return area, circumference
}
```

Function (Returning multiple values)

Karena function *calculate* akan me-return 2 nilai, maka dari itu kita perlu mendefinisikan tipe data apa saja yang akan di return dan membungkusnya dengan tanda kurung (float64, float64). Lalu kita perlu menampung function *calculate* kedalam 2 variable dikarenakan function *calculate* me-return 2 nilai. Pada gambar pertama di sebelah kanan, kita menampung nilai hasil dari function *calculate* kepada variable *area* dan *circumference*.

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

```go
package main

import (
  "fmt"
  "math"
)

func main() {
  var diameter float64 = 15

  var area, circumference = calculate(diameter)

  fmt.Println("Area:", area)
  fmt.Println("Circumference:", circumference)
}

func calculate(d float64) (float64, float64) {
  //Menghitung luas
  var area float64 = math.Pi * math.Pow(d/2, 2)
  //Menghitung keliling

  var circumference = math.Pi * d

  return area, circumference
}
```

```
Area: 176.71458676442586
Circumference: 47.12388980384689
```

Function (Predefined return value)

Kita juga dapat menggunakan sebuah variable untuk mendefinisikan nilai return dari sebuah function. Contoh pada function *calculate* pada gambar pertama di sebelah kanan. Function *calculate* tersebut masih memiliki fungsi yang sama seperti pada halaman sebelumnya, namun perbedaannya adalah kita menggunakan teknik *predefined return value* yang dimana kita dapat memberikan sebuah variable sebagai hasil nilai return dari sebuah function.

Jika kita perhatikan pada gambar pertama, variable *area* dan *circumference* yang berada di dalam function *calculate* bukanlah sebuah variable baru, melainkan kedua variable tersebut adalah variable yang digunakan sebagai nilai return.

Jika kita menggunakan teknik *predefined return value,* maka kita perlu me-reassign variable yang digunakan sebagai nilai return dengan sebuah nilai yang nantinya akan menghasilkan sebuah nilai return. Tapi perlu diingat bahwa kita masih tetap membutuhkan keyword *return* yang diletakkan di akhir baris *function*.

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

```go
package main

import (
  "fmt"
  "math"
)

func main() {
  var diameter float64 = 15

  var area, circumference = calculate(diameter)

  fmt.Println("Area:", area)
  fmt.Println("Circumference:", circumference)
}

func calculate(d float64) (area float64, circumference float64) {
  //Menghitung luas
  area = math.Pi * math.Pow(d/2, 2)
  //Menghitung keliling

  circumference = math.Pi * d

  return
}
```

```
Area: 176.71458676442586
Circumference: 47.12388980384689
```

Function (Variadic function #1)

Bahasa Go telah menyediakan sebuah *function* yang dimana function tersebut dapat menerima *argumen* yang tak terbatas jumlahnya. *Function* tersebut bernama function *variadic*. Contohnya seperti pada gambar disebelah kanan.

Function *print* merupakan sebuah function *variadic*. Cara mengetahuinya adalah dengan memperhatikan penulisan pada parameter yang diterimanya. Terdapat tanda titik sebanyak tiga kali sebelum keterangan tipe data parameter yang diterimanya ...string.

```go
package main

import "fmt"

func main() {
  studentLists := print("Airell", "Nanda", "Mailo", "Schannel", "Marco")

  fmt.Printf("%v", studentLists)
}

func print(names ...string) []map[string]string {
  var result []map[string]string

  for i, v := range names {
    key := fmt.Sprintf("student%d", i+1)
    temp := map[string]string{
      key: v,
    }
    result = append(result, temp)
  }

  return result
}
```

Function (Variadic function #1)

Dengan menambahkan tanda titik tiga tersebut pada sebuah parameter, maka sebuah dianggap sebagai sebauh function *variadic*, dan parameter yang diterminya akan di konversikan menjadi sebuah *slice*. Maka dari itu kita dapat melakukan looping terhadap parameter yang diterima oleh function *print* pada line 118 - 124.

Function *print* digunakan untuk menerima banyak list dari nama-nama murid dan setelah itu dikonversikan menjadi sebuah *slice* yang mengandung tipe data *map*. Pada line 119, kita menggunakan fungsi *fmt.Sprintf* untuk membuat penamaan *key* yang unik berdasarkan *index* looping nya dan index tersebut kita tambahkan dengan angka 1 agar tidak dimulai dari angka 0.

```
105 package main
106
107 import "fmt"
108
109 func main() {
110   studentLists := print("Airell", "Nanda", "Mailo", "Schannel", "Marco")
111
112   fmt.Printf("%v", studentLists)
113 }
114
115 func print(names ...string) []map[string]string {
116   var result []map[string]string
117
118   for i, v := range names {
119     key := fmt.Sprintf("student%d", i+1)
120     temp := map[string]string{
121       key: v,
122     }
123     result = append(result, temp)
124   }
125
126   return result
127 }
128
```

Function (Variadic function #1)

Ketika kita jalankan function *print* pada terminal kita, maka hasilnya akan seperti pada gambar kedua di bawah.

```go
package main

import "fmt"

func main() {
  studentLists := print("Airell", "Nanda", "Mailo", "Schannel", "Marco")

  fmt.Printf("%v", studentLists)
}

func print(names ...string) []map[string]string {
  var result []map[string]string

  for i, v := range names {
    key := fmt.Sprintf("student%d", i+1)
    temp := map[string]string{
      key: v,
    }
    result = append(result, temp)
  }

  return result
}
```

```
[map[student1:Airell] map[student2:Nanda] map[student3:Mailo] map[student4:Schannel] map[student5:Marco]]
```

Function (Variadic function #2)

Kita juga dapat menggunakan tipe data *slice* sebagai parameter dari sebuah function *variadic*. Contohnya seperti pada gambar pertama di sebelah kanan.

Jika kita ingin memakai tipe data *slice* sebagai argumen untuk sebuah function *variadic*, maka kita perlu menggunakan tanda ellipsis atau tanda titik tiga... .

Function *sum* berfungsi untuk menjumlah seluruh angka yang diberikan pada parameternya.

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

```go
package main      wijaysali, 5 months ago • Update 1st Versio

import "fmt"

func main() {
  numberLists := []int{1, 2, 3, 4, 5, 6, 7, 8}

  result := sum(numberLists...)

  fmt.Println("Result:", result)
}

func sum(numbers ...int) int {
  total := 0

  for _, v := range numbers {
    total += v
  }
  return total
}
```

Function (Variadic function #3)

Kita dapat menggabungkan parameter biasa dengan parameter variadic pada sebuah function *variadic*. Namun perlu diingat disini bahwa parameter *variadic* perlu diletakkan di posisi akhir parameter. Contohnya seperti pada gambar pertama di sebelah kanan.

```go
package main

import (
  "fmt"
  "strings"
)

func main() {
  profile("Airell", "pasta", "ayam geprek", "ikan roa", "sate padang")
}

func profile(name string, favFoods ...string) {
  mergeFavFoods := strings.Join(favFoods, ",")

  fmt.Println("Hello there!!! I'm", name)
  fmt.Println("I really love to eat", mergeFavFoods)
}
```

Function *profile* menerima sebuah parameter biasa bernama *name*, dan parameter *variadic* bernama *favFoods*. Jika kita perhatikan pada pemanggilan function *profile*, argumen dengan nilai "Airell" akan diterima oleh function *profile* sebagai parameter biasa, lalu argumen selanjutnya akan diterima oleh function *profile* sebagai parameter *variadic*.

```
Hello there!!! I'm Airell
I really love to eat pasta,ayam geprek,ikan roa,sate padang
```

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