**Variable**

**variable one line**

* var firstname,lstname="rafles","nainggolan"

fmt.Println("nama depan "+firstname+" nama belakang "+lastname)

* var nama=”rafles” //variable global/ bias diluar fungsi
* nama :=”rafles” //scope nya hrs di dalam fungsi
* strconv => ubah string ke integer atau sebaliknya ( “1000”+2000= 12000 )
* var name string

name=”rafles”

* const name=”rafles” //tidak dapat diubah
* const (

name=”rafles”,

address=”jakarta”

)

**tipe data STRING**

* len(string)

**Type declaration alias**

* type phone int

type maried bool

var isMaried married=true

fmt.Println(isMaried)

**ARRAY**

var people[4]string  
people[0]="rafles"  
people[1]="budi"  
people[2]="mawar"  
people[3]="anto"  
fmt.Println(people[0])

langsung

var people = [4]string{ "rafles","budi","mawar","anto",}  
fmt.Println(people)

**SLICE**

* **Method1**

func getslice(){  
 var bilangan= [...]int{1,2,3,4,5,6,7,8,9,10,11,12}  
 var slice1 = people[2:5] //3,4,5,6  
  
 bilangan[4]=66 //change index 4 to 66 in month array  
 slice1[3]=66 //change index 3 to 66 in data1 array  
 slice1=append(slice1, 999) // add data to last index,if full get new array  
 dataArr :=slice1 //get array  
 panjang :=len(slice1) //length array get  
 panjangSisa :=cap(slice1) // panjang dari index awal sampai akhir  
}

* **method Make slice**

func getslice(){  
 newSlice :=make([]string,2,10)  
 newSlice[0]="anto"  
 newSlice[1]="mawar"  
}

* **Copy Slice**

copySclice :=make([]string, len(newSlice), cap(newSclice))

**SLICE AND MAP**

var chickens = []map[string]string{  
 map[string]string{"name": "chicken blue", "gender": "male"},  
 map[string]string{"name": "chicken red", "gender": "male"},  
 map[string]string{"name": "chicken yellow", "gender": "female"},  
}  
  
for \_, chicken := range chickens {  
fmt.Println(chicken["gender"], chicken["name"])  
}

**Tipe data MAP**

* **Dengan inisiasi data awal**

func tipeMap(){  
 var person = map[string]string{  
 "name": "rafles",  
 "address": "jakarta",  
 }  
 person["phone"]="08123456" // adding data object  
 fmt.Println(person["name"])  
 // len(map),map[key],map[key]=value,delete(map,key)  
}

* **Tanpa data awal**

func tipeMap(){  
 var person = make(map[string] string)  
 person["name"]="budi"  
 person["alamat"]="bandung"  
 person["phone"]="bandung"  
  
 delete(person,"phone")  
}

**ITERASI MAP**

func iterasiMap(){  
 var chicken = map[string]int{  
 "januari": 50,  
 "februari": 40,  
 "maret": 34,  
 "april": 67,  
 }  
  
 for key, val := range chicken {  
 fmt.Println(key, " \t:", val)  
 }  
}

**EXIST MAP**

var chicken = map[string]int{"januari": 50, "februari": 40}  
var value, isExist = chicken["mei"]  
  
if isExist {  
fmt.Println(value)  
} else {  
fmt.Println("item is not exists")  
}

**IF CONDITION**

func aritMatika(a int, b int, status string) (string, string) {  
 var hasil int  
 if status == "jumlah" {  
 hasil=a+b  
 } else if status =="bagi" {  
 hasil=a/b  
 } else {  
 hasil=a-b  
 }  
 return " Hasil aritmatika adalah "+strconv.Itoa(hasil),  
 " Jenis aritmatika adalah "+status  
}

**FOR LOOP**

func forLoop(){  
 i:=0  
 for i < 10{  
 fmt.Print(i)  
 i++  
 }  
}

func forLoopArray(){  
 slice :=[]string{"budi","mawar","rales","rani","mira","rini","anto"}  
  
 for i:=0;i < len(slice) ;i++ {  
 fmt.Println(slice[i])  
 }  
  
 //OR  
 for i,value := range slice{  
 fmt.Println("index ",i," = " value)  
 }  
}

**FUNGSI**

**One return**

func aritmatika(a int, b int, status string) string {  
 var hasil int  
 if status == "jumlah" {  
 hasil=a+b  
 } else if status =="bagi" {  
 hasil=a/b  
 } else {  
 hasil=a-b  
 }  
 return string(hasil)  
}  
  
func main() {  
 println("aritmatika ", aritmatika(10,5,"jumlah"))  
   
}

**Multi return**

func aritMatika(a int, b int, status string) (string, string) {  
 var hasil int  
 if status == "jumlah" {  
 hasil=a+b  
 } else if status =="bagi" {  
 hasil=a/b  
 } else {  
 hasil=a-b  
 }  
 return " Hasil aritmatika adalah "+strconv.Itoa(hasil),  
 " Jenis aritmatika adalah "+status  
}  
  
func getMultiplefunc() {  
 hasilAritmatika, jenisAritmatika := aritMatika(10, 5, "jumlah")  
 fmt.Println(hasilAritmatika,jenisAritmatika)  
}

## Fungsi Variadic (variable params tak terbatas)

func calculate(numbers ...int) float64 {  
 var total int = 0  
 for \_, number := range numbers {  
 total += number  
 }  
  
 var avg = float64(total) / float64(len(numbers))  
 return avg  
}  
func main() {  
 var avg = calculate(2, 4, 3, 5, 4, 3, 3, 5, 5, 3)  
 var msg = fmt.Sprintf("Rata-rata : %.2f", avg)  
 fmt.Println(msg)

//OR

var numbers = []int{2, 4, 3, 5, 4, 3, 3, 5, 5, 3}  
 var avg = calculate(numbers...)  
 var msg = fmt.Sprintf("Rata-rata : %.2f", avg)  
}

**TYPE STRUCT**

type person struct {  
 name string  
 age int  
}  
  
type hobby struct {  
 hobbies1 string  
 hobbies2 string  
 person  
}  
type student struct {  
 grade int  
 hobby  
 age int // memiliki kesamaan.akses parent dgn person.age  
}  
  
func dataStruct(){  
 var st1 = student{}  
 st1.name="rafles"  
 st1.grade=90  
 st1.hobbies1="Membacas"  
 st1.hobbies2="Badminton"  
 st1.age = 21 // age of student  
 st1.person.age = 22 // age of person  
 //====atau langsung  
 /\* var p1 = person{name: "wick", age: 21}  
 var s1 = student{person: p1, grade: 2}  
 \*/

fmt.Println("nama "+st1.name+", age ",st1.age,", Grade is ",st1.grade," , Hobby is " + st1.hobbies1+" - "+ st1.hobbies2)  
}

**Slice & Struct**

type person struct {  
 name string  
 age int  
}  
  
var allStudents = []person{  
 {name: "budi", age: 23},  
 {name: "mawar", age: 25},  
 {name: "anto", age: 22},  
}  
  
func dataStruct(){  
 for \_, student := range allStudents {  
 fmt.Println(student.name, "age is", student.age)  
 }  
}

## Slice Anonymous Struct

type person struct {  
 name string  
 age int  
}  
var allStudents = []struct {  
 person  
 grade int  
}{  
 {person: person{"budi", 28}, grade: 7},  
 {person: person{"anto", 25}, grade: 5},  
 {person: person{"mawar", 21}, grade: 8},  
}  
  
func dataStruct(){  
 for \_, student := range allStudents {  
 fmt.Println(student.name, "age is", student.age)  
 }  
}

## Nested struct

type student struct {  
 person struct {  
 name string  
 age int  
 }  
 grade int  
 hobbies []string  
}

# Penggunaan Method GO

**package** main

import (

  "fmt"

)

type Pekerja struct {

  Nama string

  Gaji int

}

func (p Pekerja) lihatPekerja() {

  fmt**.**Println("Nama\t :", p**.***Nama*)

  fmt**.**Println("Gaji\t :", p**.***Gaji*)

}

func main() {

  p1 :**=** Pekerja{

    Nama: "Charly Van Houten",

    Gaji: 1000000,

  }

  p1**.**lihatPekerja()

}

import (

  "fmt"

)

const UMR int **=** 2000000

type Pekerja struct {

  Nama string

  Gaji int

}

func (p Pekerja) lihatPekerja() bool {

  fmt**.**Println("Nama\t :", p**.***Nama*)

  fmt**.**Println("Gaji\t :", p**.***Gaji*)

**if** p**.***Gaji* **<** UMR {

**return** false

  } **else** {

**return** true

  }

}

func main() {

  p1 :**=** Pekerja{

    Nama: "Charly Van Houten",

    Gaji: 1000000,

  }

  fmt**.**Println(p1**.**lihatPekerja())

}

**Tipe fungsi method sama**

**package** main

import ("fmt")

type (

  Category struct {

    Name string

  }

  Post struct {

    Title string

  }

)

func (c Category) lihatData() {

  fmt**.**Println(c)

}

func (p Post) lihatData() {

  fmt**.**Println(p)

}

func main() {

  fmt**.**Printf("From Category\n")

  cats :**=** Category{

    Name: "Berita",

  }

  cats**.**lihatData()

  fmt**.**Printf("From Post\n")

  p :**=** Post{

    Title: "Belajar Golang",

  }

  p**.**lihatData()

}