**GOLANG EXERCISE**

1. **Cara install golang**

* Buka web Golang : <https://go.dev/dl/>
* Install Golang
* Cek version Golang dengan go version di terminal.

1. **Cara membuat program hello world**

* Create New Folder Belajar Golang
* Create File index.go
* Place Code :

package main

import "fmt"

func main() {

fmt.Println("Hello World !")

}

* Compile melalui terminal dengan :
  + Go run index.go, atau
  + Go build index.go

1. **Mengenal tipe data number**

* Go memiliki 2 tipe data number yaitu int dan uint. Berikut pembagiannya

|  |  |  |
| --- | --- | --- |
| **Tipe Data** | **Nilai Minimum** | **Nilai Maksimum** |
| Int8 | -128 | 127 |
| Int16 | -32768 | 32767 |
| Int32 | -2147483648 | 214748… |
| Int64 | -92233… | 92233720… |

* Place Code

package main

import "fmt"

func main() {

fmt.Println("Satu = ", 1)

fmt.Println("Dua = ", 2)

fmt.Println("Tiga Koma Lima = ", 3.5)

}

* Output



1. **Mengenal tipe data Boolean**

* Place Code

package main

import "fmt"

func main() {

fmt.Println("Benar = ", true)

fmt.Println("Salah = ", false)

}

* Output



1. **Mengenal tipe data String**

* Function untuk String

|  |  |
| --- | --- |
| **Function** | **Keterangan** |
| Len(“string”) | Menghitung jumlah karakter pada string. |
| “string”[number] | Mengambil karakter pada posisi yang ditentukan. |

* Place Code

package main

import "fmt"

func main() {

fmt.Println("Raka Putra Eshardiansyah")

fmt.Println(len("Raka Putra Eshardiansyah"))

fmt.Println("Raka Putra Eshardiansyah"[0])

}

* Output



1. **Variabel dalam Golang**

* Variabel golang perlu untuk didefinisikan terlebih dahulu sebelum digunakan.
* Ketika inisialisasi variable wajib untuk mendefinisikan tipe datanya.
* Place Code

func main() {

var name string

name = "Raka Putra Eshardiansyah"

fmt.Println(name)

}

* Output



* Atau bisa melakukan Place Code seperti

package main

import "fmt"

func main() {

var name = "Raka Putra Eshardiansyah"

fmt.Println(name)

}

* Atau bisa melakukan Place Code seperti

package main

import "fmt"

func main() {

name := "Raka Putra Eshardiansyah"

fmt.Println(name)

name = "Programmer Golang"

fmt.Println(name)

}

* Output



* Golang juga dapat mendeklarasikan beberapa variable sekaligus.
* Place Code

package main

import "fmt"

func main() {

var(

name = "Raka Putra Eshardiansyah"

exercise = "Belajar Golang"

)

fmt.Println(name)

fmt.Println(exercise)

}

* Output



1. **Constant dalam Golang**

* Variabel konstan tidak dapat di replace setelah di deklarasikan
* Place Code

package main

import "fmt"

func main() {

const lastName = "Eshardiansyah"

const age = 23

//Konstan = Data tidak dapat diubah jiak telah didefinisikan

fmt.Println("Nama Akhir : ", lastName)

fmt.Println("Umur : ", age)

}

* Output



1. **Konversi Tipe Data**

* Konversi dapat dilakukan salah satunya untuk mengubah variable int.
* Place Code

package main

import "fmt"

func main() {

var nilai32 int32 = 100000

var nilai64 int64 = int64(nilai32)

var nilai8 int8 = int8(nilai64)

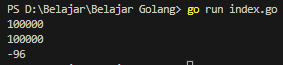
fmt.Println(nilai32)

fmt.Println(nilai64)

fmt.Println(nilai8)

}

* Output



* Contoh lainnya untuk Place Code

package main

import "fmt"

func main() {

var name = "Raka Putra Eshardiansyah"

var e byte = name[0]

var conversion string = string(e)

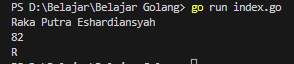
fmt.Println(name)

fmt.Println(e)

fmt.Println(conversion)

}

* Output



1. **Type Declarations**

* Type Declarations digunakan untuk membuat alias untuk suatu type data
* Place Code

package main

import "fmt"

func main() {

type noKTP string

var noKTPRaka noKTP = "1234567890"

fmt.Println(noKTPRaka)

}

* Output



1. **Operasi Matematika**

* Untuk operasi matematika hampir sama dengan Bahasa pemrograman lain
* Place Code

package main

import "fmt"

func main() {

var a = 10

var b = 10

var c = a \* b

fmt.Println(c)

}

* Output



1. **Operasi Perbandingan**

* Operasi perbandigan juga sama dengan Bahasa pemrograman lain
* Place Code

package main

import "fmt"

func main() {

var name1 = "Raka"

var name2 = "Raka"

var result = name1 == name2

fmt.Println(result)

}

* Output



1. **Operasi Boolean**

* Digunakan untuk pengkodisian
* Place Code

package main

import "fmt"

func main() {

var nilaiAkhir = 80

var absensi = 100

var nilaiKKM = 75

if (nilaiAkhir+absensi)/2 > nilaiKKM {

fmt.Println("Lulus")

}

}

* Output



1. **Tipe data Array**

* Sekumpulan data yang ditampung didalam total deklarasi
* Place Code

package main

import "fmt"

func main() {

var names [3]string

names[0] = "Raka"

names[1] = "Putra"

names[2] = "Eshardiansyah"

fmt.Println(names[0])

fmt.Println(names[1])

fmt.Println(names[2])

}

* Output



1. **Tipe data Slice**

* Tipe data slice adalah potongan dari array
* Ukuran slice bisa berubah
* Slice dan Array selalu terkoneksi
* Membuat Slice dari Array

|  |  |
| --- | --- |
| **Membuat Slice** | **Keterangan** |
| Array[low:high] | Membuat slice dimulai dari indeks low – indeks high |
| Array[low:] | Membuat slice dimulai dari indeks low - ??? |
| Array[:high] | Membuat slice dimulai dari indeks 0 – indeks high |
| Array[:] | Membuat slice dimulai dari indeks 0 - ??? |

* Perbedaan penulisan Array dan Slice

iniArray := [5]int{1, 2, 3, 4, 5}

iniSlice := []int{1, 2, 3, 4, 5}

* Place Code

var months = [...]string{

"Januari",

"Februari",

"Maret",

"April",

"Mei",

"Juni",

"Juli",

"Agustus",

"September",

"Oktober",

"November",

"Desember"}

var slice2 = months[10:]

fmt.Println(slice2)

var slice3 = append(slice2, "Syawal")

slice3[0] = "Ramadhan"

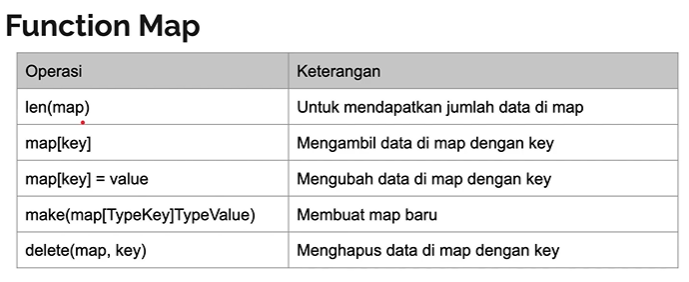
fmt.Println(slice3)

* Output



1. **Tipe data Map**

* Map digunakan untuk membuat key dan values pada 1 validasi
* Function manipulation yang dapat digunakan di Map



* Place Code

book := map[string]string{

"title": "Novel Terbaru 2023",

"author": "Lily",

"sinopsis": "..."}

fmt.Println(len(book))

fmt.Println(book["sinopsis"])

fmt.Println("")

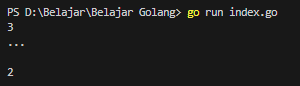
//Template delete(book, [nama\_key])

delete(book, "sinopsis")

fmt.Println(len(book))

fmt.Println(book["sinopsis"])

* Output



1. **Penggunaan IF Expression**

* IF digunakan untuk percabangan di Golang.
* Place Code

name := "Eshardiansyah"

if name == "raka" {

fmt.Println("Hai Raka, Welcome !")

} else {

fmt.Println("Are u a traitor ?")

}

//If short statement

var length = len(name)

if length > 5 {

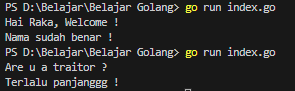
fmt.Println("Terlalu panjanggg !")

} else {

fmt.Println("Nama sudah benar !")

}

* Output



1. **Switch Expression**

* Selain menggunakan if untuk percabangan dapat menggunakan switch expression.
* Place Code

name := "Admin"

switch name {

case "Raka":

fmt.Println("Halo Raka, Welcome !")

case "Admin":

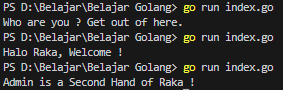
fmt.Println("Admin is a Second Hand of Raka !")

default:

fmt.Println("Who are you ? Get out of here.")

}

* Output



* Switch juga dapat digunakan untuk percabangan Boolean
* Place Code

name := "Adminnnn"

var lengthName = len(name)

switch lengthName > 5 {

case true:

fmt.Println("Nama terlalu panjang")

case false:

fmt.Println("Nama sudah cocok")

}

* Output



* Another Place Code

name := "Kara"

lengthName := len(name)

switch {

case lengthName > 10:

fmt.Println("Nama panjang sekali, harus ganti nama")

case lengthName > 5:

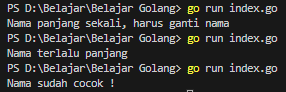
fmt.Println("Nama terlalu panjang")

default:

fmt.Println("Nama sudah cocok !")

}

* Output



1. **For Loops di Golang**

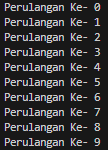
* For loops digunakan untuk mengulang
* Place Code

for counter := 0; counter < 10; counter++ {

fmt.Println("Perulangan Ke-", counter)

}

* Output



* For loops juga bisa digunakan untuk membaca nilai di dalam Slice.
* Place Code

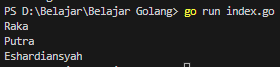
sliceName := []string{"Raka", "Putra", "Eshardiansyah"}

for i := 0; i < len(sliceName); i++ {

fmt.Println(sliceName[i])

}

* Output



* For Loops juga dapat digunakan untuk mendapatkan nilai di MAP
* Place Code

book := map[string]string{

"title": "Komik Naruto Vol.270",

"author": "Masashi Kishimoto",

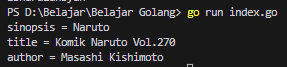
"sinopsis": "Naruto"}

for key, value := range book {

fmt.Println(key, "=", value)

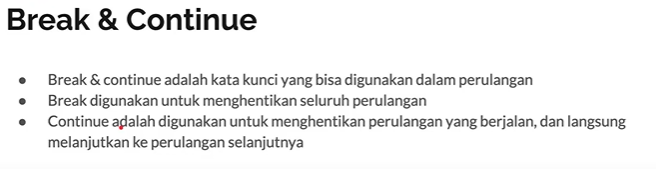
}

* Output



1. **Break & Continue di Golang**

* Penjalasan



* Place Code

for i := 0; i < 10; i++ {

fmt.Println("Perulangan ke-", i)

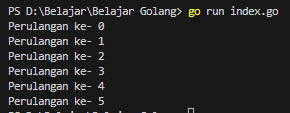
if i == 5 {

break

}

}

* Output



* Place Code

for i := 0; i < 10; i++ {

if i%2 == 0 {

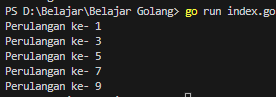
continue

}

fmt.Println("Perulangan ke-", i)

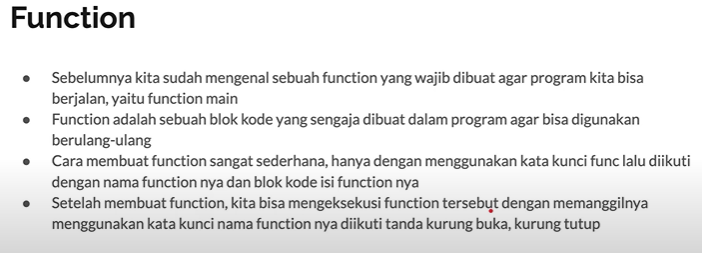
}

* Output



1. **Function di Golang**

* Penjelasan



* Place Code

func sayMyName() {

fmt.Println("Halo saya Raka")

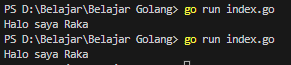
}

func main() {

sayMyName()

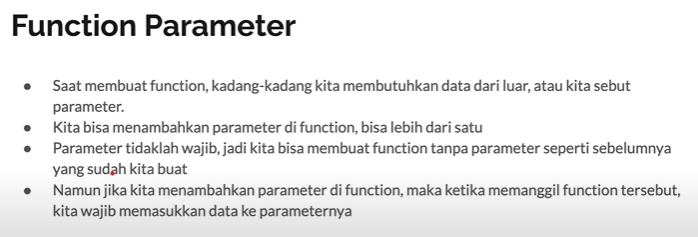
}

* Output



1. **Function dengan Parameter di Golang**

* Penjelasan



* Place Code

func sayMyName(name string, age int) {

fmt.Println("Halo saya", name, ". Umur saya", age)

}

func main() {

sayMyName("Raka Putra Eshardiansyah", 23)

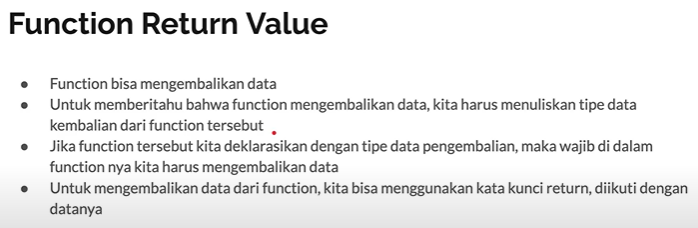
}

* Output



1. **Function Return Value di Golang**

* Penjelasan



* Place Code

func sayMyName(firstName string, lastName string) string {

return "Halo saya awal saya " + firstName + " dan nama akhir saya " + lastName

}

func main() {

result := sayMyName("Raka", "Eshardiansyah")

fmt.Println(result)

}

* Output



1. **Returning Multiple Values di Golang**

* Penjelasan
* Place Code

func sayMyName(name string, age int) (string, int) {

return name, age

}

func main() {

fullName, presentAge := sayMyName("Raka Putra Eshardiansyah", 23)

fmt.Println(fullName)

fmt.Println(presentAge)

//Menghiraukan return value

fullName, \_ = sayMyName("Karaaaes", 23)

fmt.Println(fullName)

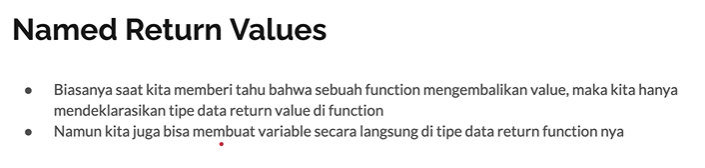
}

* Output



1. **Named Return Values di Golang**

* Penjelasan



* Place Code

func getFullName() (firstName string, middleName string, lastName string, age int) {

firstName = "Raka"

middleName = "Putra"

lastName = "Eshardiansyah"

age = 23

return

}

func main() {

firstName, middleName, lastName, \_ := getFullName()

fmt.Println(firstName)

fmt.Println(middleName)

fmt.Println(lastName)

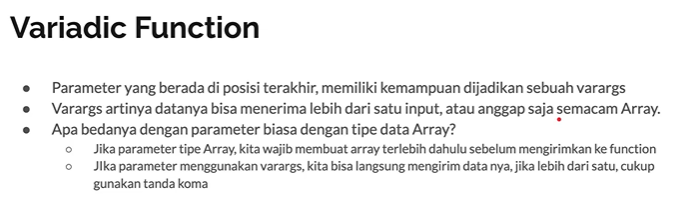
}

* Output



1. **Variadic Function di Golang**

* Penjelasan



* Place Code

func sumAll(numbers ...int) int {

total := 0

for \_, value := range numbers {

total += value

}

return total

}

func main() {

total := sumAll(10, 10, 10)

fmt.Println(total)

slice := []int{10, 25, 20}

total = sumAll(slice...)

fmt.Println(total)

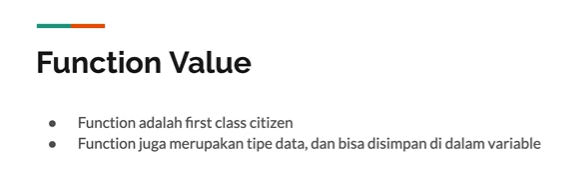
}

* Output



1. **Function Value di Golang**

* Penjelasan



* Place Code

func getGoodbye(name string) string {

return "Good Bye " + name

}

func main() {

sayGoodbye := getGoodbye

result := sayGoodbye("Raka")

fmt.Println(result)

//Or call like this

fmt.Println(getGoodbye("Raka"))

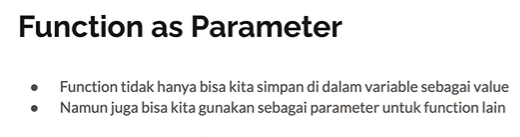
}

* Output



1. **Function as Parameter di Golang**

* Penjelasan



* Place Code

type Filter func(string) string

func sayHelloWithFilter(name string, filter Filter) {

nameFiltered := filter(name)

fmt.Println("Hello", nameFiltered)

}

func spamFilter(name string) string {

if name == "Anjing" {

return "..."

} else {

return name

}

}

//Cara penggunaan

func main() {

sayHelloWithFilter("Eko", spamFilter)

sayHelloWithFilter("Anjing", spamFilter)

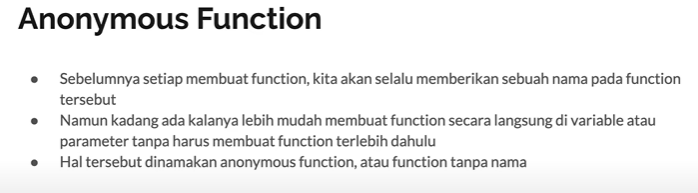
}

* Output



1. **Anonymous Function di Golang**

* Penjelasan



* Place Code

type Blacklist func(string) bool

func registerUser(name string, blacklist Blacklist) {

if blacklist(name) {

fmt.Println("You are blocked", name)

} else {

fmt.Println("Welcome", name)

}

}

func main() {

//Anonymous function, function tanpa nama

blacklist := func(name string) bool {

return name == "admin"

}

registerUser("Raka", blacklist)

registerUser("admin", blacklist)

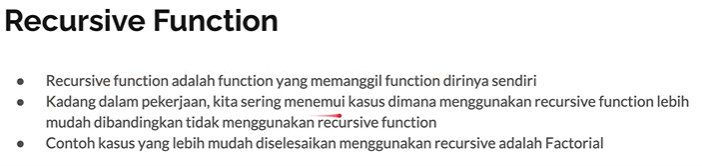
}

* Output



1. **Recursive Function di Golang**

* Penjelasan



* Place Code

func factorialLoop(value int) int {

result := 1

for i := value; i > 0; i-- {

result \*= i //result = result \* i

}

return result

}

func main() {

factorial := factorialLoop(5)

fmt.Println(factorial)

}

* Output



* Another place code

func factorialRecursive(value int) int {

if value == 1 {

return 1

} else {

return value \* factorialRecursive(value-1)

}

}

func main() {

factorial := factorialRecursive(5)

fmt.Println(factorial)

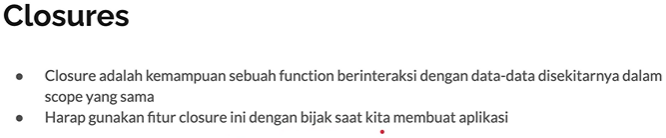
}

* Output



1. **Closure di Golang**

* Penjelasan



* Place Code

func main() {

counter := 0

increment := func() {

counter++

fmt.Println(counter)

}

increment()

increment()

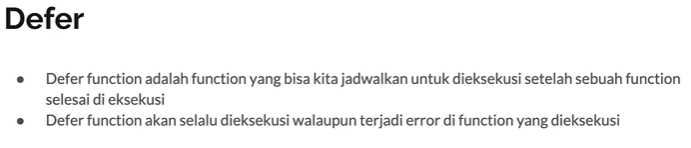
}

* Output



1. **Defer, Panic & Recover di Golang**

* Penjelasan Defer



* Place Code Defer

func logging() {

fmt.Println("Selesai memanggil function")

}

func runApplication(value int) {

defer logging()

fmt.Println("Run Application")

result := 10 / value

fmt.Println("Result", result)

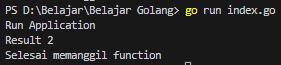
}

func main() {

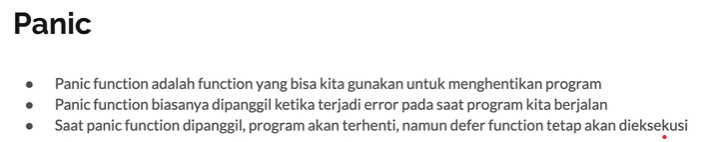
runApplication(5)

}

* Output



* Penjelasan Panic



* Place Code Panic

func endApp() {

fmt.Println("Aplikasi selesai")

}

func runApp(name string) {

defer endApp()

if name == "Admin" {

panic("ERROR")

}

fmt.Println("Aplikasi Berjalan")

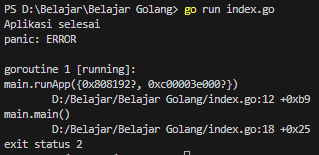
}

func main() {

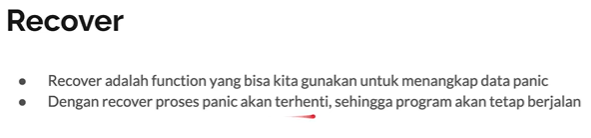
runApp("Admin")

}

* Output



* Penjelasan Recover



* Place Code Recover

func endApp() {

message := recover()

if message != nil {

fmt.Println("Error dengan message :", message)

}

fmt.Println("Aplikasi selesai")

}

func runApp(name string) {

defer endApp()

if name == "Admin" {

panic("ERROR")

}

fmt.Println("Aplikasi Berjalan")

}

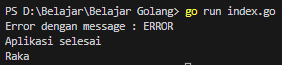
func main() {

runApp("Admin")

fmt.Println("Raka")

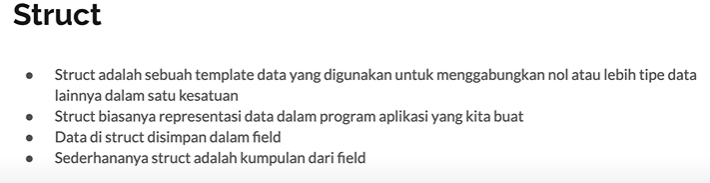
}

* Output



1. **Struct di Golang**

* Penjelasan



* Penjelasan data lanjutan



* Place Code

type Customer struct {

name, address string

age int

}

func main() {

var identitas Customer

identitas.name = "Raka Putra Eshardiansyah"

identitas.address = "Bekasi"

identitas.age = 23

fmt.Println(identitas)

//atau panggil seperti ini

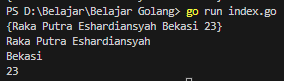
fmt.Println(identitas.name)

fmt.Println(identitas.address)

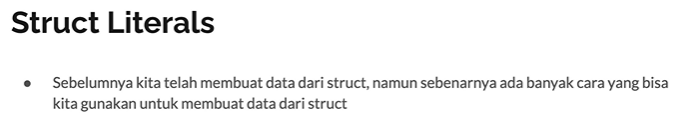
fmt.Println(identitas.age)

}

* Output



* Penjelasan Struct Literals



* Place Code

joko := Customer{

name: "Joko Pranowo",

address: "Jakarta",

age: 35,

}

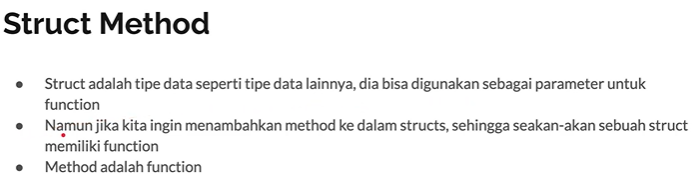
fmt.Println(joko)

* Output



1. **Struct Method**

* Penjelasan



* Place Code

func (customer Customer) sayHello(name string) {

fmt.Println("Hi,", name, "My name is", customer.name)

}

func main() {

var identitas Customer

identitas.name = "Raka Putra Eshardiansyah"

identitas.address = "Bekasi"

identitas.age = 23

identitas.sayHello("Raka")

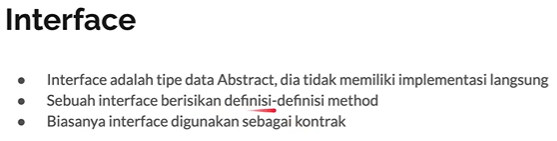
}

* Output



1. **Interface di Golang**

* Penjelasan



* Place Code

type HasName interface {

getName() string

}

func sayHello(hasName HasName) {

fmt.Println("Hello", hasName.getName())

}

type Person struct {

name string

}

func (person Person) getName() string {

return person.name

}

func main() {

var identitas Person

identitas.name = "Raka"

sayHello(identitas)

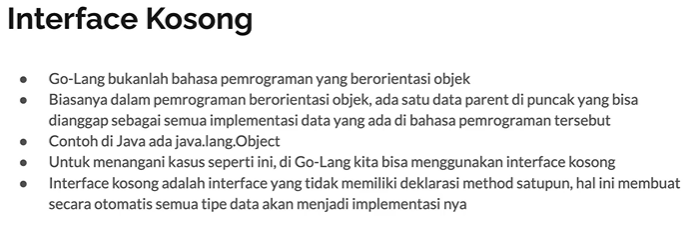
}

* Output



1. **Interface Kosong di Golang**

* Penjelasan





* Place Code

func Ups(i int) interface{} {

if i == 1 {

return 1

} else if i == 2 {

return true

} else {

return "ups"

}

}

func main() {

var data interface{} = Ups(1)

fmt.Println(data)

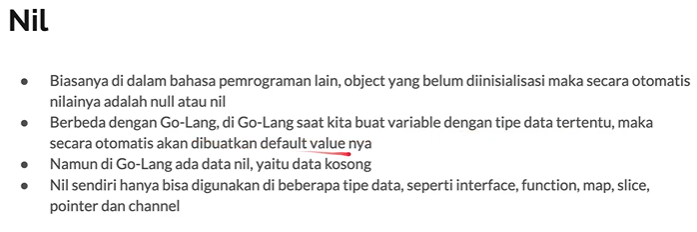
}

* Output



1. **Nil di Golang**

* Penjelasan



* Place Code

func newMap(name string) map[string]string {

if name == "" {

return nil

} else {

return map[string]string{

"name": name}

}

}

func main() {

var person map[string]string = newMap("")

if person == nil {

fmt.Println("Data Kosong")

} else {

fmt.Println(person)

}

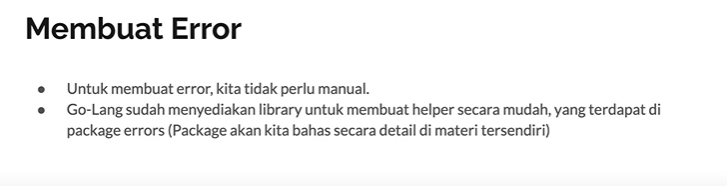
}

* Output



1. **error Interface di Golang**

* Penjelasan



* Place Code

import (

"errors"

"fmt"

)

func Pembagi(nilai int, pembagi int) (int, error) {

if pembagi == 0 {

return 0, errors.New("Pembagi tidak boleh 0")

} else {

result := nilai / pembagi

return result, nil

}

}

func main() {

hasil, err := Pembagi(100, 0)

if err == nil {

fmt.Println("Hasil", hasil)

} else {

fmt.Println("Error:", err.Error())

}

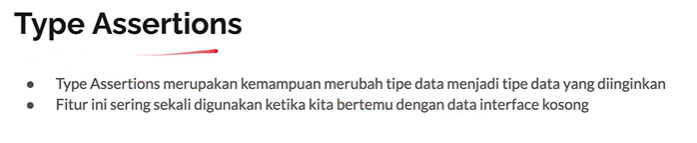
}

* Output



1. **Type Assertions di Golang**

* Penjelasan



* Place Code

func random() interface{} {

return "This is String"

}

func main() {

result := random()

resultFinal := result.(string)

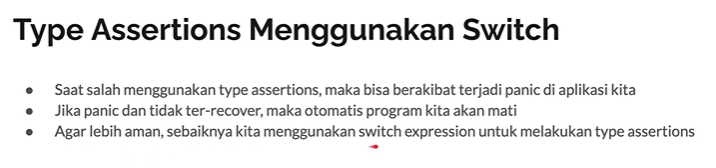
fmt.Println(resultFinal)

}

* Output



* Penjelasan Lanjutan



* Place Code

func random() interface{} {

return 15

}

func main() {

result := random()

switch value := result.(type) {

case string:

fmt.Println("Ini adalah String", value)

case int:

fmt.Println("Ini adalah Int", value)

default:

fmt.Println("Tidak termasuk type data apapun")

}

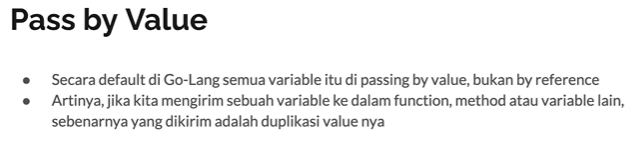
}

* Output



1. **Pointer di Golang**

* Penjelasan



* Place Code

type Address struct {

city, province, country string

}

func main() {

address1 := Address{"Jakarta", "Bekasi", "Bandung"}

address2 := address1

address2.city = "Karawang"

fmt.Println(address1)

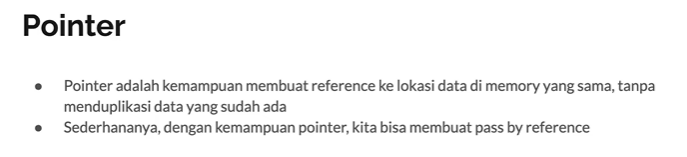
fmt.Println(address2)

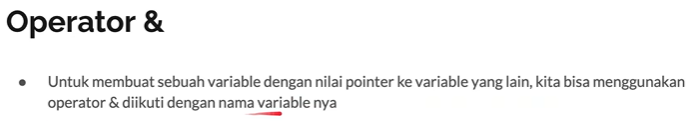
}

* Output



* Penjelasan Pointer





* Place Code

type Address struct {

city, province, country string

}

func main() {

address1 := Address{"Jakarta", "Bekasi", "Bandung"}

address2 := &address1

address2.city = "Karawang"

fmt.Println(address1)

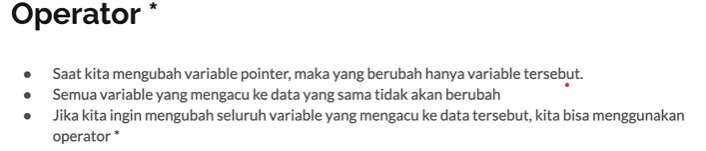
fmt.Println(address2)

}

* Output



* Penjelasan Operator \*



* Place Code

type Address struct {

city, province, country string

}

func main() {

address1 := Address{"Jakarta", "Bekasi", "Bandung"}

address2 := &address1

//Otomatis mengganti referer dengan pointer

\*address2 = Address{"Karawang", "Cikampek", "Purwakarta"}

fmt.Println(address1)

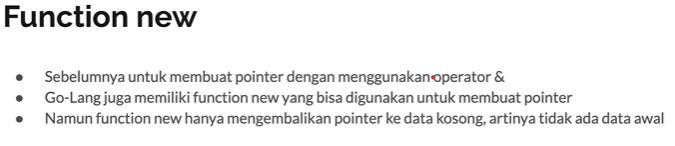
fmt.Println(address2)

}

* Output



* Penjelasan function new



* Place Code

type Address struct {

city, province, country string

}

func main() {

address1 := Address{"Jakarta", "Bekasi", "Bandung"}

address2 := &address1

//Otomatis mengganti referer dengan pointer

\*address2 = Address{"Karawang", "Cikampek", "Purwakarta"}

//Function New

alamat1 := new(Address)

alamat2 := alamat1

alamat2.city = "Purwokerto"

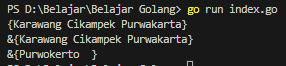
fmt.Println(address1)

fmt.Println(address2)

fmt.Println(alamat2)

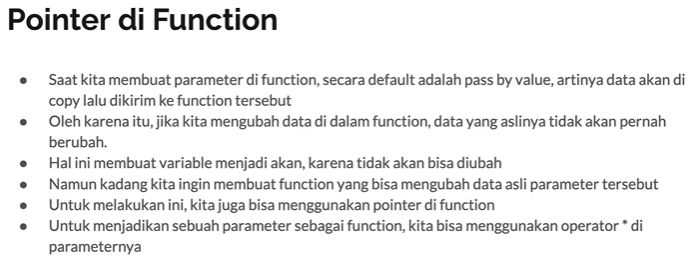
}

* Output



1. **Pointer di Function**

* Penjelasan



* Place Code

type Address struct {

city, province, country string

}

func ChangeCountryToIndonesia(address \*Address) {

address.country = "Indonesia"

}

func main() {

alamat := Address{"Jakarta", "DKI Jakarta", ""}

ChangeCountryToIndonesia(&alamat)

fmt.Println(alamat)

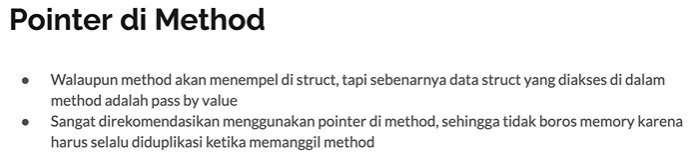
}

* Output



1. **Pointer di Method di Golang**

* Penjelasan



* Place Code

type Man struct {

name string

}

func (man \*Man) Married() {

man.name = "Hello Mr." + man.name

}

func main() {

status := Man{"Raka"}

status.Married()

fmt.Println(status)

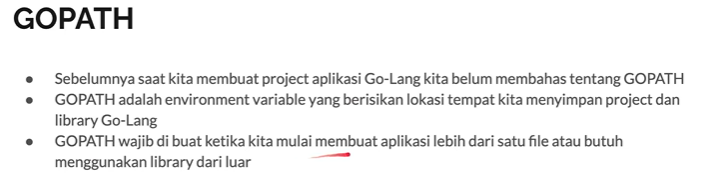
}

* Output



1. **GOPATH di Golang**

* Penjelasan



1. **Package dan Import di Golang**

* Penjelasan

