ENIGMACAMP
IT BOOTCAMP

GO FUNDAMENTAL

HELLO WORLD

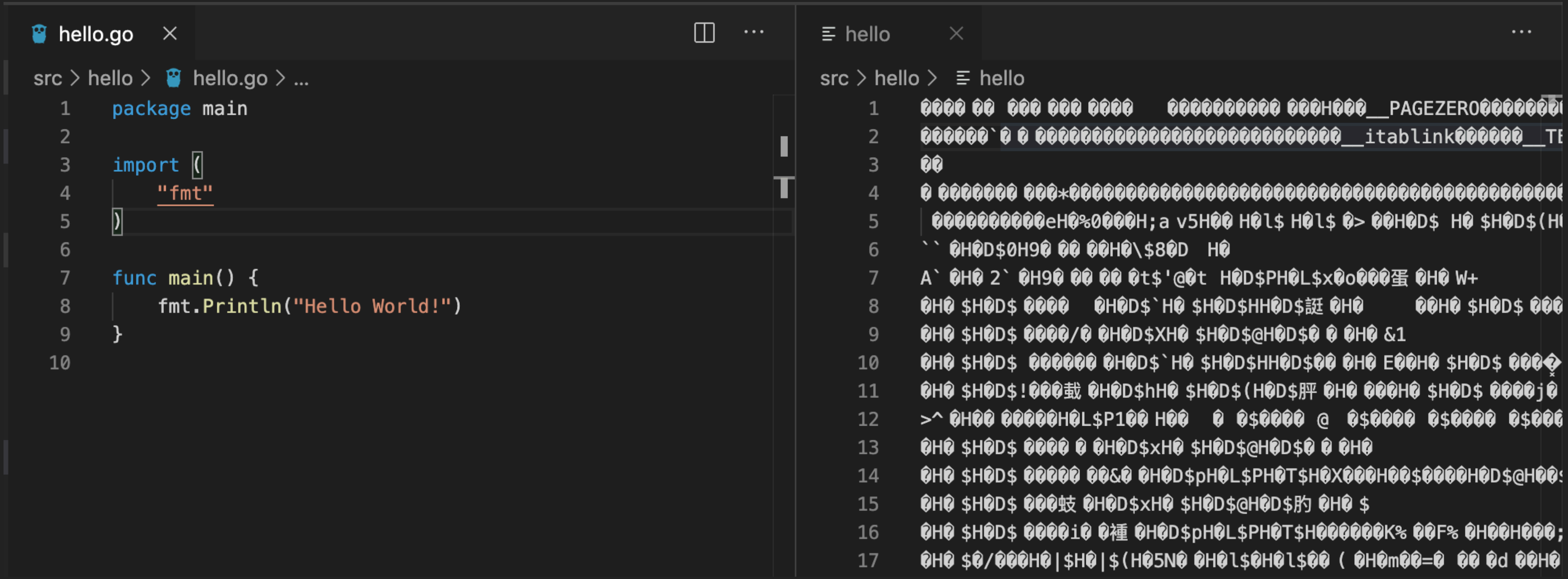
```
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Hello World!")
}
```

- Apa yg dimaksud dengan ***main*** function?
- Nama variable menggunakan camelCase atau PascalCase
- Function menggunakan camelCase atau PascalCase dan merupakan sebuah kata kerja
- Singkatan seluruhnya harus dalam huruf capital, seperti pada ServeHTTP

COMPILE & RUN

```
Joes-MacBook-Pro:hello joe$ ls
hello.go
Joes-MacBook-Pro:hello joe$ go build
Joes-MacBook-Pro:hello joe$ ls
hello           hello.go
Joes-MacBook-Pro:hello joe$ ./hello
Hello World!
```

- go build melakukan proses compile menjadi executable binary
- Executable binary dijalankan menggunakan perintah execute dari OS



```go
package main

import (
	"fmt"
)

func main() {
	fmt.Println("Hello World!")
}
```

```
���� ��  ��� ��� ����    ���������� ���H���__PAGEZERO��������
������`� � ����������������������������__itablink������__TE
��
� ������� ���*���������������������������������������������
 ����������eH�%0���H;a v5H�� H�l$ H�l$ �> ��H�D$  H� $H�D$(H
`` �H�D$0H9� �� ��H�\$8�D  H�
A` �H� 2` �H9� �� �� �t$'@�t  H�D$PH�L$x�o���蛋 �H� W+
�H� $H�D$ ����   �H�D$`H� $H�D$HH�D$詿 �H�      ��H� $H�D$ ���
�H� $H�D$ ����/� �H�D$XH� $H�D$@H�D$� � �H� &1
�H� $H�D$  ������ �H�D$`H� $H�D$HH�D$�� �H� E��H� $H�D$ ���
�H� $H�D$!���裁 �H�D$hH� $H�D$(H�D$肝 �H� ���H� $H�D$ ����j�
>^ �H�� �����H�L$P1�� H��   � �$����  @  �$����  �$����  �$���
�H� $H�D$ ���� � �H�D$xH� $H�D$@H�D$� � �H�
�H� $H�D$ ����� ��&� �H�D$pH�L$PH�T$H�X���H��$����H�D$@H��
�H� $H�D$ ���蚑 �H�D$xH� $H�D$@H�D$肑 �H� $
�H� $H�D$ ����i� �褈 �H�D$pH�L$PH�T$H������K% ��F% �H��H���;
�H� $�/���H�|$H�|$(H�5N� �H�l$�H�l$�� ( �H�m��=�  �� �d ��H�
```

GO COMMAND

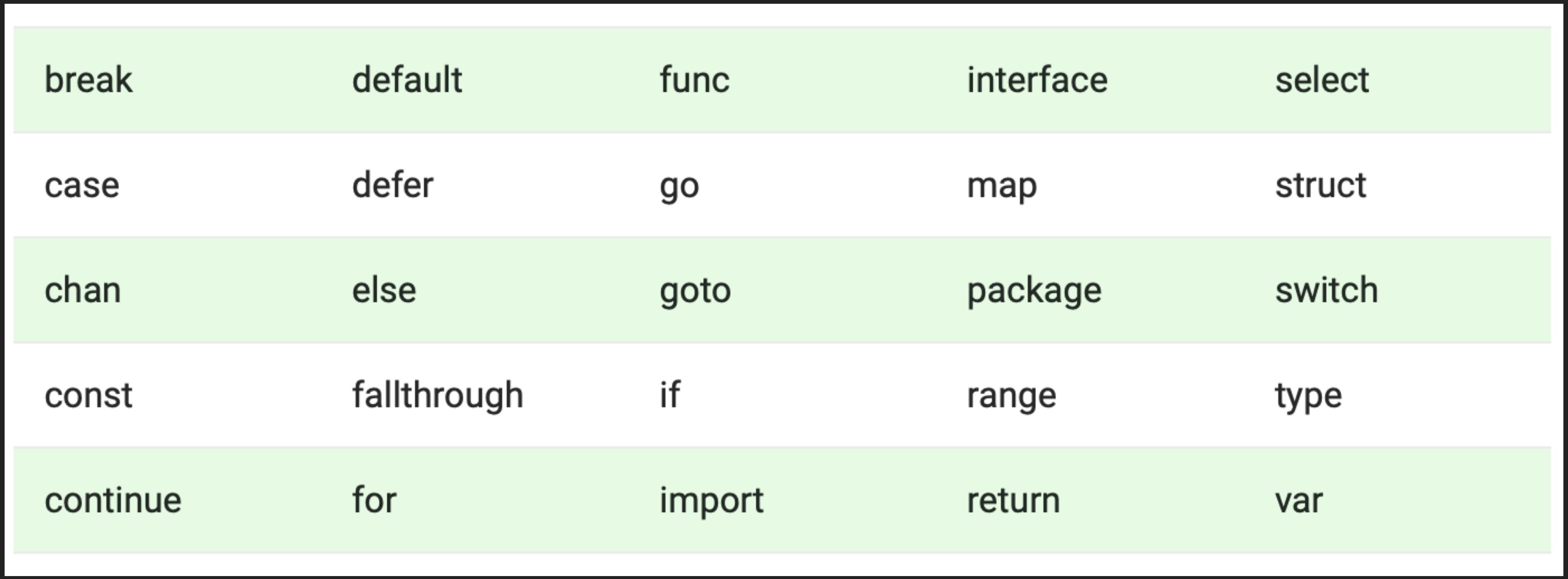
- go build : compile source code menjadi executable binary
- go run : compile source code menjadi executable binary kemudian menjalankannya
- go fmt : format source code menjadi rapi
- go install : compile source code dan meng-install-nya
- go get : download library
- go test : menjalankan unit test

BASIC VARIABLE DATA TYPE

- bool → true atau false
- int → bilangan bulat
  - int int8 int16 int32 int64 uint uint8 uint16 uint32 uint64 uintptr
- byte = uint8
- rune = int32, merepresentasikan Unicode
- float → bilangan berkoma
  - float32 float64
- string → character dan kalimat
- complex → bilangan real dan imaginer

GO KEYWORDS

There are total **25 keywords** present in the Go Lang.
This 25 keywords is **reserved** by Go Lang



break	default	func	interface	select
case	defer	go	map	struct
chan	else	goto	package	switch
const	fallthrough	if	range	type
continue	for	import	return	var

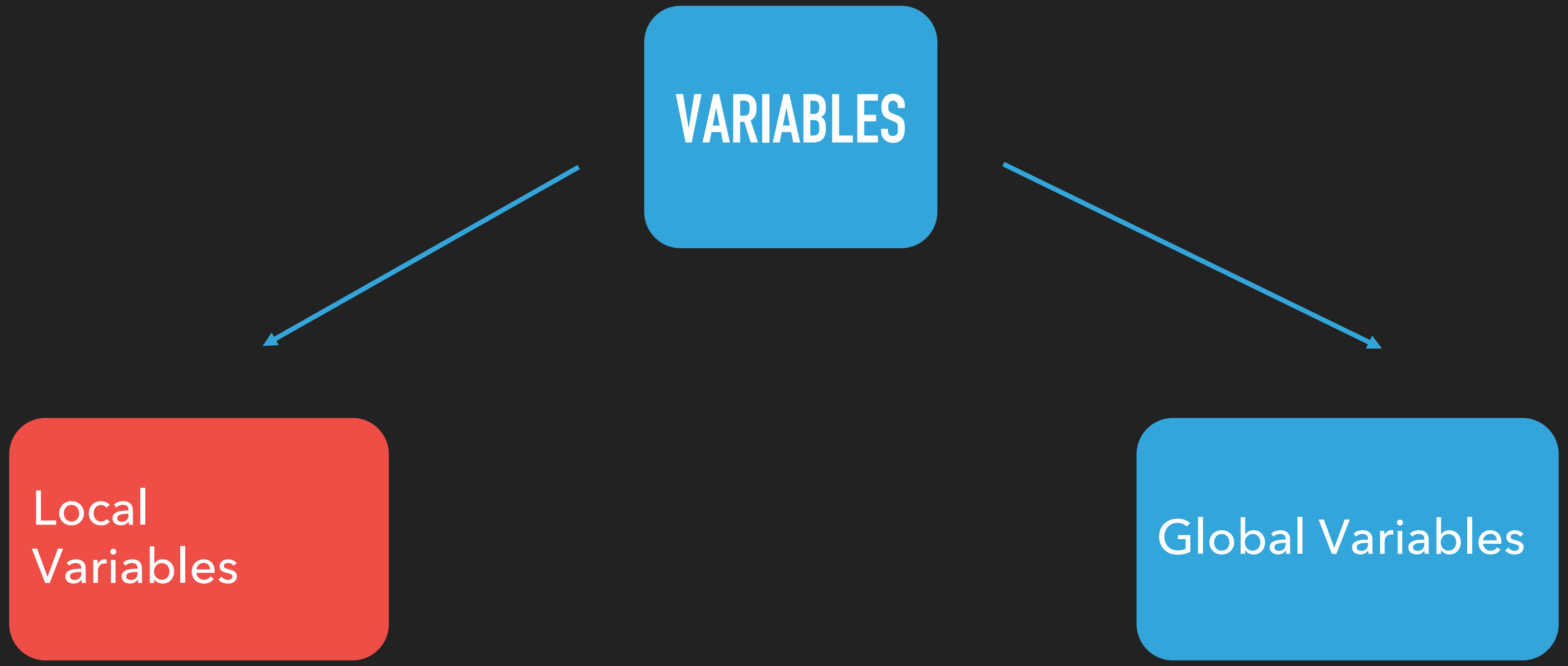
Apa jadi nya kalau kita mendeklarasikan variable dengan nama yg sama dengan salah satu keywords?

ENIGMACAMP
IT BOOTCAMP

CREATE VARIABLE

- Variable Declaration
    - var angka int = 0
- Type Inference
    - angka := 0
- Constant
    - const gravity = 9.8

VARIABLES SCOPE



Untuk basic programming kita focus menggunakan local variable

OPERATORS

Arithmetic Operators

`+ - * / % ++ --`

Relational

`< > == <= >= !=`

Logical Operators

`&& || !`

Assignment Operators

`= := += -= *= /= %= <-`

Pointer Operators

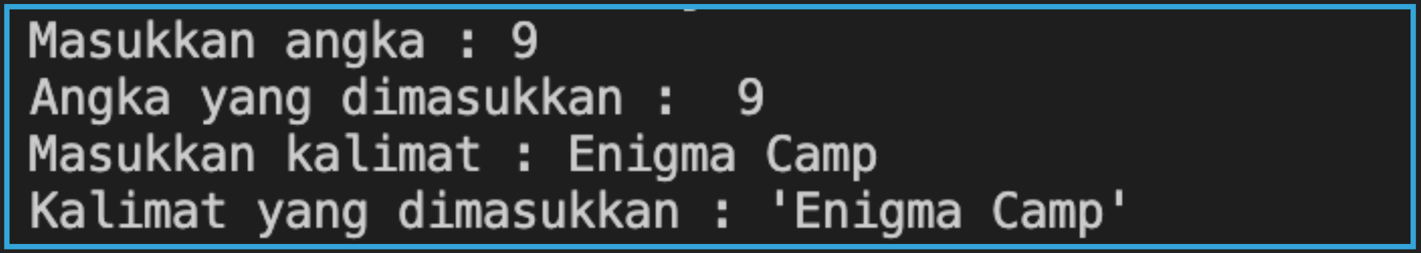
`* &`

Bitwise Operators

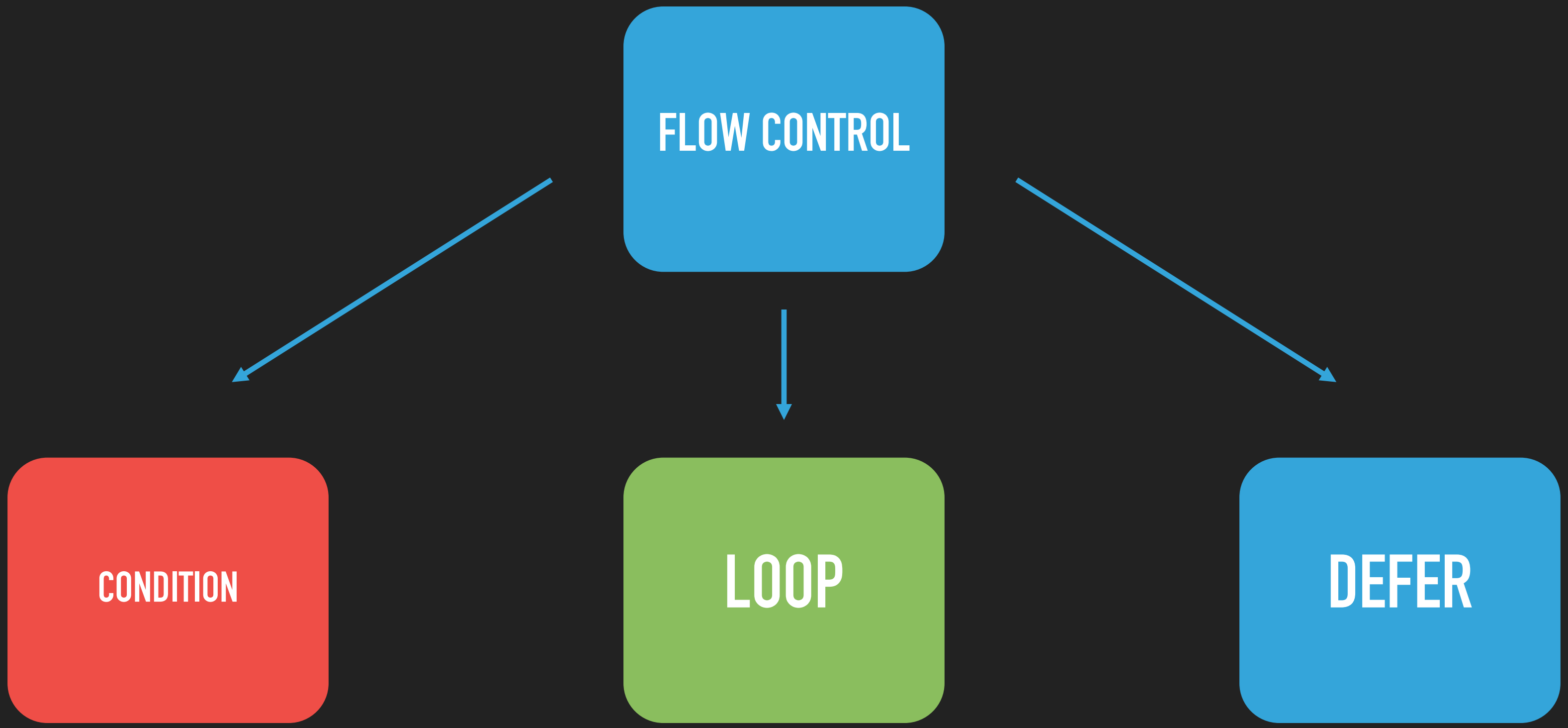
`& | ^ &^ << >>`

INPUT

```go
package main

import (
    "bufio"
    "fmt"
    "os"
)

func main() {

    var angka int
    fmt.Print("Masukkan angka : ")
    fmt.Scan(&angka)
    fmt.Println("Angka yang dimasukkan : ", angka)

    scanner := bufio.NewScanner(os.Stdin)
    fmt.Print("Masukkan kalimat : ")
    scanner.Scan()
    fmt.Println("Kalimat yang dimasukkan : '" + scanner.Text() + "'")

}
```



```
Masukkan angka : 9
Angka yang dimasukkan :  9
Masukkan kalimat : Enigma Camp
Kalimat yang dimasukkan : 'Enigma Camp'
```


FLOW CONTROL
CONDITION
LOOP
DEFER

ENIGMACAMP IT BOOTCAMP

CONDITION - IF

```
// Single If Statement
if condition {
    fmt.Println("This command is executed if condition is true")
}

// If Else Statement
if condition {
    fmt.Println("This command is executed if condition is true")
} else {
    fmt.Println("This command is executed if condition is false")
}

// Else If Statement
if condition {
    fmt.Println("This command is executed if condition is true")
} else if secondCondition {
    fmt.Println("This command is executed if condition is false then secondCondition is true")
} else {
    fmt.Println("This command is executed if condition and second condition are false")
}

// Nested If Statement
if condition {
    if secondCondition {
        fmt.Println("This command is executed if condition and second condition are true")
    }
}
```

CONDITION - SWITCH

```go
switch value {
case 0:
    fmt.Println("This command is executed if value = 0")
case 1:
    fmt.Println("This command is executed if value = 1")
default:
    fmt.Println("This command is executed if value do not match any case")
}

switch { // missing switch expression means "true"
case value == 0:
    fmt.Println("This command is executed if value = 0")
case value == 1:
    fmt.Println("This command is executed if value = 1")
default:
    fmt.Println("This command is executed if value do not match any case")
}
```

LOOP

Looping pada GO menggunakan keyword for dengan 4 bentuk

- Basic For

```go
for [initStatement]; [condition]; [postStatement] {
    statement
}

for i := 0; i < 10; i++ {
    fmt.Println(i)
}
```

- For Ever

```go
for {
    statement
}

for {
    fmt.Println("This command will be executed again and again forever")
}
```

- For a While

```go
for [condition] {
    statement
}

sum := 1
for sum < 10 {
    sum += sum
}
fmt.Println(sum)
```

- For Range

will be discussed with array, slices, and map

Apa kah fungsi 3 statement ini?

- Break
- Continue
- Return

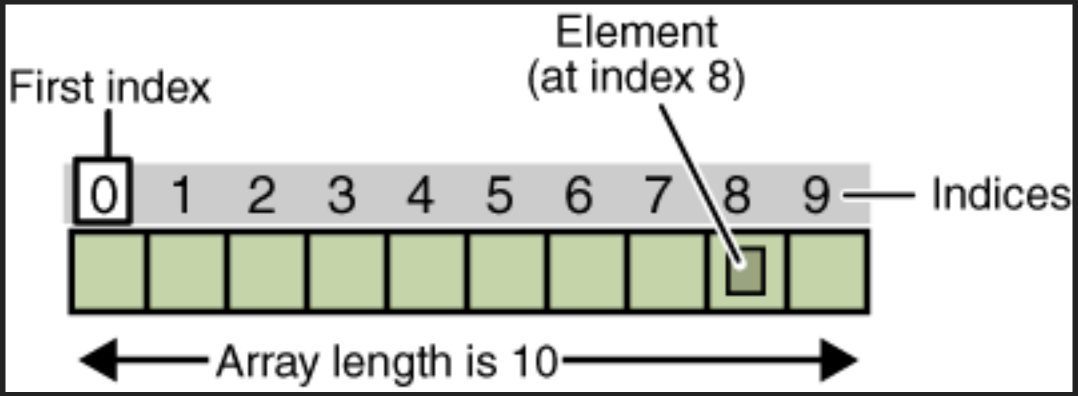
DEFER

Defer Statement will be executed after function returns
Multiple Defer Statement are executed in Last-In-First-Out order

```go
package main

import (
    "fmt"
)

func main() {

    fmt.Println("counting")

    for i := 0; i < 10; i++ {
        defer fmt.Println(i)
    }

    fmt.Println("done")
}
```

ENIGMACAMP
IT BOOTCAMP

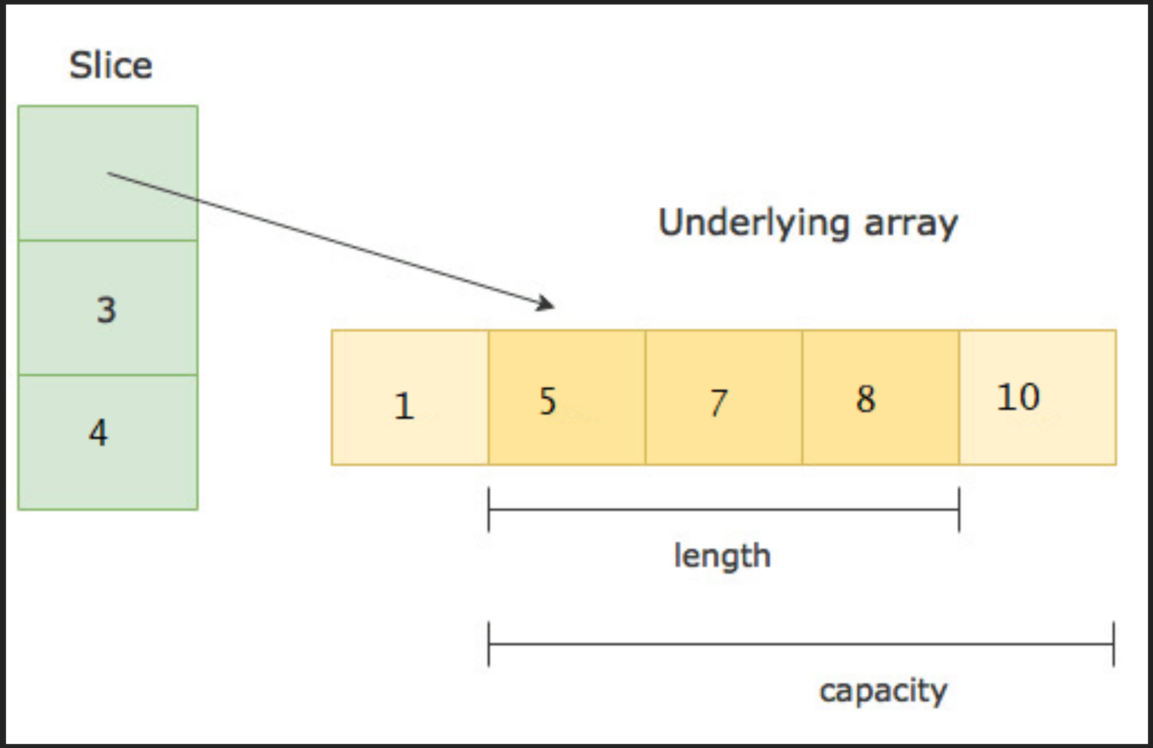
ARRAY



```
var helloWorld [10]string

helloWorld[0] = "Hello"

helloWorld[1] = "World"

fmt.Println(helloWorld[0], helloWorld[1])
primes := [6]int{2, 3, 5, 7, 11, 13}

fmt.Println(primes)
```

NOTE :
ARRAY LENGTH IS FIXED! CANNOT BE RESIZED!
WHAT IS MULTIDIMENSIONAL ARRAY?

SLICES

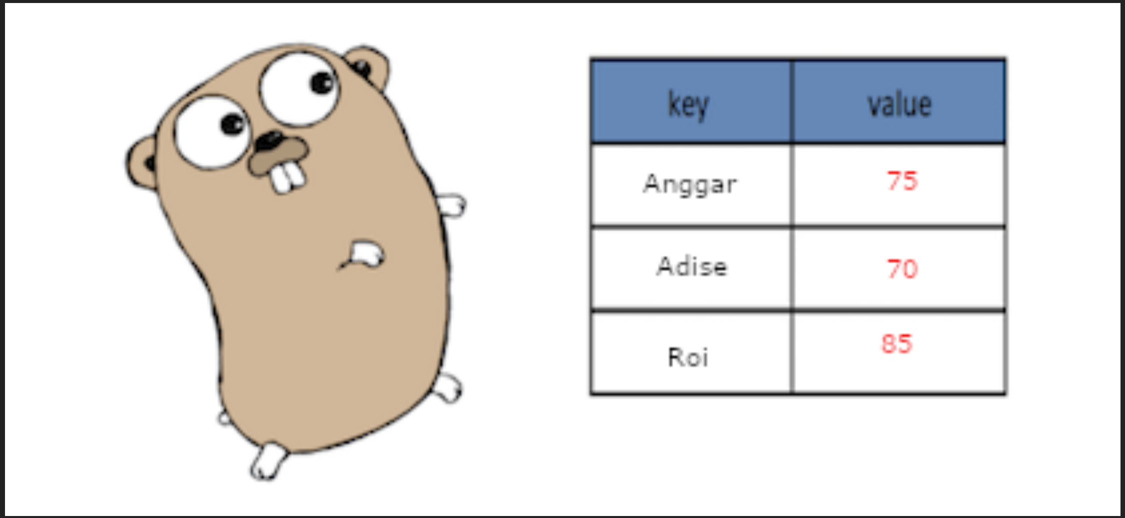


```go
arr := [6]int{1, 5, 7, 8, 10}

var slc []int = arr[1:4]
```

NOTES :
SLICES ONLY A REFERENCE, IT DOESN'T STORE ANY DATA!
SLICE SIZE IS FLEXIBLE, IT CAN BE APPENDED!

MAPS

It's like an array, but with name instead of index



key	value
Anggar	75
Adise	70
Roi	85

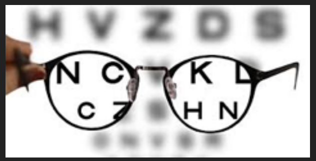
```go
mapScore := make(map[string]int)

mapScore["Anggar"] = 75
mapScore["Adise"] = 70
mapScore["Roi"] = 85

scoreAnggar := mapScore["Anggar"]

scoreIpunx, exist := mapScore["Ipunx"]
fmt.Println("The Score:", scoreIpunx, "Exist?", exist)
```

NOTES :
AM I VALUEABLE? AM I EXIST?

FUNCTION

```
func functionName(param1 string, param2 string) string {

    returnValue := fmt.Sprintf("This function concate %v and %v", param1, param2)

    return returnValue;
}

functionName("Black", "Pink");
```

- Function provide modularity
- Function enable code reusability
- Good function improve code readability

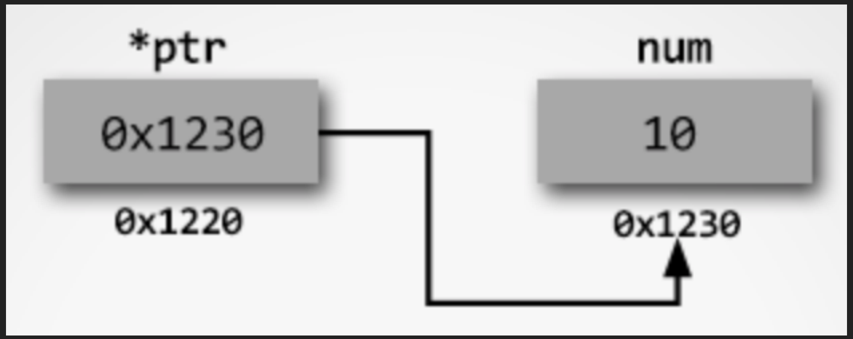


FUNCTION VALUES

```
functionVariable := func(param1 string, param2 string) string {

    returnValue := fmt.Sprintf("This function concate %v and %v", param1, param2)`

    return returnValue
}



functionVariable("Ping", "Pong");
```

NOTES :
CAN FUNCTIONS HAVE THE SAME NAME?
CAN FUNCTION RETURN MULTIPLE VALUES?
WHAT IS PASSING BY REFERENCE AND BY VALUE?

POINTER



As the name itself suggests a pointer is something that points something
It is a variable that holds the memory address of another variable located in computer memory

```go
var num int
var ptr *int      // declare ptr as pointer to integer

num = 10
ptr = &num        // & operator is used to get variable's memory address
fmt.Println(ptr) // 0x1230

*ptr = *ptr / 2  // * operator is used to access the pointed memory
fmt.Println(num) // 5
```

STRUCT

User defined data type used to group multiple(sometime 0 or only 1) field into 1 data type

```go
package main

import "fmt"

// define Vertex struct type that has X and Y field
type Vertex struct {
    X int
    Y int
}

func main() {
    var v Vertex     // Create variable v with Vertex data type
    v = Vertex{1, 2} // Create a new Vertex struct
    v.X = 4          // Access Struct field with a dot
    fmt.Println(v.X)
}
```

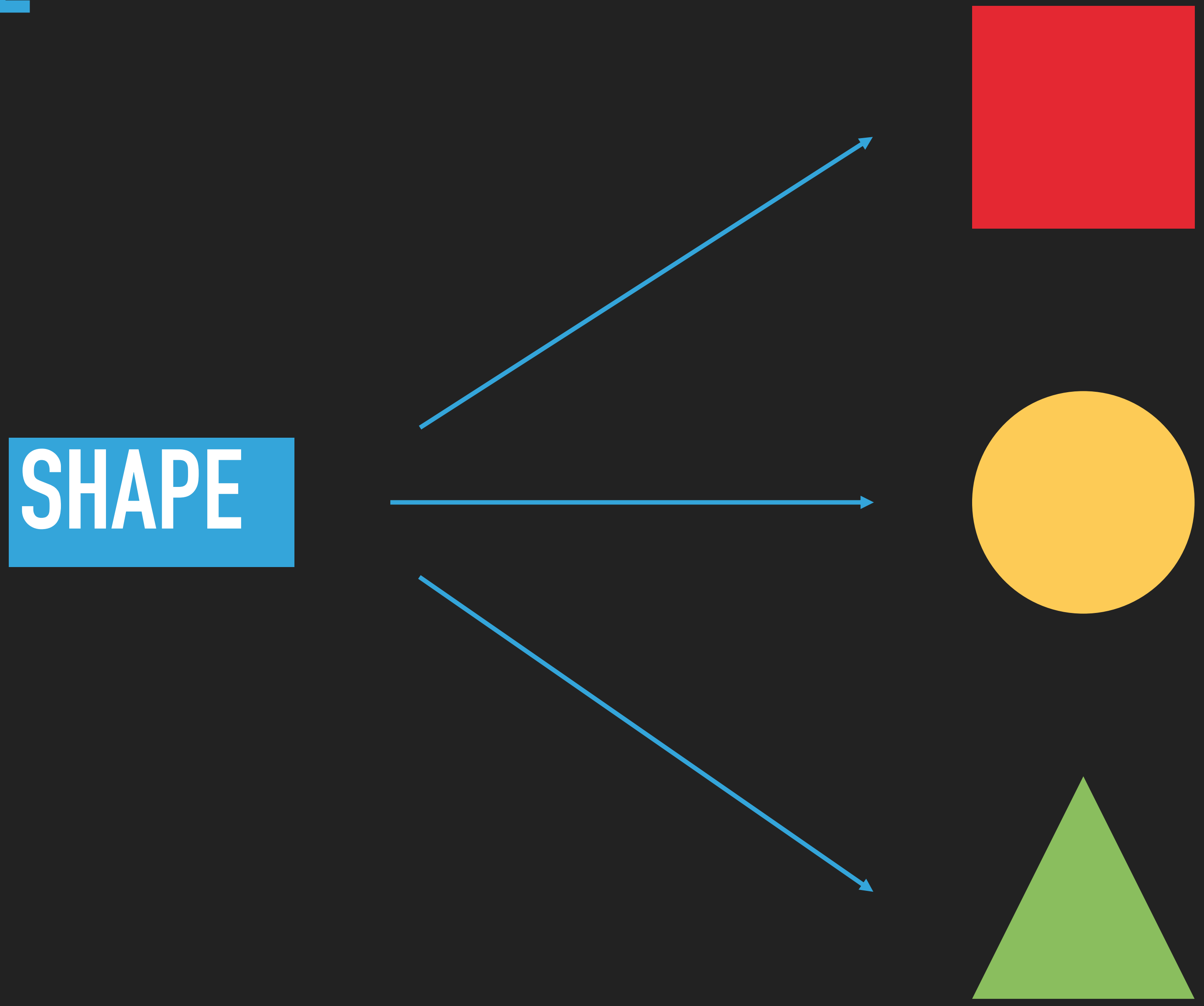
METHOD

Method adalah sebuah fungsi yang menempel pada Custom Type (umumnya struct).
Method memiliki akses ke field/property dari struct melalui receiver.

```go
// Rectangle represent a rectangle shape
type Rectangle struct {
    width  int
    length int
}

func (r Rectangle) getArea() int {
    return r.width * r.length
}

func (r *Rectangle) increaseSize() {
    r.width++
    r.length++
}

func main() {
    rect := Rectangle{
        width:  10,
        length: 5,
    }
    fmt.Println(rect.getArea()) // 50

    rect.increaseSize()
    fmt.Println(rect.width)  // 11
    fmt.Println(rect.length) // 6

}
```

NOTES:
MENGAPA ADA METHOD YANG MEMILIKI * PADA RECEIVER?

INTERFACE



INTERFACE

Interface adalah Custom Type yang hanya berisi deklarasi method tanpa body
Interface dapat menampung Type lain yang memiliki/mengimplemen semua method pada interface

- Shape Interface

```
type Shape interface {
    getArea() float32
    getPerimeter() float32
}
```

- Rectangle Struct

```
type Rectangle struct {
    width  float32
    length float32
}

func (r Rectangle) getArea() float32 {
    return r.width * r.length
}

func (r Rectangle) getPerimeter() float32 {
    return 2 * (r.width + r.length)
}
```

- Circle Struct

```
type Circle struct {
    radius float32
}

func (c Circle) getArea() float32 {
    return math.Pi * c.radius * c.radius
}

func (c Circle) getPerimeter() float32 {
    return 2 * math.Pi * c.radius
}
```

- Main

```
func main() {
    var s Shape
    s = Rectangle{5, 4}
    fmt.Println(s.getArea()) // 20

    println(math.Phi)
    s = Circle{10}
    fmt.Println(s.getArea()) // 314.15927
}
```

NOTES:
BAGAIMANA JIKA INTERFACE TIDAK MEMILIKI METHOD?

ENIGMACAMP IT BOOTCAMP

DON'T
PANIC!!!
HANDLE IT!

ENIGMACAMP
IT BOOTCAMP

FILE

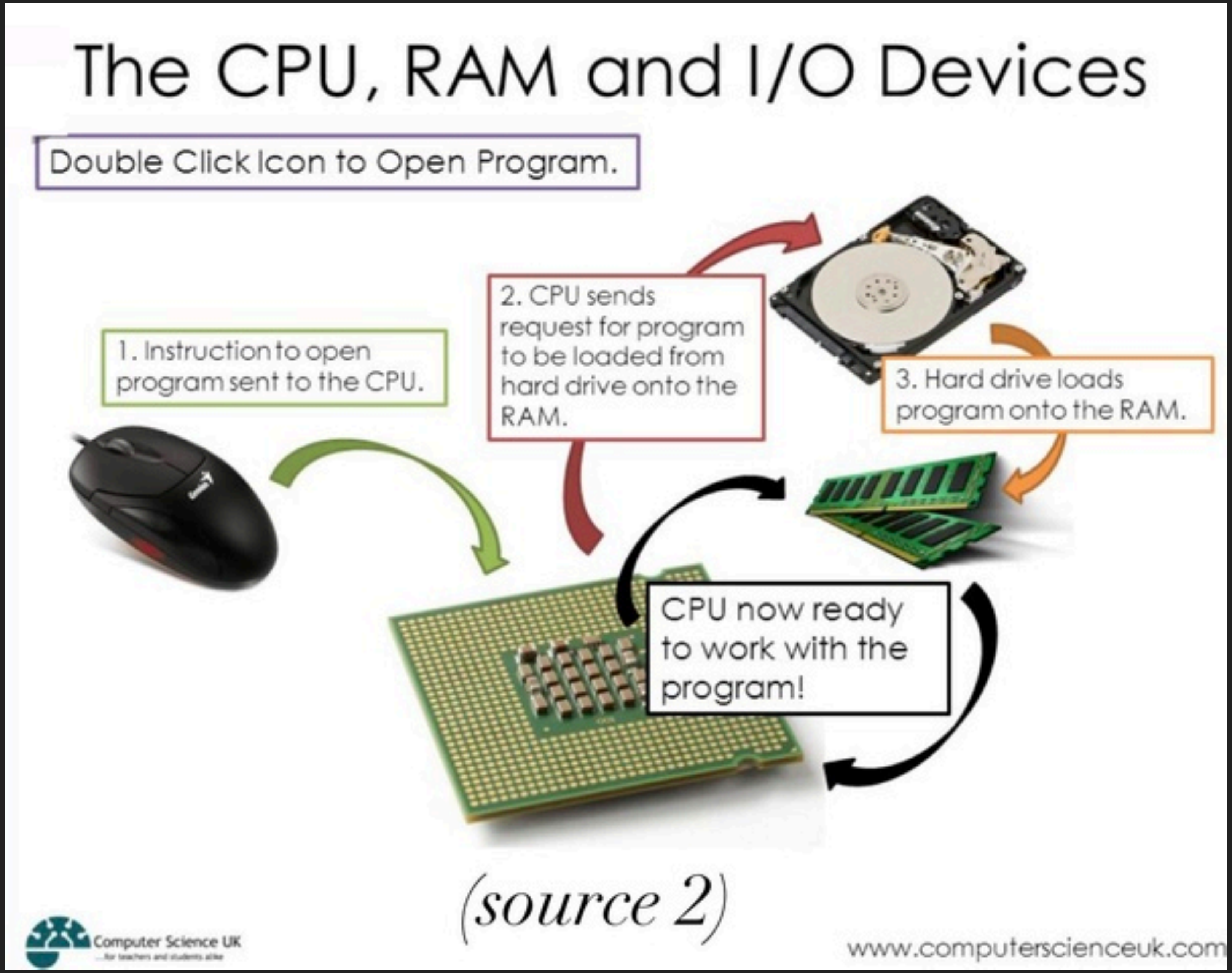
WHAT IF

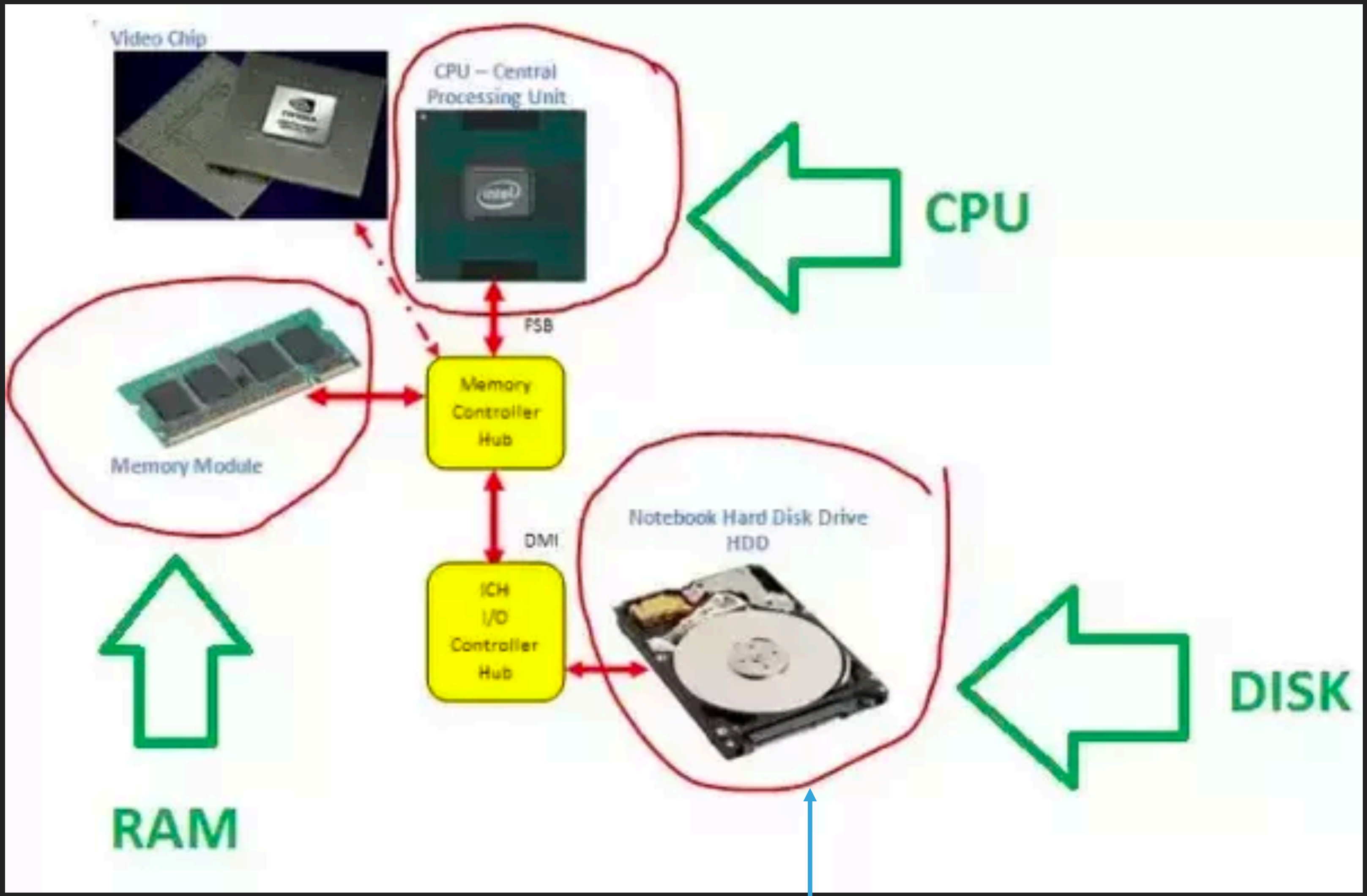
- You want to save your robot’s last position
- You want to save your heroes current hp
- You want to save you current level in you favourite console game

YOUR VARIABLES WILL DISAPPEAR AS SOON AS YOUR APPLICATION STOP RUNNING

Angga Raditya

LETS BACK TO YOUR HARDWARE


The CPU, RAM and I/O Devices
Double Click Icon to Open Program.
1. Instruction to open program sent to the CPU.
2. CPU sends request for program to be loaded from hard drive onto the RAM.
3. Hard drive loads program onto the RAM.
CPU now ready to work with the program!
(source 2)
Computer Science UK
www.computerscienceuk.com



You want to save all those informations permanently in this box as a FILE

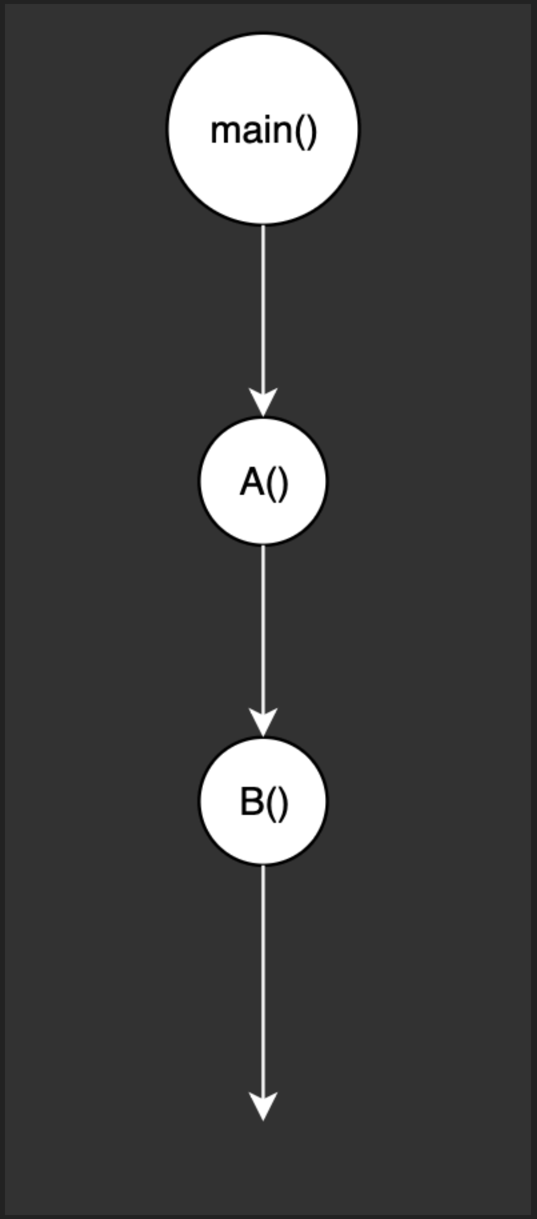
CONCURRENCY

MUTEX

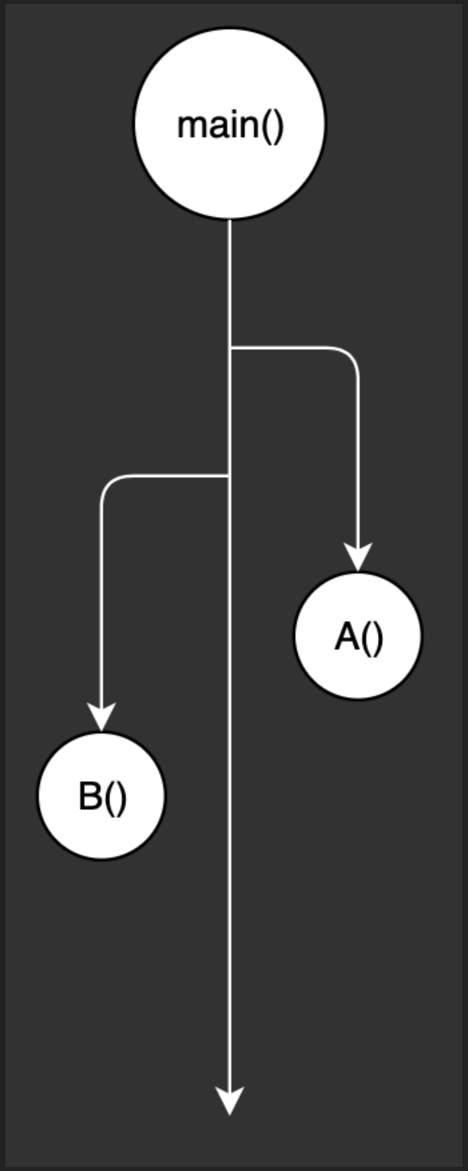
Mutual Exclusion make sure only one goroutine can access a variable at a time to avoid conflicts

GOROUTINE

A goroutine is a function that is capable of running concurrently with other functions



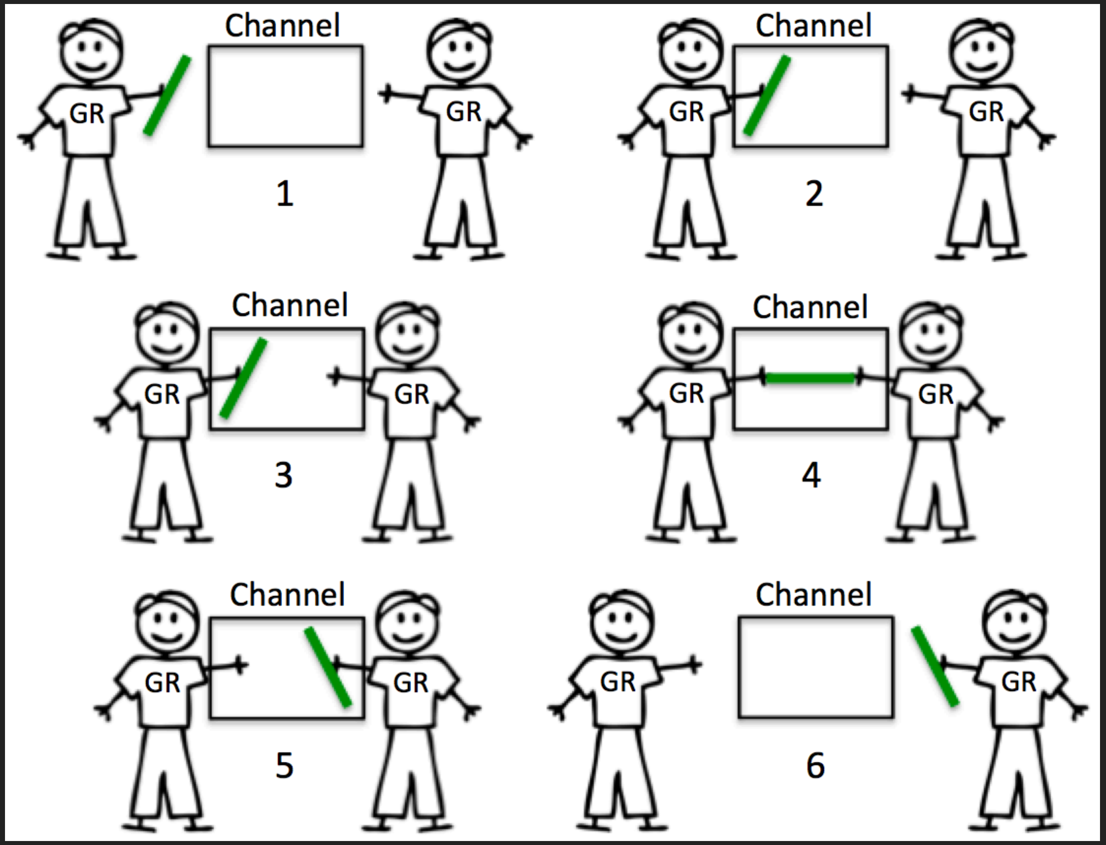
```go
// Sequential
func main() {
    A()
    B()

    time.Sleep(100 * time.Millisecond)
}
```



```go
// Goroutine
func main() {
    go A()
    go B()

    time.Sleep(100 * time.Millisecond)
}
```

CHANNEL

- What is Channel?
- Channel size?
- Channel Range and Close?
- Channel Select?



A channel provides a mechanism for concurrently executing functions to communicate by sending and receiving values of a specified element type.

In simple word you can think it as a box in which you put a item at one end and then pick it from other end