Go Functions

➔ Functions are basic building blocks, A function is used to break a large problem into smaller tasks.
➔ We can invoke a function several time
➔ Functions promote code reusability.

There are 3 types of functions in Go:

- Normal functions with an identifier
- Anonymous or lambda functions
- Method (A function with a receiver)

Function parameters, return values, together with types is called *function signature.*

```
package main

import "fmt"

func nmeage(name string, age int) {
fmt.Println("Your name is ", name)
fmt.Println("Your age is ", age)
}
func main() {
var name string
var age int
fmt.Println("Enter Your name ")
fmt.Scanln(&name)
fmt.Println("Enter Your age")
fmt.Scanln(&age)

nmeage(name, age)
}
```

Output :-

```
Enter Your name
Gautam
Enter Your age
15
Your name is  Gautam
Your age is  15
```

Go function with return :-

Example :-

```
package main

import "fmt"

func yourage(age int) int {
return age
}
func main() {
var age int
fmt.Println("Enter your age")
fmt.Scan(&age)

var myage = yourage(age)
fmt.Println("Your age is ", myage)

}
```

Output :-

```
Enter your age
19
Your age is  19
```

Go Recursion

Recursion :- Calling same function from within the function, Or we ca say a function who call itself repeatedly called recursion.

Example :- Factorial of a number

```
package main

import "fmt"

func main() {
    fmt.Println(factorial(5))
}
func factorial(num int) int {
    if num == 0 {
        return 1
    }
    return num * factorial(num-1)
}
```

Output :-

120

Go Closure

- A closure is a function which refers reference variable from outside its body.
- The function may access and assign to the referenced variable.
- Here we create anonymous function which act as function closure.
- A function which has no name is called anonymous function.

Example :-

```
package main

import "fmt"

func main() {
    var number = 10

    var squarenum = func() int {
        number = number * number
        return number
    }

    fmt.Println(squarenum())
    fmt.Println(squarenum())
    fmt.Println(squarenum())

}
```

Output :-

```
100
10000
100000000
```