Function returns the maximum value

package main

import "fmt"

func main() {

    var a int = 10

    var b int = 20

    c := max(a, b)

    fmt.Println(c)

}

func max(num1 int, num2 int) int {

    var result int

    if num1 > num2 {

        result = num1

    } else {

        result = num2

    }

    return result

}

20

Swap the value with function and passing 2 values to function and get 2 values from function

package main

import "fmt"

func main() {

    a, b := swap("casey", "jacob")

    fmt.Println(a, b)

}

func swap(value1, value2 string) (string, string) {

    return value2, value1

}

jacob casey

Call by value function

package main

import (

    "fmt"

)

func main() {

    var i int = 10

    var j int = 20

    fmt.Println("before swap", i)

    fmt.Println("before swap", j)

    swap(i, j)

    fmt.Println("after swap", i)

    fmt.Println("after swap", j)

    // output will not change after and before swap

}

func swap(value1, value2 int) int {

    var temp int

    temp = value1

    value1 = value2

    value2 = temp

    return temp

}

before swap 10

before swap 20

after swap 10

after swap 20

Call by reference function

package main

import (

    "fmt"

)

func main() {

    var i int = 10

    var j int = 20

    fmt.Println("before swap", i)

    fmt.Println("before swap", j)

    swap(&i, &j)

    fmt.Println("after swap", i)

    fmt.Println("after swap", j)

    // output will not change after and before swap

}

func swap(value1 \*int, value2 \*int) int {

    var temp int

    temp = \*value1

    \*value1 = \*value2

    \*value2 = temp

    return temp

}

before swap 10

before swap 20

after swap 20

after swap 10

**Go - functions as values**

Go programming language provides the flexibility to create functions on the fly and use them as values.

package main

import (

    "fmt"

    "math"

)

func main() {

    getSquareRoot := func(x float64) float64 {

        return math.Sqrt(x)

    }

    fmt.Println(getSquareRoot(9))

}

3

**Go function closure**

Go support anonymous function which can acts as a function closure.

package main

import "fmt"

func return\_increment() func() int {

    i := 0

    return func() int {

        i += 1

        return i

    }

}

func main() {

    a := return\_increment()

    fmt.Println(a())

    fmt.Println(a())

    fmt.Println(a())

    b := return\_increment()

    fmt.Println(b())

    fmt.Println(b())

}

1

2

3

1

2

Method in Go

Go programming language supports special types of functions called methods. In method declaration syntax, a "receiver" is present to represent the container of the function. This receiver can be used to call a function using "." operator. For example −

package main

import (

    "fmt"

    "math"

)

/\* define a circle \*/

type Circle struct {

    x, y, radius float64

}

/\* define a method for circle \*/

func (circle Circle) area() float64 {

    return math.Pi \* circle.radius \* circle.radius

}

func main() {

    circle := Circle{x: 0, y: 0, radius: 5}

    fmt.Printf("Circle area: %f", circle.area())

}

Circle area: 78.539816

**Function in go**

package main

import (

    "fmt"

)

func main() {

    var s string

    fmt.Println("Hello Beautifull world")

    s = passthestring()

    fmt.Println("String from function is", s)

}

func passthestring() string {

    return "Goodbye"

}

Hello Beautifull world

String from function is Goodbye

package main

import (

    "fmt"

)

func main() {

    //var s string

    //var t string

    fmt.Println("Hello Beautifull world")

    s, t := passthestring()

    fmt.Println("String from function is", s, t)

}

func passthestring() (string, string) {

    return "Goodbye", "World"

}

Hello Beautifull world

String from function is Goodbye World

Function as a pointer

package main

import (

    "fmt"

)

func main() {

    var a string

    passString(&a)

    fmt.Println("Value come from function is", a)

}

func passString(b \*string) {

    \*b = "name"

}

Value come from function is name

package main

import (

    "fmt"

)

func main() {

    var x int = 5748

    var p \*int

    p = &x

    fmt.Println(x)

    fmt.Println(&x)

    fmt.Println(p)

 }

5748

0xc0000aa058

0xc0000aa058

Function array

package main

import "fmt"

func main() {

    a := []int{0, 5, 3, 3}

    var j [4]\*int

    for i := 0; i < 4; i++ {

        j[i] = &a[i]

    }

    for i := 0; i < 4; i++ {

        fmt.Println(i, \*j[i])

    }

}

0 0

1 5

2 3

3 3