**Main function:**

package main

import "fmt"

func init(){

fmt.Println("initialization of main function")

}

func main() {

fmt.Println("Program execution begins")

}

**Function argument and return types:**

package main

import "fmt"

func sum(a int,b int) int{

sum\_value := a+b

fmt.Println("sum is:", sum\_value)

return sum\_value

}

func main() {

a := 90

b := 30

sum(a, b)

}

Variation of ‘sum’ function:

func sum(a, b int) int{

sum\_value := a+b

fmt.Println("sum is:", sum\_value)

return sum\_value

}

func sum(a, b int, c string) int{

sum\_value := a+b

fmt.Println("sum is:", sum\_value)

fmt.Println("value of c is:", c)

return sum\_value

}

func sum(a, b int, c string) (int, string){

sum\_value := a+b

fmt.Println("sum is:", sum\_value)

fmt.Println("value of c is:", c)

return sum\_value, c

}

**Calling a function:**

package main

import "fmt"

func sum(a, b int, c string) (int, string){

sum\_value := a+b

return sum\_value, c

}

func main() {

a := 90

b := 30

var c string

result, c := sum(a, b, "Hello Go")

fmt.Println("Values returned from the function are: ", result, c)

}

**Call by value:**

package main

import "fmt"

func call\_by\_value(x int, y int) {

x = x-20

y = y-30

fmt.Printf("In function, value of a : %d\n", x ) //prints 80

fmt.Printf("In function, value of b : %d\n", y ) //prints 170

}

func main() {

var a int = 100

var b int = 200

fmt.Printf("Before calling function, value of a : %d\n", a ) //prints 100

fmt.Printf("Before calling function, value of b : %d\n", b ) //prints 200

call\_by\_value (a, b)

fmt.Printf("After calling function, value of a : %d\n", a ) //prints 100

fmt.Printf("After calling function, value of b : %d\n", b ) //prints 200

}

**Call by reference:**

package main

import "fmt"

func call\_by\_reference(x \*int, y \*int) {

\*x = \*x-20

\*y = \*y-30

}

func main() {

var a int = 100

var b int = 200

fmt.Printf("Before calling function, value of a : %d\n", a ) //prints 100

fmt.Printf("Before calling function, value of b : %d\n", b ) //prints 200

call\_by\_reference(&a, &b)

fmt.Printf("After calling function, value of a : %d\n", a ) //prints 80

fmt.Printf("After calling function, value of b : %d\n", b ) //prints 170

}

**Named return values:**

package main

import "fmt"

func sum(a, b int, c string) ( sum\_value int, c\_val string){

sum\_value = a+b

c\_val = c

return

}

func main() {

a := 90

b := 30

var c string

result, c := sum(a, b, "Hello Go")

fmt.Println("Values returned from the function are: ", result, c)

}

**Blank identifier:**

package main

import "fmt"

func sum(a, b int, c string) ( sum\_value int, c\_val string){

sum\_value = a+b

c\_val = c

return

}

func main() {

a := 90

b := 30

result, \_ := sum(a, b, "Hello Go")

fmt.Println ("Values returned from the function are:", result) // prints “Values returned from the function are: 120”

}

**Function as type:**

package main

import "fmt"

type sub func(a int, b int) int

func main() {

var result sub = func(a int, b int) int {

return a-b

}

fmt.Println("Result is: ", result(100, 70))

}

**Anonymous functions:**

func main() {

a := func(){

fmt.Println("Introducing anonymous function")

}

a()

}

func main() {

func(){

fmt.Println("Introducing anonmyous function")

}()

}

func main() {

a := func() string{

value := "Introducing anonymous function"

return value

}

value := a()

fmt.Println(value)

}

**Return Anonymous functions:**

package main

import "fmt"

func return\_anon() func(string){ //declare type of return value as function

return func(message string){

fmt.Println(message)

}

}

func main() {

return\_anon\_variable := return\_anon()

fmt.Println(return\_anon\_variable) //prints the address of function returned by ‘return\_anon’ function

return\_anon\_variable("Hello")

}

**Closures**

package main

import "fmt"

func return\_anon() func() int{

count := 0

return func() int{

count++;

return count

}

}

func main() {

return\_anon\_variable := return\_anon()

fmt.Println("value returned by 'return\_anon' is:", return\_anon\_variable) //prints ‘ox47da70’

fmt.Println("count is:", return\_anon\_variable()) //prints “count is 1”

fmt.Println("count is:", return\_anon\_variable()) //prints “count is 2”

return\_anon\_variable2 := return\_anon() //new instance

fmt.Println("count is:", return\_anon\_variable2()) //prints “count is 1”

}

**Recursive functions**

package main

import "fmt"

func factorial(num int) int {

if num > 1 {

return num \* factorial(num-1)

} else {

return 1

}

}

func main() {

num := 10

result := factorial(num)

fmt.Println("Factorial result is:", result)

}

**Defer keyword**

package main

import "fmt"

func defer\_fun() {

fmt.Println("Hi I am introducing deferred function")

}

func main() {

fmt.Println("Before defer statement")

defer defer\_fun()

fmt.Println("After defer statement")

}

**Variadic functions**

func add(args ...int) int {

total := 0

for \_, v := range args {

total += v

}

return total

}

func main() {

fmt.Println(add(1, 2, 3))

}