TutorialKart

Golang Functions – Define and Calling Function – Examples

Golang Functions

A function is a set of statements that serve a behavior or functionality. Functions can make your application modular. Golang supports functions. The syntax of user defined Golang function is:

```
func function_name( [parameter_list] ) [return_types] {
    statement(s)
}
```

where

- `func` is the keyword used to define a function.
- `function_name` is the name by which you can call the function.
- `parameter_list` is optional and these are the arguments that you provide to your function to transform them or consider them as input.
- `return_types` is the list of data types of the multiple values that your function can return.

Sometimes, you may have a function that does not return any value. In that case, you may not mention any return type.

Example 1 – Golang Function – Return Single Value

In the following example, we have an add function that takes two arguments, compute their addition and return the result.

example.go

```
package main

import "fmt"

func add(a int, b int) int {
    var c int
    c = a+b
    return c
}
```

```
func main() {
    var n = add(3, 5)
    fmt.Printf("The sum is : %d", n)
}
```

Output

```
The sum is : 8
```

Example 2 – Golang Function – Return Multiple Values

In the following example, we have an calculate function that takes two arguments, computes their addition and multiplication, and returns all the three values.

example.go

```
package main

import "fmt"

func calculate(a int, b int) (int, int) {
    var add int
    var mul int
    add = a+b
    mul = a*b
    return add, mul
}

func main() {
    var add, mul = calculate(3, 5)
    fmt.Printf("Addition : %d\nMultiplication : %d", add, mul)
}
```

Output

```
Addition : 8
Multiplication : 15
```

Example 3 – Golang Function – Return Nothing

In the following example, we have an calculate function that takes two arguments, computes their addition and multiplication, and returns all the three values.

example.go

```
package main

import "fmt"
```

```
func add(a int, b int) {
    fmt.Printf("Addition is : %d", a+b)
}

func main() {
    add(3, 5)
}
```

Output

```
Addition is : 8
```

We are not returning any value from the function. We are just printing the result right inside the function.

Conclusion

In this Golang Tutorial, we learned about Golang Functions and how to return values from a golang function, with the help of examples.

Golang String Operations

- Golang String Length
- Golang String Concatenation
- Golang Split String
- Golang String - Get Index of Substr
- Golang String to Uppercase
- Golang String to Lowercase
- Golang Convert String to Integer