



Containers or Serverless?  
Why not both. With Google Cloud Run  
get 2 million requests free per month.

My Udemy Bestseller Course - Building Real-Time REST APIs with Spring Boot and Deploy on AWS Cloud (Production)

## Go (Golang) Functions with Examples

author: Ramesh Fadatara  
golang



In this tutorial, we will learn how to work functions in Golang with examples.

Go is an open-source programming language that makes it easy to build simple, reliable, and efficient software.



## Go (Golang) Operators with Examples

### Table of Contents

In this tutorial, we will learn the following topics with examples:

- Golang Function - Simple Example
- Go - Function Multiple Return Values
- Go - Anonymous Function
- Go - variadic Function
- Go - Recursive Function
- Go - defer Function Call
- Go - Pass Parameters by Value
- Go - Function as a Parameter
- Go - Custom Function Types

Apple iPhone 14 Plus (128GB, Midnight)	Redmi 10 (6GB RAM, 128GB, Pacific Blue)	Redmi 10 (4GB RAM, 64GB, Caribbean Green)	OnePlus 9 5 (RAM, 256GB Mist)
\$69,999	\$14,999 \$12,499	\$14,099 \$9,499	\$54,099 \$42,99

Let's get started with a defining simple function in Go.

### Golang Function - Simple Example

In this example, we will show you how to create a simple function in Go with an example.

A function is a mapping of zero or more input parameters to zero or more output parameters. Functions in Go are created with the `func` keyword. We use the `return` keyword to return values from functions.

The following example creates a simple function in Go:

```
package main

import "fmt"

func main() {

    x := 4
    y := 5

    z := add(x, y)

    fmt.Printf("Output: %d\n", z)
}

func add(a int, b int) int {

    return a + b
}
```

Output:

Output: 9

In the above example, we define a function that adds two values.

### Go - Function Multiple Return Values

Go functions allow to return of multiple values.

In the example, we have a `threerandom()` function, which returns three random values.

```
package main

import (
    "fmt"
    "math/rand"
    "time"
)

func threerandom() (int, int, int) {

    rand.Seed(time.Now().UnixNano())
    x := rand.Intn(10)
    y := rand.Intn(10)
    z := rand.Intn(10)

    return x, y, z
}

func main() {

    r1, r2, r3 := threerandom()

    fmt.Println(r1, r2, r3)
}
```

Output:

8 8 7

### Go - Anonymous Function

We create an anonymous function that adds three values. We pass three parameters to the function right after its definition.

```
package main

import "fmt"

func main() {

    sum := func(a, b, c int) int {

        return a + b + c
    }
}
```

Spring Boot Thymeleaf Real-Time Web Application Course



Subscriber to my top YouTube Channel (90K+Subscribers)



My Udemy Course - Testing Spring Boot Application with JUnit and Mockito



My Udemy Course - Building Real-Time REST APIs with Spring Boot



My Udemy Course - Master Spring Data JPA with Hibernate



My Udemy Course - Spring Boot RabbitMQ Course - Event-Driven Microservices



My Udemy Course - Spring Boot + Apache Kafka Course



About Me

Hi, I am Ramesh Fadatara. I am VMware Certified Professional For Spring and Spring Boot 2022.

I am founder and author of this blog website [JavaGuides](#), a technical blog dedicated to the Java/JSP EE technologies and Full-Stack Java development.

All the articles, guides, tutorials(2000+) written by me so connect with me if you have any questions/queries. Read more about me at [About Me](#).

**Top YouTube Channel (75K+ Subscribers):** Check out my YouTube Channel for free videos and courses - [Java Guides YouTube Channel](#)

**My Udemy Courses -** <https://www.udemy.com/user/ramesh-fadatara/>

Connect with me on [Twitter](#), [Facebook](#), [LinkedIn](#), [GitHub](#), and [StackOverflow](#)

Apple iPhone 14 Plus (128G

Check out My YouTube Channel with 90K subscribers

Yes please! No, thank!

AddThis



Containers or Serverless?  
Why not both. With Google Cloud Run  
get 2 million requests free per month.



Output:

1+3+5 = 9

## Go - variadic Function

A variadic function can accept a variable number of parameters. For instance, when we want to calculate the sum of values, we might have four, five, six, etc. values to pass to the function.

We use the **...** (**ellipses**) operator to define a variadic function.

In the example, we have a sum function that accepts a variable number of parameters.

```
package main

import "fmt"

func main() {
    s1 := sum(1, 2, 3)
    s2 := sum(1, 2, 3, 4)
    s3 := sum(1, 2, 3, 4, 5)

    fmt.Println(s1, s2, s3)
}

func sum(nums ...int) int {
    res := 0
    for _, n := range nums {
        res += n
    }
    return res
}
```

Output:

6 10 15

The **nums** variable is a slice, which contains all values passed to the sum function. We loop over the slice and calculate the sum of the parameters:

```
func sum(nums ...int) int {
    res := 0
    for _, n := range nums {
        res += n
    }
    return res
}
```

## Go - Recursive Function

A recursive method calls itself to do its task. Recursion is a widely used approach to solve many programming tasks.

In this code example, we calculate the factorial of three numbers:

```
package main

import "fmt"

func fact(n int) int {
    if n == 0 || n == 1 {
        return 1
    }
    return n * fact(n-1)
}

func main() {
    fmt.Println(fact(7))
    fmt.Println(fact(10))
    fmt.Println(fact(15))
}
```

Output:

5040  
3628800  
1307674368000

Inside the body of the fact function, we call the fact function with a modified argument. The function calls itself

```
func fact(n int) int {
    if n == 0 || n == 1 {
        return 1
    }
    return n * fact(n-1)
}
```

## Go - defer Function Call

The defer statement defers the execution of a function until the surrounding function returns. The deferred call's arguments are evaluated immediately, but the function call is not executed until the surrounding function returns.

In the example, the sayHello function is called after the main function finishes.

```
package main

import "fmt"

func main() {
    fmt.Println("begin main")

    defer sayHello()

    fmt.Println("end main")
}

func sayHello() {
    fmt.Println("hello")
}
```

Output:

begin main  
end main  
hello

Apple iPhone 14 Plus (128G)



100%

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Redmi 10 (6GB RAM, 128GB)

Check out My YouTube Channel  
with 90K subscribers

Yes please! No, thanks!

AdMob



**Containers or Serverless?**  
Why not both. With Google Cloud Run  
get 2 million requests free per month.



In the following example, an **integer** and a **User** structure are passed as parameters to functions.

```
package main

import "fmt"

type User struct {
    name      string
    occupation string
}

func main() {
    x := 10
    fmt.Printf("Inside main %d\n", x)

    inc(x)

    fmt.Printf("Inside main %d\n", x)

    fmt.Println("-----")

    u := User{"Raj", "Engineer"}
    fmt.Printf("Inside main %v\n", u)

    change(u)
    fmt.Printf("Inside main %v\n", u)
}

func inc(x int) {
    x++
    fmt.Printf("Inside inc %d\n", x)
}

func change(u User) {
    u.occupation = "driver"
    fmt.Printf("Inside change %v\n", u)
}
```

Output:

```
Inside main 10
Inside inc 11
Inside main 10
-----
Inside main (Raj Engineer)
Inside change (Raj driver)
Inside main (Raj Engineer)
```

In the above example, the original values of the **x** and **User** struct are not modified.

A copy of the integer value is created. Inside the function, we increment the value of this copy. So the original variable is intact.

```
func inc(x int) {
    x++
    fmt.Printf("Inside inc %d\n", x)
}
```

## Go - Function as a Parameter

A Go function can be passed to other functions as a parameter. Such a function is called a higher-order function.

In the example, the apply function takes the **inc()** and **dec()** functions as parameters.

```
package main

import "fmt"

func inc(x int) int {
    x++
    return x
}

func dec(x int) int {
    x--
    return x
}

func apply(x int, f func(int) int) int {
    r := f(x)
    return r
}

func main() {
    r1 := apply(5, inc)
    r2 := apply(4, dec)
    fmt.Println(r1)
    fmt.Println(r2)
}
```

Output:

```
6
3
```

## Go - Custom Function Types

Go allows the creation of reusable function signatures with the **type** keyword. In simple words, Golang also supports defining our own function types.

In this example, we use the **type** keyword to create a function type that accepts one string parameter and returns a string.

```
package main

import "fmt"

type output func(string) string

func hello(name string) string {
    return fmt.Sprintf("hello %s", name)
}

func main() {
    var f output

    f = hello
    fmt.Println(f("Raj"))
}
```

Output:

```
hello Raj
```

## Golang Related Tutorials

- [Go \(Golang\) Functions with Examples](#)
- [Go \(Golang\) Operators with Examples](#)
- [Go \(Golang\) - Read Input from User or Console](#)

✕

Check out My YouTube Channel  
with 90K subscribers

Yes please! No, thanks!

AdMob



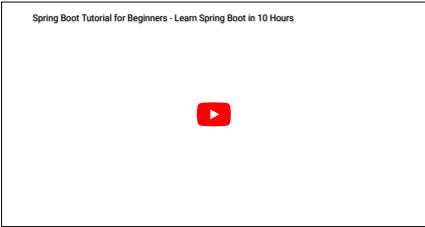
Containers or Serverless?  
Why not both. With Google Cloud Run  
get 2 million requests free per month.

Go (Golang) Structs Tutorial with Examples

golang

## Free Spring Boot Tutorial | Full In-depth Course | Learn Spring Boot in 10 Hours

Watch this course on YouTube at [Spring Boot Tutorial](#) | [Free 10 Hours Full Course](#)



Apple iPhone 14 Plus  
(128GB, Midnight)

Ad Demo

GOLANG



To leave a comment, click the button below to sign in with Google.

SIGN IN WITH GOOGLE

100% 100% 100%

Apple iPhone 14 Plus (128GB, Midnight) SAMSUNG Galaxy F42 5G (8GB RAM, 128GB, Matte Black) Redmi 10 (4GB RAM, Redmi 10 (6GB RAM, 64GB, Pacific Blue) 128GB, Pacific Blue) Apple iPhone 13 (128GB, Alpine Green) Apple (128GB)

Free Courses on YouTube

- Spring Boot Tutorial
- Java Collections Framework
- Spring Boot AWS Deployment
- Spring MVC Tutorial Course
- 5 Spring Boot Projects in 10 Hours
- Spring Boot Restful Web Services Tutorial
- Event-Driven Microservices using Spring Boot and Kafka
- Spring Boot Kafka Real-World Project Tutorial
- Spring Boot + Apache Kafka Course
- Angular + Spring Boot CRUD Full Stack
- ReactJS + Spring Boot CRUD Full Stack
- Spring Boot Thymeleaf Full Stack

My Udemy Courses

- Building Real-Time REST APIs with Spring Boot
- Testing Spring Boot Application with JUnit and Mockito
- Master Spring Data JPA with Hibernate
- Spring Boot + Apache Kafka - The Quickest Practical Guide
- Spring Boot + RabbitMQ (Includes Event-Driven Microservices)
- Spring Boot Thymeleaf Real-Time Web Application - Blog App

Connect

- YouTube
- Twitter
- Facebook
- GitHub
- LinkedIn
- StackOverflow

Dev Tools

- JSON Formatter | Beautifier
- Online HTML Editor and Compiler
- Base64 Encode Online
- Base64 Decode Online
- URL Encoder Online
- URL Decoder Online

Copyright © 2018 - 2022 Java Guides All rights reserved | Privacy Policy | Contact | About Me | YouTube | GitHub

Powered by Blogger

Check out My YouTube Channel  
with 90K subscribers

Yes please! No, thank!

AdMob