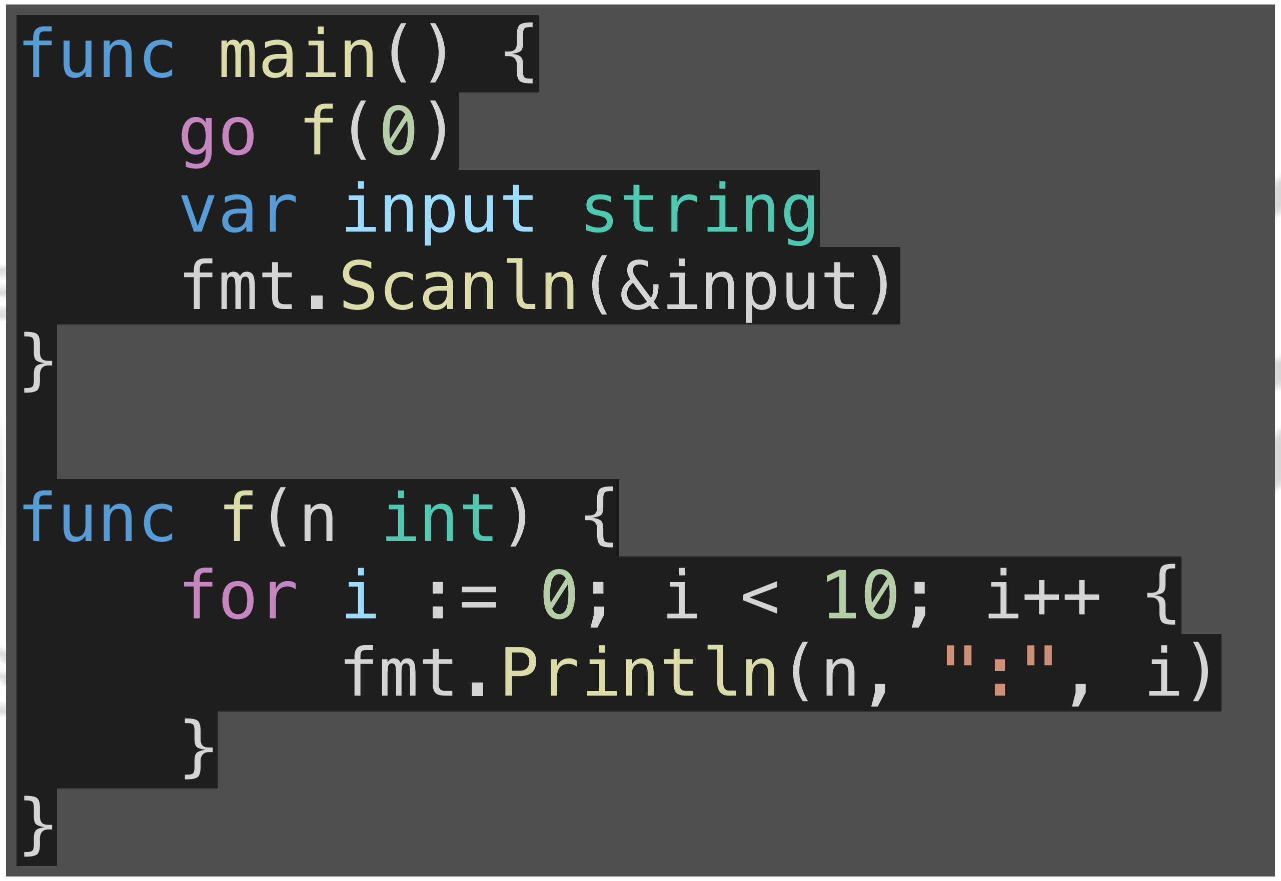
Goroutine

golang : goroutine

create main.go in folder chapter13-1 :



```
func main() {
    go f(0)
    var input string
    fmt.Scanln(&input)
}

func f(n int) {
    for i := 0; i < 10; i++ {
        fmt.Println(n, ":", i)
    }
}
```

run -> no error -> push to your git repository

golang : goroutine

create main.go in folder chapter13-2 :

```go
func main() {
    for i:= 0; i < 10; i++ {
        go f(i)
    }
    var input string
    fmt.Scanln(&input)
}

func f(n int) {
    for i := 0; i < 10; i++ {
        fmt.Println(n, ":", i)
    }
}
```

run -> no error -> push to your git repository

golang : goroutine

create main.go in folder chapter13-3 :

```
func main() {
    runtime.GOMAXPROCS(8)

    for i := 0; i < 10; i++ {
        go f(i)
    }
    var input string
    fmt.Scanln(&input)
}

func f(n int) {
    for i := 0; i < 10; i++ {
        fmt.Println(n, ":", i)
    }
}
```

run -> no error -> push to your git repository

golang : goroutine

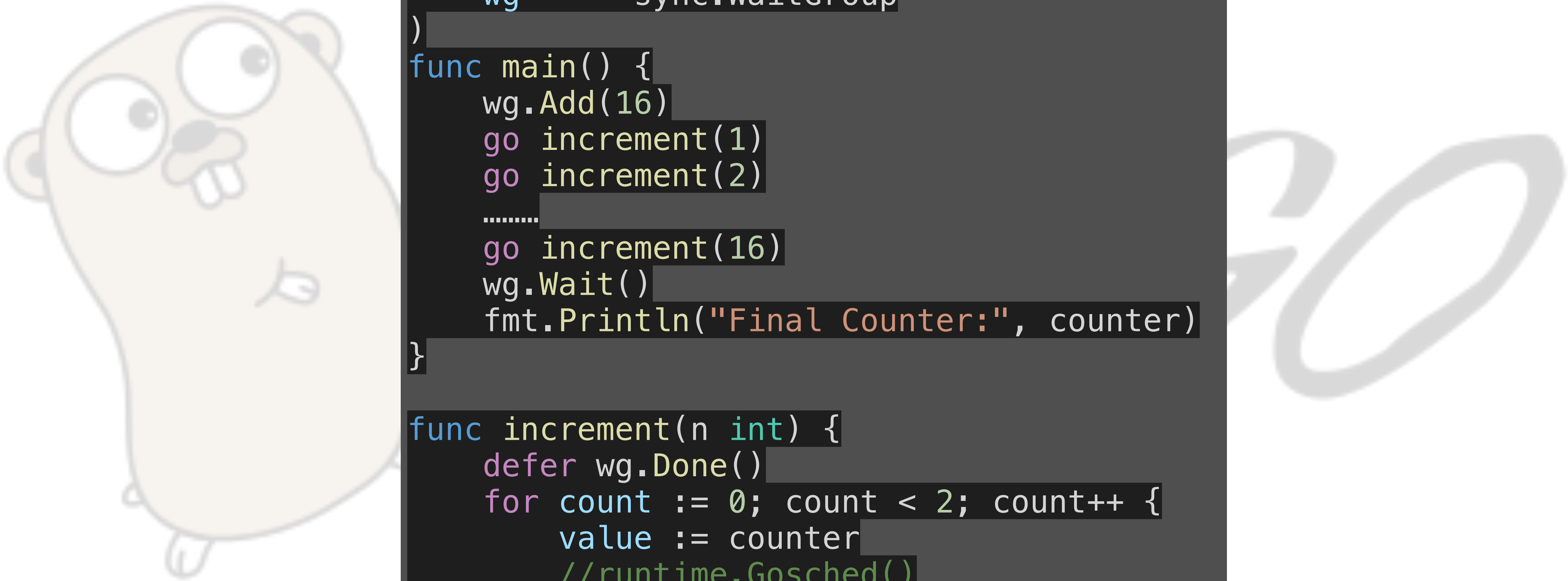
create main.go in folder chapter13-4 :

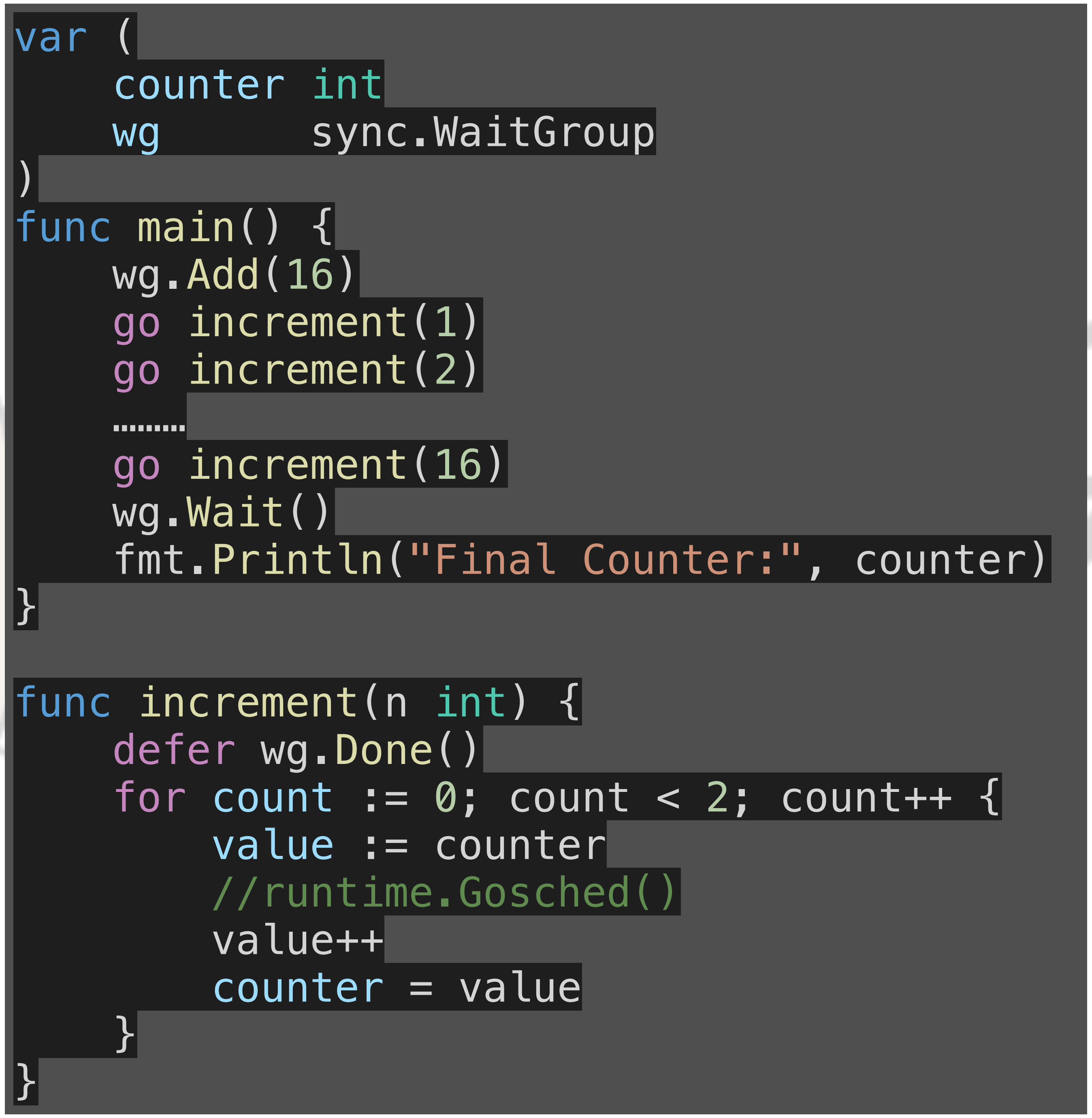
```go
func main() {
    var wg sync.WaitGroup
    wg.Add(2)

    for i := 0; i < 2; i++ {
        go func(n int) {
            defer wg.Done()
            for i := 0; i < 10; i++ {
                fmt.Println(n, ":", i)
            }
        }(i)
    }
    wg.Wait()
    fmt.Println("Finished")
}
```

run -> no error -> push to your git repository

golang : goroutine

create main.go in folder chapter13-5 :





```go
var (
    counter int
    wg      sync.WaitGroup
)
func main() {
    wg.Add(16)
    go increment(1)
    go increment(2)
    .........
    go increment(16)
    wg.Wait()
    fmt.Println("Final Counter:", counter)
}

func increment(n int) {
    defer wg.Done()
    for count := 0; count < 2; count++ {
        value := counter
        //runtime.Gosched()
        value++
        counter = value
    }
}
```

run -> no error -> push to your git repository

golang : Atomic

create main.go in folder chapter13-6 :

```go
var (
    counter int64
    wg      sync.WaitGroup
)

func main() {
    wg.Add(16)

    go increment(1)
    go increment(2)
    ...........
    go increment(16)

    wg.Wait()
    fmt.Println("Final Counter:", counter)
}

func increment(n int) {
    defer wg.Done()
    for count := 0; count < 2; count++ {
        atomic.AddInt64(&counter, 1)
    }
}
```

run -> no error -> push to your git repository

golang : Mutex

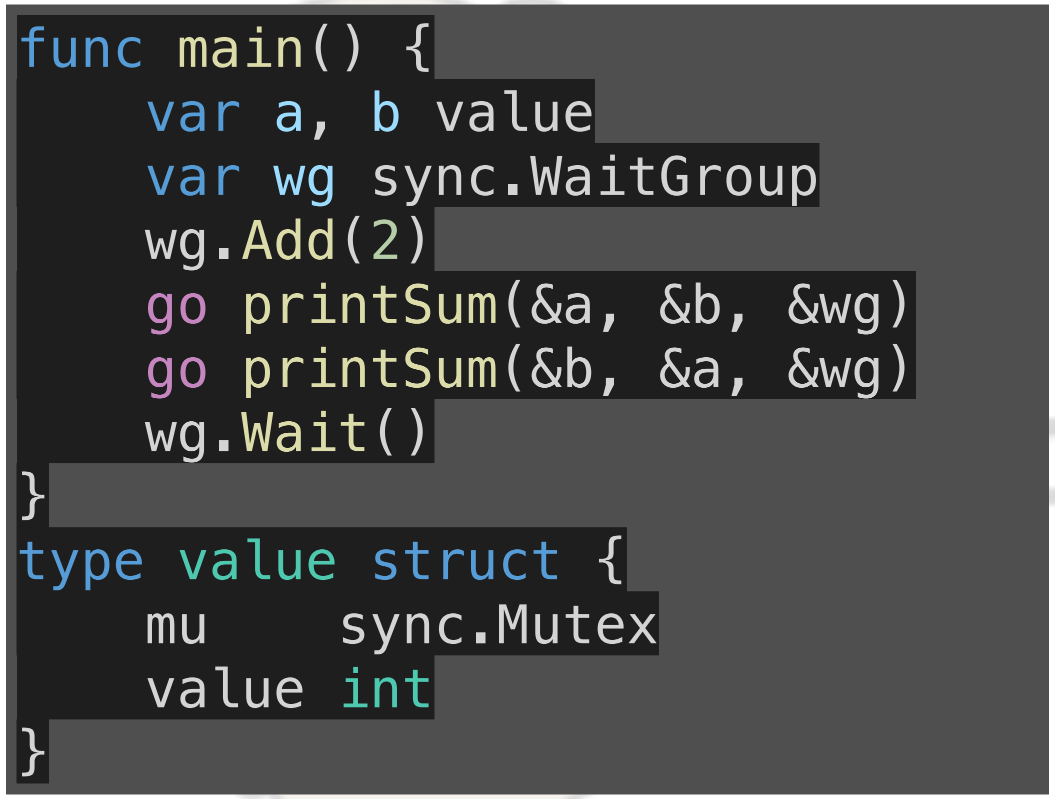
create main.go in folder chapter13-7 :

```go
var (
    counter int64
    wg      sync.WaitGroup
    mu sync.Mutex
)
func main() {
     wgnum := 16
    wg.Add(wgnum)
    for i := 1; i <= wgnum; i++ {
        go increment(i)
    }
    wg.Wait()
    fmt.Println("Final Counter:", counter)
}
func increment(n int) {
    defer wg.Done()
    mu.Lock()
    for count := 0; count < 2; count++ {
        atomic.AddInt64(&counter, 1)
    }
    mu.Unlock()
}
```

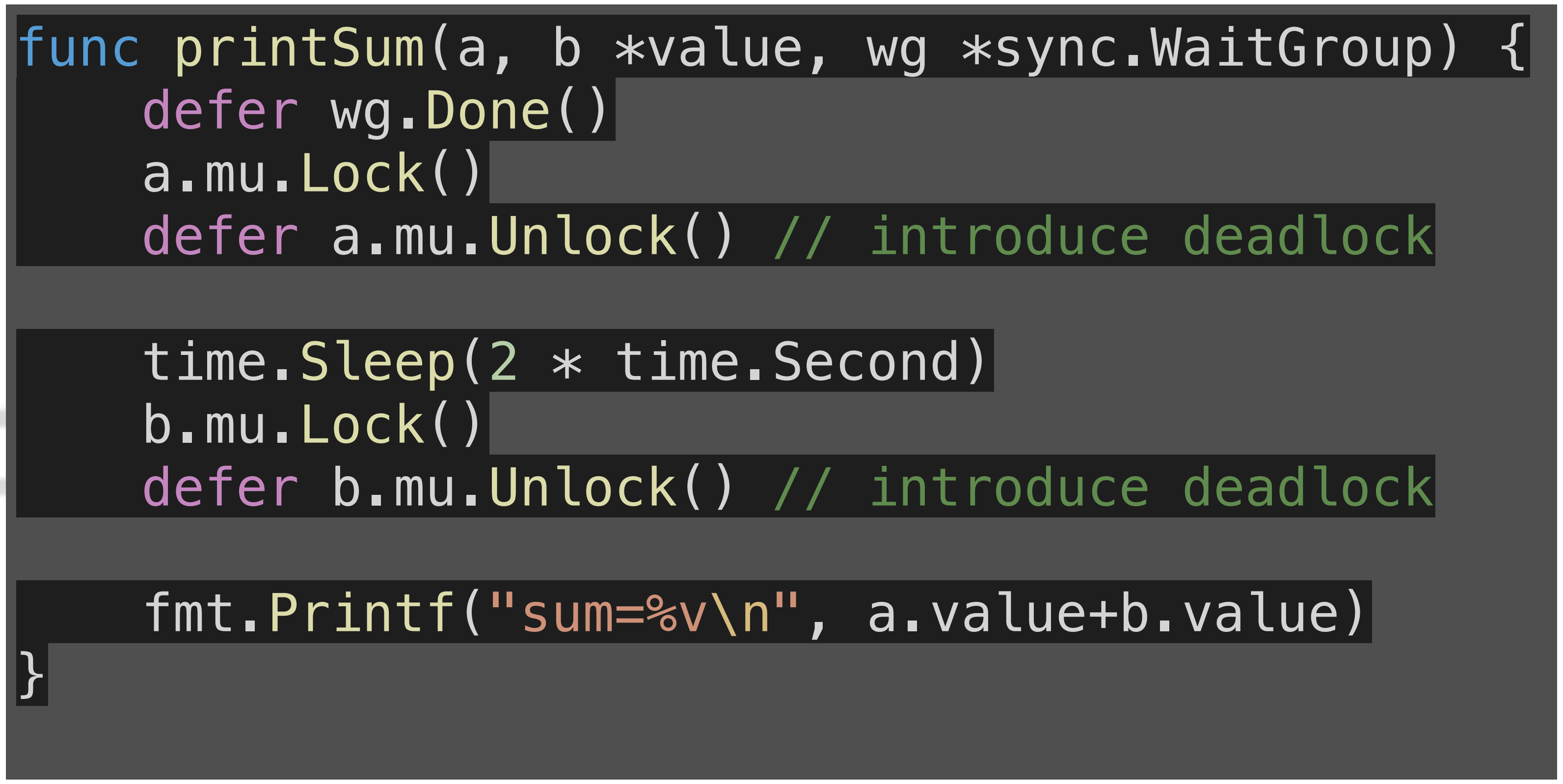
run -> no error -> push to your git repository

golang : Deadlock

create main.go in folder chapter13-8 :



```go
func main() {
	var a, b value
	var wg sync.WaitGroup
	wg.Add(2)
	go printSum(&a, &b, &wg)
	go printSum(&b, &a, &wg)
	wg.Wait()
}
type value struct {
	mu    sync.Mutex
	value int
}
```



```go
func printSum(a, b *value, wg *sync.WaitGroup) {
	defer wg.Done()
	a.mu.Lock()
	defer a.mu.Unlock() // introduce deadlock

	time.Sleep(2 * time.Second)
	b.mu.Lock()
	defer b.mu.Unlock() // introduce deadlock

	fmt.Printf("sum=%v\n", a.value+b.value)
}
```

run -> no error -> push to your git repository