My code:

package main

import (

    "fmt"

    "time"

)

func getNumber() {

    var i int = 0

    go func() {

        i = 22

    }()

    go func() {

        i = 44

    }()

    time.Sleep(time.Second)

    fmt.Println(i)

}

func main() {

    getNumber()

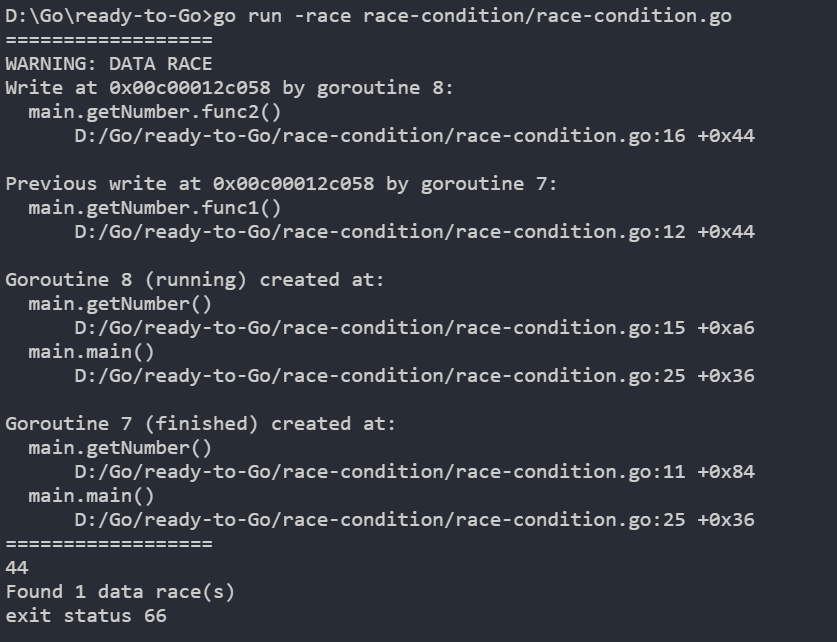
}

A race condition is when two or more device/system have access to the same resource, such as a variable or data structure and attempt to read/write to that resource without any regard to the other routines.

We can see that the getNumber function is change the value of i in 2 differents goroutines. Then we print i without any knowledge of whether which goroutine has completed first. So now, there are 2 operations that are taking place:

1. The value of i is being set to 22
2. The value of i is being set to 44

We can see that go has found 1 data race in our code when execute it:



Different result when executing:

