In this example, we have a global variable counter that is being accessed and modified by two goroutines concurrently using the increment function. The sync.WaitGroup is used to wait for both goroutines to finish before printing the final value of counter.

A race condition can occur in this program because the counter variable is not synchronized or protected by a mutex. When both goroutines try to increment the counter variable concurrently, there is a possibility of a data race - where the final value of counter may not be as expected. The outcome of the program is non-deterministic because the order in which the goroutines access and modify the counter variable is not guaranteed.