Solution Review: Demonstrate the Blocking Nature

This lesson discusses the solution to the challenge given in the previous lesson.

```go
package main
import "fmt"
import "time"

func main() {
	c := make(chan int)
	go func() {
		time.Sleep(5 * 1e9)
		x := <- c // value recieved from channel
		fmt.Println("received", x)
	}()
	fmt.Println("sending", 10)
	c <- 10 // putting 10 on the channel
	fmt.Println("sent", 10)
}
```

Blocking Channel

At **line 6**, we make a channel `c` for integers. Then, from **line** 7 to **line 11**, we start a goroutine with an anonymous function that we call at **line 11**. `Main()` continues immediately. It prints **sending** and puts the integer **10** on the channel. Then `main()` waits until the value is taken from the channel. This is done in the goroutine at **line 9**, but only after 5s have passed. Then, the goroutine prints **received**, and `main()` prints **sent**.

That is it for the solution. In the next lesson, we'll discuss the semaphore pattern, which was mentioned earlier.