Range and close

This lesson explains the use of range and how it can be used in closing a channel in Go

WE'LL COVER THE FOLLOWING

Closing a channel

A sender can close a channel to indicate that no more values will be sent. Receivers can test whether a channel has been closed by assigning a second parameter to the receive expression: after

```
v, ok := <-ch
```

`ok` is false if there are no more values to receive and the channel is closed.

The loop `for i := range ch` receives values from the channel repeatedly until it is closed.

> **Note:** Only the sender should close a channel, never the receiver. Sending on a closed channel will cause a panic.

> **Another note:** Channels aren't like files; you don't usually need to close them. Closing is only necessary when the receiver must be told there are no more values coming, such as to terminate a range loop.

Example

Environment Variables

Key:	Value:
GOPATH	/go

```
package main

import (
    "fmt"
)

func fibonacci(n int, c chan int) {
    x, y := 0, 1
    for i := 0; i < n; i++ {
        c <- x
        x, y = y, x+y
    }
    close(c)
}

func main() {
    c := make(chan int, 10)
    go fibonacci(cap(c), c)
    for i := range c {
        fmt.Println(i)
    }
}
```

As you can see in above example, the code only outputs **10** values since the value of `c` passed in line **18** is **10**. And you know that the **forloop** will only receive values from channel till it closes, which is after the value of `c` is reached in this case **10**.

In the next lesson we will discuss the concept of *select* in Go.