Range and Close

This lesson will teach you a way to close a channel.

If you remember the lesson on channels, you are already familiar with this pattern.

In Go, we have a `range` function which lets us iterate over elements in different data structures. Using this function, we can *range* over the items we receive on a channel until it is closed. Also, note that only the sender, not the receiver, should close the channel when it feels that it has no more values to send.

Let's have a look at an example:

```
package main
import "fmt"

type Money struct{
  amount int
  year int
}

func sendMoney(parent chan Money){

  for i:=0; i<=18; i++ {
    parent <- Money{5000,i}
  }
  close(parent)
}

func main() {
  money := make(chan Money)

  go sendMoney(money)

  for kidMoney:= range money {
    fmt.Printf("Money received by kid in year %d : %d\n", kidMoney.year, kidMoney.amount)
  }
}
```

The above code is an over-simplification of the money transfer that happens between parents and their kids. So your parents keep sending you money until you hit the age of 18 and that's when they close the channel.

The for-loop `kidMoney:= range money` receives data from the `money` channel until the `parent` channel closes itself in the `sendMoney` function on **line 14**. `close(parent)` sends a signal to the `range` loop to terminate.

> Note: If you are sending on a closed channel, it will cause a panic.

So that was easy peasy, right? Now, let's move on to the next lesson and learn about the for-select loop.