Module 4: Synchronized Communication
Topic 1.1: Blocking on Channels

Iterating Through a Channel

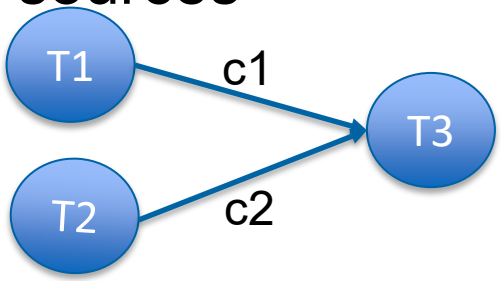
- Common to iteratively read from a channel

```
for i := range c {
    fmt.Println(i)
}
```

- Continues to read from channel c
- One iteration each time a new value is received
- i is assigned to the read value
- Iterates when sender calls `close(c)`

Receiving from Multiple Goroutines

- Multiple channels may be used to receive from multiple sources



- Data from both sources may be needed
- Read sequentially

```
a := <- c1
b := <- c2
fmt.Println(a*b)
```

Select Statement

- May have a choice of which data to use
  - i.e. First-come first-served
- Use the **`select`** statement to wait on the first data from a set of channels

```
select {
    case a = <- c1:
        fmt.Println(a)
    case b = <- c2:
        fnt.Println(b)
}
```