The For-Select Loop

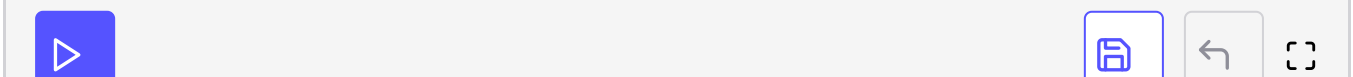
Let's see how we can combine the `for` and `select` statements in order to communicate over channels in Go.

The for-select loop can be structured as follows:

```
for { //either range over a channel or loop infinitely
  select {
    //handle channel operations
  }
}
```

First, let's try to loop over something that goes on infinitely.

```
package main
import "fmt"
func getNews(newsChannel chan string){
  NewsArray := [] string {"Roger Federer wins the Wimbledon","Space Exploration has reached r
  for _, news := range NewsArray{
    newsChannel <- news
  }

  newsChannel <- "Done"
  close(newsChannel)

}

func main() {
  myNewsChannel := make(chan string)

  go getNews(myNewsChannel)

  for {
    select{
      case news := <-myNewsChannel:
        fmt.Println(news)
        if news == "Done"{
          return
        }
      default:
    }

  }
}
```



You might have wondered why the loop did not run infinitely. Well, that's because I checked for my `"Done"` signal and returned from the main routine in **lines 23-25** as to prevent an infinite loop from running.

Now the for-loop below enables us to execute the select statement infinitely. This is pretty handy in situations where we have to keep taking inputs from different channels.

```
for {
    select{
      case news := <-myNewsChannel:
        fmt.Println(news)
        if news == "Done" {
          return
        }
      default:

    }

  }
```

In the code above, I just have three headlines but obviously, the headlines never stop and you want to keep looking for them.

Let's look at another situation:

```
package main
import "fmt"

func main() {
  done := make(chan string)

  for _, fruit := range []string{"apple", "banana", "cherry"}{
    select{
      case <-done:
        return
      default:
       fmt.Println(fruit)

    }
  }

}
```

In the above code, we *range* over an array `fruits` and have a dummy channel case written in the select statement. The default case prints out the elements of the array. Hence, this technique can be used if we already know how many times we have to run our select statement and also if we need the iterative values in our code.

That was all regarding the `for-select` loop. We'll move on to the next lesson to learn about quit channels!