Goroutines

This lesson introduces the concept of goroutines, their definition, and gives an example.

Definition

A goroutine is a lightweight thread managed by the Go runtime. Goroutines can be functions or methods that run concurrently with other functions or methods

```
go f(x, y, z)
```

starts a new goroutine running:

```
f(x, y, z)
```

The evaluation of `f`, `x`, `y`, and `z` happens in the current goroutine, and the execution of `f` happens in the new goroutine.

Goroutines run in the same address space, so access to shared memory must be **synchronized**. The sync package provides useful primitives, although you won't need them much in Go as there are other primitives.

Example

Environment Variables

Key:	Value:

GOPATH	/go

```
package main

import (
        "fmt"
        "time"
)

func say(s string) {
        for i := 0; i < 5; i++ {
                time.Sleep(100 * time.Millisecond)
                fmt.Println(s)
        }
}

func main() {
  go say("world") //parallel execution: both calls to say() will execute concurrently
        say("hello")
}
```

The example above is a good representation of how *goroutine* works. As you can see from the output, "world" isn't printed first even though line number **16** is written first in the code; this is because it executes in parallel to the next line which will print "hello".

In the next lesson, we will discuss *channels* used in Go. Keep on reading to find out more!