Golang Session

- Harsh Dusane

Topic : Deferred Functions Calls

Deferred Functions Calls

- Go has a special statement called defer that schedules a function call to be run after the function completes. Consider the following example:

```
package main
import "fmt"
func first() {
        fmt.Println("First")
}
func second() {
        fmt.Println("Second")
}
func main() {
        defer second()
        first()
}
```

Without defer

```
func ReadWrite() bool {
        file.Open("file")
        if failureX {
        file.Close()   //And here...
        return false
        }
        if failureY {
        file.Close()  //And here...
        return false
        }
        file.Close()  //And here...
        return true
}
```

With defer

```
func ReadWrite() bool {
file.Open("file")
defer file.Close()   //file.Close() is added to defer list
// Do your thing
if failureX {
return false   // Close() is now done automatically
}
if failureY {
return false   // And here too
}
return true   // And here
}
```

LIFO Order of defer functions

- It keeps our Close call near our Open call so it's easier to understand.
- If our function had multiple return statements (perhaps one in an if and one in an else), Close will happen before both of them.
- Deferred Functions are run even if a runtime panic occurs.
- Deferred functions are executed in LIFO order, so the above code prints: 4 3 2 1 0.
- You can put multiple functions on the "deferred list", like this example.

```
package main
import "fmt"
func main() {
        for i := 0; i < 5; i++ {
                defer fmt.Printf("%d ", i)
        }
}
```