Golang session

- Harsh Dusane

Topic : Functions

Creating a function

```
package main
import "fmt"
// SimpleFunction prints a message
func SimpleFunction() {
        fmt.Println("Hello World")
}
func main() {
        SimpleFunction()
}
```

Function with Parameters

```
package main
import "fmt"
// Function accepting arguments
func add(x int, y int) {
        total := 0
        total = x + y
        fmt.Println(total)
}
func main() {
        // Passing arguments
        add(20, 30)
}
```

Function with Return Type

```
package main
import "fmt"
// Function with int as return type
func add(x int, y int) int {
        total := 0
        total = x + y
        return total
}
func main() {
        // Accepting return value in varaible
        sum := add(20, 30)
        fmt.Println(sum)
}
```

Named Return Values

```
package main
import "fmt"
func rectangle(l int, b int) (area int) {
        var parameter int
        parameter = 2 * (l + b)
        fmt.Println("Parameter: ", parameter)
        area = l * b
        return // Return statement without specify variable name
}
func main() {
        fmt.Println("Area: ", rectangle(20, 30))
}
```

Returning Multiple Values

```
package main
import "fmt"
func rectangle(l int, b int) (area int, parameter int) {
        parameter = 2 * (l + b)
        area = l * b
        return // Return statement without specify variable name
}
func main() {
        var a, p int
        a, p = rectangle(20, 30)
        fmt.Println("Area:", a)
        fmt.Println("Parameter:", p)
}
```

Passing Address to a Function

```
package main
import "fmt"
func update(a *int, t *string) {
        *a = *a + 5      // defrencing pointer address
        *t = *t + " Doe" // defrencing pointer address
        return
}
func main() {
        var age = 20
        var text = "John"
        fmt.Println("Before:", text, age)
        update(&age, &text)
        fmt.Println("After :", text, age)
}
```

Anonymous Functions

```
package main
import "fmt"
var (
        area = func(l int, b int) int {
                return l * b
        }
)
func main() {
        fmt.Println(area(20, 30))
}
```

```
package main
import "fmt"
func main() {
        func(l int, b int) {
                fmt.Println(l * b)
        }(20, 30)
}
```

```
package main
import "fmt"
func main() {
        fmt.Printf(
                "100 (°F) = %.2f (°C)\n",
                func(f float64) float64 {
                        return (f - 32.0) * (5.0 / 9.0)
                }(100),
        )
}
```

Closure Functions

```
package main
import "fmt"
func main() {
        l := 20
        b := 30
        func() {
                var area int
                area = l * b
                fmt.Println(area)
        }()
}
```

```
package main
import "fmt"
func main() {
        for i := 10.0; i < 100; i += 10.0 {
                rad := func() float64 {
                        return i * 39.370
                }()
                fmt.Printf("%.2f Meter = %.2f Inch\n", i, rad)
        }
}
```

Higher Order Functions

```
package main
import "fmt"
func sum(x, y int) int {
        return x + y
}
func partialSum(x int) func(int) int {
        return func(y int) int {
                return sum(x, y)
        }
}
func main() {
        partial := partialSum(3)
        fmt.Println(partial(7))
}
```

```
package main
import "fmt"
func squareSum(x int) func(int) func(int) int {
        return func(y int) func(int) int {
                return func(z int) int {
                        return x*x + y*y + z*z
                }
        }
}
func main() {
        // 5*5 + 6*6 + 7*7
        fmt.Println(squareSum(5)(6)(7))
}
```