Go Training

Session 2

First Program

- Every go program must be included in a package
- Standalone executable
  - Must have ‘main function’
  - Must be included in the ‘main package’
- The main function is the entry point of the execution
- fmt package has been imported from GO standard library
- How to import a package?

hello.go

```go
package main

import "fmt"

func main() {

    // prints Hello World
    fmt.Println("Hello World")
}
```

Variables and Constants

```go
package main

import "fmt"

var x int

func main() {
    var y bool
    fmt.Println(x,y)
}
```

```go
package main

import "fmt"

var x,z int

func main() {
    var y bool
    fmt.Println(x,y)
}
```

```go
package main

import "fmt"

var (x int
    z bool)

func main() {
    var y bool
    fmt.Println(x,y,z)
}
```

```go
package main

import "fmt"

var x,z int = 1, 5

func main() {
    var y bool = true
    fmt.Println(x,y,z)
}
```

```go
package main

import "fmt"

var x,z = 1, 5

func main() {
    var y = true
    fmt.Println(x,y,z)
}
```

```go
package main

import "fmt"

var (x = 1
    z = 5)

func main() {
    var y = true
    fmt.Println(x,y,z)
}
```

Variables and Constants

```go
package main

import "fmt"

var x = 1
// x := 1 would give an error

func main() {
    y := true
    fmt.Println(x,y)
}
```

```go
package main

import "fmt"

var x = 1
// x := 1 would give an error

func main() {
    y, z, w := true, 5, "hello"
    fmt.Println(x,y, z, w)
}
```

```go
package main

import "fmt"

const x = 1

func main() {
    const y, z, w := true, 5, "hello"
    y = false  //would give an error
    fmt.Println(x,y, z, w)
}
```

```go
package main

import "fmt"

const x
x = 1  // would give an error

func main() {
    const y, z, w = true, 5, "hello"
    fmt.Println(x,y, z, w)
}
```

Basic Data types

type	use
bool	Boolean data type. It can store value `true` or `false`.
string	String data type. It can store UTF-8 string. All strings in go are UTF-8 by default. Unlike JavaScript, strin
int	Integer data type. It can store 32-bit or 64-bit signed integer. A 32-bit system will allocate 32 bits of mem
uint	Integer data type. Same as `int`, `uint` can store 32 bits or 64 bits **unsigned** integer.
int8	Integer data type. System will allocate 8 bits of memory to store an integer. Hence it can store values be
uint8	Integer data type. Same as `int8`, `uint8` can store 8-bit **unsigned** integer. Hence it can store values
int16	Integer data type. System will allocate 16 bits of memory to store an integer. Hence it can store values b
uint16	Integer data type. Same as `int8`, `uint8` can store 16-bit **unsigned** integer. Hence it can store values
int32	Integer data type. System will allocate 32 bits of memory to store an integer. Hence it can store values b
uint32	Integer data type. Same as `int8`, `uint8` can store 32-bit **unsigned** integer. Hence it can store values
int64	Integer data type. System will allocate 64 bits of memory to store an integer. Hence it can store values b
uint64	Integer data type. Same as `int8`, `uint8` can store 64-bit **unsigned** integer. Hence it can store values
uintptr	Integer data type. It is an integer type that is large enough to hold the bit pattern of any pointer. Howeve
float32	Float data type. System will allocate 32 bits of memory to store a float value. Hence it can store values b
float64	Float data type. System will allocate 64 bits of memory to store a float value. Hence it can store values b
complex64	Go supports complex numbers out of this box. `complex64` has `float32` real part and `float32` imaginar
complex128	Similar to `complex64`, `complex128` has `float64` real part and `float64` imaginary part.
byte	Alias for `uint8`.
rune	Alias for `int32`. It represents a Unicode code point.
error	Error data type. It is be used to store error value.

Function

```go
package main

import "fmt"

func mult(x int, y float32) int {
    return x * int(y)
}

func main() {
    fmt.Println(mult(2, 3.5))
}

```

Function

```go
package main

import "fmt"

func mult(x int ,y int) int {
    return x * int(y)
}

func main() {
    fmt.Println(mult(2, 3))
}

```

```go
package main

import "fmt"

func mult(x,y int) int {
    return x * int(y)
}

func main() {
    fmt.Println(mult(2, 3))
}

```

Function

```go
package main

import "fmt"

func sumMult(x, y int) (int, int) {
    return x+y, x*y
}

func main() {
    a, b := sumMult(3, 7)
    fmt.Println(a, b)
}

```

```go
package main

import "fmt"

func sumMult(x, y int) (a,b int) {
    a = x+y
    b = x*y
    return
}

func main() {
    p, q := sumMult(3, 7)
    fmt.Println(p, q)
}

```

Assignment

1. Self study: all the 17 pages from https://tour.golang.org/basics/1
2. Write a go program which has the function definition ‘myFunc’.

   myFunc will take an integer as an argument and will return its square value and a Boolean ‘yes’ if the given integer is odd else a Boolean ‘no’ .

   Call this myFunc three times in your program with arguments 0, -3 and 8 at a time.

   Print the results in console.

   You should run this program on your local machine as well as on https://play.golang.org/.

   Submit the go playground share link and a screenshot of output from your local machine.

   Helpful resource: https://golang.org/pkg/math/#pkg-examples

Thank You