Anatomy of [Golang] Made Simple

By John Melody

1. Hello World Programme:

```
// the name of the package as "Java-like"
package main
// the Module used import like Python
import "fmt"
// Main function as in C/C++
func main() {
// Prints String , no semi colon required
    fmt.Println("Hello, World")
}

// output :

// Hello, World
// Program exited.
```

2. Variable:

   a. Specific:

> var 'name' 'type' = 'value'

```
func main() {
    var i int = 12
    fmt.Println(i)
}
```

   b. Automatic:

> 'name' := 'value'

```
func main() {
    i := 12
    fmt.Println(i)
}
```

   Both of Specific declared and Automatically declared variable gave the same output value of `i` which equals to `12`.

   Cluster `var` declaration can be made by simply place all values in a `'var()'`.

```
var (
    name string = "This is a Name"
    number int = 12
)

func main() {
    fmt.Printf("%v, %s", number, name)
}

// output:
// 12, This is a Name
// Program exited.
```

3. Printing:
   The most Common printing in Golang will be `%v, %T, %t, %s` which stands for `value, boolean, type, string`. As sample programme below explains:

```
func main() {
    x := 12
    y := "Hello"
    z := false
fmt.Printf("%v, %T, %t", x, y, z)

// output:
// 12, string, false
// Program exited.
}
```

4. Type Conversion:
   a. `int/int64` to `string`

```
 package main

 import (
     "fmt"
     "strconv"
 )

 var (
     a int    = 12
     s string = strconv.Itoa(a)
 )

 func main() {
 fmt.Printf("%s, %T", s, s)
 }

 // output is ` 12, string `.
```

b. string1 to int/int64

```
package main

import (
    "fmt"
    "strconv"
)

var (
    s string = "23"
)

func main() {
    if n, err := strconv.Atoi(s); err == nil {
        fmt.Println("The value is ", n)
        fmt.Printf("%T", s)
    } else {
        fmt.Println(s, " is not an integer.")
    }
}

// output
// The value is  23
//string
```