Go Basics

Packages, Variables, and Functions

Based on "A Tour of Go"

Packages

Every Go program is made up of packages. Programs start running in package main.

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    fmt.Println("My favorite number is", rand.Intn(10))
}
```

Imports

This code groups the imports into a parenthesized, "factored" import statement.

```
import (
    "fmt"
    "math"
)
```

Functions

A function can take zero or more arguments. Notice that the type comes after the variable name.

```
func add(x int, y int) int {
    return x + y
}

func main() {
    fmt.Println(add(42, 13))
}
```

Multiple Results

A function can return any number of results. Here, the swap function returns two strings.

```
func swap(x, y string) (string, string) {
    return y, x
}

func main() {
    a, b := swap("hello", "world")
    fmt.Println(a, b)
}
```

Variables

The var statement declares a list of variables; as in function argument lists, the type is last.

```
var c, python, java bool

func main() {
    var i int
    fmt.Println(i, c, python, java)
}
```

Variables with Initializers

If an initializer is present, the type can be omitted; the variable will take the type of the initializer.

```
var i, j int = 1, 2

func main() {
    var c, python, java = true, false, "no!"
    fmt.Println(i, j, c, python, java)
}
```

Short Variable Declarations

Inside a function, the := short assignment statement can be used in place of a var declaration with implicit type.

```
func main() {
    var i, j int = 1, 2
    k := 3
    c, python, java := true, false, "no!"
    fmt.Println(i, j, k, c, python, java)
}
```

Basic Types

Go's basic types include:

bool
string
int int8 int16 int32 int64
uint uint8 uint16 uint32 uint64 uintptr
byte (alias for uint8)
rune (alias for int32)
float32 float64
complex64 complex128

Zero Values

Variables declared without an explicit initial value are given their zero value:

0 for numeric types,

false for the boolean type,

"" (the empty string) for strings.

```
var i int
var f float64
var b bool
var s string
fmt.Printf("%v %v %v %q", i, f, b, s)
```