Golang Cheat Sheet

Importing Packages

```
import "fmt"

// import multiple packages
// r is aliasing for calling
math/rand package
import (
    "fmt"
    r "math/rand"
)
```

Running Go Program

```
// build
$ go build hello.go
$ ./hello

// build and run
$ go run hello.go
```

Variables

```
// explicitly typed
var num2 int = 6

// declare and init
num3 := 7

// declare as string
var str string

// multiple declares and assign
var num5, num6 int = 8, 9

var (
    age = 25
    name = "Michael"
)
```

Type Conversion

```
num1 := 2
num2 := float32(num1)

num2 = 3.5
num4 := int(num2)

// convert string to int
sum, err := strconv.Atoi(sumStr)

// convert int to string
str := strconv.Itoa(sum)
```

Arrays

```
// array with 5 elements
var nums [5] int // [0 0 0 0 0]
```

Slices

```
primes := [] int {2, 3, 5, ↓
                  7, 11, 13}
```

Slicing

```
var c[3] string
c[0] = "iOS"
c[1] = "Android"
c[2] = "Windows Phone"

fmt.Println(c[0:2])
//[iOS Android]
```

Decision Making

```
if true {
    fmt.Println(true)
} else {
    fmt.Println(false)
}
```

Switch

```
grade := "B"
switch grade {
case "A":
    fallthrough
case "B":
    fallthrough
default:
    fmt.Println("Undefined")
}
```

Structs

```
type Point struct {
    X int
    Y int
}
ptA := Point{5,6}
```

Looping

```
for i:=0; i<5; i++ {
    fmt.Println(i)
}

// "while" loop
counter := 0
for counter <5 {
    fmt.Println(counter)
    counter++
}

// iterate through an array
for i,n:= range c {
    fmt.Println(i, n)
}
```

Maps

```
// declaring a map type
var heights map[string] int
// initialize the map
heights = make(map [string] int)
// checking for a key
value, ok := heights["Michael"]
// ok = true -> key exist
// ok = false -> key doesn't exist

// iterating over a map
for k, v := range heights {
    fmt.Println(k,v)
}

// creating an empty map
weights := map[string] float32 {}

// create and init
weights = map[string] float32 {
    "Mamang": 45.9,
    "Wida": 56.8,
}
```

Function

```
func doSomething() {
   fmt.Println("Hello")
}
// calling a function
doSomething()

// returns int result
func addNum(num1 int, num2 int) int {
    return num1 + num2
}

// Multiple Return Type Functions
func countOddEven(s string) (int,int) {
    odds, evens := 0,0
    for _, c := range s {
        if int(c) % 2 == 0 {
            evens++
        } else {
            odds++
        }
    }
    return odds,evens
}

// Named Return Type Functions
func countOddEven(s string) (odds,evens int) {
    odds, evens = 0, 0
    for _, c := range s {
        if int(c) % 2 == 0 {
            evens++
        } else {
            odds++
        }
    }
    return
}
```

Packages

```
package mystrings

// In Go, a name is not exported
// if it begins with non capital
// letter
func internalFunction() {
    ...
}

// Must begin with a capital
// letter in order to be
// exported
func CountOddEven(s string)(odds,evens int) {
    ...
}
```