Golang Cheat Sheet

Dainish Jabeen

March 14, 2023

1 Packages

Go uses packages, which can contain multiple files. **The app will start running in the main application.**

Names exported outside the packages, must use a Capital letter.

Listing 1: Golang basics

```
1
2       package main
3
4       import "fmt"
5
6       func main(){
7               fmt.Println("Hello World")
8       }
```

2 General

2.1 Types

Listing 2: Golang types

```
1
2         := //Declare and initialize non explicit type
3         >> //Shift bitwise right
4         <<  //Shift bitwise left
5
6         //ARRAY
7         name []string
8         var := []string("blah","meh")
9
10        //MAPS
11        map[key type]val type //Dict has to be made
12        m := make(map[string]String)
13
14        elem, ok = m[key] // check key exists
15
16        // Initialization
17
18        var i int // initializes as 0
19
20        // Constants
21
22        const(
23                x=1 // the type of this can change on context
24        )
25
26        p := &i //point to i
27        *p //value of i, changes will change also change i
28
29        type Name struct{
30                x int
31                y int
32        }
33
34        // Pointers to structs
35
36        v := StructName{1,2}
37
38        p = &v
39        p.x = //Will change the value of v
40
41        // Struct constructors
42
43        v := StructName{x:1} //Others members made 0
44
45        //SLICES
```

```
46
47          //Slices acts as pointers
48
49          a[1:] // slice to end
50          s := a[:3] // slice start to 3
51
52          cap(s) // Capacity, elements in underlying array
53
54          // Dynamically size arrays
55
56          a := make(type, len, cap)
57          append(arr, val)
```

2.2 Functions

Listing 3: Functions

```
1
2       func Name(name type) type{}
3
4       //FUNC PARAMS
5
6       func name(x int,y int)
7       func name(x,y int)
8
9       //NAKED RETURN
10
11      func() (x,y int) {
12              x:=1
13              y:=2
14              return
15      } // Will return x and y
```

2.2.1 Funcs as params

Listing 4: Params

```
1
2       func compute(fn func(float64, float64) float64) float64 {
3       return fn(3, 4)
4       }
5
6       func main() {
7               hypot := func(x, y float64) float64 {
8                       return math.Sqrt(x*x + y*y)
9               }
10              fmt.Println(hypot(5, 12))
11
12              fmt.Println(compute(hypot))
13              fmt.Println(compute(math.Pow))
14      }
```

2.3 Control

Listing 5: Control

```
1
2       for i:=0;i<10;i++{}
3
4       for x<100 {} //Same as while loop
5
6       if statement; cond {}
7
8       switch statement; val {
```

```
9
10              case x: //x same as val == x
11
12              case y: //y same as val == y
13
14              default:
15          }
16
17          defer expr //execute expr at the end of func, can stack defers
18
19          for i,v := range arr {} // Loops through array (can do _,v or i,_)
```