1 Golang CheatSheet LANGUAGES

- PDF Link: cheatsheet-golang-A4.pdf, Category: languages
- Blog URL: `https://cheatsheet.dennyzhang.com/cheatsheet-golang-A4`

File me Issues or star this repo.

1.1 Golang Handy Commands

Name	Comment
Online Go Playgroud	`https://play.golang.org/`
Declare variables with initializers	var ischecked, v, str = false, 2, "yes!"
One line if statement	if a >= 1 { fmt.Print("yes") }
Golang switch	example-switch.go

1.2 Syntax Sugar: From Python To Golang

Name	Python	Golang
sum slice	`sum([1, 2, 3])`	sum := 0; for i := range nums { sum += nums[i] }
Get last item	`nums[-1]`	nums[len(nums)-1]
For	`for i in range(10):`	for i := 0; i < 10; i++
Loop list	`for num in [1, 2]`	for num := range[]int{1, 2} { fmt.Print(num) }
Loop string	`for ch in str:`	for _, ch := range str { fmt.Print(ch) }
Iterator	`for num in nums:`	for _, num := range nums {fmt.Print(num)}
While	`while isOK:`	`for isOK`
Check ch range	`ord(ch) in range(ord('a'), ord('z')+1)`	ch >='a' && ch <='z'
Get min	`min(2, 6, 5)`	
Check is nil	`root is None`	root == nil
Reverse list	`nums[::-1]`	Need to create your own function. Weird!

1.3 Array/List/Slice

Name	Comment
Make a array	var a [2]string; a[0]="hello"; a[1]="world"
Create array with given values	l := [6]int{2, 3, 7, 5, 11, 13}
Create array with given values	l := []string{"a", "c", "b", "d"}
Create dynamically-sized arrays	a := make([]int, 5)
Create dynamically-sized arrays	a := make([]int, 1, 5) // 5 is capacity
Sort string array	`sort.Strings(l); fmt.Print(l)`
Sort int array	`sort.Ints(l)` //in-place change
Append item	l = append(l, "e")
Append items	l = append(l, "e", "b", "c")
Append item to head/prepend	`l = append([]string{"a"}, l...)`
Remove last item	`l = l[:len(l)-1]`
Remove item by index	`l = append(l[0:1], l[2:]...)`
Slices of a array	`var l2 = l[1:3]` // Notice: it's a reference
Copy a list	b := make([]l, len(a)); copy(b, a)
Join two lists	`l1 = append(l1, l2...)`
Use pointer of array list	code/pointer-array.go

1.4 String

Package strings

Name	Comment
Format string	`fmt.Sprintf("At %v, %s", e.When, e.What)`
Format string	`fmt.Printf("int: %d, float: %f, bool: %t\n", 123, 78.9, true)`
Split string	`var L = strings.Split("hi,golang", ",")`
Replace string	`var str2 = strings.Replace("hi,all", ",", ";", -1)`
Replace string	`strings.Replace("aaaa", "a", "b", 2)` //bbaa
Split string by separator	`strings.Split(path, " ")`
Count characters	`strings.Count("test", "t")`
Substring	`strings.Index("test", "e")`
Join string	`strings.Join([]string{"a","b"}, "-")`
Repeat string	`strings.Repeat("a", 2)` // aa
Lower string	`strings.ToLower("TEST")`
Trim whitespace in two sides	`strings.TrimSpace("\t Hello world!\n ")`
Trim trailing whitespace	`strings.TrimRight("\t Hello world!\n ", "\n ")`
Concact string	`fmt.Sprintf("%s%s", str1, str2)`

1.5 Conversion

Name	Comment
Convert string to int	i, _ := strconv.ParseInt("12345", 10, 64)
Convert string to int	i, err := strconv.Atoi("-42")
Convert string to list	L := strings.Split("hi,golang", "")
Convert string to []byte	[]byte("abcXX")
Convert string to float32	f, _ := strconv.ParseFloat("3.1415", 32)
Convert int to float32	0.5*float32(age)+7>= float32(age2)
Convert int to string	s := strconv.Itoa(-42)
Convert list to string	strings.Join(list, ", ")
Convert byte to int	`int(byte('a'))`
Convert int32 to int32 Pointer	`func int32Ptr(i int32) *int32 { return &i }`
Convert string[] to string	strings.Join([]string{"a", "b"}, ",")

1.6 Integer/Float

Name	Comment
Int max	`MaxInt32 = 1«31 - 1` golang math
Int min	`MinInt32 = -1 « 31` golang math
Pass int as reference	sample code

1.7 Package management

Name	Comment
go mod	link: go modules
go get fix	`GO111MODULE=off go get -fix ./...`

1.8 Ascii

Name	Comment
get character ascii	`byte('0')`
ascii offset	`fmt.Println(string('B' + byte('a')-byte('A')))`

1.9 Dict/Hashmap/Map

Name	Comment
Create dict	`map[string]int{"a": 1, "b": 2}`
Create dict	`make(map[string]int)`
Check existence	_, ok := m[k]
Delete key	`delete(m, "k1")`
Create a map of lists	m := make(map[string][]string)

1.10 Goroutines

Name	Comment
Basic goroutine	code/example-goroutine.go

1.11 Inteface

Name	Comment
Hash map with both key and value dynamic	`map[interface{}]interface{}`
Convert map[interface {}]interface {} to map[string]string	code/interface-conversion.go

1.12 Files & Folders

Name	Comment
Read files	code/example-read-file.go
Write files	code/example-write-file.go

1.13 Bit Operator & Math

Name	Comment
Shift left	`fmt.Print(1 « 10)` // 1024
Shift right	`fmt.Print(1024 » 3)` // 128
pow(2, 3)	`int(math.Pow(2, 3))` // Default is float64

- Create 2D arrays

```
// static
board := [][]string{
         []string{"_", "_", "_"},
         []string{"_", "_", "_"},
         []string{"_", "_", "_"},
}

// dynamic
a := make([][]uint8, dy)
for i := range a {
    a[i] = make([]uint8, dx)
}
```

- Logging

```
import "github.com/op/go-logging"
log := logging.MustGetLogger("my-app")
log.Info("Some info...")
log.Warning("Some warning...")
log.Error("Some error!")
log.Critical("Some critical!")
```

- struct

```
type Point struct {
  X, Y int
}

var (
  v1 = Point{10, 8}
  v2 = Point{X: 1}  // Y would be 0
  v3 = Point{}      // Both X and Y is 0
  p  = &Point{10, 8} // reference: type *Point
)

func main() {
  fmt.Println(p, v1, v2, v3)
}
```

- Print Map

```
import "encoding/json"

b, err := json.MarshalIndent(x, "", "  ")
fmt.Println(string(b))

for key := range record {
   fmt.Printf("key: %s, value: %s\n", key, record[key])
}
```

- Print TreeNode

```
func printTreeNodePreOrder(root *TreeNode) {
    if root != nil { fmt.Println(root.Val) }
    if root.Left != nil { printTreeNodePreOrder(root.Left) }
    if root.Right != nil { printTreeNodePreOrder(root.Right) }
}
```

- Goroutines & Channels

```
// Goroutines
go func() {
  // do something
}

// Channels
c := make(chan T [, capacity ])
c <- t // blocks on unbuffered channels until another routine receives the value

d := <-c // blocks on unbuffered channels until another routine sends the value

close(c)
```

1.14 More Resources

- https://tour.golang.org/list
- https://golang.org/doc/
- https://github.com/a8m/go-lang-cheat-sheet