

1 Golang CheatSheet

LANGUAGES

- PDF Link: [cheatsheet-golang-A4.pdf](#), Category: languages
- Blog URL: <https://cheatsheet.dennyzhang.com/cheatsheet-golang-A4>
- Related posts: Ruby CheatSheet, Python CheatSheet, #denny-cheatsheets

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	<code>var ischecked, v, str = false, 2, "yes!"</code>
One line if statement	<code>if a >= 1 { fmt.Print("yes") }</code>
Golang switch	<code>example-switch.go</code>

1.2 Conversion

Name	Comment
Convert string to int	<code>i, _ := strconv.ParseInt("12345", 10, 64)</code>
Convert string to int	<code>i, err := strconv.Atoi("-42")</code>
Convert string to list	<code>L := strings.Split("hi,golang", ",")</code>
Convert string to []byte	<code>[]byte("abcXX")</code>
Convert string to float32	<code>f, _ := strconv.ParseFloat("3.1415", 32)</code>
Convert int to float32	<code>0.5*float32(age)+7>= float32(age2)</code>
Convert int to string	<code>s := strconv.Itoa(-42)</code>
Convert rune to string	<code>string(rune1)</code>
Convert list to string	<code>strings.Join(list, ", ")</code>
Convert list to byte	<code>byteI := byte(65)</code>
Convert byte to int	<code>int(byte('a'))</code>
Convert bytes to string	<code>string([]byte("abcXX"))</code>
Convert int32 to int32 Pointer	<code>func int32Ptr(i int32) *int32 { return &i }</code>
Convert string[] to string	<code>strings.Join([]string{"a", "b"}, ",")</code>

1.3 Syntax Sugar: From Python To Golang

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

1.4 Array/List/Slice

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

1.5 String

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

1.6 Integer/Float

Name	Comment
Int max	<code>MaxInt32 = 1<31 - 1 golang math</code>
Int min	<code>MinInt32 = -1 < 31 golang math</code>
Pass int as reference	sample code

1.7 Package management

Name	Comment
go mod	Link: go modules
go get fix	<code>G0111MODULE=off go get -fix ./...</code>

1.8 Ascii

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

1.9 Dict/Hashmap/Map

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

1.10 Goroutines

Name	Comment
Basic goroutine	<code>code/example-goroutine.go</code>

1.11 Interface

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

1.12 Files & Folders

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

1.13 Bit Operator & Math

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

1.14 Golang Common Algorithms

Name	Comment
bfs	<code>code/tree-bfs.go</code>
trie tree	<code>code/tree-trie.go</code>

1.15 Code snippets

- 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 { return }
    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)
```

1.16 More Resources

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

License: Code is licensed under MIT License.