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 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 rune to string	`string(rune1)`
Convert list to string	`strings.Join(list, ", ")`
Convert list to byte	byteI := byte(65)
Convert byte to int	`int(byte('a'))`
Convert byte to string	`string(byte('a'))`
Convert bytes to string	`string([]byte("abcXX"))`
Convert int32 to int32 Pointer	`func int32Ptr(i int32) *int32 { return &i }`
Convert string[] to string	`strings.Join([]string{"a", "b"}, ",")`
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)`

1.2 Golang Quality Improvement Tools

Name	Comment
gosec	Golang security checker, `gosec ./...`
golangci-lint	lint check, `golint`
errcheck	errcheck checks that you checked errors, `errcheck ./...`
delve	Delve is a debugger for the Go programming language.
ginkgo	BDD Testing Framework for Go
mock	GoMock is a mocking framework for Golang
envtest	provides libraries for integration testing by starting a local control plane
go-junit-report	Convert go test output to junit xml
gocover-cobertura	go tool cover to XML (Cobertura) export tool
gocovmerge	Merge coverprofile results from multiple go cover runs

1.3 Golang Dependencies

Name	Comment
Go modules	export GO111MODULE=on, `go mod download`
goimports	updates your Go import lines, adding missing ones and removing unreferenced ones
dep	Go deps, now recommend `go modules`
Install go dep	`go get -u github.com/golang/dep/cmd/dep`

1.4 Golang Common

Name	Comment
Upgrade golang to 1.12 in mac	`brew upgrade go`, `go version`
Reference	Link: The Go Programming Language Specification

1.5 Golang Code Structure & Common Algorithms

Name	Comment
Online Go Playgroud	`https://play.golang.org/`
One line if statement	if a >= 1 { fmt.Print("yes") }
Declare variables with initializers	var ischecked, v, str = false, 2, "yes!"
goroutine	Define functions to run as distince subprocesses
switch	code/example-switch.go
queue	Leetcode: Number of Recent Calls
bfs	code/tree-bfs.go
trie tree	code/tree-trie.go

1.6 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!

1.7 Surprises In Golang

Name	Comment
Modulus returns negative numbers	In golang, -3 % 2 == -1

1.8 Golang 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
Golang sort one array by another array	Leetcode: Largest Values From Labels
Sort in descending order	`sort.Sort(sort.Reverse(sort.IntSlice(keys)))`
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([]int, len(a)); copy(b, a)
Join two lists	`l1 = append(l1, l2...)`
Use pointer of array list	code/pointer-array.go

1.9 Golang String

Name	Comment
Reference	Link: package strings
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)`
Padding zero	`fmt.Printf("%02d:%02d", 2, 10)`
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)`

1.10 Golang 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

1.11 Golang Env

Name	Comment
GOPATH	It is called as the workspace directory for Go programs
GOROOT	The location of your Go installation. No need to set any more
`go env`	Show a full list of environment variables
Reference	Link: GOPATH, GOROOT, GOBIN

1.12 Golang Package management

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

1.13 Golang Ascii

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

1.14 Golang 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)
Get keys of a map	Loop the dictionary and append the key to a list

1.15 Golang Goroutines

Name	Comment
Basic goroutine	code/example-goroutine.go

1.16 Golang Inteface

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

1.17 Golang Files & Folders

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

1.18 Golang 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
sqrt	`math.Sqrt(100)`

1.19 More Resources

`https://play.golang.org/`
`https://tour.golang.org/list`
`https://golang.org/doc/`
`https://github.com/a8m/go-lang-cheat-sheet`
License: Code is licensed under MIT License.