

# Go Lang Cheat Sheet

by Navron via cheatography.com/185051/cs/38662/

# packages package main -Every package file has to start with package.

Importing	
import "fmt"	import (
import "math/rand"	"fmt"
	"math/rand")
-Both are the same	

## Control Loops

```
// Normal For
for i := 0; i < 10; i++ {
sum += i
// For loop is Also While Loop
in Go
i := 0
for i < 5 {
       fmt.Pr int ln(i)
// For with Range
nums := []int{ 2,4,6,8}
for index,val := range nums {
       fmt.Pr int ln( index,
val)
//Here range will give the index
of nums to var index
- No paranthesis required around condit-
```

	'	
ionals and increments		
- Variable Initialization and Increment is		
optional here		
	By <b>Navron</b>	
	cheatography.com/navron/	

Maps	
// Creating Map with Make	// creating Map
var name = make([type]type)	var name = map[type]- type{val:val, val:val}
var m = make([string], string)	

Insert or update an element in map	Retrieve an element:
m[key] = elem	elem = m[key]

Delete an element:	
delete(m, key)	

Variable Types	
bool	string
int int8 int16 int32 int64	uint uint8 uint16 uint32 uint64 uintptr
byte // alias for uint8	rune // alias for int32
float32 float64	complex64 complex128

Variable Conversion		
x := 8		
y := float32(x)		
you can replace float32 with different types		

Array	
// Array Declaration Only	// Array Declaration without var
var name [size]type	name := [size]type{val,- val,val} //size is optional
var a [10]string	primes := [6]int{2, 3, 5, 7, 11, 13}

Not published yet.
Last updated 12th May, 2023.
Page 1 of 2.

Pointers	
Initailize	The & operator generates a pointer to its operand
var p *int	p = &i

The * operator denotes the pointer	S
underlying value.	

*p	=	21

If-else & Switch Case	
If-Else	if-else with Short Statement
if conditional { code }	if short statement; conditional {code}
else { code }	if $v := math.Pow(x, n); v < lim {$
	return v
	} else {
	fmt.Printf("%g >= %g\n", v, lim) }

Switch Case	
switch conditional	
case 0:	
code	
case 1:	
code	
default:	
-Variables declared inside an if short	
statement are only in scope until the end of	
if-else block	

statement are only in scope until the end of if-else block
-Go's switch cases need not be constants, and the values involved need not be integers

Sponsored by CrosswordCheats.com Learn to solve cryptic crosswords! http://crosswordcheats.com



## Go Lang Cheat Sheet

by Navron via cheatography.com/185051/cs/38662/

Variables basic	
// var name type =	// Declaration without
value	type
var x int = 5	x := 5

#### // Constant Declaration with const keyword

const Phi = 1.618

":=" is called Short Assignment statement
Outside of function ":=" is not allowed

Exported names :- To use any variable outside of package it must start with Capital Letter or else it will not run

Slices	
// Slice Declar- ation	// Creating Slice with Make
a[low : high]	var := make([ ]type, size)
a[1:4]	a := make([ ]int, 5) // len(a)=5

#### // Appending Item

name = append(name, val)

a = append(a, "Something")

- An array has a fixed size. A slice, on the other hand, is a dynamically-sized, flexible view into the elements of an array
- A slice does not store any data, it just describes a section of an underlying array

### Functions

```
' func function_name(args type)
(output types) {
CODE
}'
func swap(x, y string) (string,
string) {
return y, x
}
```

- func is used to declaration function
- in case of different input type we have to specify each one differently
- Output type is must if there is single return then no need to add paranthesis

Structs		
initailize	Struct fields are accessed using a dot	
type Name struct {	v := Vertex{1, 2}	
var type }	v.X = 4	
type Vertex struct { X int }		
v := Vertex{1, 2}		

Pointers to structs	Struct Literals
v := Vertex{1, 2}	var (
p := &v	v1 = Vertex{1, 2} // has type Vertex
fmt.Print- In(p.X)	v2 = Vertex{X: 1} // Y:0 is implicit
	v3 = Vertex{} // X:0 and Y:0
	p = &Vertex{1, 2} // has type *Vertex )

- -A struct is a collection of fields
- -A struct literal denotes a newly allocated struct value by listing the values of its fields



cheatography.com/navron/

Not published yet. Last updated 12th May, 2023. Page 2 of 2. Sponsored by CrosswordCheats.com Learn to solve cryptic crosswords! http://crosswordcheats.com