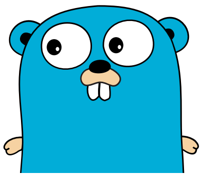
cheat sheet GO



Installing Go

```
https://golang.org/doc/install
$ go version
```

Go program

```
package main
import "fmt"
import r "math/rand"

func main() {
    fmt.Println("Hello",r.Int())
}
```

Build & Run

```
$ ### RUN ###
$ go run .
$ ### VET & BUILD & RUN ###
$ go vet hello.go
$ go build hello.go
$ ./hello
$ ### INSTALL & RUN ###
$ go install hello.go
$ $GOBIN/hello
```

Variables & Constants

```
// declaration
var msg string
msg = "Hello"
// short with type inference
msg := "Hello"
// constants
const Pi = 3.14159
ip, port := "127.0.0.1", "8080"
fmt.Println(ip + ":" + port)
```

Types

```
str := "Hello" // string
str := `Multiline string`
num := 3 // int
num := 3. // float64
num := 3 + 4i // complex128
var c rune = '♬' // UTF-8
num := byte('A') // byte/uint8
fmt.Printf("%T\n", i) // print
s := reflect.TypeOf(i).String()
type Weight float64 // custom
w := Weight(70) // conversion
```

Pointers

```
var pi *int = &i // point to i
p := &i // point to i
*p = *p * 2 // dereferencing
ps.x == (*ps).x // equivalent
```

Arrays

```
var a [5]int // fixed size
a[0] = 3 // assignment
a := [...]int{1,3:2,3,6:-1}
var a [2][3]int
pa := *[32]byte{}
```

Slices

```
var s []int // dynamic size
s := []int {1,2,3}
s := []byte("Hello")
s := make([]string, 3)
s = append(s, "d", "e")
c := make([]string, len(s))
copy(dst, src)
x := s[2:5] // elem. 2,3,4
y := s[:5] // elem. < 5
```

Maps

```
m := make(map[string]int)
m["key1"] = 42
fmt.Println("map: ", m)
m := map[string]int{"foo": 1,
     "bar": 2} // initialize
v := m["key1"]
_, contains := m["key2"]
length := len(m)
delete(m, "key1")
```

Loops

```
for i := 0; i < 10; i++ {/**/}
for i <= 3 { i = i + 1 }
for {/**/ continue /**/ break}
```

Ranges

```
s := []string{"a", "b", "c"}
for idx, val := range s {/**/}
m := map[string]int{"a": 1}
for k, v := range m {/**/}
```

Conditionals

```
if d == "Sun" || d == "Sat" {
} else if d == "Mon" && foo() {
} else if _, err := f();
  err != nil {
} else {/**/}
```

Switches

```
switch time.Now().Weekday() {
case 0:
    fallthrough
case 1: fmt.Println("Weekend")
default: fmt.Println("Workday")
}
```

Resources

https://tour.golang.org/
https://github.com/golang/go/wiki
https://repl.it/languages/go
https://awesome-go.com/

Functions

```
func add(a int, b int) float64 {
    return float64(a + b) }
func tuple() (int, int) {
    return 4, 2 }
x, y := tuple()
func fvar(nums ...int) {/**/}
```

Closures & Lambdas

```
func adder() func(int) int {
    sum := 0
    return func(x int) int {
        sum += x
        return sum } }
myLambda := func() bool {/**/}
```

Defer

```
file, err := os.Create("foo")
if err != nil {
  return err
}
defer func() { file.Close() }()
```

Structs & Methods

```
type Person struct {
    name string
    age int
}
func (p *Person) Aging(y int) {
    p.age = p.age + y
}
p := Person{name: "Bob", age: 4}
p.age = 30
p.Aging(1)
```

Interfaces

```
type geometry interface {
    area() float64
    perim() float64
}
func (r rect) area() float64 {}
func (r rect) perim() float64 {}
func measure(g geometry) {}
r := rect{width: 2, height: 4}
measure(r)
```

Concurrency

```
func f(c chan int) {}
c := make(chan int)
cb := make(chan int, bufferLen)
go func() {
  fmt.Println(<-c)
}()
c <- 2 // send 2 to c
x, y := <-c, <-c // recv from c
close(c) // close chan
select { case c <- x: /**/
case <-quit: return }
```

Sync

```
var mu sync.Mutex // sync.Once
// .Lock();.Unlock();once.Do(f)
var wg sync.WaitGroup
// .Add(int);.Done();.Wait()
```

golang.sk

challenge | reshape | boost

hello@golang.sk
https://www.golang.sk

Published 7th November 2020