Hello World

```
package main

import “fmt”

func main() {
  fmt.Println(“Hello World”)
} → Hello World
```

For Loop

```
for i:=0; i<10; i++ {
  sum += i
}
```

```
sum:=1
for ; sum<1000 ; {
  sum += sum
} → for sum<1000 {
```

```
for {
  fmt.Println(“infinite ∞”)
} → endless loop
```

if

```
if x < 0 {
  fmt.Println(“neg”)
}
```

```
if x:=y+1; x < 0 {
  fmt.Println(“%d is neg”,x)
} → x in scope if{} only
```

```
if x:=y+1; x < 0 {
  fmt.Println(“%d is neg”,x)
} else {
  fmt.Println(“%d >= 0”,x)
} → else is part of if{}
```

Functions

```
func addsub(x, y int) (int,int) {
  return x + y, x-y
}

func main() {
  fmt.Println(add(2,3))
} → 5 -1
```

```
func addsub(x, y int) (a,s int) {
  a = x + y
  b = x – y
  return
}
```

Switch

```
switch x := y+1; x {
  case x < 0:
    fmt.Println(“%d is neg”,x)
  case 0:
    fmt.Println(“%d is zero”,x)
  default:
    fmt.Println(“%d is pos”,x)
} → x in scope switch{} only
```

```
switch {
  case x < 0:
    fmt.Println(“%d is neg”,x)
  case x < 1:
    fmt.Println(“%d is zero”,x)
  default:
    fmt.Println(“%d is pos”,x)
} → switch is an if-chain!
```

Defer

```
func main() {
  for i := 0; i < 10; i++ {
    defer fmt.Print(i, ” “)
  }
} → preprocess, stack execute

→ 9 8 7 6 5 4 3 2 1 0
```

Structs

```
type Vertex struct {
  X int
  Y int
}

                                  ⇒
func main() {
  v := Vertex{1,2}
  v.X = 4
  fmt.Println(v)
} → {4 2}
```

Declarations

```
var i int (→ i=0)
var j,k int = 1,2
l := 3 (implicit type)
m,n,s := 4, 5, ”str”
```

Types

```
int  int8  int16
int32  int64 → 0

uint  uint8  uint16
uint32  uint64
uintptr  → 0

byte ↔ uint8
rune ↔ int32 (unicode)

float32  float64 → 0

complex64  complex128 → 0

bool   → false
string → “”
```

Type conversion

```
var i int = 42
f := float64(i)
u := uint(f)
```

Constants

```
const Pi = 3.1415926535897
```

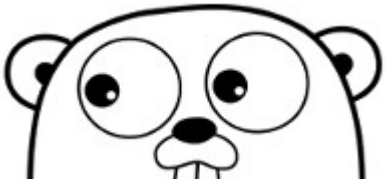
Pointers

```
var p *int → nil
i := 42
p = &i

*int → pointer to an int
&i   → the address of I
p=&i → address ⇒ ptr
```

*struct

```
func main() {
  v := Vertex{}
  p := &v
  p.X = 100
  fmt.Println(v)
} → {100 0}
```



nommed from tour.golang.org