Hello World

```
package main

import "fmt"

func main() {
  fmt.Println("Hello World")
}
```

Variables

Without Initial Value

```
// variable will be assigned with zero value
var i int
var s string
var ok bool
```

With Initial Value

```
var s string = "hello"
var s = "hello"
var s = `hello
multiple
Line`
var a, b = 1, 2 // multivariable
```

With Initial Value (Only in a Function)

```
i := 14
s := "hello"
ok := true
```

Zero Values

```
number = 0
string = ""
boolean = false
pointer = nil
```

Constants

Assigned Value

```
const defaultValue = 1
const defaultTitle = "Go"
const defaultStatus = true
```

IOTA

```
const (
  sunday = iota
  monday
  tuesday
  wednesday
  thursday
  friday
  saturday
)

//sunday = 0
monday = 1
tuesday = 2
wednesday = 3
thursday = 4
friday = 5
saturday = 6
```

Functions

No return value

```
func greeting(firstName, lastName string) {
  fmt.Println("Hello", firstName, last Name)
}
```

1 return value

```
func add(a, b int) int {
  return a + b
}
```

2 return value

```
func swap(a, b int) (int, int) {
  return b, a
}
```

Conditionals

If - *Basic If*

```
if name != "" {
  fmt.Printf("Hello %s\n", name)
} else {
  fmt.Println("Hello friend")
}
```

If - *If with Statement*

```
if n, err := strconv.Atoi("5s"); err != nil {
  fmt.Println("NaN:", err)
} else {
  fmt.Println(n)
}
```

Switch

```
switch day {
  case "saturday"
    fallthrough
  case "sunday"
    fmt.Println("take some rest")
  default:
    fmt.Println("go working")
}
```

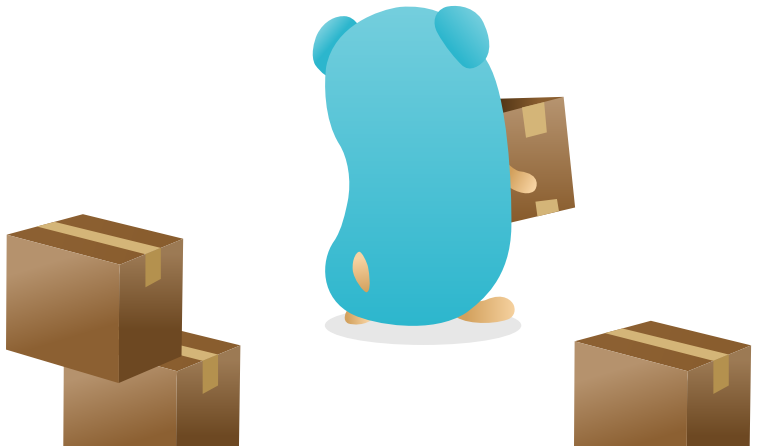
Loops

For

```
// Standard for loop
for i := 0; i < 10; i++ {
  // Do something
}

// Only condition
for i < 10 {
  // Do something
}

// Infinite loop
for {
  // Do something
}
```



Go Types

Frequently Used

```
// bool
b := true

// string
s := "Hello World"

// int
i := 42

// rune - alias for int32 represents a Unicode code point
r := 'A'

// byte - alias for uint8
b := byte('A')

// float64
f := 1.

// complex128
z := 4 + 5i
```

Primitives

```
bool

string

int  int8  int16  int32  int64
uint uint8 uint16 uint32 uint64 uintptr

byte // alias for uint8

rune // alias for int32 - represents a Unicode code point

float32 float64

complex64 complex128
```

Pointers

```
func add2 (d *int) {
  *d = *d + 2
}

func main () {
  d := 2
  add2(&d)
  fmt.Println(d) // outputs: 4
}
```

Arrays

```
// specific-size array
a1 := [5]int{1, 3, 5, 7, 9}

// auto-size array
a2 := [...]int{2, 4, 6, 8, 10}
```

Slices

```
// create slice from initial value
aphabet := []string{"a", "b", "C"}

b := aphabet[1:2]
a := aphabet[:1]
c := aphabet[2:]
abc := aphabet[:]

// append slice
abcd := append(abc, "d")
```

Map

```
m := map[string]string{
  "a": "apple",
  "b": "banana",
}

fmt.Println(m[“a”]) // outputs: “apple”
m[“a”] = “orange”
fmt.Println(m[“a”]) // outputs: “orange”
```

Range

```
// Can be use with arrays, slices, maps, channels
s := [...]int {1, 2, 3}
for i := range s {
}

s := []int {1, 2, 3}
for i := range s {
}

s := []int {1, 2, 3}
for i, v := range s {
}

s := []int {1, 2, 3}
for j, v := range s {
}

m := map[string]string { "a": "apple"}
for k, v := range m {
}

ch := make(chan int)
for v := range ch {
}
```

Struct

```
type Rectangle struct {
  w, l float64
}

func main() {
  rec := Rectangle{w: 1, l: 2}
  fmt.Println(rec.w, rec.l) // outputs: 1 2

  rec.w = 3
  fmt.Println(rec.w, rec.l) // outputs: 3 2
}
```

Methods

```
func (rec Rectangle) Area() float64 {
  return rec.w * rec.l
}

func main() {
  fmt.Println(rec.Area())
}
```

Pointer Receiver

```
func (rec *Rectangle) SetWidth(w float64) {
  rec.w = w
}

func main() {
  rec := Rectangle{w: 1, l: 2}
  rec.SetWidth(3)
  fmt.Println(rec.w, rec.l) // outputs: 3 2
}
```

Interface

```
// Declare interface
type Shape interface {
  Area() float64
}

// Struct
type Rectangle struct {
  w, l float64
}

// Methods
func (rec Rectangle) Area() float64 {
  return rec.w * rec.l
}

// Example
func main() {
  var rec Shape = Rectangle{w: 3, l: 4}
  fmt.Printf(rec.Area()) // outputs: 12
}
```

Empty Interface

```
var i interface{}
i = "string"
i = 9
```

Concurrency

Goroutines - *Example*

```
func doSomething() {
  time.Sleep(100 * time.Millisecond)
}

func main() {
  start := time.Now()

  go doSomething()
  go doSomething()
  go doSomething()

  time.Sleep(120 * time.Millisecond)

  fmt.Println(time.Since(start)) // outputs: 123.771667 ms
}
```

Channel - *Example*

```
func main() {
  rw := make(chan string)
  go greeting(rw)

  rw <- "John"
  fmt.Println(<-rw) // outputs: Hello, John
}

func greeting(rw chan string) {
  s := <-rw
  rw <- "Hello, " + s
}
```

Error

Error Handling

```
type error interface {
  Error() string
}

// Error Handling
if ok {
}

if n, err := strconv.Atoi("1"); err != nil {
  fmt.Println("NaN:", err)
}
```

Advanced Concepts

Defer

```
resp, err := http.Get("url")
if err != nil {
// handler error
}
defer resp.Body.Close()
```

Recovery

```
func someFunc() {
  defer func() {
    if r := recover(); r != nil {
      log.Println(r)
    }
  }()
}
```

First Class Function

```
var add = func(a, b int) int {
  return a + b
}
```

Higher Order Function

```
func cal(add func(int, int) int, a, b int) string {
    return fmt.Sprintf("result is %d", add(a, b))
}

func newAddFunc() func(int, int) int {
    return func(a, b, int) int {
      return a + b
    }
}
```

fmt.Stringer

```
type Stringer interface {
  String() string
}
```

Type Assertion

```
var i interface{}
i = "string"

if s, ok := i.(string); ok {
}
```

Popular Lib

Lib	Link
web framework	https://github.com/gin-gonic/gin
	https://github.com/labstack/echo
	https://github.com/go-chi/chi
	https://github.com/gofiber/fiber
	https://github.com/gorilla/mux
log	https://github.com/uber-go/zap
	https://github.com/Sirupsen/logrus
ORM	gorm.io/gorm
gin/CORS	github.com/gin-contrib/cors
auto load env	github.com/joho/godotenv
rate limit	golang.org/x/time/rate
jwt	github.com/dgrijalva/jwt-go
configuration	https://github.com/spf13/viper
kafka	https://github.com/confluentinc/confluent-kafka-go
	https://github.com/Shopify/sarama

Lib	Link
testing	https://github.com/stretchr/testify
BDD	https://onsi.github.io/ginkgo/
UUID	https://github.com/google/uuid
validation	https://github.com/go-playground/validator
CLI framework	https://github.com/spf13/cobra

CLI

Environment	
แสดงค่า environment ของ Go	go env
ตั้งค่า environment ของ Go หรือทับค่าเดิม	go env -w GOPATH=~/gopath

Module	
สร้างโมดูล(โปรเจค)ใหม่	go mod init [project_name]
สร้างโมดูลใหม่สำหรับ open source	go mod init github.com/account /projectname
download dependencies modules ที่ต้องการ import	go get github.com/pkg/errors
ลบ module ที่ไม่ได้ใช้ และเพิ่มตัวที่เรียกใช้แต่ยังไม่มีใน module	go mod tidy
download dependencies modules to local cache	go mod download

Run	
รันโปรแกรม	go run main.go
รันโปรแกรมในกรณีที่มีหลายไฟล์	go run .
รันโปรแกรมพร้อมตรวจสอบ race condition	go run -race main.go

Build	
compile โปรแกรม	go build -o app main.go

Clean	
clean cache ที่รันโปรแกรม	go clean --cache
ลบ module ที่ download มา	go clean --modcache

Unit Testing	
เทสทุกเคสตั้งแต่ directory ปัจจุบันไปในชั้นลึกทั้งหมด	go test ./...
เทสพร้อมรายงาน coverage	go test ./... -cover
เทสพร้อมบอกรายละเอียดแต่ละเคส	go test ./... -v -cover
เทสพร้อมตรวจสอบ race condition	go test -race

Unit Testing	
install any tools written in Go	go install github.com/tsenart/ vegeta@latest