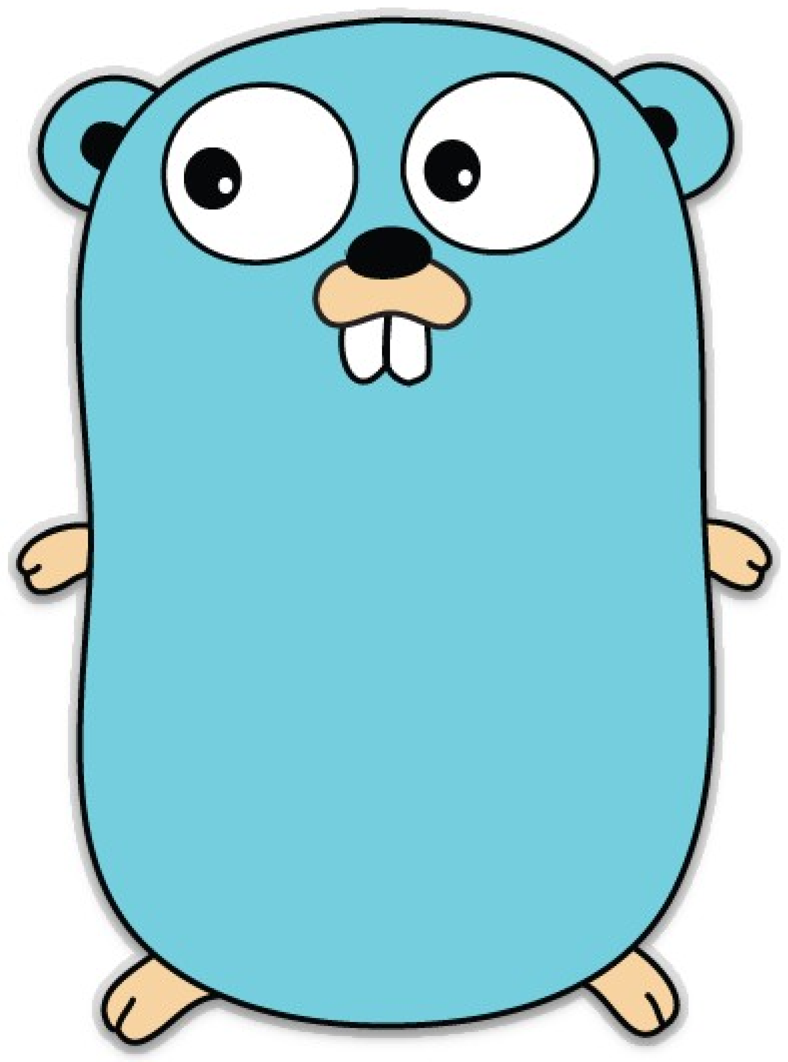

Book Two
Happy going going

- **Some examples won't compile. They have been written to explain some rules.**

This page has been left blank for you to create index of the book if required

S.No	Topic	Page No.

eg65.go

```
package main
import "fmt"
type Car interface {
manual()
}
type HondaCity struct {
// some properties
}
func (hondaCity *HondaCity) manual() {
fmt.Println("Some details of honda city")
}
type HondaJazz struct {
// some properties
}
func (hondaJazz *HondaJazz) manual() {
fmt.Println("Some details of honda jazz")
}
func doService(c Car){
c.manual()
}
func main(){
hc:=new(HondaCity)
hz:=new(HondaJazz)
doService(hc)
doService(hz)
}
```

eg66.go

```
package main
import "fmt"
func main(){
var g string
g="Good"
fmt.Println("Length of : ",g,"is",len(g))
}
```

eg67.go

```
package main
import "fmt"
func main(){
var k string
fmt.Print("Enter a string ")
fmt.Scanf("%s\n",&k)
for x:=0;x<len(k);x++ {
fmt.Println(k[x])
}
}
```

eg68.go

```
package main
import "fmt"
func main(){
var k string
fmt.Print("Enter a string ")
fmt.Scanf("%s\n",&k)
for x:=0;x<len(k);x++ {
fmt.Println(string(k[x]))
}
}
```

---

eg69.go

```
package main
import "fmt"
import "strings"
func main(){
var k string
fmt.Print("Enter a string ")
fmt.Scanf("%s\n",&k)
fmt.Println(strings.ToUpper(k))
fmt.Println(strings.ToLower(k))
k="  Cool"
fmt.Println(strings.TrimLeft(k," "))
k="   Cool   "
fmt.Println(strings.Trim(k," "))
k="Cool  "
fmt.Println(strings.TrimRight(k," "))
fmt.Println(strings.TrimPrefix("Cool","Co"))
fmt.Println(strings.TrimSuffix("Cool","ol"))
fmt.Println(strings.TrimPrefix("CoCool","Co"))
fmt.Println(strings.TrimSuffix("Coolol","ol"))
mytrimmer:=func(r int32) bool {
return r>=65 && r<=90
}
fmt.Println(strings.TrimFunc("Cooling Fan",mytrimmer))
fmt.Println(strings.TrimFunc("COOLing Fan are GREAT",mytrimmer))
/*
Assignment :
TrimLeftFunc,TrimRightFunc
*/
k="    Cool    "
fmt.Println(strings.TrimSpace(k))
fmt.Println(strings.Compare("Bombay","Baroda"))
fmt.Println(strings.Compare("Baroda","Baroda"))
fmt.Println(strings.Compare("Baroda","Bombay"))
fmt.Println(strings.Compare("Bombay","Bomb"))
fmt.Println(strings.Compare("Bomb","Bombay"))
fmt.Println(strings.Contains("Bombay","omb"))
fmt.Println(strings.Contains("omb","Bombay"))
fmt.Println(strings.ContainsAny("Bombay","obH"))
fmt.Println(strings.ContainsRune("Bombay",98))
fmt.Println(strings.ContainsRune("Bombay",122))
fmt.Println(strings.Count("Ujjain is the city of gods"," "))
fmt.Println(strings.EqualFold("Ujjain","UJJAIN"))
fmt.Println(strings.EqualFold("Ujjain","UJJAIN2"))
s:=strings.Fields("God   is     great")
for i,v:=range s {
fmt.Println(i,v)
}
myseparator:=func(c rune) bool{
if (c>=65 && c<=90) || (c>=97 && c<=122) || (c>=48 && c<=57) {
return false
}else{
return true
}
}
j:=strings.FieldsFunc("I live in Ujjain,What a city to live in. 0734-2514118",myseparator)
for i,v:=range j {
fmt.Println(i,v)
}
}
```

---

eg70.go

```
package main
import (
"fmt"
"strings"
)
func main(){
fmt.Println(strings.HasPrefix("Good","Go"))
fmt.Println(strings.HasPrefix("Good","go"))
fmt.Println(strings.HasSuffix("Good","od"))
fmt.Println(strings.HasSuffix("Good","OD"))
fmt.Println(strings.Index("Good","oo"))
fmt.Println(strings.Index("Good","ooz"))
fmt.Println(strings.IndexAny("Good","ooz"))
fmt.Println(strings.IndexByte("Good",'A'))
fmt.Println(strings.IndexByte("Good",'d'))
fmt.Println(strings.IndexRune("Good",65))
fmt.Println(strings.IndexRune("Good",100))
x:=[]string{"Good","Bad","Ugly"}
cs:=strings.Join(x,"-")
fmt.Println(cs)
fmt.Println(strings.LastIndex("Good","ooz"))
fmt.Println(strings.LastIndexAny("Good","ooz"))
fmt.Println(strings.LastIndexByte("Good",'o'))
mylocator:=func(x rune) bool {
if x>=48 && x<=57 {
return true
} else {
return false
}
}
fmt.Println(strings.LastIndexFunc("Good2Bad",mylocator))
fmt.Println(strings.LastIndexFunc("Good2Bad3Ugly",mylocator))
fmt.Println(strings.LastIndexFunc("Good",mylocator))
mymapper:=func(x rune) rune {
if x>=48 && x<=57 {
return x+16
} else {
return x
}
}
fmt.Println(strings.Map(mymapper,"Good1Bad2Ugly3Worst"))
}
```

eg71.go

```
package main
import (
"fmt"
"strings"
)
func addDivider(){
fmt.Println(strings.Repeat("-",20))
}
func main(){
s:=strings.Repeat("Cool",10)
fmt.Println(s)
fmt.Println(strings.Replace(s,"ool","old",2))
fmt.Println(s)
fmt.Println(strings.Replace(s,"ool","old",-1))
addDivider()
s="Sameer,Rohit,Mohan,Lokesh"
sep:=","
k:=strings.Split(s,sep)
for i,v:=range k {
fmt.Println(i,v)
}
addDivider()
k=strings.SplitAfter(s,sep)
for i,v:=range k {
fmt.Println(i,v)
}
addDivider()
k=strings.SplitAfter(s,"")
for i,v:=range k {
fmt.Println(i,v)
}
addDivider()
}
```

---

eg72.go

```
package main
import (
"fmt"
"strings"
)
func addDivider(){
fmt.Println(strings.Repeat("-",20))
}
func main(){
s:="We are dreaming of a 20 Lac Job PA"
k:=strings.SplitAfterN(s," ",3)
for i,v:=range k {
fmt.Println(i,v)
}
addDivider()
k=strings.SplitAfterN(s,"",5)
for i,v:=range k {
fmt.Println(i,v)
}
addDivider()
k=strings.SplitN(s," ",5)
for i,v:=range k {
fmt.Println(i,v)
}
addDivider()
k=strings.SplitN(s,"",5)
for i,v:=range k {
fmt.Println(i,v)
}
s="pune is the most idiotically sought after place for a student"
fmt.Println(strings.Title(s))
fmt.Println(s)
s="cool. fool,tool,bool"
fmt.Println(strings.Title(s))
s="god is great"
fmt.Println(strings.ToTitle(s))
}
```

---

eg73.go

```
package main
import (
"fmt"
"os"
)
func main(){
gg:=os.Args;
fmt.Println(gg)
for i,v:=range gg[1:] {
fmt.Println(i,v)
}
}
// while running the above code, pass command line arguments
```

---

eg74.go

```
package main
import (
"fmt"
"strconv"
"os"
)
func main(){
gg:=os.Args;
if len(gg)==1 {
fmt.Println("Usage : go run eg74.go  num1 num2  num3 ...")
return
}
t:=0
for _,v:=range gg[1:] {
num,err:=strconv.Atoi(v)
if err!=nil { continue }
t+=num
}
fmt.Println("Total is ",t)
}
```

---

eg75.go

```
package main
import (
"fmt"
"strconv"
)
func main(){
var x int64
var y error
x,y=strconv.ParseInt("65",10,64)
fmt.Println(x,y)
x,y=strconv.ParseInt("41",16,64)
fmt.Println(x,y)
x,y=strconv.ParseInt("101",8,64)
fmt.Println(x,y)
x,y=strconv.ParseInt("1000001",2,64)
fmt.Println(x,y)
x,y=strconv.ParseInt("256",10,8)
fmt.Println(x,y)
/*
try
ParseBool(str string)  (bool error)
ParseFloat(s string,bitSize int) (float64 error)
ParseUInt(s string,base int,bitSize int) (int64 error)
*/
}
```

eg76.go

```
package main
import (
"fmt"
"strconv"
)
func main(){
var g string
g=strconv.FormatBool(false)
fmt.Println(g)
g=strconv.FormatInt(65,10)
fmt.Println(g)
g=strconv.FormatInt(65,2)
fmt.Println(g)
g=strconv.FormatInt(65,8)
fmt.Println(g)
g=strconv.FormatInt(65,16)
fmt.Println(g)
/* Try
FormatUInt(i int64,base int)
*/
g="God is great"
fmt.Println(strconv.Quote(g))
fmt.Println(strconv.QuoteRune(123))
fmt.Println(strconv.QuoteRune(65))
}
```

go77.go

```
package main
import (
"fmt"
"strconv"
)
func main(){
f:=123.453222123132;
s1:=strconv.FormatFloat(f, 'E', -1, 32)
fmt.Println(s1)
s2:=strconv.FormatFloat(f, 'E', -1, 64)
fmt.Printf(s2)
}
```

go78.go

```
package main
import(
"fmt"
"time"
)
func main(){
var x time.Time
x=time.Now()
fmt.Println(x)
fmt.Println(x.Format("dd/MM/yyyy"))
fmt.Println(x.Format("dd/MM/06"))
fmt.Println(x.Format("dd/MM/2006"))
fmt.Println(x.Format("2/1/2006"))
fmt.Println(x.Format("02/01/2006"))
}

/*
Understand yourself

const (
   stdLongMonth      = "January"
   stdMonth          = "Jan"
   stdNumMonth       = "1"
   stdZeroMonth      = "01"
   stdLongWeekDay    = "Monday"
   stdWeekDay        = "Mon"
   stdDay            = "2"
   stdUnderDay       = "_2"
   stdZeroDay        = "02"
   stdHour           = "15"
   stdHour12         = "3"
   stdZeroHour12     = "03"
   stdMinute         = "4"
   stdZeroMinute     = "04"
   stdSecond         = "5"
   stdZeroSecond     = "05"
   stdLongYear       = "2006"
   stdYear           = "06"
   stdPM             = "PM"
   stdpm             = "pm"
   stdTZ             = "MST"
   stdISO8601TZ      = "Z0700"  // prints Z for UTC
   stdISO8601ColonTZ = "Z07:00" // prints Z for UTC
   stdNumTZ          = "-0700"  // always numeric
   stdNumShortTZ     = "-07"    // always numeric
   stdNumColonTZ     = "-07:00" // always numeric
)

*/
```

---

eg79.go

```
package main
import (
"fmt"
"strings"
)
func addDivider(s string){
fmt.Println(strings.Repeat("-",50))
fmt.Println(s)
fmt.Println(strings.Repeat("-",50))
}
func main(){
var x int32
x=65
addDivider("General")
fmt.Printf("%v\n",x)
fmt.Printf("%T\n",x)
fmt.Printf("%%\n")
fmt.Printf("\\n\n")
addDivider("Integer")
fmt.Printf("%b\n",x)
fmt.Printf("%c\n",x)
fmt.Printf("%d\n",x)
fmt.Printf("%o\n",x)
fmt.Printf("%q\n",x)
fmt.Printf("%x\n",x)
fmt.Printf("%X\n",x)
fmt.Printf("%U\n",x)
fmt.Printf("%+d\n",x)
x=-65
fmt.Printf("%+d\n",x)
fmt.Printf("(%10+d)",x)
addDivider("Floating point")
var k float64
k=float64(22/7)
fmt.Println(k)
var m float64
m=3.14159265359
fmt.Println(m)
var r float64
r=22.0/7.0
fmt.Println(r)
fmt.Printf("%b\n",r)
fmt.Printf("%e\n",r)
fmt.Printf("%E\n",r)
fmt.Printf("%f\n",r)
fmt.Printf("(%10.2f)\n",r)
fmt.Printf("%F\n",r)
fmt.Printf("%g\n",r)
fmt.Printf("%G\n",r)
addDivider("boolean")
fmt.Printf("%t\n",x<50)
addDivider("String and Slices of bytes")
var j string
j="God is great"
fmt.Printf("%s\n",j)
b:=[]byte {65,66,67}
fmt.Printf("%s\n",b)
fmt.Printf("%q\n",j)
fmt.Printf("%q\n",b)
fmt.Printf("%x\n",b)
j="abcd"
fmt.Printf("%x\n",j)
fmt.Printf("%p\n",&j)
type Toy struct {
brand string
price int
}
var doll Toy
doll.brand="Lego"
doll.price=1000
fmt.Printf("%v\n",doll)
}
```

eg80.go

```
package main
import "fmt"
func main() {
x:=10
y:=20
z:=x+y
g:=fmt.Sprintf("Total of %d and %d is %d",x,y,z)
fmt.Println(g)
}
```

---

eg81.go

```
package main
import "fmt"
type Element struct {
num int
next *Element
}
func (element *Element) Next() *Element {
return element.next
}
func (element *Element) Value() int {
return element.num
}
type List struct {
start *Element
end *Element
size uint32
}
func (list *List) PushBack(num int) {
element:=new(Element)
element.num=num
if list.start==nil {
list.start=element
list.end=element
} else {
list.end.next=element
list.end=element
}
list.size++
}
func (list *List) Size() uint32{
return list.size
}
func (list *List) PushFront(num int) {
element:=new(Element)
element.num=num
if list.start==nil {
list.start=element
list.end=element
} else {
element.next=list.start
list.start=element
}
list.size++
}
func (list *List) Front() *Element {
return list.start
}
func main(){
var list List
list.PushBack(10)
list.PushBack(20)
list.PushBack(30)
list.PushFront(5)
fmt.Println(list.Size())
for k:=list.Front();k!=nil;k=k.Next() {
fmt.Println(k.Value())
}
}
```

eg82.go

```
package main
import ("fmt";"container/list")
func main(){
var list list.List
list.PushBack(10)
list.PushBack(20)
list.PushBack(30)
for e:=list.Front();e!=nil;e=e.Next() {
fmt.Println(e.Value)
}
fmt.Println(list.Len())
list.Init()
fmt.Println(list.Len())
/*
list of function prototypes of List
type Element  (The node in the List)
   func (e *Element) Next() *Element
   func (e *Element) Prev() *Element
type List      (The data structure )
   func New() *List
   func (l *List) Back() *Element
   func (l *List) Front() *Element
   func (l *List) Init() *List
   func (l *List) InsertAfter(v interface{}, mark *Element) *Element
   func (l *List) InsertBefore(v interface{}, mark *Element) *Element
   func (l *List) Len() int
   func (l *List) MoveAfter(e, mark *Element)
   func (l *List) MoveBefore(e, mark *Element)
   func (l *List) MoveToBack(e *Element)
   func (l *List) MoveToFront(e *Element)
   func (l *List) PushBack(v interface{}) *Element
   func (l *List) PushBackList(other *List)
   func (l *List) PushFront(v interface{}) *Element
   func (l *List) PushFrontList(other *List)
   func (l *List) Remove(e *Element) interface{}
*/
}
```

---

eg83.go

```
package main
import (
"fmt"
"bufio"
"os"
)
func write(){
file,err:=os.OpenFile("abcd.abc",os.O_CREATE | os.O_APPEND | os.O_WRONLY,0600)
if err!=nil {
panic(err)
}
defer file.Close()
writer:=bufio.NewWriter(file)
writer.WriteString("God is great\n")
writer.Flush()
}
func read(){
if _,err:=os.Stat("abcd.abc"); os.IsNotExist(err) {
fmt.Println("File abcd.abc does not exist")
return
}
file,err:=os.OpenFile("abcd.abc",os.O_RDONLY,0600)
if err!=nil {
panic(err)
}
defer file.Close()
scanner:=bufio.NewScanner(file)
for scanner.Scan() {
line:=scanner.Text()
fmt.Println(line)
}
}
func main(){
fmt.Println("----------------------------------------------")
fmt.Println("Reading")
fmt.Println("-----------------------------------------------")
read()
fmt.Println("-----------------------------------------------")
fmt.Println("Writing")
fmt.Println("-----------------------------------------------")
write()
fmt.Println("-----------------------------------------------")
fmt.Println("Reading")
fmt.Println("-----------------------------------------------")
read()
fmt.Println("Done")
}


/*
0600 file – owner can read and write
0700 file – owner can read, write and execute
0666 file – all can read and write
0777 file – all can read, write and execute

    O_RDONLY int = syscall.O_RDONLY // open the file read-only.
    O_WRONLY int = syscall.O_WRONLY // open the file write-only.
    O_RDWR   int = syscall.O_RDWR   // open the file read-write.
    O_APPEND int = syscall.O_APPEND // append data to the file when writing.
    O_CREATE int = syscall.O_CREAT  // create a new file if none exists.
    O_EXCL   int = syscall.O_EXCL   // used with O_CREATE, file must not exist
*/
```

eg84.go

```
package main
import "fmt"
func sam(){
for x:=1;x<=50;x++ {
fmt.Print(x," ")
}
}
func main(){
go sam()
for y:=300;y<=350;y++ {
fmt.Print(y," ")
}
}
```

eg85.go

```
package main
import "fmt"
func sam(){
for x:=1;x<=50;x++ {
```

```
fmt.Print(x," ")
}
}
func main(){
go sam()
for y:=300;y<=350;y++ {
fmt.Print(y," ")
}
var x int
fmt.Scanf("%d\n",&x) // to disallow application to end
}
```

---

eg86.go

```
package main
import (
"fmt"
"time"
)
func sam(){
for x:=1;x<=50;x++ {
time.Sleep(50 * time.Millisecond)
fmt.Print(x," ")
}
}
func main(){
go sam()
for y:=300;y<=350;y++ {
fmt.Print(y," ")
time.Sleep(50 * time.Millisecond)
}
var x int
fmt.Scanf("%d\n",&x) // to disallow application to end
}
```

---

eg87.go

```
package main
import (
"fmt"
"time"
)
func producer(pipeline chan int){
for x:=1;x<=10;x++ {
pipeline <- x
}
}
func consumer(pipeline chan int){
for {
m:= <- pipeline
fmt.Println(m)
time.Sleep(time.Second * 1)
}
}
func main(){
var pipeline chan int=make(chan int)
go producer(pipeline)
go consumer(pipeline)
var x int
fmt.Scanf("%d\n",&x)
}
```

eg88.go

```
package main
import (
"fmt"
"time"
)
func producer1(pipeline chan int){
for x:=1;x<=10;x++ {
pipeline <- x
}
}
func producer2(pipeline chan int){
for x:=51;x<=60;x++ {
pipeline <- x
}
}
func consumer(pipeline chan int){
for {
m:= <- pipeline
fmt.Println(m)
time.Sleep(time.Second * 1)
}
}
func main(){
var pipeline chan int=make(chan int)
go producer1(pipeline)
go producer2(pipeline)
go consumer(pipeline)
var x int
fmt.Scanf("%d\n",&x)
}
// this is just a tip of the iceberg, we will dive in detail in book three
```

eg89.go

```
/*
create table in mysql (database of your choice)
create table Student
(
roll_number int primary key,
name char(50)
)
*/
package main
import (
_ "github.com/go-sql-driver/mysql"
"database/sql"
"fmt"
)
func main(){
db,err:=sql.Open("mysql","root:kelkar@tcp(127.0.0.1:3306)/mydb2017")
if err!=nil {
panic(err)
}
defer db.Close()
statement,err:=db.Prepare("insert into student values (101,'Ram')")
if err!=nil{
panic(err.Error())
}
_,err=statement.Exec()
if err!=nil{
panic(err.Error())
}
fmt.Println("Done")
}
```

OOPS, the above example panicked

Well the package is not installed "github.com/go-sql-driver/mysql"

Now do the following systematically

visit this url and down git setup for windows or LINUX (whatever)
git-scm.com
Install git
Note : While installing
select run from windows command prompt on one installtion UI and
select windows console option on installation another UI

if not set
set the environment variable
GOROOT with value as c:\go
create a folder named as goextra on c:\

set the environment variable
GOPATH with c:\goextra
do the same in LINUX without c:\ and replace \ with /
the GOROOT should be correctly pointing to go installation folder
run git-cmd
(it must be located in C:\Program Files\Git on Windows)
(search for it using locate in Linux/Unix)
on the opened command prompt through (git-cmd) type
go get github.com/go-sql-driver/mysql

Now rerun the eg89 code and data should get inserted

Assignment : write code for update and delete

---

eg90.go (Another technique)

```
package main
import (
_ "github.com/go-sql-driver/mysql"
"database/sql"
"fmt"
)
func main(){
db,err:=sql.Open("mysql","root:kelkar@tcp(127.0.0.1:3306)/mydb2017")
if err!=nil {
panic(err)
}
defer db.Close()
_,err=db.Exec("insert into student values (102,'Rohan')")
if err!=nil{
panic(err.Error())
}
fmt.Println("Done")
}
```

---

eg91.go

```
package main
import (
_ "github.com/go-sql-driver/mysql"
"database/sql"
"fmt"
"os"
)
func main(){
db,err:=sql.Open("mysql","root:kelkar@tcp(127.0.0.1:3306)/mydb2017")
if err!=nil {
panic(err)
}
defer db.Close()
statement,err:=db.Prepare("insert into student values (?,?)")
if err!=nil{
panic(err.Error())
}
_,err=statement.Exec(os.Args[1],os.Args[2])
if err!=nil{
panic(err.Error())
}
fmt.Println("Done")
}
```

---

eg92.go

```
package main
import (
_ "github.com/go-sql-driver/mysql"
"database/sql"
"fmt"
"os"
)
func main(){
db,err:=sql.Open("mysql","root:kelkar@tcp(127.0.0.1:3306)/mydb2017")
if err!=nil {
panic(err)
}
defer db.Close()
_,err=db.Exec("insert into student values (?,?)",os.Args[1],os.Args[2])
if err!=nil{
panic(err.Error())
}
fmt.Println("Done")
}
```

```
package main
import (
_ "github.com/go-sql-driver/mysql"
"database/sql"
"fmt"
)
func main(){
var db *sql.DB
var err error
var rows *sql.Rows
db,err=sql.Open("mysql","root:kelkar@tcp(127.0.0.1:3306)/mydb2017")
if err!=nil {
panic(err)
}
defer db.Close()
rows,err=db.Query("select * from student")
if err!=nil{
```

```
panic(err.Error())
}
defer rows.Close()
var rollNumber int
var name string
for rows.Next(){
err=rows.Scan(&rollNumber,&name)
if err!=nil{
panic(err.Error())
}
fmt.Println(rollNumber,name)
}
err=rows.Err()
if err!=nil{
panic(err.Error())
}
fmt.Println("Done")
}
```

```
/*
create table item
(
code int primary key auto_increment,
name char(50)
)
*/
package main
import (
_ "github.com/go-sql-driver/mysql"
"database/sql"
"fmt"
"os"
)
func main(){
db,err:=sql.Open("mysql","root:kelkar@tcp(127.0.0.1:3306)/mydb2017")
if err!=nil {
panic(err)
}
defer db.Close()
statement,err:=db.Prepare("insert into item (name) values (?)")
if err!=nil{
panic(err.Error())
}
var result sql.Result
var code int64
result,err=statement.Exec(os.Args[1])
if err!=nil{
```

```
panic(err.Error())
}
code,err=result.LastInsertId()
if err!=nil{
panic(err.Error())
}
fmt.Println("Item added with code as : ",code)
fmt.Println("Done")
}
```