Go tutorial:

The commands are:

```
    bug         start a bug report
    build       compile packages and dependencies
    clean       remove object files and cached files
    doc         show documentation for package or symbol
    env         print Go environment information
    fix         update packages to use new APIs
    fmt         gofmt (reformat) package sources
    generate    generate Go files by processing source
    get         add dependencies to current module and install them
    install     compile and install packages and dependencies
    list        list packages or modules
    mod         module maintenance
    work        workspace maintenance
    run         compile and run Go program
    telemetry   manage telemetry data and settings
    test        test packages
    tool        run specified go tool
    version     print Go version
    vet         report likely mistakes in packages
```

Use "go help <command>" for more information about a command.

Additional help topics:

buildconstraint build constraints

buildmode build modes

c calling between Go and C

cache build and test caching

environment environment variables

filetype file types

go.mod the go.mod file

gopath GOPATH environment variable

goproxy module proxy protocol

importpath import path syntax

modules modules, module versions, and more

module-auth module authentication using go.sum

packages package lists and patterns

private configuration for downloading non-public code

testflag testing flags

testfunc testing functions

vcs controlling version control with GOVCS

Input :

```go
package main

import (
    "bufio"
    "fmt"
    "os"
)

func main() {
    reader := bufio.NewReader(os.Stdin)
    input, _ := reader.ReadString('\n')
    fmt.Println(input)
}
```

```go
package main

import (
    "fmt"
)

func main() {
    var input string
    fmt.Print("Enter input: ")
    fmt.Scanln(&input) // Reads input u
    fmt.Println("You entered:", input)
}
```

```
C:\Users\tallu\OneDrive\Documents\go>go run input.go
Enter input: tnr
You entered: tnr
```

Scanf and scan are also allowed

TIME:

```go
package main
import ("fmt"
        "time")
func main(){
    fmt.Println("welcome to study plan")
    ptime:=time.Now()
    fmt.Println("current time",ptime)
    fmt.Println("format time:",ptime.Format("01-02-2006"))
    fmt.Println("format time:",ptime.Format("01-02-2006 15:04:05 monday"))
    create:=time.Date(2004,time.April,5,1,1,1,1,time.Local)
    //time.Date(year int,time.monthname,day int,hour int,min int,sec int,nsec int,time.Location)
    fmt.Println(create)
}
```

```
C:\Users\tallu\OneDrive\Documents\go>go run time.go
welcome to study plan
current time 2024-12-09 17:33:31.7692276 +0530 IST m=+0.000601801
format time: 12-09-2024
format time: 12-09-2024 17:33:31 monday
2004-04-05 01:01:01.000000001 +0530 IST
```

DEFER :

```go
package main
import ("fmt")
func main(){
    defer fmt.Println("world")
    //dfer stop execting that line and excute just befor
    //end curly bracket
    fmt.Println("hello")
}
```

```
C:\Users\tallu\OneDrive\Documents\go>go run defer.go
hello
world
```

```go
package main
import ("fmt")
func main(){
    defer fmt.Println("one")
    defer fmt.Println("two")
    defer fmt.Println("three")
    //last in first out 321
    fmt.Println("hello")
}
```

```
C:\Users\tallu\OneDrive\Documents\go>go run defer2.go
hello
three
two
one
```

FILES:

Ioutils package

```go
package main
import ("fmt"
"os"
"io")
func main(){
    fmt.Println("files")
    content:="tnr from srm"
    file,err:=os.Create("./tnr.txt")
    if err!=nil {
        panic(err)//stop the execution and show the error
    }
    length,err:=io.WriteString(file,content)
    if err!=nil{
        panic(err)
    }
    fmt.Println("length is :",length)
    file.Close()
}
```

```
C:\Users\tallu\OneDrive\Documents\go>go run main.go
files
length is : 12
```

```go
package main

import (
    "fmt"
    "os"
)

func main() {
    file, err := os.ReadFile("./tnr.txt") // Uses os.ReadFil
    if err != nil {
        panic(err)
    }
    fmt.Println(string(file)) // Convert []byte to string
}

```

```
C:\Users\tallu\OneDrive\Documents\go>go run readingfile.go
tnr from srm
```

```go
package main
import (
    "fmt"
    "os"
    "io"
)
func main(){
    file,err:=os.Open("./tnr.txt")
    if err!=nil{
        panic(err)
    }
    defer file.Close()
    content,err:=io.ReadAll(file)
    if err!=nil{
        panic(err)
    }
    fmt.Println(content)
    fmt.Println(string(content))

}
```

```
C:\Users\tallu\OneDrive\Documents\go>go run readingfile.go
[116 110 114 32 102 114 111 109 32 115 114 109]
tnr from srm
```

GET PARAMETER:

The net/http package in Go is used to build HTTP clients and servers. It allows you to make HTTP requests (like GET, POST) and also create simple web servers**.**

The net/http package allows you to make HTTP requests. The most common methods are:

- GET: Retrieve information from the server.
- POST: Send data to the server (like form data or JSON).
- PUT: Update existing resources on the server.
- DELETE: Delete a resource on the server.

```go
package main

import (
    "fmt"
    "io"
    "net/http"
)

const url = "https://api.quicksell.co/v1/internal/frontend-assignment"

func main() {
    fmt.Println("Sending HTTP request...")
    response, err := http.Get(url)
    if err != nil {
        panic(err)
    }
    defer response.Body.Close()
    body, err := io.ReadAll(response.Body)
    if err != nil {
        panic(err)
    }
    fmt.Println(string(body))
}
```

```
C:\Users\tallu\OneDrive\Documents\go>go run http.go
Sending HTTP request...
{"tickets":[{"id":"CAM-1","title":"Update User Profile Page UI","tag":["Feature request"],"userId":"usr-1","status":"Todo","priority":4},{"id":"CAM-2","title":"Add Multi-Language Support - Enable multi-language support within the application.","tag":["Feature Request"],"userId":"usr-2","status":"In progress","priority":3},{"id":"CAM-3","title":"Optimize Database Queries for Performance","tag":["Feature Request"],"userId":"usr-2","status":"In progress","priority":1},{"id":"CAM-4","title":"Implement Email Notification System","tag":["Feature Request"],"userId":"usr-1","status":"In progress","priority":3},{"id":"CAM-5","title":"Enhance Search Functionality","tag":["Feature Request"],"userId":"usr-5","status":"In progress","priority":0},{"id":"CAM-6","title":"Third-Party Payment Gateway","tag":["Feature Request"],"userId":"usr-2","status":"Todo","priority":1},{"id":"CAM-7","title":"Create Onboarding Tutorial for New Users","tag":["Feature Request"],"userId":"usr-1","status":"Backlog","priority":2},{"id":"CAM-8","title":"Implement Role-Based Access Control (RBAC)","tag":["Feature Request"],"userId":"usr-3","status":"In progress","priority":3},{"id":"CAM-9","title":"Upgrade Server Infrastructure","tag":["Feature Request"],"userId":"usr-5","status":"Todo","priority":2},{"id":"CAM-10","title":"Conduct Security Vulnerability Assessment","tag":["Feature Request"],"userId":"usr-4","status":"Backlog","priority":1}],"users":[{"id":"usr-1","name":"Anoop sharma","available":false},{"id":"usr-2","name":"Yogesh","available":true},{"id":"usr-3","name":"Shankar Kumar","available":true},{"id":"usr-4","name":"Ramesh","available":true},{"id":"usr-5","name":"Suresh","available":true}]}
```

http.Get(url) sends a GET request to the URL.

io.ReadAll(response.Body) reads the entire response body.

defer response.Body.Close() ensures the body is closed after reading.

```go
package main
import (
    "fmt"
    "net/url"
)
const urls string="https://tnrdeveloping-hub.web.app/"
func main(){
    fmt.Println("url creation")
    result,err:=url.Parse(urls)
    if err!=nil{
        panic(err)
    }
    fmt.Println(result.Scheme)
    fmt.Println(result.Host)
    fmt.Println(result.Path)
    fmt.Println(result.RawQuery)
    fmt.Println(result.Fragment)
    fmt.Println(result.Port())
}
```

```
C:\Users\tallu\OneDrive\Documents\go>go run url.go
url creation
https
tnrdeveloping-hub.web.app
/
```

JSON DATA:

The encoding/json package in Go is used to encode (serialize) and decode (deserialize) data between Go objects and JSON format. This package allows you to convert Go structs, maps, and slices into JSON strings and vice versa.

This package provides methods like:

- **json.Marshal(v interface{}) ([]byte, error)** — Converts a Go object to JSON (encoding).
- **json.Unmarshal(data []byte, v interface{}) error** — Converts JSON data to a Go object (decoding).

In Go, you typically define a struct to represent the shape of the data you're encoding/decoding. Each field is tagged with **json:"field_name"** to specify the name it will have in the JSON object.

Method	Description
`json.Marshal(v interface{})`	Converts Go object to JSON string.
`json.Unmarshal(data []byte, v interface{})`	Converts JSON string to Go object.
`json.Indent(dst, src []byte, prefix, indent string)`	Indents JSON for pretty printing.
`json.Valid(data []byte)`	Checks if a byte slice is valid JSON.

- Use **json:"field_name"** tags to control JSON key names.
- Use **named struct fields** (like P_ID: 1, P_Name: "Watch") to avoid errors caused by incorrect field positions.
- To work with unknown JSON structures, use **map[string]interface{}**.
- Use **json.Indent()** to make the JSON output more readable.

```go
import (
    "fmt"
    "encoding/json"
)

type watch struct {
    P_ID   int    `json:"p_id"`
    P_Name string `json:"p_name"`
    P_Cost int    `json:"p_cost"`
    P_Cat  string
    P_Desc string `json:"p_desc"`
    P_Img  string `json:"p_img"`
}

func main() {
    fmt.Println("JSON Data:")
    encode()
}

func encode() {
    watches := []watch{
        {1, "Provogue Basic Watch", 599, "Men's Wat
        {2, "Elegant Black Watch", 899, "Unisex Wat
    }
    finaljson, err := json.Marshal(watches)
    if err != nil {
        panic(err)
    }
```

```
C:\Users\tallu\OneDrive\Documents\go>go run json.go
JSON Data:
[{"p_id":1,"p_name":"Provogue Basic Watch","p_cost":599,"P_Cat":"Men's Watch","p_desc":"Provogue Basic Watch with day and date display. Stylish brown strap and white dial.","p_img":"https://rukminim2.flixcart.com/image/612/612/xif0q/watch/e/v/e/1-sk-pg-4078-wyt-brwn-basic-with-day-and-date-display-provogue-original-imahffrywrx3x8zb.jpeg?q=70"},{"p_id":2,"p_name":"Elegant Black Watch","p_cost":899,"P_Cat":"Unisex Watch","p_desc":"Classic black watch with minimalist design, suitable for both men and women.","p_img":"https://rukminim2.flixcart.com/image/612/612/xif0q/watch/6/n/v/-original-imagpzzk4h4eqwmf.jpeg?q=70"}]
```