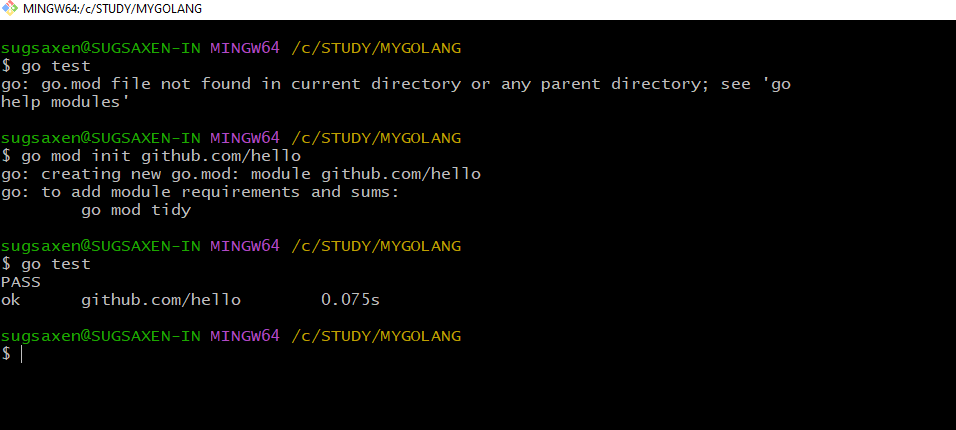
A module is a collection of [Go packages](https://go.dev/ref/spec#Packages) stored in a file tree with a go.mod file at its root. The go.mod file defines:

1. the module’s module path, which is also the import path used for the root directory.
2. Its dependency requirements, which are the other modules needed for a successful build. Each dependency requirement is written as a module path and a specific [semantic version](http://semver.org/).



Please note :

go: creating new go.mod: module github.com/hello

go: to add module requirements and sums:

go mod tidy

The go.mod file only appears in the root of the module. Packages in subdirectories have import paths consisting of the module path plus the path to the subdirectory. For example, if we created a subdirectory world, we would not need to (nor want to) run go mod init there. The package would automatically be recognized as part of the example.com/hello module, with import path example.com/hello/world.

* Now update hello.go with dependency added in code first
* Shows error
* The go.mod before is :

module github.com/hello

go 1.19

go test

sugsaxen@SUGSAXEN-IN MINGW64 /c/STUDY/MYGOLANG

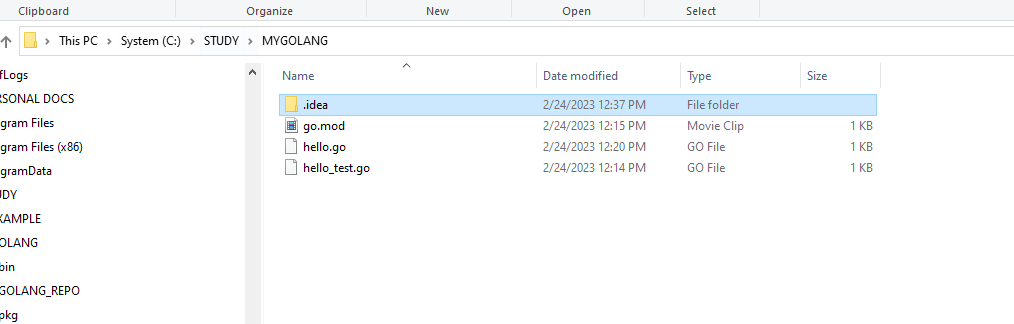
$ go test

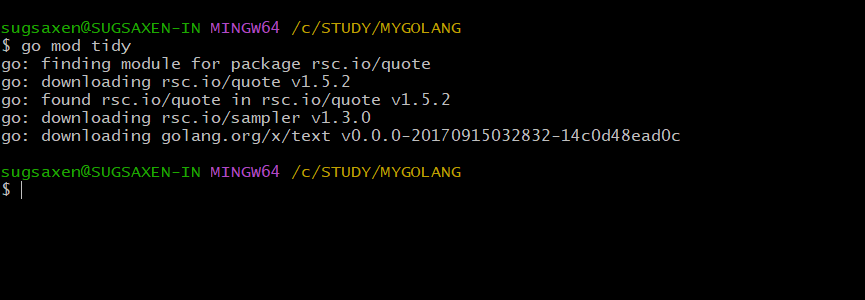
hello.go:3:8: no required module provides package rsc.io/quote; to add it:

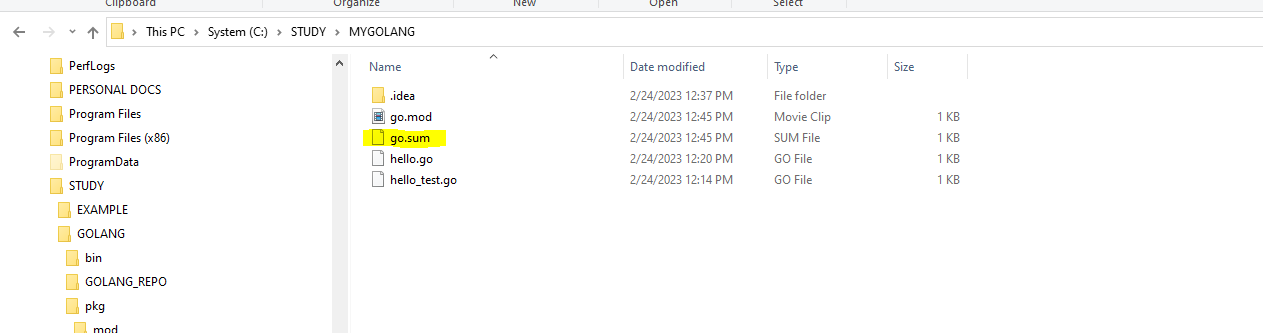
go get rsc.io/quote

Now we can add module requirement and sum :

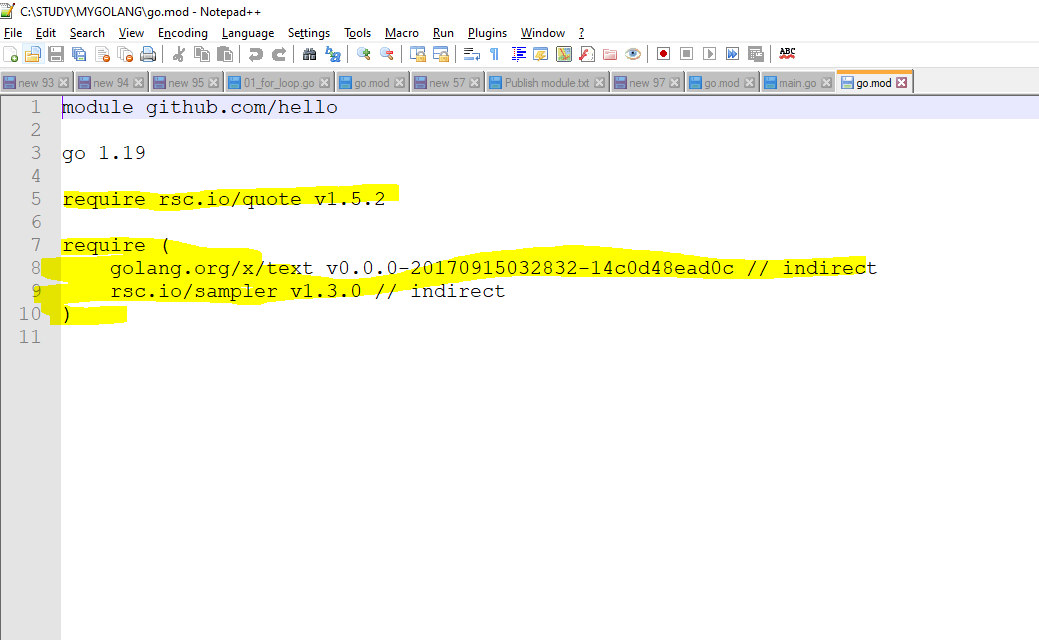
Go mod tidy



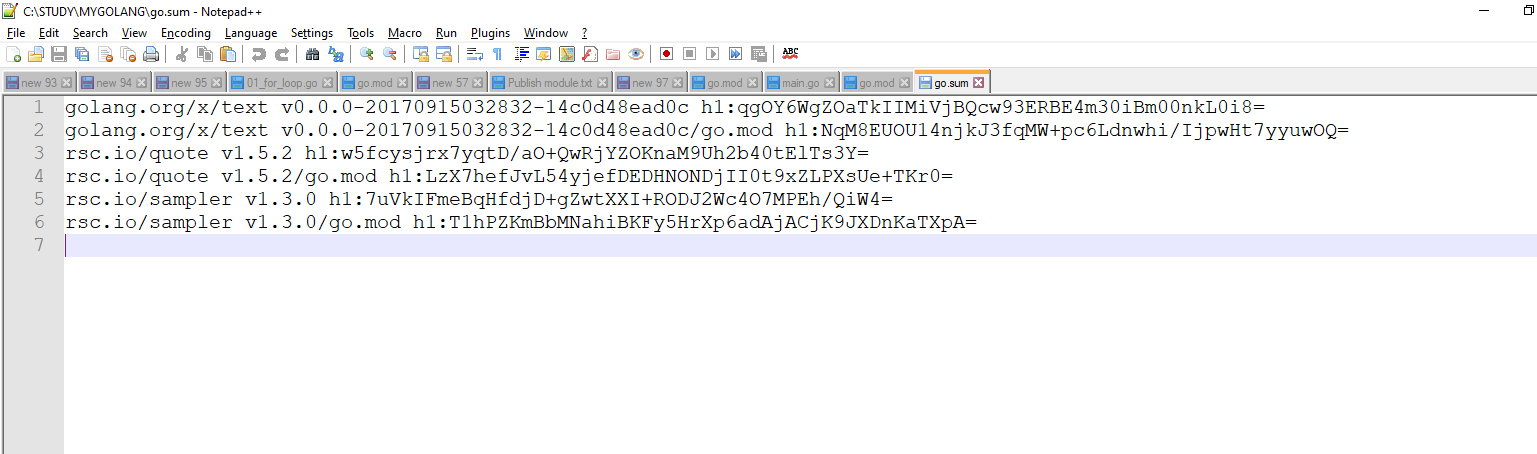




Go mod is modified to include dependency and go.sum added for security



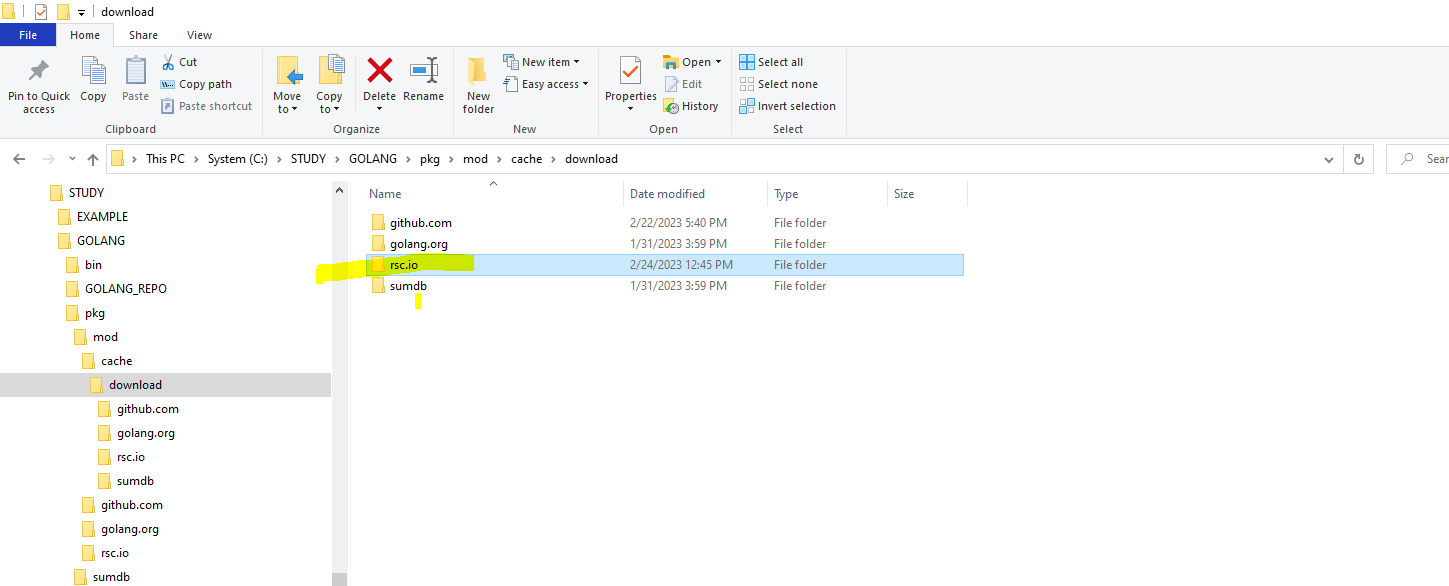
Go sum file :



By mod tidy , dependency was fetched from the web but after that when used in another time it will Not fetched from web , but from your cache directory:

C:\STUDY\GOLANG\pkg\mod\cache\download

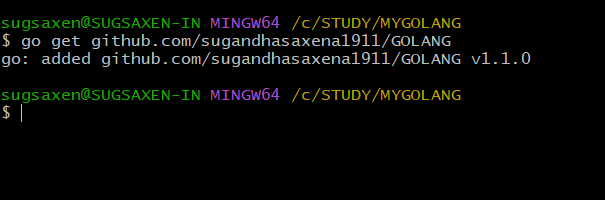
You can see new folder in cache for that



So this time we have used dependency in our code first and then used go get and go mod tidy to install dependencies and update go mod file .

Now this time we will first get dependency , then see what change it had on go.mod file &go.sum file .

After that we will use in the code and then again see change in the go.mod file and go.sum

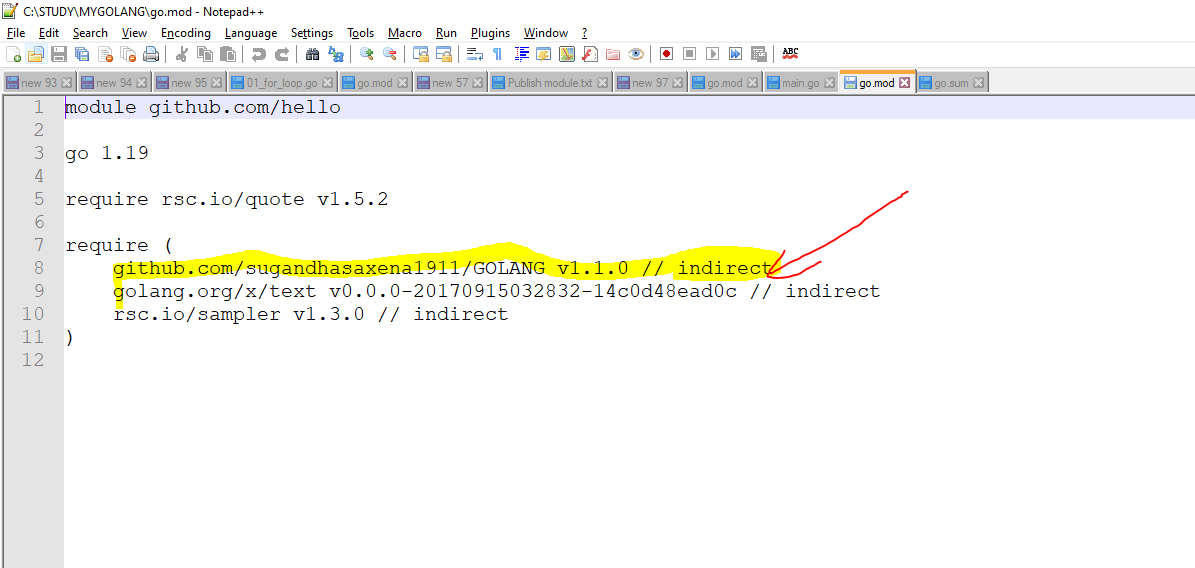


PS: it s better to get the module EXCERCISES rather than repository

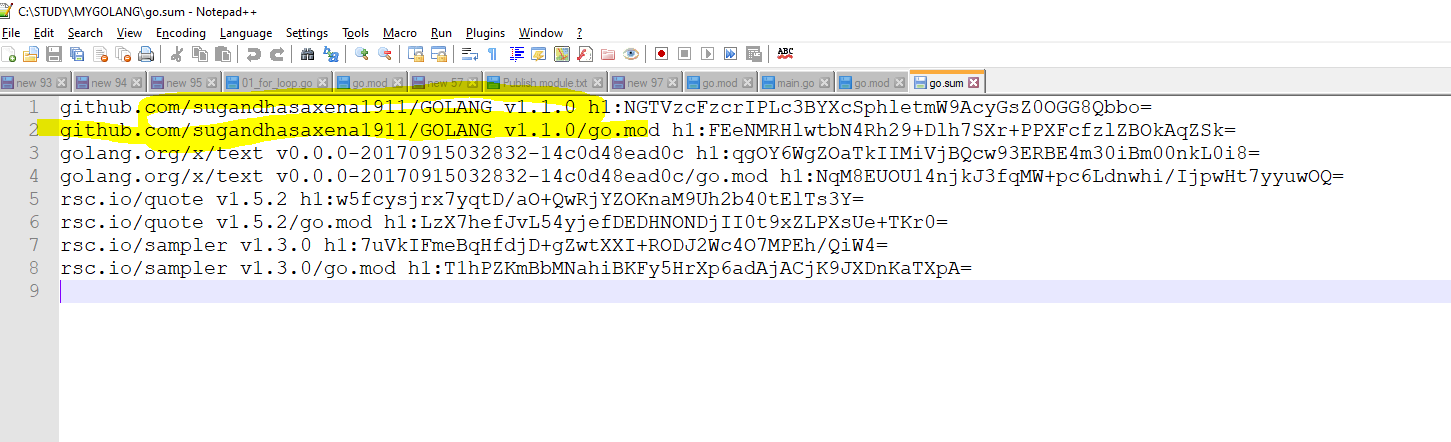
Notice : we got this but didn’t use it anywhere in our code .

So mod file has indirect written on that .

Go sum file would have been modifies as well with hash



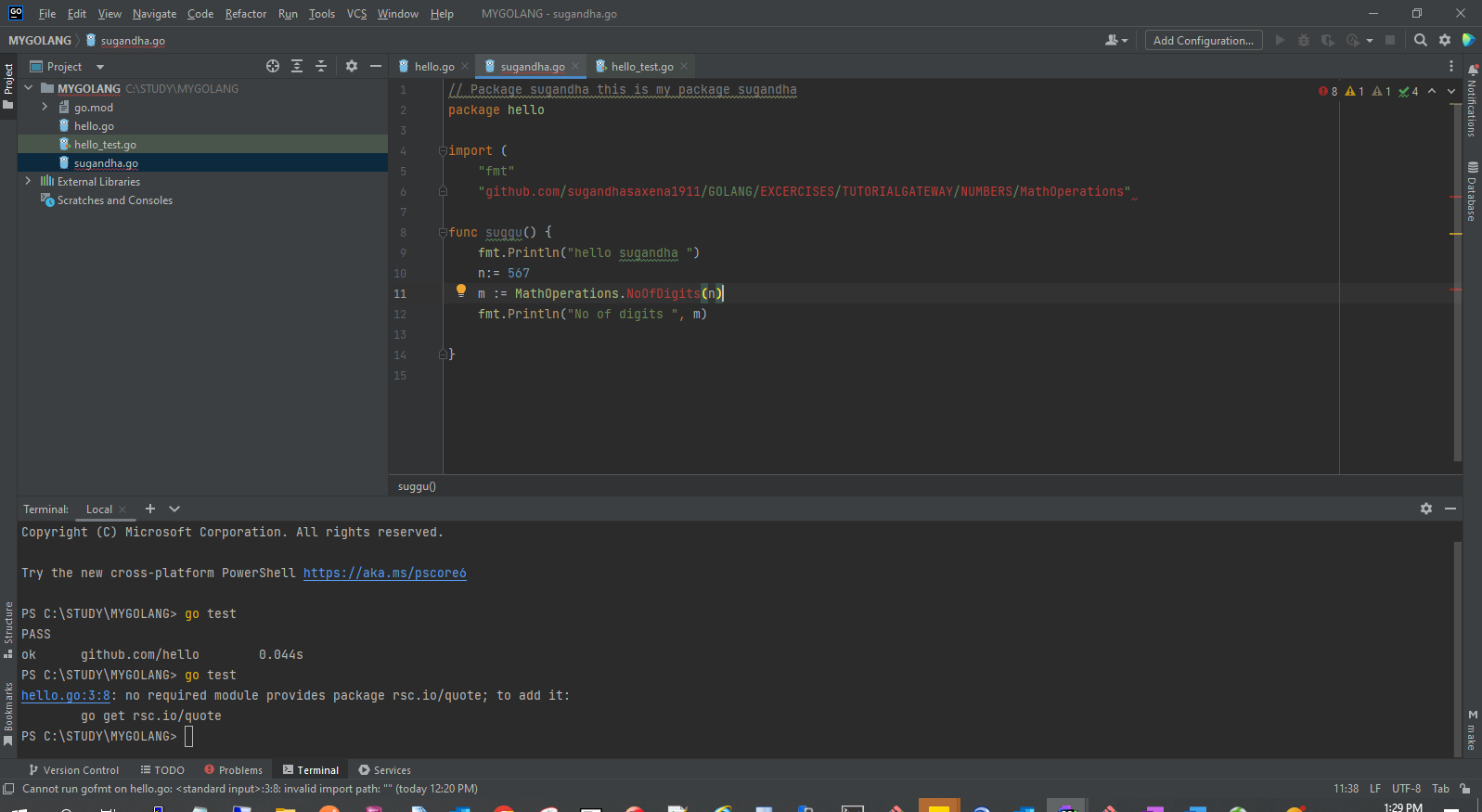
Go sum :



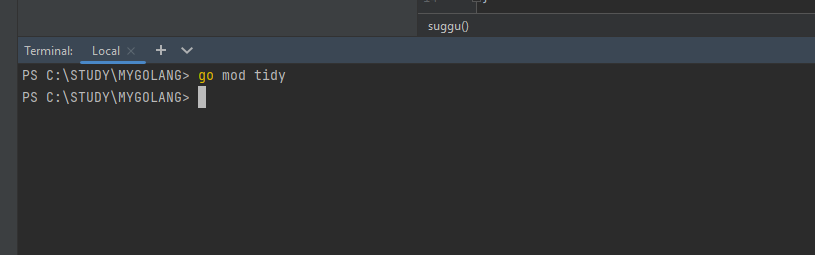
Lets use this dependency in our code

Even if we have added dependency in the mod file , the code still gives issue and it is unable to import the package properly .

Reason : because the mod file still says indirect usage



So , use go mod tidy to update he file

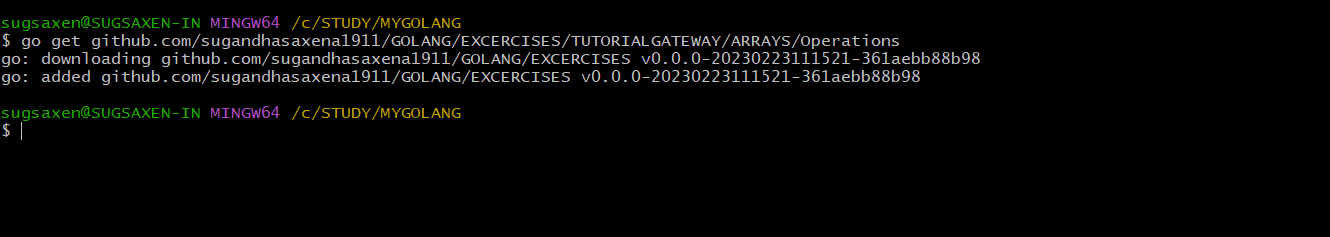


It removed the dependency from mod ??? WHY !!

REASON : we did go get repository path and not the module

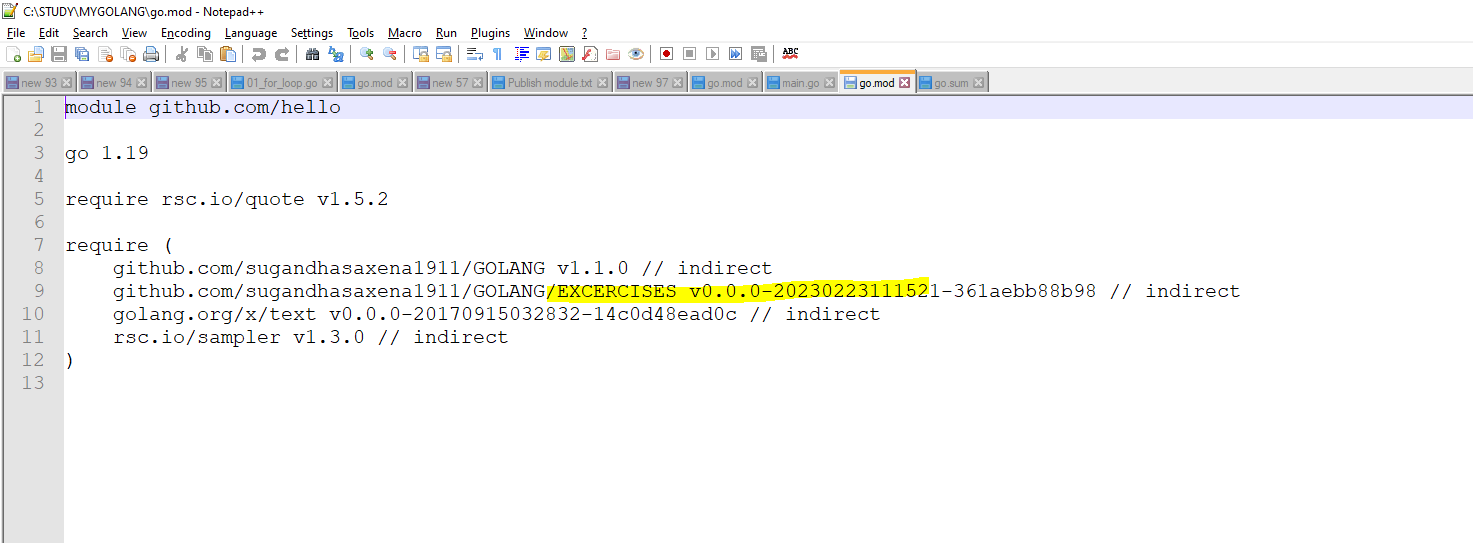
Fine , remove dependency from code .

Go get the exact package or module :

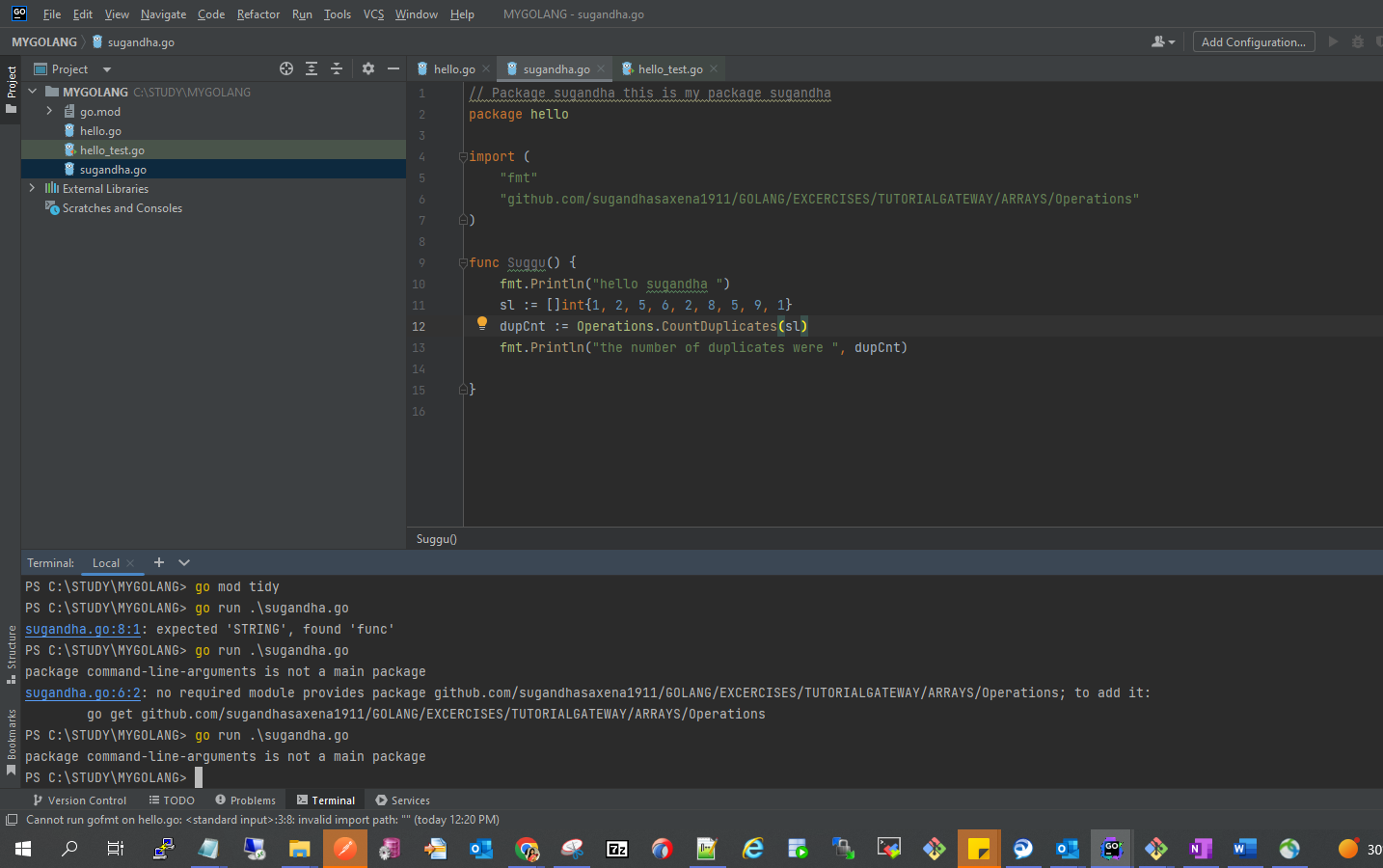


It changes mod file with indirect:

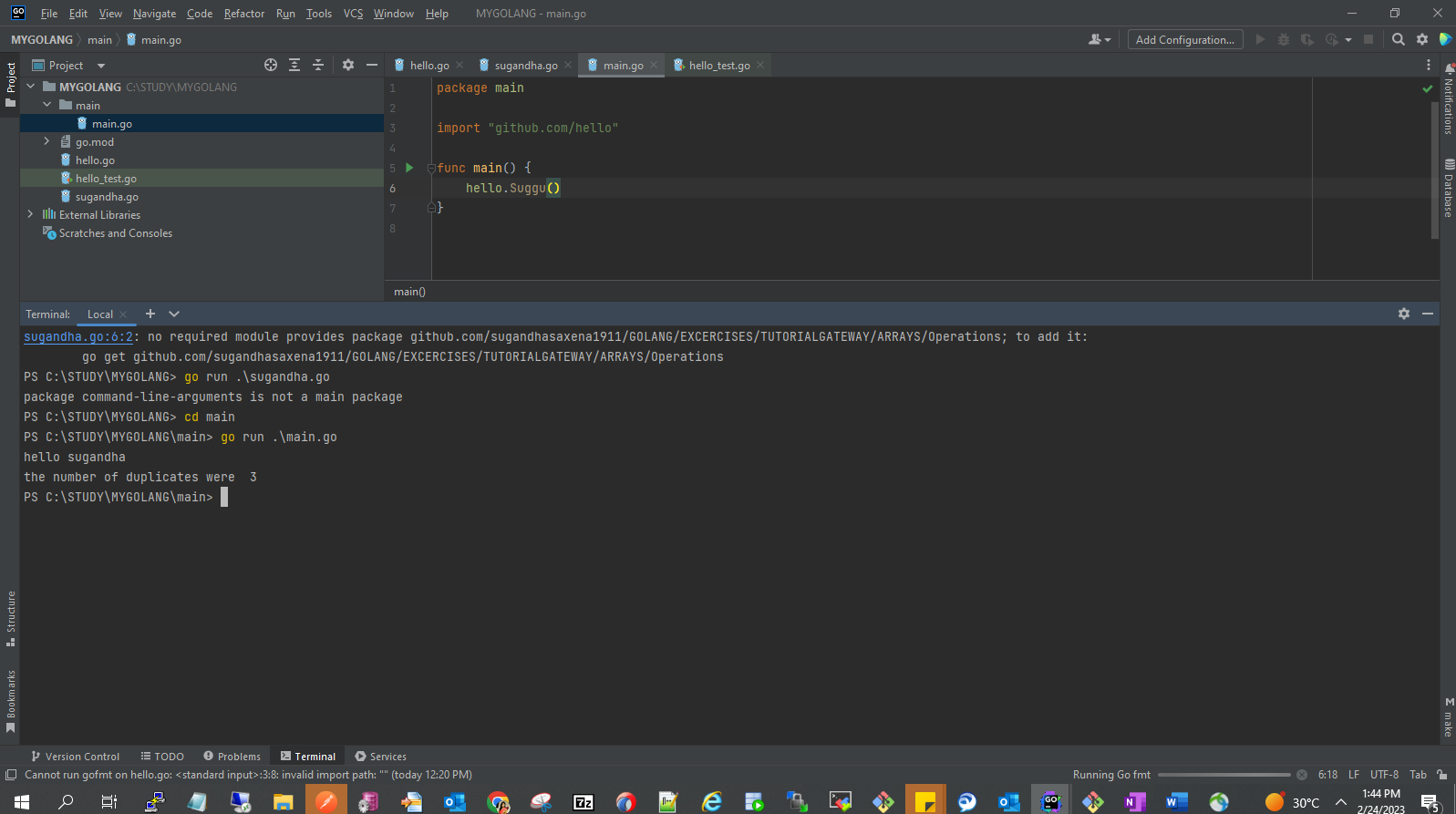
Noe that that the module is updated even if we go get package



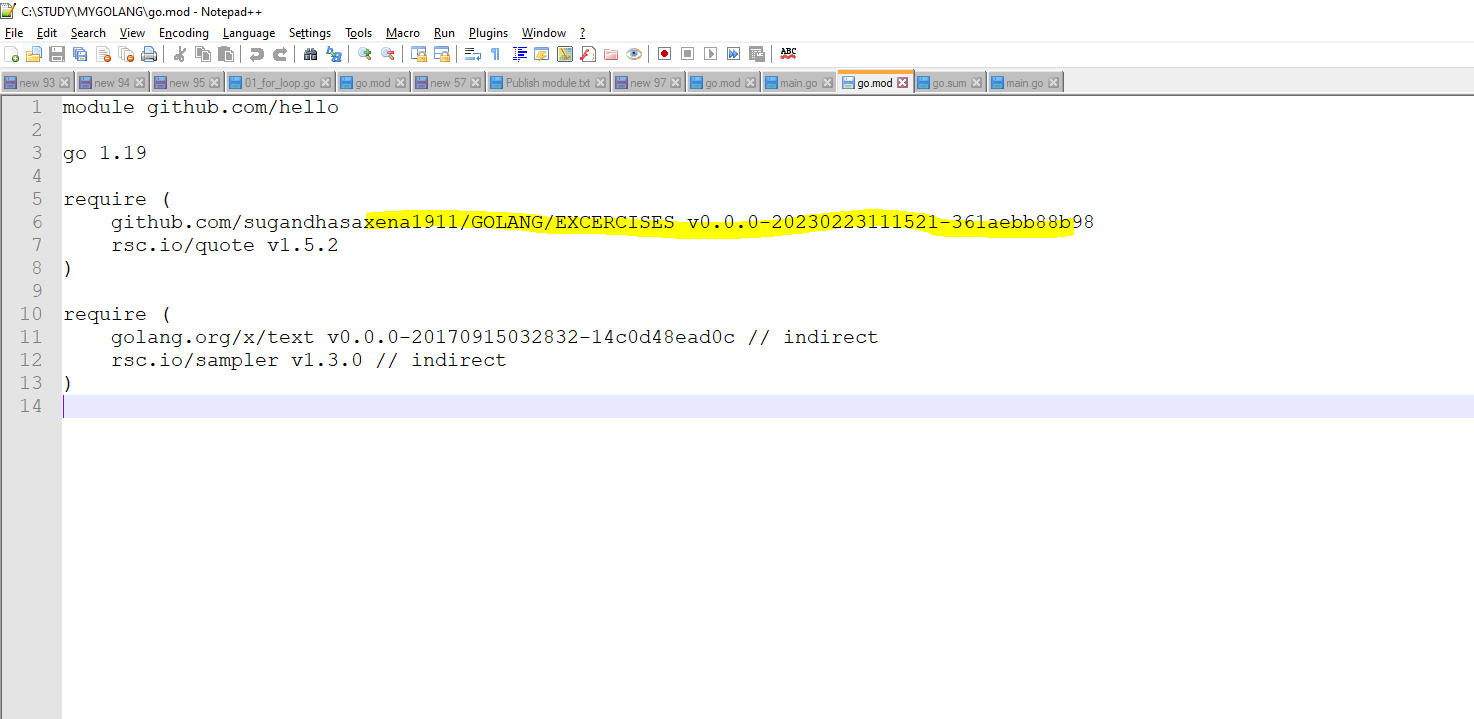
Now , we are able to use the package :



Lets run :



Go mod tidy updates the indirect to direct:

­

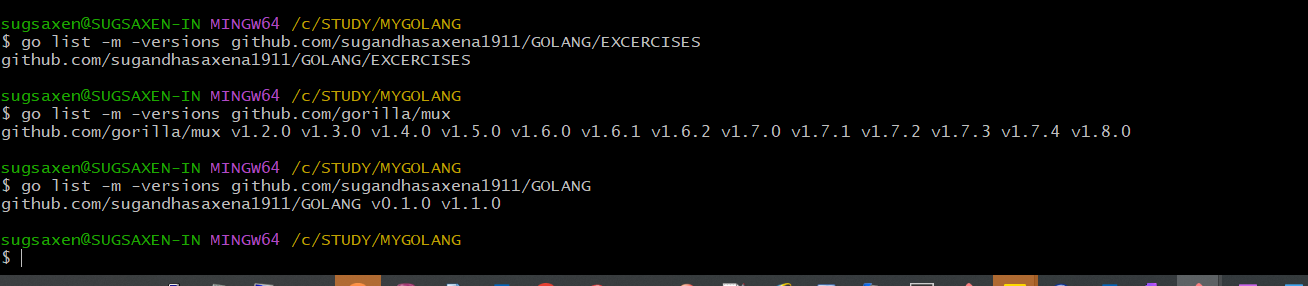
Go mod verify : go to mod file and compare all dependencies with the hash is go sum file and verify that everything is correct

Go list

Go list all

Go list -m all

Go list -m -versions github.com/gorilla/mux



Go mod why

Go mod graph

Go mod vendor: every package/module dependency etc goes into vendor

Go run -mod=vendor main.go