GoLang (Next Gen Language)

Instructor - Hitesh Choudhary

Compiled

GoLang is a Compiled Language. Go tool can run file Directly ,without VM.

Executables are different for OS

What

System Apps to Web Apps-Cloud

Already in Production for different Companies

“It is Just Like C Language written in this Era”

GoLang (Next Gen Language)

-Rabindra Mishra

Object Oriented

Yes or No Both.
Because whatever you see on the screen is the only code

Missing in GO

Try and Catch Statements
Lexer does a Lot of Works.

"Was invented in 2007 by a team at Google"

GoLang Installation

-Rabindra Mishra

Website: go.dev

Can access open playground (remote server)

‘go mod init hello’

Create a module ie. (go.mod named hello)
It is quite case sensitive.

To run go File

go run <File_name>.

“Was designed to combine the speed of development of dynamic languages like Python with the performance and safety of compiled languages like C or C++”

GoLang - My First Code

EXPLORER

GO TUTORIAL

Day1

go.mod

main.go

main.go | go.mod

Day1 > main.go > main

```
1  package main
2  import "fmt"
3
4  func main() {  //there is always one main entry point in Go Language like c/c++
5      fmt.Println("Hello world from Rabindra")
6      fmt.Println("It's fun learning GoLang")
7  }
```

package fmt

fmt on pkg.go.dev

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE

```
PS C:\Users\admin\Desktop\Go Tutorial\Day1> go run main.go
Hello world from Rabindra
It's fun learning GoLang
PS C:\Users\admin\Desktop\Go Tutorial\Day1>
```

GoLang fmt?

-Rabindra Mishra

What is fmt?

fmt is a standard library package used for formatted I/O (input and output). It provides functions to format and print text or to read formatted input

‘go help’

This command gives you docs for any help.

“A clean, minimalistic syntax that avoids unnecessary verbosity. Existing languages often required extensive boilerplate code, slowing down development. ”

go help Command

```
PS C:\Users\admin\Desktop\Go Tutorial\Day1> go help
Go is a tool for managing Go source code.

Usage:

        go <command> [arguments]

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
```

GOPATH

-Rabindra Mishra

What is GOPATH?

GOPATH is an environment variable that specifies the root directory for your Go workspace.

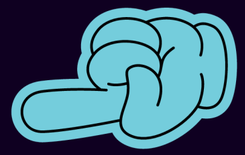
‘ go env GOPATH’

- **src: Contains the source code of your Go projects and libraries, organized by repository paths (e.g., github.com/user/project).**
- **pkg: Stores compiled .a files for package dependencies.**
- **bin: Contains compiled binaries from the go install command.**

“ Go avoids features like inheritance and offers clear, consistent syntax”

go env GOPATH

```
PS C:\Users\admin\Desktop\Go Tutorial\Day1> cd C:\Users\admin\go
PS C:\Users\admin\go> ls
```



```
    Directory: C:\Users\admin\go


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        19-12-2024     01:36                bin
d-----        19-12-2024     01:31                pkg


PS C:\Users\admin\go> cd pkg
PS C:\Users\admin\go\pkg> ls


    Directory: C:\Users\admin\go\pkg


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        19-12-2024     01:32                mod
d-----        19-12-2024     01:31                sumdb


PS C:\Users\admin\go\pkg> cd mod
PS C:\Users\admin\go\pkg\mod> ls


    Directory: C:\Users\admin\go\pkg\mod
```

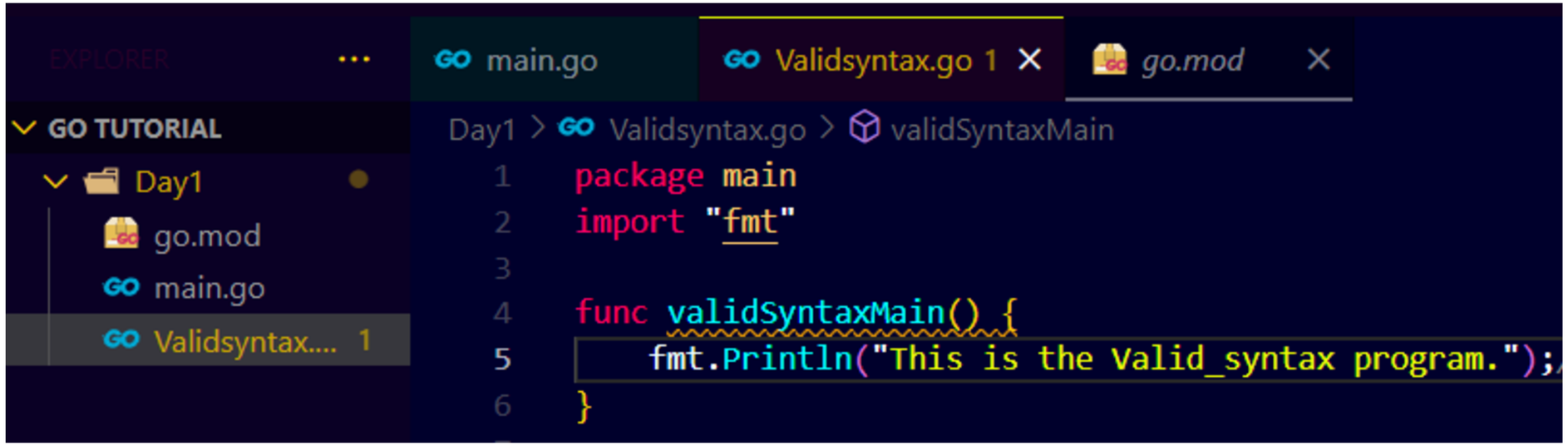
GO

go Directories

-Rabindra Mishra

Go treats all .go files in the same directory with package main as part of a single package.

If you're running only one program, remove or rename the main() function from one of the files.



“Extremely fast compilation, thanks to its simple dependency management and modern compiler design.”

GO - Lexer

-Rabindra Mishra

Lexer

When the code goes into lexer it automatically puts semicolon for you thus at many places you do not actually need to mention semicolon making development faster.

Semicolons

The formal syntax uses semicolons ";" as terminators in a number of productions. Go programs may omit most of these semicolons using the following two rules:

1. When the input is broken into tokens, a semicolon is automatically inserted into the token stream immediately after a line's final token if that token is
    - an identifier
    - an integer, floating-point, imaginary, rune, or string literal
    - one of the keywords `break`, `continue`, `fallthrough`, or `return`
    - one of the operators and punctuation `++`, `--`, `)`, `]`, or `}`
2. To allow complex statements to occupy a single line, a semicolon may be omitted before a closing ")" or "}".

“GOPATH: Points to your workspace, where your projects and third-party dependencies are stored.”

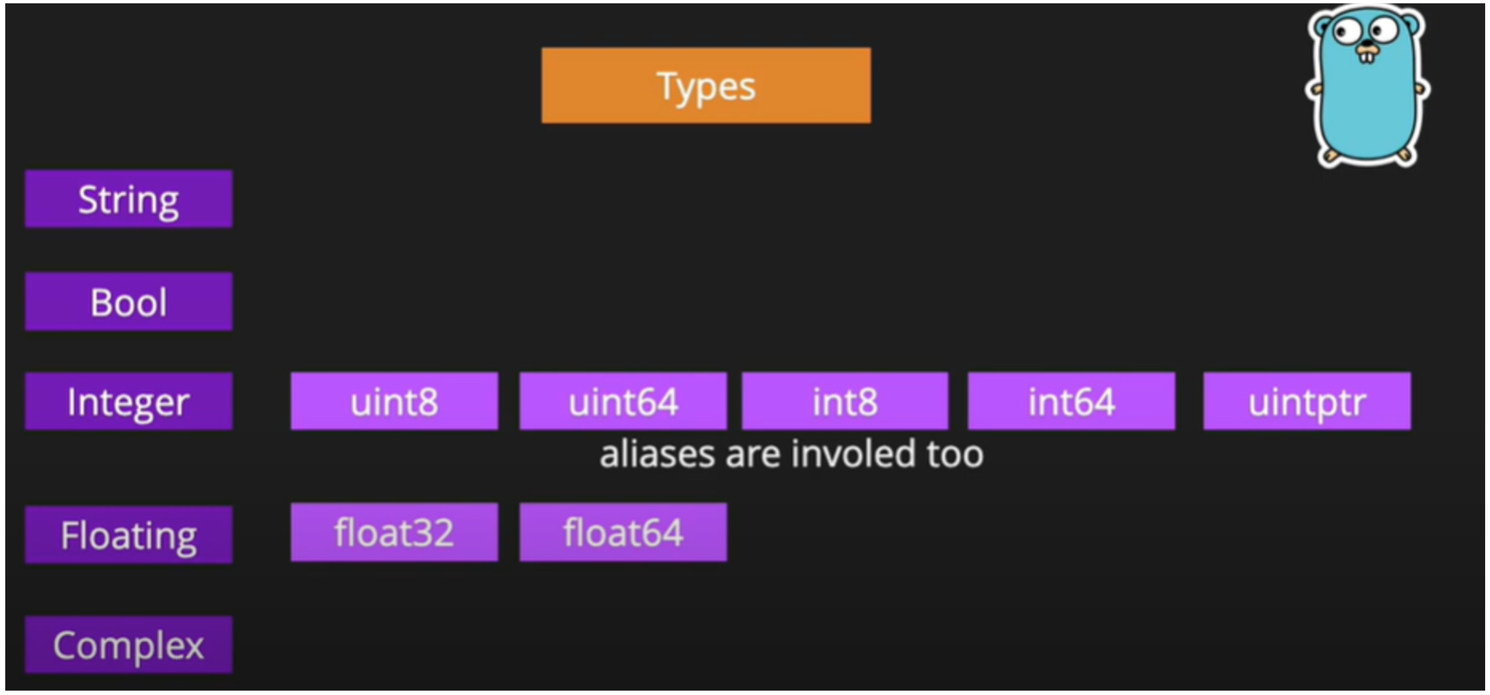
GO - Type

-Rabindra Mishra

Types

- Case Sensitive; almost
- Variable type should be known in advance
- Everything is Type
- For Public type first letter is Uppercase

  eg:Println() which means wherever this Println was defined it can be used publicly.



GO - Type

-Rabindra Mishra

Types

Array

Slices

Maps

Structs

Pointers

Almost everything

Functions

Channels

.....

Variable

-Rabindra Mishra

Variable

- Use var keyword to declare a variable.
- If you declare a variable and don’t use it then lexer finds it inappropriate and may show error.
- eg: var age int=33

“Built-in support for concurrency through goroutines and channels, making it easier to write scalable and concurrent programs.”

Variable:(implicit,no var [walrus operator])

-Rabindra Mishra

```go
func main() {
    var username string = "Rabindra Mishra"
    fmt.Println(username)
    fmt.Printf("Variable username is of type: %T \n",username) //

    var isLoggedin bool = false
    fmt.Println(isLoggedin)
    fmt.Printf("Variable username is of type: %T \n",isLoggedin)

    var smallval uint8 = 255   //Max range of smallval will be 25
    fmt.Println(smallval)
    fmt.Printf("Variable username is of type: %T \n",smallval)

    var value int= 999999
    fmt.Println(value)
    fmt.Printf("Variable username is of type: %T \n",value)

    var price float64= 10.26   //Can also use float32
    fmt.Println(price)
    fmt.Printf("Variable username is of type: %T \n",price)

    // Defaut values and some Aliases
    var rabindra float64   //In this case what be the default flo
    fmt.Println(rabindra)
    fmt.Printf("Variable username is of type: %T \n",rabindra)
```

```go
// Implicit way to declare a

var webiste="Learning.in"
fmt.Println(webiste)


// no var style of declaring
numberofuser:=300
fmt.Println(numberofuser)
numberofuser=2

fmt.Println(Id)  //printing
```

GO-Variable

-Rabindra Mishra

```
import "fmt"
//jwt:=3000 this is
const Id ="12euo"
```

Public Variable?

Here, Id is public variable since “I” is capital

no var method to declare variable

- **Walrus Operator:** Using walrus operator we can declare a variable without var keyword.
- But can only use walrus operator inside a method ie. cannot be used as global operator.

```
// no var style of declaring a varib
numberofuser:=300
fmt.Println(numberofuser)
numberofuser=2
```

Packages in GO

Website: https://pkg.go.dev

-Rabindra Mishra

- **We can explore packages like bufio (Documentation).**
- **In GoLang we usually dont import packages we simply use it.**
- **GoLang does not have try-catch to catch the error thus assumes error to be treated as avariable.**

“Integrated package management via go mod and the Go module system.
Compared To:

- **C/C++: Requires third-party tools like CMake or vcpkg.**
- **Java: Relies on Maven or Gradle.**
- **Python: Uses pip, but dependency conflicts are common.”**

Comma ok || Error

-Rabindra Mishra

```go
func main() {
    welcome := "Welcome our dear Users"
    fmt.Println(welcome)
    reader := bufio.NewReader(os.Stdin) //Note we are reading
    fmt.Println("Enter the rating of the services: ")

    //Comma ok|| error
    //if error is there then it will get store in '_' varibal
    //simply think of this that first part is try varibale an
    input,_:=reader.ReadString('\n')
    fmt.Println("Rating for our given service is",input)
    fmt.Printf("Tye of Rating is:%T",input)  //But there is o

    //Note: If we are not concerned with input then istead of
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

PS C:\Users\admin\Desktop\Go Tutorial> cd 03Userinput
PS C:\Users\admin\Desktop\Go Tutorial\03Userinput> go run main.go
Welcome our dear Users
Enter the rating of the services:
7
Rating for our given service is 7

Tye of Rating is:string
```

Comma ok Syntax

- First part of variable acts like try variable and second part of variable acts like catch variable.

eg:- input,err:=ReadString(‘\n’)

Variable Conversion

-Rabindra Mishra

strconv

- This package is used in go for variable type conversion.
- However we need to perform certain transformations like trim,slice etc.

```
04Conversion > main.go > main
10  func main()  {
18
19      //  Package used for string conversion in Go will be Strconv
20
21      numRating,err:=strconv.ParseFloat(strings.TrimSpace(input),64)
22      if err!=nil{
23       fmt.Println(err)
24       //panic(err) can also be used to stop the program
25      }
26      fmt.Println("Added one to your rating:",numRating+1)
27  }
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL CONSOLE

PS C:\Users\admin\Desktop\Go Tutorial\04Conversion> go run main.go
Welcome to our pizza App
Please rate us between 1 and 5
4
Rating received by user was: 4

Added one to your rateing: 5
```

Advances in GO

-Rabindra Mishra

Go mod V/S Go Path

- Before Go Modules (< Go 1.11) → GOPATH was mandatory.
- Now (Go 1.11+) → Go Modules (go mod) replaced GOPATH, making it optional.
- Today, you don’t need GOPATH if you use Go Modules (go mod).

“A Go Module is a self-contained project that tracks dependencies using a go.mod file. This allows Go projects to exist anywhere on your system, not just inside GOPATH.
”

Advances in GO

-Rabindra Mishra

🔹 Summary: GOPATH vs Go Modules

Feature	GOPATH (Old)	Go Modules (New)
Project Location	Must be in `$GOPATH/src`	Can be anywhere
Dependency Management	Global (`GOPATH/pkg`)	Per-project (`go.mod` , `go.sum`)
Version Control	No versioning	Tracks versions
Builds	Not reproducible	Reproducible

“

1️⃣ More Flexibility → No need to be inside $GOPATH/src.
2️⃣ Works Like Other Languages → Similar to Python (venv) or Node.js (package.json).
3️⃣ Independent Dependencies → Go Modules store dependencies in $GOPATH/pkg/mod, not globally in the project. ”

Time Handling

-Rabindra Mishra

import “time”

- This package is used in go for handling time and various operations.

```
You, 10 minutes ago | 1 author (You)
package main

import ("fmt"
"time")

func main() {
    fmt.Println("Welcome to the Time package!")

    presentTime := time.Now() //Get current time

    fmt.Println("Current Time: ",presentTime)

    //We can also format time as per requirement
    fmt.Println(presentTime.Format("01-02-2006 Monday")) //Note: This

    //Creating a Date
    createdDate:=time.Date(2021,time.April,10,23,0,0,0,time.UTC)
    fmt.Println("createdDate:",createdDate)
}       You, 3 days ago • Working to time …
```

Go for Different OS

-Rabindra Mishra

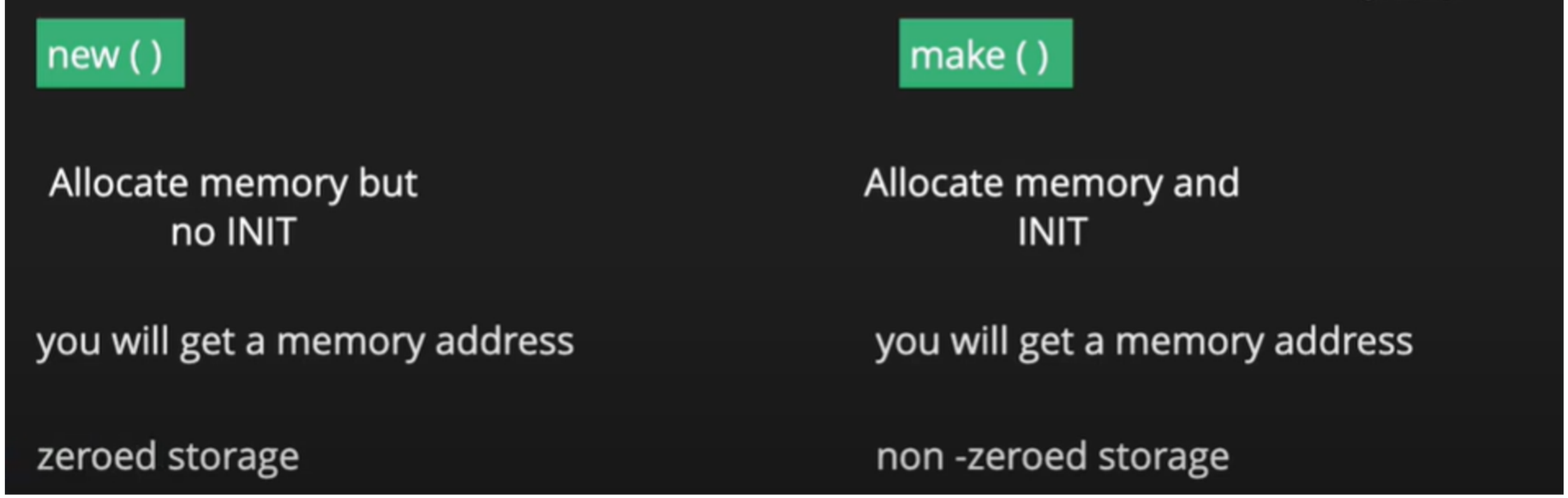
```
PROBLEMS    GITLENS    OUTPUT    DEBUG CONSOLE    TERMINA

PS C:\Users\admin\Desktop\Go Tutorial> go env
set GO111MODULE=
set GOARCH=amd64
set GOBIN=
set GOCACHE=C:\Users\admin\AppData\Local\go-build
set GOENV=C:\Users\admin\AppData\Roaming\go\env
set GOEXE=.exe
set GOEXPERIMENT=
set GOFLAGS=
set GOHOSTARCH=amd64
set GOHOSTOS=windows
set GOINSECURE=
set GOMODCACHE=C:\Users\admin\go\pkg\mod
set GONOPROXY=
set GONOSUMDB=
set GOOS=windows
```

GOOS= “ “

- GOLANG allows us to build .exe for windows,mac,linux etc.

- To build a Golang Application particularly for windows we can write GOOS=”windows” go build



Memory Management

Memory Allocation and Deallocation automatically.

INIT: Initialize

In Zeroed Storage we cannot put any Data

Go for Different OS

-Rabindra Mishra

```
PROBLEMS    GITLENS    OUTPUT    DEBUG CONSOLE    TERMINA
PS C:\Users\admin\Desktop\Go Tutorial> go env
set GO111MODULE=
set GOARCH=amd64
set GOBIN=
set GOCACHE=C:\Users\admin\AppData\Local\go-build
set GOENV=C:\Users\admin\AppData\Roaming\go\env
set GOEXE=.exe
set GOEXPERIMENT=
set GOFLAGS=
set GOHOSTARCH=amd64
set GOHOSTOS=windows
set GOINSECURE=
set GOMODCACHE=C:\Users\admin\go\pkg\mod
set GONOPROXY=
set GONOSUMDB=
set GOOS=windows
```

GOOS= “ “

- GOLANG allows us to build .exe for windows,mac,linux etc.
- To build a Golang Application particularly for windows we can write GOOS=”windows” go build

Pointers in GO

-Rabindra Mishra

```go
import "fmt"

func main() {
    fmt.Println("Lets Learn Pointers") //Sometimes program creates a copy of var

    var ptr *int

    fmt.Println("The value of ptr is:",ptr) //Right now the value of ptr is none

    MyBtech_Rollno:=23
    var ptr1 = &MyBtech_Rollno

    fmt.Println("Address of MyBtech_Rollno is:",ptr1)
    fmt.Println("Value  that ptr1 is refrencing:",*ptr1)

    *ptr1=*ptr1+2
    fmt.Println("New ptr1 value is:",*ptr1) // Thus pointer ensure that operatio

}
```

Arrays in GOLANG

```
> GO main.go > main
package main
import (
    "fmt"
)
func main()  {
    fmt.Println("Welcome to Arrays in GOLANG")

    var myHero [3]string
    myHero[0]="Dr.APJ Abdul Kalam"
    myHero[2]="The Great Ashok Samrat"

    fmt.Println("myHero array:",myHero," ","Length of Array is:",len(myHero))
    and will include space in myHero Array

    var wishlist =[4]string{"AWS","Hadoop/Apache","Linux","RUST"}
    fmt.Println("MY Wishlist is:",wishlist)

}
```

"In GO unlike other Languages we don't use array datatype much due to GO specifications. However, under the Hook other datatypes uses array."

SLICES in GO

```go
package main

import "fmt"

func main() {
    var highscore = []int{10, 45, 78, 62}
    fmt.Println("Highscore:", highscore)

    //Removing a Value from Slice based on index
    var programming = []string{"Python","Ruby","Swift","flutter"}
    index:=2
    programming=append(programming[:index],programming[index+1:]...)
    fmt.Println("Removed indexed 2:",programming)
}
```

“Slices are Arrays with Abstraction Layer and some more added Features”

MAPS

-Rabindra Mishra

```go
func main() {
    languages:= make(map[int]string) //Here I am defin
    languages[0]="Python"
    languages[1]="Ruby"
    languages[2]="Rust"
    languages[3]="PHP"

    fmt.Println("List of Languages:",languages)
    //Acessing value using key
    fmt.Println("Accesing key 2:",languages[2])

    //Deleting a key value pair
    delete(languages,2)

    //for loop in Go
    for i,j := range languages{ //Note: Walrus operato
        fmt.Printf("For key %v -> value is %v\n",i,j)
    }
}
```

“In Go you will find arrays, maps etc will print space separated values instead of comma separated.”

Structs in GO

-Rabindra Mishra

```
You, 5 minutes ago | 1 author (You)
package main

import "fmt"

func main() {       You, 5 minutes ago • Struct in GoLang
    fmt.Println("Understanding structs in Go Lang")
    rabindra := User{"Rabindra","01tcet@gmail.com",20,1500000,true}
    fmt.Println("User struct:",rabindra)

    //for structure we must %+v format specifier
    fmt.Printf("\nDetails of rabindra:%+v",rabindra)
}

You, 5 minutes ago | 1 author (You)
type User struct { //Declare first letter in Capital ie. Struct name "Us
    Name     string //Since if we would be exporting it then must follow
    Email    string
    Age      int
    Salary   int
    Verified bool
}
//Use printf while formating the the output strings
```

"Structs are super important in Go lang because here we don't have classes we use struct. Thus we don't have concept of inheritance in Go lang"

If-else in GO

-Rabindra Mishra

```go
func main() {
    }else{
        result="Unreliable investor"
    }
    fmt.Println("Result:",result)

    if 4%2==0{
        fmt.Println("Number is even")
    }else{
        fmt.Println("Number is odd")
    }

    //In case of API handling we can use this syntax of ini
    if num:=4;num<10{
        fmt.Println("Number is Less than 10")
    }else{
        fmt.Println("Number is more than 10")
    }//This approach is  used to fetch api data and validat
}
```

“Go was designed to be simple and readable, with a minimalistic syntax that makes it easy to learn and use.”

Declaration type

- **Walrus Operator (:=):** Use for local variables within functions for concise declaration and initialization.
- **var with Explicit Type:** Use for package-level variables, when you want to declare without initializing, or when you want to be explicit about the type.
- **var with Type Inference:** Use for package-level variables where you want type inference but prefer using the var keyword for stylistic consistency.

Can you have multiple main() functions?

- ✅ **Yes, if each is in a different main package in different folders, and you're compiling them separately.**
- ❌ **No, if you're trying to run all files together with go run . — Go will look for only one main() in the root main package.**

✅ This will work if and only if:

You're using Go modules (with go mod init yourmodulename)
main1.go is placed in a subfolder named helper/ (not in the same folder as main.go)

🔥 Side-by-Side Comparison:

Feature	`make()`	`new()`
Returns	**Value** (not pointer)	**Pointer**
Used for	slices, maps, channels	all types (structs, arrays, etc)
Initializes?	Yes (fully usable immediately)	No (just zeroes the memory)
Common use case	`make([]int, 10)`	`p := new(MyStruct)`

Because new() just allocates zeroed memory and returns a pointer — it does not initialize internal structures (like slices/maps/channels need).

code:

```
x := new(int)  // allocates int, sets it to 0, returns *int
fmt.Println(*x)  // prints: 0

m := new(map[string]int)
```

Because the map wasn't initialized — it's like trying to write on a paper that doesn't exist yet.

Use make() when working with slices, maps, or channels.
Use new() when you want a pointer to a zero-initialized value of any type.

********** Part 1 :GO **********