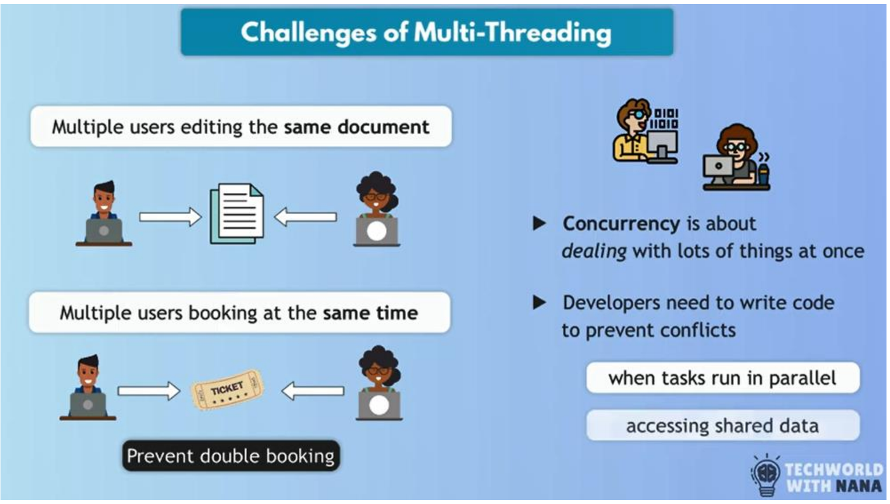


Challenges of Multi-Threading

Multiple users editing the **same document**

Multiple users booking at the **same time**

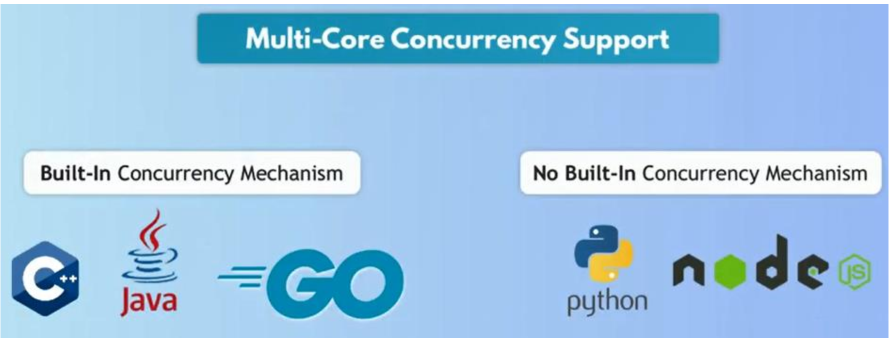
Prevent double booking

- **Concurrency** is about *dealing* with lots of things at once
- Developers need to write code to prevent conflicts

when tasks run in parallel

accessing shared data

****

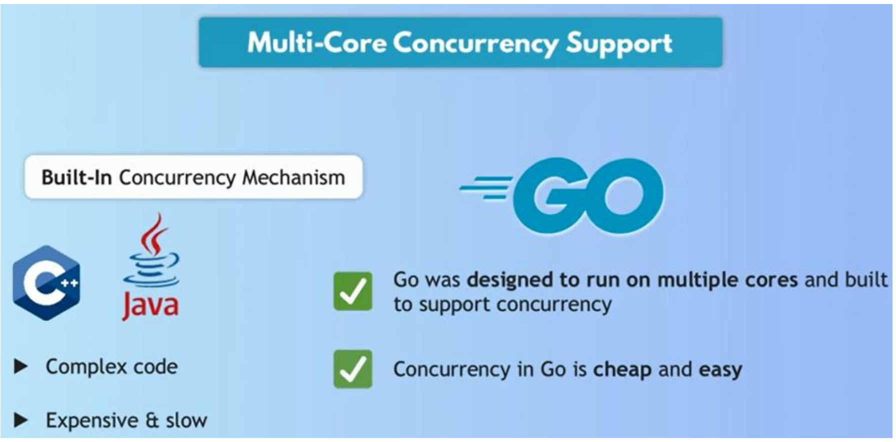


Multi-Core Concurrency Support

Built-In Concurrency Mechanism

No Built-In Concurrency Mechanism

****



Multi-Core Concurrency Support

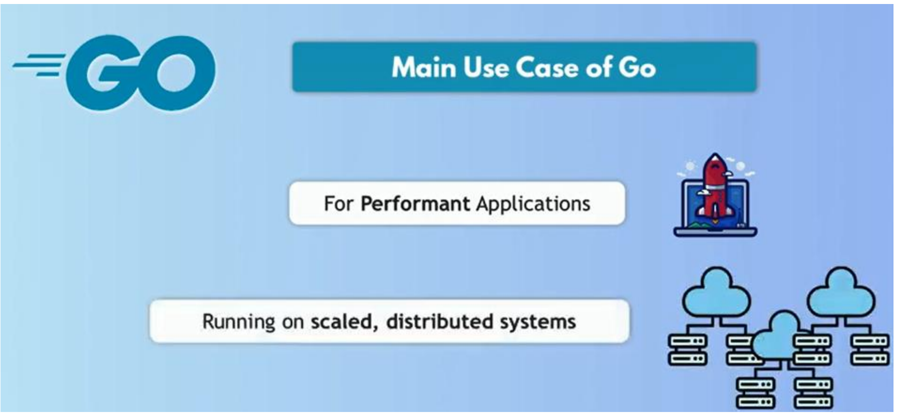
Built-In Concurrency Mechanism

- Complex code
- Expensive & slow

✓ Go was **designed to run on multiple cores** and built to support concurrency

✓ Concurrency in Go is **cheap** and **easy**

****

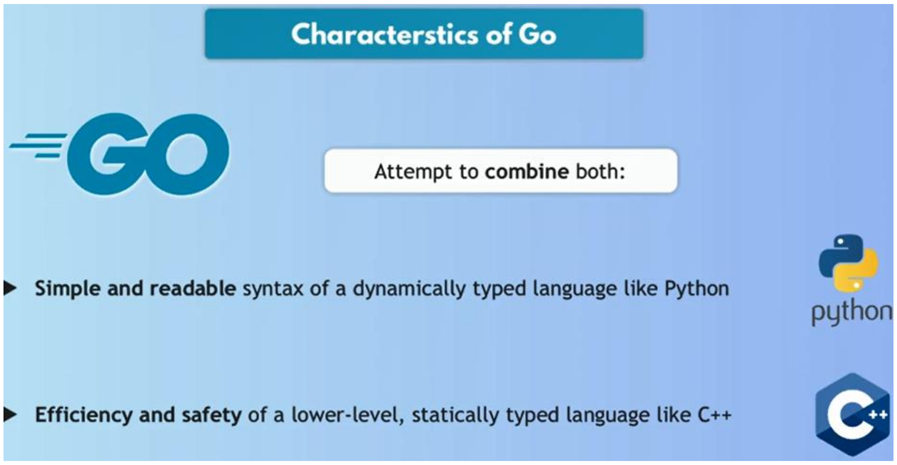


Main Use Case of Go

For **Performant** Applications

Running on **scaled, distributed systems**

****

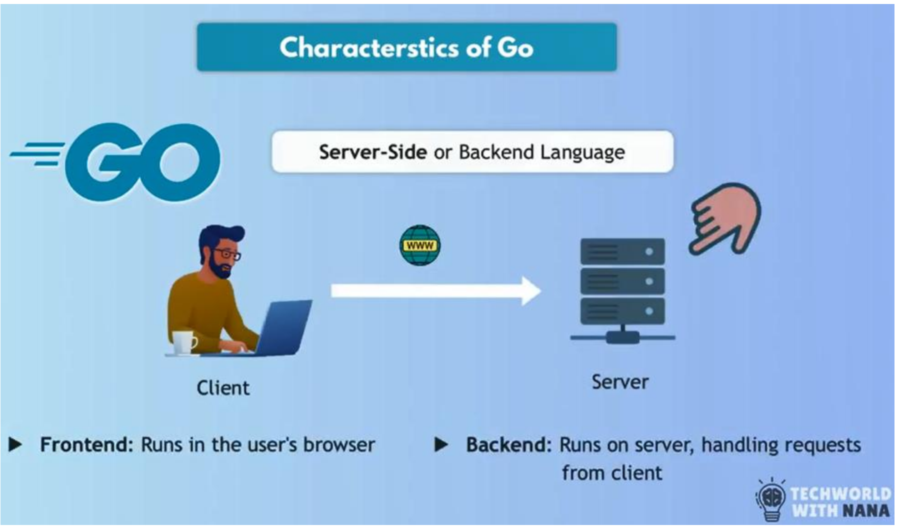


Characterstics of Go

Attempt to **combine** both:

- **Simple and readable** syntax of a dynamically typed language like Python
- **Efficiency and safety** of a lower-level, statically typed language like C++

****

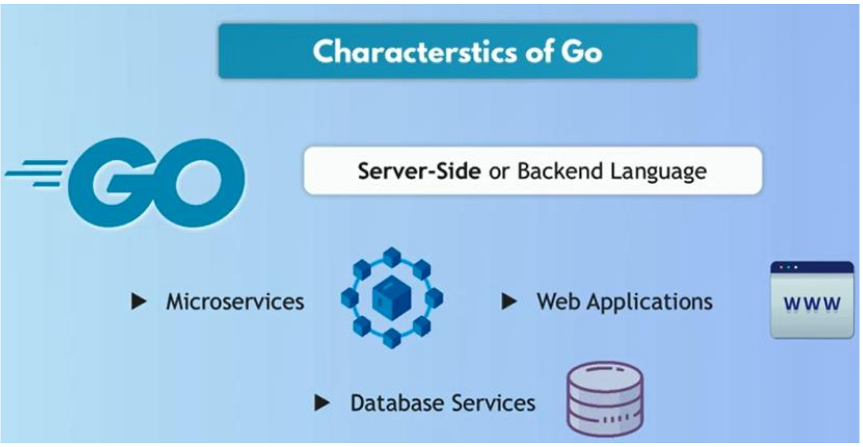


Characterstics of Go

Server-Side or Backend Language

- **Frontend:** Runs in the user's browser
- **Backend:** Runs on server, handling requests from client

****

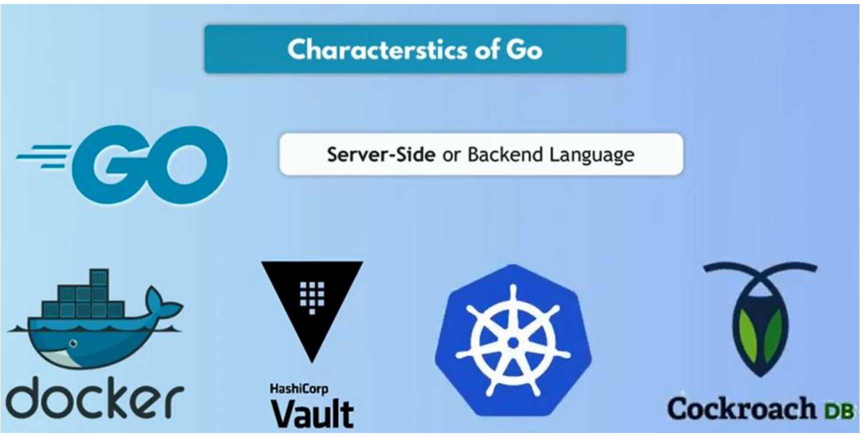


Characterstics of Go

Server-Side or Backend Language

- Microservices
- Web Applications
- Database Services

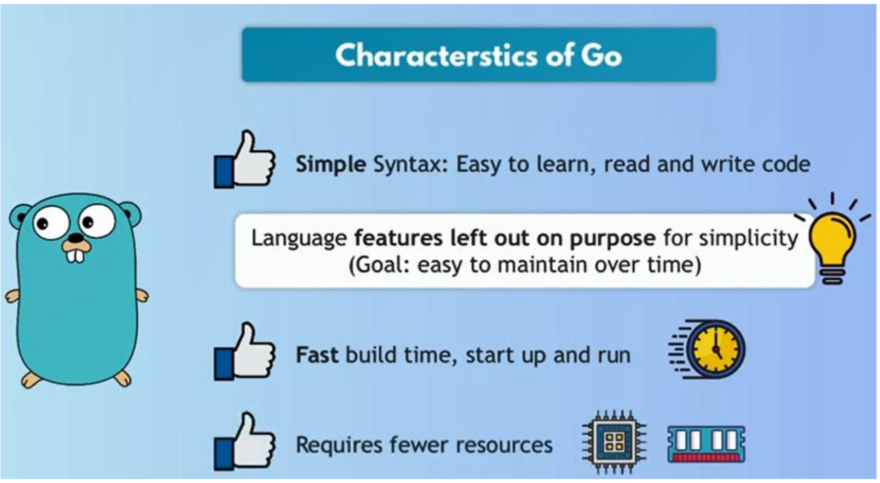
****



Characterstics of Go

Server-Side or Backend Language

****



Characterstics of Go

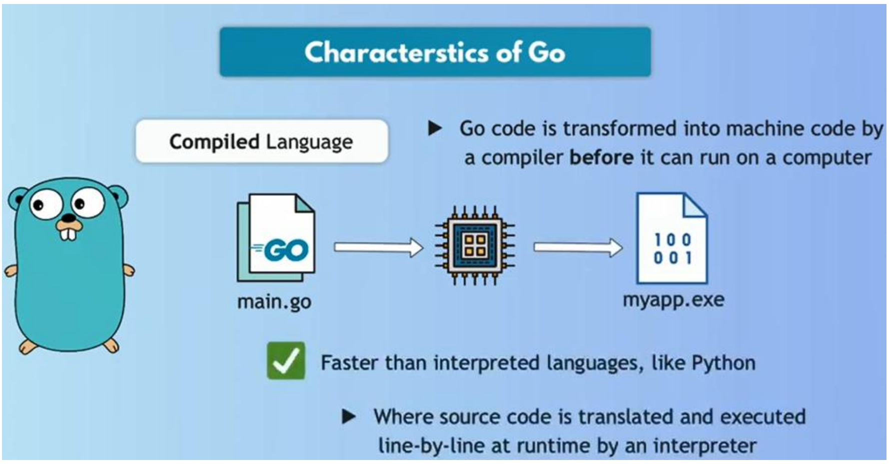
Simple Syntax: Easy to learn, read and write code

Language **features left out on purpose** for simplicity
(Goal: easy to maintain over time)

Fast build time, start up and run

Requires fewer resources

****



Characterstics of Go

Compiled Language

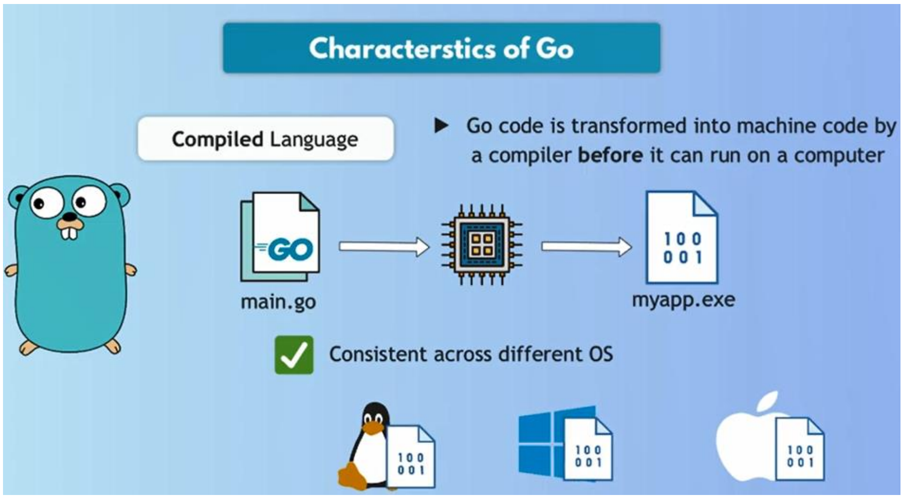
▶ Go code is transformed into machine code by a compiler **before** it can run on a computer

✓ Faster than interpreted languages, like Python

▶ Where source code is translated and executed line-by-line at runtime by an interpreter

****

Run The Same Binary For Different OS:



Characterstics of Go

Compiled Language

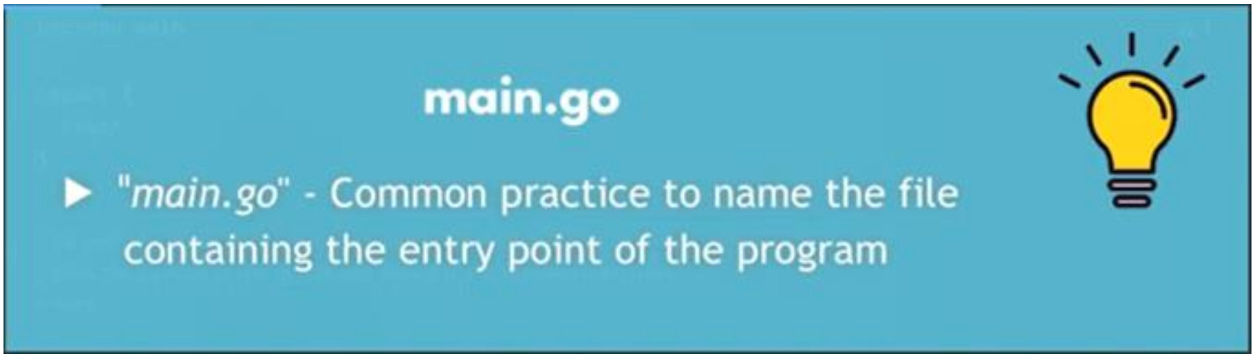
▶ Go code is transformed into machine code by a compiler **before** it can run on a computer

✓ Consistent across different OS

****

Characterstics of Go

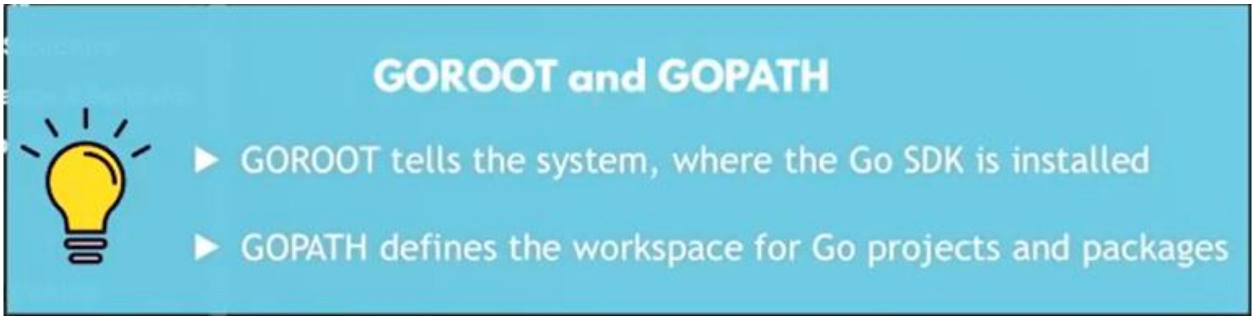
****



main.go

- "*main.go*" - Common practice to name the file containing the entry point of the program

****



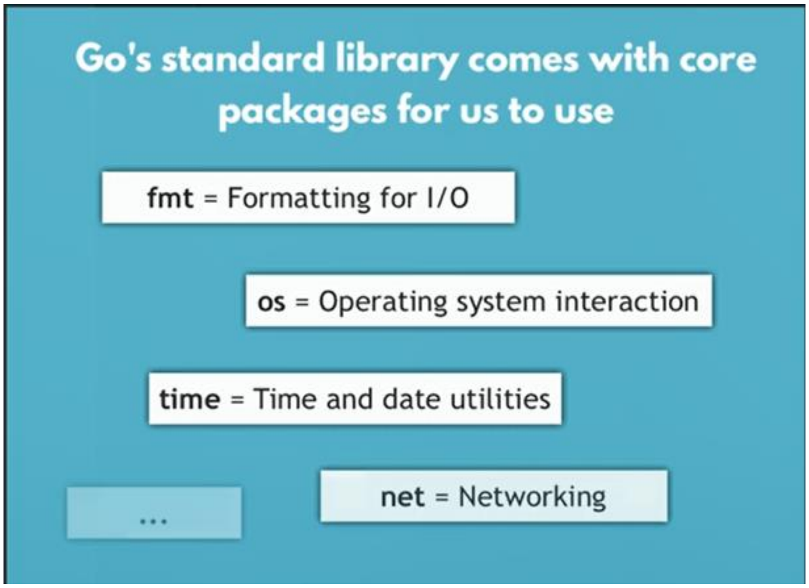
GOROOT and GOPATH

- GOROOT tells the system, where the Go SDK is installed
- GOPATH defines the workspace for Go projects and packages

****

Every Thing In Go Organize in packages

****



Go's standard library comes with core packages for us to use

fmt = Formatting for I/O

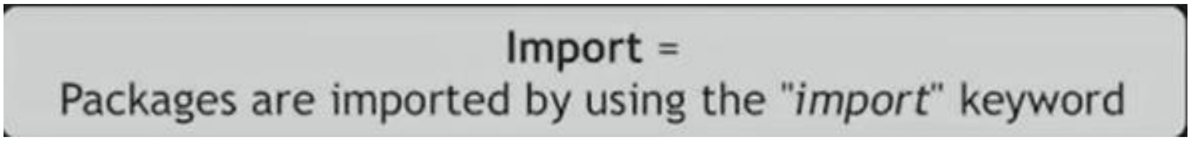
os = Operating system interaction

time = Time and date utilities

net = Networking

...

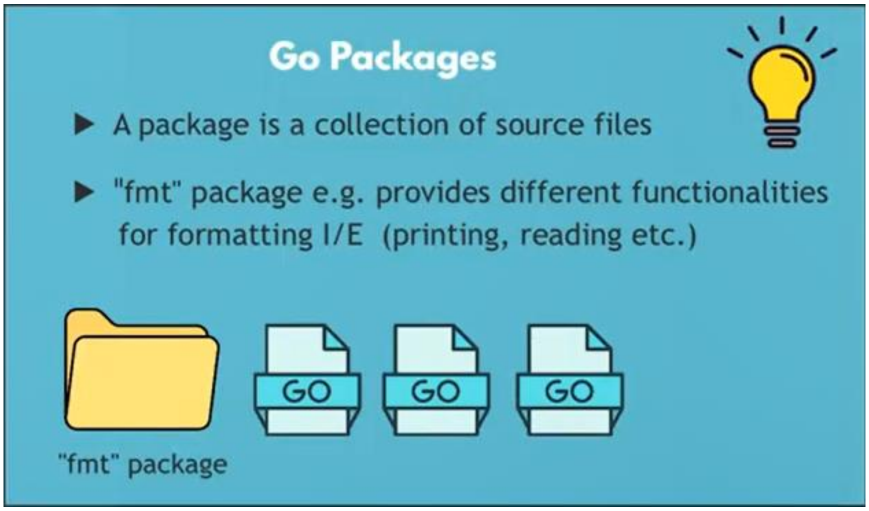
****



Import =

Packages are imported by using the "*import*" keyword

****



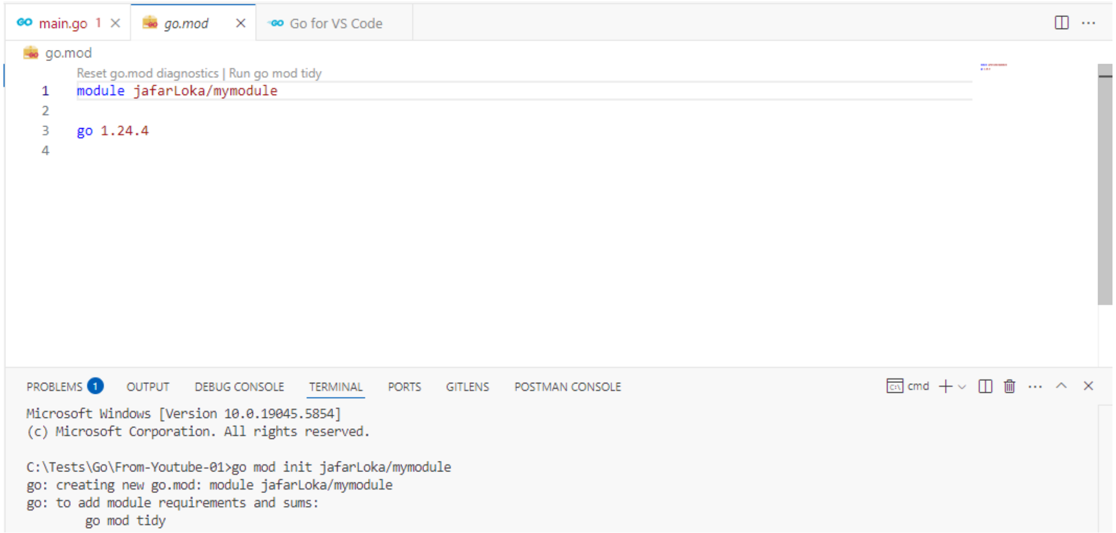
Go Packages

- ▶ A package is a collection of source files
- ▶ "fmt" package e.g. provides different functionalities for formatting I/E (printing, reading etc.)

****

For Creating New Mod File For Go-Project, We Run Command:

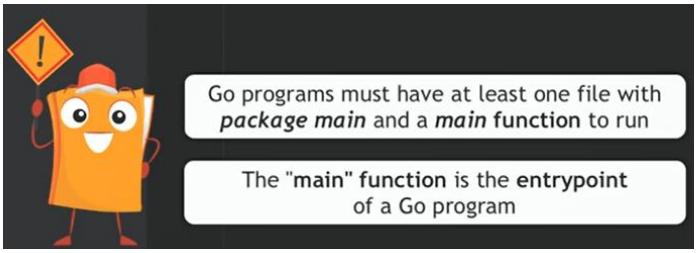
- Command: *go mod init companyName/ModuleName*
  - **Example**: *go mod init jafarLoka/mymodule*



```
module jafarLoka/mymodule

go 1.24.4

```

```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Tests\Go\From-Youtube-01>go mod init jafarLoka/mymodule
go: creating new go.mod: module jafarLoka/mymodule
go: to add module requirements and sums:
        go mod tidy
```

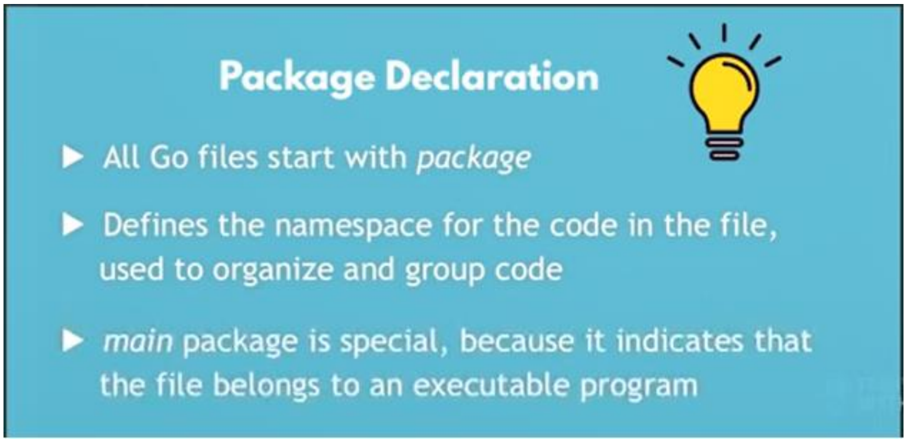
****



Go programs must have at least one file with *package main* and a *main* **function** to run

The "**main**" **function** is the **entrypoint** of a Go program

****



Package Declaration

- All Go files start with *package*
- Defines the namespace for the code in the file, used to organize and group code
- *main* package is special, because it indicates that the file belongs to an executable program

****

To Run The Program: *go run main.go*

To Compile The Program: *go build main.go*

****

Functions In Go, Starts With: **func**

****

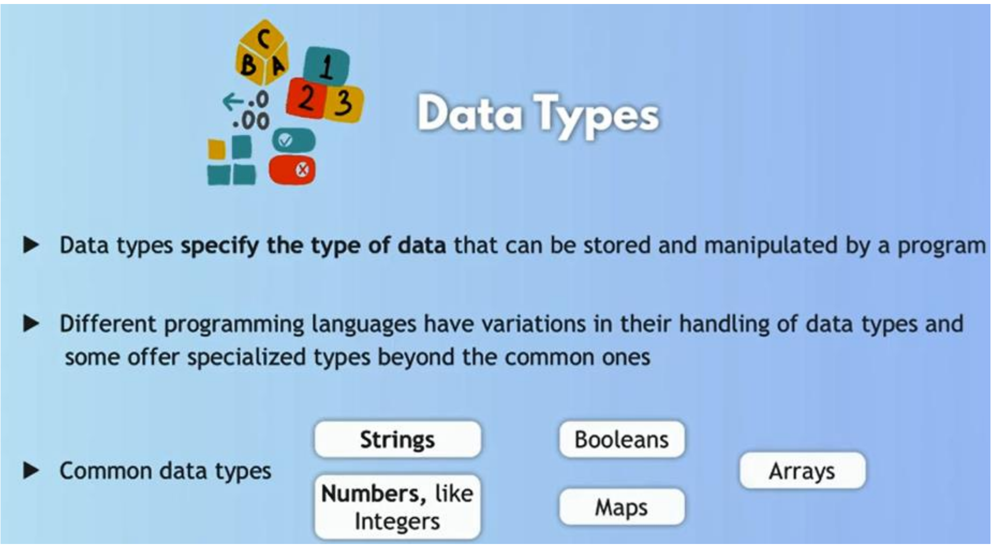
The basic I/O library in Go is: **fmt**

****

The Basic Structure For Simple Go Program Is:

```
package main

import "fmt"
func main() {
    fmt.Println("Welcome to our Todolist App!")
}
```

****



Data Types

- Data types **specify the type of data** that can be stored and manipulated by a program
- Different programming languages have variations in their handling of data types and some offer specialized types beyond the common ones
- Common data types

Strings

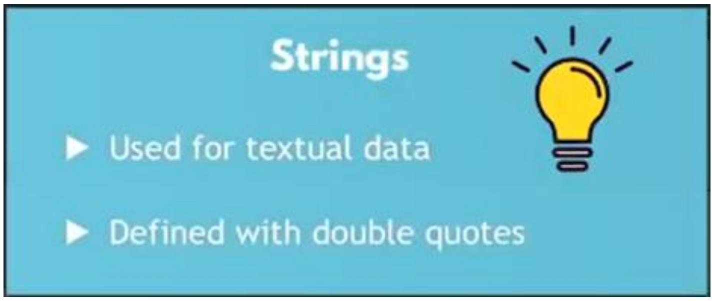
Numbers, like Integers

Booleans

Maps

Arrays

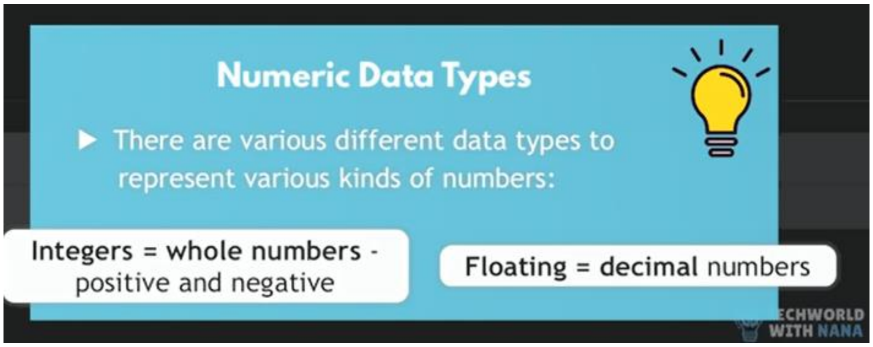
****



Strings

- Used for textual data
- Defined with double quotes

****



Numeric Data Types

- There are various different data types to represent various kinds of numbers:

Integers = whole numbers - positive and negative

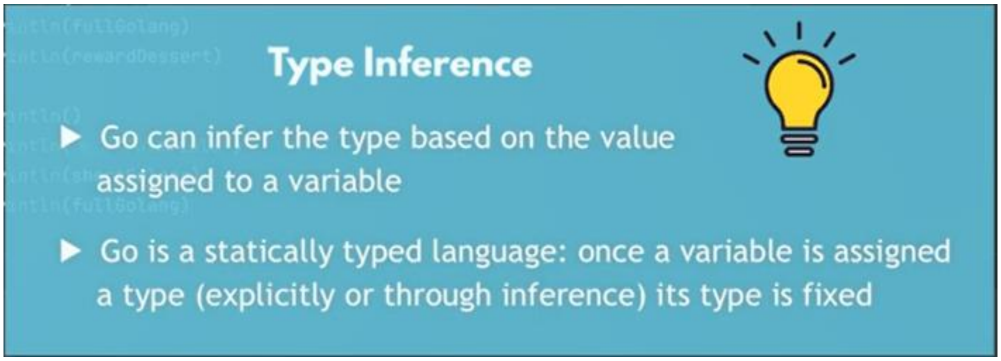
Floating = decimal numbers

****

To use Variables In go, We Declare Them Using *var-keyword*.

```
var task01 = "1. Watch the crash course of Go"

var task02 = "2. Build network automation tools with Go"

var task03 = "3. Build Microservices Applications With gRPC"

var task04 = "4. Build my Applications using Go"
```

****



Type Inference

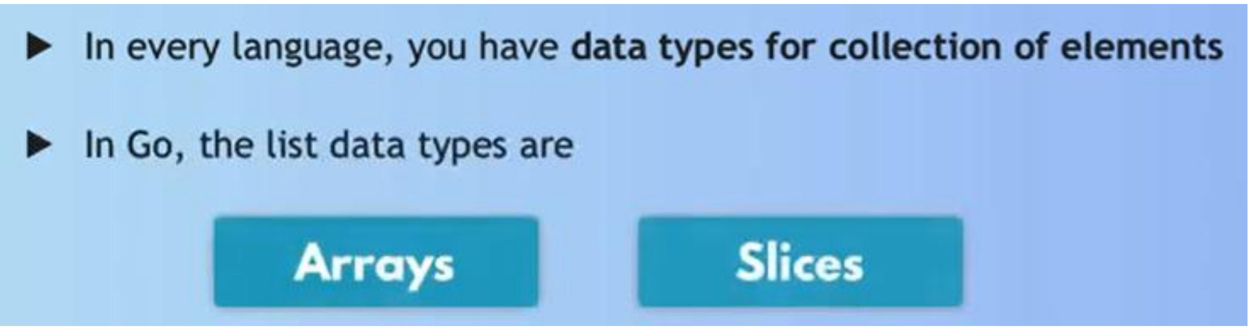
- Go can infer the type based on the value assigned to a variable
- Go is a statically typed language: once a variable is assigned a type (explicitly or through inference) its type is fixed

****

The new way to declare the *go-variables* is by using: **:=**

```
task05 := "5. Test new type of declaring variables"

fmt.Println("*** New Variable Declaration ***")
fmt.Println(task05)
```

****



- In every language, you have **data types for collection of elements**
- In Go, the list data types are

Arrays

Slices

****

Arrays and Slices in Go

- Can hold only elements of the same type

****