Kharagpur Open Source Society

FOSS UNITED

# GO lang Workshop

Speakers: Jai, Akash, Yogansh

# Follow Along!

# TABLE OF CONTENTS

01

Installing Go

02

Advanced Topics

03

Basics

04

A RESTful API

# But before all that, why Go?

- Go was designed at Google in 2007 to improve programming productivity.

- The designers wanted to address pitfalls they found in other languages in use at Google

# But before all that, why Go?

- Fast Compilation Time
- Efficient Execution
- Readability and Ease of use
- Static Typing
- Lightweight and easy to use Concurrency

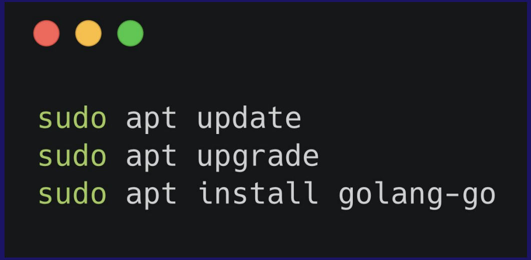
# 01

# Installing Go

# Installing Go

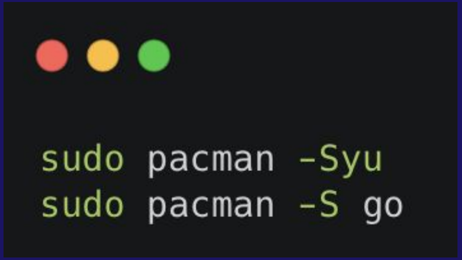
- Follow the instructions on https://go.dev/doc/install corresponding to your operating system
- Alternatively:

On Ubuntu



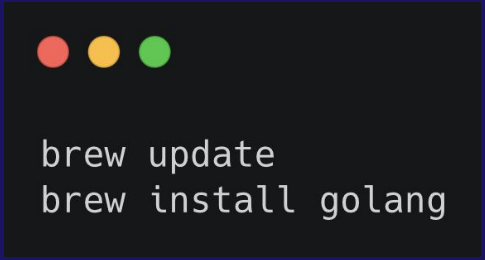
```
sudo apt update
sudo apt upgrade
sudo apt install golang-go
```

On Arch



```
sudo pacman -Syu
sudo pacman -S go
```

On MacOS



```
brew update
brew install golang
```

# Installing Go

- Facing issues?
- Don’t worry! You can follow along by using Go’s own online playground at https://go.dev/play

# 02

# Basics of Go

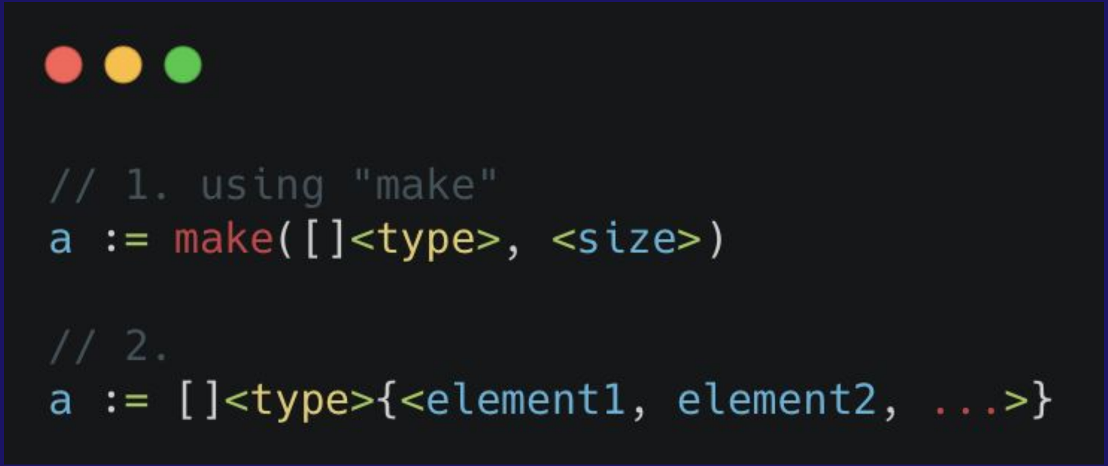
# Slices

# Slices

- Slices provide a convenient and efficient means of working with sequences of data.
- They are similar to arrays in other languages (more on that later!), but have some special properties.
- The type []T is a slice with elements of type T

# Slices

- You can create a slice in the following ways:

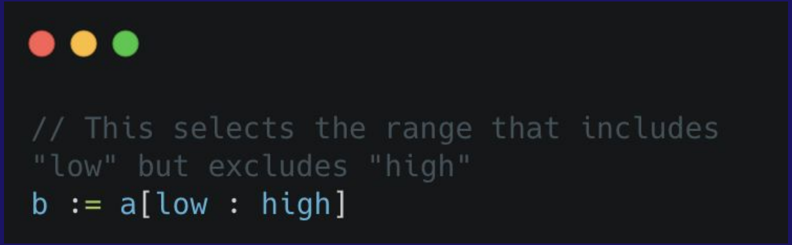


```
// 1. using "make"
a := make([]<type>, <size>)

// 2.
a := []<type>{<element1, element2, ...>}
```

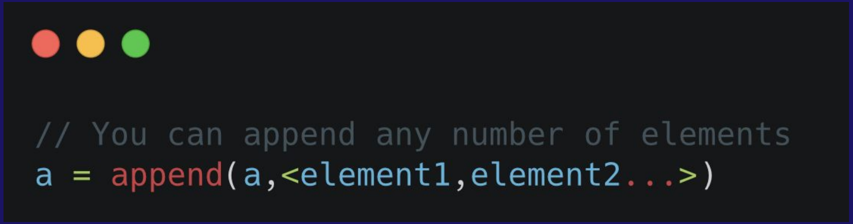
# Slices

- You can “slice” a slice in the following way:



```
// This selects the range that includes
"low" but excludes "high"
b := a[low : high]
```

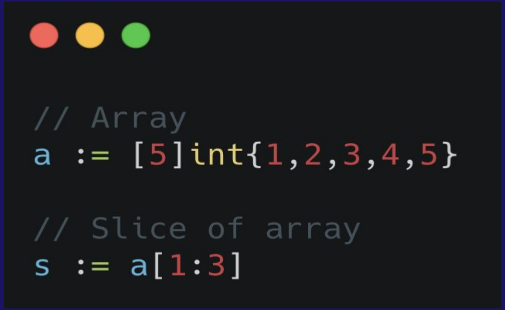
- You can also append elements to a slice as follows:



```
// You can append any number of elements
a = append(a,<element1,element2...>)
```

# Slices v/s Arrays

- Slices are actually references to arrays
- Changes to the elements of a slice will affect the corresponding underlying array
- Arrays in Go are of constant size (like C), so they are usually not preferred
- When we declare a slice on it’s own, it actually creates the same array, then references the underlying array



```
// Array
a := [5]int{1,2,3,4,5}

// Slice of array
s := a[1:3]
```

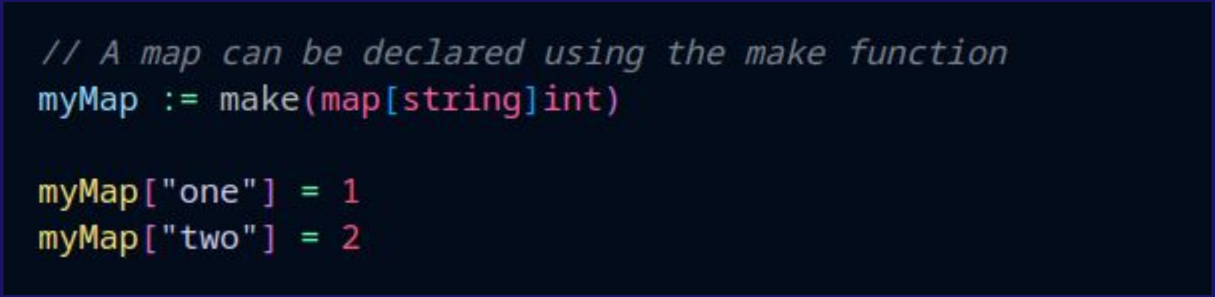
# 03

# Data Types in Go

# 2. Maps

# Maps in go

- Map is a data structure that stores key values pairs
- They behave like dictionaries in python or a map in c++
- They return two values
  - The Value corresponding to a key
  - A boolean telling us if the key is present in the map



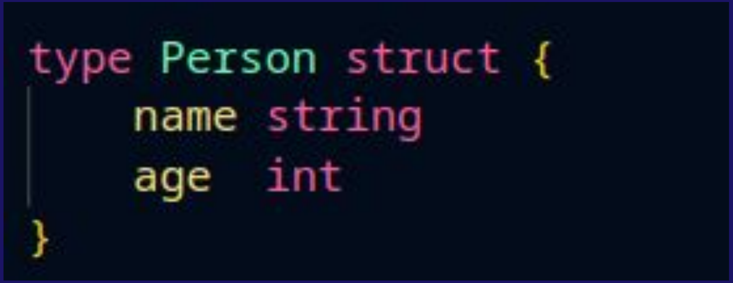
```
// A map can be declared using the make function
myMap := make(map[string]int)

myMap["one"] = 1
myMap["two"] = 2
```

# 3. Structs

# Structs in go

- Structs in go provide a way to declare custom types
- They function very similarly to structs in C but have some additional features



```go
type Person struct {
    name string
    age  int
}
```

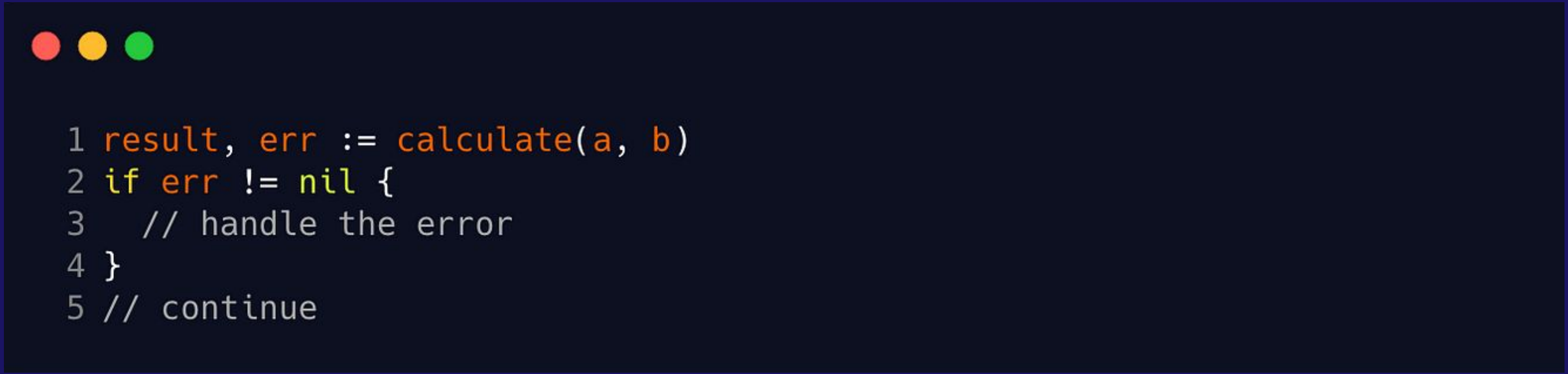
# 03

# Some Advanced Topics

# 1. Errors

# Errors

- Go has a built in error type which helps us handle errors
- Errors can be returned from functions just like any other value
- The panic statement can be used to exit out of the programme in case of an unexpected error



```go
result, err := calculate(a, b)
if err != nil {
  // handle the error
}
// continue
```

# 2. Defers

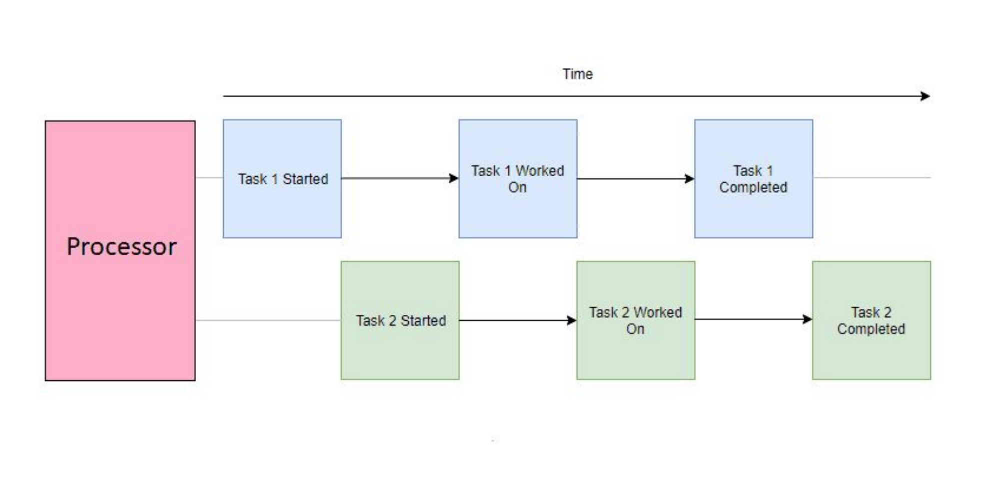
# Defer

- The defer statement is used to “defer” a piece of code to the end of its code block
- It is analogous to “finally” or “ensure” statements in other programming languages
- This means GO is clearly the best programming language as it has a built in procrastination keyword :P

✗ <insert programming language> ✓ GO

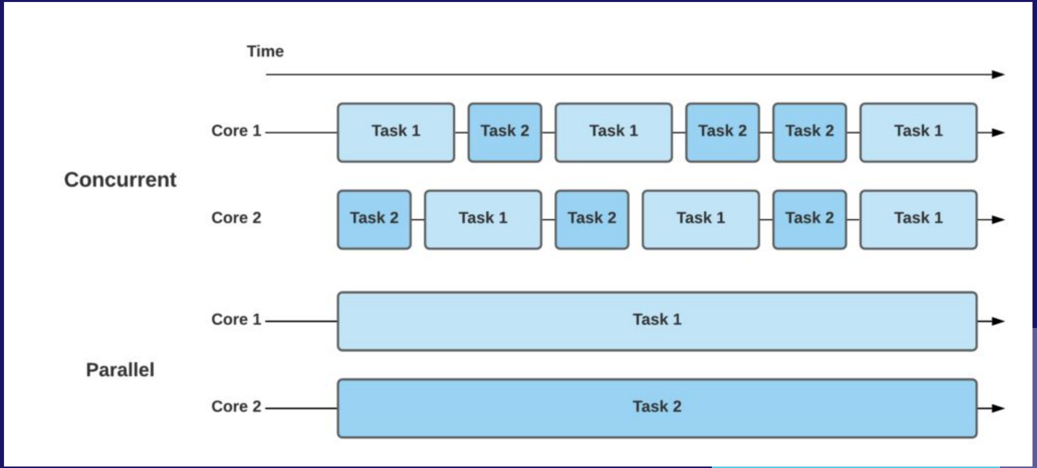
# 3. Concurrency

“Concurrency is not parallelism”



# Concurrency

# parallelism



# 4. Goroutines

# 04

# REST API

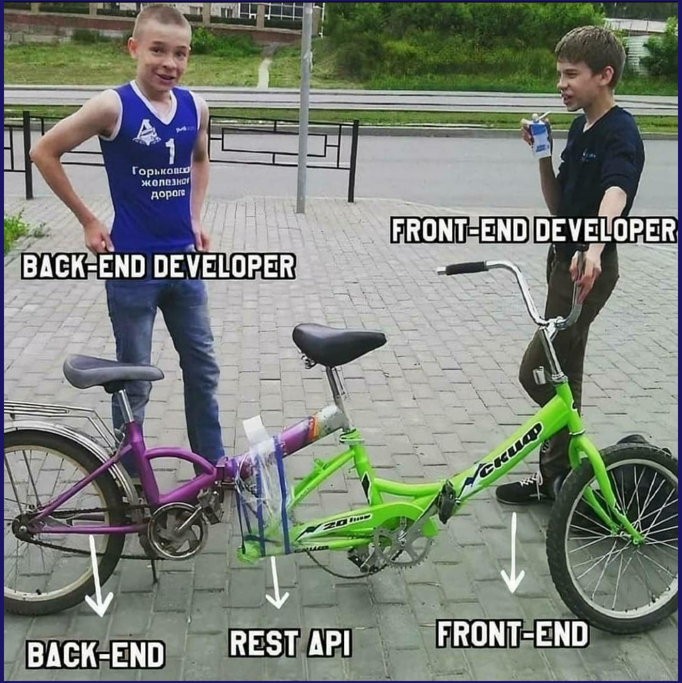
# How the web works?

Mostly web apps consists of two parts

1. The which you see (Frontend)
2. The other which you don’t see. (Backend)

**Consider Youtube**

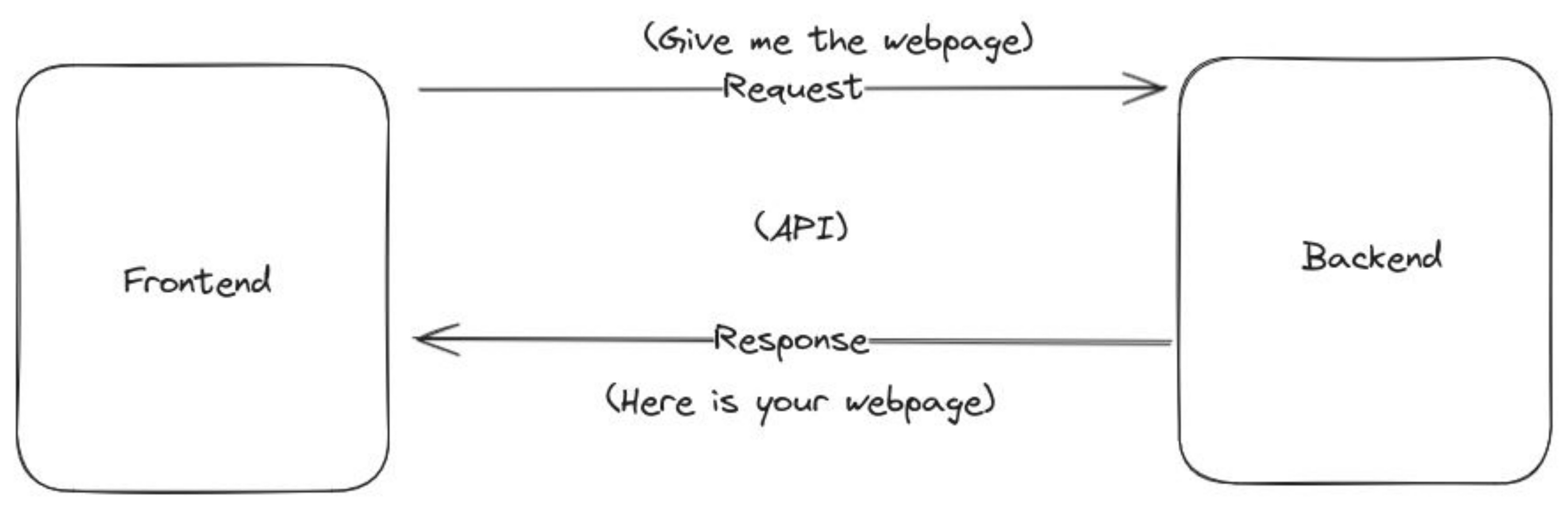
- Frontend - video player ,comments etc
- Backend - The video data,the database of Comments.It handles the part of adding comments,deleting comments, editing it.



# REST API’s

- REST stands for Representational State Transfer.
- Most used Web API style

# HTTP Request



# HTTP Request Methods

### GET

used
to obtain a resource
from a server

### POST

A POST
request creates a
fresh entry onto the
database

### PUT

These requests are
used to update
resources on the
server.

### DELETE

This request deletes a
resource from the
server.

# HTTP response status codes

Informational responses (100 – 199)

Successful responses (200 – 299)

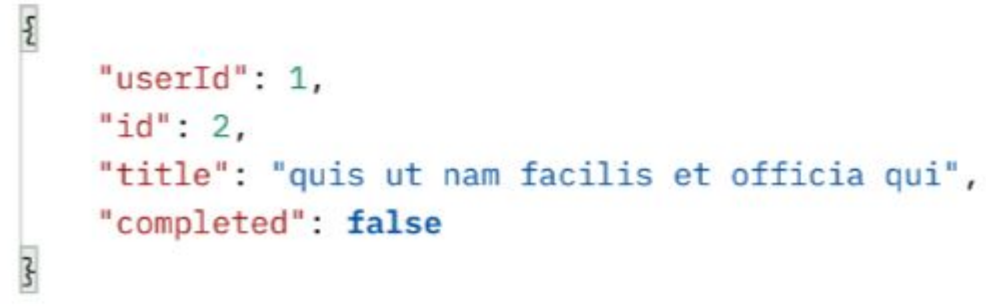
Redirection messages (300 – 399)

Client error responses (400 – 499)

Server error responses (500 – 599)

# JSON

```
{
"name" : "John",
"age" : 18
}
```



```
{
    "userId": 1,
    "id": 2,
    "title": "quis ut nam facilis et officia qui",
    "completed": false
}
```

Structured format to exchange data between frontend and backend

# Routers

- Routes are predefined path (URL) kossiitkpg.org/about
- HTTP Requests are **routed** to the code that **handles** them by the **Router**
- Handlers are the functions that define the logic

# Further Reading

1. https://go.dev/doc/
2. https://pkg.go.dev/std
3. https://gobyexample.com/
4. https://go.dev/doc/effective_go
5. https://quii.gitbook.io/learn-go-with-tests/

# Your Feedback Matters!

# Thank You!