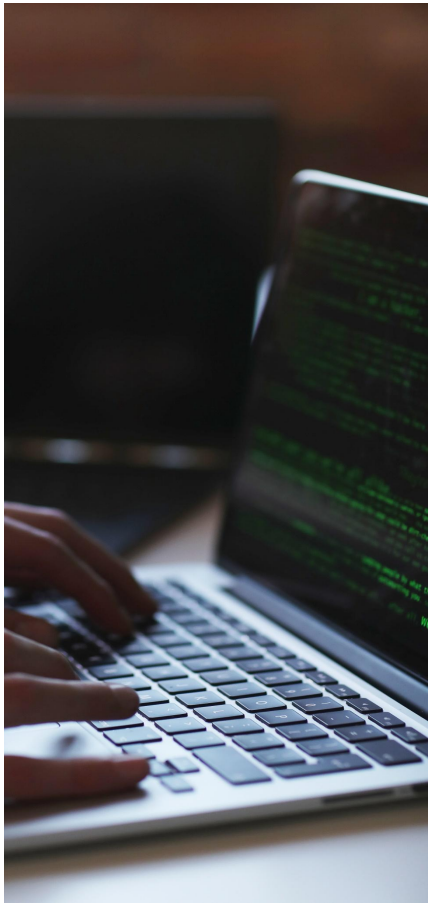
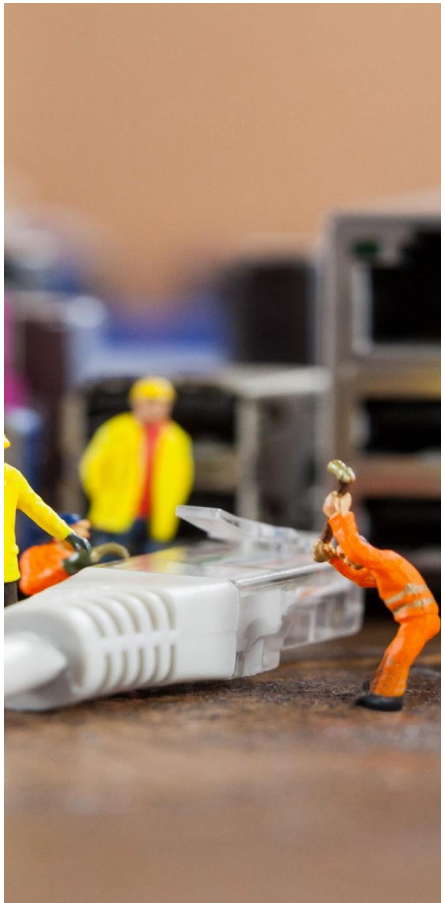
Course-Net
Empowering IT People

# Advanced Golang 2

Course-Net
Empowering IT People

#EmpoweringITPeople





# What to Learn Today?

## Advanced Golang

1. Async vs Sync
2. Go Routine
3. JSON

Course-Net
Empowering IT People

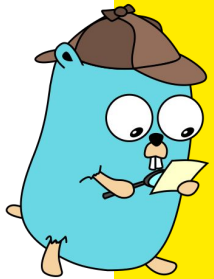
#EmpoweringITPeople

# Synchronous

In synchronous operations tasks are performed one at a time and only when one is completed, the following is unblocked. In other words, you need to wait for a task to finish to move to the next one.

# Asynchronous

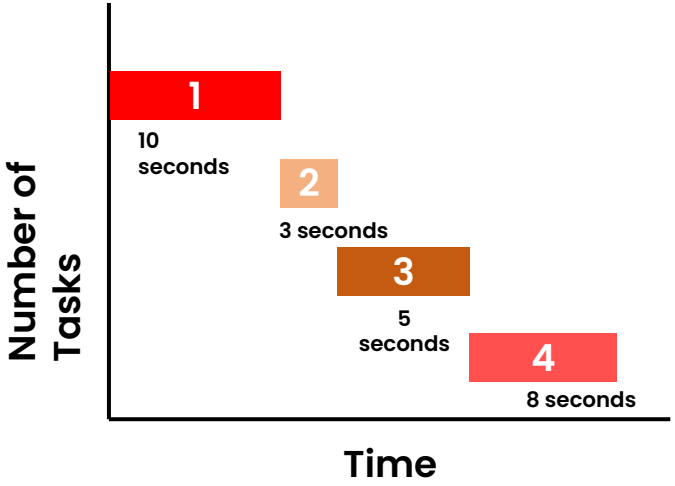
In asynchronous operations, on the other hand, you can move to another task before the previous one finishes. This way, with asynchronous programming you're able to deal with multiple requests simultaneously, thus completing more tasks in a much shorter period of time.



Course-Net
Empowering IT People

#EmpoweringITPeople

# Synchronous




From the synchronous table above, we can see the program need 26 seconds to finish the task.

# Asynchronous


From the asynchronous table above, we can see the program need 10 seconds to finish the task.

# Asynchronous in Go

With Go, you just need to define “go” keyword and Go will know that execution will be run asynchronously. This also called as Go routines.

```
go print(10, "abc")
```

# Asynchronous in Go

There's a chance that your process can't be seen because it run asynchronously. If you want the process to be seen, you can use wait group in Go.

```
func print(x int, angka int, wg *sync.WaitGroup
) {
    for i := 0; i <= x; i++ {
        fmt.Println(i)
    }
    wg.Done()
}

func main() {
    wg.Add(1)
    go print(10, "abc")
}
```

# JSON

There's a chance of that your JSON variable name different between the struct field in Go. Therefore, you can define the real JSON variable name in the struct, so Go will know which JSON variable name correspond the struct field.

```
type Student struct {
    ID      int
    Name    string `json:"student_name"`
    Address string `json:"student_address"`
}

var studentData Student

err = json.Unmarshal(jsonData, &studentData
)
if err != nil {
    fmt.Println(err.Error())
    return
}
```