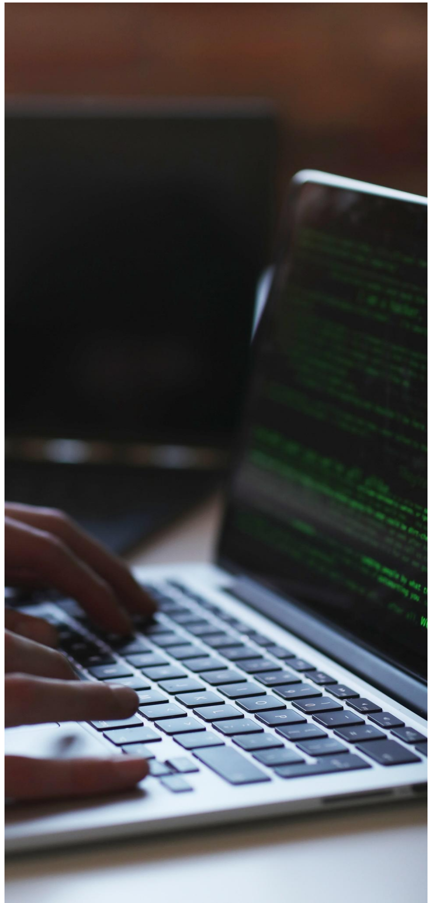
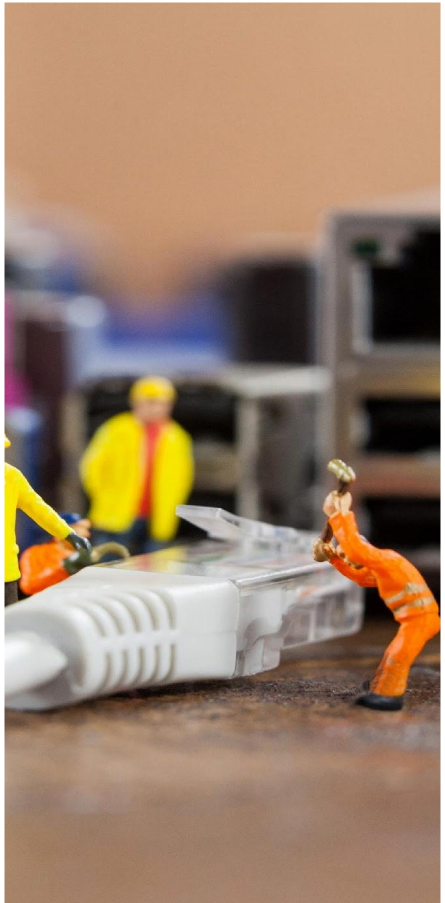
Course-Net
Empowering IT People

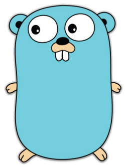
# Advanced Golang 1





# What to Learn Today?

## Advanced Golang



1. Interface
2. Method
3. Reflect
4. Regex

1 Walk

## Interface

Imagine animal abilities!

Making Sound 2

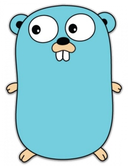
3 Breath

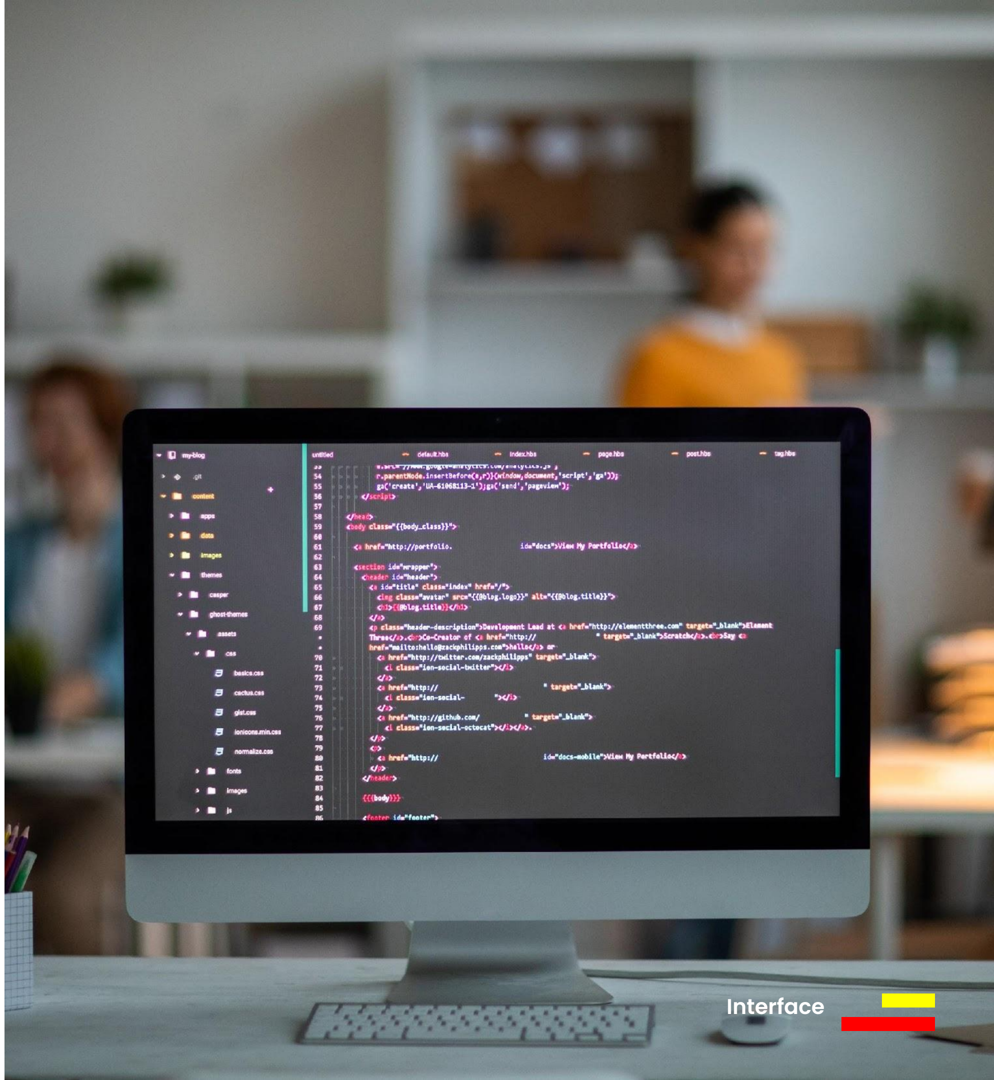
Course-Net
Empowering IT People

#EmpoweringITPeople

# Interface

```
type Animal interface {
    walk()
    makeSound()
    breath()
}
```





Interface

Course-Net
Empowering IT People

# EmpoweringITPeople

# Method

**Dog Profile**

Legs □ 4
Sound □ Bark!
Respiratory □ Lungs

**Dog**

**Cat**

**Cat Profile**

Legs □ 4
Sound □ Meow!
Respiratory □ Lungs

**Duck Profile**

Legs □ 2
Sound □ Quack!
Respiratory □ Lungs

**Duck**

**Chicken**

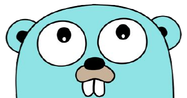
**Chicken Profile**

Legs □ 2
Sound □ Cha-chow!
Respiratory □ Lungs

**Fish Profile**

Legs □ Fins
Sound □ No Sound
Respiratory □ Gill

**Fish**



Course-Net
Empowering IT People

# EmpoweringITPeople

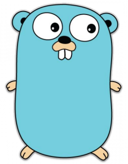
# Method

```
type Animal struct {
    legs int
    sound string
    respiratory string
}

func (a Animal) cat() {
    a.legs = 4
    a.sound = "Meow~"
    a.respiratory = "Lungs"
    fmt.Println("Cat have" + a.legs + " le
gs")

fmt.Println("Cat have" + a.sound + " sound
")

fmt.Println("Cat breath using " + a.respir
atory)
}
```

Course-Net
Empowering IT People

# EmpoweringITPeople

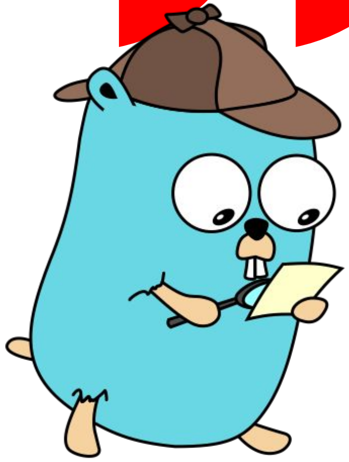
## Reflect



```
var number int64
fmt.Println(reflect.typeOf(number))
```

## What is The Meaning Of Regex?

A regular expression (shortened as regex or regexp; also referred to as rational expression) is a sequence of characters that specifies a search pattern. Usually, such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation. It is a technique developed in theoretical computer science and formal language theory.



Source: https://en.wikipedia.org/wiki/Regular_expression

# Regex Example

```go
regex1, err := regexp.Compile("1992")
if err != nil {
    fmt.Println(err.Error())
}

nip := "1992050192023"

resp1 := regex1.MatchString(nip)
```