Basics of Golang

1.Go is an open-source, statically typed, compiled programming language that was developed by Google in 2009.

2.Go is designed for building scalable and high-performance software systems.

3.Go has a simple and concise syntax, which makes it easy to learn and write.

4.Go has built-in concurrency support, which makes it easy to write concurrent programs.

5.Go uses a garbage collector for memory management, which makes memory management easy for developers.

6.Go has a strong standard library, which includes many packages for common tasks such as network programming, file I/O, and cryptography.

7.Go has a strong focus on performance and is designed to be compiled into efficient machine code.

8.Go supports multiple platforms, including Linux, macOS, Windows, and various other Unix-like operating systems.

9.Go is widely used for building web applications, network servers, command-line tools, and other software systems.

10.Some of the key features of Go include its simplicity, concurrency support, garbage collector, strong standard library, and performance optimizations.

```go
package main

import "fmt"

const LoginToken string = "ghabbhhjd" // Public

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

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

    var smallVal uint8 = 255
    fmt.Println(smallVal)
    fmt.Printf("Variable is of type: %T \n", smallVal)

    var smallFloat float64 = 255.45544511254451885
    fmt.Println(smallFloat)
    fmt.Printf("Variable is of type: %T \n", smallFloat)

    // default values and some aliases
    var anotherVariable int
    fmt.Println(anotherVariable)
    fmt.Printf("Variable is of type: %T \n", anotherVariable)

    // implicit type

    var website = "learncodeonline.in"
    fmt.Println(website)

    // no var style

    numberOfUser := 300000.0
    fmt.Println(numberOfUser)

    fmt.Println(LoginToken)
    fmt.Printf("Variable is of type: %T \n", LoginToken)

}
```

```go
package main

import (
    "bufio"
    "fmt"
    "os"
)

func main() {
    welcome := "Welcome to user input"
    fmt.Println(welcome)

    reader := bufio.NewReader(os.Stdin)
    fmt.Println("Enter the rating for our Pizza:")

    // comma ok || err err

    input, _ := reader.ReadString('\n')
    fmt.Println("Thanks for rating, ", input)
    fmt.Printf("Type of this rating is %T", input)
}
```

```go
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "strings"
)

func main() {
    fmt.Println("Welcome to our pizza app")
    fmt.Println("Please rate our pizza between 1 and 5")

    reader := bufio.NewReader(os.Stdin)

    input, _ := reader.ReadString('\n')

    fmt.Println("Thanks for rating, ", input)

    numRating, err := strconv.ParseFloat(strings.TrimSpace(input), 64)

    if err != nil {
        fmt.Println(err)

    } else {
        fmt.Println("Added 1 to your rating: ", numRating+1)
    }
}
```

```go
package main

import (
    "fmt"
    "math/big"

    //"math/rand"
    "crypto/rand"
)

func main() {
    fmt.Println("Welcome to maths in golang")

    //var mynumberOne int = 2
    //var mynumberTwo float64 = 4.5

    // fmt.Println("The sum is: ", mynumberOne+int(mynumberTwo))

    //random number
    // rand.Seed(time.Now().UnixNano())
    // fmt.Println(rand.Intn(5) + 1)

    //random from crypto

    myRandomNum, _ := rand.Int(rand.Reader, big.NewInt(5))
    fmt.Println(myRandomNum)

}
```

```go
package main

import (
    "fmt"
    "math/big"

    //"math/rand"
    "crypto/rand"
)

func main() {
    fmt.Println("Welcome to maths in golang")

    //var mynumberOne int = 2
    //var mynumberTwo float64 = 4.5

    // fmt.Println("The sum is: ", mynumberOne+int(mynumberTwo))

    //random number
    // rand.Seed(time.Now().UnixNano())
    // fmt.Println(rand.Intn(5) + 1)

    //random from crypto

    myRandomNum, _ := rand.Int(rand.Reader, big.NewInt(5))
    fmt.Println(myRandomNum)

}
```

```go
package main

import (
    "fmt"
    "time"
)

func main() {
    fmt.Println("Welcome to time study of golang")

    presentTime := time.Now()
    fmt.Println(presentTime)

    fmt.Println(presentTime.Format("01-02-2006 15:04:05 Monday"))

    createdDate := time.Date(2020, time.August, 12, 23, 23, 0, 0, time.UTC)
    fmt.Println(createdDate)
    fmt.Println(createdDate.Format("01-02-2006 Monday"))
}
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Welcome to a class on pointers")

    // var ptr *int
    // fmt.Println("Value of pointer is ", ptr)

    myNumber := 23

    var ptr = &myNumber

    fmt.Println("Value of actual pointer is ", ptr)
    fmt.Println("Value of actual pointer is ", *ptr)

    *ptr = *ptr + 2
    fmt.Println("New value is: ", myNumber)

}
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Welcome to array in golangs")

    var fruitList [4]string

    fruitList[0] = "Apple"
    fruitList[1] = "Tomato"
    fruitList[3] = "Peach"

    fmt.Println("Fruit list is: ", fruitList)
    fmt.Println("Fruit list is: ", len(fruitList))

    var vegList = [5]string{"potato", "beans", "mushroom"}
    fmt.Println("Vegy list is: ", len(vegList))

}
```

```go
package main

import (
    "fmt"
    "sort"
)

func main() {
    fmt.Println("Welcome to video on slices")

    var fruitList = []string{"Apple", "Tomato", "Peach"}
    fmt.Printf("Type of fruitlist is %T\n", fruitList)

    fruitList = append(fruitList, "Mango", "Banana")
    fmt.Println(fruitList)

    fruitList = append(fruitList[:3])
    fmt.Println(fruitList)

    highScores := make([]int, 4)

    highScores[0] = 234
    highScores[1] = 945
    highScores[2] = 465
    highScores[3] = 867
    //highScores[4] = 777

    highScores = append(highScores, 555, 666, 321)

    fmt.Println(highScores)

    //fmt.Println(sort.IntsAreSorted(highScores))
    sort.Ints(highScores)
    //fmt.Println(highScores)

    //how to remove a value from slices based on index

    var courses = []string{"reactjs", "javascript", "swift", "python", "ruby"}
    fmt.Println(courses)
    var index int = 2
    courses = append(courses[:index], courses[index+1:]...)
    fmt.Println(courses)

}
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Maps in golang")

    languages := make(map[string]string)

    languages["JS"] = "Javascript"
    languages["RB"] = "Ruby"
    languages["PY"] = "Python"

    fmt.Println("List of all languages: ", languages)
    fmt.Println("JS shorts for: ", languages["JS"])

    delete(languages, "RB")
    fmt.Println("List of all languages: ", languages)

    // loops are interesting in golang

    for _, value := range languages {
        fmt.Printf("For key v, value is %v\n", value)
    }

}
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Structs in golang")
    // no inheritance in golang; No super or parent

    hitesh := User{"Hitesh", "hitesh@go.dev", true, 16}
    fmt.Println(hitesh)
    fmt.Printf("hitesh details are: %+v\n", hitesh)
    fmt.Printf("Name is %v and email is %v.", hitesh.Name, hitesh.Email)

}

type User struct {
    Name   string
    Email  string
    Status bool
    Age    int
}
```

```go
package main

import "fmt"

func main() {
    fmt.Println("If else in golang")

    loginCount := 10
    var result string

    if loginCount < 10 {
        result = "Regular user"
    } else if loginCount > 10 {
        result = "Watch out"
    } else {
        result = "Exactly 10 login count"
    }

    fmt.Println(result)

    if 9%2 == 0 {
        fmt.Println("Number is even")
    } else {
        fmt.Println("Number is odd")
    }

    if num := 3; num < 10 {
        fmt.Println("Num is less than 10")
    } else {
        fmt.Println("Num is NOT less than 10")
    }

    // if err != nil {

    // }

}
```

```go
package main

import (
    "fmt"
    "math/rand"
    "time"
)

func main() {
    fmt.Println("Switch and case in golang")

    rand.Seed(time.Now().UnixNano())
    diceNumber := rand.Intn(6) + 1
    fmt.Println("Value of dice is ", diceNumber)

    switch diceNumber {
    case 1:
        fmt.Println("Dice value is 1 and you can open")
    case 2:
        fmt.Println("You can move 2 spot")
    case 3:
        fmt.Println("You can move to 3 spot")
        fallthrough
    case 4:
        fmt.Println("you can move to 4 spot")
        fallthrough
    case 5:
        fmt.Println("You can move to 5 spot")
    case 6:
        fmt.Println("You can move to 6 spot and roll dice again")
    default:
        fmt.Println("What was that!")
    }

}
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Welcome to loops in golang")

    days := []string{"Sunday", "Tuesday", "Wednesday", "Friday", "Saturday"}

    fmt.Println(days)

    // for d := 0; d < len(days); d++ {
    //  fmt.Println(days[d])
    // }

    // for i := range days {
    //  fmt.Println(days[i])
    // }

    // for _, day := range days {
    //  fmt.Printf("index is  and value is %v\n", day)
    // }

    rougueValue := 1

    for rougueValue < 10 {

        if rougueValue == 2 {
            goto lco
        }

        if rougueValue == 5 {
            rougueValue++
            continue
        }

        fmt.Println("Value is: ", rougueValue)
        rougueValue++
    }

lco:
    fmt.Println("Jumping at LearnCodeonline.in")

}
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Welcome to functions in golang")
    greeter()

    result := adder(3, 5)
    fmt.Println("Result is: ", result)

    proRes, myMessage := proAdder(2, 5, 8, 7, 3)
    fmt.Println("Pro result is: ", proRes)
    fmt.Println("Pro Message is: ", myMessage)

}

func adder(valOne int, valTwo int) int {
    return valOne + valTwo
}

func proAdder(values ...int) (int, string) {
    total := 0

    for _, val := range values {
        total += val
    }

    return total, "Hi Pro result function"

}

func greeter() {
    fmt.Println("Namstey from golang")
}
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Structs in golang")
    // no inheritance in golang; No super or parent

    hitesh := User{"Hitesh", "hitesh@go.dev", true, 16}
    fmt.Println(hitesh)
    fmt.Printf("hitesh details are: %+v\n", hitesh)
    fmt.Printf("Name is %v and email is %v.\n", hitesh.Name, hitesh.Email)
    hitesh.GetStatus()
    hitesh.NewMail()
    fmt.Printf("Name is %v and email is %v.\n", hitesh.Name, hitesh.Email)

}

type User struct {
    Name   string
    Email  string
    Status bool
    Age    int
}

func (u User) GetStatus() {
    fmt.Println("Is user active: ", u.Status)
}

func (u User) NewMail() {
    u.Email = "test@go.dev"
    fmt.Println("Email of this user is: ", u.Email)
}
```

```go
package main

import "fmt"

func main() {
    defer fmt.Println("World")
    defer fmt.Println("One")
    defer fmt.Println("Two")
    fmt.Println("Hello")
    myDefer()

}

// world, One, Two
// 0, 1, 2, 3, 4
// hello, 43210, two, One, world

func myDefer() {
    for i := 0; i < 5; i++ {
        defer fmt.Print(i)
    }
}

```

```go
package main

import (
    "fmt"
    "io"
    "io/ioutil"
    "os"
)

func main() {
    fmt.Println("Welcome to files in golang")
    content := "This needs to go in a file - LearnCodeOnline.in"

    file, err := os.Create("./mylcogofile.txt")

    // if err != nil {
    //  panic(err)
    // }
    checkNilErr(err)

    length, err := io.WriteString(file, content)
    checkNilErr(err)
    fmt.Println("length is: ", length)
    defer file.Close()
    readFile("./mylcogofile.txt")
}

func readFile(filname string) {
    databyte, err := ioutil.ReadFile(filname)
    checkNilErr(err)

    fmt.Println("Text data inside the file is \n", string(databyte))

}

func checkNilErr(err error) {
    if err != nil {
        panic(err)
    }
}
```

```go
var signals = []string{"test"}

var wg sync.WaitGroup //pointer
var mut sync.Mutex    // pointer

func main() {
    // go greeter("Hello")
    // greeter("world")
    websitelist := []string{
        "https://lco.dev",
        "https://go.dev",
        "https://google.com",
        "https://fb.com",
        "https://github.com",
    }

    for _, web := range websitelist {
        go getStatusCode(web)
        wg.Add(1)
    }

    wg.Wait()
    fmt.Println(signals)
}

// func greeter(s string) {
//  for i := 0; i < 6; i++ {
//      time.Sleep(3 * time.Millisecond)
//      fmt.Println(s)
//  }
// }

func getStatusCode(endpoint string) {
    defer wg.Done()

    res, err := http.Get(endpoint)

    if err != nil {
        fmt.Println("OOPS in endpoint")
    } else {
        mut.Lock()
        signals = append(signals, endpoint)
        mut.Unlock()

        fmt.Printf("%d status code for %s\n", res.StatusCode, endpoint)

    }
}
```

```go
package main

import (
    "fmt"
    "sync"
)

func main() {
    fmt.Println("Race condition - LearnCodeonline.in")

    wg := &sync.WaitGroup{}
    mut := &sync.RWMutex{}

    var score = []int{0}

    wg.Add(3)
    go func(wg *sync.WaitGroup, m *sync.RWMutex) {
        fmt.Println("One R")
        mut.Lock()
        score = append(score, 1)
        mut.Unlock()
        wg.Done()
    }(wg, mut)
    //wg.Add(1)
    go func(wg *sync.WaitGroup, m *sync.RWMutex) {
        fmt.Println("Two R")
        mut.Lock()
        score = append(score, 2)
        mut.Unlock()
        wg.Done()
    }(wg, mut)
    go func(wg *sync.WaitGroup, m *sync.RWMutex) {
        fmt.Println("Three R")
        mut.Lock()
        score = append(score, 3)
        mut.Unlock()
        wg.Done()
    }(wg, mut)
    go func(wg *sync.WaitGroup, m *sync.RWMutex) {
        fmt.Println("Three R")
        mut.RLock()
        fmt.Println(score)
        mut.RUnlock()
        wg.Done()
    }(wg, mut)

    wg.Wait()
    fmt.Println(score)
}
```

```go
package main

import (
    "fmt"
    "sync"
)

func main() {
    fmt.Println("Channels in golang-  LearnCodeOnline.in")

    myCh := make(chan int, 2)
    wg := &sync.WaitGroup{}

    // fmt.Println(<-myCh)
    // myCh <- 5
    wg.Add(2)
    // R ONLY
    go func(ch <-chan int, wg *sync.WaitGroup) {

        val, isChanelOpen := <-myCh

        fmt.Println(isChanelOpen)
        fmt.Println(val)

        //fmt.Println(<-myCh)

        wg.Done()
    }(myCh, wg)
    // send ONLY
    go func(ch chan<- int, wg *sync.WaitGroup) {
        myCh <- 0
        close(myCh)
        // myCh <- 6
        wg.Done()
    }(myCh, wg)

    wg.Wait()
}
```

Waitgroups, channels, and goroutines are used for concurrent programming.

•**Goroutines:** Goroutines are lightweight threads of execution in Golang. They are used for concurrent programming and enable multiple functions to run concurrently in a single operating system thread.

•**Channels:** Channels are used to enable communication and synchronization between Goroutines. They are used to pass data between Goroutines and can be used to control the flow of execution.

•**WaitGroups:** WaitGroups are used to synchronize Goroutines. They are used to ensure that Goroutines complete their execution before the main function exits. A WaitGroup waits for a collection of Goroutines to finish executing and blocks the execution of the main function until all Goroutines have completed.

Together, these features provide a powerful way to write concurrent programs in Golang. By using Goroutines, channels, and WaitGroups, it is possible to write high-performance, efficient, and scalable programs that can take advantage of multiple CPU cores and can handle multiple requests concurrently.

```go
package main

import (
    "fmt"
    "log"
    "net/http"

    "github.com/tarm/serial"
)

var port *serial.Port

func main() {
    // Open serial port
    c := &serial.Config{Name: "/dev/ttyACM0", Baud: 9600}
    var err error
    port, err = serial.OpenPort(c)
    if err != nil {
        log.Fatal(err)
    }

    // Define HTTP endpoints
    http.HandleFunc("/", indexHandler)
    http.HandleFunc("/on", onHandler)
    http.HandleFunc("/off", offHandler)

    // Start HTTP server
    log.Println("Starting HTTP server...")
    err = http.ListenAndServe(":8080", nil)
    if err != nil {
        log.Fatal(err)
    }
}

func indexHandler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintln(w, "Welcome to the LED control center!")
}

func onHandler(w http.ResponseWriter, r *http.Request) {
    // Send "1" to turn on the LED
    _, err := port.Write([]byte("1"))
    if err != nil {
        log.Fatal(err)
    }
    fmt.Fprintln(w, "LED turned on.")
}

func offHandler(w http.ResponseWriter, r *http.Request) {
    // Send "0" to turn off the LED
    _, err := port.Write([]byte("0"))
    if err != nil {
        log.Fatal(err)
    }
    fmt.Fprintln(w, "LED turned off.")
}
```