17. Using example code.

After installing Go example code, we can review my own goworkspace folder's pkg/mod/github.com.

18. IDE's

go get -d [github.com/GoesToEleven/go-programming](http://github.com/GoesToEleven/go-programming)

go get -d [github.com/GoesToEleven/GolangTraining](http://github.com/GoesToEleven/GolangTraining)

go get -d [github.com/GoesToEleven/golang-web-dev](http://github.com/GoesToEleven/golang-web-dev)

19. Go Commands

"go fmt" == ls,

"go fmt ./..." ==> formatting all code.

"go run main.go" ==> running programm.

"go build" ==> for running programm simply. it is created .exe. if you go to that .exe, programm wil bee worked.

go run, go build, go install,

20. Github repos.

git add --all

git commit -m "example"

git remote add origin "link"

git push -u origin master

21. Github explored

SSH key added.

22. Package Management

<https://research.swtch.com/deps>

23. Go Modules

<https://go.dev/blog/using-go-modules>

24. Using Go Modules

"go list -m all" for seeing all of dependecies.

"go mod init <example/username/repo>" for creating modules.

27. Playground.

<https://go.dev/play/> from this link, we can run our go code.

28. Hello World

There are three types of Control flow

1.Sequence

2.Loop

3.conditional

30. Short Decloration Operator

x := 2 --> this is short declaration operator.

fmt.Println(x) --> this is output.

Terminalogy

>keywords

>operator: "2 + 2" + is the operator

>operand: 2 is the operand

>statement: 1 statement

>expression: 2 + 3 = 5

31. Var keyword.

var x = 2 --> this is var keyword.

() --> this is parens

{} --> this is curly braces

"var x int" --> we can use this out of the function for declaring value.

32. Exploring type

How to explore type of value? For this we can use this

"fmt.Printf("%T\n", value)" == type of value, for example, int pr string.

"var x string = 'I am a string text'" or "var x = 'I am a string text'"

"var y int = 8" or "var y = 8"

Data types: Boolean, Numiric, String.

33. Zero value

"%T\n" what is it?

%T for declaring type of value

\n for new line.

34. The fmt package

fmt.Printf("%T\n", y) --> type

fmt.Printf("%b\n", y) --> binary

fmt.Printf("%x\n", y) --> hexadecimal

fmt.Printf("%#x", y) --> dexadicemal with zero

<https://pkg.go.dev/fmt>

fmt.Printf("%#x\t%b\t%x" , y, y, y)

s := Sprintf("%#x\t%b\t%x", y, y, y)

fmt.Println(s)

35. Creating type

Creating our own data type as lik this:

"type int <my\_type\_name>"

"var A <my\_type\_name>"

36. Conversion, Casting

how to converse type

{

type int hotdo

var x int

var y hotdog

}

{

x = 12

y= 13

y = int(x) //this is main point

fmt.Println(y)

}

A great place to ask questions is the "golang bridge forum" at<https://forum.golangbridge.org/>

An "identifier" is the name assigned to a variable or a function or a constant.

What are the data types in Go?

Go has three basic data types: bool: represents a boolean value and is either true or false. Numeric: represents integer types, floating point values, and complex types. string: represents a string value.

43. Bool type

"var x bool" ---> this is bool type

"a := 2

b := 34

fmt.Println( a == b)

Output: false "

45. Numeric types

import ( "fmt", "runtime" )

func main() {

fmt.Println(runtime.GOOS)

fmt.Println(runtime.GOARCH)

}

46. String type

package main

import "fmt"

func main() {

s := "Hello javokhir"

fmt.Println(s)

fmt.Printf("%T\n", s)

------------------------------------------------

bs := []byte(s)

fmt.Println(bs)

fmt.Printf("%T\n", bs)

------------------------------------------------

for i := 0; i < len(s); i++ {

fmt.Printf("%#U", s[i])

}

for i, v := range s {

fmt.Println(i, v)

}

}