Go basic notes

Go mod init name\_of\_app - Init go project

While loop

for i < len(arr) {

        fmt.Printf("\n%v", arr[i])

        i++

    }

For loop

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

        fmt.Printf("\n%v", arr[i])

    }

Do while

for {

        fmt.Printf("\n%v", arr[i])

        i++

        if i >= len(arr) {

            break

        }

    }

For range

for index, item := range arr {

        fmt.Printf("\nindex %v value is %v ", index, item)

    }

Array

names := [100]string{"jocel", "eli"}

names := […]string{"jocel", "eli"}

// initialize the elements of index 0 and 3 only

arrayOfIntegers := [5]int{0: 7, 3: 9}

multi dimensional array

// create a 2 dimensional array

    arrayInteger := [...][]int{{1, 2}, {3, 4,5}}

    // access the values of 2d array

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

        for j := 0; j < len(arrayInteger[i]); j++ {

            fmt.Print(arrayInteger[i][j], " ")

        }

    }

Slice – similar to array but doesn’t have fixed size

// this is an array

numbers := [5]int{1, 2, 3, 4, 5}

slice array

numbers := [8]int{10, 20, 30, 40, 50, 60, 70, 80}

sliceNumbers = numbers[4, 7]

slice operation

Append

primeNumbers := []int{2, 3}

// add elements 5, 7 to the slice

primeNumbers = append(primeNumbers, 5, 7)

fmt.Println("Prime Numbers:", primeNumbers)

// add elements of oddNumbers to evenNumbers

evenNumbers = append(evenNumbers, oddNumbers...)

Copy Equal Len

Make – use to declare data without initial value

student := make(map[int]string)

Map (key value pair)

subjectMarks := map[string]float32{"Golang": 85, "Java": 80, "Python": 81}

map of array

    n := map[string][]int{

        "photos":  {1, 2, 3, 4, 5, 6, 5, 6},

        "photos2": {1, 2, 3, 4, 5, 6, 5, 6},

        "photos3": {1, 2, 3, 4, 5, 6, 5, 6},

        "photos4": {1, 2, 3, 4, 5, 6, 5, 6},

    }

    fmt.Println(n)

map of array maps

    m := map[string][]map[string]string{

        "photos":   {{"a": "1"}, {"b": "2"}},

        "pictures": {{"a": "1"}, {"b": "2"}},

    }

    fmt.Println(m)

delete in map

delete(personAge, "John") \\map,key

loop

    m := map[string]string{

        "team1": "team1v",

        "team2": "team2v",

        "team3": "team3v",

        "team4": "team4v",

        "team5": "team5v",

    }

    for key, value := range m {

        fmt.Printf("\ni=%s value=%s", key, value)

    }

Access key only

for key := range m {

        fmt.Printf("\nkey=%s ", key)

    }

Struct (similar to c structure) – use to store multiple data types in one variable like class

// declare a struct

    type Person struct {

        name string

        age  int

    }

    // assign value to struct while creating an instance

    person1 := Person{"John", 25}

    fmt.Println(person1)

    // define an instance

    var person2 Person

    // assign value to struct variables

    person2 = Person{

        name: "Sara",

        age:  29,

    }

    fmt.Println(person2)

access property

person1.name

function inside struct

// Program to use function as a field  of struct

package main

import "fmt"

// initialize the function Rectangle

type Rectangle func(int, int) int

// create structure

type rectanglePara struct {

  length  int

  breadth int

  color   string

  // function as a field of struct

  rect Rectangle

    or

  rect func(int, int) int

}

func main() {

  // assign values to struct variables

  result := rectanglePara{

    length:  10,

    breadth: 20,

    color:   "Red",

    rect: func(length int, breadth int) int {

      return length \* breadth

    },

  }

  fmt.Println("Color of Rectangle: ", result.color)

  fmt.Println("Area of Rectangle: ", result.rect(result.length, result.breadth))

}