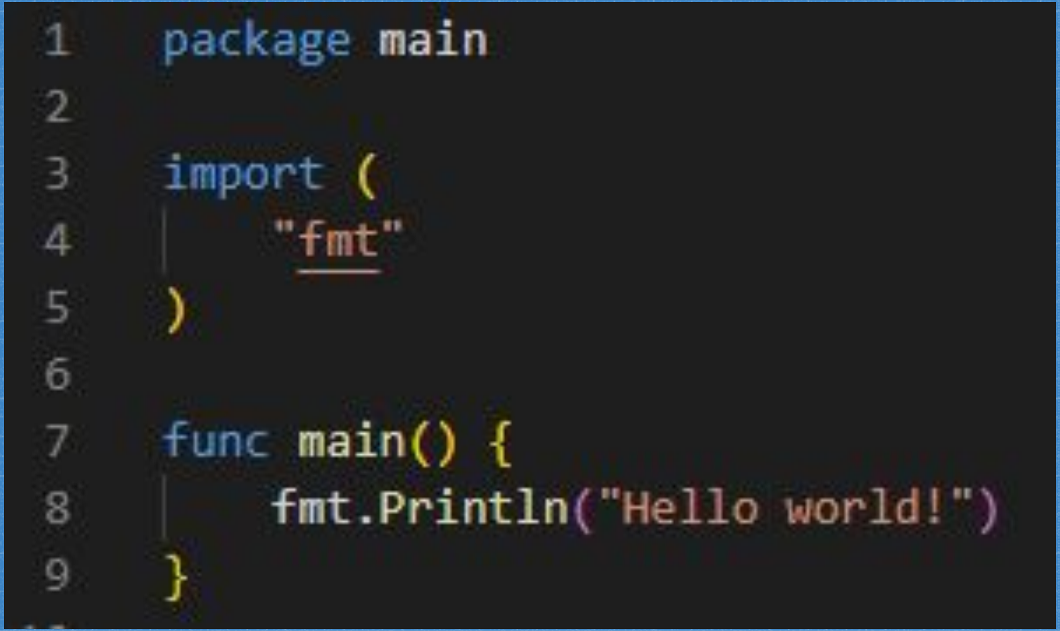
Golang Session 1

What is Go?

A programming language that is:

- Compiled
- Static Typed
- Garbage Collected
- Concurrent & Parallel
- Object-Oriented (Kinda)
- Easy

Hello World!



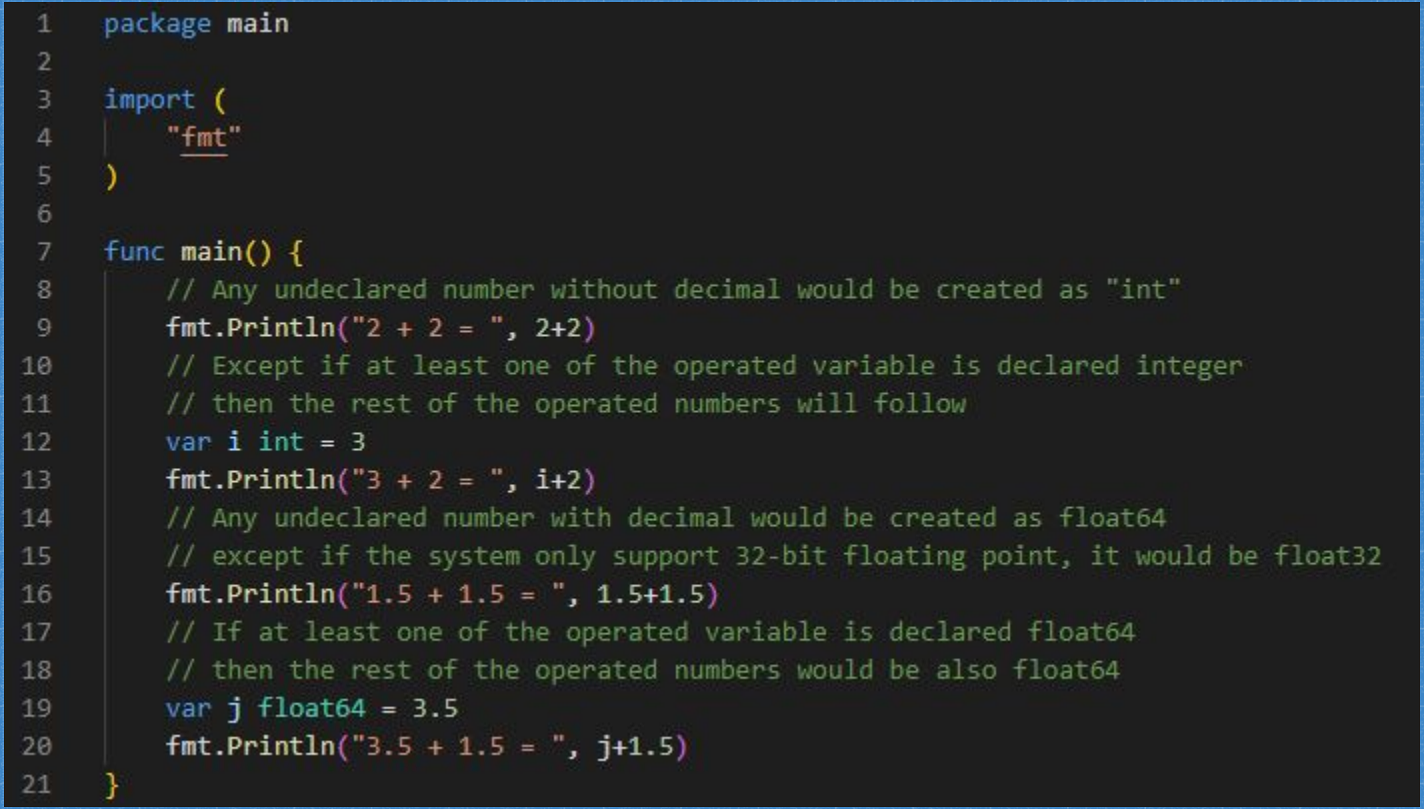
```go
package main

import (
	"fmt"
)

func main() {
	fmt.Println("Hello world!")
}
```

Types in Go

Primitive Types:

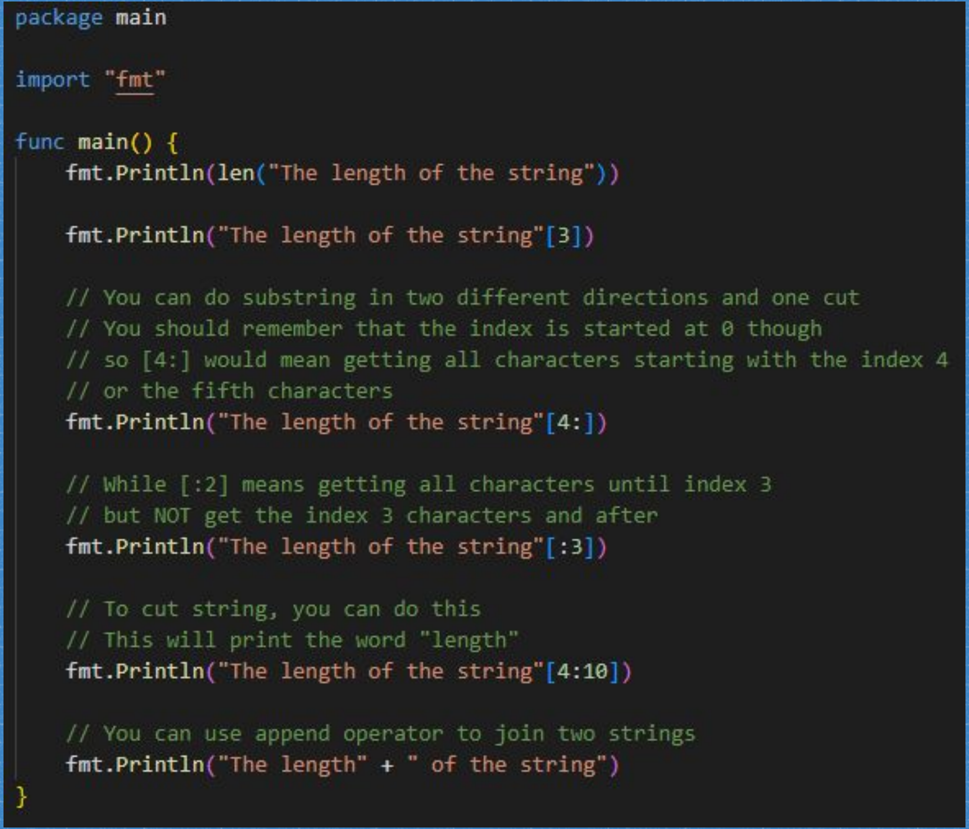
- Integers
  - int, int8, int16, int32, int64
  - uint, uint8, uint16, uint32, uint64
  - byte, uintptr
- Floats
  - float32, float64
  - complex64, complex128
- Strings
- Booleans

Numbers

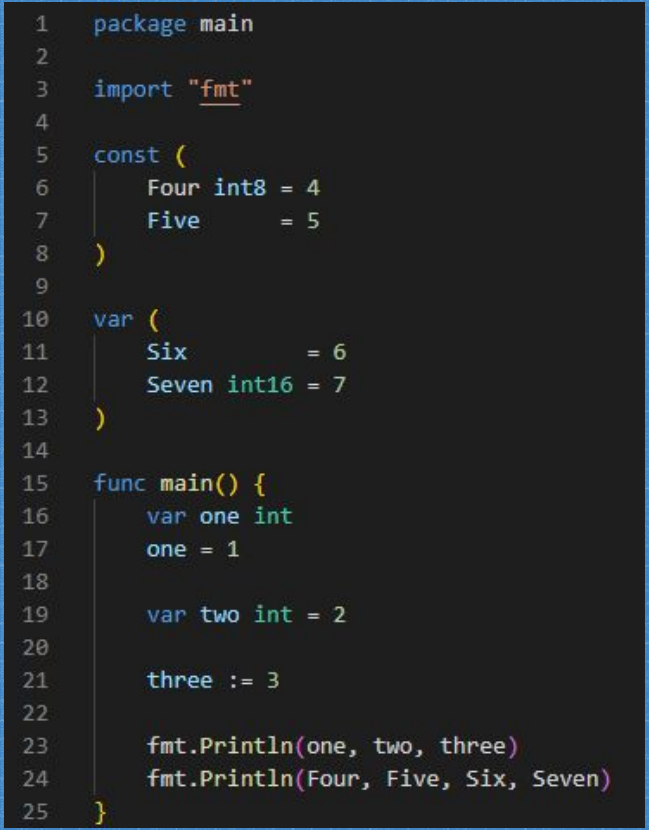


```go
package main

import (
	"fmt"
)

func main() {
	// Any undeclared number without decimal would be created as "int"
	fmt.Println("2 + 2 = ", 2+2)
	// Except if at least one of the operated variable is declared integer
	// then the rest of the operated numbers will follow
	var i int = 3
	fmt.Println("3 + 2 = ", i+2)
	// Any undeclared number with decimal would be created as float64
	// except if the system only support 32-bit floating point, it would be float32
	fmt.Println("1.5 + 1.5 = ", 1.5+1.5)
	// If at least one of the operated variable is declared float64
	// then the rest of the operated numbers would be also float64
	var j float64 = 3.5
	fmt.Println("3.5 + 1.5 = ", j+1.5)
}
```

Strings


```go
package main

import "fmt"

func main() {
	fmt.Println(len("The length of the string"))

	fmt.Println("The length of the string"[3])

	// You can do substring in two different directions and one cut
	// You should remember that the index is started at 0 though
	// so [4:] would mean getting all characters starting with the index 4
	// or the fifth characters
	fmt.Println("The length of the string"[4:])

	// While [:2] means getting all characters until index 3
	// but NOT get the index 3 characters and after
	fmt.Println("The length of the string"[:3])

	// To cut string, you can do this
	// This will print the word "length"
	fmt.Println("The length of the string"[4:10])

	// You can use append operator to join two strings
	fmt.Println("The length" + " of the string")
}
```

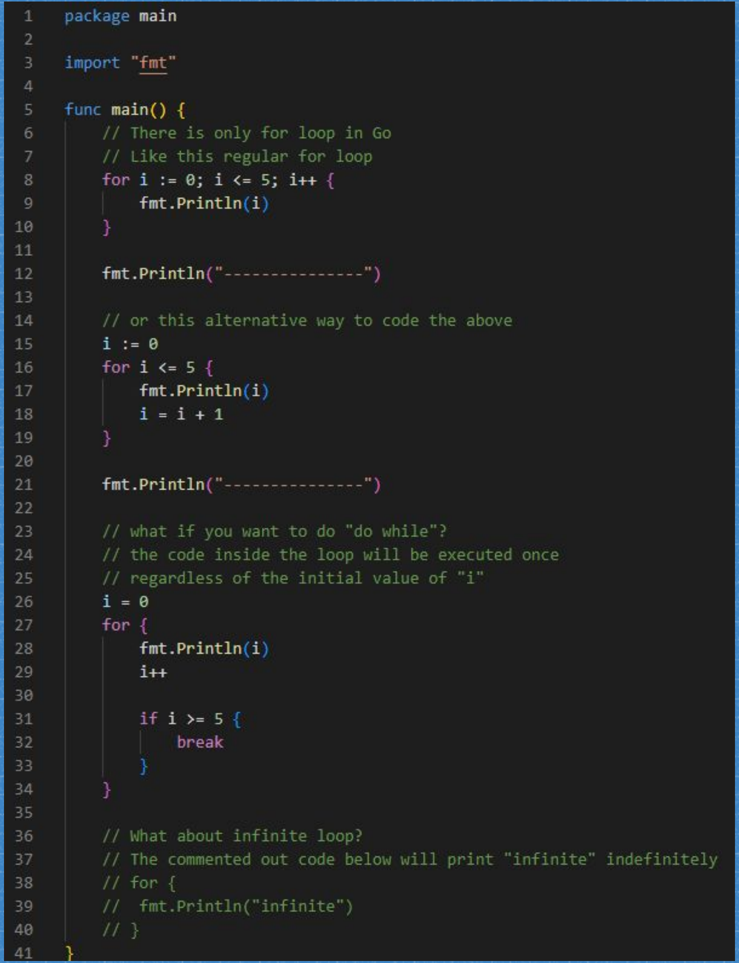
Variables



```go
package main

import "fmt"

const (
	Four int8 = 4
	Five      = 5
)

var (
	Six         = 6
	Seven int16 = 7
)

func main() {
	var one int
	one = 1

	var two int = 2

	three := 3

	fmt.Println(one, two, three)
	fmt.Println(Four, Five, Six, Seven)
}
```

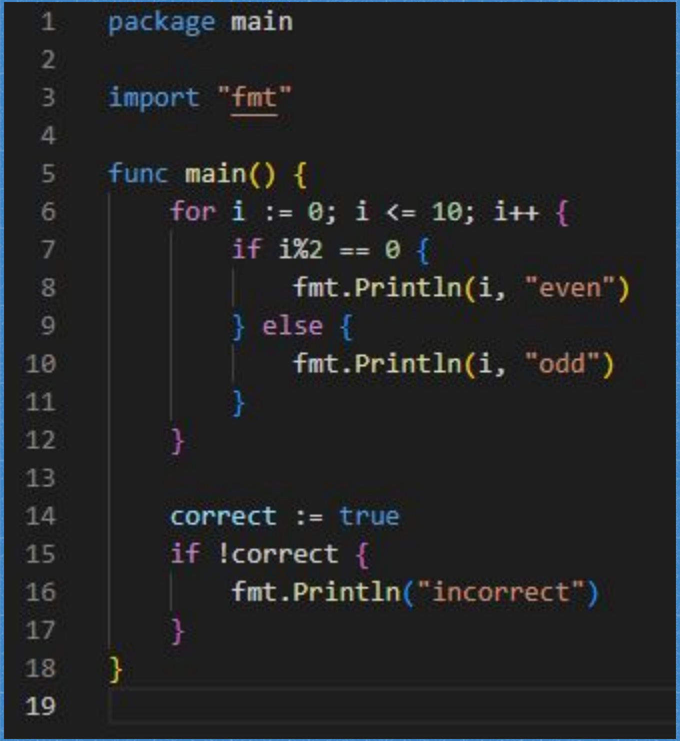
Control Structures

- for
- if
- switch



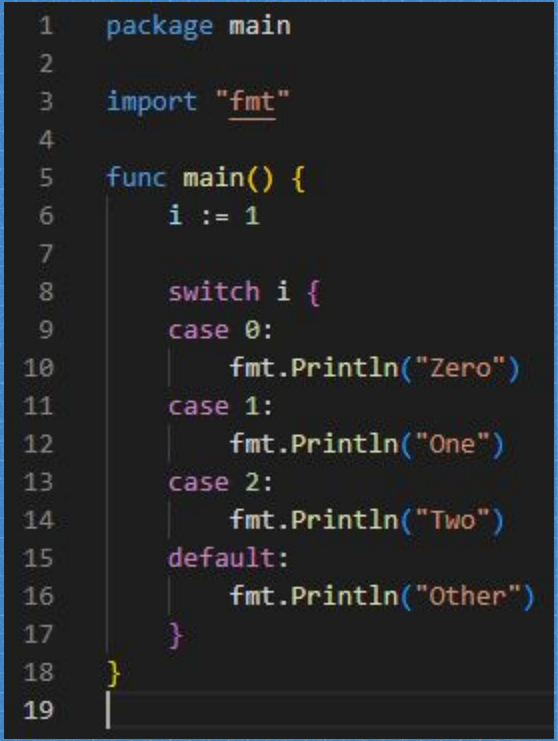
```go
package main

import "fmt"

func main() {
	// There is only for loop in Go
	// Like this regular for loop
	for i := 0; i <= 5; i++ {
		fmt.Println(i)
	}

	fmt.Println("--------------")

	// or this alternative way to code the above
	i := 0
	for i <= 5 {
		fmt.Println(i)
		i = i + 1
	}

	fmt.Println("--------------")

	// what if you want to do "do while"?
	// the code inside the loop will be executed once
	// regardless the initial value of "i"
	i = 0
	for {
		fmt.Println(i)
		i++

		if i >= 5 {
			break
		}
	}

	// What about infinite loop?
	// The commented out code below will print "infinite" indefinitely
	// for {
	// 	fmt.Println("infinite")
	// }
}
```

Control Structures

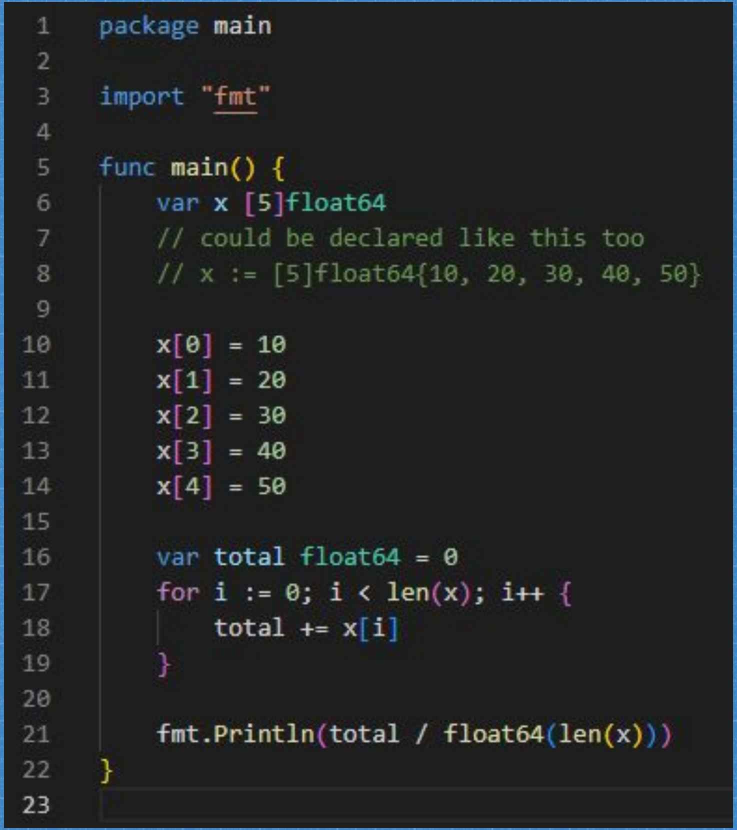


```go
package main

import "fmt"

func main() {
	for i := 0; i <= 10; i++ {
		if i%2 == 0 {
			fmt.Println(i, "even")
		} else {
			fmt.Println(i, "odd")
		}
	}

	correct := true
	if !correct {
		fmt.Println("incorrect")
	}
}

```

Control Structures

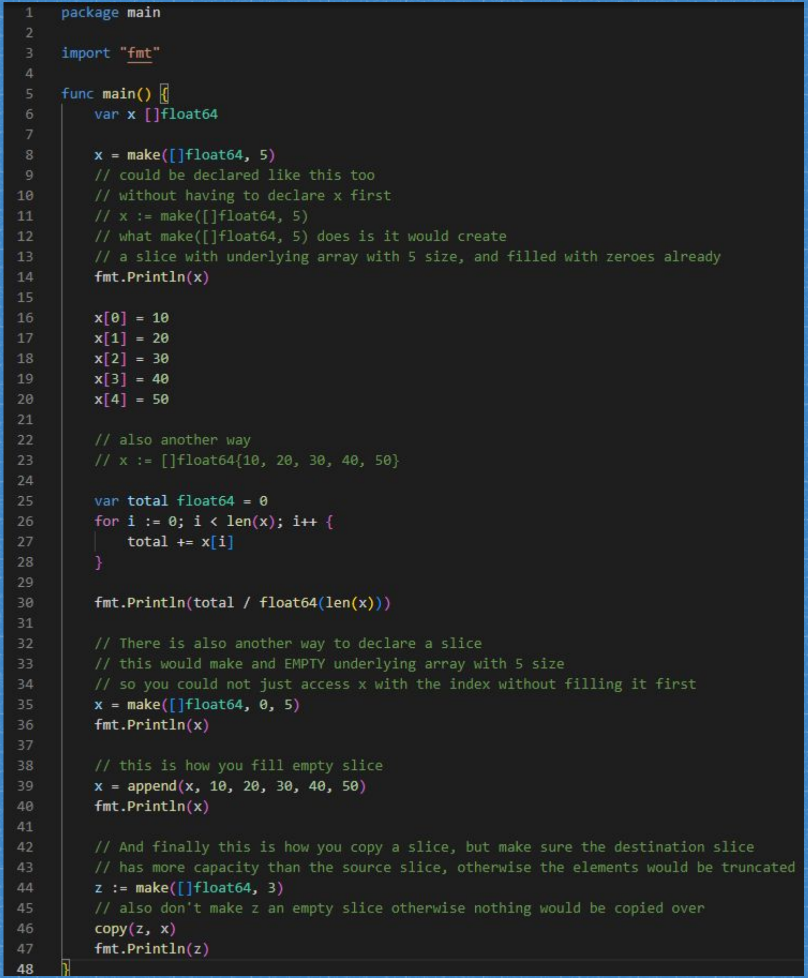

```go
package main

import "fmt"

func main() {
	i := 1

	switch i {
	case 0:
		fmt.Println("Zero")
	case 1:
		fmt.Println("One")
	case 2:
		fmt.Println("Two")
	default:
		fmt.Println("Other")
	}
}
```

Arrays, Slices, Maps



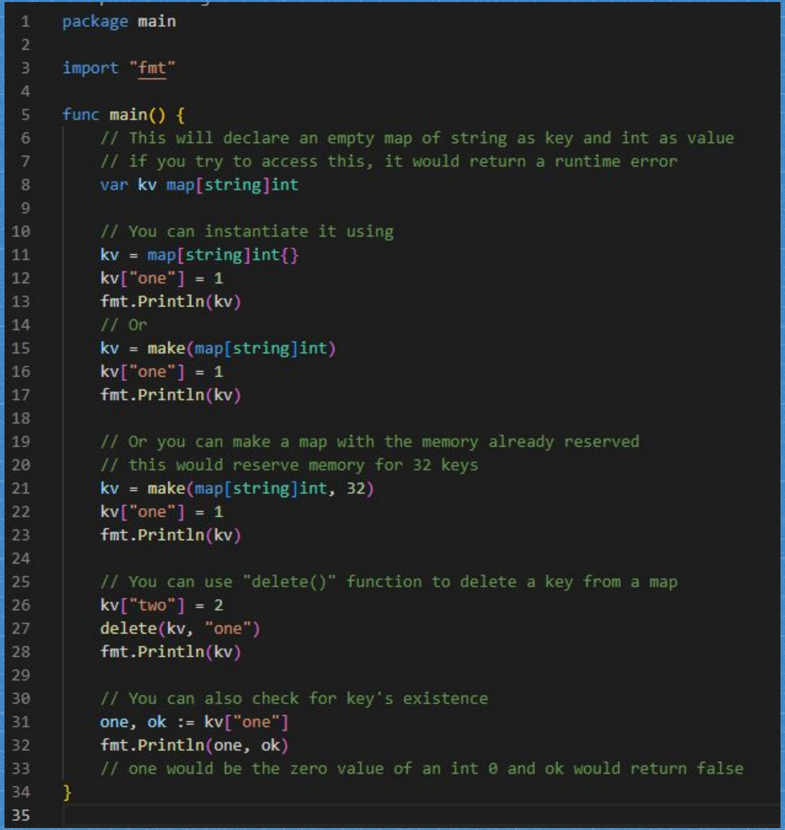
```go
package main

import "fmt"

func main() {
	var x [5]float64
	// could be declared like this too
	// x := [5]float64{10, 20, 30, 40, 50}

	x[0] = 10
	x[1] = 20
	x[2] = 30
	x[3] = 40
	x[4] = 50

	var total float64 = 0
	for i := 0; i < len(x); i++ {
		total += x[i]
	}

	fmt.Println(total / float64(len(x)))
}

```

Arrays, Slices, Maps



```go
package main

import "fmt"

func main() {
	var x []float64

	x = make([]float64, 5)
	// could be declared like this too
	// without having to declare x first
	// x := make([]float64, 5)
	// what make([]float64, 5) does is it would create
	// a slice with underlying array with 5 size, and filled with zeroes already
	fmt.Println(x)

	x[0] = 10
	x[1] = 20
	x[2] = 30
	x[3] = 40
	x[4] = 50

	// also another way
	// x := []float64{10, 20, 30, 40, 50}

	var total float64 = 0
	for i := 0; i < len(x); i++ {
		total += x[i]
	}

	fmt.Println(total / float64(len(x)))

	// There is also another way to declare a slice
	// this would make and EMPTY underlying array with 5 size
	// so you could not just access x with the index without filling it first
	x = make([]float64, 0, 5)
	fmt.Println(x)

	// this is how you fill empty slice
	x = append(x, 10, 20, 30, 40, 50)
	fmt.Println(x)

	// And finally this is how you copy a slice, but make sure the destination slice
	// has more capacity than the source slice, otherwise the elements would be truncated
	z := make([]float64, 3)
	// also don't make z an empty slice otherwise nothing would be copied over
	copy(z, x)
	fmt.Println(z)
}
```

Arrays, Slices, Maps



```go
package main

import "fmt"

func main() {
    // This will declare an empty map of string as key and int as value
    // if you try to access this, it would return a runtime error
    var kv map[string]int

    // You can instantiate it using
    kv = map[string]int{}
    kv["one"] = 1
    fmt.Println(kv)
    // Or
    kv = make(map[string]int)
    kv["one"] = 1
    fmt.Println(kv)

    // Or you can make a map with the memory already reserved
    // this would reserve memory for 32 keys
    kv = make(map[string]int, 32)
    kv["one"] = 1
    fmt.Println(kv)

    // You can use "delete()" function to delete a key from a map
    kv["two"] = 2
    delete(kv, "one")
    fmt.Println(kv)

    // You can also check for key's existence
    one, ok := kv["one"]
    fmt.Println(one, ok)
    // one would be the zero value of an int 0 and ok would return false
}

```

Thanks!

ANY QUESTIONS?

You can find me at:

andputra@alamisharia.co.id

github.com/JesusIslam