GO PROGRAMMING LANGAUGE

- **GoLang** Launched in Nov 2009 by Google (Robert Griesemer, Rob Pike, and Ken Thompson)
- It is a statically-typed compiled language having syntax similar to that of C.
- It provides garbage collection, type safety, dynamic-typing capability.
- Go supports concurrent programming, i.e. it allows running multiple processes simultaneously.
- It provides many advanced built-in types such as variable length arrays and key-value maps.

Go Program File Extension (. go)

Open Command Line / Terminal to Build the **Go** program
```
>>> go build filename.go
```

Open Command Line / Terminal to Run the **Go** program
```
>>> go run filename.go
```

Go Tokens: keyword, an identifier, a constant, string literal or a symbol.

Line Separator: the semicolon **;** is **optional** in GoLang. **\n** can also be separating the statements.

Keywords:

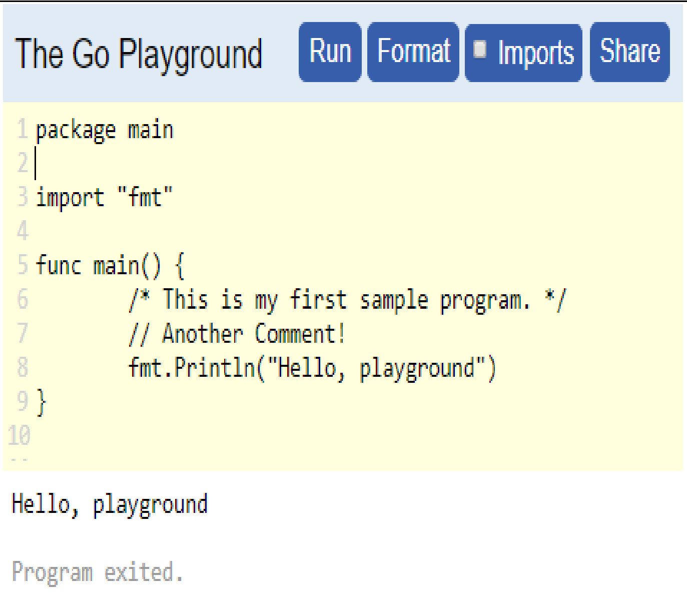
break	default	Func	interface	select
case	defer	Go	map	Struct
chan	else	Goto	package	Switch
const	fallthrough	If	range	Type
continue	for	Import	return	Var

Data Types:
1. **Numerical Types** = byte, int, int8, int16, int32, int64, uint8, uint16, uint32, uint64, float32, float64, Complex64, complex128. | Type Format of int is %d, float and complex is %g.
2. **String Types** = string. | Type Format: %s
   - Strings are immutable types that is once created, it is not possible to change the contents of a string.
3. **Boolean Types** = true, false | Type Format: %t. | Type Format for Chan and Pointers is %p

Variables:
Syntax:
- <variable_name> := <value>
- var <variable_name> <type>
- var <variable_name> <type> = <value>
- var <variable_name> = <value>
- var <variable_name1>, <variable_name2> <type>
- var <variable_name1>, <variable_name2> = <value1>, <value2>

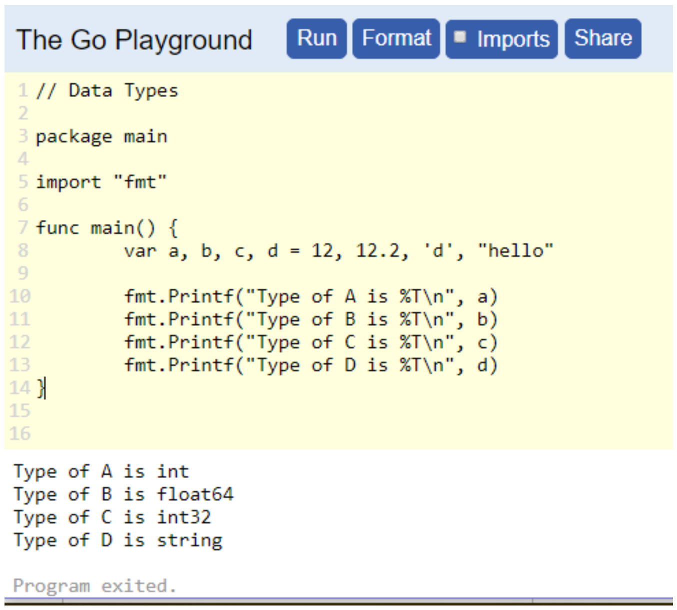
Go Online Playground - https://play.golang.org/

Code Snippets



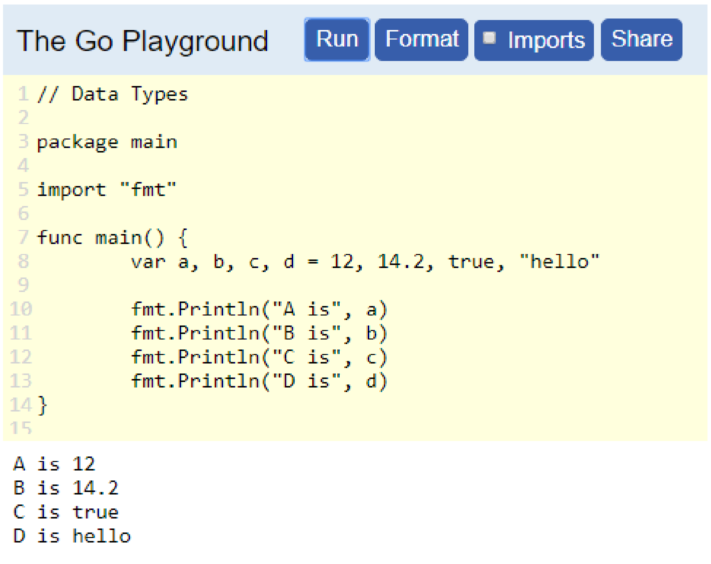
```go
package main

import "fmt"

func main() {
        /* This is my first sample program. */
        // Another Comment!
        fmt.Println("Hello, playground")
}

```

```
Hello, playground

Program exited.
```



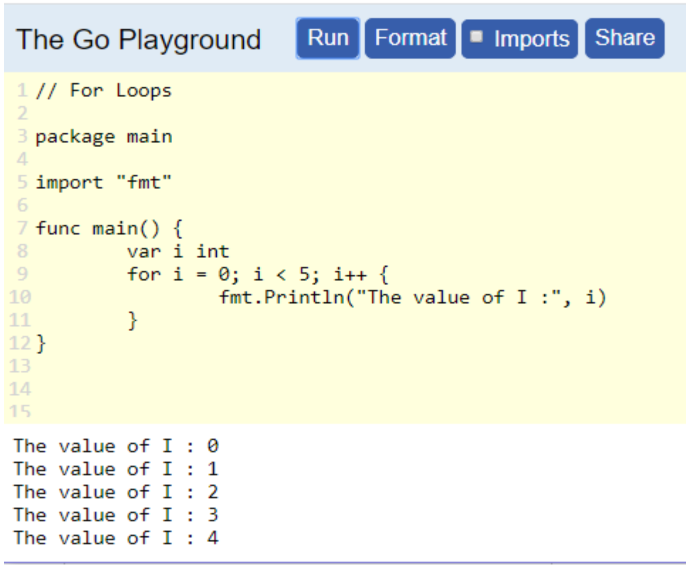
```go
// Data Types

package main

import "fmt"

func main() {
        var a, b, c, d = 12, 12.2, 'd', "hello"

        fmt.Printf("Type of A is %T\n", a)
        fmt.Printf("Type of B is %T\n", b)
        fmt.Printf("Type of C is %T\n", c)
        fmt.Printf("Type of D is %T\n", d)
}


```

```
Type of A is int
Type of B is float64
Type of C is int32
Type of D is string

Program exited.
```



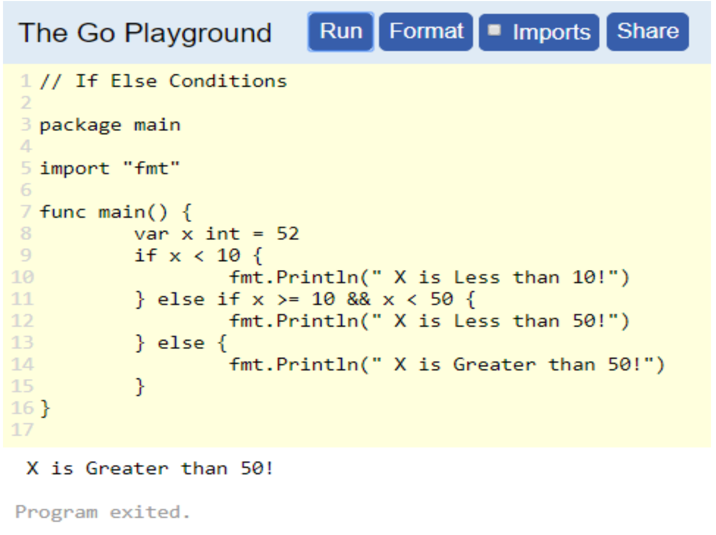
```go
// Data Types

package main

import "fmt"

func main() {
        var a, b, c, d = 12, 14.2, true, "hello"

        fmt.Println("A is", a)
        fmt.Println("B is", b)
        fmt.Println("C is", c)
        fmt.Println("D is", d)
}

```

```
A is 12
B is 14.2
C is true
D is hello
```



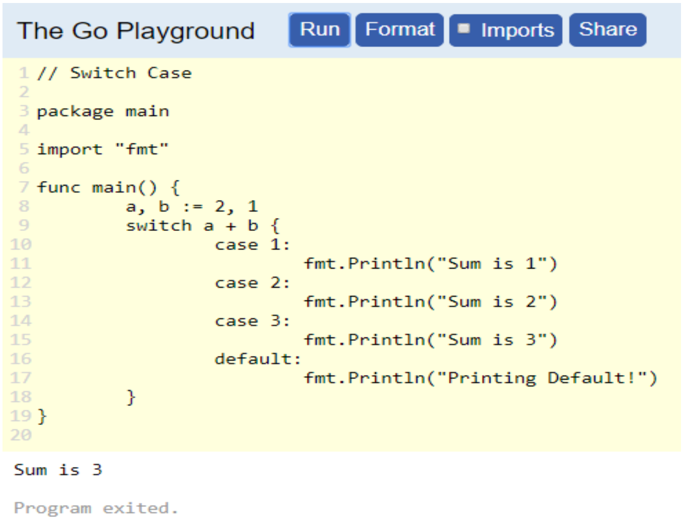
```go
// For Loops

package main

import "fmt"

func main() {
        var i int
        for i = 0; i < 5; i++ {
                fmt.Println("The value of I :", i)
        }
}



```

```
The value of I : 0
The value of I : 1
The value of I : 2
The value of I : 3
The value of I : 4
```



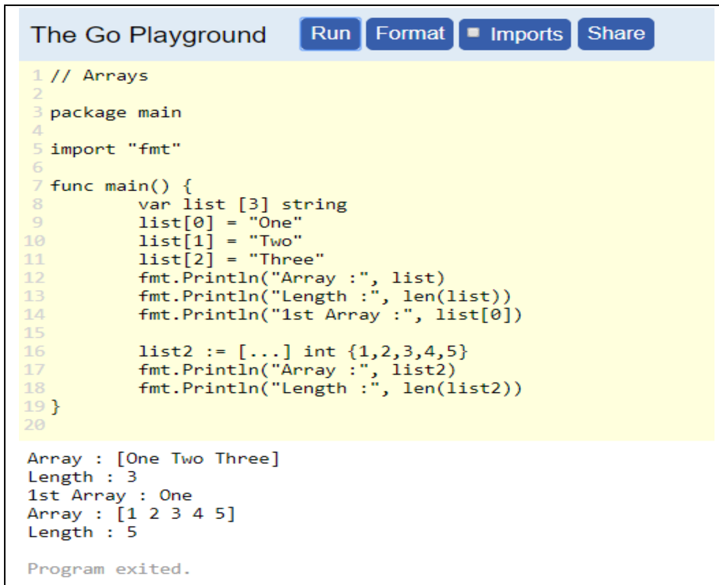
```go
// If Else Conditions

package main

import "fmt"

func main() {
        var x int = 52
        if x < 10 {
                fmt.Println(" X is Less than 10!")
        } else if x >= 10 && x < 50 {
                fmt.Println(" X is Less than 50!")
        } else {
                fmt.Println(" X is Greater than 50!")
        }
}

```

```
 X is Greater than 50!

Program exited.
```



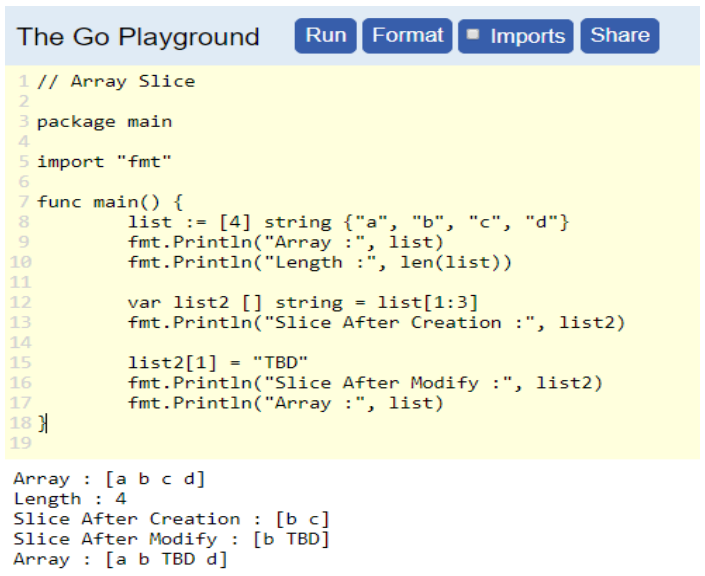
```go
// Switch Case

package main

import "fmt"

func main() {
        a, b := 2, 1
        switch a + b {
                case 1:
                        fmt.Println("Sum is 1")
                case 2:
                        fmt.Println("Sum is 2")
                case 3:
                        fmt.Println("Sum is 3")
                default:
                        fmt.Println("Printing Default!")
        }
}

```

```
Sum is 3

Program exited.
```



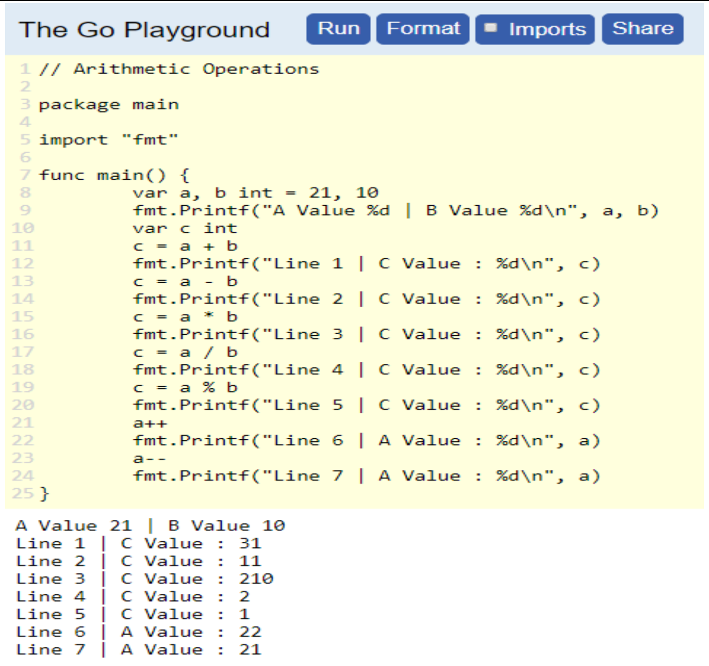
```go
// Arrays

package main

import "fmt"

func main() {
	var list [3] string
	list[0] = "One"
	list[1] = "Two"
	list[2] = "Three"
	fmt.Println("Array :", list)
	fmt.Println("Length :", len(list))
	fmt.Println("1st Array :", list[0])

	list2 := [...] int {1,2,3,4,5}
	fmt.Println("Array :", list2)
	fmt.Println("Length :", len(list2))
}
```

```
Array : [One Two Three]
Length : 3
1st Array : One
Array : [1 2 3 4 5]
Length : 5

Program exited.
```



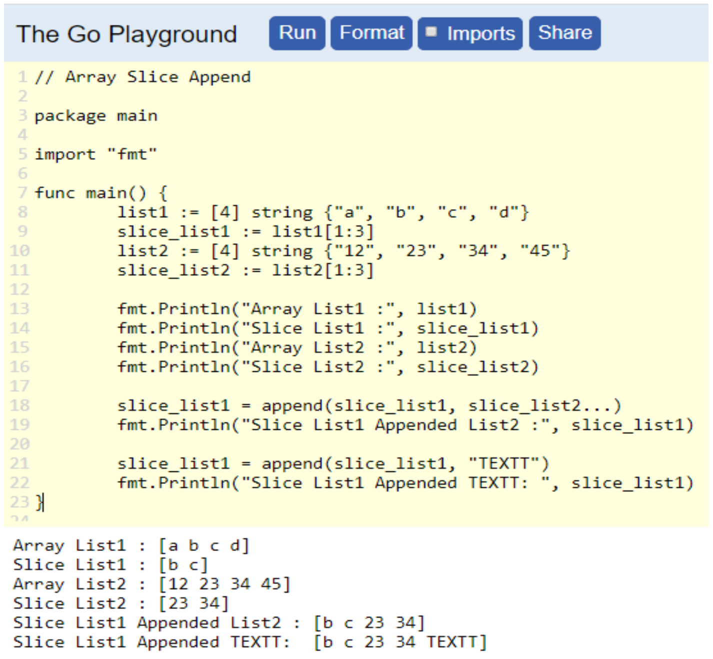
```go
// Array Slice

package main

import "fmt"

func main() {
	list := [4] string {"a", "b", "c", "d"}
	fmt.Println("Array :", list)
	fmt.Println("Length :", len(list))

	var list2 [] string = list[1:3]
	fmt.Println("Slice After Creation :", list2)

	list2[1] = "TBD"
	fmt.Println("Slice After Modify :", list2)
	fmt.Println("Array :", list)
}
```

```
Array : [a b c d]
Length : 4
Slice After Creation : [b c]
Slice After Modify : [b TBD]
Array : [a b TBD d]
```



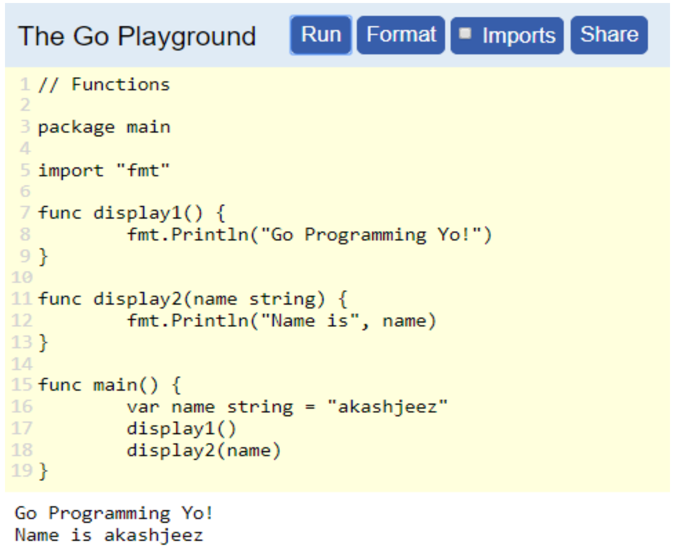
```go
// Arithmetic Operations

package main

import "fmt"

func main() {
	var a, b int = 21, 10
	fmt.Printf("A Value %d | B Value %d\n", a, b)
	var c int
	c = a + b
	fmt.Printf("Line 1 | C Value : %d\n", c)
	c = a - b
	fmt.Printf("Line 2 | C Value : %d\n", c)
	c = a * b
	fmt.Printf("Line 3 | C Value : %d\n", c)
	c = a / b
	fmt.Printf("Line 4 | C Value : %d\n", c)
	c = a % b
	fmt.Printf("Line 5 | C Value : %d\n", c)
	a++
	fmt.Printf("Line 6 | A Value : %d\n", a)
	a--
	fmt.Printf("Line 7 | A Value : %d\n", a)
}
```

```
A Value 21 | B Value 10
Line 1 | C Value : 31
Line 2 | C Value : 11
Line 3 | C Value : 210
Line 4 | C Value : 2
Line 5 | C Value : 1
Line 6 | A Value : 22
Line 7 | A Value : 21
```



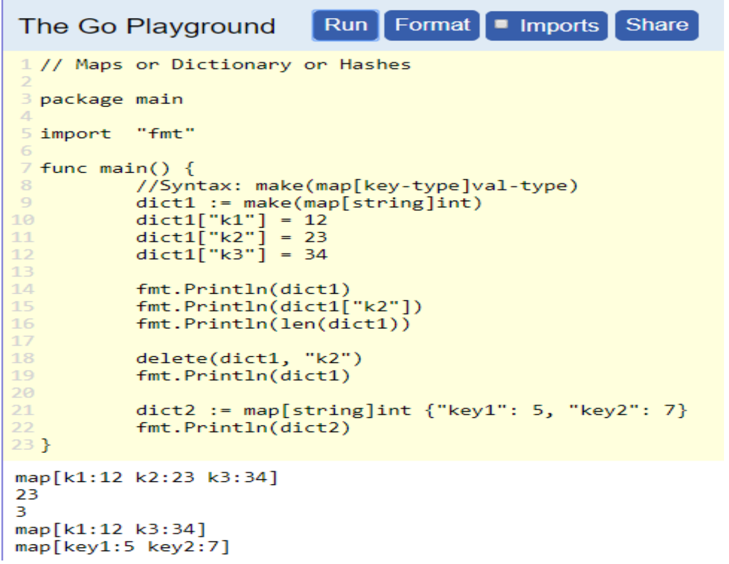
```go
// Array Slice Append

package main

import "fmt"

func main() {
	list1 := [4] string {"a", "b", "c", "d"}
	slice_list1 := list1[1:3]
	list2 := [4] string {"12", "23", "34", "45"}
	slice_list2 := list2[1:3]

	fmt.Println("Array List1 :", list1)
	fmt.Println("Slice List1 :", slice_list1)
	fmt.Println("Array List2 :", list2)
	fmt.Println("Slice List2 :", slice_list2)

	slice_list1 = append(slice_list1, slice_list2...)
	fmt.Println("Slice List1 Appended List2 :", slice_list1)

	slice_list1 = append(slice_list1, "TEXTT")
	fmt.Println("Slice List1 Appended TEXTT: ", slice_list1)
}
```

```
Array List1 : [a b c d]
Slice List1 : [b c]
Array List2 : [12 23 34 45]
Slice List2 : [23 34]
Slice List1 Appended List2 : [b c 23 34]
Slice List1 Appended TEXTT:  [b c 23 34 TEXTT]
```



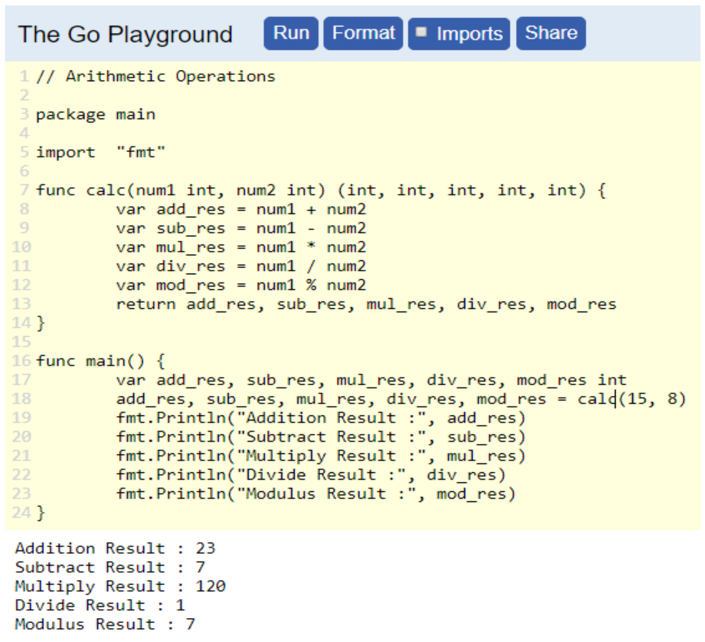
```go
// Functions

package main

import "fmt"

func display1() {
	fmt.Println("Go Programming Yo!")
}

func display2(name string) {
	fmt.Println("Name is", name)
}

func main() {
	var name string = "akashjeez"
	display1()
	display2(name)
}
```

```
Go Programming Yo!
Name is akashjeez
```



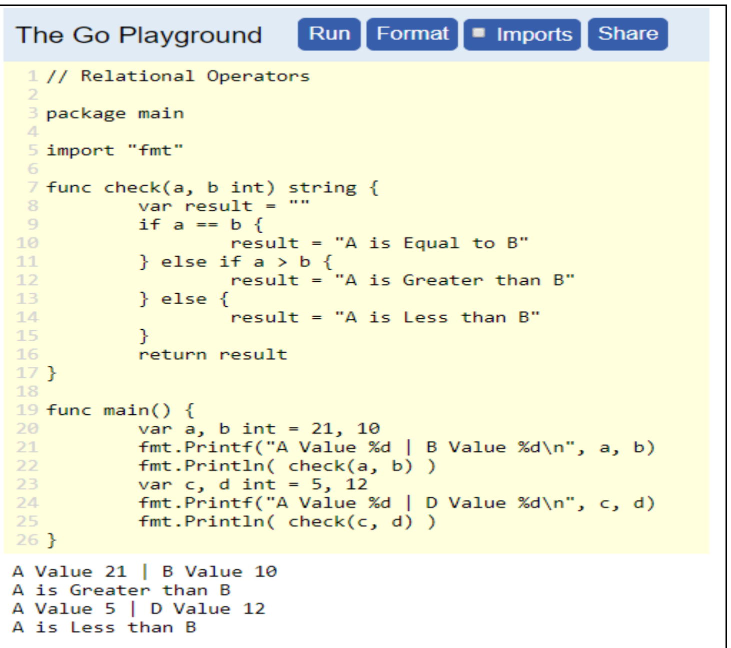
```go
// Maps or Dictionary or Hashes

package main

import  "fmt"

func  main() {
	//Syntax: make(map[key-type]val-type)
	dict1 := make(map[string]int)
	dict1["k1"] = 12
	dict1["k2"] = 23
	dict1["k3"] = 34

	fmt.Println(dict1)
	fmt.Println(dict1["k2"])
	fmt.Println(len(dict1))

	delete(dict1, "k2")
	fmt.Println(dict1)

	dict2 := map[string]int {"key1": 5, "key2": 7}
	fmt.Println(dict2)
}
```

```
map[k1:12 k2:23 k3:34]
23
3
map[k1:12 k3:34]
map[key1:5 key2:7]
```



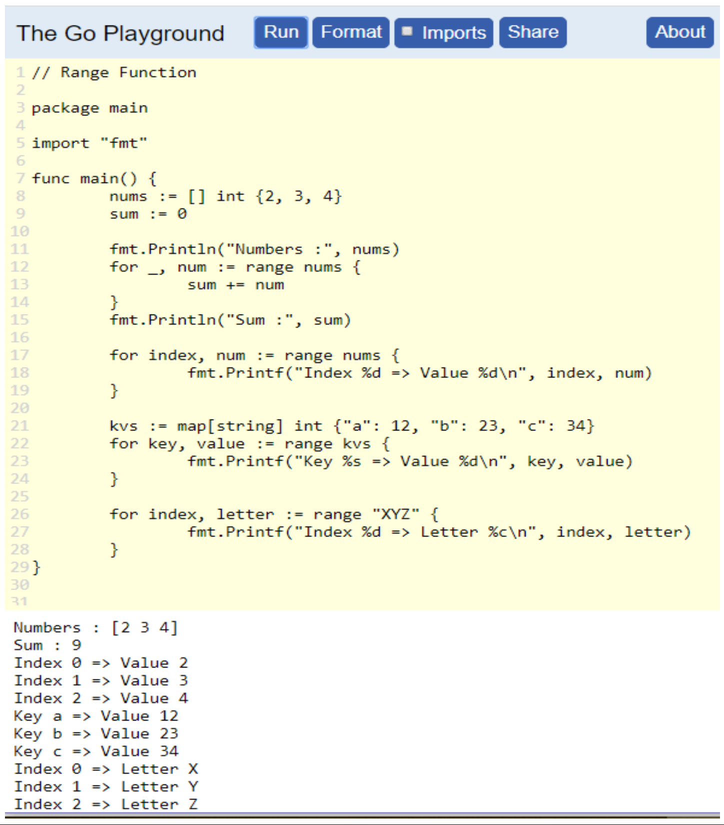
```go
// Arithmetic Operations

package main

import  "fmt"

func calc(num1 int, num2 int) (int, int, int, int, int) {
        var add_res = num1 + num2
        var sub_res = num1 - num2
        var mul_res = num1 * num2
        var div_res = num1 / num2
        var mod_res = num1 % num2
        return add_res, sub_res, mul_res, div_res, mod_res
}

func main() {
        var add_res, sub_res, mul_res, div_res, mod_res int
        add_res, sub_res, mul_res, div_res, mod_res = calc(15, 8)
        fmt.Println("Addition Result :", add_res)
        fmt.Println("Subtract Result :", sub_res)
        fmt.Println("Multiply Result :", mul_res)
        fmt.Println("Divide Result :", div_res)
        fmt.Println("Modulus Result :", mod_res)
}
```

```
Addition Result : 23
Subtract Result : 7
Multiply Result : 120
Divide Result : 1
Modulus Result : 7
```



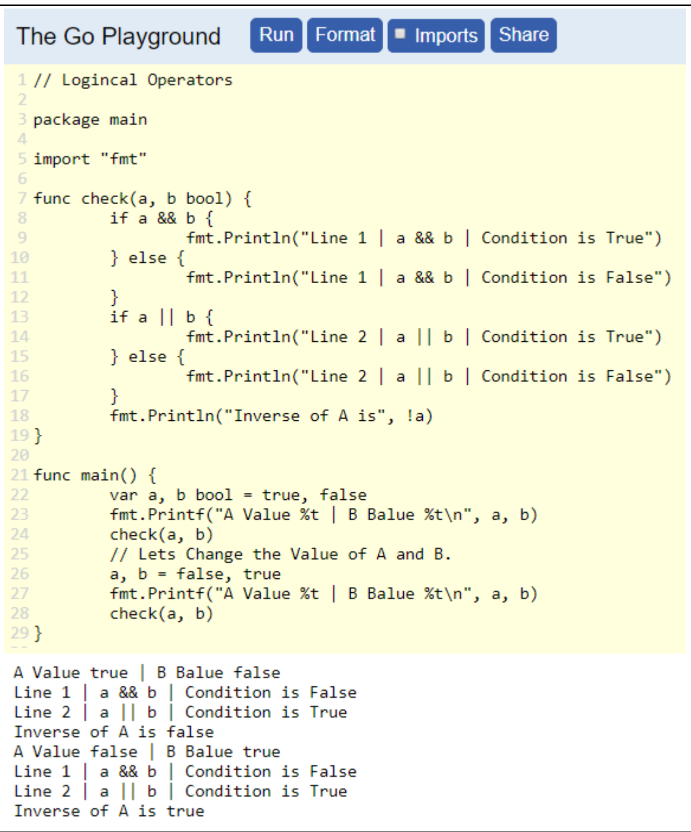
```go
// Relational Operators

package main

import "fmt"

func check(a, b int) string {
        var result = ""
        if a == b {
                result = "A is Equal to B"
        } else if a > b {
                result = "A is Greater than B"
        } else {
                result = "A is Less than B"
        }
        return result
}

func main() {
        var a, b int = 21, 10
        fmt.Printf("A Value %d | B Value %d\n", a, b)
        fmt.Println( check(a, b) )
        var c, d int = 5, 12
        fmt.Printf("A Value %d | D Value %d\n", c, d)
        fmt.Println( check(c, d) )
}
```

```
A Value 21 | B Value 10
A is Greater than B
A Value 5 | D Value 12
A is Less than B
```



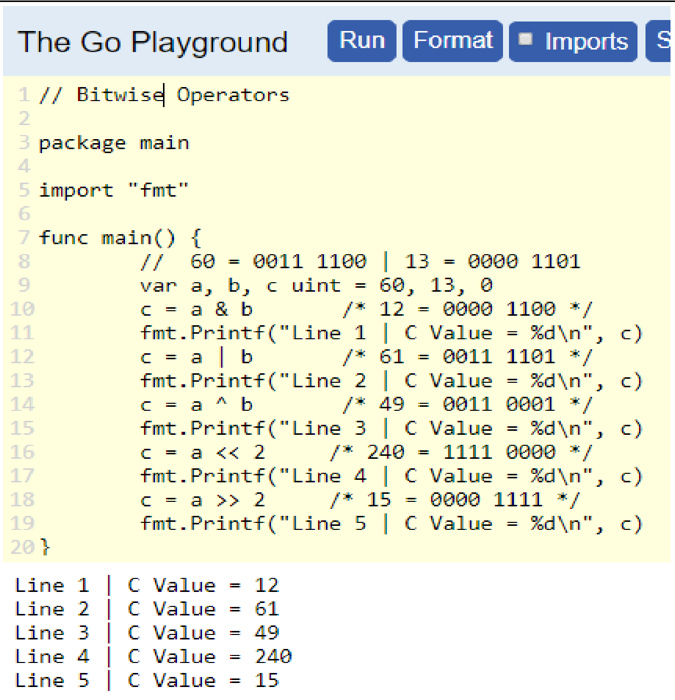
```go
// Range Function

package main

import "fmt"

func main() {
        nums := [] int {2, 3, 4}
        sum := 0

        fmt.Println("Numbers :", nums)
        for _, num := range nums {
                sum += num
        }
        fmt.Println("Sum :", sum)

        for index, num := range nums {
                fmt.Printf("Index %d => Value %d\n", index, num)
        }

        kvs := map[string] int {"a": 12, "b": 23, "c": 34}
        for key, value := range kvs {
                fmt.Printf("Key %s => Value %d\n", key, value)
        }

        for index, letter := range "XYZ" {
                fmt.Printf("Index %d => Letter %c\n", index, letter)
        }
}
```

```
Numbers : [2 3 4]
Sum : 9
Index 0 => Value 2
Index 1 => Value 3
Index 2 => Value 4
Key a => Value 12
Key b => Value 23
Key c => Value 34
Index 0 => Letter X
Index 1 => Letter Y
Index 2 => Letter Z
```



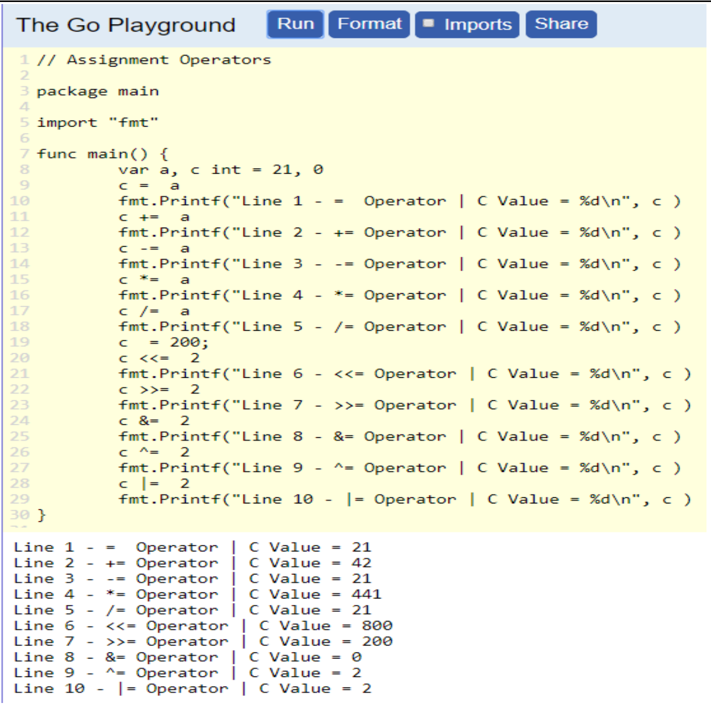
```go
// Logincal Operators

package main

import "fmt"

func check(a, b bool) {
        if a && b {
                fmt.Println("Line 1 | a && b | Condition is True")
        } else {
                fmt.Println("Line 1 | a && b | Condition is False")
        }
        if a || b {
                fmt.Println("Line 2 | a || b | Condition is True")
        } else {
                fmt.Println("Line 2 | a || b | Condition is False")
        }
        fmt.Println("Inverse of A is", !a)
}

func main() {
        var a, b bool = true, false
        fmt.Printf("A Value %t | B Balue %t\n", a, b)
        check(a, b)
        // Lets Change the Value of A and B.
        a, b = false, true
        fmt.Printf("A Value %t | B Balue %t\n", a, b)
        check(a, b)
}
```

```
A Value true | B Balue false
Line 1 | a && b | Condition is False
Line 2 | a || b | Condition is True
Inverse of A is false
A Value false | B Balue true
Line 1 | a && b | Condition is False
Line 2 | a || b | Condition is True
Inverse of A is true
```



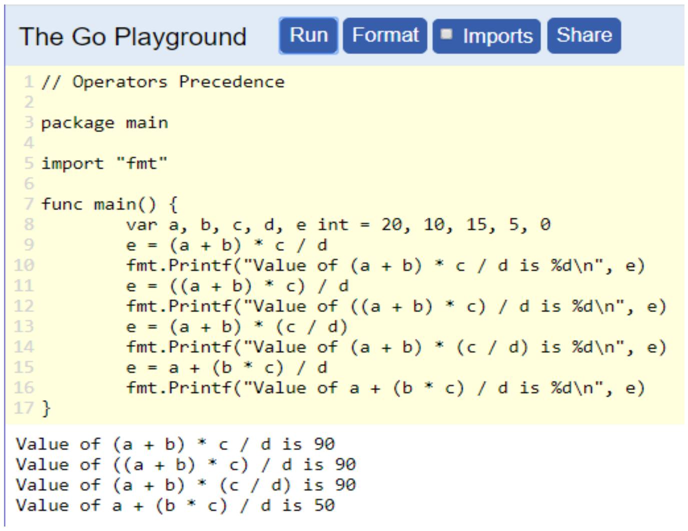
```go
// Bitwise Operators

package main

import "fmt"

func main() {
	//  60 = 0011 1100 | 13 = 0000 1101
	var a, b, c uint = 60, 13, 0
	c = a & b       /* 12 = 0000 1100 */
	fmt.Printf("Line 1 | C Value = %d\n", c)
	c = a | b       /* 61 = 0011 1101 */
	fmt.Printf("Line 2 | C Value = %d\n", c)
	c = a ^ b       /* 49 = 0011 0001 */
	fmt.Printf("Line 3 | C Value = %d\n", c)
	c = a << 2     /* 240 = 1111 0000 */
	fmt.Printf("Line 4 | C Value = %d\n", c)
	c = a >> 2     /* 15 = 0000 1111 */
	fmt.Printf("Line 5 | C Value = %d\n", c)
}
```

```
Line 1 | C Value = 12
Line 2 | C Value = 61
Line 3 | C Value = 49
Line 4 | C Value = 240
Line 5 | C Value = 15
```



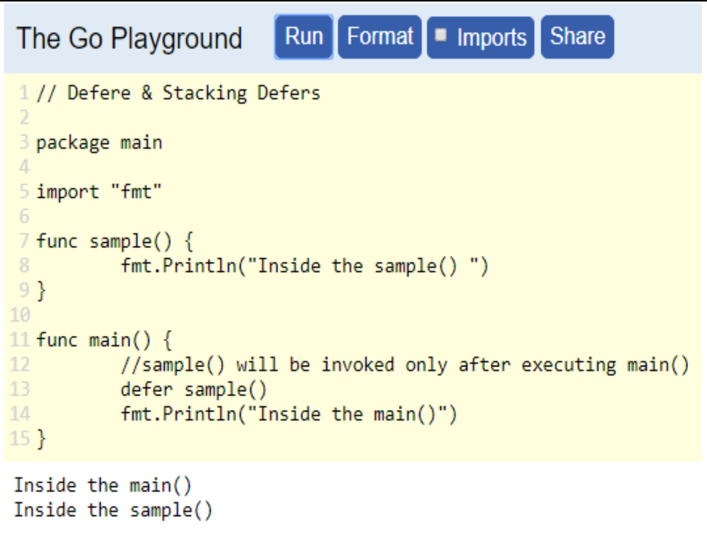
```go
// Assignment Operators

package main

import "fmt"

func main() {
	var a, c int = 21, 0
	c =  a
	fmt.Printf("Line 1 - =  Operator | C Value = %d\n", c )
	c +=  a
	fmt.Printf("Line 2 - += Operator | C Value = %d\n", c )
	c -=  a
	fmt.Printf("Line 3 - -= Operator | C Value = %d\n", c )
	c *=  a
	fmt.Printf("Line 4 - *= Operator | C Value = %d\n", c )
	c /=  a
	fmt.Printf("Line 5 - /= Operator | C Value = %d\n", c )
	c  = 200;
	c <<=  2
	fmt.Printf("Line 6 - <<= Operator | C Value = %d\n", c )
	c >>=  2
	fmt.Printf("Line 7 - >>= Operator | C Value = %d\n", c )
	c &=  2
	fmt.Printf("Line 8 - &= Operator | C Value = %d\n", c )
	c ^=  2
	fmt.Printf("Line 9 - ^= Operator | C Value = %d\n", c )
	c |=  2
	fmt.Printf("Line 10 - |= Operator | C Value = %d\n", c )
}
```

```
Line 1 - =  Operator | C Value = 21
Line 2 - += Operator | C Value = 42
Line 3 - -= Operator | C Value = 21
Line 4 - *= Operator | C Value = 441
Line 5 - /= Operator | C Value = 21
Line 6 - <<= Operator | C Value = 800
Line 7 - >>= Operator | C Value = 200
Line 8 - &= Operator | C Value = 0
Line 9 - ^= Operator | C Value = 2
Line 10 - |= Operator | C Value = 2
```



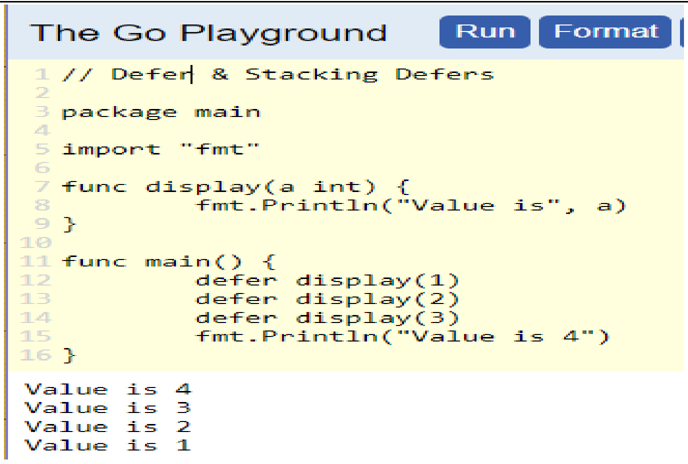
```go
// Operators Precedence

package main

import "fmt"

func main() {
	var a, b, c, d, e int = 20, 10, 15, 5, 0
	e = (a + b) * c / d
	fmt.Printf("Value of (a + b) * c / d is %d\n", e)
	e = ((a + b) * c) / d
	fmt.Printf("Value of ((a + b) * c) / d is %d\n", e)
	e = (a + b) * (c / d)
	fmt.Printf("Value of (a + b) * (c / d) is %d\n", e)
	e = a + (b * c) / d
	fmt.Printf("Value of a + (b * c) / d is %d\n", e)
}
```

```
Value of (a + b) * c / d is 90
Value of ((a + b) * c) / d is 90
Value of (a + b) * (c / d) is 90
Value of a + (b * c) / d is 50
```



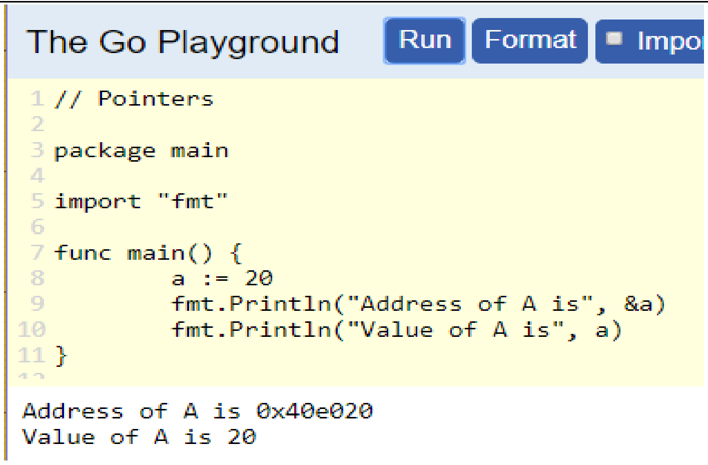
```go
// Defere & Stacking Defers

package main

import "fmt"

func sample() {
	fmt.Println("Inside the sample() ")
}

func main() {
	//sample() will be invoked only after executing main()
	defer sample()
	fmt.Println("Inside the main()")
}
```

```
Inside the main()
Inside the sample()
```



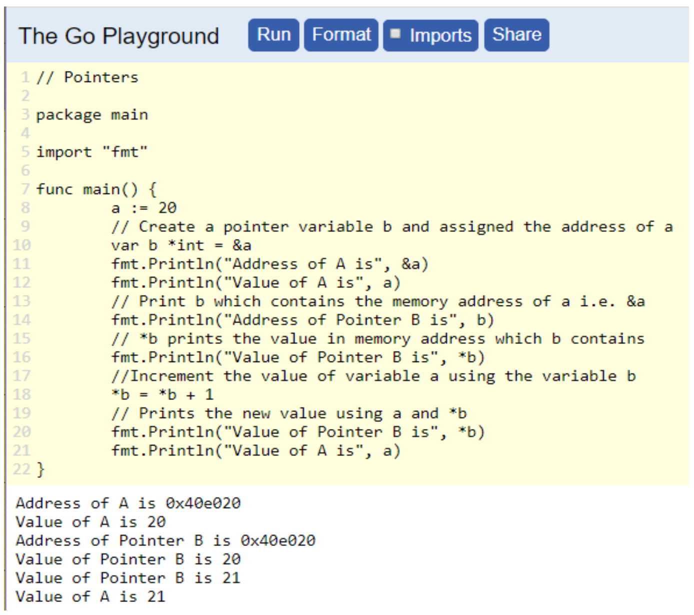
```go
// Defer & Stacking Defers

package main

import "fmt"

func display(a int) {
	fmt.Println("Value is", a)
}

func main() {
	defer display(1)
	defer display(2)
	defer display(3)
	fmt.Println("Value is 4")
}
```

```
Value is 4
Value is 3
Value is 2
Value is 1
```



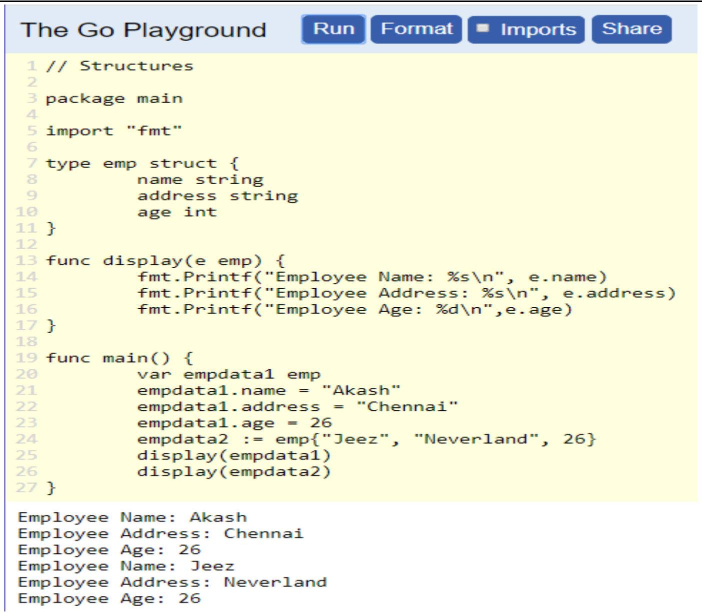
```go
// Pointers

package main

import "fmt"

func main() {
	a := 20
	fmt.Println("Address of A is", &a)
	fmt.Println("Value of A is", a)
}
```

```
Address of A is 0x40e020
Value of A is 20
```



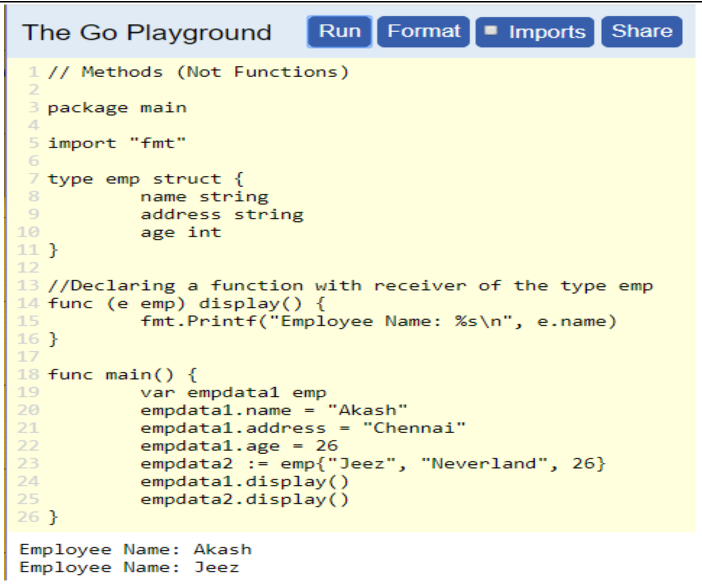
```go
// Pointers

package main

import "fmt"

func main() {
        a := 20
        // Create a pointer variable b and assigned the address of a
        var b *int = &a
        fmt.Println("Address of A is", &a)
        fmt.Println("Value of A is", a)
        // Print b which contains the memory address of a i.e. &a
        fmt.Println("Address of Pointer B is", b)
        // *b prints the value in memory address which b contains
        fmt.Println("Value of Pointer B is", *b)
        //Increment the value of variable a using the variable b
        *b = *b + 1
        // Prints the new value using a and *b
        fmt.Println("Value of Pointer B is", *b)
        fmt.Println("Value of A is", a)
}
```

```
Address of A is 0x40e020
Value of A is 20
Address of Pointer B is 0x40e020
Value of Pointer B is 20
Value of Pointer B is 21
Value of A is 21
```



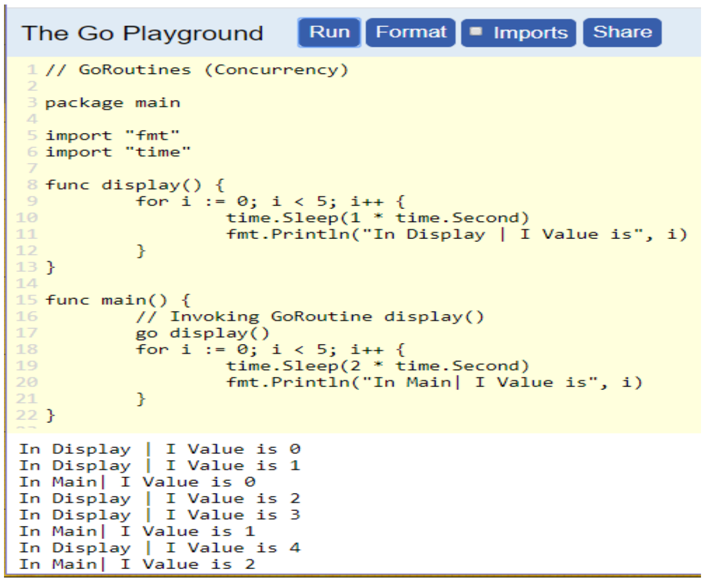
```go
// Structures

package main

import "fmt"

type emp struct {
        name string
        address string
        age int
}

func display(e emp) {
        fmt.Printf("Employee Name: %s\n", e.name)
        fmt.Printf("Employee Address: %s\n", e.address)
        fmt.Printf("Employee Age: %d\n",e.age)
}

func main() {
        var empdata1 emp
        empdata1.name = "Akash"
        empdata1.address = "Chennai"
        empdata1.age = 26
        empdata2 := emp{"Jeez", "Neverland", 26}
        display(empdata1)
        display(empdata2)
}
```

```
Employee Name: Akash
Employee Address: Chennai
Employee Age: 26
Employee Name: Jeez
Employee Address: Neverland
Employee Age: 26
```



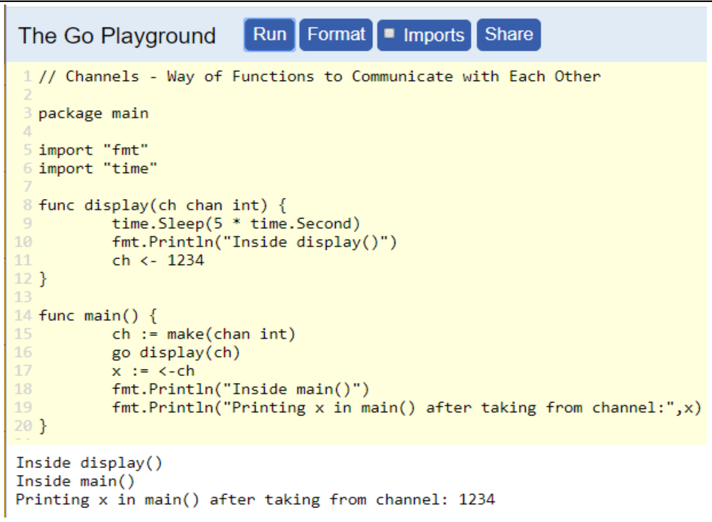
```go
// Methods (Not Functions)

package main

import "fmt"

type emp struct {
        name string
        address string
        age int
}

//Declaring a function with receiver of the type emp
func (e emp) display() {
        fmt.Printf("Employee Name: %s\n", e.name)
}

func main() {
        var empdata1 emp
        empdata1.name = "Akash"
        empdata1.address = "Chennai"
        empdata1.age = 26
        empdata2 := emp{"Jeez", "Neverland", 26}
        empdata1.display()
        empdata2.display()
}
```

```
Employee Name: Akash
Employee Name: Jeez
```



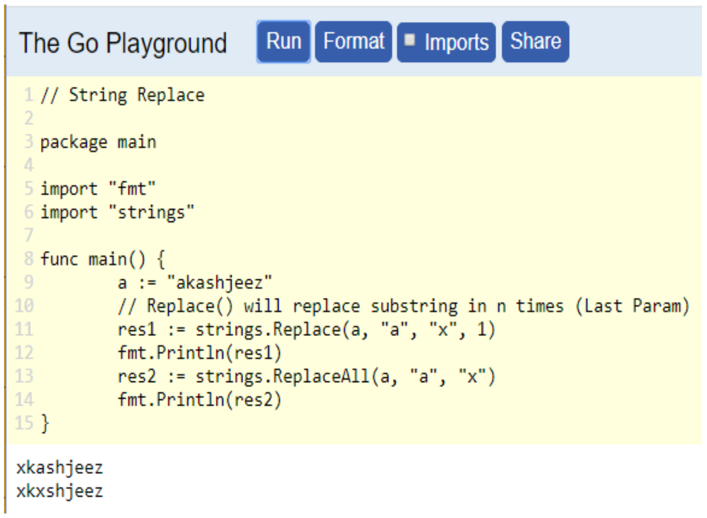
```go
// GoRoutines (Concurrency)

package main

import "fmt"
import "time"

func display() {
        for i := 0; i < 5; i++ {
                time.Sleep(1 * time.Second)
                fmt.Println("In Display | I Value is", i)
        }
}

func main() {
        // Invoking GoRoutine display()
        go display()
        for i := 0; i < 5; i++ {
                time.Sleep(2 * time.Second)
                fmt.Println("In Main| I Value is", i)
        }
}
```

```
In Display | I Value is 0
In Display | I Value is 1
In Main| I Value is 0
In Display | I Value is 2
In Display | I Value is 3
In Main| I Value is 1
In Display | I Value is 4
In Main| I Value is 2
```



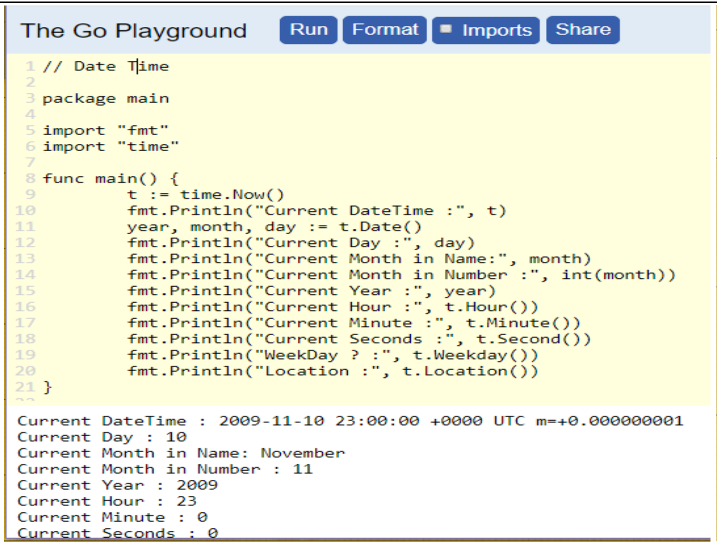
```go
// Channels - Way of Functions to Communicate with Each Other

package main

import "fmt"
import "time"

func display(ch chan int) {
        time.Sleep(5 * time.Second)
        fmt.Println("Inside display()")
        ch <- 1234
}

func main() {
        ch := make(chan int)
        go display(ch)
        x := <-ch
        fmt.Println("Inside main()")
        fmt.Println("Printing x in main() after taking from channel:",x)
}
```

```
Inside display()
Inside main()
Printing x in main() after taking from channel: 1234
```



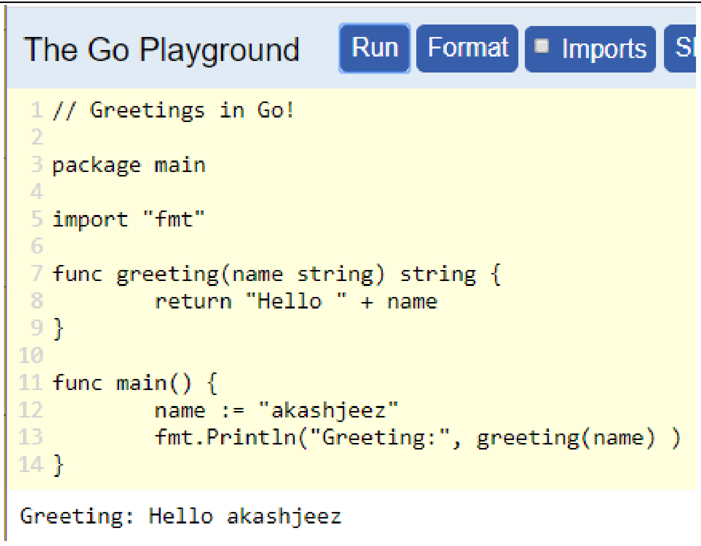
```go
// String Replace

package main

import "fmt"
import "strings"

func main() {
        a := "akashjeez"
        // Replace() will replace substring in n times (Last Param)
        res1 := strings.Replace(a, "a", "x", 1)
        fmt.Println(res1)
        res2 := strings.ReplaceAll(a, "a", "x")
        fmt.Println(res2)
}
```

```
xkashjeez
xkxshjeez
```



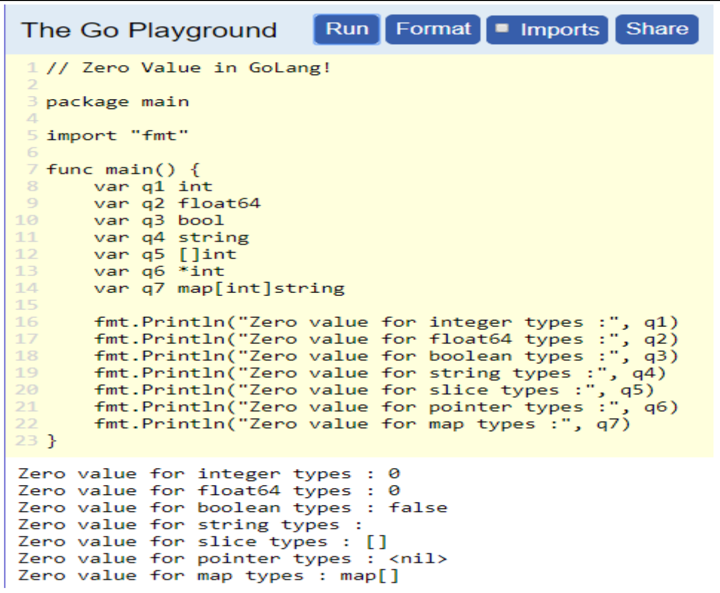
```
// Date Time

package main

import "fmt"
import "time"

func main() {
	t := time.Now()
	fmt.Println("Current DateTime :", t)
	year, month, day := t.Date()
	fmt.Println("Current Day :", day)
	fmt.Println("Current Month in Name:", month)
	fmt.Println("Current Month in Number :", int(month))
	fmt.Println("Current Year :", year)
	fmt.Println("Current Hour :", t.Hour())
	fmt.Println("Current Minute :", t.Minute())
	fmt.Println("Current Seconds :", t.Second())
	fmt.Println("WeekDay ? :", t.Weekday())
	fmt.Println("Location :", t.Location())
}
```

```
Current DateTime : 2009-11-10 23:00:00 +0000 UTC m=+0.000000001
Current Day : 10
Current Month in Name: November
Current Month in Number : 11
Current Year : 2009
Current Hour : 23
Current Minute : 0
Current Seconds : 0
```



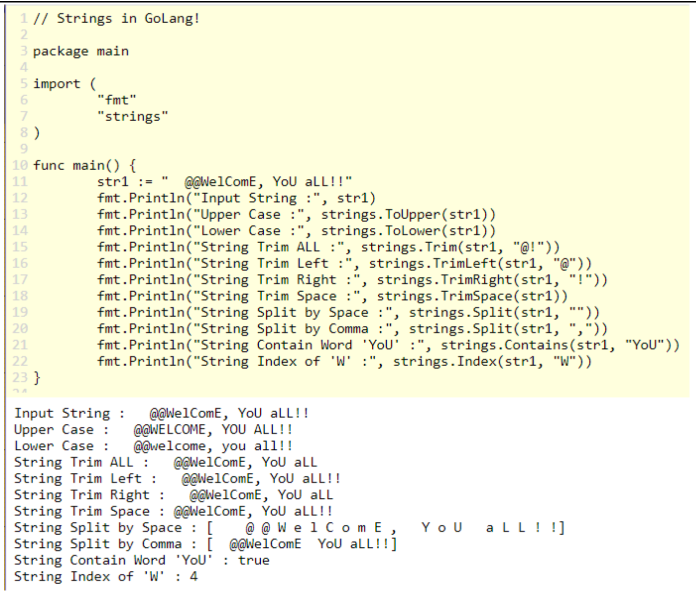
```
// Greetings in Go!

package main

import "fmt"

func greeting(name string) string {
	return "Hello " + name
}

func main() {
	name := "akashjeez"
	fmt.Println("Greeting:", greeting(name) )
}
```

```
Greeting: Hello akashjeez
```



```
// Zero Value in GoLang!

package main

import "fmt"

func main() {
    var q1 int
    var q2 float64
    var q3 bool
    var q4 string
    var q5 []int
    var q6 *int
    var q7 map[int]string

    fmt.Println("Zero value for integer types :", q1)
    fmt.Println("Zero value for float64 types :", q2)
    fmt.Println("Zero value for boolean types :", q3)
    fmt.Println("Zero value for string types :", q4)
    fmt.Println("Zero value for slice types :", q5)
    fmt.Println("Zero value for pointer types :", q6)
    fmt.Println("Zero value for map types :", q7)
}
```

```
Zero value for integer types : 0
Zero value for float64 types : 0
Zero value for boolean types : false
Zero value for string types :
Zero value for slice types : []
Zero value for pointer types : <nil>
Zero value for map types : map[]
```



```
// Strings in GoLang!

package main

import (
	"fmt"
	"strings"
)

func main() {
	str1 := "  @@WelComE, YoU aLL!!"
	fmt.Println("Input String :", str1)
	fmt.Println("Upper Case :", strings.ToUpper(str1))
	fmt.Println("Lower Case :", strings.ToLower(str1))
	fmt.Println("String Trim ALL :", strings.Trim(str1, "@!"))
	fmt.Println("String Trim Left :", strings.TrimLeft(str1, "@"))
	fmt.Println("String Trim Right :", strings.TrimRight(str1, "!"))
	fmt.Println("String Trim Space :", strings.TrimSpace(str1))
	fmt.Println("String Split by Space :", strings.Split(str1, ""))
	fmt.Println("String Split by Comma :", strings.Split(str1, ","))
	fmt.Println("String Contain Word 'YoU' :", strings.Contains(str1, "YoU"))
	fmt.Println("String Index of 'W' :", strings.Index(str1, "W"))
}
```

```
Input String :   @@WelComE, YoU aLL!!
Upper Case :   @@WELCOME, YOU ALL!!
Lower Case :   @@welcome, you all!!
String Trim ALL :   @@WelComE, YoU aLL
String Trim Left :   @@WelComE, YoU aLL!!
String Trim Right :   @@WelComE, YoU aLL
String Trim Space : @@WelComE, YoU aLL!!
String Split by Space : [    @ @ W e l C o m E ,   Y o U   a L L ! !]
String Split by Comma : [  @@WelComE  YoU aLL!!]
String Contain Word 'YoU' : true
String Index of 'W' : 4
```