# UNIT : 5

# Operators

- Arithmetic
- Comparison
- Logical

## Operators

- ➢ An operator is a symbol that tells the compiler to perform certain actions
- ➢ The following lists describe the different operators used in Golang
  - Arithmetic Operators
  - Assignment Operators
  - Comparison Operators
  - Logical Operators
  - Bitwise Operators

## Arithmetic Operators

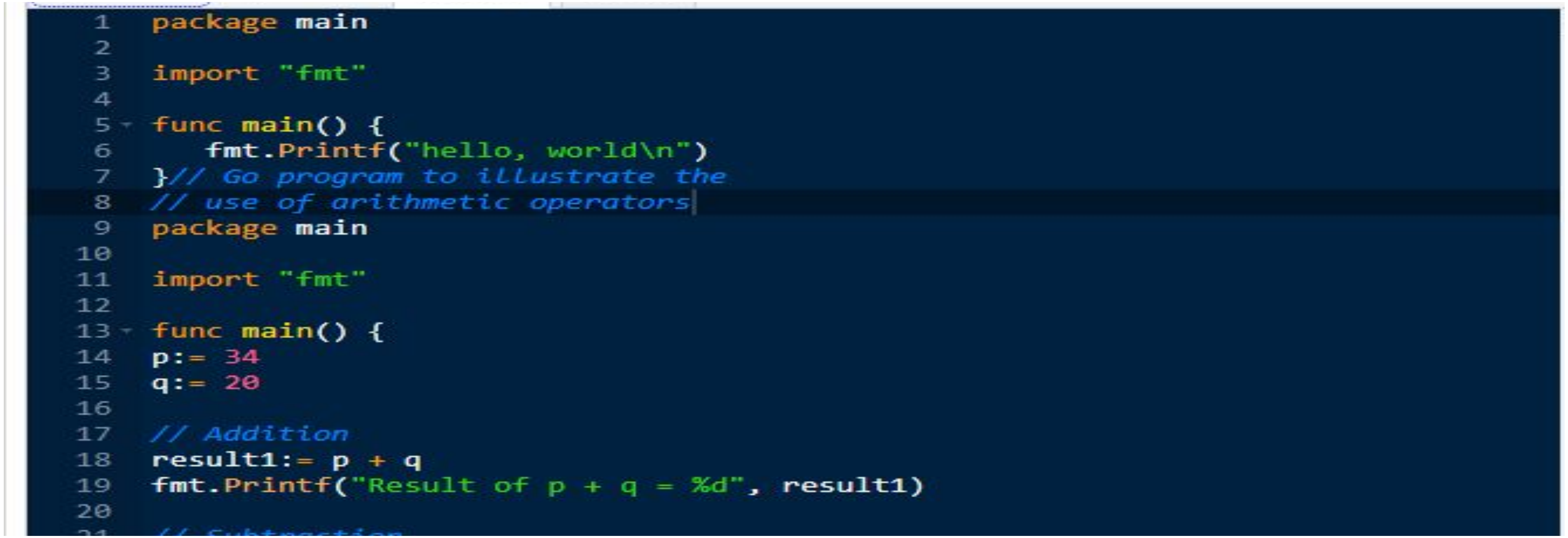
- The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc
- Arithmetic operators apply to numeric values and yield a result of the same type as the first operand
- The four standard arithmetic operators (+, -, *, /) apply to integer, floating-point, and complex types; + also applies to strings

## Arithmetic Operators

- ➢ Here's a complete list of arithmetic operators:

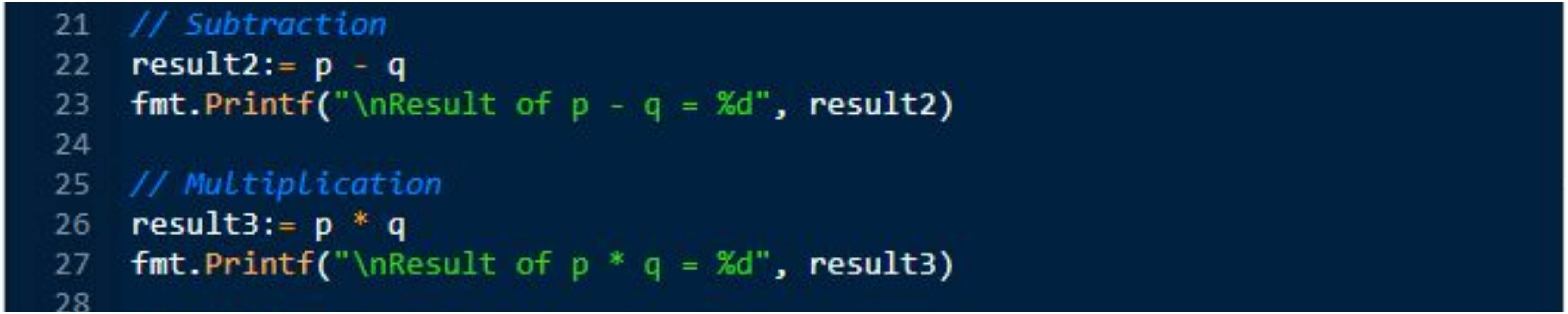
| Operator | Description | Example | Result |
|---|---|---|---|
| **+** | Addition | x + y | Sum of x and y |
| **-** | Subtraction | x - y | Subtracts one value from another |
| ***** | Multiplication | x * y | Multiplies two values |
| **/** | Division | x / y | Quotient of x and y |
| **%** | Modulus | x % y | Remainder of x divided by y |
| **++** | Increment | x++ | Increases the value of a variable by 1 |
| **--** | Decrement | x-- | Decreases the value of a variable by 1 |

## Example

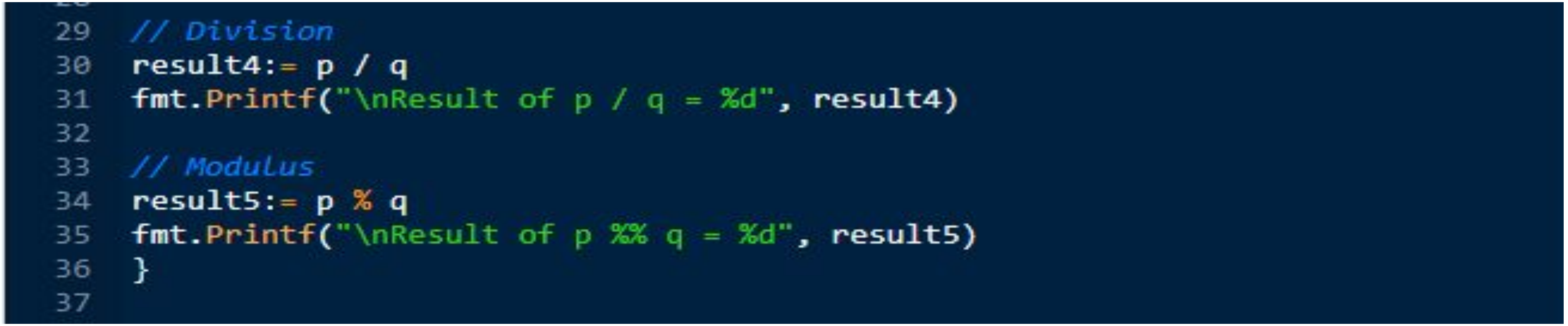


```go
package main

import "fmt"

func main() {
    fmt.Printf("hello, world\n")
}// Go program to illustrate the
// use of arithmetic operators
package main

import "fmt"

func main() {
p:= 34
q:= 20

// Addition
result1:= p + q
fmt.Printf("Result of p + q = %d", result1)

```

## Example

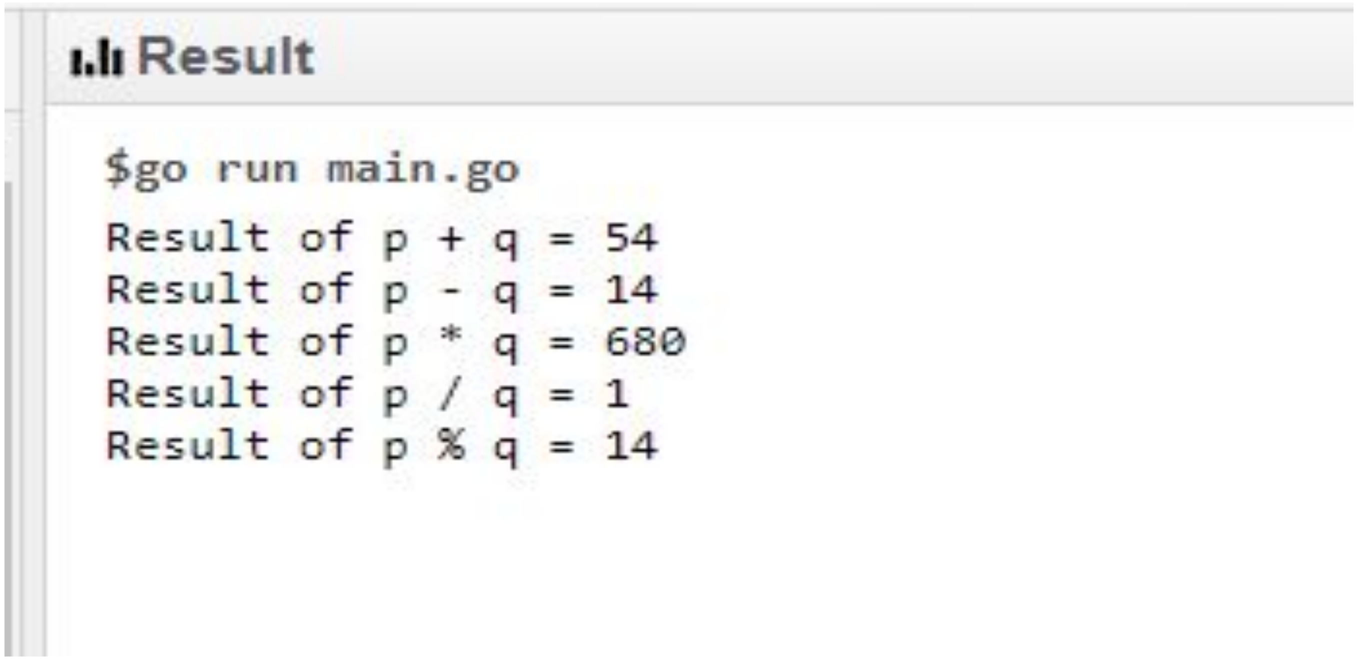


```go
// Subtraction
result2:= p - q
fmt.Printf("\nResult of p - q = %d", result2)

// Multiplication
result3:= p * q
fmt.Printf("\nResult of p * q = %d", result3)

```



```go
// Division
result4:= p / q
fmt.Printf("\nResult of p / q = %d", result4)

// Modulus
result5:= p % q
fmt.Printf("\nResult of p %% q = %d", result5)
}

```

## Example- output


Result

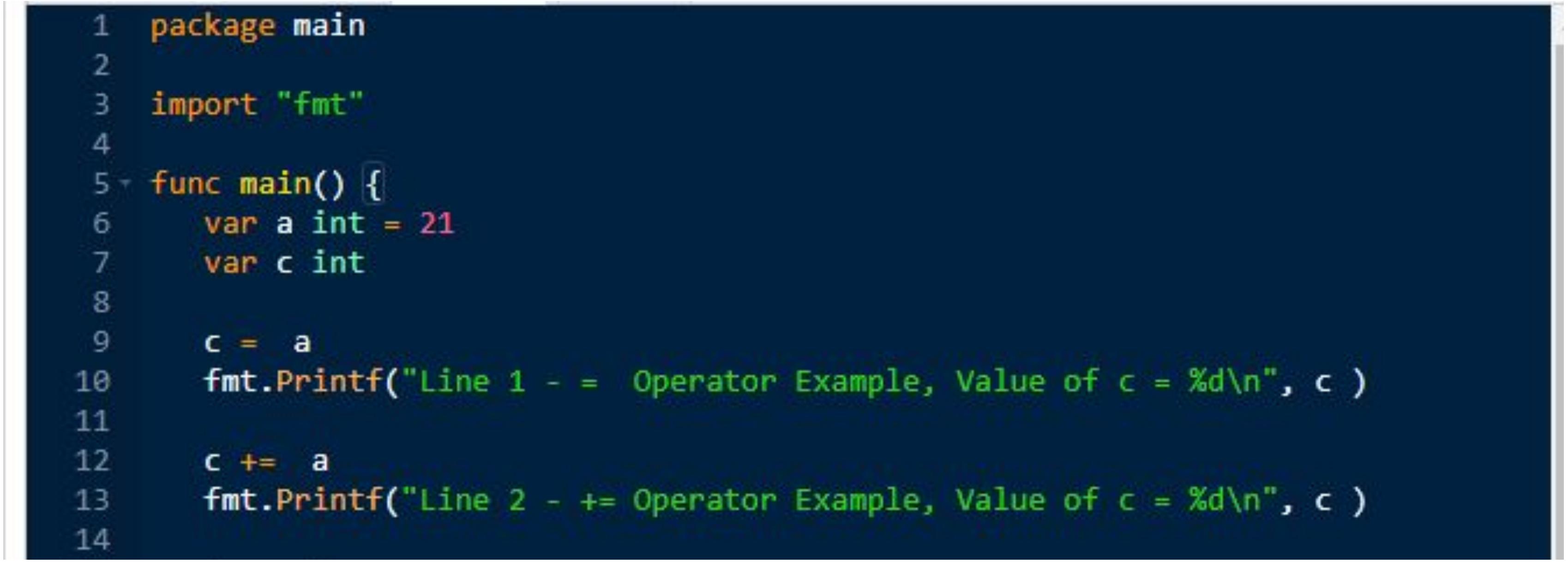
```
$go run main.go
Result of p + q = 54
Result of p - q = 14
Result of p * q = 680
Result of p / q = 1
Result of p % q = 14
```

## Assignment Operators

➢ The assignment operators are used to assign values to variables

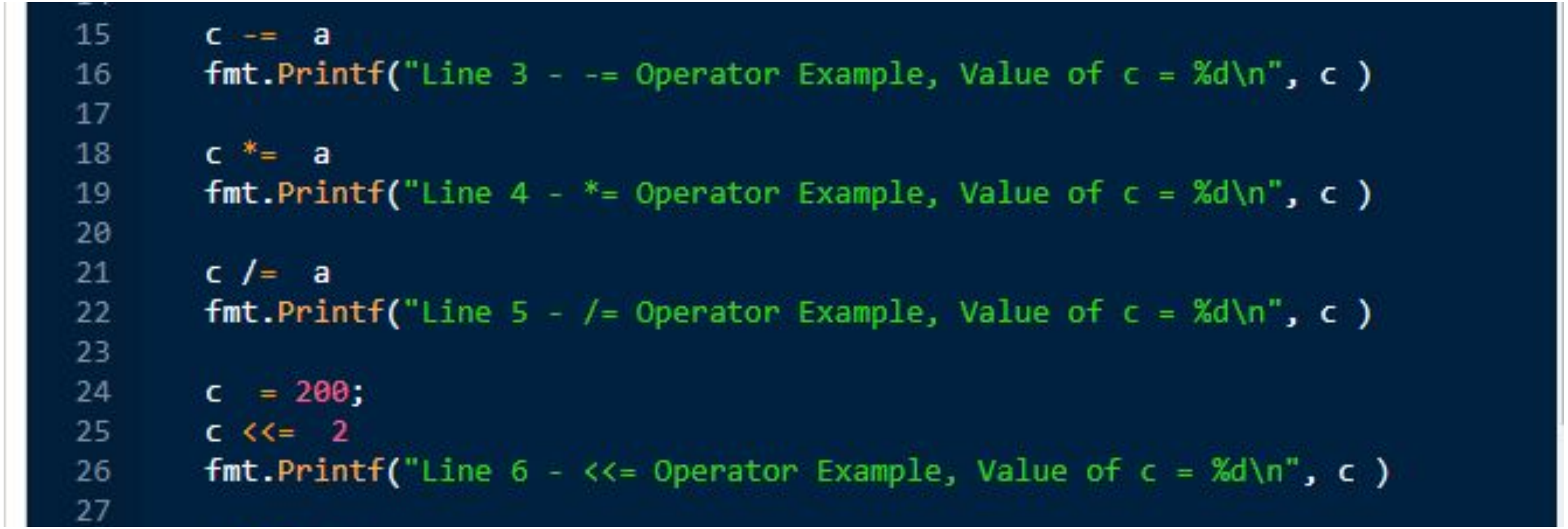
| Assignment | Description | Example |
|---|---|---|
| x = y | Assign x = y | Assign x = y |
| x += y | Add and assign | x = x + y |
| x -= y | Subtract and assign | x = x - y |
| x *= y | Multiply and assign | x = x * y |
| x /= y | Divide and assign quotient | x = x / y |
| x %= y | Divide and assign modulus | x = x % y |

## Example



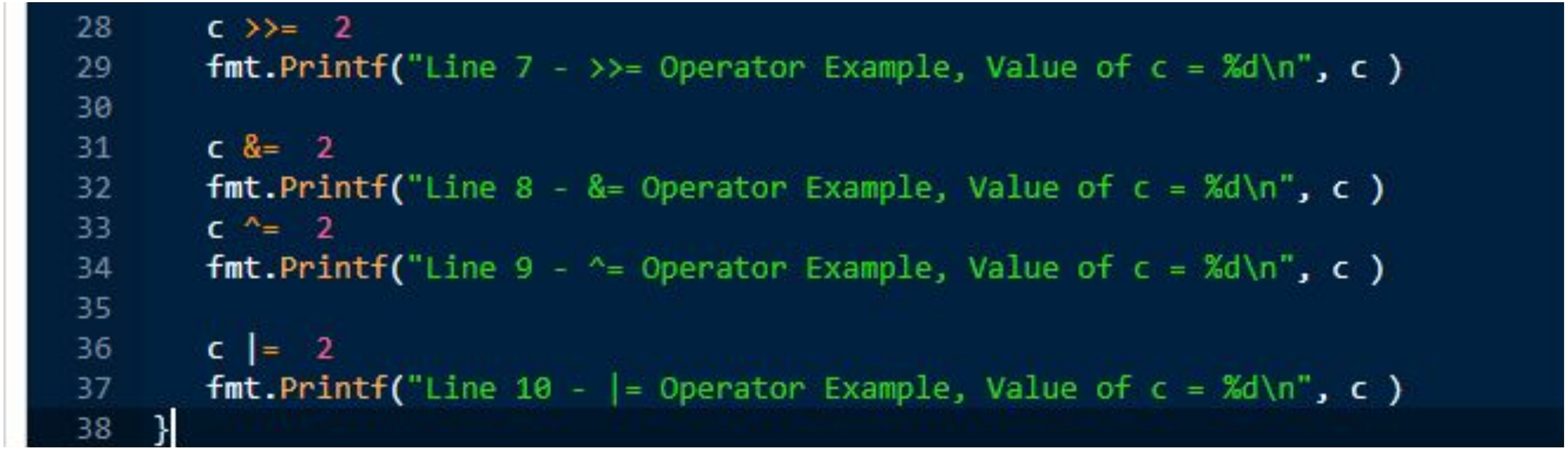
```go
package main

import "fmt"

func main() {
   var a int = 21
   var c int

   c =  a
   fmt.Printf("Line 1 - =  Operator Example, Value of c = %d\n", c )

   c +=  a
   fmt.Printf("Line 2 - += Operator Example, Value of c = %d\n", c )

```

## Example



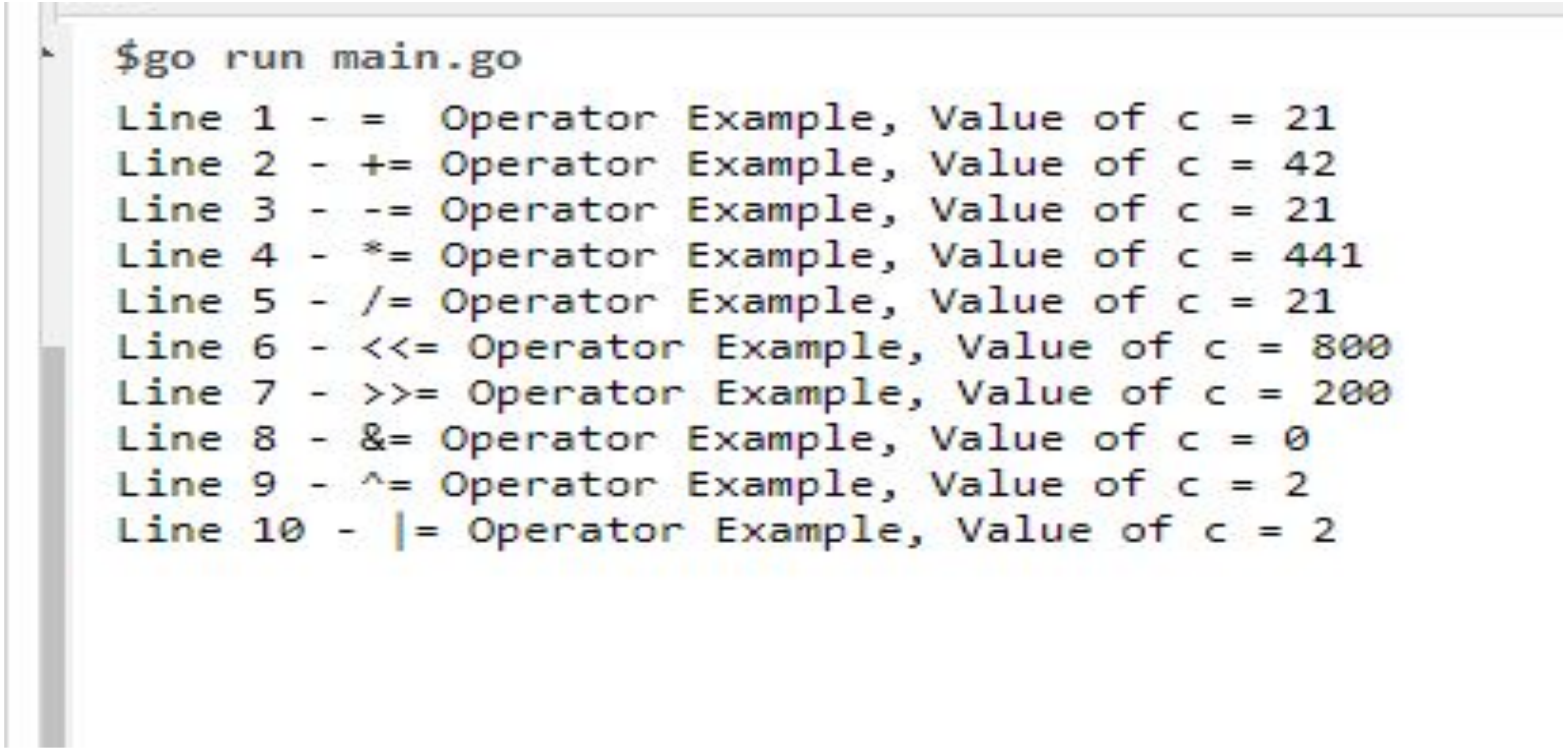
```
    c -=  a
    fmt.Printf("Line 3 - -= Operator Example, Value of c = %d\n", c )

    c *=  a
    fmt.Printf("Line 4 - *= Operator Example, Value of c = %d\n", c )

    c /=  a
    fmt.Printf("Line 5 - /= Operator Example, Value of c = %d\n", c )

    c  = 200;
    c <<=  2
    fmt.Printf("Line 6 - <<= Operator Example, Value of c = %d\n", c )

```

## Example



```
    c >>=  2
    fmt.Printf("Line 7 - >>= Operator Example, Value of c = %d\n", c )

    c &=  2
    fmt.Printf("Line 8 - &= Operator Example, Value of c = %d\n", c )
    c ^=  2
    fmt.Printf("Line 9 - ^= Operator Example, Value of c = %d\n", c )

    c |=  2
    fmt.Printf("Line 10 - |= Operator Example, Value of c = %d\n", c )
}
```

## Example- output



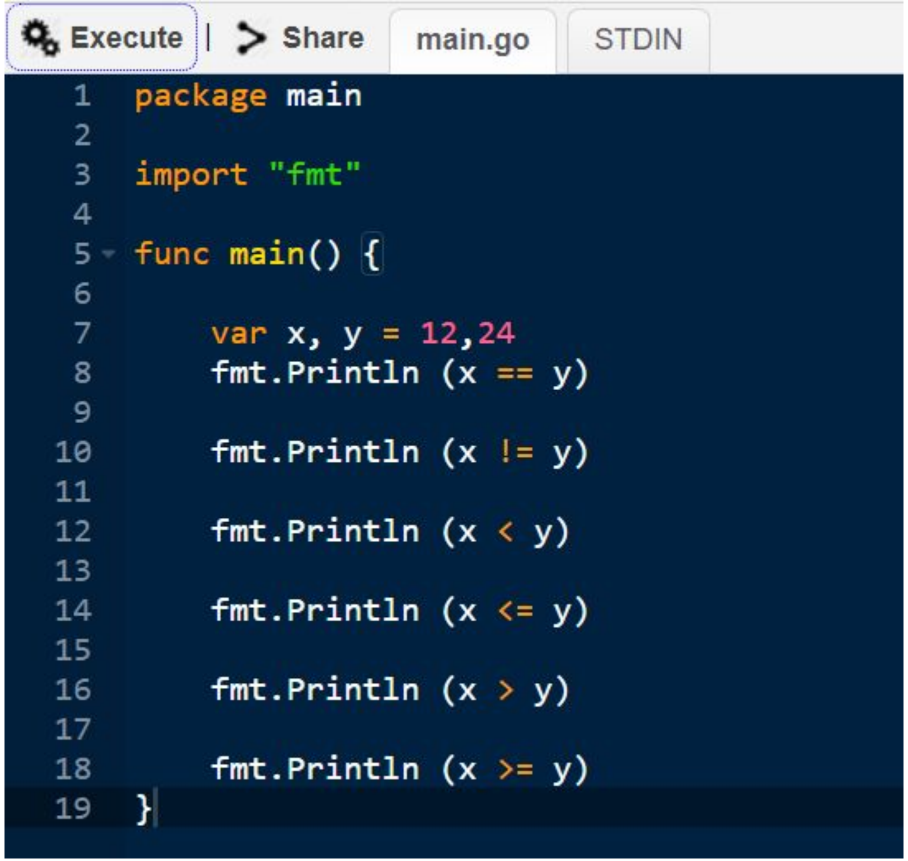
```
$go run main.go
Line 1 - =  Operator Example, Value of c = 21
Line 2 - += Operator Example, Value of c = 42
Line 3 - -= Operator Example, Value of c = 21
Line 4 - *= Operator Example, Value of c = 441
Line 5 - /= Operator Example, Value of c = 21
Line 6 - <<= Operator Example, Value of c = 800
Line 7 - >>= Operator Example, Value of c = 200
Line 8 - &= Operator Example, Value of c = 0
Line 9 - ^= Operator Example, Value of c = 2
Line 10 - |= Operator Example, Value of c = 2
```

## Comparison Operators

➢ Comparison operators are used to compare two values

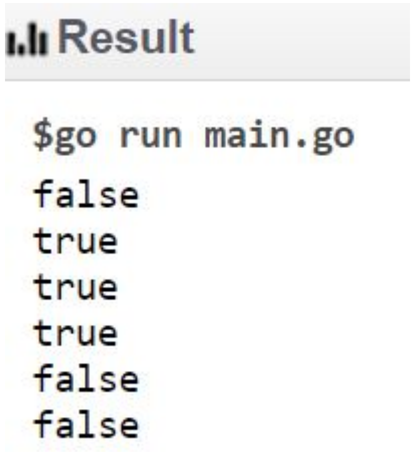
| Operator | Name | Example | Result |
|---|---|---|---|
| == | Equal | x == y | True if x is equal to y |
| != | Not equal | x != y | True if x is not equal to y |
| < | Less than | x < y | True if x is less than y |
| <= | Less than or equal to | x <= y | True if x is less than or equal to y |
| > | Greater than | x > y | True if x is greater than y |
| >= | Greater than or equal to | x >= y | True if x is greater than or equal to y |

## Example



```go
package main

import "fmt"

func main() {

    var x, y = 12,24
    fmt.Println (x == y)

    fmt.Println (x != y)

    fmt.Println (x < y)

    fmt.Println (x <= y)

    fmt.Println (x > y)

    fmt.Println (x >= y)
}
```

➢ **Output**



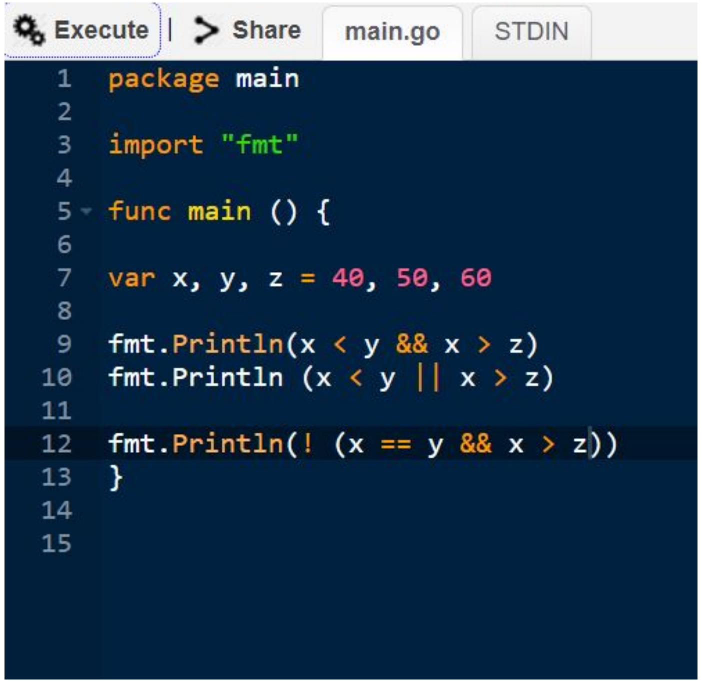
```
$go run main.go
false
true
true
true
false
false
```

## Logical Operators

➢ Logical operators are used to determine the logic between variables or values

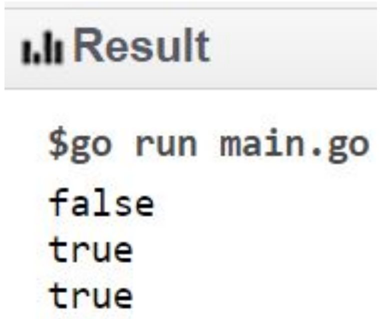
| Operator | Name | Description | Example |
|---|---|---|---|
| && | Logical And | Returns true if both statements are true | x < y && x > z |
| \|\| | Logical Or | Returns true if one of the statements is true | x < y \|\| x > z |
| ! | Logical Not | Reverse the result, returns false if the result is true | !(x == y && x > z) |

# Example



```go
package main

import "fmt"

func main () {

var x, y, z = 40, 50, 60

fmt.Println(x < y && x > z)
fmt.Println (x < y || x > z)

fmt.Println(! (x == y && x > z))
}
```

➢ **Output**



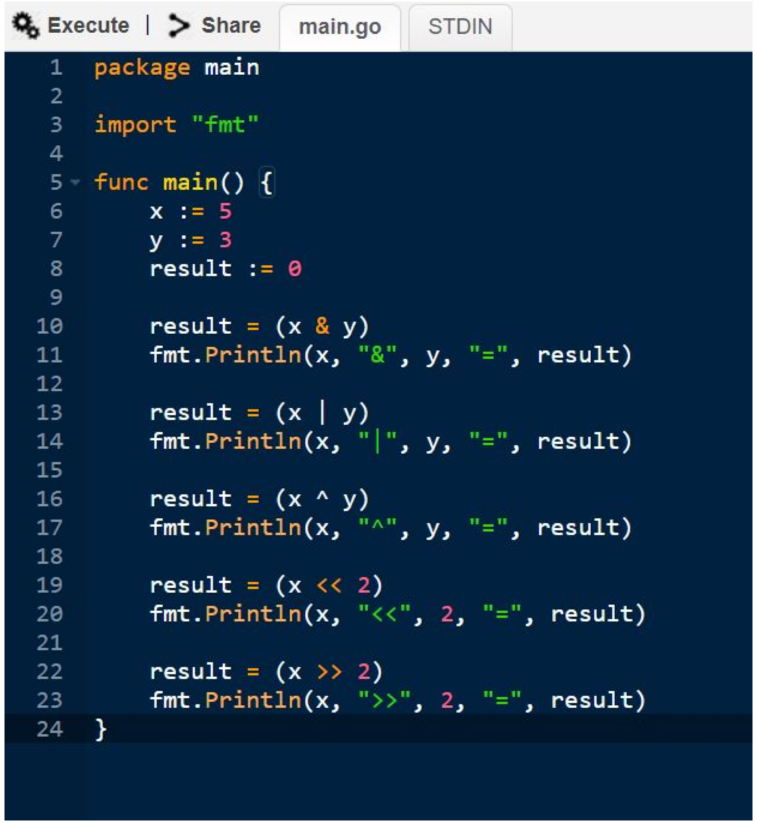
```
$go run main.go
false
true
true
```

## Bitwise Operators

➢ Bitwise operators are used to compare (binary) numbers

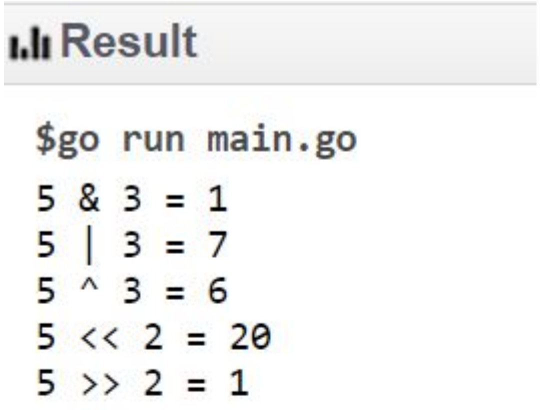
| Operator | Name | Description |
|---|---|---|
| **&** | AND | Sets each bit to 1 if both bits are 1 |
| **\|** | OR | Sets each bit to 1 if one of two bits is 1 |
| **^** | XOR | Sets each bit to 1 if only one of two bits is 1 |
| **<<** | Zero fill left shift | Shift left by pushing zeros in from the right and let the leftmost bits fall off |
| **>>** | Signed right shift | Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off |

## Example



```go
package main

import "fmt"

func main() {
	x := 5
	y := 3
	result := 0

	result = (x & y)
	fmt.Println(x, "&", y, "=", result)

	result = (x | y)
	fmt.Println(x, "|", y, "=", result)

	result = (x ^ y)
	fmt.Println(x, "^", y, "=", result)

	result = (x << 2)
	fmt.Println(x, "<<", 2, "=", result)

	result = (x >> 2)
	fmt.Println(x, ">>", 2, "=", result)
}
```

- ➢ **Output**



```
$go run main.go
5 & 3 = 1
5 | 3 = 7
5 ^ 3 = 6
5 << 2 = 20
5 >> 2 = 1
```