Day 19 - Go (Golang) Basics - Type Conversions and Operators

Type Conversion in Go

The process of converting one data type to another is known as **type conversion** or **type casting**.

Integer to Float

```go
package main

import "fmt"

func main() {
    var i int = 10
    var f float64 = float64(i)
    fmt.Printf("%.2f\n", f) // Output: 10.00
}
```

Float to Integer

Go allows this, but it **truncates the decimal part** and you lose precision.

```go
package main

import "fmt"

func main() {
    var y float64 = 10.7
    var x int = int(y)
    fmt.Printf("%d\n", x) // Output: 10
}
```

Using strconv Package

The `strconv` package helps with conversions between **strings and numeric types**.

`strconv.Itoa()`: Integer to String

```go
package main

import (
    "fmt"
    "strconv"
)

func main() {
    var i int = 42
    var str string = strconv.Itoa(i)
    fmt.Printf("%q\n", str) // Output: "42"
}
```

`strconv.Atoi()`: String to Integer

Returns two values: integer and error

```go
package main

import (
    "fmt"
    "strconv"
)

func main() {
    var str string = "42"
    i, err := strconv.Atoi(str)
    fmt.Printf("%v %T\n", i, i)       // Output: 42 int
    fmt.Printf("%v %T\n", err, err)   // Output: <nil> error
}
```

Operators in Go

Comparison Operators

Compare two values and return a boolean (`true` or `false`)

Operator	Description
==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Example:

```go
package main

import "fmt"

func main() {
  a := 10
  b := 20

  fmt.Println("a =", a)
  fmt.Println("b =", b)

  fmt.Println("a == b:", a == b)   // false - checks if a equals b
  fmt.Println("a != b:", a != b)   // true  - checks if a not equal to b
  fmt.Println("a < b:", a < b)     // true  - is a less than b
  fmt.Println("a <= b:", a <= b)   // true  - is a less than or equal to b
  fmt.Println("a > b:", a > b)     // false - is a greater than b
  fmt.Println("a >= b:", a >= b)   // false - is a greater than or equal to b
  }
```

Arithmetic Operators

Used to perform mathematical operations

Operator	Description
+	Addition / Concatenation
-	Subtraction
*	Multiplication
/	Division
%	Modulus (remainder)
++	Increment (unary)
--	Decrement (unary)

Example:

```go
package main

import "fmt"

func main() {
    a := 15
    b := 4

    fmt.Println("a =", a)
    fmt.Println("b =", b)

    // + Addition (also used for string concatenation)
    fmt.Println("a + b =", a + b)  // 19

    // - Subtraction
    fmt.Println("a - b =", a - b)  // 11

    // * Multiplication
    fmt.Println("a * b =", a * b)  // 60

    // / Division (integer division, fractional part is truncated)
    fmt.Println("a / b =", a / b)  // 3

    // % Modulus (remainder)
    fmt.Println("a % b =", a % b)  // 3

    // ++ Increment (unary) — only works as `a++`, not `++a`
    a++
    fmt.Println("a++ =>", a)       // 16

    // -- Decrement (unary)
    b--
    fmt.Println("b-- =>", b)       // 3
}

```

Logical Operators

Operator	Description
&&	Logical AND
||	Logical OR
!	Logical NOT

Example:

```go
package main

import "fmt"

func main() {
    a := true
    b := false

    fmt.Println("a =", a)
    fmt.Println("b =", b)

    // Logical AND: true only if both are true
    fmt.Println("a && b:", a && b)  // false

    // Logical OR: true if at least one is true
    fmt.Println("a || b:", a || b)  // true

    // Logical NOT: reverses the value
    fmt.Println("!a:", !a)          // false
    fmt.Println("!b:", !b)          // true
}

```

Assignment Operators

Used to assign and modify values in a variable

Operator	Description
=	Assignment
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modulus and assign

Example:

```go
package main
```

```go

import "fmt"

func main() {
    a := 10  // = Assignment
    fmt.Println("Initial value of a:", a)

    a += 5   // Add and assign → a = a + 5
    fmt.Println("After a += 5:", a) // 15

    a -= 3   // Subtract and assign → a = a - 3
    fmt.Println("After a -= 3:", a) // 12

    a *= 2   // Multiply and assign → a = a * 2
    fmt.Println("After a *= 2:", a) // 24

    a /= 4   // Divide and assign → a = a / 4
    fmt.Println("After a /= 4:", a) // 6

    a %= 5   // Modulus and assign → a = a % 5
    fmt.Println("After a %= 5:", a) // 1
}

```

Bitwise Operators

Operate on bits of integer types

Operator	Description
&	AND
|	OR
^	XOR
<<	Left shift
>>	Right shift

Example:

```go
package main

import "fmt"

func main() {
    a := 5  // binary: 0101
    b := 3  // binary: 0011

    fmt.Println("a =", a)
    fmt.Println("b =", b)

    // & Bitwise AND: Only 1 where both bits are 1
    fmt.Println("a & b =", a & b)  // 1  -> 0001

    // | Bitwise OR: 1 where either bit is 1
    fmt.Println("a | b =", a | b)  // 7  -> 0111

    // ^ Bitwise XOR: 1 where bits differ
    fmt.Println("a ^ b =", a ^ b)  // 6  -> 0110

    // << Left shift: shifts bits of a to the left by 1 (adds a zero to the right)
    fmt.Println("a << 1 =", a << 1)  // 10 -> 1010

    // >> Right shift: shifts bits of a to the right by 1 (removes rightmost bit)
    fmt.Println("a >> 1 =", a >> 1)  // 2 -> 0010
}

```