# Day 18 – Go (Golang) Basics-Scope, Zero Values, User Input, and Type Checking

## Variable Scope in Go

The scope of a variable is the region of the program where it is accessible.

In Go, scope is defined using **blocks**, which are groups of code enclosed in curly braces `{}`.

- Inner blocks can access variables from outer blocks
- Outer blocks **cannot** access variables from inner blocks

Example:

```go
package main

import "fmt"

func main() {
    var city string = "Islamabad"

    {
        var country string = "Pakistan"
        fmt.Println(country) // OK
        fmt.Println(city)    // OK
    }

    // fmt.Println(country) // Error: country is out of scope here
}
```

## Local Variables

Declared inside a function or block and accessible only within that block.

Example:

```go
package main

import "fmt"

func main() {
    var name string = "Fatima"
    fmt.Println(name)
}
```

## Global Variables

Declared outside any function or block. Accessible throughout the program.

Example:

```go
package main

import "fmt"

var name string = "Fatima"
```

```go

func main() {
    fmt.Println(name)
}
```

## Zero Values in Go

If a variable is declared but not initialized, it is automatically assigned a default value known as the **zero value**, depending on its type:

- `int` → `0`
- `float64` → `0.0`
- `bool` → `false`
- `string` → `""` (empty string)
- `pointers` → `nil`

Example:

```go
package main

import "fmt"

func main() {
    var a int
    var b string
    var c bool
    fmt.Println(a, b, c)
}
```

## Taking User Input with `fmt.Scanf`

Use `fmt.Scanf` to take user input.

Example 1: Taking an integer as input

```go
package main

import "fmt"

func main() {
    var a int
    fmt.Println("Enter a number:")
    fmt.Scanf("%d", &a)
    fmt.Println("You entered:", a)
}
```

Example 2: Taking a string and an integer

```go
package main

import "fmt"

func main() {
    var a string
    var b int
    fmt.Println("Enter a string and an int:")
    n, err := fmt.Scanf("%s %d", &a, &b)
    fmt.Println("Scanned:", n, "values")
    fmt.Println("Error:", err)
    fmt.Println("Values:", a, b)
}
```

Format Specifiers:

- `%d` : integer
- `%s` : string
- `%f` : float
- `%t` : boolean

## Finding the Type of a Variable

To check the type of a variable at runtime, use:

- `%T` with `fmt.Printf`
- `reflect.TypeOf(variable)` from `reflect` package

Example:

```go
package main

import (
    "fmt"
    "reflect"
)

func main() {
    var grade int = 88
    var message string = "Pass"

    fmt.Printf("%v is of type %T
", grade, grade)
    fmt.Println("Type using reflect:", reflect.TypeOf(message))
}
```