Data Types in Go

In Go programming language, data types is an extensive system used for declaring variables or functions of different types.

The type of variable determines how much space it occupies in storage and how the bit pattern stored is interpreted.

The DataTypes in Go are classified as :-

1. Boolean types :- Boolean data types are consists of two predefined constants: “true” and “false”
2. Numeric types :- Numeric are arithmetic data types and represents:- “Integer types” or “floating point” values throughout the program.
3. String types :- A string type represents the set of string values. Its value is a sequence of bytes. Strings are immutable types that is once created, it is not possible to change the contents of string.
4. Derived Types :- The include Pointer types, Array types, Structure types, Union types, Function types, Slice types, Interface types, Map types, Channel Types.

#Numeric types:-

a) Integer types :- the predefined architecture-independent integer type are –
   - unit8 :- Unsigned 8-bit integers (0 to 255)
   - unit16 :- Unsigned 16-bit integers (0 to 65535)
   - unit32 :- Unsigned 32-bit integers (0 to 65535)
   - unit64 :- Unsigned 64-bit integers (0 to 18446744073709551615)
   - ■ int8 :- Signed 8-bit integers (-128 to 127)
   - ■ int16 :- Signed 16-bit integers (-32768 to 32767)
   - ■ int32 :- Signed 32-bit integers (-2147483648 to 2147483647)
   - ■ int64 :- Signed 64-bit integers (-9223378068547775808 to 9223372036854775807)

b) Floating Types :- The predefined architecture-independent float types are-
   - float32 :- IEEE-754 32-bit floating-point numbers
   - float64 :- IEEE-754 64-bit floating-point numbers

- complex64 :- Complex numbers with float32 real and imaginary parts
- complex128 :- Complex numbers with float64 real and imaginary parts.

Note:- The value of an n-bit integers is n bits and is represented using two’s compliment arithmetic operations .

c) Other Numeric Types :- There is also a set of numeric types with implementation-sizes –
    - byte :- same as uit8
    - rune :- same as int32
    - unit :- 32 or 64 bits
    - int :- same size as unit
    - uintptr :- an unsigned integer to store the uninterpreted bits of a pointer value.

Variables in Go

A variable is nothing but a name given to a storage area that the programs can manipulate.

Each variable in go has specific type, which determines the size and layout of the variable's memory and range of values that can be stores within that memory.

Name of variable can be composed of letters, digits and underscores.

It must begin with letter or underscores.

Upper and lowercase letters are distinct because Go is case-sensitive.

# Variable definition in Go

A variable definition tells the compiler where and how much storage to create for the variable.

A variable definition specifies a datatype and contains a list of one or more variables of that type as follows –

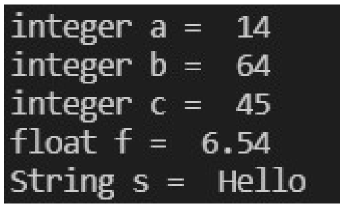
Syntax:-

```
var variable_list optional_data_type;
```

Example :-

```go
package main

import "fmt"

func main() {
    var a, b, c int
    a = 14
    b = 64
    c = 45

    var f float32
    f = 6.54

    var s string
    s = "Hello"

    fmt.Println("integer a = ", a)
    fmt.Println("integer b = ", b)
    fmt.Println("integer c = ", c)
    fmt.Println("float f = ", f)
    fmt.Println("String s = ", s)

}
```

Output:-

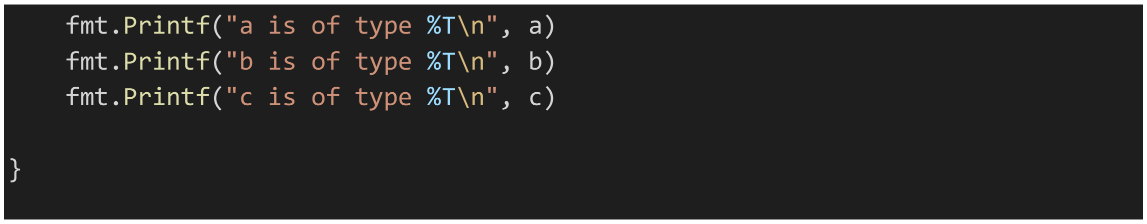


```
integer a =  14
integer b =  64
integer c =  45
float f =  6.54
String s =  Hello
```

Declaring data type is optional in Go language you can just declare the variable without data type.

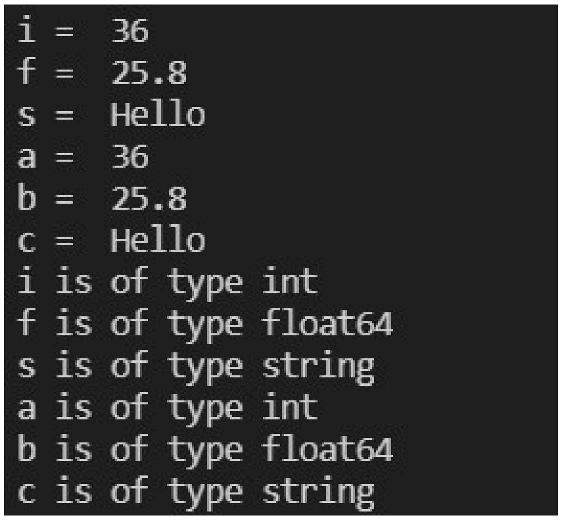
Example:-

```
package main

import "fmt"

func main() {
    var i = 36
    var f = 25.8
    var s = "Hello"

    // we can also declare all of them in one line like:-
    var a, b, c = 36, 25.8, "Hello"

    fmt.Println("i = ", i)
    fmt.Println("f = ", f)
    fmt.Println("s = ", s)
    fmt.Println("a = ", a)
    fmt.Println("b = ", b)
    fmt.Println("c = ", c)

    // to know the data types of the variable we can use printf and "%T"

    fmt.Printf("i is of type %T\n", i)
    fmt.Printf("f is of type %T\n", f)
    fmt.Printf("s is of type %T\n", s)
```



```
    fmt.Printf("a is of type %T\n", a)
    fmt.Printf("b is of type %T\n", b)
    fmt.Printf("c is of type %T\n", c)

}
```

Output :-



```
i =  36
f =  25.8
s =  Hello
a =  36
b =  25.8
c =  Hello
i is of type int
f is of type float64
s is of type string
a is of type int
b is of type float64
c is of type string
```