Friday, September 22, 2023 2:22 PM

BASIC TYPES:

- int, int8, int16, int32, int64
- uint, uint8, uint16, uint32, uint64
- float32, float64
- byte - alias for unit8
- rune - alias for int32 , represents unicode point
- bool, string
- complex64, complex128

NOTE: The value of an initialized variable with no assignment will be its zero-value

ZERO VALUES:

- `0` for numeric types,
- `false` for the boolean (bool) type
- "" (an empty string) for strings.
- (0+0i) is for complex numbers

VARIABLE DECLARATION:

Examples:

- var myNumber int
- var myString string
- var myFLoat float32
- var myFLoat float64
- var myBooleanVal bool
- var myComplexNum complex64

NOTE: Default values for above mentioned variables (in the same order) would be 0, "", 0, 0, false, (0+0i)

VARIABLE ASSIGNMENT AND SHORT VARIABLE DECLARATION:

- The **:=** short assignment operator infers the type of a new variable based on the value mentioned
- It replaces the var assignment to a variable
- **:=** or **=** used to declare a variable without specifying an explicit type

Examples for Variable Assignment:

```
• var myNumber int  = 12;
• var myString string = "Golang"
```

TYPE INFERENCE:

Examples:

```
• myNumber := 12               // type inferred here will be an integer
• myString := "Golang"         // type inferred here will be a string
• myComplexNumber :=0.867+0.5i // type would be complex128
• var pi = 3.1459     // it's a direct assignment here and type here would be a
  float
• var myVariable int
  j := myVariable  // here j is also an int
```

SAME LINE DECLARATIONS:

- Multiple variables can be declared on a same line
  E.g:

  ```
  myNumber, myString := 27, "Golang"
  ```

  The above same line variable declaration is same as below:

  ```
  myNumber := 23
  myString := "Golang"
  ```

TYPE SIZES:

Size: Indicates the number of bits that will be stored in the memory for a specific type

Standard Sizes used are:

- Int
- uint
- float64
- Complex128

TYPE CONVERSION:

Example:

- Integer to Float:
  - `testMileage := 35`
  - `actualMileage := float64(testMileage); // 35`

- Float to integer:
  - `testMileage := 3.5`
  - `actualMileage := int(testMileage); // 3`

NOTE: float to int conversion results in truncating the floating / decimal values.

TYPES THAT SHOULD BE USED:

If you do not have a specific requirement, following types should be used.

- bool
- string

- int
- iint32
- byte
- rune
- float64
- Complex128

CONSTANTS:

- These can be declared as Variables But we use const keyword
  `const myConstVariable = "GO"`
- Can be character, string, bool, or numeric values but not more complex types like slices, maps and structs
- Must be initialized once a variable is declared with const keyword
- Should not a declare a const variable that will be computed at runtime

FORMATTING STRINGS:

- GO follows C languages printf style
- **Sprinf()** method returns the formatted string
- ```
  package main

  import "fmt"

  func main() {
      const name = "Saul Goodman"
      const openRate = 30.5

      msg := fmt.Sprintf("Hi %s, your open rate is %f percent", name, openRate)

      fmt.Println(msg)
  }
  ```
- Output: Hi Saul Goodman, your open rate is 30.5 percent

Default Representation:

- %v is used here
- s :=fmt.Sprintf("I am %v years old",10)
  // I am 10 years old
- s :=fmt.Sprintf("I am %v years old","way too many")
  // I am way too many years old

Interpolate a decimal value:

- s :=fmt.Sprintf("I am %d years old",10)// I am 10 years old
  The ".2" rounds the number to 2 decimal places
- s :=fmt.Sprintf("I am %.2f years old",10.523) // I am 10.53 years old
  ```