Go Introduction

Developed By	Google In 2007
Featues	Open-Source , Fast, Statically-Typed Language, Efficient, Scalable, Concurrency and Automatic Garbage Collection
Purpose	Concurrent Software Applications

Variables: Used to store a data value of a particular type.

Syntax

var name type = expression	var n1 int = 10

1. Write a go program to add of two numbers.

Program	package main
	import "fmt"
	func main() {
	var n1 int = 10
	var n2 int = 10
	fmt.Println("sum")
	fmt.Println(n1 + n2)
	}
Output	Sum 20

Constants: These are fixed values.

Program	package main
	import "fmt"
	const name string = "wisdom materials"
	const age = 5
	func main() {
	fmt.Println(name)
	fmt.Println(age)
	}
Output	wisdom materials, 5
Program For Multilple Constants Declaration Block	package main
	import "fmt"
	const (
	name = "Mobile"
	age = 50
	)
	func main() {
	fmt.Println(name)
	fmt.Println(age)
	}
Output	wisdom materials, 5

Data Types

bool, Numeric(int, float and complex types) and string

Program

```
package main
import ("fmt")
func main() {
var n1 bool = true
var n2 int = 25
var n3 float32 = 25.25
var n4 string = "wisdom materials"

fmt.Println("Boolean: ", n1)
fmt.Println("Integer: ", n2)
fmt.Println("Float:   ", n3)
fmt.Println("String:  ", n4)
}
```

Output

```
Boolean:  true
Integer:  25
Float:    25.25
String:   wisdom materials
```

Operators
It is a symbol used to perform a particular operation.

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 variable value by 1
--	Decrement	x--	Decreases variable value by 1

Golang Conditional Statements

Conditional Statements	Details
if	executes a block of code if the condition is true
if...else	executes a block of code if a condition is true otherwise it will execute other a block of code if the condition is false
if...else if....else	executes different codes for more than two conditions
switch...case	selects one of many blocks of code to be executed

1. Write a program on if statement

	Program	**Output**

if statement	package main
	import ("fmt")
	func main() {
	var n1 int = 55
	var n2 int = 25
	if n1 > n2 {
	fmt.Println("n1 is big")
	}
	}
		n1 is big

if else statement	package main
	import ("fmt")
	
	func main() {
	var n1 int = 55
	var n2 int = 25
	if n1 > n2 {
	fmt.Println("n1 is big")
	}else {
	fmt.Println("n2 is big")
	}
	}
		n2 is big

if else statement	package main
	import ("fmt")
	func main() {
	var n1 int = 55
	var n2 int = 55
	if n1 > n2 {
	fmt.Println("n1 is big")
	} else if n1 > n2 {
	fmt.Println("n2 is big")
	} else {
	fmt.Println("both are same")
	}
	}
		both are same

Switch Case	package main
	import ("fmt")
	func main() {
	    var n1 int
	fmt.Print("Enter a Number :")
	fmt.Scan(&n1)
	
	switch n1 {
	case 1, 3, 5, 7:
	fmt.Println("Odd Day")
	case 2, 4, 6: Enter a Number :3
	Odd Day	

	fmt.Println("Even DAy")
	default:
	fmt.Println("Enter Correct information")
	}
	}	

For Loop in Golang

	Program	**Output**

Print numbers	package main
	import "fmt"
	func main() {
	var n1 int
	fmt.Print("Enter a Number :")
	fmt.Scan(&n1)
	
	for i := 1;
	i <= n1; i++ {
	fmt.Println(i)
	}
	}
		Enter a Number :5
		1
		2
		3
		4
		5

range	package main
	import "fmt"
	func main() {
	odds := [9]int{1, 2, 3, 4, 5, 6, 7, 8, 9}
	  for i, abcd := range odds {
	    fmt.Printf("odds %d = %d \n", i, abcd)
	 }
	}
		odds 0 = 1
		odds 1 = 2
		odds 2 = 3
		odds 3 = 4
		odds 4 = 5
		odds 5 = 6
		odds 6 = 7
		odds 7 = 8
		odds 8 = 9

Function
It is a set of lines of used to perform a particular task.

	Program	Output

Hello World	package main
	import "fmt"
	
	func helloworld() {
	fmt.Println("Hello World program using go")
	}
	
	func main() {
	helloworld()
	}
		Hello World program using go

Parameter passing to	package main import "fmt"	5

functions and return	
	func add(n1 int, n2 int) int {
	        sum := 0
	        sum = n1 + n2
	        return sum
	}
	
	func main() {
	        result := add(2, 3)
	        fmt.Println(result)
	}	

Returning Multiple Values	package main
	import "fmt"
	
	func triangle(a int, b int, c int) (Tarea int, Tparameter int) {
	        Tparameter = a+b+c
	        Tarea = (a+b)/2
	        return
	}
	
	func main() {
	        var ta, tp int
	        ta, tp = triangle(2, 3, 4)
	        fmt.Println("Tarea:", ta)
	        fmt.Println("Tparameter:", tp)
	}
		Tarea: 2
		Tparameter: 9

Passing Address to a Function	package main
	import "fmt"
	
	func update(n1 *int, t1 *string) {
	        *n1 = *n1 + 1
	        *t1 = *t1 + " materials"
	        return
	}
	
	func main() {
	        var id = 998
	        var name = "wisdom"
	        fmt.Println("Before:", name, id)
	
	        update(&id, &name)
	        fmt.Println("After :", name, id)
	}
		Before: wisdom 998
		After : wisdom materials 999

Variadic Functions (Accepts a	package main import "fmt"	mango orange

variable number of arguments.)	func main() {
	    variadicfunctionExample("mango", "apple", "bannana", "orange")
	}
	
	func variadicfunctionExample(fruitslist ...string) {
	    fmt.Println(fruitslist[0])
	    fmt.Println(fruitslist[3])
	}	

Passing multiple values	package main
	import "fmt"
	
	func main() {
	    variadicfunctionExample()
	    variadicfunctionExample("mango", "apple")
	    variadicfunctionExample("mango", "apple", "bannana")
	    variadicfunctionExample("mango", "apple", "bannana", "orange")
	}
	
	func variadicfunctionExample(fruitslist ...string) {
	    fmt.Println(fruitslist)
	}
		[]
		[mango apple]
		[mango apple bannana]
		[mango apple bannana orange]

Normal with Variadic parameters	package main
	import "fmt"
	func main() {
	    fmt.Println(calculation("Rectangle", 1, 4))
	}
	
	func calculation(str string, y ...int) int {
	    for _, val := range y {
	    fmt.Println("Wisdom materials", val)
	    }
	    return 9
	}
		Wisdom materials 1
		Wisdom materials 4
		9

Arrays in GoLang
It is collection of elements of similar data type. Elements of the array are addressed using the index.

	Program	Output
Creation	package main	wisdom

String array	import "fmt"
	
	func main() {
	    var myarrary [3]string
	    myarrary[0] = "wisdom"
	    myarrary[1] = "materials"
	    myarrary[2] = "company"
	
	    fmt.Println(myarrary[0])
	    fmt.Println(myarrary[1])
	    fmt.Println(myarrary[2])
	}
		materials
		company

Creation
 	
int array
	package main
	import "fmt"
	
	func main() {
	    myarrary1 := [5]int{1, 2, 3, 4, 5}
	    var myarrary2 [5]int = [5]int{6, 7, 8, 9}
	    fmt.Println(myarrary1)
	    fmt.Println(myarrary2)
	}
		[1 2 3 4 5]
		[6 7 8 9 0]

Access Array Elements	package main
	import "fmt"
	
	func main() {
	    myarrary := [5]int{1, 2, 3, 4, 5}
	
	    fmt.Println("\nArrary Elements\n")
	    for i := 0; i < len(myarrary); i++ {
	        fmt.Println(myarrary[i])
	    }
	}
		Arrary Elements
		
		1
		2
		3
		4
		5

Check if Element Exists		