GO - BASIC SYNTAX

You have seen a basic structure of Go program, so it will be easy to understand other basic building blocks of the Go programming language.

Tokens in Go

A Go program consists of various tokens and a token is either a keyword, an identifier, a constant, a string literal, or a symbol. For example, the following Go statement consists of six tokens:

```
fmt.Println("Hello, World!")
```

The individual tokens are:

```
fmt
.
Println
(
"Hello, World!"
)
```

Line Seperator

In Go program, the line seperator key is a statement terminator. That is, each individual statement don't need a special seperator like ; in C. Go compiler internally places ; as statement terminator to indicate the end of one logical entity.

For example, following are two different statements:

```
fmt.Println("Hello, World!")
fmt.Println("I am in Go Programming World!")
```

Comments

Comments are like helping text in your Go program and they are ignored by the compiler. They start with /* and terminates with the characters */ as shown below:

```
/* my first program in Go */
```

You cannot have comments within comments and they do not occur within a string or character literals.

Identifiers

A Go identifier is a name used to identify a variable, function, or any other user-defined item. An identifier starts with a letter A to Z or a to z or an underscore _ followed by zero or more letters, underscores, and digits $0 to 9$.

identifier = letter { letter | unicode_digit } .

Go does not allow punctuation characters such as @, $, and % within identifiers. Go is a **case sensitive** programming language. Thus, *Manpower* and *manpower* are two different identifiers in Go. Here are some examples of acceptable identifiers:

```
mahesh   kumar   abc   move_name   a_123
myname50   _temp   j   a23b9   retVal
```

Keywords

The following list shows the reserved words in Go. These reserved words may not be used as

constant or variable or any other identifier names.

break	default	func	interface	select
case	defer	go	map	struct
chan	else	goto	package	switch
const	fallthrough	if	range	type
continue	for	import	return	var

Whitespace in Go

A line containing only whitespace, possibly with a comment, is known as a blank line, and a Go compiler totally ignores it.

Whitespace is the term used in Go to describe blanks, tabs, newline characters and comments. Whitespace separates one part of a statement from another and enables the compiler to identify where one element in a statement, such as int, ends and the next element begins. Therefore, in the following statement:

```
var age int;
```

There must be at least one whitespace character *usuallyaspace* between int and age for the compiler to be able to distinguish them. On the other hand, in the following statement:

```
fruit = apples + oranges;   // get the total fruit
```

No whitespace characters are necessary between fruit and =, or between = and apples, although you are free to include some if you wish for readability purpose.