LISP - BASIC SYNTAX

Basic Building Blocks in LISP

LISP programs are made up of three basic building blocks:

- atom
- list
- string

An **atom** is a number or string of contiguous characters. It includes numbers and special characters.

Following are examples of some valid atoms:

```
hello-from-tutorials-point
name
123008907
*hello*
Block#221
abc123
```

A **list** is a sequence of atoms and/or other lists enclosed in parentheses.

Following are examples of some valid lists:

```
( i am a list)
(a ( a b c) d e fgh)
(father tom ( susan bill joe))
(sun mon tue wed thur fri sat)
( )
```

A **string** is a group of characters enclosed in double quotation marks.

Following are examples of some valid strings:

```
" I am a string"
"a ba c d efg #$%^&!"
"Please enter the following details :"
"Hello from 'Tutorials Point'! "
```

Adding Comments

The semicolon symbol ; is used for indicating a comment line.

For Example,

```
(write-line "Hello World") ; greet the world

; tell them your whereabouts

(write-line "I am at 'Tutorials Point'! Learning LISP")
```

When you click the Execute button, or type Ctrl+E, LISP executes it immediately and the result returned is:

```
Hello World

I am at 'Tutorials Point'! Learning LISP
```

Some Notable Points before Moving to Next

Following are some of the important points to note:

- The basic numeric operations in LISP are +, -, *, and /
- LISP represents a function call fx as fx, for example cos45 is written as cos 45
- LISP expressions are case-insensitive, cos 45 or COS 45 are same.
- LISP tries to evaluate everything, including the arguments of a function. Only three types of elements are constants and always return their own value
    - Numbers
    - The letter **t,** that stands for logical true.
    - The value **nil,** that stands for logical false, as well as an empty list.

Little More about LISP Forms

In the previous chapter, we mentioned that the evaluation process of LISP code takes the following steps.

- The reader translates the strings of characters to LISP objects or **s-expressions.**
- The evaluator defines syntax of Lisp **forms** that are built from s-expressions. This second level of evaluation defines a syntax that determines which s-expressions are LISP forms.

Now, a LISP forms could be.

- An Atom
- An empty or non-list
- Any list that has a symbol as its first element

The evaluator works as a function that takes a valid LISP form as an argument and returns a value. This is the reason why we put the **LISP expression in parenthesis,** because we are sending the entire expression/form to the evaluator as arguments.

Naming Conventions in LISP

Name or symbols can consist of any number of alphanumeric characters other than whitespace, open and closing parentheses, double and single quotes, backslash, comma, colon, semicolon and vertical bar. To use these characters in a name, you need to use escape character (\).

A name can have digits but not entirely made of digits, because then it would be read as a number. Similarly a name can have periods, but can't be made entirely of periods.

Use of Single Quotation Mark

LISP evaluates everything including the function arguments and list members.

At times, we need to take atoms or lists literally and don't want them evaluated or treated as function calls.

To do this, we need to precede the atom or the list with a single quotation mark.

The following example demonstrates this.

Create a file named main.lisp and type the following code into it.

```
(write-line "single quote used, it inhibits evaluation")
(write '(* 2 3))
(write-line " ")
(write-line "single quote not used, so expression evaluated")
(write (* 2 3))
```

When you click the Execute button, or type Ctrl+E, LISP executes it immediately and the result returned is:

```
single quote used, it inhibits evaluation
(* 2 3)
single quote not used, so expression evaluated
6
```