LISP - PROGRAM STRUCTURE

LISP expressions are called symbolic expressions or s-expressions. The s-expressions are composed of three valid objects, atoms, lists and strings.

Any s-expression is a valid program.

LISP programs run either on an **interpreter** or as **compiled code.**

The interpreter checks the source code in a repeated loop, which is also called the read-evaluate-print loop *REPL*. It reads the program code, evaluates it, and prints the values returned by the program.

A Simple Program

Let us write an s-expression to find the sum of three numbers 7, 9 and 11. To do this, we can type at the interpreter prompt.

```
(+7911)
```

LISP returns the result:

```
27
```

If you would like to run the same program as a compiled code, then create a LISP source code file named myprog.lisp and type the following code in it.

```
(write (+ 7 9 11))
```

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

```
27
```

LISP Uses Prefix Notation

You might have noted that LISP uses **prefix notation.**

In the above program the + symbol works as the function name for the process of summation of the numbers.

In prefix notation, operators are written before their operands. For example, the expression,

```
a * ( b + c ) / d
```

will be written as:

```
(/ (* a (+ b c) ) d)
```

Let us take another example, let us write code for converting Fahrenheit temp of 60o F to the centigrade scale:

The mathematical expression for this conversion will be:

```
(60 * 9 / 5) + 32
```

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

```
(write(+ (* (/ 9 5) 60) 32))
```

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

```
140
```

Evaluation of LISP Programs

Evaluation of LISP programs has two parts:

- Translation of program text into Lisp objects by a reader program
- Implementation of the semantics of the language in terms of these objects by an evaluator program

The evaluation process 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.
- 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.

The 'Hello World' Program

Learning a new programming language doesn't really take off until you learn how to greet the entire world in that language, right!

So, please create new source code file named main.lisp and type the following code in it.

```
(write-line "Hello World")

(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
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js