LISP - VARIABLES

In LISP, each variable is represented by a **symbol**. The variable's name is the name of the symbol and it is stored in the storage cell of the symbol.

Global Variables

Global variables have permanent values throughout the LISP system and remain in effect until a new value is specified.

Global variables are generally declared using the **defvar** construct.

For example

```
(defvar x 234)
(write x)
```

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

```
234
```

Since there is no type declaration for variables in LISP, you directly specify a value for a symbol with the **setq** construct.

For Example

```
->(setq x 10)
```

The above expression assigns the value 10 to the variable x. You can refer to the variable using the symbol itself as an expression.

The **symbol-value** function allows you to extract the value stored at the symbol storage place.

For Example

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

```
(setq x 10)
(setq y 20)
(format t "x = ~2d y = ~2d ~%" x y)

(setq x 100)
(setq y 200)
(format t "x = ~2d y = ~2d" x y)
```

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

```
x = 10 y = 20
x = 100 y = 200
```

Local Variables

Local variables are defined within a given procedure. The parameters named as arguments within a function definition are also local variables. Local variables are accessible only within the respective function.

Like the global variables, local variables can also be created using the **setq** construct.

There are two other constructs - **let** and **prog** for creating local variables.

The let construct has the following syntax.

```
(let ((var1  val1) (var2  val2).. (varn  valn)))
```

Where var1, var2, ..varn are variable names and val1, val2, .. valn are the initial values assigned to the respective variables.

When **let** is executed, each variable is assigned the respective value and lastly the *s-expression* is evaluated. The value of the last expression evaluated is returned.

If you don't include an initial value for a variable, it is assigned to **nil.**

Example

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

```
(let ((x 'a) (y 'b)(z 'c))
(format t "x = ~a y = ~a z = ~a" x y z))
```

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

```
x = A y = B z = C
```

The **prog** construct also has the list of local variables as its first argument, which is followed by the body of the **prog,** and any number of s-expressions.

The **prog** function executes the list of s-expressions in sequence and returns nil unless it encounters a function call named **return.** Then the argument of the **return** function is evaluated and returned.

Example

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

```
(prog ((x '(a b c))(y '(1 2 3))(z '(p q 10)))
(format t "x = ~a y = ~a z = ~a" x y z))
```

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

```
x = (A B C) y = (1 2 3) z = (P Q 10)
```