LUA - VARIABLES

A variable is nothing but a name given to a storage area that our programs can manipulate. It can hold different types of values including functions and tables.

The name of a variable can be composed of letters, digits, and the underscore character. It must begin with either a letter or an underscore. Upper and lowercase letters are distinct because Lua is case-sensitive. There are eight basic types of values in Lua −

In Lua, though we don't have variable data types, we have three types based on the scope of the variable.

- **Global variables** − All variables are considered global unless explicitly declared as a local.
- **Local variables** − When the type is specified as local for a variable then its scope is limited with the functions inside their scope.
- **Table fields** − This is a special type of variable that can hold anything except nil including functions.

Variable Definition in Lua

A variable definition means to tell the interpreter where and how much to create the storage for the variable. A variable definition have an optional type and contains a list of one or more variables of that type as follows −

```
type variable_list;
```

Here, **type** is optionally local or type specified making it global, and **variable_list** may consist of one or more identifier names separated by commas. Some valid declarations are shown here −

```
local    i, j
local    i
local    a,c
```

The line **local i, j** both declares and defines the variables i and j; which instructs the interpreter to create variables named i, j and limits the scope to be local.

Variables can be initialized *assignedaninitialvalue* in their declaration. The initializer consists of an equal sign followed by a constant expression as follows −

```
type variable_list = value_list;
```

Some examples are −

```
local d , f = 5 ,10 --declaration of d and f as local variables.
d , f = 5, 10;      --declaration of d and f as global variables.
d, f = 10           --[[declaration of d and f as global variables. Here value of f is
nil --]]
```

For definition without an initializer − variables with static storage duration are implicitly initialized with nil.

Variable Declaration in Lua

As you can see in the above examples, assignments for multiples variables follows a variable_list and value_list format. In the above example **local d, f = 5,10** we have d and f in variable_list and 5 and 10 in values list.

Value assigning in Lua takes place like first variable in the variable_list with first value in the value_list and so on. Hence, the value of d is 5 and the value of f is 10.

Example

Try the following example, where variables have been declared at the top, but they have been defined and initialized inside the main function −

```
-- Variable definition:
local a, b

-- Initialization
a = 10
b = 30

print("value of a:", a)

print("value of b:", b)

-- Swapping of variables
b, a = a, b
print("value of a:", a)

print("value of b:", b)

f = 70.0/3.0
print("value of f", f)
```

When the above code is built and executed, it produces the following result −

```
value of a: 10
value of b: 30
value of a: 30
value of b: 10
value of f 23.333333333333
```

Lvalues and Rvalues in Lua

There are two kinds of expressions in Lua −

- **lvalue** − Expressions that refer to a memory location is called "lvalue" expression. An lvalue may appear as either the left-hand or right-hand side of an assignment.
- **rvalue** − The term rvalue refers to a data value that is stored at some address in memory. An rvalue is an expression that cannot have a value assigned to it, which means an rvalue may appear on the right-hand side, but not on the left-hand side of an assignment.

Variables are lvalues and so may appear on the left-hand side of an assignment. Numeric literals are rvalues and so may not be assigned and cannot appear on the left-hand side. Following is a valid statement −

```
g = 20
```

But following is not a valid statement and would generate a build-time error −

```
10 = 20
```

In Lua programming language, apart from the above types of assignment, it is possible to have multiple lvalues and rvalues in the same single statement. It is shown below.

```
g,l = 20,30
```

In the above statement, 20 is assigned to g and 30 is assigned to l.