# LUA - FUNCTIONS

 

A function is a group of statements that together perform a task. You can divide up your code into separate functions. How you divide up your code among different functions is up to you, but logically the division usually unique, is so each function performs a specific task.

The Lua language provides numerous built-in methods that your program can call. For example, method **print** to print the argument passed as input in console.

A function is known with various names like a method or a sub-routine or a procedure etc.

## Defining a Function

The general form of a method definition in Lua programming language is as follows −

```
optional_function_scope function function_name( argument1, argument2, argument3...,
argumentn)
function_body
return result_params_comma_separated
end
```

A method definition in Lua programming language consists of a *method header* and a *method body*. Here are all the parts of a method −

- **Optional Function Scope** − You can use keyword *local* to limit the scope of the function or ignore the scope section, which will make it a global function.
- **Function Name** − This is the actual name of the function. The function name and the parameter list together constitute the function signature.
- **Arguments** − An argument is like a placeholder. When a function is invoked, you pass a value to the argument. This value is referred to as the actual parameter or argument. The parameter list refers to the type, order, and number of the arguments of a method. Arguments are optional; that is, a method may contain no argument.
- **Function Body** − The method body contains a collection of statements that define what the method does.
- **Return** − In Lua, it is possible to return multiple values by following the return keyword with the comma separated return values.

## Example

Following is the source code for a function called **max**. This function takes two parameters num1 and num2 and returns the maximum between the two −

```
--[[ function returning the max between two numbers --]]
function max(num1, num2)

   if (num1 > num2) then
      result = num1;
   else
      result = num2;
   end

   return result;
end
```

## Function Arguments

If a function is to use arguments, it must declare the variables that accept the values of the arguments. These variables are called the **formal parameters** of the function.

The formal parameters behave like other local variables inside the function and are created upon entry into the function and destroyed upon exit.

## Calling a Function

While creating a Lua function, you give a definition of what the function has to do. To use a method, you will have to call that function to perform the defined task.

When a program calls a function, program control is transferred to the called function. A called function performs the defined task and when its return statement is executed or when its function's end is reached, it returns program control back to the main program.

To call a method, you simply need to pass the required parameters along with the method name and if the method returns a value, then you can store the returned value. For example −

```
function max(num1, num2)

   if (num1 > num2) then
      result = num1;
   else
      result = num2;
   end

   return result;
end

-- calling a function
print("The maximum of the two numbers is ",max(10,4))
print("The maximum of the two numbers is ",max(5,6))
```

When we run the above code, we will get the following output.

```
The maximum of the two numbers is  10
The maximum of the two numbers is  6
```

## Assigning and Passing Functions

In Lua, we can assign the function to variables and also can pass them as parameters of another function. Here is a simple example for assigning and passing a function as parameter in Lua.

```
myprint = function(param)
   print("This is my print function -   ##",param,"##")
end

function add(num1,num2,functionPrint)
   result = num1 + num2
   functionPrint(result)
end

myprint(10)
add(2,5,myprint)
```

When we run the above code, we will get the following output.

```
This is my print function -   ## 10 ##
This is my print function -   ## 7 ##
```

## Function with Variable Argument

It is possible to create functions with variable arguments in Lua using '...' as its parameter. We can get a grasp of this by seeing an example in which the function will return the average and it can take variable arguments.

```
function average(...)
   result = 0
```

```
   local arg={...}
   for i,v in ipairs(arg) do
      result = result + v
   end
   return result/#arg
end

print("The average is",average(10,5,3,4,5,6))
```

When we run the above code, we will get the following output.

```
The average is 5.5
```