MATLAB - FUNCTIONS

A function is a group of statements that together perform a task. In MATLAB, functions are defined in separate files. The name of the file and of the function should be the same.

Functions operate on variables within their own workspace, which is also called the **local workspace**, separate from the workspace you access at the MATLAB command prompt which is called the **base workspace**.

Functions can accept more than one input arguments and may return more than one output arguments.

Syntax of a function statement is −

```
function [out1,out2, ..., outN] = myfun(in1,in2,in3, ..., inN)
```

Example

The following function named *mymax* should be written in a file named *mymax.m*. It takes five numbers as argument and returns the maximum of the numbers.

Create a function file, named mymax.m and type the following code in it −

```
function max = mymax(n1, n2, n3, n4, n5)
%This function calculates the maximum of the
% five numbers given as input
max =  n1;
if(n2 > max)
    max = n2;
end
if(n3 > max)
   max = n3;
end
if(n4 > max)
    max = n4;
end
if(n5 > max)
    max = n5;
end
```

The first line of a function starts with the keyword **function**. It gives the name of the function and order of arguments. In our example, the *mymax* function has five input arguments and one output argument.

The comment lines that come right after the function statement provide the help text. These lines are printed when you type −

```
help mymax
```

MATLAB will execute the above statement and return the following result −

```
This function calculates the maximum of the
 five numbers given as input
```

You can call the function as −

```
mymax(34, 78, 89, 23, 11)
```

MATLAB will execute the above statement and return the following result −

```
ans = 89
```

Anonymous Functions

An anonymous function is like an inline function in traditional programming languages, defined within a single MATLAB statement. It consists of a single MATLAB expression and any number of input and output arguments.

You can define an anonymous function right at the MATLAB command line or within a function or script.

This way you can create simple functions without having to create a file for them.

The syntax for creating an anonymous function from an expression is

```
f = @(arglist)expression
```

Example

In this example, we will write an anonymous function named power, which will take two numbers as input and return first number raised to the power of the second number.

Create a script file and type the following code in it −

```
power = @(x, n) x.^n;
result1 = power(7, 3)
result2 = power(49, 0.5)
result3 = power(10, -10)
result4 = power (4.5, 1.5)
```

When you run the file, it displays −

```
result1 =  343
result2 =  7
result3 =  1.0000e-10
result4 =  9.5459
```

Primary and Sub-Functions

Any function other than an anonymous function must be defined within a file. Each function file contains a required primary function that appears first and any number of optional sub-functions that comes after the primary function and used by it.

Primary functions can be called from outside of the file that defines them, either from command line or from other functions, but sub-functions cannot be called from command line or other functions, outside the function file.

Sub-functions are visible only to the primary function and other sub-functions within the function file that defines them.

Example

Let us write a function named quadratic that would calculate the roots of a quadratic equation. The function would take three inputs, the quadratic co-efficient, the linear co-efficient and the constant term. It would return the roots.

The function file quadratic.m will contain the primary function *quadratic* and the sub-function *disc*, which calculates the discriminant.

Create a function file *quadratic.m* and type the following code in it −

```
function [x1,x2] = quadratic(a,b,c)
%this function returns the roots of
% a quadratic equation.
% It takes 3 input arguments
```

```
% which are the co-efficients of x2, x and the
%constant term
% It returns the roots
d = disc(a,b,c);
x1 = (-b + d) / (2*a);
x2 = (-b - d) / (2*a);
end % end of quadratic

function dis = disc(a,b,c)
%function calculates the discriminant
dis = sqrt(b^2 - 4*a*c);
end % end of sub-function
```

You can call the above function from command prompt as −

```
quadratic(2,4,-4)
```

MATLAB will execute the above statement and return the following result −

```
ans = 0.7321
```

Nested Functions

You can define functions within the body of another function. These are called nested functions. A nested function contains any or all of the components of any other function.

Nested functions are defined within the scope of another function and they share access to the containing function's workspace.

A nested function follows the following syntax −

```
function x = A(p1, p2)
...
B(p2)
   function y = B(p3)
   ...
   end
...
end
```

Example

Let us rewrite the function *quadratic*, from previous example, however, this time the disc function will be a nested function.

Create a function file *quadratic2.m* and type the following code in it −

```
function [x1,x2] = quadratic2(a,b,c)
function disc  % nested function
d = sqrt(b^2 - 4*a*c);
end % end of function disc
disc;
x1 = (-b + d) / (2*a);
x2 = (-b - d) / (2*a);
end % end of function quadratic2
```

You can call the above function from command prompt as −

```
quadratic2(2,4,-4)
```

MATLAB will execute the above statement and return the following result −

```
ans =  0.73205
```

Private Functions

A private function is a primary function that is visible only to a limited group of other functions. If you do not want to expose the implementation of a functions, you can create them as private functions.

Private functions reside in **subfolders** with the special name **private**.

They are visible only to functions in the parent folder.

Example

Let us rewrite the *quadratic* function. This time, however, the *disc* function calculating the discriminant, will be a private function.

Create a subfolder named private in working directory. Store the following function file *disc.m* in it −

```
function dis = disc(a,b,c)
%function calculates the discriminant
dis = sqrt(b^2 - 4*a*c);
end % end of sub-function
```

Create a function quadratic3.m in your working directory and type the following code in it −

```
function [x1,x2] = quadratic3(a,b,c)
%this function returns the roots of
% a quadratic equation.
% It takes 3 input arguments
% which are the co-efficient of x2, x and the
%constant term
% It returns the roots
d = disc(a,b,c);
x1 = (-b + d) / (2*a);
x2 = (-b - d) / (2*a);
end % end of quadratic3
```

You can call the above function from command prompt as −

```
quadratic3(2,4,-4)
```

MATLAB will execute the above statement and return the following result −

```
ans =  0.73205
```

Global Variables

Global variables can be shared by more than one function. For this, you need to declare the variable as global in all the functions.

If you want to access that variable from the base workspace, then declare the variable at the command line.

The global declaration must occur before the variable is actually used in a function. It is a good practice to use capital letters for the names of global variables to distinguish them from other variables.

Example

Let us create a function file named average.m and type the following code in it −

```
function avg = average(nums)
```

```
global TOTAL
avg = sum(nums)/TOTAL;
end
```

Create a script file and type the following code in it −

```
global TOTAL;
TOTAL = 10;
n = [34, 45, 25, 45, 33, 19, 40, 34, 38, 42];
av = average(n)
```

When you run the file, it will display the following result −

```
av =  35.500
```