

In MATLAB[®] a *script* is an m-file which is a list of MATLAB[®] commands. Scripts do not accept or return arguments to the calling program although the variables in a script are available in the command window after the script runs. A MATLAB[®] *function* is similar to a script but it can accept and return arguments and it has local variables that are not known outside the function (even in the command window). Functions can be called from scripts, the command window, or from other functions. Functions can have sub-functions which are not accessible outside the function itself. For functions defined in a separate script file the file name must be the same as the first function.

Beginning with MATLAB 2016b, MATLAB allows functions to exist inside a script file. Such functions are accessible only to the script where the function exists or to other functions defined inside that script.

The syntax for creating a function is:

```
function [out1, out2, ...] = functionName(in1, in2, ...);  
% Comments explaining how the function is used.  
% These comments are optional but they are useful  
% because they show up in the command window when the  
% user types help functionName.  
Standard MATLAB® commands form the body of the function  
...  
end % this is optional
```

This function should be saved in a script file named `functionName.m` and is available to any other script file which calls it.

Functions can be effectively overloaded such that the number input and output arguments can be variable. There are two MATLAB[®] functions which make this possible: `nargin` and `nargout`.

`nargin` is the number of input arguments and `nargout` is the number of output arguments that are in the user's calling statement.

Example

```
function [x, y] = Example1(a, b)  
if(nargin > 2 | nargin < 1)  
    disp('Error');  
    return;  
end  
if(nargin == 1)  
    b = 0;  
end
```

```

if(nargout == 1)
    x = a;
else
    x = a;
    y = b;
end

```

Functions can also receive or return a cell array containing any number of arguments. This is done with the two MATLAB[®] functions called `varargin` and `varargout`. See the MATLAB[®] help files for a description of how these work.

Functions which are defined within an existing script file are not available to other scripts. For example the following script named `FunctionExmp.m` calls a function named `Fun` which is included in the main script. `Fun` is not available outside of `FunctionExmp`.

```

%FunctionExmp.m
%Main program
a = 5;b = 6;
c = Fun(a, b);
disp(c);

function x = Fun(y, z);
x = y + z;
end

```

Sub-Functions

MATLAB[®] functions can have sub-functions which are not available outside the function. Here is an example of a MATLAB[®] function which contains a sub-function.

```

function x = Example2(a, b)
    if(length(a) > 1)
        x = subFun(a, b);
    else
        x = (a + b)/2;
    end
end

function y = subFun(a, b)
y = (sum(a) + sum(b))/2;
end

```

Note that we can run `Example2` from the command line by using its name but `subFun` is not available outside the function.

Global Variables

Functions have access only to those variables in the main program that are in the parameter list. All other main program variables are not defined inside the function. In some cases, especially where constants are used, it is convenient to declare variables as

global. Declaring a variable global in a function gives that function access to the variable in the main program. As a matter of style all global variables should be written in all upper-case letters so that they are not inadvertently changed. Here is an example.

```
>> global C;  
>> C = exp(1);  
>> x = Example3(6);
```

```
x =  
    8.7183
```

```
function x = Example3(a)  
global C; %C comes from the main program
```

```
x = a + C;
```

```
end
```

Inline Functions

Ordinary scripts (m-files) can call functions defined in other files as well as functions defined in the same file (using MATLAB 2016b or later). They can also have *inline functions*. An inline function is defined on one line and creates a short function, usually an equation with variables, which may then be called by the script. Here is an example which creates an inline function to evaluate $y = 3x^3 - 10$

```
%Example4.m  
y = inline('3*x^3 - 10');  
x = 2;  
disp(y(x));
```

We can run Example 4 from the command window like this:

```
>> Example4  
    14
```

In this example our inline function had only a single argument but it is possible to have multiple arguments. Here is an example of an inline function with two arguments.

```
%Example5.m  
y = inline('a*cos(2*pi*f*t)', 'a', 'f', 't');  
a = 3;  
f = 1;  
t = 0.1;  
disp(y(a, f, t));
```

From the main program we can run this example as:

```
>> Temp5  
    2.4271
```