Function files

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.

Types of Functions

There are several types of functions available with MATLAB®, including local functions, nested functions, and anonymous functions, function handle.

Local Functions

Local functions are subroutines that are available within the same file. Local functions are the most common way to break up programmatic tasks.

a- In a function file, which contains only function definitions, local functions can appear in the file in any order after the main function in the file.

b- In a script file, which contains commands and function definitions, local function must be at the end of the file.

How to create the function file

General formula

function output_variable = function_name (input_variable)

steps:

first :to create new script after you open the program, click on the icon Or from

File ----- new ----- script

Second: the programmer writes the required code and corrects mistakes

1. Write the first line: general formula for function file

   function output_variable = function_name (input_variable)

2. Write required code

Third: Save the file when it locks with save and put it in the appropriate path and stored in the function name itself, which was written in the general version.

How to call the function file

- in command window

a - Fix the path in which the function file is in the mat lab program.

b. 1-Enter the value of input variables in the command window first if necessary.

   2- Write down the filename of the function to be called .

c- also: 1. The function is called with a single line containing the function name and the values of the input variables involved.

   2-You can write another variable name for the output variable in function call line.

Notes about the variables used in the function

1. Each of the input variables or output variables can be single or vector.
2. Variables used in the function are usually local variables within the function file.
3. The output variable is written in the code and the general formula, but at the call it can be written with another variable name as shown in the examples.
4. Types of variables:

A - Local variable whose value is declared inside a function such as variable

   a = 2

Example

```
function y = comp (x);
 a = 2; % the variable a has the value 2, but only within this function
 y = a * x;
```

Callback:

a - Command window

```
>> x = 2
>> comp (x) or >> y = comp (x)
```

b - Command window

```
>> comp (2)                    or
>> y = comp (2)
```

B. A local variable (a) takes its value from outside the function.

Example

```
 function y = comp (x, a);
y = a * x; % variable a gets its value outside the function & given to the
function.
```

Callback:

a -

```
 >> a = 2
>> x = 2
>> comp (x, a)      or        >> y = comp (x, a)
```

b-

```
>> comp (2,2) or >> y = comp (2,2)
```

C-Global Variables

1- Declares the required variable within the function code as Global Variables.

2-when the function file callback in command window declare that variable was a Global Variable and enter it's value if not found in function file code.

3-then the function file called

Example: create a new function in a file called falling.m:

1- Create function file

```
function h = falling(t)
global GRAVITY
h = 1/2*GRAVITY*t.^2;
```

2- calling function file

```
>>  global GRAVITY
>> GRAVITY = 2;
  y = falling (0:.1:5)
```

Remark: The declaration with global should only be used when there is no other possibility.

The following formulas to write the function are equal

a-Write the formula

function [W] = myfun (x, y, z); Equivalent **function W = myfun (x, y, z)**

b – Call the function

[M] = myfun (2,4, -3) is equivalent to **M = myfun (2,4, -3)**

Example

A general example of creating and calling a function file

Write a function that performs an ideal gas calculation, where the function is called (named)as follows: IdealGas (P, V, T, R), and returns the value of n, the number of moles.

1- Create a function file

```
function n = IdealGas (P, V, T, R)
n = P * V / R * T;
```

2 - call the function file

a- Enter the values of the input variables first and then call the function

In Command window:

```
>> p = 1
>> V = 22.4
>> T = 273
>> R = 0.082

>> IdealGas (P, V, T, R)   or      n = IdealGas (P, V, T, R)
```

b- Call the function file and values of input variables in one line.

```
>> IdealGas(1,22.4,273,0.082)
```

or

```
>> n= IdealGas(1,22.4,273,0.082)
```

c-enter the values of input variables using input command and the output variable with disp command with function callback, this can be in command window or in script file then executed.

1- Command window

```
>>R = input('Enter the universal gas constant: ');
>>P = input('Enter the gas pressure: ');
>>T = input('Enter the gas temperature :');
>>V = input('Enter the gas volume: ');
>> n = IdealGas(P,V,T,R);
>> disp('Moles of ideal gas =')
>>disp(n)
```

2- in script file IG_main.m

```
R = input('Enter the universal gas constant: ');
P = input('Enter the gas pressure: ');
```

```
T = input('Enter the gas temperature :');
V = input('Enter the gas volume: ');
n = IdealGas(P,V,T,R);
disp('Moles of ideal gas =')
disp(n)
```

callback function file

```
>> IG_main.m        or    >> IG_main
```

Add Local Functions

To add local functions to a script, first, create the script. Go to the Home tab and select New > Script. After you create the script, add code to the script and save it.

Nested functions

Nested functions are completely contained within another function. The primary difference between nested functions and local functions is that nested functions can use variables defined in parent functions without explicitly passing those variables as arguments.

Example

```
function parent
disp('This is the parent function')
nestedfx
  function nestedfx
    disp('This is the nested function')
  end
end
```

Example

```
function [A,M,D]=arith1(a,b)
A=add(a,b);
function c=add(a,b)
c=a+b;
end
M=mul(a,b);
function d=mul(a,b)
d=a*b;
end
D=div(a,b);
function e=div(a,b);
e=a/b;
end
end
```

In Command window:

```
[add,mul,div]=arith1(10,5)

add =

    15


mul =

    50


div =

     2
```

Anonymous Functions

An anonymous function is a function that is not stored in a program file, but is associated with a variable whose data type is function_handle. Anonymous functions can accept inputs and return outputs, they can contain only a single executable statement.

Example

Create a handle to an anonymous function that finds the square of a number:

```
sqr = @(x) x.^2;
```

Variable sqr is a function handle. The **@** operator creates the handle, and the parentheses () immediately after the @ operator include the function input arguments.

Call the anonymous function sqr

```
a = sqr(5)
a =   25
```

Example

Find the integral of the sqr function from 0 to 1 by passing the function handle to the integral function:

```
q = integral(sqr,0,1);
```

To call to the integral function:

```
q = integral(@(x) x.^2,0,1);
```

Anonymous Functions with Multiple Inputs or Outputs

Anonymous functions require that you explicitly specify the input arguments as you would for a standard function, separating multiple inputs with commas.

Example, this function accepts two inputs, x and y:

```
myfunction = @(x,y) (x^2 + y^2 + x*y);

x = 1;

y = 10;

z = myfunction(x,y)

z = 111
```

Multiple Anonymous Functions

The expression in an anonymous function can include another anonymous function. This is useful for passing different parameters to a function that you are evaluating over a range of values.

Example

You can solve the equation

$$g(c) = \int_0^1 (x^2 + cx + 1)dx$$

For varying values of c by combining two anonymous functions:

```
g = @(c) (integral(@(x) (x.^2 + c*x + 1),0,1));
```

Call the function g

```
g(2)

  ans =

         2.3333
```

$$\begin{bmatrix} 5 & 2r & r \\ 3 & 6 & 2r-1 \\ 2 & r-1 & 3r \end{bmatrix} \bullet \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 2 \\ 3 \\ 5 \end{Bmatrix}$$

$$\begin{bmatrix} 5 & 2r & r \\ 3 & 6 & 2r-1 \\ 2 & r-1 & 3r \end{bmatrix} \bullet \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 2 \\ 3 \\ \ \end{Bmatrix}$$