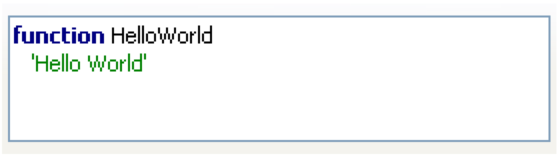
Matlab/Freemat/Octave/Scilab: Functions: M-files

An engineering product may be composed of lots of components, each with their own purpose. It is considered good practice that a computer program – or software – should be engineered in a similar way. That is that when there is a discernible service that is required in a program – and often required many times – then that function is separated from the main program. Thus in *procedural programming* (or *modular programming*[1]) , a program consists of a set of functions (or modules, procedures or subroutines – but *functions* in Matlab/Freemat/Octave) that are called from the main program or from other functions.

In Matlab/Freemat/Octave a function is placed in a separate file, called an M-file. M-files can be created through the Tools-Editor menu.

For example if we created the following M-file HelloWorld.m

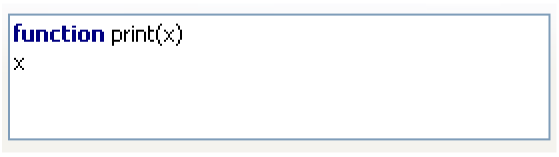


```
function HelloWorld
  'Hello World'
```

Note that the function name is the same as the file name and the file extension is .m . The function may then be called, giving the following result.

```
--> HelloWorld
ans =
Hello World
```

A function may have input parameters. For example presume that we have a function in file print.m with a parameter x.



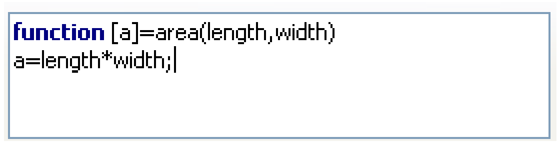
```
function print(x)
x
```

The function may then be called, giving the following result.

```
--> print('Hello World')
ans =
Hello World
```

[1] Modular Programming

A function may also return values. For example consider the following function that computes the area of a rectangle. Note that the semicolon prevents the printing of 'a' from within the function. The function must be placed in a file area.m .

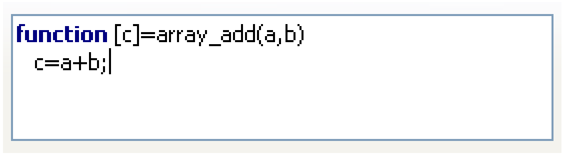


```
function [a]=area(length,width)
a=length*width;
```

The function may then be called, giving the following result.

```
--> area(2,3)
ans =
6
```

Functions may have arrays as input and/or output. For example the following function returns the sum in c of two arrays, a and b. The function must be placed in a file array_add.m .



```
function [c]=array_add(a,b)
  c=a+b;
```

The function may then be called, giving the following result.

```
--> a=[1 2 3]
a =
1 2 3
--> b=[1 -1 2]
b =
1 -1  2
--> c=array_add(a,b)
c =
2 1 5
```