Review of Lecture 8

- **for loop**

```
for iterate_var = range
    <action>
end
```

```
for i = 1:3
    disp('Hello world');
end
```

- **while loop**

```
while condition
    <action>
end
```

```
itr = 1
while itr < 5
    disp('Hello world’);
itr = itr + 1;
end
```

Info 3

Introduction to MATLAB

M. Bouzenita
2nd year Engineer - University of Jijel

Lecture 9

Function file in MATLAB

1. Introduction

The functions are an essential part of MATLAB programming, which can be defined as **a block containing a set of instructions for generating some computations**.

In MATLAB, to execute a function it is necessary to store it in a separate .m file where **the file name should be the same name of the function**.

1. Main parts of a function

General syntax of a function in MATLAB

```
function output = functionName(input)
      instructions
end
```

Functions can support several input arguments and may generate more than one output argument.

```
function [output1, output2, ...] = functionName(input1, input2, ...)
      instructions
end
```

1. Main parts of a function

General syntax of a function in MATLAB

Function declaration: The declaration of a function is presented by two parts separated by the equal sign.
The left part began with the keyword **function** followed by the output arguments.
The right part starts with the **function name** followed by the **input arguments**.

```
function output = functionName(input)
        instructions
end
```

function body: It contains the set of computation instructions including different MATLAB commands.

End of the function: Any function ends with the keyword **end**.

1. Main parts of a function

The detailed form of a function can be given as follows:

```
function [output1, output2, ...] = functionName(input1, input2, ...)
   % Help text: Comments introduced to denote a description of the
   % function instructions and to be displayed with help.
   ...
   Output1 = ...
   Output2 = ...
   ...
end
```

2. Local variables and Global variables

Local variables

Any variables declared **inside the function** are called local variables which are not accessible **outside** it, where the function has its own workspace.

Global variables

By default, the variables defined in the base workspace are not available inside the function.
To make any variable accessible inside any function we declare it using the keyword `'global'` in the command window and in the function.
If a variable is declared as `global` in several functions and in the base workspace, then any assignment to that variable, inside any function, is available to all functions declaring it global.

2. Local variables

Local variables

Any variables declared **inside t**
which are not accessible **outsid**
workspace.

Global variables

By default, the variables defin
available inside the function.
To make any variable accessible i
the keyword `'global'` in the c
If a variable is declared as `globa`
workspace, then any assignment
available to all functions declarin

```
function  myF

global B
A = 10
B = B + A ;

end
```

```
>> A = 2;

>> B = 5;

>> global B

>> myF
```

2. Local variables and Global variables

Some functions are useful when using global variables:

`isglobal(var):` introduced to check if the variable is global or not, which returns 1 if `'var'` is global, and 0 otherwise.

`who global:` gives a list of global variables

`clear global` : makes all variables nonglobal

`clear VAR` : makes VAR nonglobal if it is declared as global.

3. First example

```
function area = rectangle_area(length, width)
      % This function computes the area of a rectangle
  if length < 0 || width < 0
       error('Both length and width must be positive!');
  else
        area = length * width;       % Calculate the area
  end
end
```

To use this function, it is necessary to save it in an .m file that takes the same name as the function name (`rectangle_area.m`).

4. Anonymous function

The anonymous function is the simple way to create a simple function **without saving it in a script file**. In general, it is used in case of a simple calculation in one line code.

The syntax of the anonymous function is given by:

```
fvar  = @(arguments) expression
```

where
`fvar` is the function handle variable name.
`arguments` represents the arguments passed to the function and
`expression` is the corresponding code of the function.

4. Anonymous function

The anonymous function is the simple way to creat
without saving it in a script file. In general, it i
simple calculation in one line code.

The syntax of the anonymous function is given by:

```
fvar  = @(arguments) expression
```

where
`fvar` is the function handle variable name.
`arguments` represents the arguments passed to the
`expression` is the corresponding code of the func

```
>> F = @(x) 2*x^2 +3*x -2;

>> F(3)

ans =

    25
```

4. Anonymous function

The **function handle** can be also used for **referring** to other defined user or standard MATLAB functions.

As an example, we can use a function handle to built-in the key MATLAB function **`sqrt`**.

```
>> Fs = @sqrt

Fs =

function_handle with value:

    @sqrt


>> Fs(4)

ans =

     2
```

Practice