# **MATLAB - if... end Statement**

An **if ... end** statement consists of an **if** statement and a boolean expression followed by one or more statements. It is delimited by the **end** statement.

## Syntax

The syntax of an if statement in MATLAB is −

if <expression>

% statement(s) will execute if the boolean expression is true

<statements>

end

If the expression evaluates to true, then the block of code inside the if statement will be executed. If the expression evaluates to false, then the first set of code after the end statement will be executed.

## Example

Create a script file and type the following code −

a = 10;

% check the condition using if statement

if a < 20

% if condition is true then print the following

fprintf('a is less than 20\n' );

end

fprintf('value of a is : %d\n', a);

When you run the file, it displays the following result −

a is less than 20

value of a is : 10

# **MATLAB - if...else...end Statement**

An if statement can be followed by an optional else statement, which executes when the expression is false.

## Syntax

The syntax of an if...else statement in MATLAB is −

if <expression>

% statement(s) will execute if the boolean expression is true

<statement(s)>

else

<statement(s)>

% statement(s) will execute if the boolean expression is false

end

If the boolean expression evaluates to true, then the if block of code will be executed, otherwise else block of code will be executed.

## Example

Create a script file and type the following code −

a = 100;

% check the boolean condition

if a < 20

% if condition is true then print the following

fprintf('a is less than 20\n' );

else

% if condition is false then print the following

fprintf('a is not less than 20\n' );

end

fprintf('value of a is : %d\n', a);

When the above code is compiled and executed, it produces the following result −

a is not less than 20

value of a is : 100

# **MATLAB - if...elseif...elseif...else...end Statements**

n **if** statement can be followed by one (or more) optional **elseif...** and an **else** statement, which is very useful to test various conditions.

When using if... elseif...else statements, there are few points to keep in mind −

* An if can have zero or one else's and it must come after any elseif's.
* An if can have zero to many elseif's and they must come before the else.
* Once an else if succeeds, none of the remaining elseif's or else's will be tested.

## Syntax

if <expression 1>

% Executes when the expression 1 is true

<statement(s)>

elseif <expression 2>

% Executes when the boolean expression 2 is true

<statement(s)>

Elseif <expression 3>

% Executes when the boolean expression 3 is true

<statement(s)>

else

% executes when the none of the above condition is true

<statement(s)>

end

## Example

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

a = 100;

%check the boolean condition

if a == 10

% if condition is true then print the following

fprintf('Value of a is 10\n' );

elseif( a == 20 )

% if else if condition is true

fprintf('Value of a is 20\n' );

elseif a == 30

% if else if condition is true

fprintf('Value of a is 30\n' );

else

% if none of the conditions is true '

fprintf('None of the values are matching\n');

fprintf('Exact value of a is: %d\n', a );

end

When the above code is compiled and executed, it produces the following result −

None of the values are matching

Exact value of a is: 100

# **MATLAB - The Nested if Statements**

It is always legal in MATLAB to nest if-else statements which means you can use one if or elseif statement inside another if or elseif statement(s).

## Syntax

The syntax for a nested if statement is as follows −

if <expression 1>

% Executes when the boolean expression 1 is true

if <expression 2>

% Executes when the boolean expression 2 is true

end

end

You can nest elseif...else in the similar way as you have nested if statement.

## Example

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

a = 100;

b = 200;

% check the boolean condition

if( a == 100 )

% if condition is true then check the following

if( b == 200 )

% if condition is true then print the following

fprintf('Value of a is 100 and b is 200\n' );

end

end

fprintf('Exact value of a is : %d\n', a );

fprintf('Exact value of b is : %d\n', b );

When you run the file, it displays −

Value of a is 100 and b is 200

Exact value of a is : 100

Exact value of b is : 200

# **MATLAB - The switch Statement**

A switch block conditionally executes one set of statements from several choices. Each choice is covered by a case statement.

An evaluated switch\_expression is a scalar or string.

An evaluated case\_expression is a scalar, a string or a cell array of scalars or strings.

The switch block tests each case until one of the cases is true. A case is true when −

* For numbers, **eq(case\_expression,switch\_expression)**.
* For strings, **strcmp(case\_expression,switch\_expression)**.
* For objects that support the **eq(case\_expression,switch\_expression)**.
* For a cell array case\_expression, at least one of the elements of the cell array matches switch\_expression, as defined above for numbers, strings and objects.

When a case is true, MATLAB executes the corresponding statements and then exits the switch block.

The **otherwise** block is optional and executes only when no case is true.

## Syntax

The syntax of switch statement in MATLAB is −

switch <switch\_expression>

case <case\_expression>

<statements>

case <case\_expression>

<statements>

...

...

otherwise

<statements>

end

## Example

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

grade = 'B';

switch(grade)

case 'A'

fprintf('Excellent!\n' );

case 'B'

fprintf('Well done\n' );

case 'C'

fprintf('Well done\n' );

case 'D'

fprintf('You passed\n' );

case 'F'

fprintf('Better try again\n' );

otherwise

fprintf('Invalid grade\n' );

end

When you run the file, it displays −

Well done

# **MATLAB - The while Loop**

he while loop repeatedly executes statements while condition is true.

## Syntax

The syntax of a while loop in MATLAB is −

while <expression>

<statements>

end

The while loop repeatedly executes program statement(s) as long as the expression remains true.

An expression is true when the result is nonempty and contains all nonzero elements (logical or real numeric). Otherwise, the expression is false.

## Example

Create a script file and type the following code −

a = 10;

% while loop execution

while( a < 20 )

fprintf('value of a: %d\n', a);

a = a + 1;

end

When you run the file, it displays the following result −

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

value of a: 16

value of a: 17

value of a: 18

value of a: 19

# **MATLAB - The for Loop**

A **for loop** is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

## Syntax

The syntax of a **for loop** in MATLAB is −

for index = values

<program statements>

...

end

*values* has one of the following forms −

|  |  |
| --- | --- |
| **Sr.No.** | **Format & Description** |
| 1 | *initval:endval*  increments the index variable from *initval* to *endval* by 1, and repeats execution of *program statements*until *index* is greater than *endval*. |
| 2 | *initval:step:endval*  increments *index* by the value step on each iteration, or decrements when step is negative. |
| 3 | *valArray*  creates a column vector *index* from subsequent columns of array *valArray* on each iteration. For example, on the first iteration, index = valArray(:,1). The loop executes for a maximum of n times, where n is the number of columns of *valArray*, given by numel(valArray, 1, :). The input *valArray* can be of any MATLAB data type, including a string, cell array, or struct. |

## Example 1

Create a script file and type the following code −

for a = 10:20

fprintf('value of a: %d\n', a);

end

When you run the file, it displays the following result −

value of a: 10

value of a: 11

value of a: 12

value of a: 13

value of a: 14

value of a: 15

value of a: 16

value of a: 17

value of a: 18

value of a: 19

value of a: 20

## Example 2

Create a script file and type the following code −

for a = 1.0: -0.1: 0.0

disp(a)

end

When you run the file, it displays the following result −

1

0.90000

0.80000

0.70000

0.60000

0.50000

0.40000

0.30000

0.20000

0.10000

0

## Example 3

Create a script file and type the following code −

for a = [24,18,17,23,28]

disp(a)

end

When you run the file, it displays the following result −

24

18

17

23

28

# **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

### 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