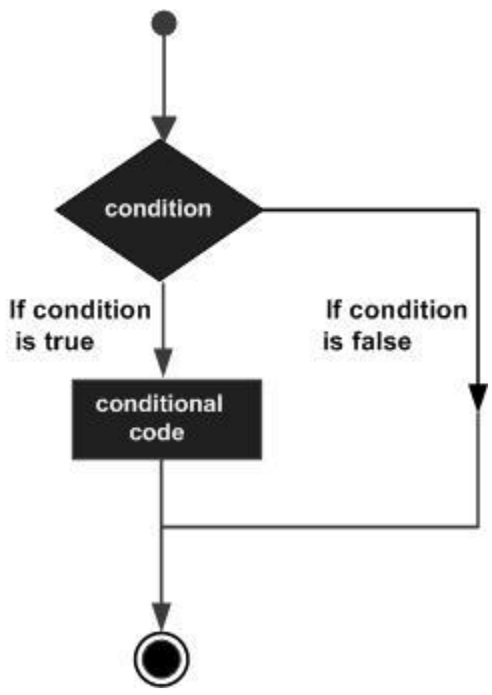
DECISION MAKING AND LOOPING

MATLAB - Decision Making

Decision making structures require that the programmer should specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

Following is the general form of a typical decision making structure found in most of the programming languages –

Flowchart :



MATLAB provides following types of decision making statements. Click the following links to check their detail −

Sr.No.	Statement & Description

1	if ... end statement An **if ... end statement** consists of a boolean expression followed by one or more statements.
2	if...else...end statement An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is false.
3	If... elseif...elseif...else...end statements An **if** statement can be followed by one (or more) optional **elseif...** and an **else** statement, which is very useful to test various conditions.
4	nested if statements You can use one **if** or **elseif** statement inside another **if** or **elseif** statement(s).
5	switch statement A **switch** statement allows a variable to be tested for equality against a list of values.
6	nested switch statements You can use one **switch** statement inside another **switch** statement(s).

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

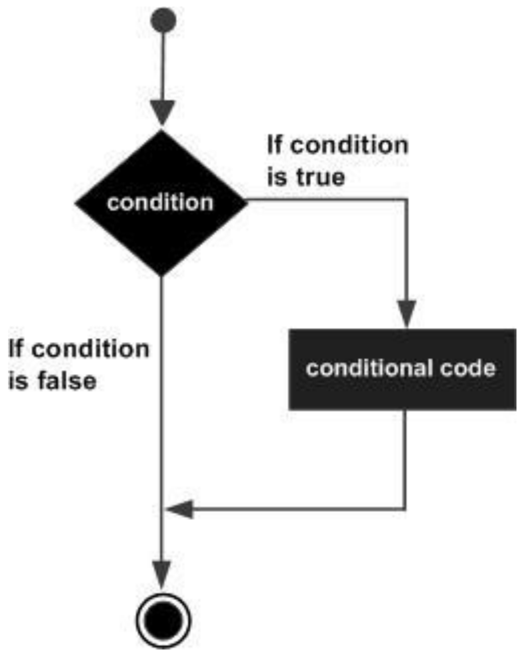
The syntax of an if statement in MATLAB is −

```
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.

Flow Diagram



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.

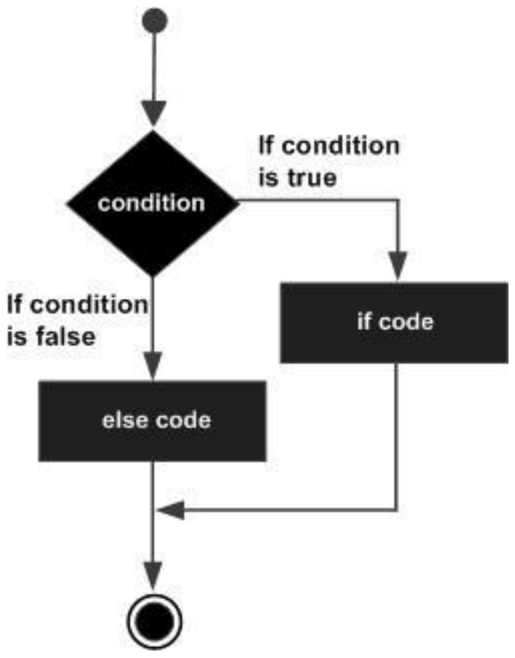
Syntax

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.

Flow Diagram



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

An **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

Previous Page

Next Page

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 Nested switch Statements

Previous Page

Next Page

It is possible to have a switch as part of the statement sequence of an outer switch. Even if the case constants of the inner and outer switch contain common values, no conflicts will arise.

Syntax

The syntax for a nested switch statement is as follows −

```
switch(ch1)
   case 'A'
      fprintf('This A is part of outer switch');
      switch(ch2)
         case 'A'
         fprintf('This A is part of inner switch' );

         case 'B'
         fprintf('This B is part of inner switch' );
      end
   case 'B'
      fprintf('This B is part of outer switch' );
end
```

Example

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

```
a = 100;
b = 200;
switch(a)
   case 100
      fprintf('This is part of outer switch %d\n', a );
      switch(b)
         case 200
            fprintf('This is part of inner switch %d\n', a );
      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 −

```
This is part of outer switch 100
This is part of inner switch 100
Exact value of a is : 100
Exact value of b is : 200
```

MATLAB - Loop Types

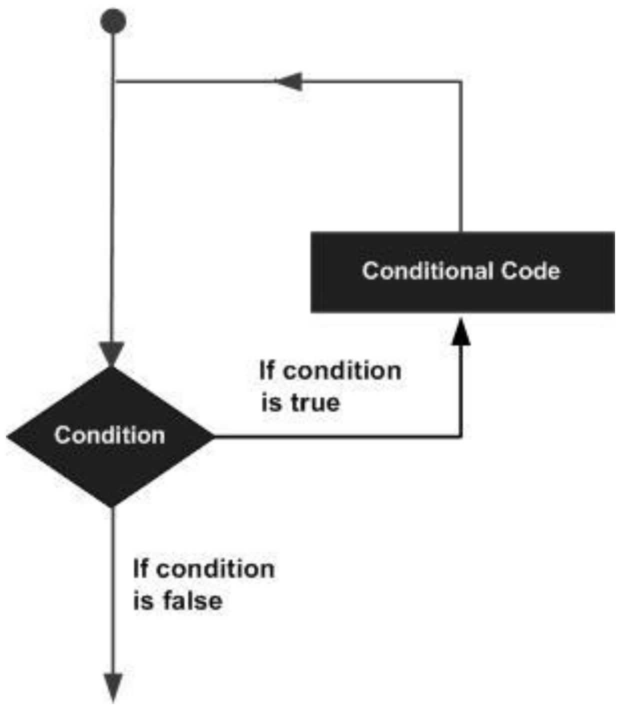
Previous Page

Next Page

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially. The first statement in a function is executed first, followed by the second, and so on.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages −



MATLAB provides following types of loops to handle looping requirements. Click the following links to check their detail −

Sr.No.	Loop Type & Description
1	while loop Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
2	for loop Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
3	nested loops You can use one or more loops inside any another loop.

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

MATLAB supports the following control statements. Click the following links to check their detail.

Sr.No.	Control Statement & Description
1	break statement Terminates the **loop** statement and transfers execution to the statement immediately following the loop.
2	continue statement Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

MATLAB - The while Loop

Previous Page

Next Page

The 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
```

Previous Page Print Page

MATLAB - The for Loop

Previous Page

Next Page

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 - The Nested Loops

Previous Page
Next Page

MATLAB allows to use one loop inside another loop. Following section shows few examples to illustrate the concept.

Syntax

The syntax for a nested for loop statement in MATLAB is as follows −

```
for m = 1:j
   for n = 1:k
      <statements>;
   end
```

```
end
```

The syntax for a nested while loop statement in MATLAB is as follows −

```
while <expression1>
   while <expression2>
      <statements>
   end
end
```

Example

Let us use a nested for loop to display all the prime numbers from 1 to 100. Create a script file and type the following code −

```
for i = 2:100
   for j = 2:100
      if(~mod(i,j))
         break; % if factor found, not prime
      end
   end
   if(j > (i/j))
      fprintf('%d is prime\n', i);
   end
end
```

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

```
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime
53 is prime
59 is prime
61 is prime
67 is prime
71 is prime
73 is prime
```

```
79 is prime
83 is prime
89 is prime
97 is prime
```

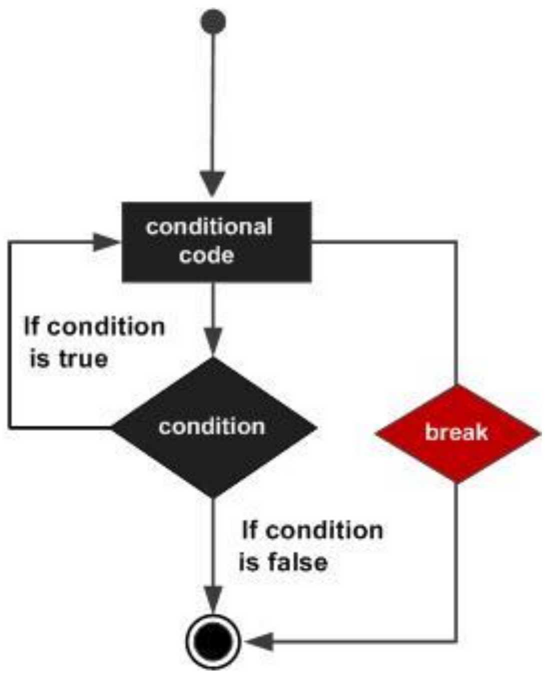
MATLAB - The break Statement

 Previous Page
Next Page

The break statement terminates execution of **for** or **while** loop. Statements in the loop that appear after the break statement are not executed.

In nested loops, break exits only from the loop in which it occurs. Control passes to the statement following the end of that loop.

Flow Diagram



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;
      if( a > 15)
         % terminate the loop using break statement
         break;
      end
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
```

MATLAB - continue Statement

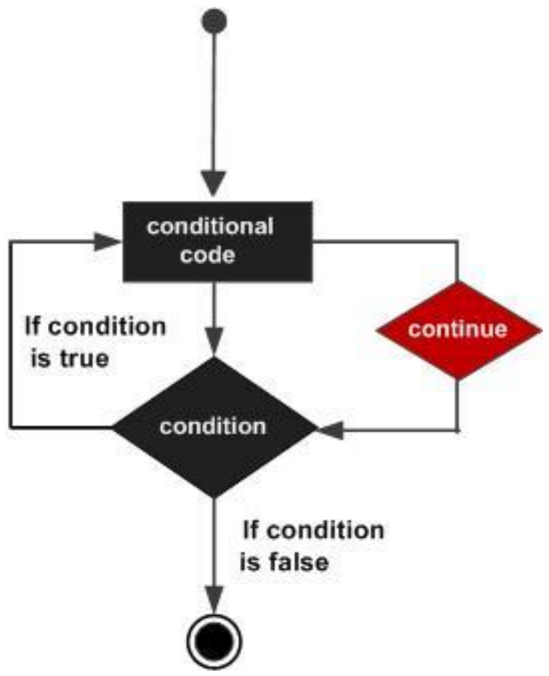
Previous Page

Next Page

The continue statement is used for passing control to next iteration of for or while loop.

The continue statement in MATLAB works somewhat like the break statement. Instead of forcing termination, however, 'continue' forces the next iteration of the loop to take place, skipping any code in between.

Flow Diagram



Example

Create a script file and type the following code −

```
a = 9;
%while loop execution
while a < 20
   a = a + 1;
   if a == 15
      % skip the iteration
      continue;
   end
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: 16
value of a: 17
value of a: 18
value of a: 19
value of a: 20
```