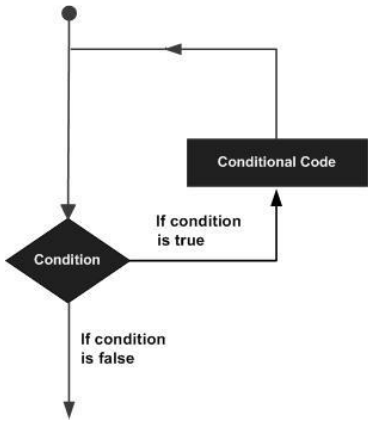
Matlab – Loop types

There may be a situation when you need to execute a block of code several 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. The drawing shows the general form of a loop statement for most programming languages.

Matlab provides various types of loops to handle looping requirements including: while loops, for loops, and nested loops. If you are trying to declare or write your own loops, you need to make sure that the loops are written as scripts and not directly in the Command Window.

To start a new script, locate the button in the upper left corner of the window labeled ***New Script***.

While Loop

The while loop repeatedly executes statements while a specified condition is true.

The syntax of a while loop in MATLAB is as following:

```
while <expression>
   <statements>
end
```

The while loop repeatedly executes a 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

```
a = 10;
% while loop execution
while( a < 20 )
  fprintf('value of a: %d\n', a);
  a = a + 1;
end
```

When the code above is executed, the result will be:

```
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
```

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.

The syntax of a for loop in MATLAB is as following:

```
for index = values
   <program statements>
            ...
end
```

values can be one of the following forms:

initval:endval --- increments the index variable from initval to endval by 1, and repeats execution of program statements until index is greater than endval.

Initval:step:endval --- increments index by the value step on each iteration, or decrements when step is negative.

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

```
for a = 10:20
   fprintf('value of a: %d\n', a);
end
```

When the code above is executed, the result will be:

```
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
```

```
for a = 1.0: -0.1: 0.0
   disp(a)
end
```

When the code above is executed, the result is:

```
1
9/10
4/5
7/10
3/5
1/2
2/5
3/10
1/5
1/10
0
```

```
for a = [24,18,17,23,28]
   disp(a)
end
```

When the code above is executed, the result will be:

```
24
```

```
18

17

23

28
```

The Nested Loops

Matlab also allows to use one loop inside another loop.

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

We can use a nested for loop to display all the prime numbers from 1 to 100.

```
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 the code above is executed, the result is:

```
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
```

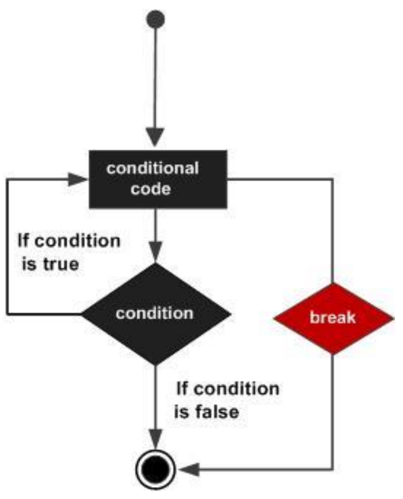
Loop Control Statements

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. The scope defines where the variables can be valid in Matlab, typically a scope within a loop body is from the beginning of conditional code to the end of conditional code. It tells Matlab what to do when the conditional code fails in the loop. Matlab supports both break statement and continue statement.

Break statement:

The break statement terminates execution of for or while loops. 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.



Example

```
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
```

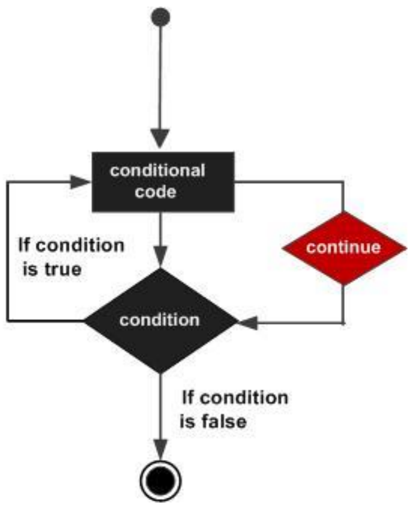
When the code above is executed, the result is:

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
```

Continue Statement

The continue statement is used for passing control to thenext iteration of a 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.



Example

```
a = 10;
%while loop execution
while a < 20
   if a == 15
      % skip the iteration
      a = a + 1;
      continue;
   end
fprintf('value of a: %d\n', a);
a = a + 1;
end
```

When the code above got executed, the result is:

```
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
```