# Lecture 4: Conditional Statements & Loops in Matlab

## if, elseif, else

- Execute statements if condition is true
- An **if** statement can be followed by one (or more) optional **elseif**... and an **else** statement, which is very useful to test various conditions.

### Syntax

```
if expression
    statements
elseif expression
    statements
else
    statements
end
```

### Example

```
% Generate a random number
a = randi(100, 1);
% If it is even, divide by 2
if rem(a, 2) == 0
    disp('a is even')
    b = a/2;
end
```

if statements can include alternate choices, using the optional keywords **elseif** or **else**.

**Example**

```
a = randi(100, 1);

if a < 30
   disp('small')
elseif a < 80
   disp('medium')
else
   disp('large')
end
```

**Example**

```
a = 10
min = 2
max = 20
if ( a > = min ) & & ( a < = max )
   disp ( ' a is within range ' )
elseif ( a < = min )
   disp ( ' a is less than minimum ' )
else
    disp ( ' a is more than maximum value ' )
end
```

**Notes**

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

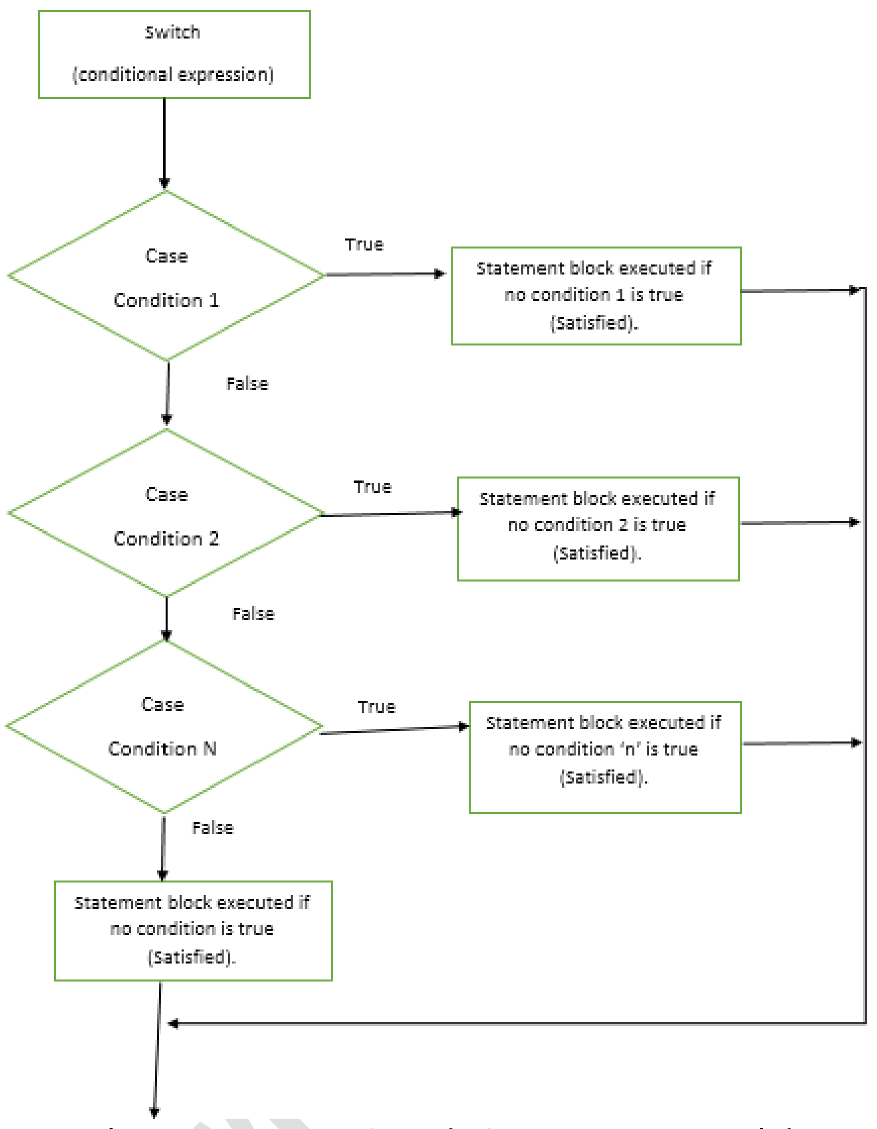
- An if can have zero or one **else's** and it must come after any **elseif's**.
- An if can have zero to many **elseif's** and they must come **before** the else.
- Once an else if succeeds, none of the remaining elseif's or else's will be tested.

## switch statement

we can create multiple alternative selection logic using the **if-else**, **If** statements. However, that method makes the code long and also hard to read and debug, we have another good way or let’s say a better way to make those selections. A **switch** statement helps us choose one among a number of options using code that is easier to read, as said and less time-consuming to typing and editing. The results from both these methods are essentially the same, but the method of implementation varies.

### Syntax

```
switch switch_condition
case case_condition
statements_if_true
case case_condition
statements_if_true
...
Otherwise_condition
Statements_if_true
end
```


Switch
(conditional expression)
Case
Condition 1
True
Statement block executed if no condition 1 is true (Satisfied).
False
Case
Condition 2
True
Statement block executed if no condition 2 is true (Satisfied).
False
Case
Condition N
True
Statement block executed if no condition 'n' is true (Satisfied).
False
Statement block executed if no condition is true (Satisfied).

Flow Diagram in Switch Statement in Matlab

**Example**

The core idea is to pass through a switch statement and print message based on some condition. We create a basic logic of matching the number and providing an output based on the number.

```
N = input('Enter a number of your choice: ');
switch N
case -2
disp('negative one selected')
case 0
disp('zero selected')
case 2
disp('positive one selected')
otherwise
disp('Some other value')
end
```

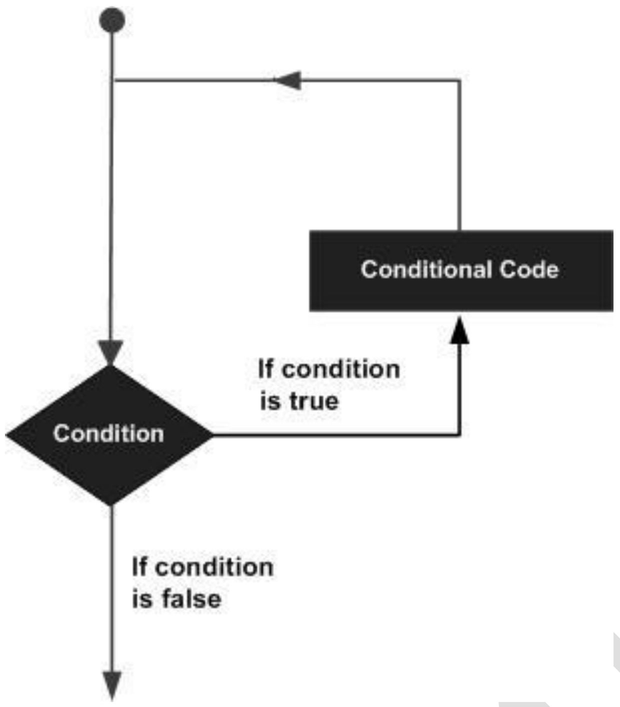
**Example**

```
[dayNum, dayString] = weekday(date, 'long', 'en_US');

switch dayString
   case 'Monday'
      disp('Start of the work week')
   case 'Tuesday'
      disp('Day 2')
   case 'Wednesday'
      disp('Day 3')
   case 'Thursday'
      disp('Day 4')
   case 'Friday'
      disp('Last day of the work week')
   otherwise
      disp('Weekend!')
end
```

## Loops in Matlab

| Sr.No. | Loop Type & Description |
|---|---|
| 1 | **while loop**<br>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**<br>Executes a sequence of statements multiple times and abbreviates the cod that manages the loop variable. |
| 3 | **nested loops**<br>You can use one or more loops inside any another loop. |



## While Loop

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

**Syntax**

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

**Example**

```
% program to find the number ten from a series of random numbers
% using while loop
k = 1;
while k
   if randi(50,1) == 10
      disp(['The random number equivalent to 10 found at ',num2str(k),' step'])
      break
   end
   k = k + 1;
end
```

**Output:**

```
The random number equivalent to 10 found at 36 step.
```

## for loop

A for loop is used to repeat a statement or a group of statements for a fixed number of times.

**Syntax**

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

**Example:**

```
% program to print multiples of first prime number between 1000 and 2000
% using for loop
pr = 0;
for k = 1000:2000
    if isprime(k)
        pr = k;
        disp(['The first prime number is : ',num2str(pr)])
        for m = pr:pr:pr*10
            disp(m)
            end

            break
          end
       end
```

**Output:**

```
The first prime number is: 1009
        1009
        2018
        3027
        4036
        5045
        6054
        7063
        8072
        9081
       10090
```

**Example:**

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

**Output**

```
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**

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

**Output**

```
1
0.9000
0.8000
0.7000
0.6000
0.5000
0.4000
0.3000
0.2000
0.1000
0
```

**Example**

```
s = 10;
H = zeros(s);

for c = 1:s
   for r = 1:s
      H(r,c) = 1/(r+c-1);
   end
end
```

## MATLAB Nested Loop

MATLAB also allows using one loop inside another loops.

**Syntax**

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

    end
end
```

**Syntax**

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

**Example:**
Use the 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
```

**Output:**

```
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
```

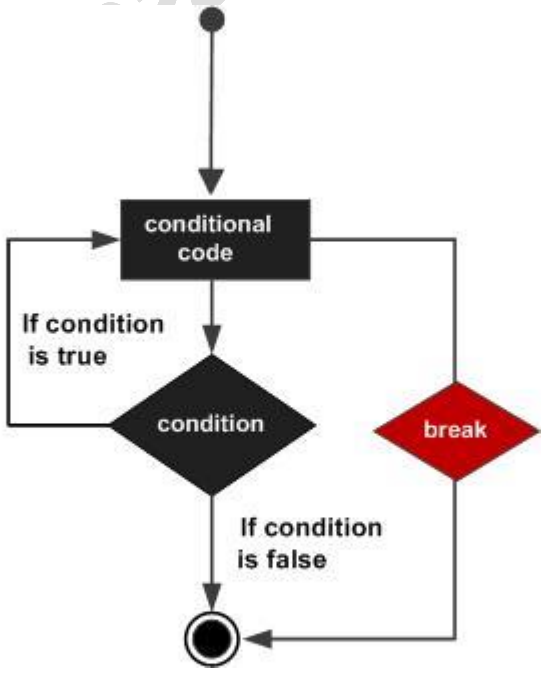
## Loop Control Statements

### MATLAB break

- The **break** statement terminates the execution of a **for** loop or **while** loop.
- When a break statement is encountered, execution proceeds with the next statement outside of the loop. In nested loops, break exists from the **innermost** loop only.

**Syntax**

```
break
```



Flow diagram of Break Statement.

**Example**

```
% program to break the flow at a specified point
a = randi(100,6,6)
k = 1;
while k
    disp('program running smoothly')
    if a(k) == 27
        disp('program encounters the number 27, which is not useful for the current program;')
        disp(['at index no.:',num2str(k)])
disp('so loop terminates now.....bye bye')
            break
    end
      k = k+1;
    end
```

**Output:**

```
a = 6×6
   82    17    70    54    54    10
   27    18    70    66    33    27
   60    43    64    41    11    16
    3    10     4    82    62    29
   43    60     7    72    78    45
   32    48    32    97    43    53
program running smoothly
program running smoothly
program encounters the number 27, which is not useful for the current program;
at index no.:2
so loop terminates now.....bye bye
```

**Example**

```
a = 10;
% while loop execution
while (a < 20)
   disp(['value of a:', num2str(a)]);
   a = a + 1;
      if( a > 15)
         % terminate the loop using break statement
         break;
      end
end
```

**Output**

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

**Example**

```
% program to terminate the execution on finding negative input
a = randn(4)
k = 1;
 while k < numel(a)
  if a(k) < 0
      break
  end
  k = k + 1;
end
 disp(['negative number :', num2str(a(k)), ',found at index: ', num2str(k)
 ,',hence the program terminated'])
```

**Output**

```
a = 4×4
   0.2398   -1.6118    0.8617    0.5812
  -0.6904   -0.0245    0.0012   -2.1924
  -0.6516   -1.9488   -0.0708   -2.3193
   1.1921    1.0205   -2.4863    0.0799
negative number:-0.69036,found at index: 2,hence the program terminated
```

## MATLAB continue

The **continue** statement works within a **for** or **while** loop and passes control to the next iteration of the loop.

**Syntax:**

```
Continue
```

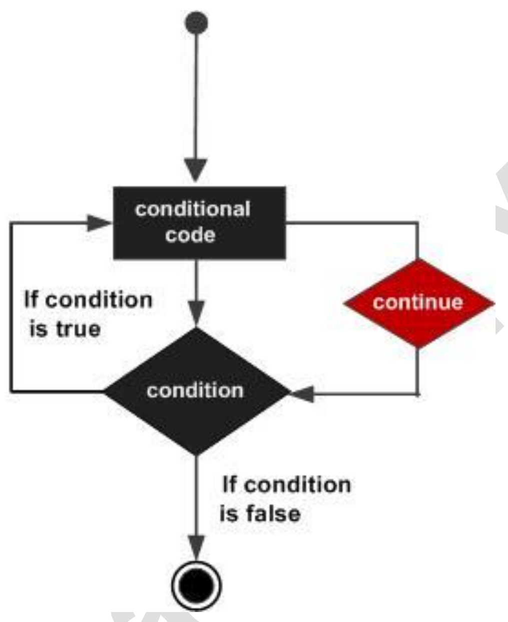
**Following are the points while using a continue statement in MATLAB**

- Continue statement passes the control of the execution to the next iteration of a for or while loop.
- All remaining statements following the continue statement do not execute for the current iteration.
- The continue statement applies only to the body of the loop where it is called, hence in nested loops, it affects the execution of the loop in which it occurs.

- The continue statement only works inside a **for** or **while** loop, and it can't be used inside a function. But if a function is having a for or while loop, there we can use **continue** inside the loop.

**Note**

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 = 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
```

**Example**

```
% program to print all numbers divisible by 3 and skip remaining
a = (1:4:50); % creates row vector from 1 to 50 with a step of 4
for k = 1:numel(a)
    if rem(a(k),3)
        continue
    end
    disp(a(k))
end
```

**Output:**

```
  9
 21
 33
 45
```

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