- Logical arrays:

Relational operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
~=	not equal to

logical operator	Description
&	AND
|	OR
~	NOT

في حالة مقارنة بين arrays والناتج للعناصر لتغير لديها
True (1) elements, and False (0)

```
EDU>> x = -3:3 % Create data
x =
   -3  -2  -1   0   1   2   3
EDU>> abs(x) > 1 % |x| = √x² positive values for x only
ans =
    1   1   0   0   0   1   1
EDU>> y = x(abs(x)>1) % y = x(1 2 6 7)
```

كأننا نقول أوجد y = x(1 2 6 7)

```
y =
   -3  -2   2   3
EDU>> y = x([1 1 0 0 0 1 1])
??? ERROR
```

لايجوز استخدام الترتيب للموقع في حالة سالب أو صفر ??? ERROR

أي لا يجوز استخدام numeric array مع متجه منطقي

```
EDU>> y = x (logical ([1 1 0 0 0 1 1]))
      y = -3   -2   2   3
```

logical arrays works on matrices, as well as vectors, also:

```
EDU>> B = [5 -3; 2 -4]  %  B = 5  -3
                               2  -4

EDU>>  x = abs(B) > 2
       x = 1   1
           0   1

EDU>>  y = b(x)
       y = 5
          -3
          -4
```

نلاحظ ان النتيجة تحولت الى عمود لانه لا توجد طريقة لتعيين مصفوفة لها ثلاثة عناصر، وذلك لان الموقع الثاني b(2) منطقياً صفر

The above array-addressing techniques are summarized in the following table

Array Addressing

A(r,c)	Address a subarray within A defined by the index vector of desired rows in r and index vector of desired columns in c.
A(r,:)	Address a subarray within A defined by the index vector of desired rows in r and all columns.
A(:,r)	Address a subarray within A defined by all rows and the index vector of desired columns in r.
A(:)	Address all elements of A as a column vector taken column by column.
A(i)	Address a subarray within A defined by the single index vector of desired elements in i, as if A was the ~~index~~ column vector, a(:)
A(x)	Address a subarray within A defined by the logical array x. x must be the same size as A.

Subarray Searching:

Many times, it is desirable to know the indices or subscripts of those elements of an array that satisfy some relational expression. In Matlab, this task is performed by the function find, which returns the subscripts where a relational expression is True.

```
EDU >> x = -3:3
       x = -3  -2  -1  0  1  2  3
EDU >> k = find(abs(x)>1)  % finds those subscripts where abs(x)>1.
       k =
            1   2   6   7
EDU >> y = x(k)  % creates y using the indexes in k.
       y =
           -3  -2   2   3
```

The find function also works for matrices:

```
EDU >> A = [1 2 3; 4 5 6; 7 8 9]
EDU >> [i,j] = find(A>5.)
       i =
            3
            3
            2
            3
       j =
            1
            2
            3
            3
```

Here, the indices stored in i and j are the associated row and column indices, respectively, where the relational expression is true, that is, A(i(1),j(1)) is the first element of A where A>5, and so on.

Summary

i = find(x)	Return indices of the array x where its elements are nonzero.
[r,c] = find(x)	Return row and column indices of the array x where its elements are nonzero.

Array Comparisons:

```
EDU>> A=[1 2 3; 4 5 6; 7 8 9]' % transpose of earlier data.

A = 1 4 7
    2 5 8
    3 6 9

EDU>> B = A.*(-1).^A % an array close to A.

B = -1  4 -7
     2 -5  8
    -3  6 -9

EDU>> C=1:9 % an array having the same values of A, but
            reshaped

C = 1 2 3 4 5 6 7 8 9

EDU>> isequal(A,C)
ans = 0

EDU>> isequal(A,B)
ans = 0

EDU>> isequal(A,A)
ans = 1

EDU>> isequal(C,C')
ans = 0
```

The isequal function returns logical True (1) when two arrays have the same dimensions and identical elements. Otherwise, it returns logical False (0).

```
EDU>> ismember(A,B) % note result is a column vector
ans = 0
      1
      0
      1
      0
      1
      0
      1
      0
```

The function ismember identifies elements that are identical in two arrays.

```
EDU>> ismember(A,C)
ans = 1
      1
      1
      1
      1
      1
      1
      1
      1
```

ismember returns logical True for those indices in A that are in the second argument of ismember. The two arguments need not be the same size.


```
EDU>> X=0:2:20    % an array with 11 elements
    X=
      0  2  4  6  8  10  12  14  16  18  20
EDU>> ismember(X,A)
    ans=
      0  1  1  1  1  0  0  0  0  0  0
```

is an array the same size as X, with True at common elements.

```
EDU>> ismember(A,X) % is an array with the same number
% of elements as A, with True at common elements. So ismember
% compares its first argument to the second and returns a vector
% having the same number of elements as the first.
    ans=
      0
      1
      0
      1
      0
      1
      0
      1
      0
EDU>> union(A,B) % all elements common to two arrays, sorted
                 % ascending order ترتيب تصاعدي
    ans=
     -9
     -7
     -5
     -3
     -1
      1
      2
      3
      4
      5
      6
      7
      8
      9
```

```
EDU>> intersect(A,B) % the intersection of two arrays, sorted
ans =
     2
     4
     6
     8

EDU>> setdiff(A,B) % the values in A that are not in B, sorted
ans =              % A-B
     1
     3
     5
     7
     9

EDU>> setxor(A,B) % exclusive OR of two arrays, sorted
ans =             % المختلفة بالإشارة
    -9            % وفقاً لـ OR
    -7            % والترتيب تصاعدي.
    -5
    -3
    -1
     1
     3
     5
     7
     9
```