Chapter Four
Matrix

Matrices are the basic elements of the MATLAB environment.

A matrix is a two dimensional array consisting of **m** rows and **n** columns.

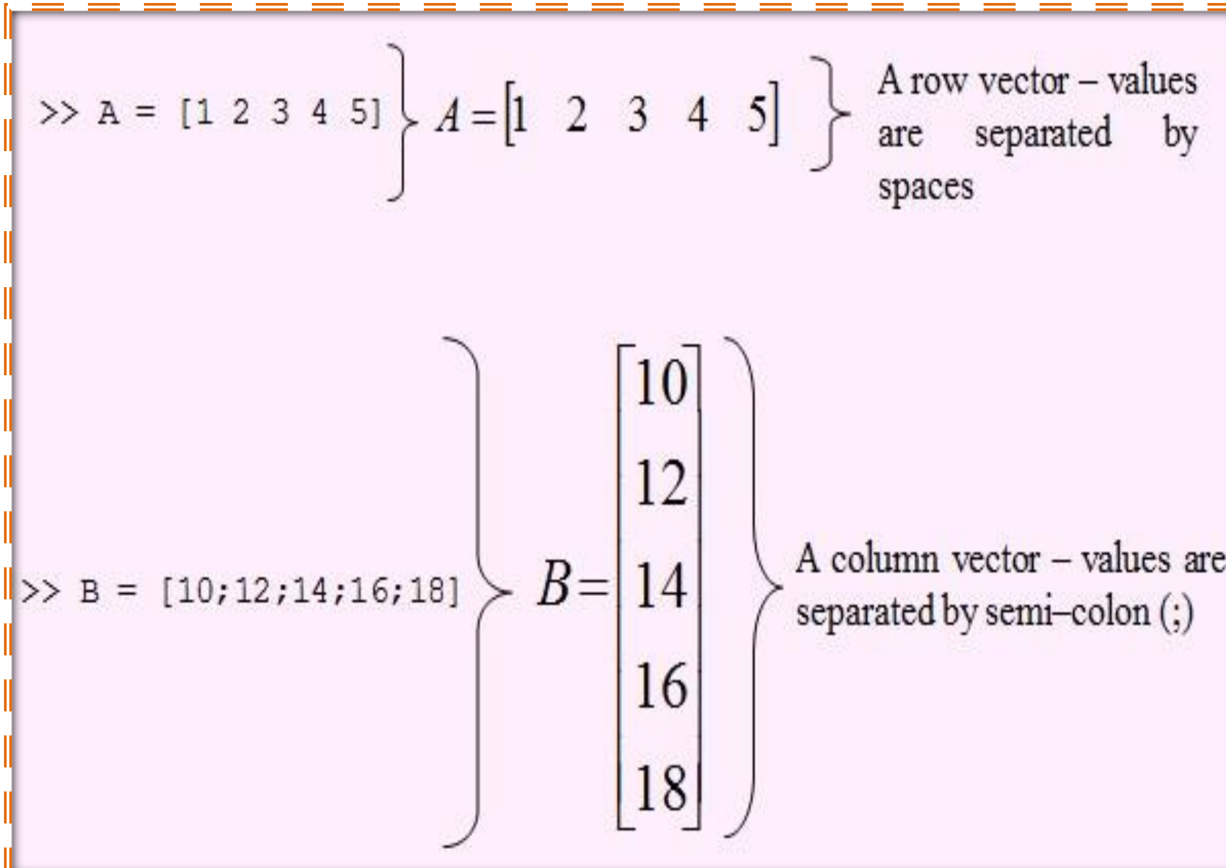
1. Entering a vector

In MATLAB an array of dimension **1× n** is called a row vector, where as an array of dimension **m×1** is called a column vector.

>> v = [1 4 7 10 13]

v =

1 4 7 10 13



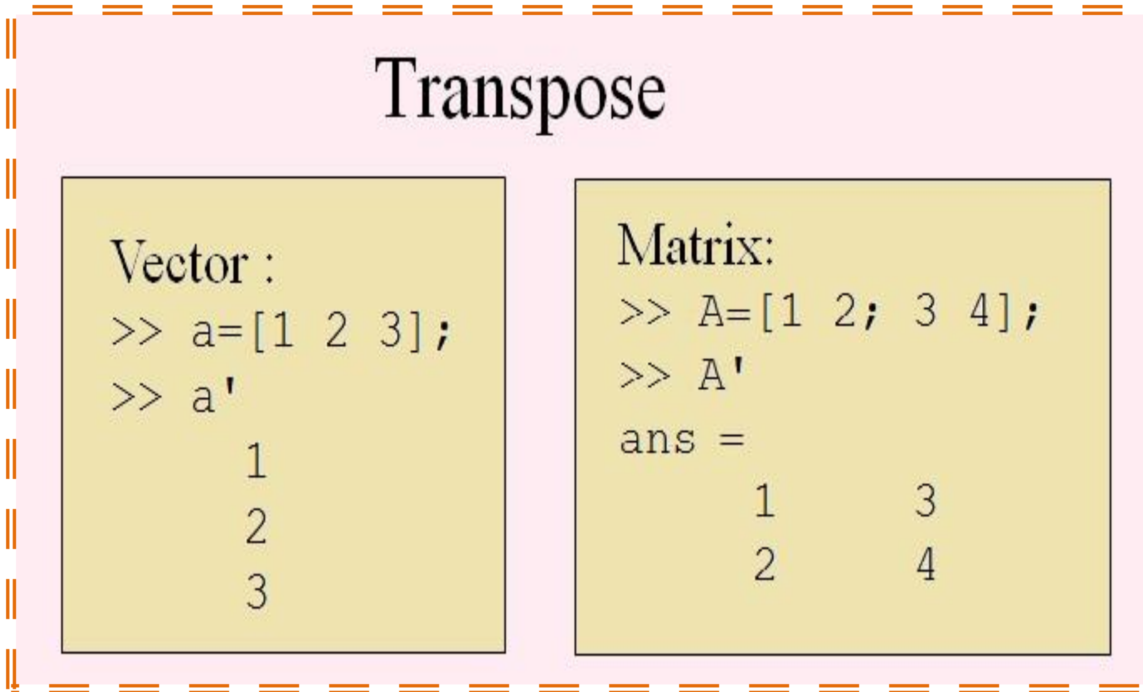
>> w = [1 ; 4 ; 7 ; 10 ; 13]

w =

1
4
7
10
13

On the other hand, a row vector is **converted** to a column vector using the transpose operator. The transpose operation is denoted by an apostrophe or a single quote (**'**).

```
>> w = v'
  w =
       1
       4
       7
      10
      13
```

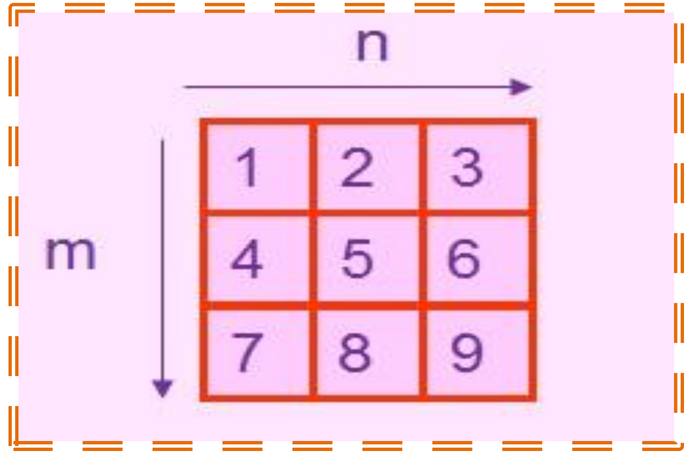



```
>> v(1:3)
    ans =
          1   4   7

>> v(3:end)
    ans =
          7   10   13
```

>> v(:) OR **>> v(1:end)** To display all items of the vector

2. Entering a matrix



To type a matrix into MATLAB you must:

- ❖ Begin with a square bracket, [
- ❖ Separate elements in a row with spaces or commas (,)
- ❖ Use a semicolon (;) to separate rows
- ❖ End the matrix with another square bracket,]

```
>> A = [ 1 2 3 ; 4 5 6 ; 7 8 9 ]
```

$$A = [1\ 2\ 3;\ 4\ 5\ 6;\ 7\ 8\ 9] = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

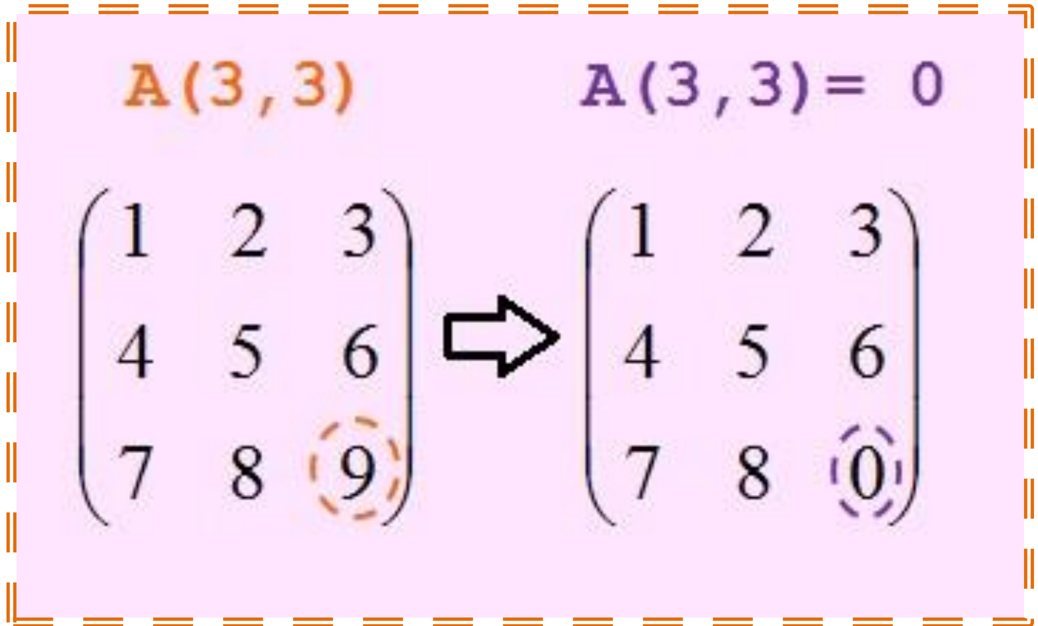
```
>> A (2,1)
   ans=
       4
```

3. Matrix indexing

The matrix **A** is denoted by **A(i,j)**. The element of row **i** and column **j**

```
>> A(1,3)
ans
     = 3
```

```
>> A(3,3) = 0
   A =
    1    2    3
    4    5    6
    7    8    0
```



Single elements of a matrix are accessed as **A(i,j)**, where **i ≥ 1** and **j ≥ 1**. **Zero** or **negative** subscripts are not supported in MATLAB.

4. Colon operator

Example, suppose we want to enter a vector **x** consisting of points (0 ; 0.1 ; 0.2 ; 0.3 ; ….. ; 5). We can use the command:

>> x = 0 : 0.1 : 5 ;

5. Linear spacing

For example:

y = linspace (a , b)

Generates a row vector **y** of **100** points linearly spaced between and including **a** and **b**.

y = linspace (a , b , n)

Generates a row vector **y** of **n** points linearly spaced between and including **a** and **b**.

>> theta = linspace (0 , 2*pi , 101)

Divides the interval [0 ; 2π] into 100 equal subintervals, then creating a vector of 101 elements.

Create vector with equally spaced intervals

```
>> x=0:0.5:pi
x =
 0 0.5000 1.0000 1.5000 2.0000 2.5000 3.0000
```

Create vector with *n* equally spaced intervals

```
>> x=linspace(0, pi, 7)
x =
  0 0.5236 1.0472 1.5708 2.0944 2.6180 3.1416
```

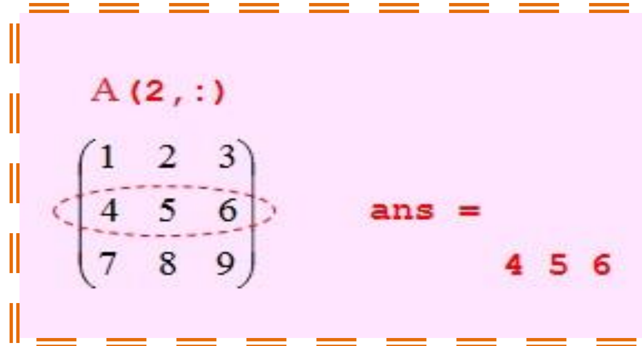
6. Colon operator in a matrix

The statement **A (m : n , k : l**) specifies rows **m** to **n** and column **k** to **l**.

>> A (2,:)

ans =

4 5 6



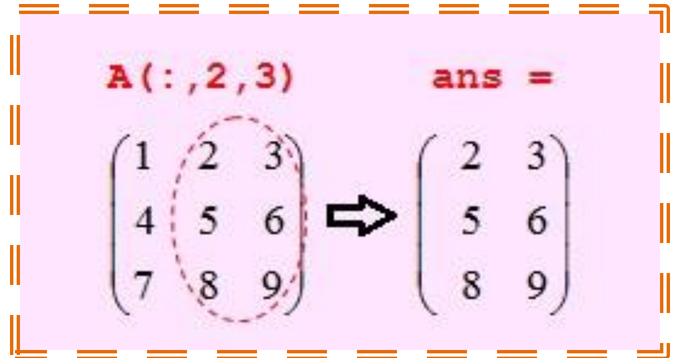
>> A (: , 2 : 3)

ans =

2 3

5 6

8 9



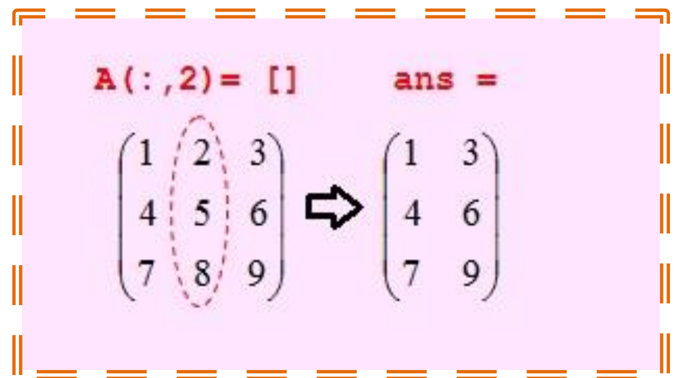
>> A (: , 2) = []

A =

1 3

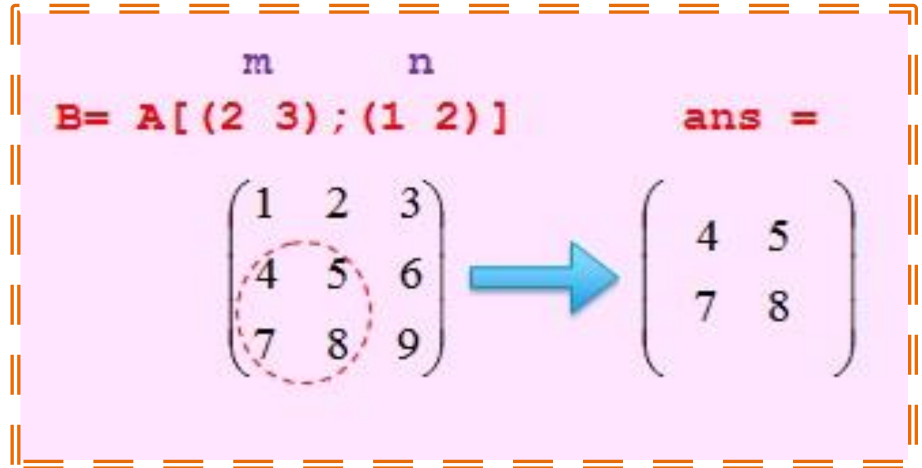
4 6

7 9

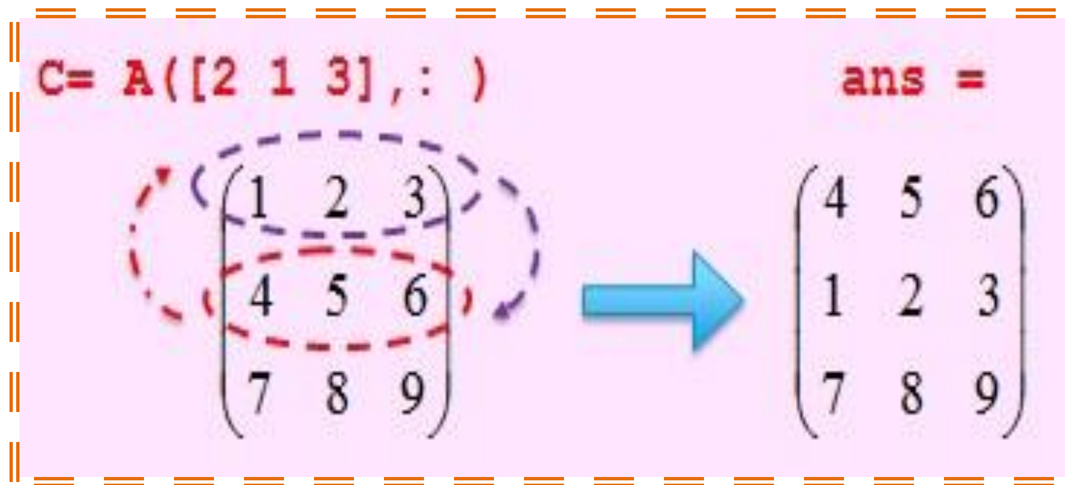


7. Creating a sub-matrix

To extract a sub matrix **B** consisting of rows **2** and **3** and columns **1** and **2** of the matrix **A**, do the following:




```
>> B = A ( [ 2  3 ] , [ 1  2 ] )
```




```
>> C = A ( [ 2  1  3] , : )
   C =
        4    5    6
        1    2    3
        7    8    9
```

>> A(:)

ans=

1

4

7

2

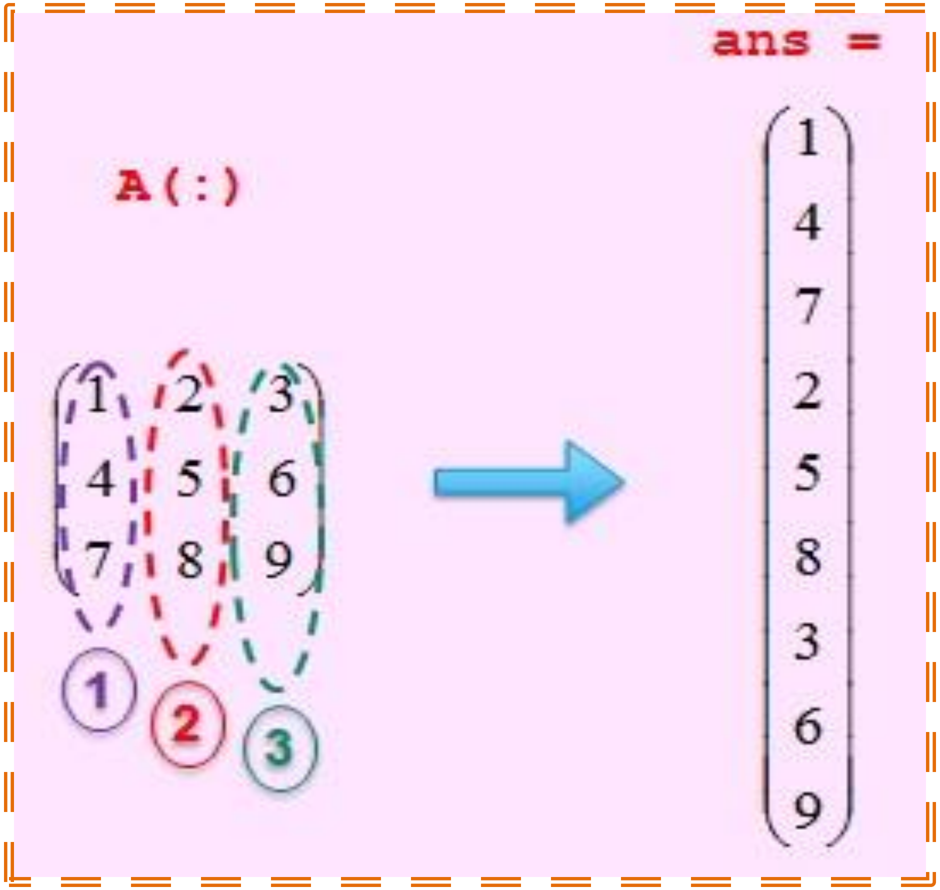
5

8

3

6

9



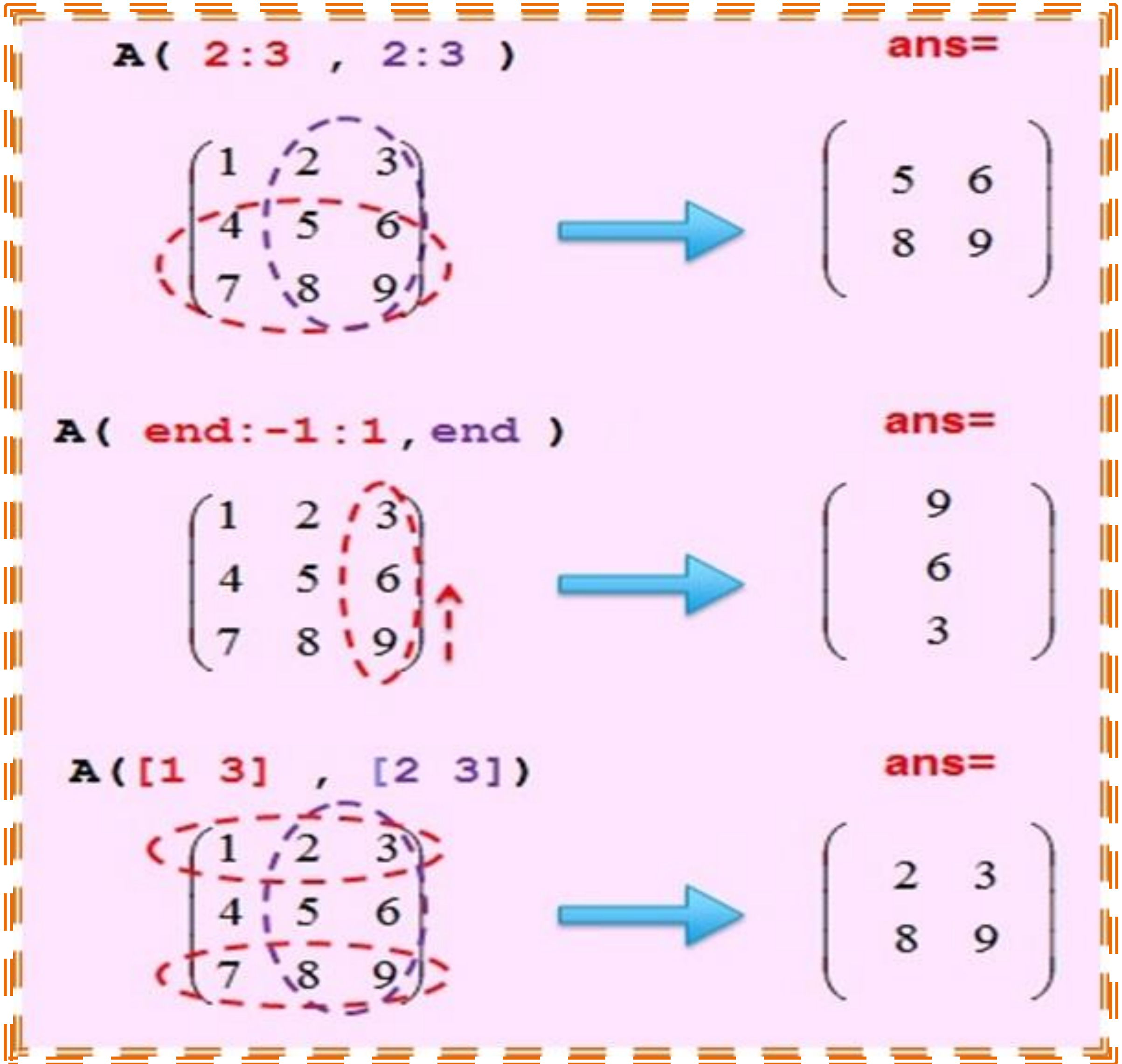
- **A (: , j)** is the **j**th column of A
- **A (i , :)** is the **i**th row of A
- **A (end , :)** picks out the last row of A

```
>> A
A =
    1     2     3
    4     5     6
    7     8     9

>> A ( 2 : 3 , 2 : 3 )
ans =
    5     6
    8     9

>> A ( end : -1 : 1 , end )
ans =
    9
    6
    3

>> A ( [1 3] ,  [2 3])
ans =
    2     3
    8     9
```

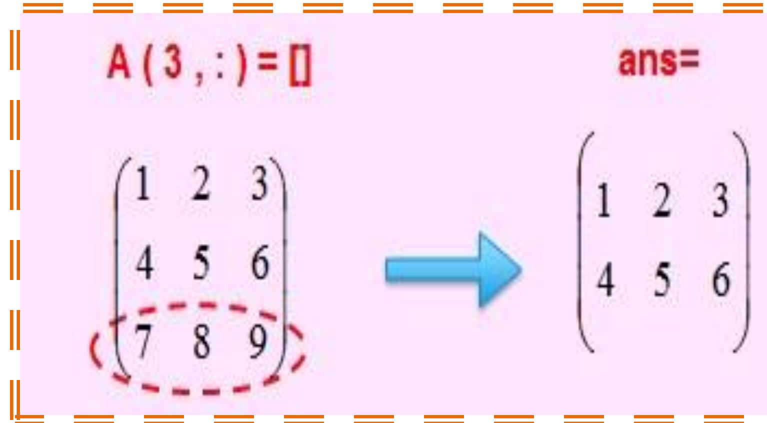


8. Deleting row or column

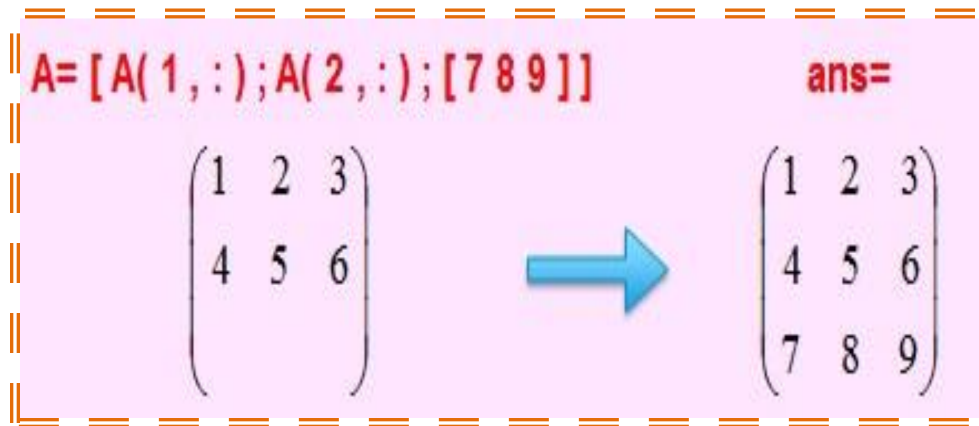
To delete a row or column of a matrix, use the empty vector operator, **[]**.

```
>> A ( 3 , : ) = []

A =
    1    2    3
    4    5    6
```




```
>> A= [ A( 1 , : ) ; A( 2 , : ) ; [ 7 8 9 ] ]
  A =
   1    2    3
   4    5    6
   7    8    9
```



Matrix **A** is now restored to its original form.

9. Dimension

To determine the dimensions of a matrix or vector, use the command **size**.
For example:

```
>> size(A)
  ans =
       3      3
```

Means **3** rows and **3** columns. Or more explicitly with:

```
>> [m , n] = size(A)

   m =
        3
   n =
        3
```

10. Continuation

If it is not possible to type the entire input on the same line, use consecutive periods, called an ellipsis …., to signal continuation, then continue the input on the next line.

```
B = [  4/5        7.23 * tan(x)          sqrt(6) ;  ...
       1/x^2          0              3/(x*log(x)) ;  ...
        x-7        sqrt(3)               x*sin(x)] ;
```

Note that blank spaces around **+ , - , =** signs are optional, but they improve readability.