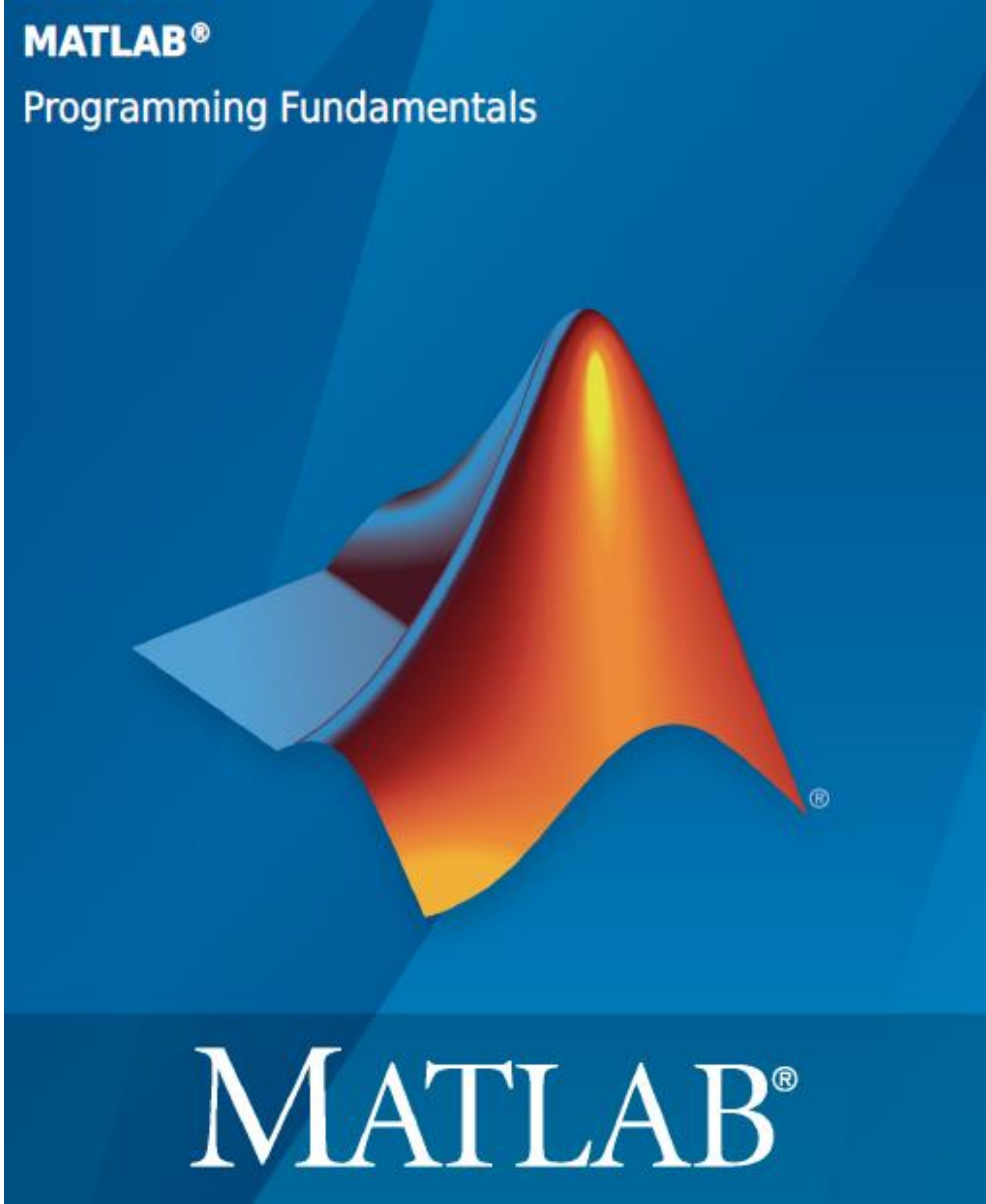
Petroleum and Mining Eng. Department

First Grade- Spring Semester

Lecture 2- Matrices using MATLAB

LECTURER: MS. JWAN KHALEEL M.

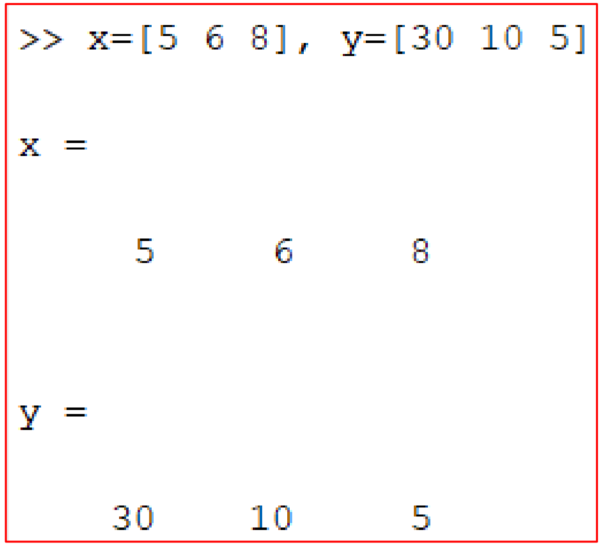


• Vectors and Matrices

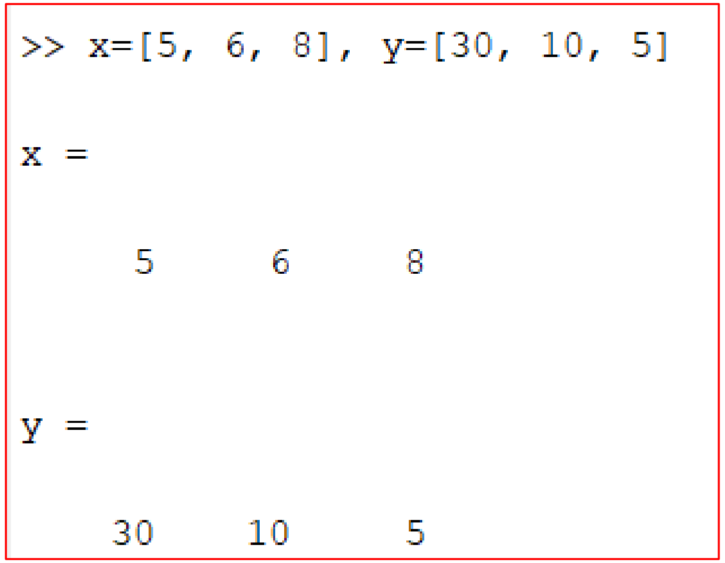
In the MATLAB environment, a matrix is a rectangular array of numbers. Special meaning is sometimes attached to 1-by-1 matrices, which are scalars, and matrices with only one row or column, which are vectors. MATLAB has other ways of storing both numeric and nonnumeric data, but in the beginning, it is usually best to think of everything as a matrix. The operations in MATLAB are designed to be as natural as possible. Where other programming languages work with numbers one at a time, MATLAB allows you to work with entire matrices quickly and easily.

Using the semicolon (;) and comma (,)

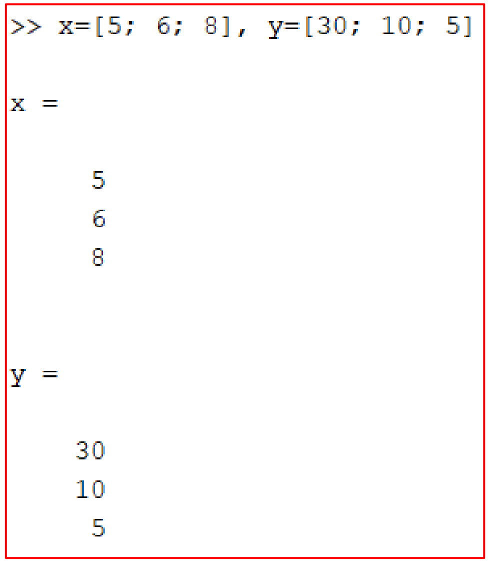
- The semicolon and comma can be used to write multiple MATLAB sentences in one line, but the difference is that the semicolon (;) execute the line without showing the result. Example:



```
>> x=[5 6 8], y=[30 10 5]

x =

     5     6     8


y =

    30    10     5
```

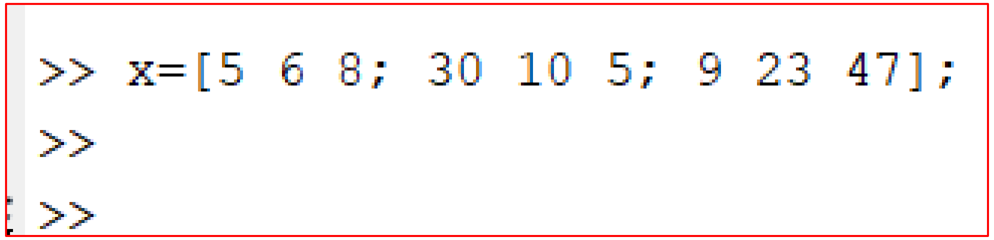


```
>> x=[5, 6, 8], y=[30, 10, 5]

x =

     5     6     8


y =

    30    10     5
```



```
>> x=[5; 6; 8], y=[30; 10; 5]

x =

     5
     6
     8


y =

    30
    10
     5
```

- The comma gives the result for the executed MATLAB code. Example:



```
>> x=[5 6 8; 30 10 5; 9 23 47];
>>
>>
```

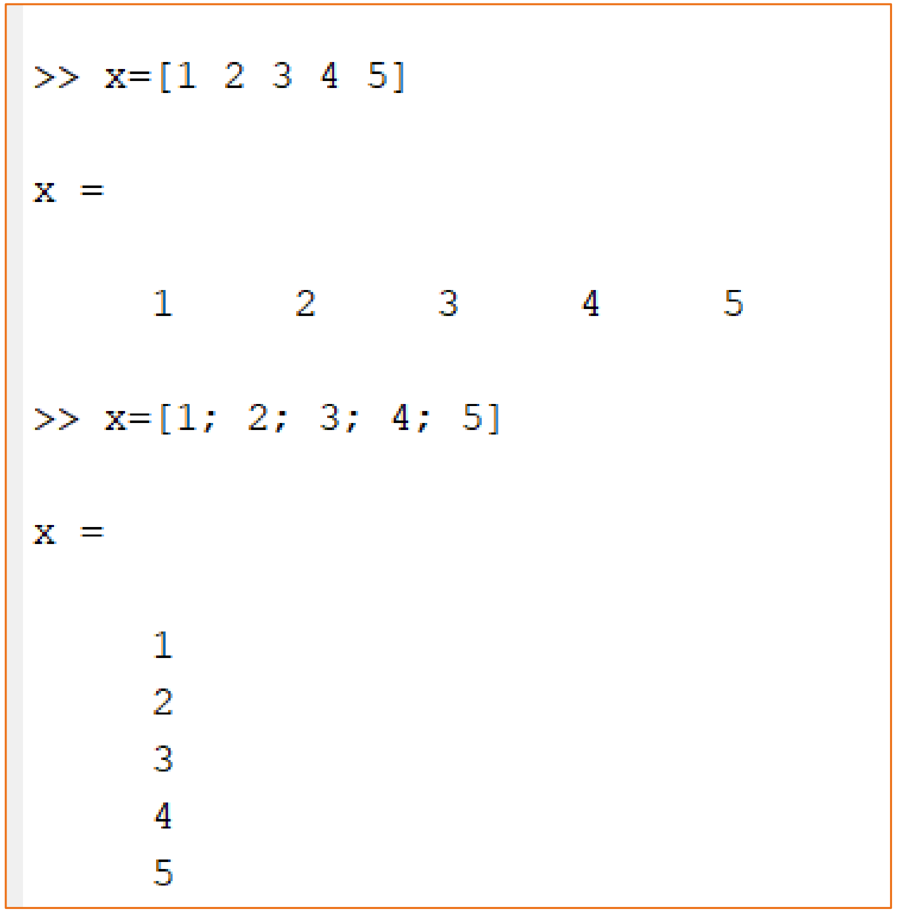
Vectors

The vector is one-dimensional array. The vector whose has (1 X n) dimension is called row vector, while whose has (n X 1) is called column vector.

<u>**The vector can be written by using the following procedure:**</u>

1. Separate the elements of a row with blanks or commas (,) for row vector and by semicolon (;) for column vector.

2. Surround the entire list of elements with square brackets, [ ].



```
>> x=[1 2 3 4 5]

x =

     1     2     3     4     5

>> x=[1; 2; 3; 4; 5]

x =

     1
     2
     3
     4
     5
```

The Colon Operator

The colon (:), is one of the most important MATLAB operators.

It occurs in several different forms.

Syntax of colon operator is:

vector_name = start : step :end

Where:

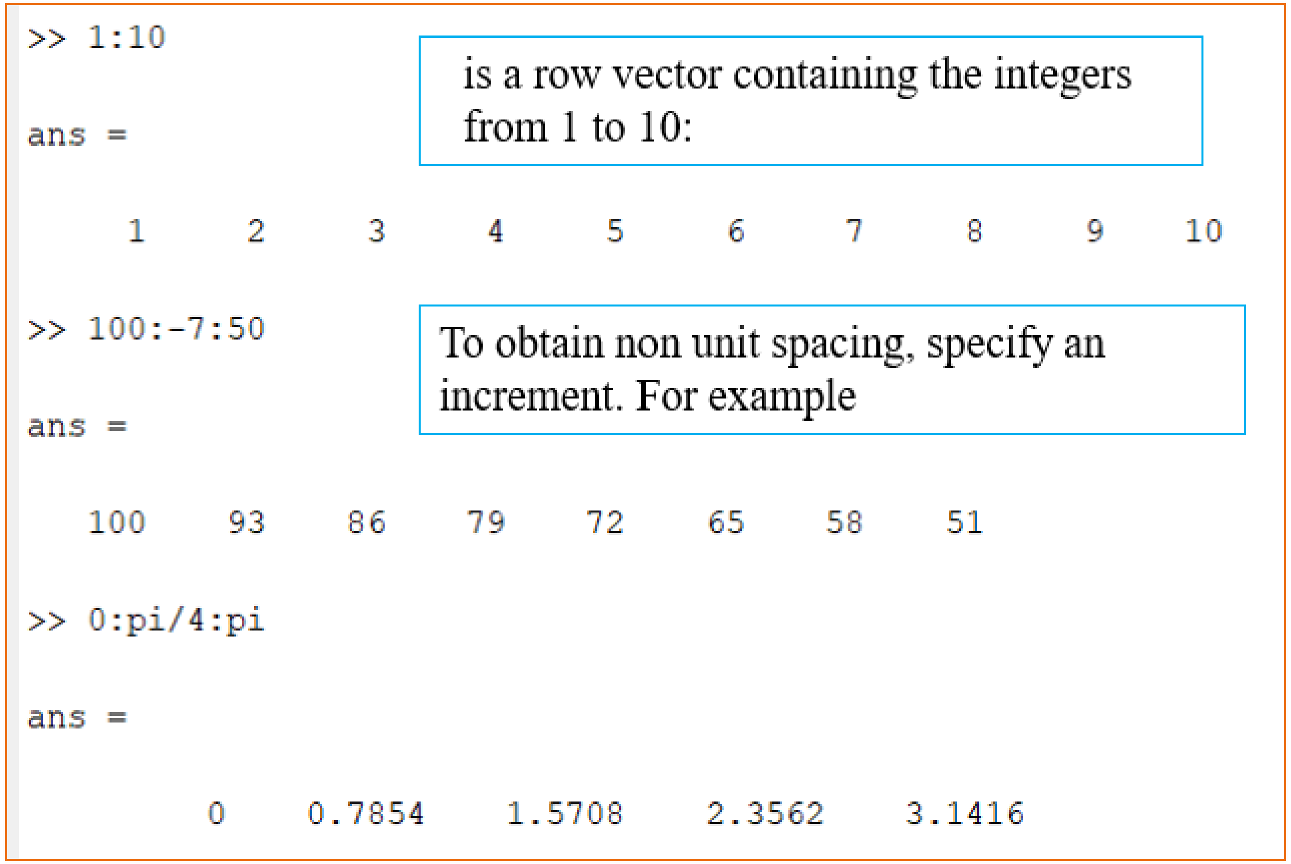
***Vector_name** is the name of the vector's variable name.*

***Start** is the first element in the vector.*

***End** is the last element in the vector.*

***Step** is the increment value.*

For example the expression

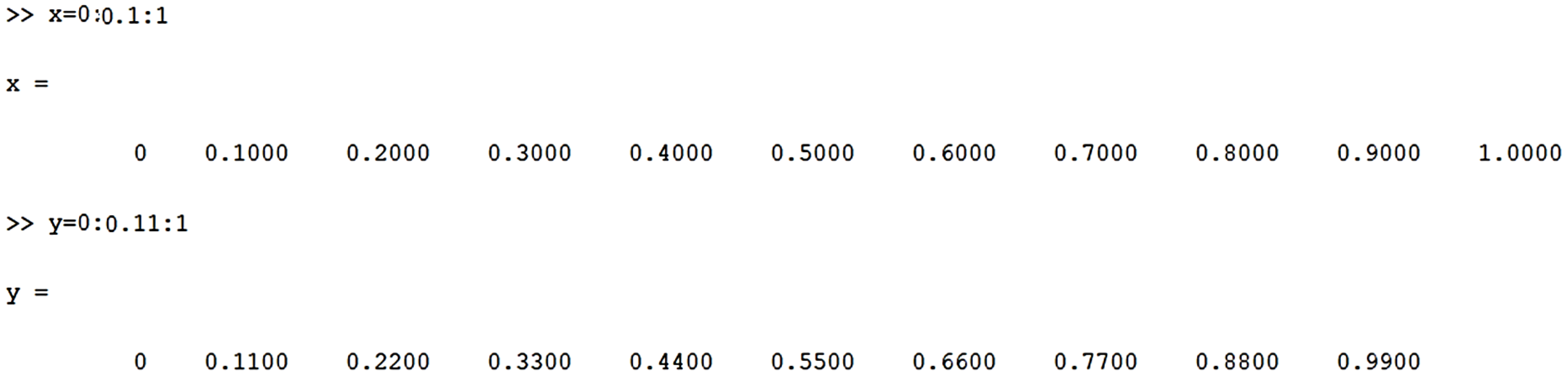


```
>> 1:10

ans =

     1     2     3     4     5     6     7     8     9    10

>> 100:-7:50

ans =

   100    93    86    79    72    65    58    51

>> 0:pi/4:pi

ans =

         0    0.7854    1.5708    2.3562    3.1416
```

is a row vector containing the integers from 1 to 10:

To obtain non unit spacing, specify an increment. For example

Example: Write a number between zero to 1 with the increment 0.1 and 0.11.



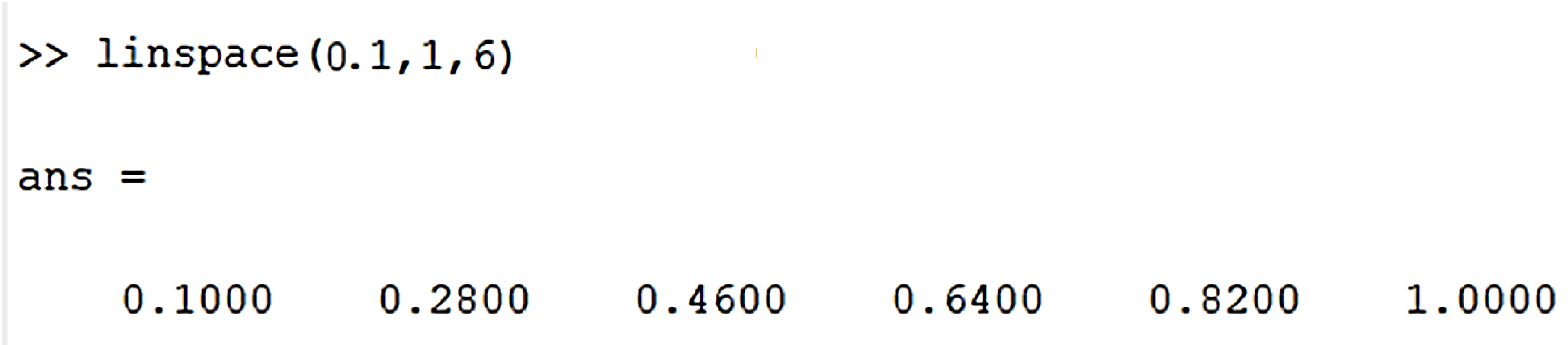
```
>> x=0:0.1:1

x =

         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000    1.0000

>> y=0:0.11:1

y =

         0    0.1100    0.2200    0.3300    0.4400    0.5500    0.6600    0.7700    0.8800    0.9900
```

• linspace Function

There is another way by using linspace, this syntax used to generate specified number

linspace (start value , end value , number of values)

Example: show six numbers between 0.1 to 1.



```
>> linspace(0.1,1,6)

ans =

    0.1000    0.2800    0.4600    0.6400    0.8200    1.0000
```

• Class work

- Create a set of numbers between 1 to 100 based on the following requirements:

  1. The increment between numbers = 0.5
  2. 15 numbers between 1 to 100.

Matrices

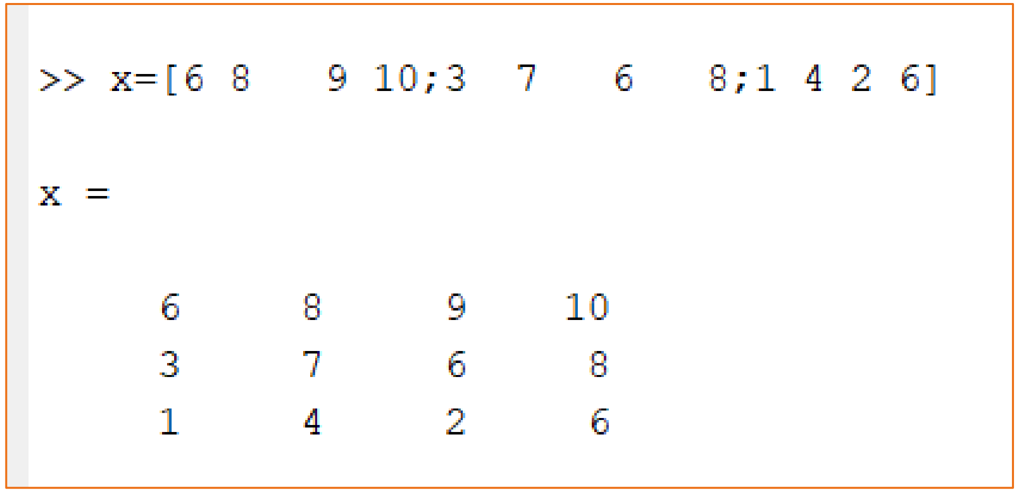
A matrix is a two-dimensional array. There are many ways to enter matrix in MATLAB.

- Load matrices from external data files.
- Generate matrices using built-in functions.
- Create matrices with your own functions in M-files.

The basic conventions could be followed

1. Separate the elements of a row with blanks or commas.
2. Use a semicolon (;) to indicate the end of each row.
3. Surround the entire list of elements with square brackets, [ ].

Example:

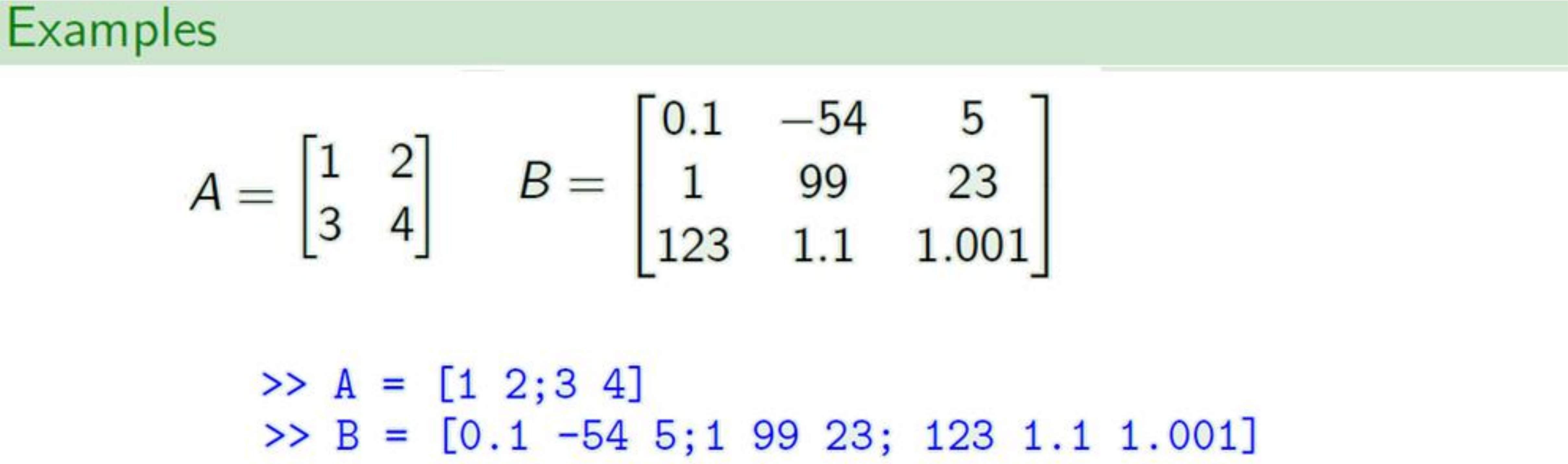


```
>> x=[6 8   9 10;3  7   6   8;1 4 2 6]

x =

     6     8     9    10
     3     7     6     8
     1     4     2     6
```

Once you have entered the vector or matrix, it is automatically remembered in the MATLAB workspace.

MATLAB software provides four functions that generate basic matrices.

Function Name	Description
zeros	All zeros
ones	All ones
rand	Uniformly distributed random elements
eye	identity matrix



Examples

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 0.1 & -54 & 5 \\ 1 & 99 & 23 \\ 123 & 1.1 & 1.001 \end{bmatrix}$$

```
>> A = [1 2;3 4]
>> B = [0.1 -54 5;1 99 23; 123 1.1 1.001]
```

Example

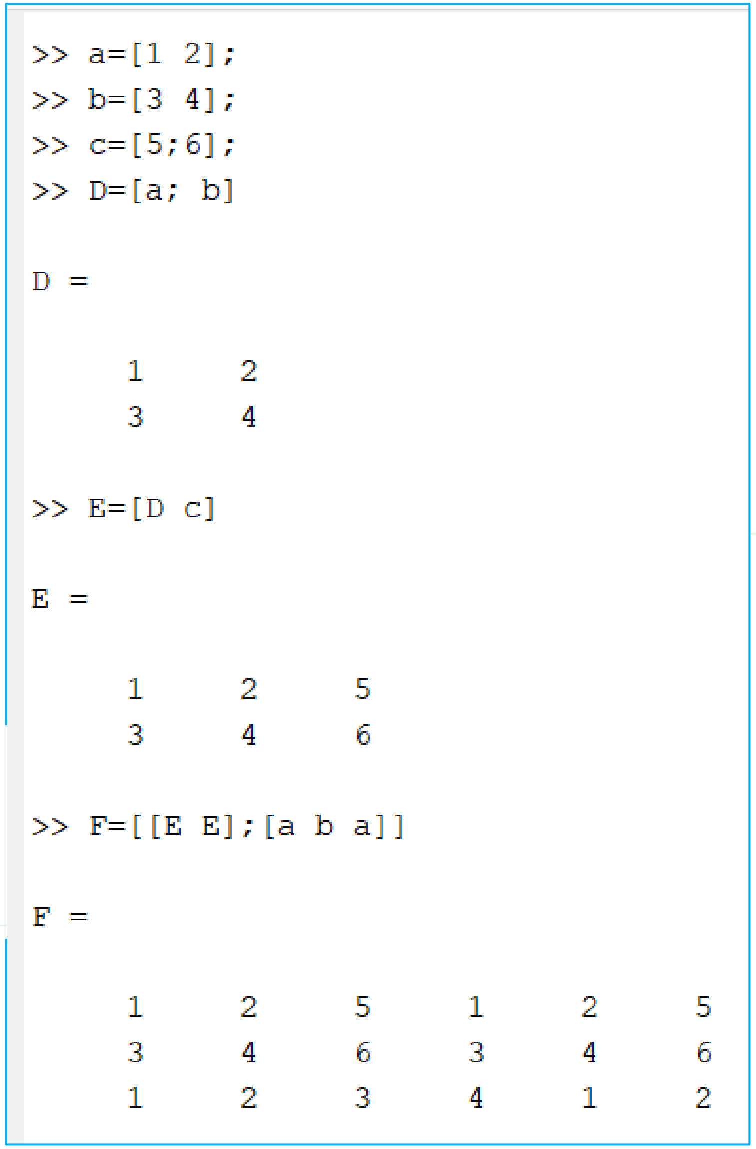
$$a = \begin{bmatrix}1 & 2\end{bmatrix} \quad b = \begin{bmatrix}3 & 4\end{bmatrix} \quad c = \begin{bmatrix}5 \\ 6\end{bmatrix}$$

Create:

- Matrix **D** contains elements of vector **a** and **b**
- Matrix **E** contains elements of matrix **D** and vector ***c***
- Matrix **F** contains elements of matrix **E, E** and ***a, b, a***

```
>> a = [1 2];
>> b = [3 4];
>> c = [5;6];
>> D = [a;b];
>> E = [D c];
>> F = [[E E];[a b a]];
```

$$D = \begin{bmatrix}1 & 2 \\ 3 & 4\end{bmatrix} \quad E = \begin{bmatrix}1 & 2 & 5 \\ 3 & 4 & 6\end{bmatrix} \quad F = \begin{bmatrix}1 & 2 & 5 & 1 & 2 & 5 \\ 3 & 4 & 6 & 3 & 4 & 6 \\ 1 & 2 & 3 & 4 & 1 & 2\end{bmatrix}$$



```
>> a=[1 2];
>> b=[3 4];
>> c=[5;6];
>> D=[a; b]

D =

     1     2
     3     4

>> E=[D c]

E =

     1     2     5
     3     4     6

>> F=[[E E];[a b a]]

F =

     1     2     5     1     2     5
     3     4     6     3     4     6
     1     2     3     4     1     2
```

• Zeros function

Create array of all zeros

Syntax

```
B = zeros(n)
B = zeros(m,n)
```

`B = zeros(n)` returns an `n−by−n` matrix of zeros.

An error message appears if `n` is not a scalar.

`B = zeros(m,n)` or `B = zeros([m n])` returns an `m`-by-`n` matrix of zeros.

• ones

Create array of all ones

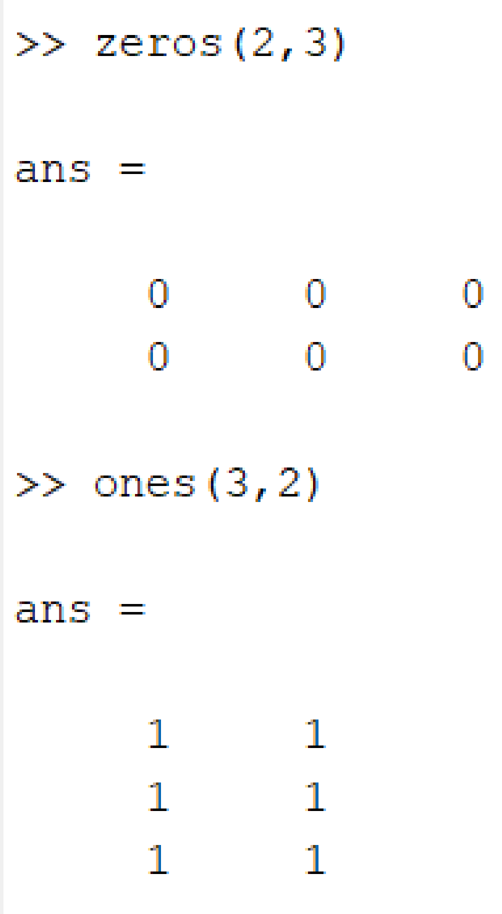
Syntax

```
Y = ones(n)

Y = ones(m,n)
```

`Y = ones(n)` returns an `n`-by-`n` matrix of `1`s. An error message appears if `n` is not a scalar.

`Y = ones(m,n)` or `Y = ones([m n])` returns an `m`-by-`n` matrix of ones.

Example



```
>> zeros(2,3)

ans =

     0     0     0
     0     0     0

>> ones(3,2)

ans =

     1     1
     1     1
     1     1
```

- # rand

Uniformly distributed pseudorandom numbers

Syntax

```
r = rand(n)
rand(m,n)
```

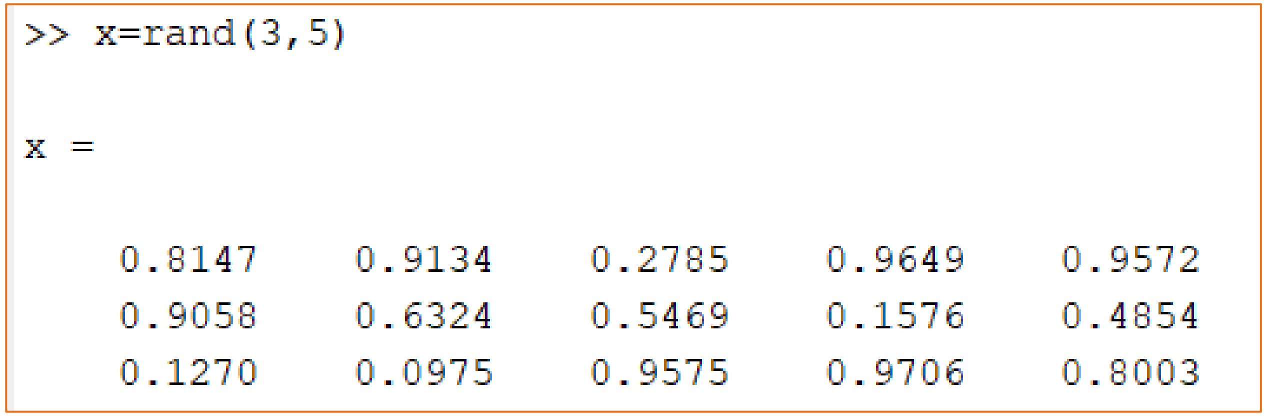
`r = rand(n)` returns an `n`-by-`n` matrix containing pseudorandom values drawn from the the standard uniform distribution on the open interval (0,1).

`rand(m,n)` or `rand([m,n])` returns an `m`-by-`n` matrix.

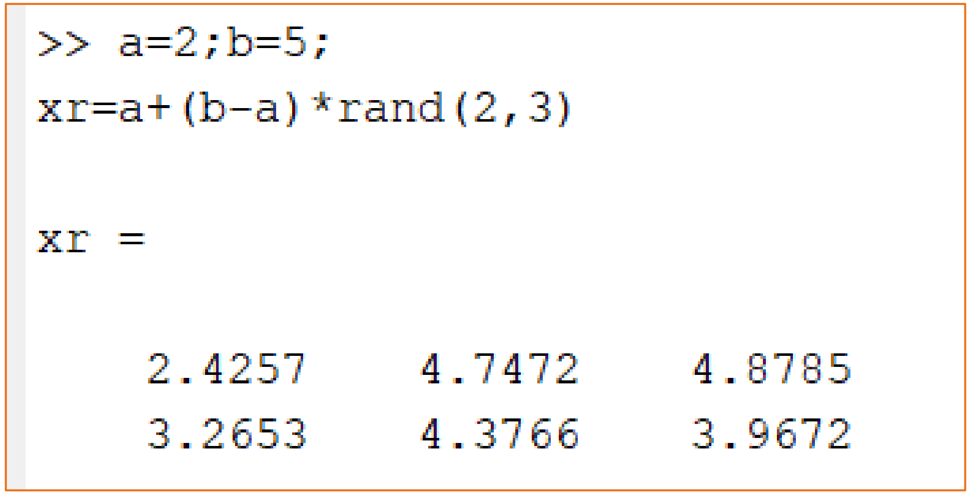
Example

1. Create a random matrix contain 3 rows and 5 columns
2. If a= 2 and b=3

Find a random matrix in 2 row and 3 column for c = a+(b-a)



```
>> x=rand(3,5)

x =

    0.8147    0.9134    0.2785    0.9649    0.9572
    0.9058    0.6324    0.5469    0.1576    0.4854
    0.1270    0.0975    0.9575    0.9706    0.8003
```



```
>> a=2;b=5;
xr=a+(b-a)*rand(2,3)

xr =

    2.4257    4.7472    4.8785
    3.2653    4.3766    3.9672
```

eye

Generate identity matrix

Syntax

```
Y =eye(n)
Y = eye(m,n)
```

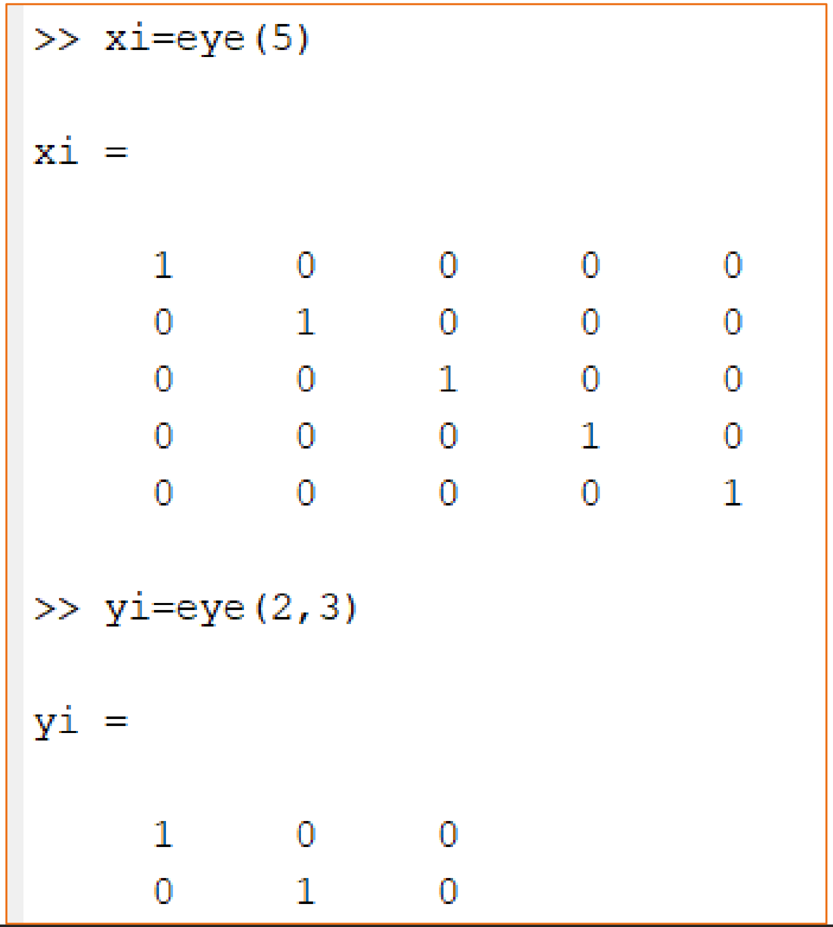
`Y = eye(`*n*`)` returns the *n*-by-*n* identity matrix.

`Y = eye(`*m*`,`*n*`)` or

`Y = eye([`*m n*`])`

returns an *m*-by-*n* matrix with `1`'s on the diagonal and `0`'s elsewhere. The size inputs *m* and *n* should be nonnegative integers.

Negative integers are treated as 0.



```
>> xi=eye(5)

xi =

     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1

>> yi=eye(2,3)

yi =

     1     0     0
     0     1     0
```

• Subscripts

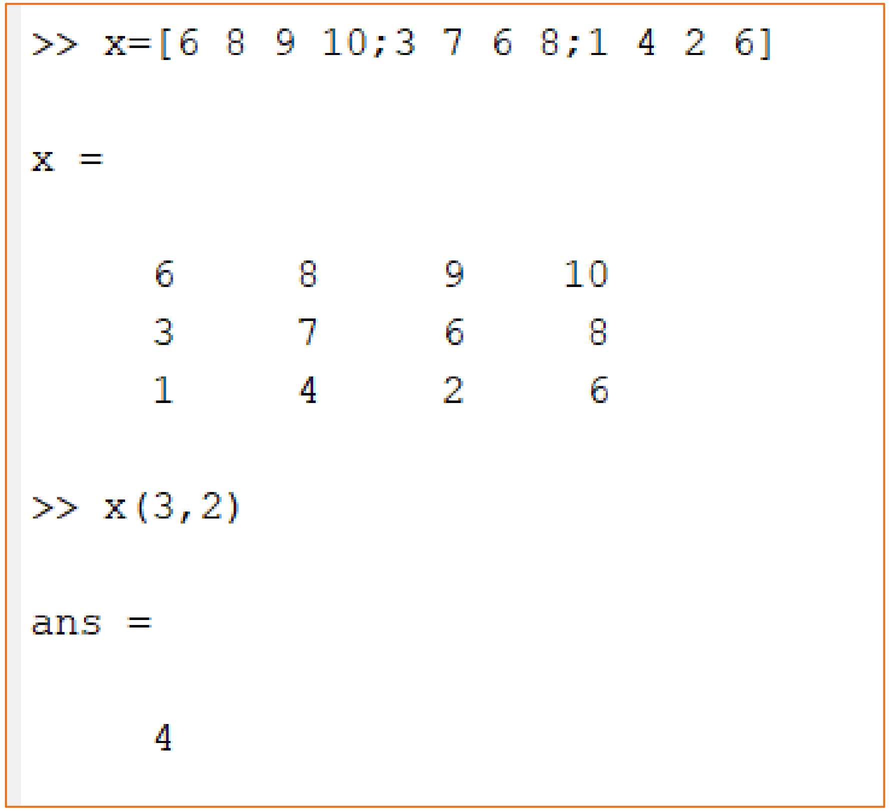
Mathematically, the matrix element can be represented by $\boldsymbol{A_{ij}}$ where $\boldsymbol{i}$ is the row index for that element and **j** is its column index.

In MATLAB the element in row i and column j of A matrix is denoted by $A(i, j)$. The same representation can be applied to specify the vector elements x(i).

Subscripts

- To display all content of a matrix, just write name of vector and press enter key.
- To display a row in a matrix is, *name of matrix (row number to be display, :)*
- To display a column in a matrix is, *name of matrix (: , column number to be display)*
- To display a specific element in a specific row of a matrix is, *name of matrix (row number, number of first element : number of second element).*
- To display a specific element in a specific column of a matrix is, *name of matrix (number of first element : number of second element, column number).*

Example

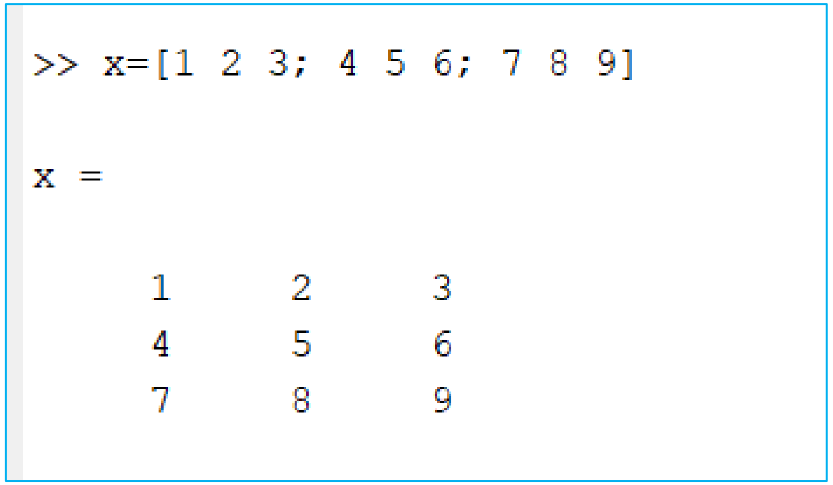


```
>> x=[6 8 9 10;3 7 6 8;1 4 2 6]

x =

     6     8     9    10
     3     7     6     8
     1     4     2     6

>> x(3,2)

ans =

     4
```

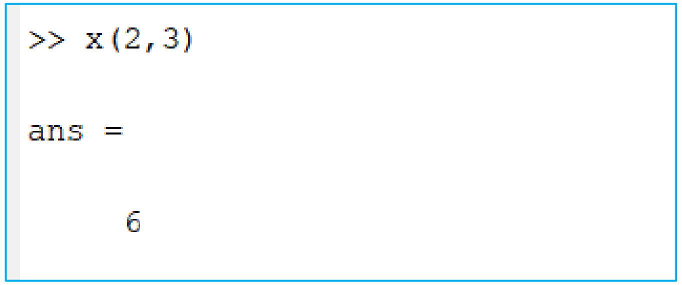
X(1,1)	6	X(1,2)	8	X(1,3)	9	X(1,4)	10
X(2,1)	3	X(2,2)	7	X(2,3)	6	X(2,4)	8
X(3,1)	1	X(3,2)	4	X(3,3)	2	X(3,4)	6

Note: the colon operation can be used to loop through the matrix elements.

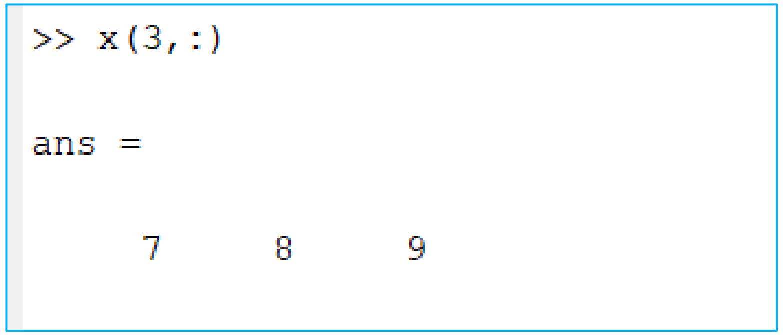
Example (showing a row or column in a matrix)



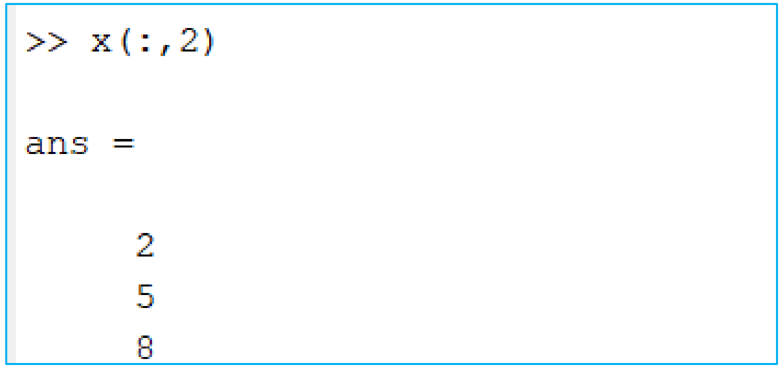
```
>> x=[1 2 3; 4 5 6; 7 8 9]

x =

     1     2     3
     4     5     6
     7     8     9
```



```
>> x(2,3)

ans =

     6
```

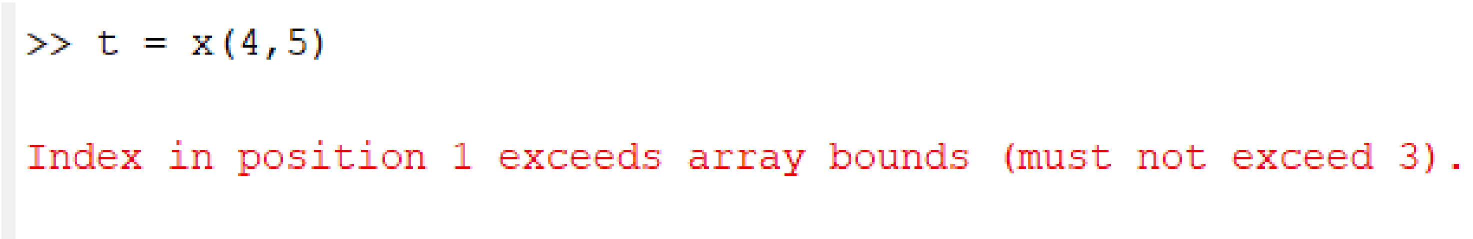


```
>> x(3,:)

ans =

     7     8     9
```



```
>> x(:,2)

ans =

     2
     5
     8
```

If you try to use the value of an element outside of the matrix, it is an error:



```
>> t = x(4,5)

Index in position 1 exceeds array bounds (must not exceed 3).
```

• Deleting Rows and Columns

You can delete rows and columns from a matrix using just a pair of square brackets. Start with Then, to delete the second column of X, use

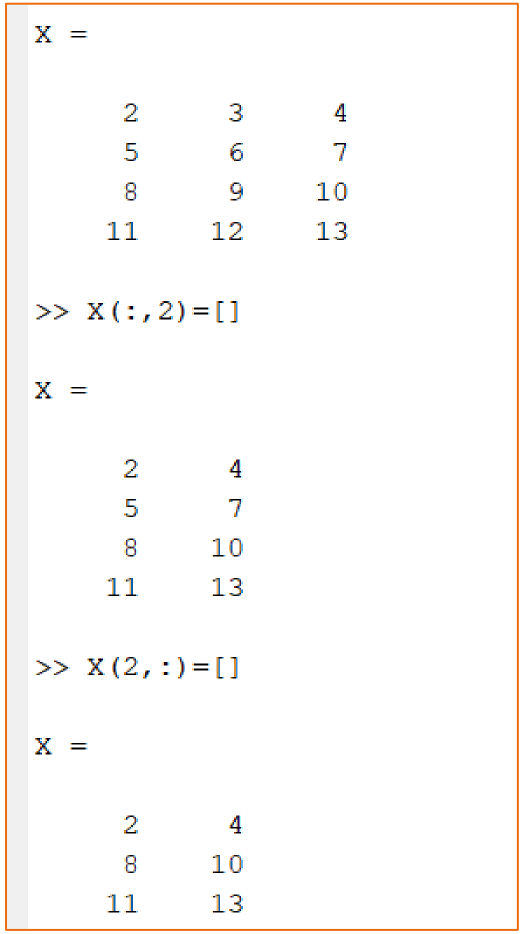
Syntax

Name or matrix (: , sequence of column)=[ ]

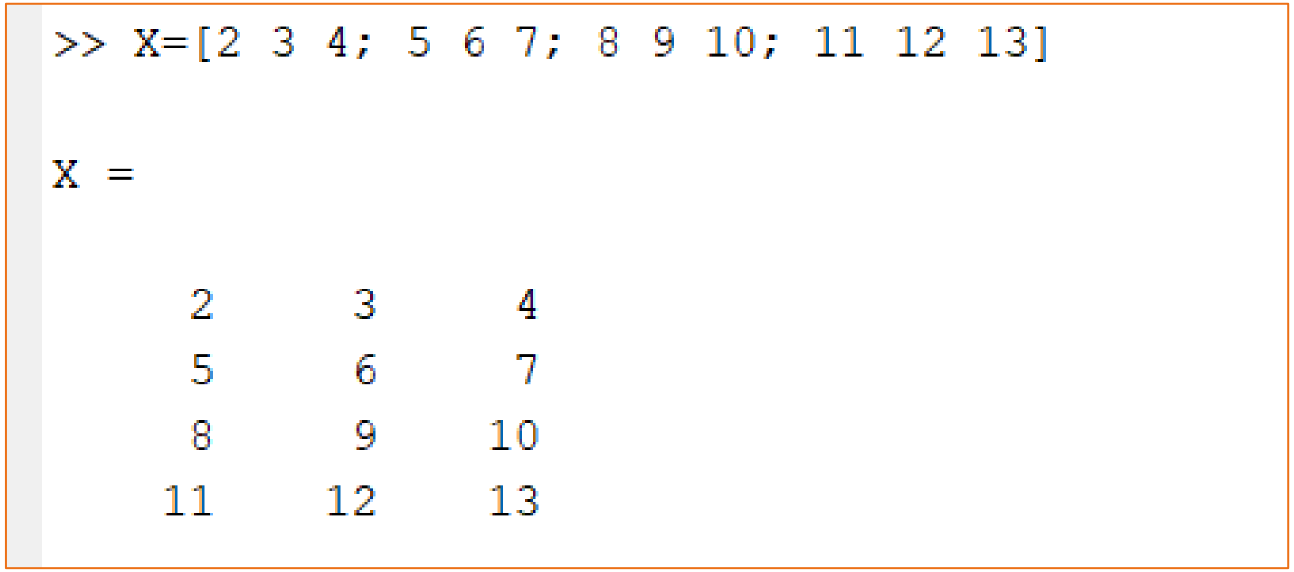
Name or matrix (sequence of row, :)=[ ]

If you delete a single element from a matrix, the result is not a matrix anymore. So, expressions like
X(1,2) = [ ]   result in an error. However, using a single subscript deletes a single element, or sequence of elements, and reshapes the remaining elements into a row vector.

- # Deleting Rows and Columns

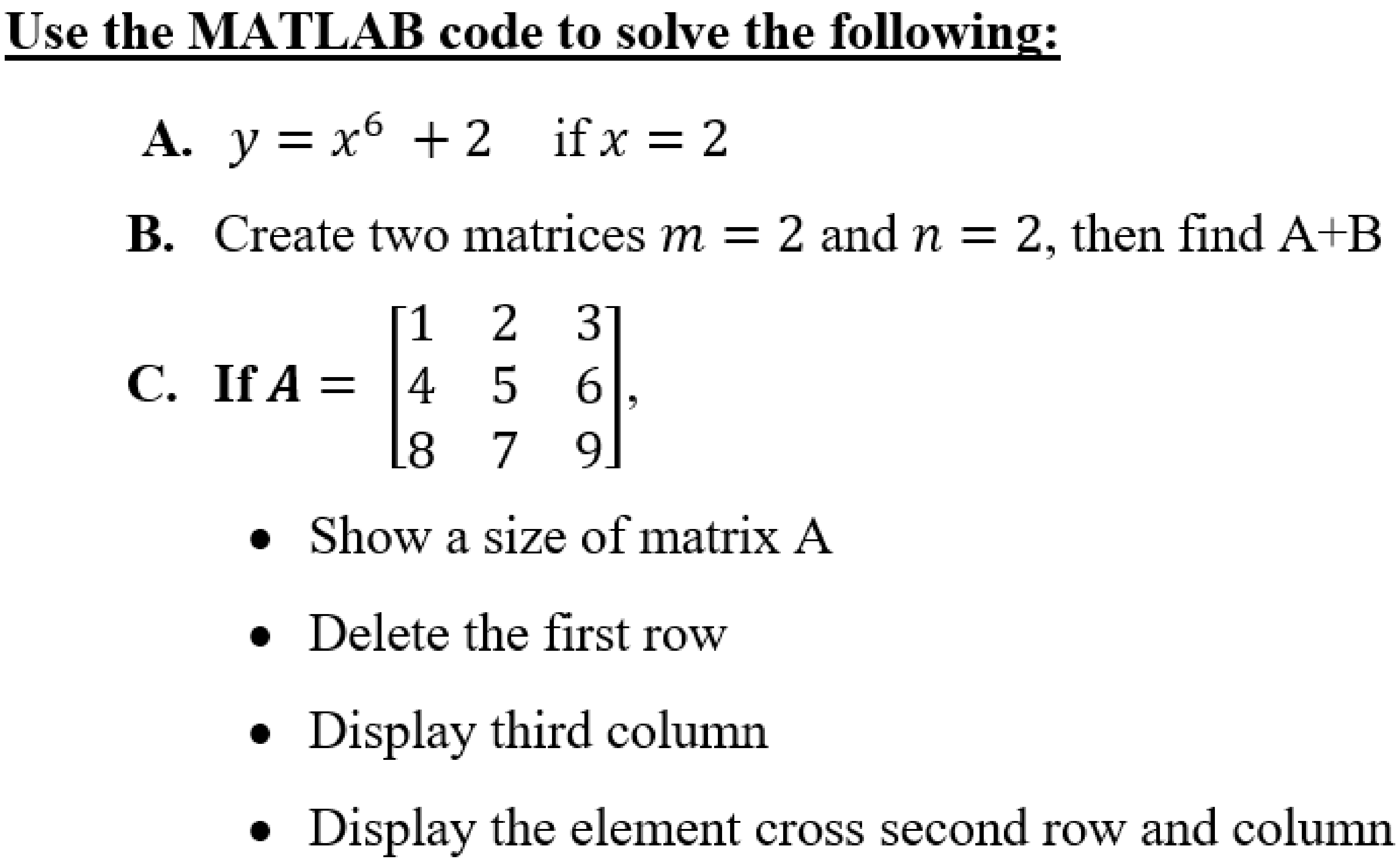


```
X =

     2     3     4
     5     6     7
     8     9    10
    11    12    13

>> X(:,2)=[]

X =

     2     4
     5     7
     8    10
    11    13

>> X(2,:)=[]

X =

     2     4
     8    10
    11    13
```



```
>> X=[2 3 4; 5 6 7; 8 9 10; 11 12 13]

X =

     2     3     4
     5     6     7
     8     9    10
    11    12    13
```

Class activities:



Use the MATLAB code to solve the following:

A. $y = x^6 + 2 \quad \text{if } x = 2$

B. Create two matrices $m = 2$ and $n = 2$, then find A+B

C. If $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 8 & 7 & 9 \end{bmatrix}$,

- Show a size of matrix A
- Delete the first row
- Display third column
- Display the element cross second row and column

The End of The Lecture
Enjoy your day