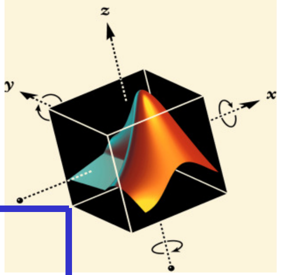
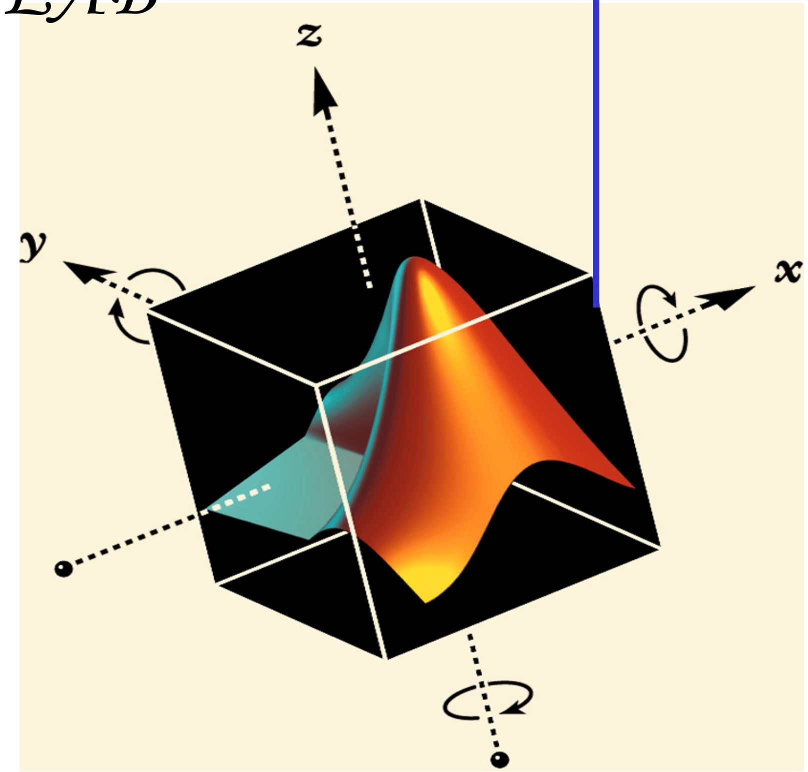
AL- Mustansirya University
College of Education
Department of Computer Science

Course: Intelligent Applications
Lecturer: Iman Hussein
Lab: Matlab Language
Fourth Class

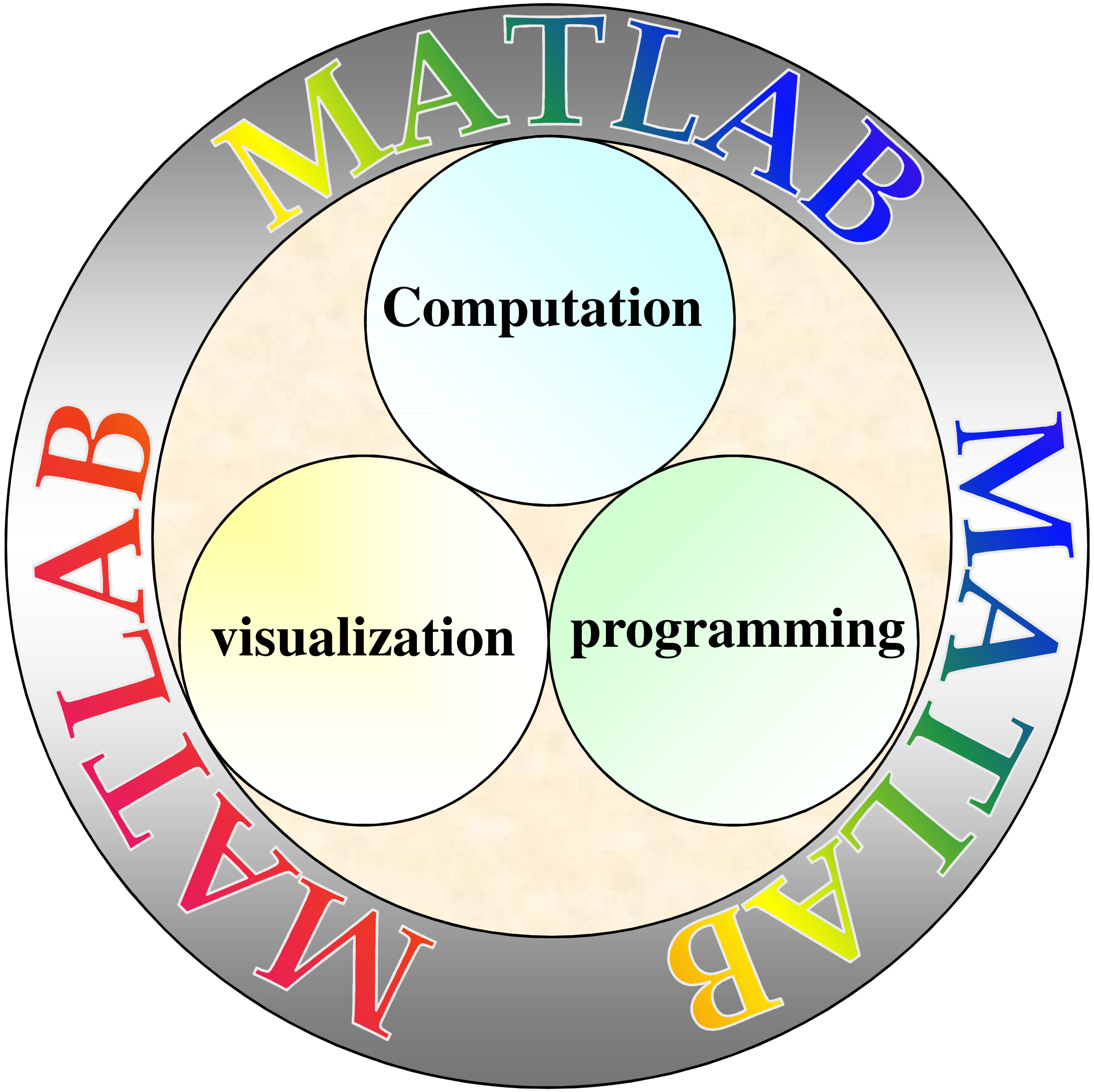


Chapter 1

Introduction to MATLAB



What is MATLAB ?
(MATrix LABoratory)



Course Outline

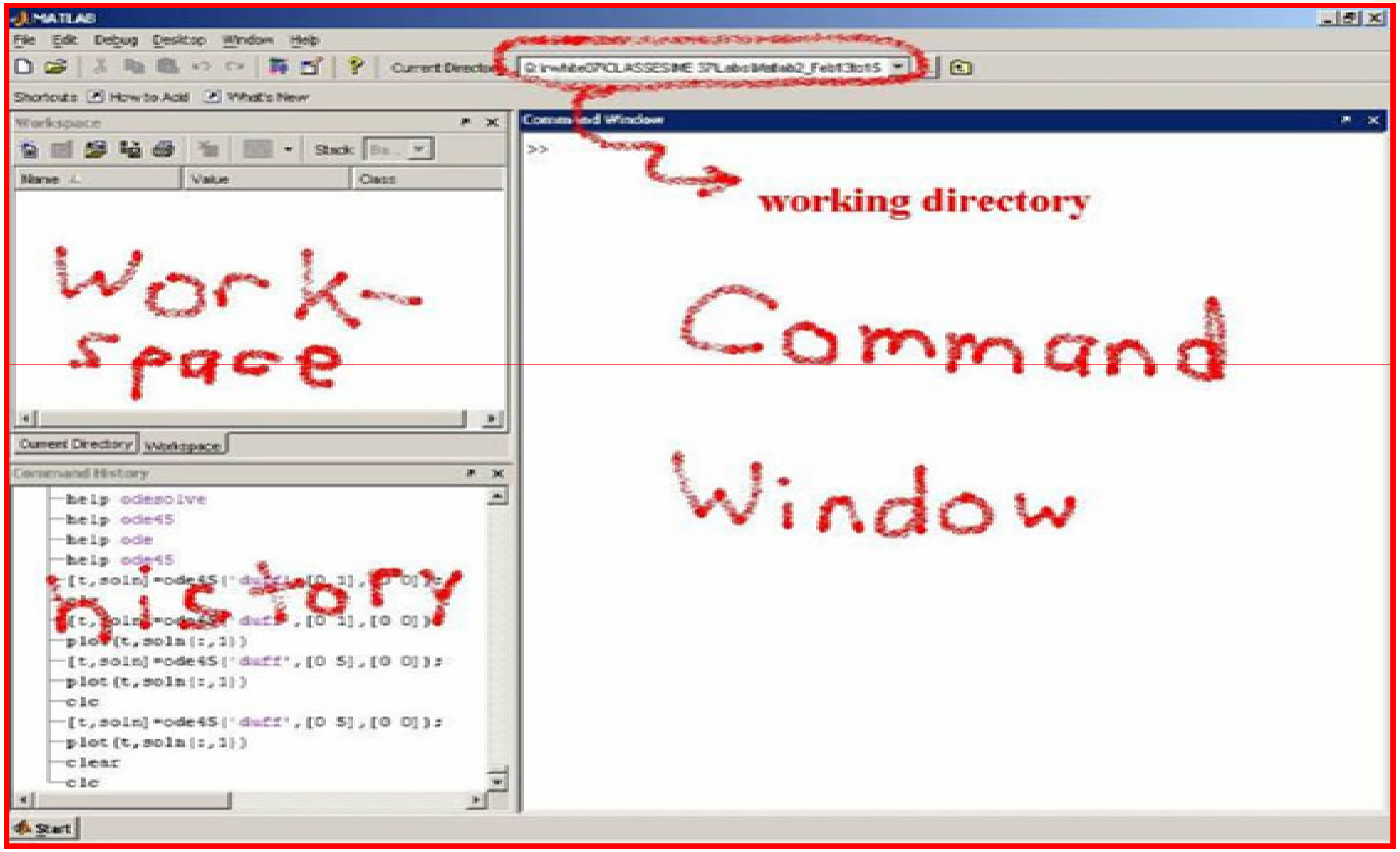
- MATLAB Basics
- MATLAB Programming
- Graphical User Interface

Basics of MATLAB

- Toolboxes
  - Symbolic
  - DSP
  - Image Processing
- Simulink

MATLAB for Engineers

Start …


MATLAB
File Edit Debug Desktop Window Help
Current Directory
Shortcuts How to Add What's New
Workspace
Stack
Name Value Class
Current Directory Workspace
Command History
Command Window
Start

```
help odesolve
help ode45
help ode
help ode45
[t,soln]=ode45('duff',[0 1],[0 0]);
clc
[t,soln]=ode45('duff',[0 1],[0 0]);
plot(t,soln(:,1))
[t,soln]=ode45('duff',[0 5],[0 0]);
plot(t,soln(:,1))
clc
[t,soln]=ode45('duff',[0 5],[0 0]);
plot(t,soln(:,1))
clear
clc
```

```
>>
```

working directory

Work-space

Command Window

history

General Notes

- Three ways to work on MATLAB
  1. Command Window
  2. M-file
  3. Simulink

- Any parameter (scalar,vector,matrix,..) are saved directly in the workspace after run the program.

- MATLAB is very Sensitive . . . . !

Command Window

- Just enter is enough to run.
- Simple, but can’t save.
- Good for small program.
- Each line start with >> .
- Any parameter saved in workspace .
- Use semi column at end of each line .

Examples

>>5

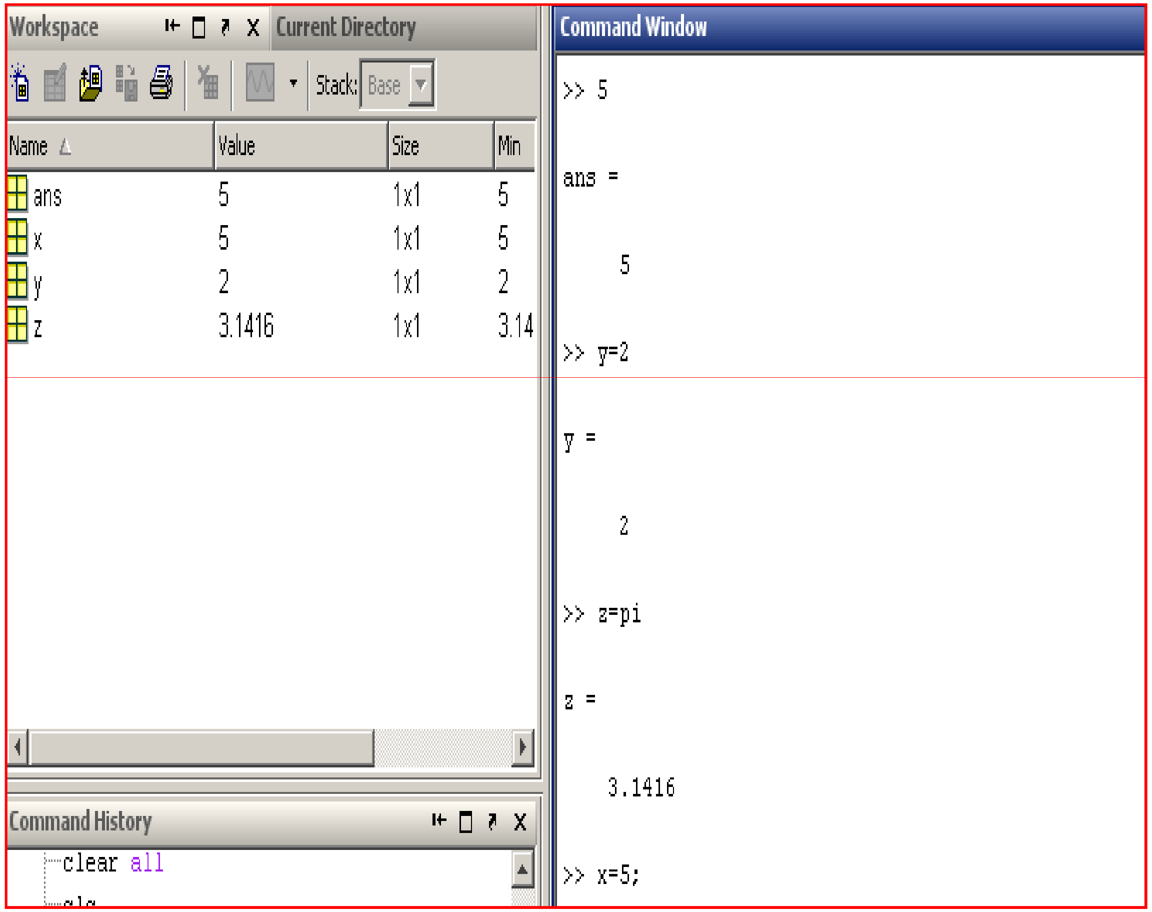
>> y=2

>> z=pi

>> pi

>> x=5;

(clc, clear all)

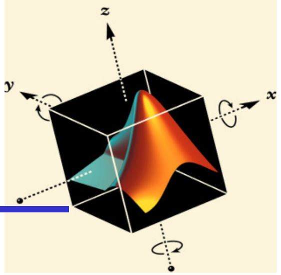


Name	Value	Size	Min
ans	5	1x1	5
x	5	1x1	5
y	2	1x1	2
z	3.1416	1x1	3.14

```
>> 5

ans =

     5

>> y=2

y =

     2

>> z=pi

z =

    3.1416

>> x=5;
```

```
clear all
clc
```

Notes



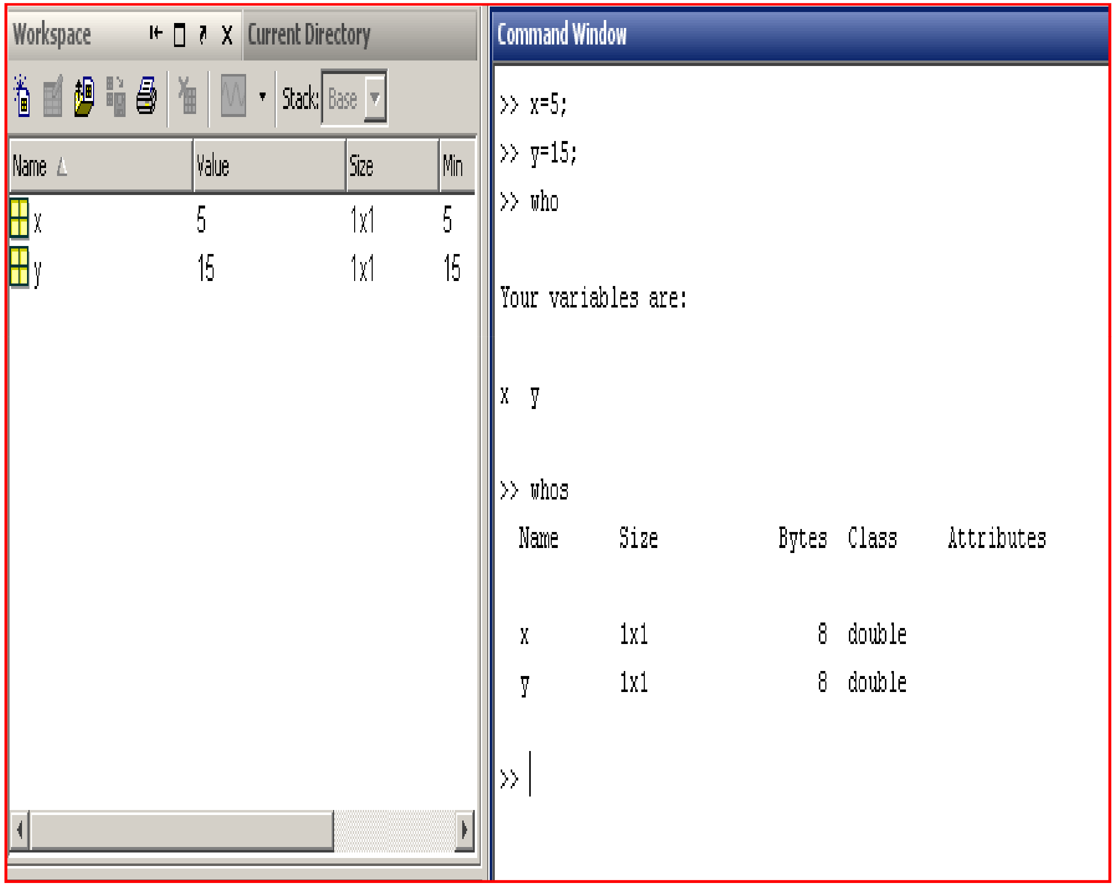
- When use Semi column ...
- x=5; x=6; then in work space (x=6)

Special Variables

- ans: Default variable name for results
- pi: Value of π
- eps: Smallest incremental number
- inf: Infinity
- NaN: Not a number e.g. 0/0
- i or j (imaginary number)

Managing Variables

```
>>clc
>>clear all
>>  x=5;
>> y=15;
>> who
>> whos
```



Name	Value	Size	Min
x	5	1x1	5
y	15	1x1	15

```
>> x=5;
>> y=15;
>> who

Your variables are:

x  y

>> whos
  Name      Size                Bytes  Class     Attributes

  x         1x1                     8  double

  y         1x1                     8  double

>> |
```

Scalar , Vector and Matrix

- a scalar `x = pi`

- a vector `x = [1 2 5 1]`

```
x =
       1    2    5    1
```

- a matrix `x = [1 2 3; 5 1 4; 3 2 -1]`

```
x =
       1      2      3
       5      1      4
       3      2     -1
```

Matrix

- x(i,j) subscription

```
y=x(2,3)
y =
     4
```

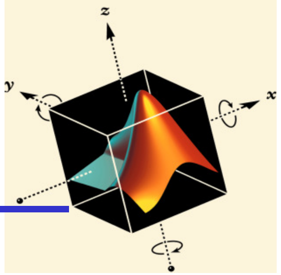
- whole row

```
y=x(3,:)
y =
     3     2    -1
```

- whole column

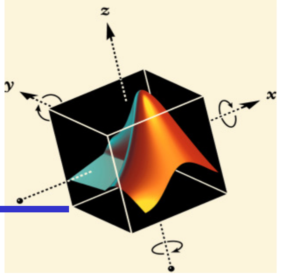
```
y=x(:,2)
y =
     2
     1
     2
```

Mathematical Functions



- exp(x)
- Sqrt(x)
- Log(x)
- Log10(x)
- abs(x)
- angle(x)
- conj(x)
- imag(x)
- real(x)
- sign(x)
- max(x)
- min(x)
- sum(x)
- mean(x)
- diag(x)
- Prod(x)
- mean2(x)

Matrix notes



For matrix A with (mxn) and matrix B with (mxk), then C=[A B] is a new matrix with (mx(n+k)) .

For matrix A with (mxn) and matrix B with (kxn), then C=[A;B] is a new matrix with ((m+k)xn) .

Example

What is the out of following

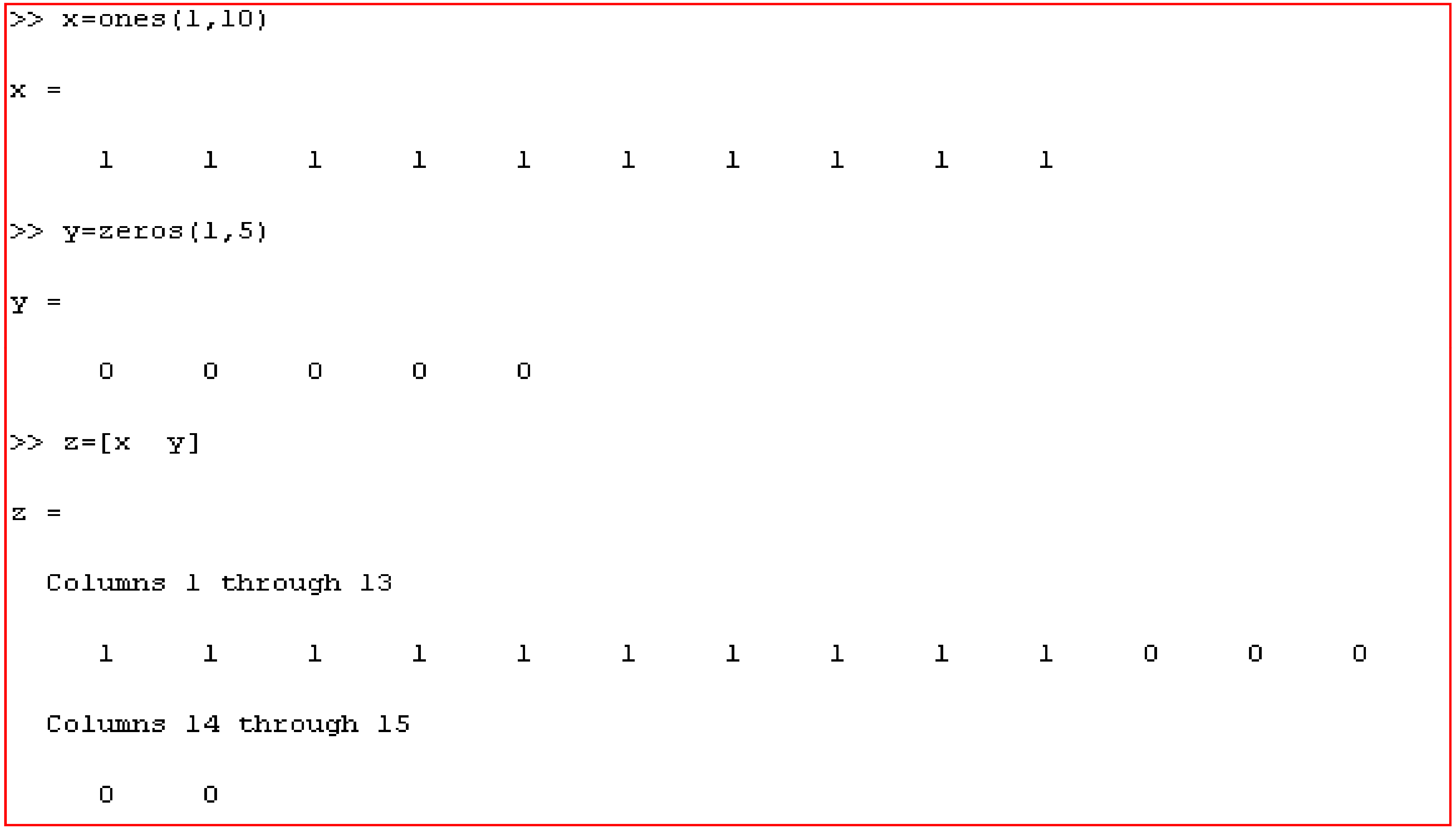
1.

```
x=ones(1,10)
y=zeros(1,5)
z=[x  y]
```

2.

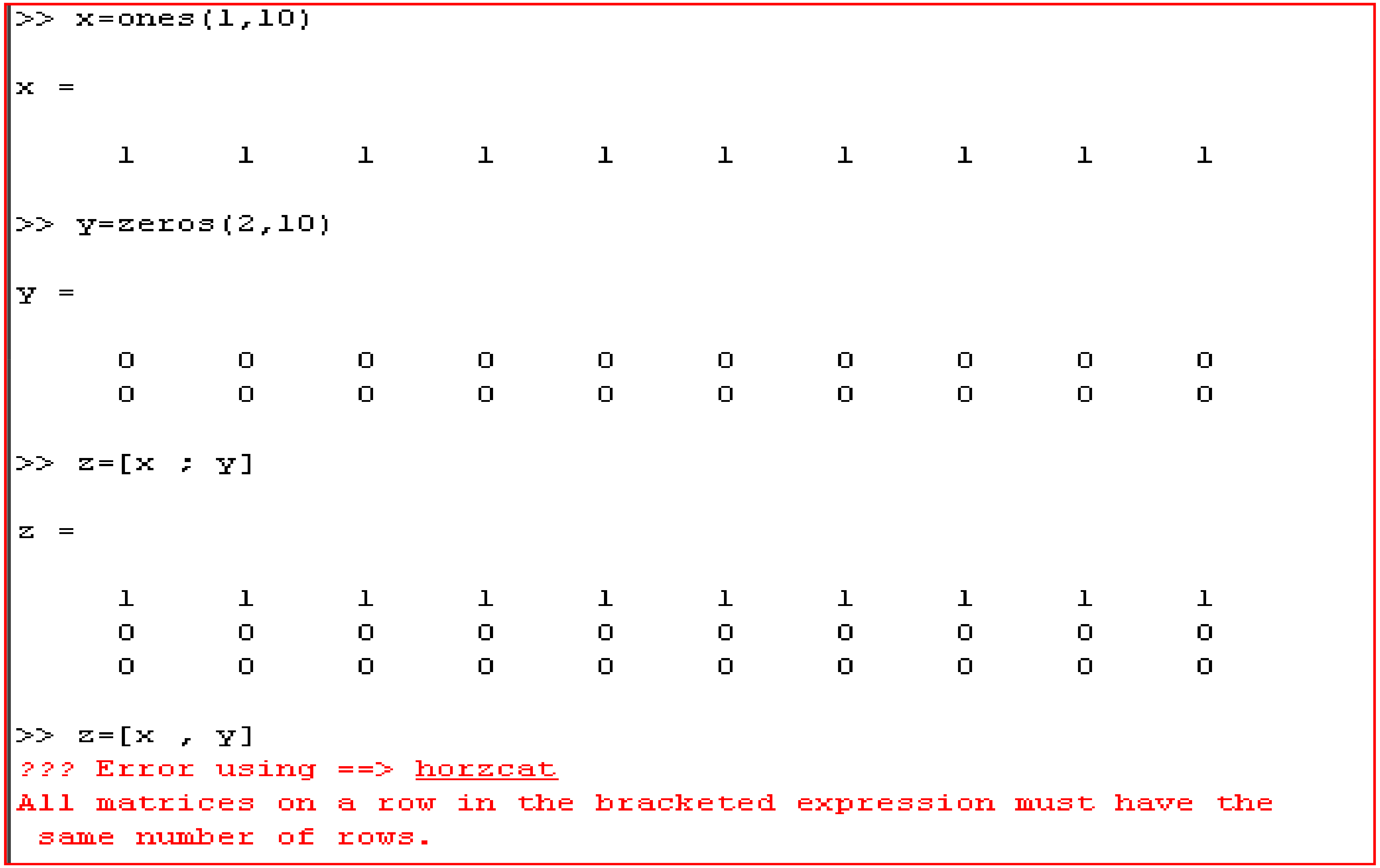
```
x=ones(1,10)
y=zeros(2,10)
z=[x ; y]
```

First output



```
>> x=ones(1,10)

x =

     1     1     1     1     1     1     1     1     1     1

>> y=zeros(1,5)

y =

     0     0     0     0     0

>> z=[x  y]

z =

  Columns 1 through 13

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

  Columns 14 through 15

     0     0
```

Second output



```
>> x=ones(1,10)

x =

     1     1     1     1     1     1     1     1     1     1

>> y=zeros(2,10)

y =

     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0

>> z=[x ; y]

z =

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

>> z=[x , y]
??? Error using ==> horzcat
All matrices on a row in the bracketed expression must have the
 same number of rows.
```

Operators (arithmetic)

+	addition
-	subtraction
*	multiplication
/	division
^	power
.*	element-by-element multiplication
./	element-by-element div
.^	element-by-element power
‘	transpose

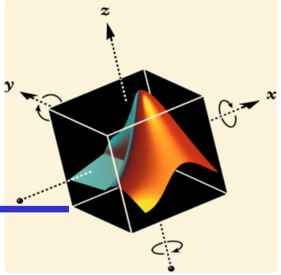
Operators (relational, logical)

==	equal
~=	not equal
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal

&	AND
|	OR
~	NOT

pi	3.14159265…
j	imaginary unit, $\sqrt{-1}$
i	same as j

Generating Vectors from functions



- zeros(M,N) MxN matrix of zeros

```
x = zeros(1,3)
x =
   0     0     0
```

- ones(M,N) MxN matrix of ones

```
x = ones(1,3)
x =
   1     1     1
```

- rand(M,N) MxN matrix of uniformly distributed random numbers on (0,1)

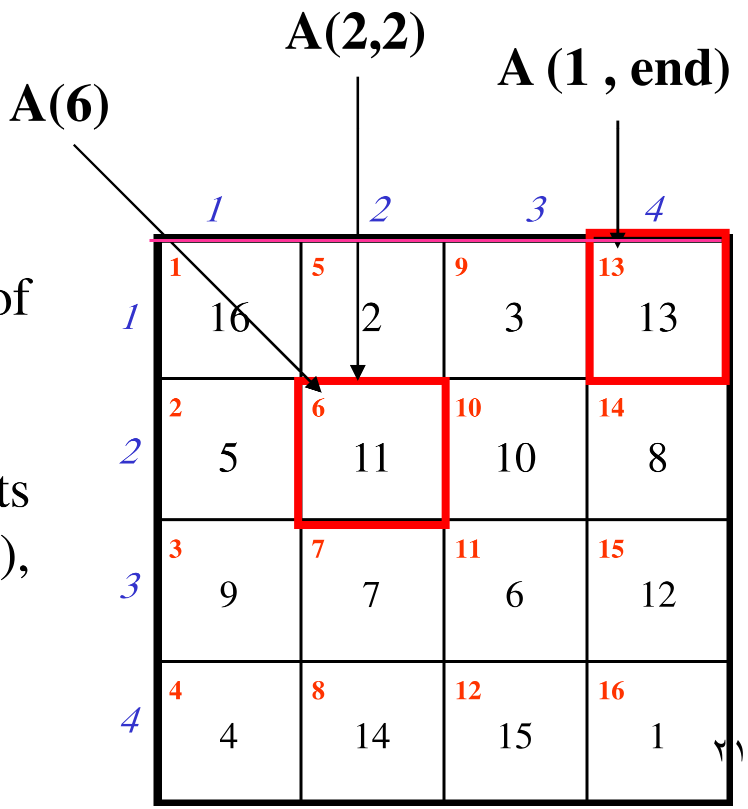
```
x = rand(1,3)
x =
    0.9501   0.2311   0.6068
```

Example

```
>>A = [16  3  2  13;  5   10  11  8;  9  6  7  12;  4  15  14  1];
```

Define the following matrix:

$$
A = \begin{array}{cccc}
16 & 3 & 2 & 13 \\
5 & 10 & 11 & 8 \\
9 & 6 & 7 & 12 \\
4 & 15 & 14 & 1
\end{array}
$$

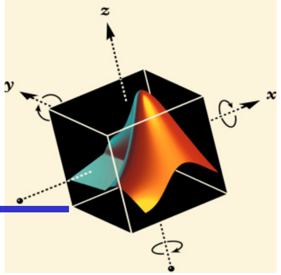


	1	2	3	4
1	1 16	5 2	9 3	13 13
2	2 5	6 11	10 10	14 8
3	3 9	7 7	11 6	15 12
4	4 4	8 14	12 15	16 1

Notes

- The element in row i and column j of matrix A is denoted by A(i,j) *(row-column subscript).*
- It is also possible to refer to the elements of a matrix with a single subscript, A(k), *(element colum wise index).*
- *"end"* specifies maximum index value.

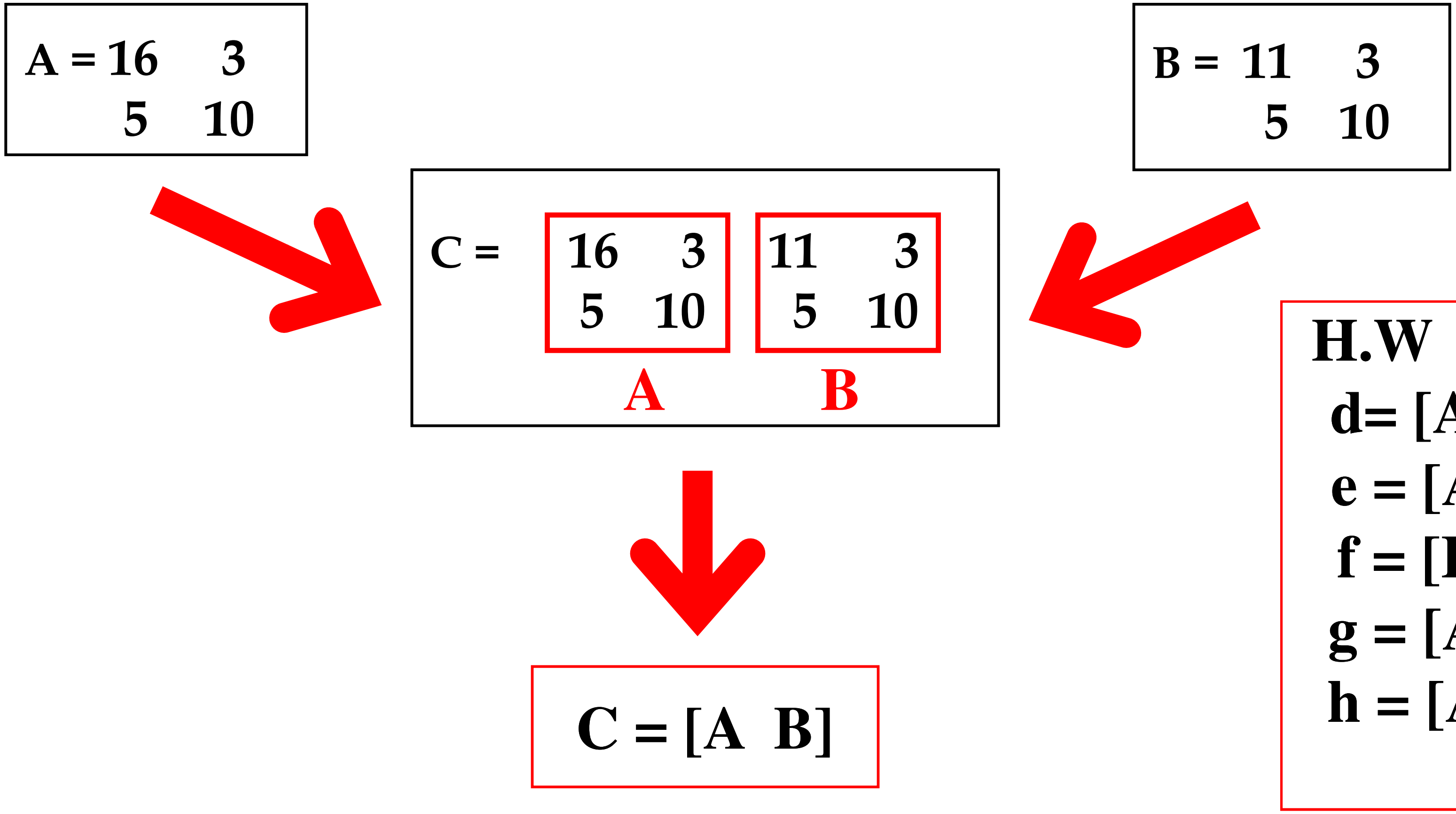
Example



Define the following matrix:
1 – zeros matrix with dimensions (2x4)
2 – five’s matrix with dimensions (2x2)

```
>> Z = zeros (2,4)
      Z =
              0         0         0         0
              0         0         0         0

>> F = 5*ones(2,2)
      F =
              5         5
              5         5
```

Concatenation



$$A = \begin{matrix} 16 & 3 \\ 5 & 10 \end{matrix}$$

$$B = \begin{matrix} 11 & 3 \\ 5 & 10 \end{matrix}$$

$$C = \underbrace{\boxed{\begin{matrix} 16 & 3 \\ 5 & 10 \end{matrix}}}_{A}\ \underbrace{\boxed{\begin{matrix} 11 & 3 \\ 5 & 10 \end{matrix}}}_{B}$$

C = [A B]

H.W

d= [A , B]

e = [A ; B]

f = [B A]

g = [A+ B]

h = [A - B]

Example

```
>>A = [1  2 ; 3  4];
>>B = [A  A+10;  A+20   zeros(2) ]
```

```
B =
        1       2       11      12
        3       4       13      14
        21      22      0       0
        23      24      0       0
```

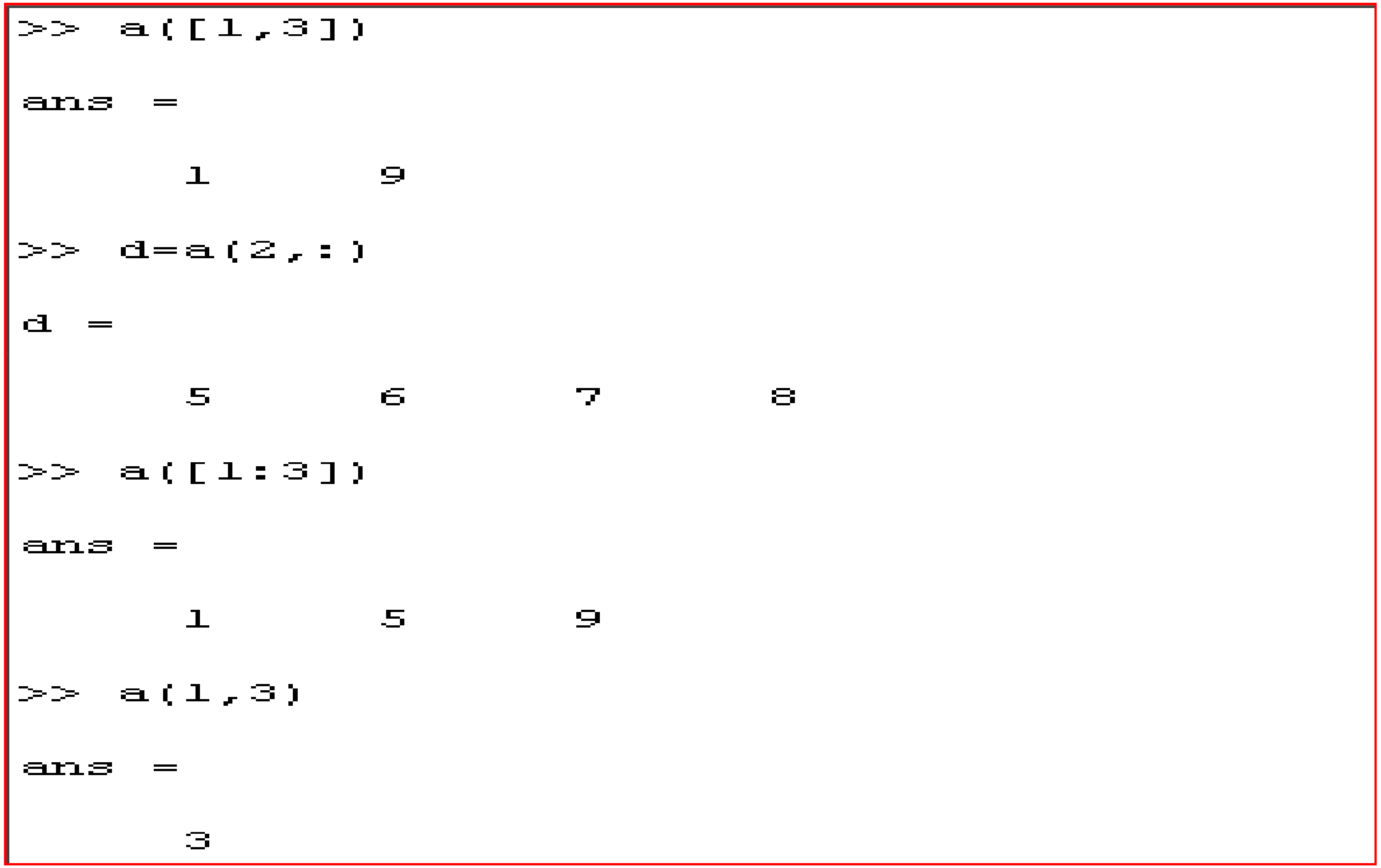
Colon Operator

j:k	same as [j,j+1,...,k] is empty if j > k
j:i:k	same as [j,j+i,j+2i, ..,k] is empty if i > 0 and j > k or if i < 0 and j < k
A(:,j)	is the j-th column of A
A(i,:)	is the i-th row of A
A(:,:)	equivalent to the same as A.
A(j:k)	is A(j), A(j+1),...,A(k)

Exercise

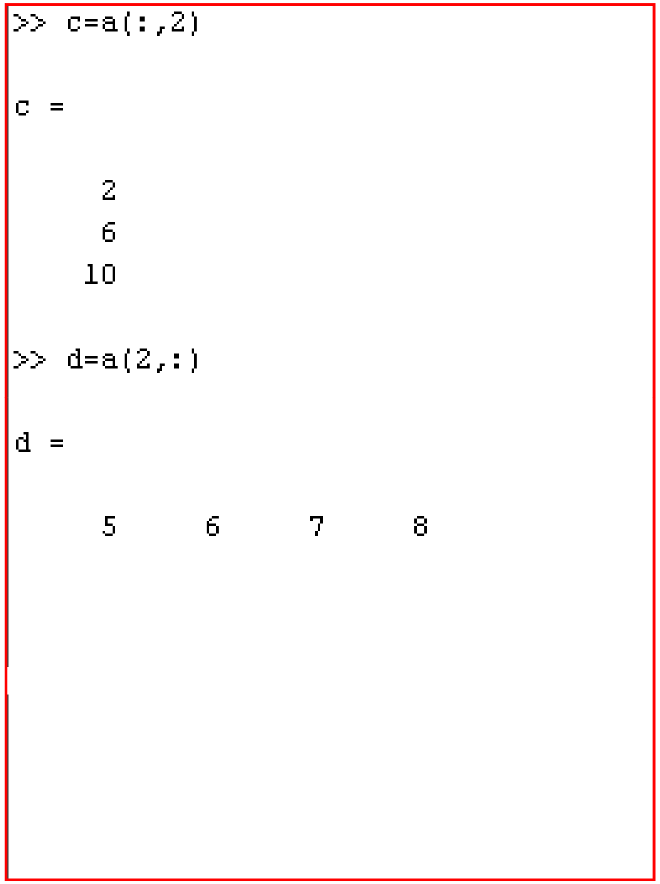
- a=[1 2 3 4 ; 5 6 7 8 ; 9 10 11 12]
- a([1,3])
- d=a(2,:)
- a([1:3])
- a(1,3)
- c=a(:,2)
- d=a(2,:)

Results



```
>>  a([1,3])

ans =

     1     9

>>  d=a(2,:)

d =

     5     6     7     8

>>  a([1:3])

ans =

     1     5     9

>>  a(1,3)

ans =

     3
```

Results



```
>> c=a(:,2)

c =

     2
     6
    10

>> d=a(2,:)

d =

     5     6     7     8
```

Operators

[ ] concatenation

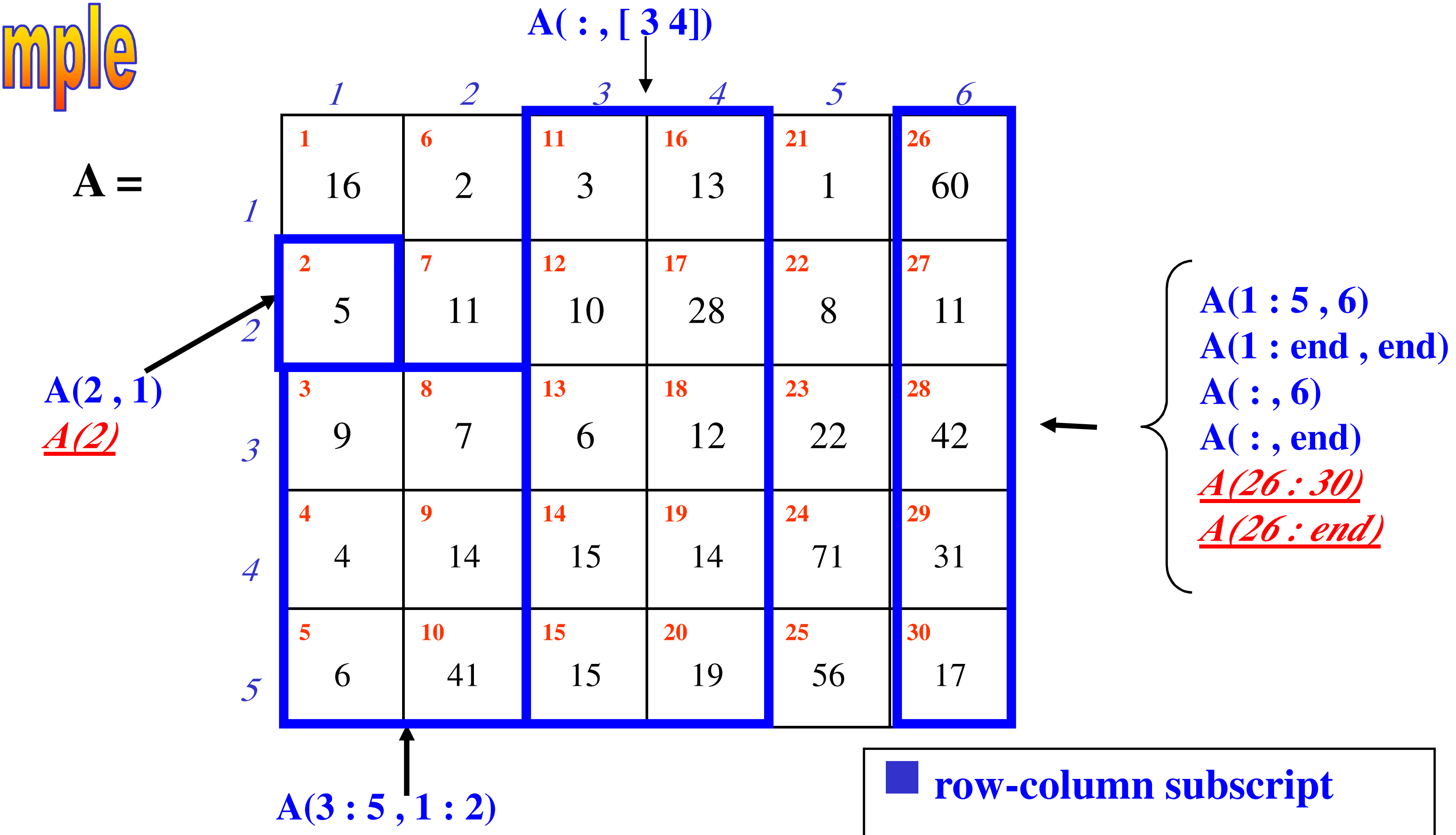
```
x = [ zeros(1,3) ones(1,2) ]
x =
     0   0   0   1   1
```

( ) subscription

```
x = [ 1 3 5 7 9]
x =
     1   3   5   7   9

y = x(2)
y =
     3
y = x(2:4)
y =
     3   5   7
```

Indexing

Example



A =

	1	2	3	4	5	6
1	1 16	6 2	11 3	16 13	21 1	26 60
2	2 5	7 11	12 10	17 28	22 8	27 11
3	3 9	8 7	13 6	18 12	23 22	28 42
4	4 4	9 14	14 15	19 14	24 71	29 31
5	5 6	10 41	15 15	20 19	25 56	30 17

A(: , [3 4])

A(2 , 1)
A(2)

A(3 : 5 , 1 : 2)

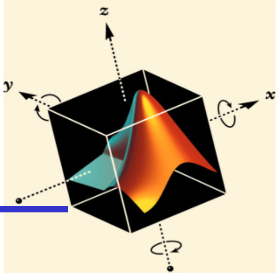
A(1 : 5 , 6)
A(1 : end , end)
A(: , 6)
A(: , end)
A(26 : 30)
A(26 : end)

■ row-column subscript

□ *element wise index*

Delete rows and columns from a matrix using colons and just a pair of square brackets.

Magic square



Example

```
B = magic (4);
B( : , 3) = [ ]
```

```
B =
       16      2     ~~3~~     13
        5     11    ~~10~~      8
        9      7     ~~6~~     12
        4     14    ~~15~~      1
```

```
B =
       16      2     13
        5     11      8
        9      7     12
        4     14      1
```

If you delete a single element from a matrix, the result isn’t a matrix anymore. So, expressions like **B(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. So

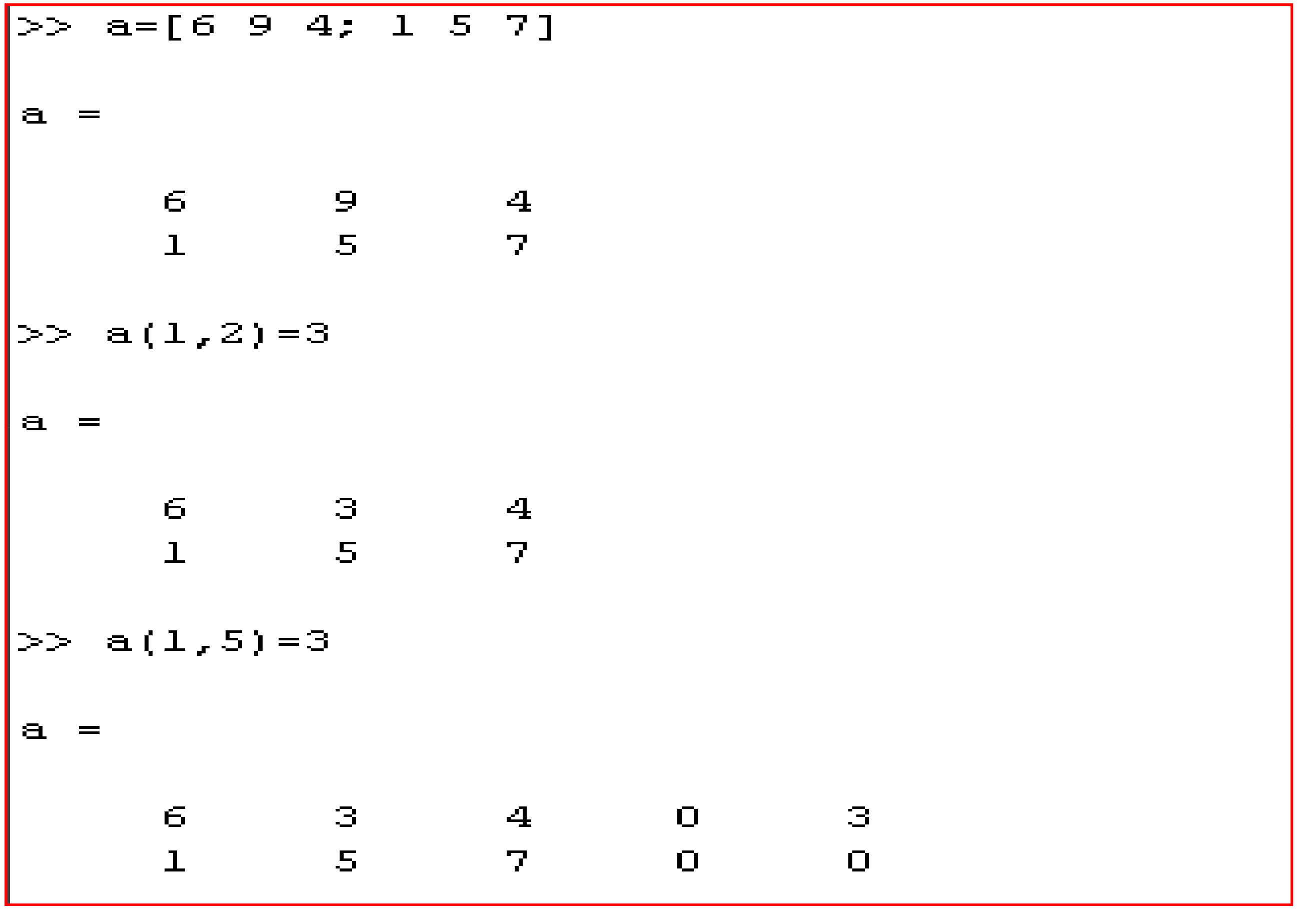
B(2:2:10) = [ ]   results in

```
B =
    16     9     2     7     13     12     1
```

Examples

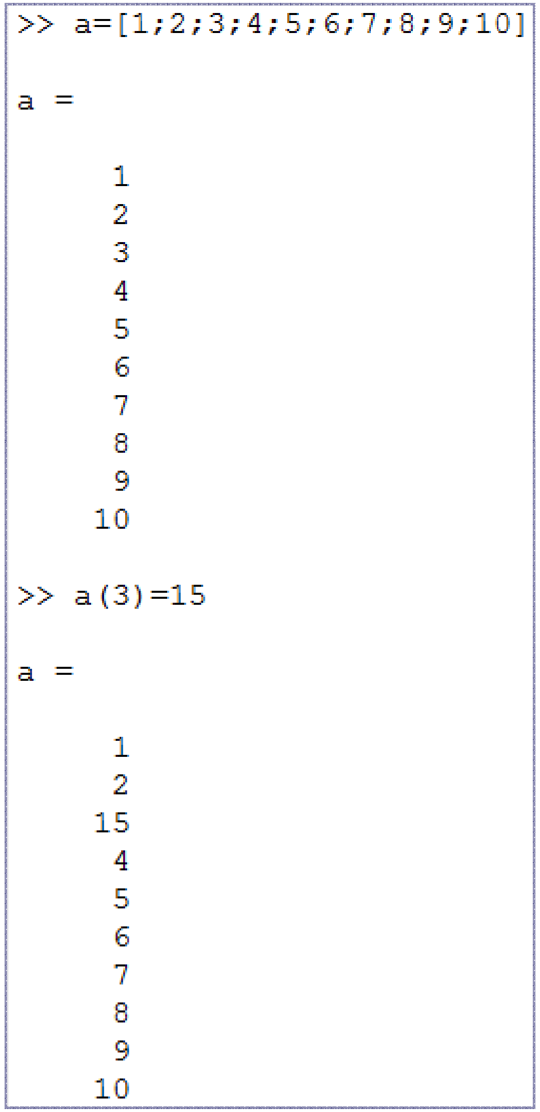
```
# Suppose a=[6 9 4; 1 5 7]
What is the difference between following
a(1,2)=3
a(1,5)=3

# let a=[1;2;3;4;5;6;7;8;9;10]
      a(3)=15
      a(1:5)=zeros(1,5)
      a(1:5)=zeros(1,3)
      a(10)=[]
      a(6:8)=[]
      a(6:10)=[]
```

Results

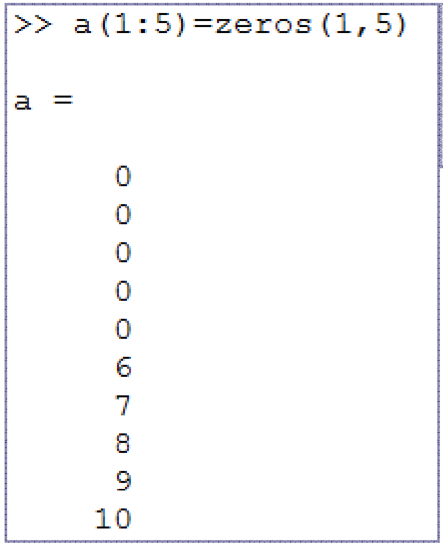


```
>>  a=[6  9  4;  1  5  7]

a  =


         6          9          4
         1          5          7

>>  a(1,2)=3

a  =


         6          3          4
         1          5          7

>>  a(1,5)=3

a  =


         6          3          4          0          3
         1          5          7          0          0
```

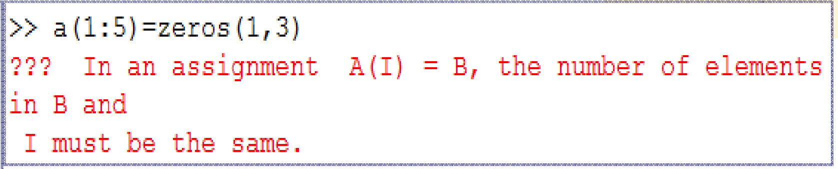
Results



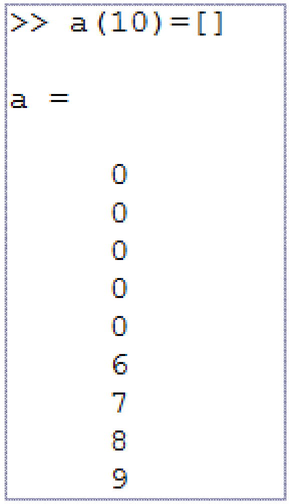
```
>> a=[1;2;3;4;5;6;7;8;9;10]

a =

     1
     2
     3
     4
     5
     6
     7
     8
     9
    10

>> a(3)=15

a =

     1
     2
    15
     4
     5
     6
     7
     8
     9
    10
```



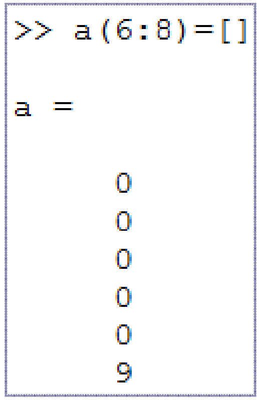
```
>> a(1:5)=zeros(1,5)

a =

     0
     0
     0
     0
     0
     6
     7
     8
     9
    10
```



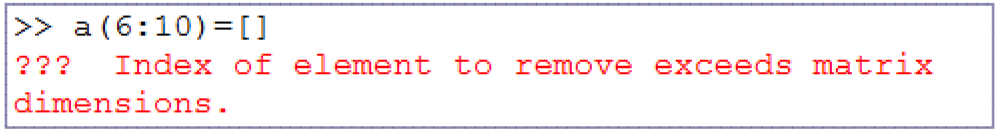
```
>> a(1:5)=zeros(1,3)
???  In an assignment  A(I) = B, the number of elements
in B and
 I must be the same.
```



```
>> a(10)=[]

a =

     0
     0
     0
     0
     0
     6
     7
     8
     9
```



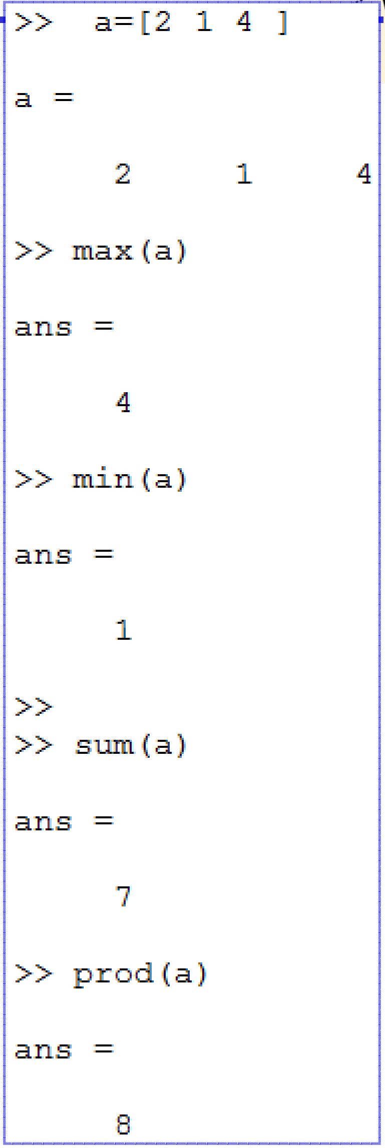
```
>> a(6:8)=[]

a =

     0
     0
     0
     0
     0
     9
```



```
>> a(6:10)=[]
???  Index of element to remove exceeds matrix
dimensions.
```

Commands on Vectors

```
# Suppose a=[2 1 4 ] Find:
max(a)
min(a)
sum(a)
prod(a)
```



```
>>  a=[2 1 4 ]

a =

     2     1     4

>> max(a)

ans =

     4

>> min(a)

ans =

     1

>>
>> sum(a)

ans =

     7

>> prod(a)

ans =

     8
```

Commands on Matrix

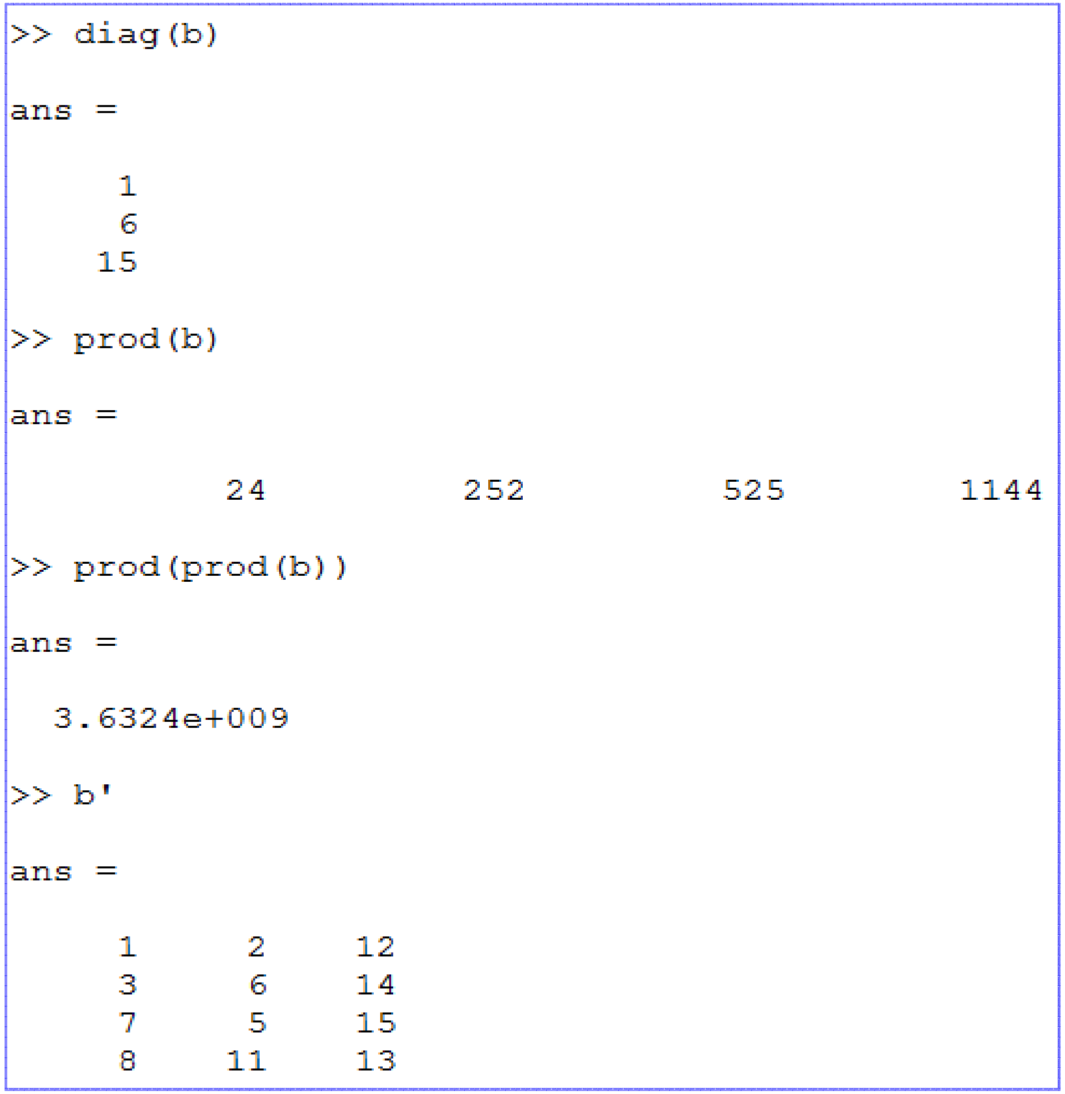
```
b=[1 3 7 8; 2 6 5 11; 12 14 15 13]
max(b)
min(b)
sum(b)
mean(b)
diag(b)
prod(b)
 b'
(b ') '
b(2,5)=42
b(4,1:4)=[31 54 13 11]
b(1:3,1:2)=0
b(3,:)=[]
b(2,end)
```

Results

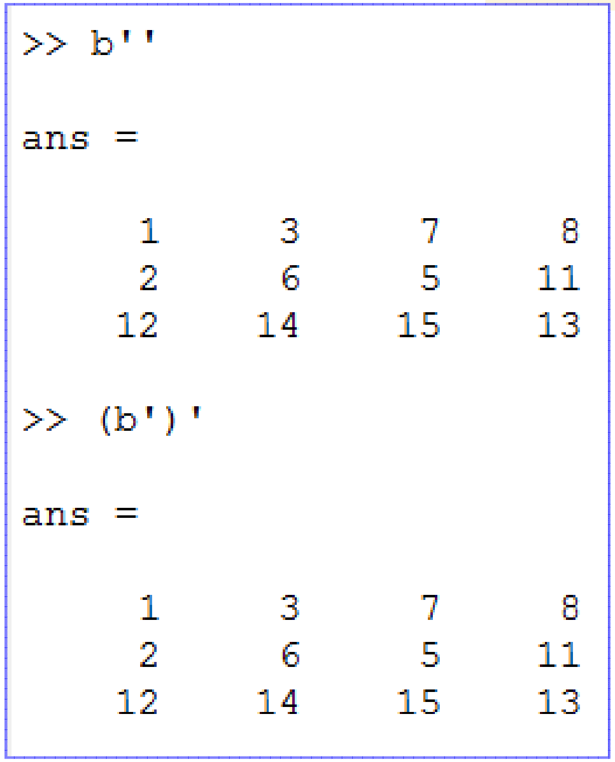
```
>> b=[1 3 7 8; 2 6 5 11; 12 14 15 13]

b =

     1     3     7     8
     2     6     5    11
    12    14    15    13

>> max(b)

ans =

    12    14    15    13

>> max(max(b))

ans =

    15

>> min(b)

ans =

     1     3     5     8

>> min(min(b))

ans =

     1
```

```
>> sum(b)

ans =

    15    23    27    32

>> sum(sum(b))

ans =

    97

>> mean(b)

ans =

    5.0000    7.6667    9.0000   10.6667

>> mean(mean(b))

ans =

    8.0833

>> mean2(b)

ans =

    8.0833
```

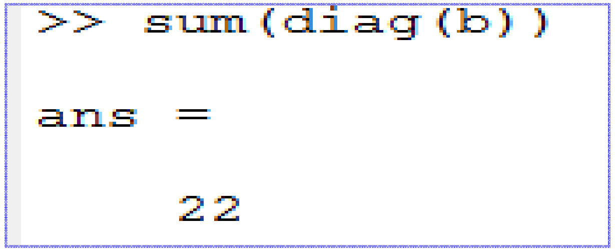
Results



```
>> diag(b)

ans =

     1
     6
    15

>> prod(b)

ans =

          24          252          525         1144

>> prod(prod(b))

ans =

  3.6324e+009

>> b'

ans =

     1     2    12
     3     6    14
     7     5    15
     8    11    13
```

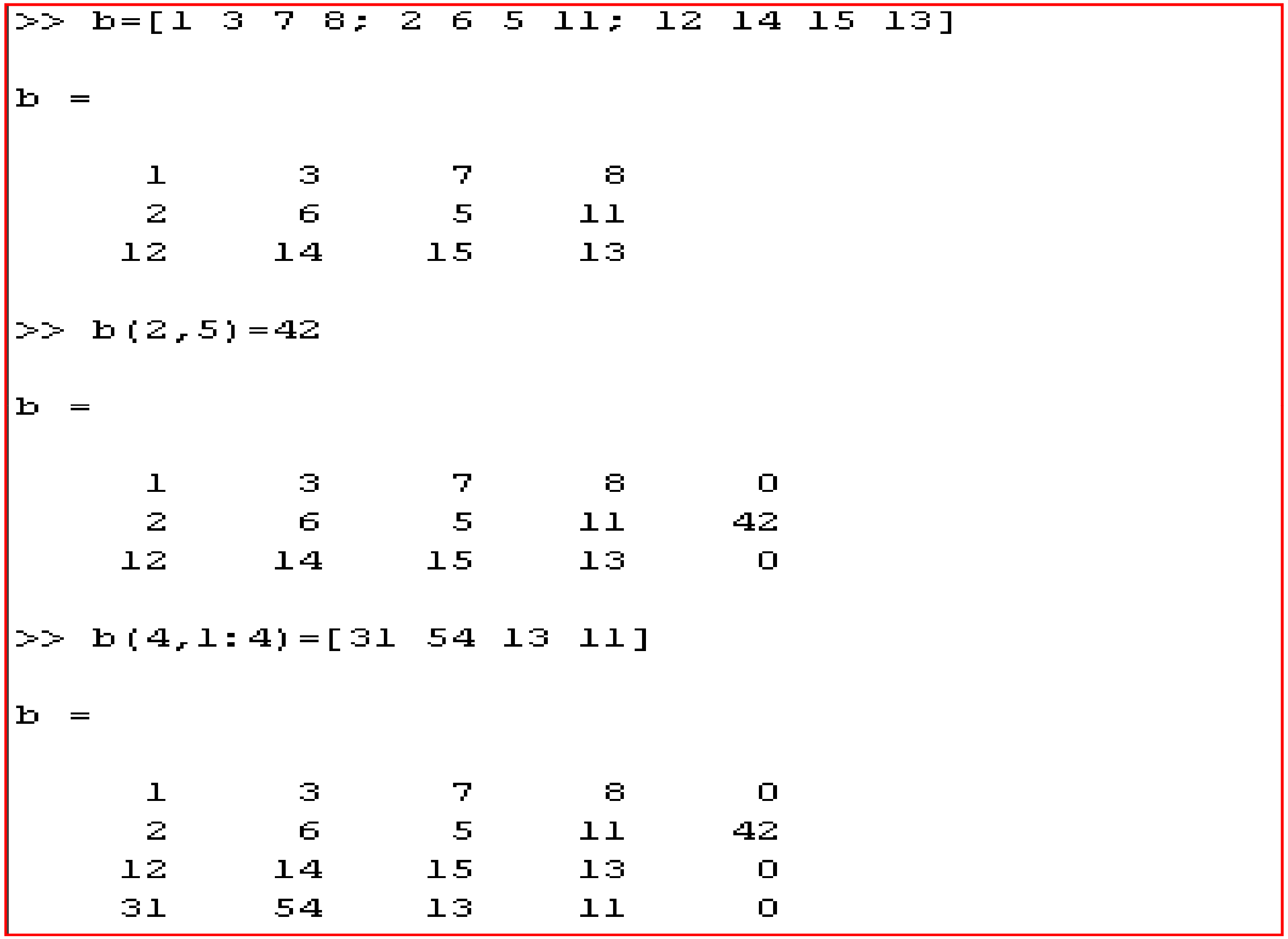


```
>> b''

ans =

     1     3     7     8
     2     6     5    11
    12    14    15    13

>> (b')'

ans =

     1     3     7     8
     2     6     5    11
    12    14    15    13
```



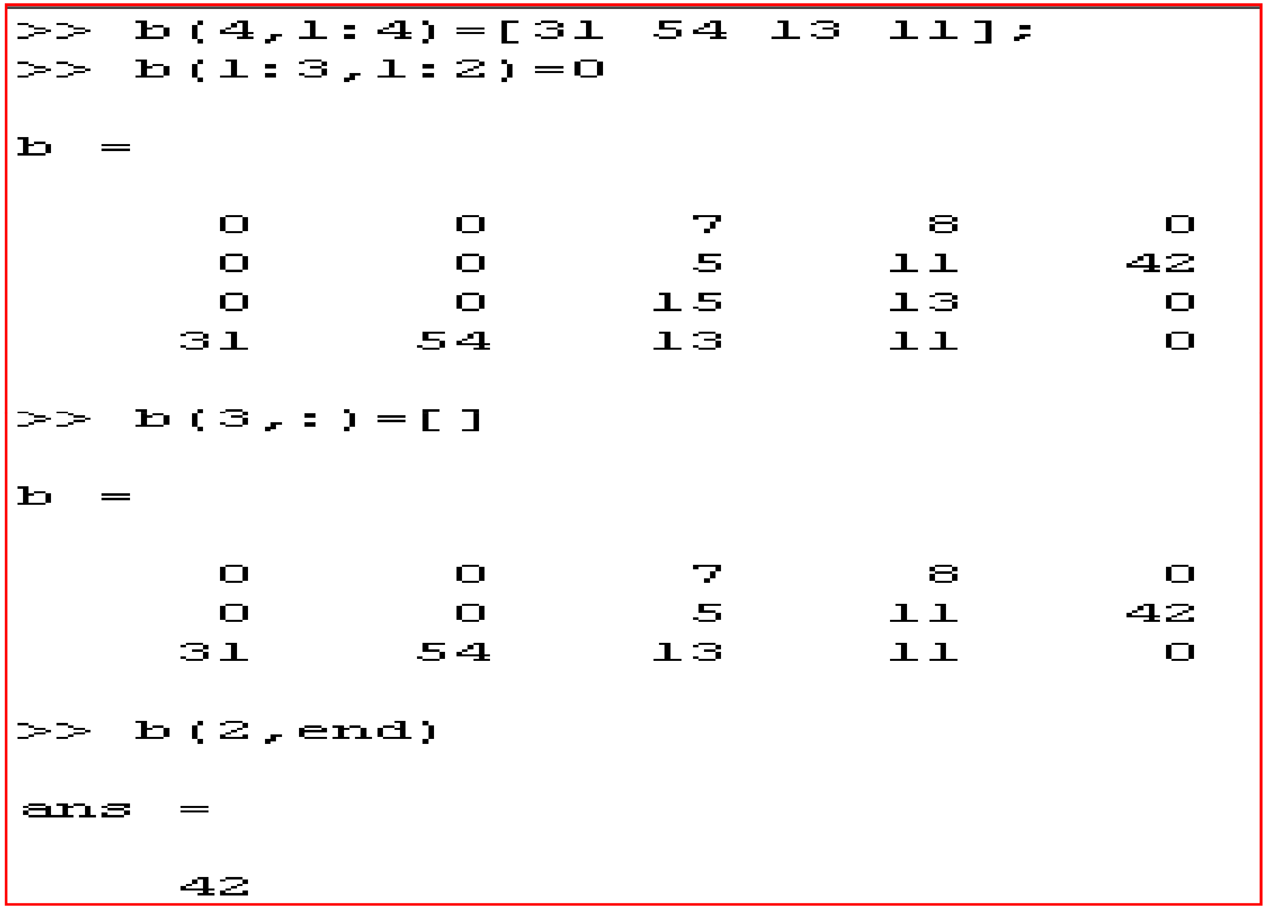
```
>> sum(diag(b))

ans =

    22
```

Results



```
>> b=[1 3 7 8; 2 6 5 11; 12 14 15 13]

b =

     1     3     7     8
     2     6     5    11
    12    14    15    13

>> b(2,5)=42

b =

     1     3     7     8     0
     2     6     5    11    42
    12    14    15    13     0

>> b(4,1:4)=[31 54 13 11]

b =

     1     3     7     8     0
     2     6     5    11    42
    12    14    15    13     0
    31    54    13    11     0
```

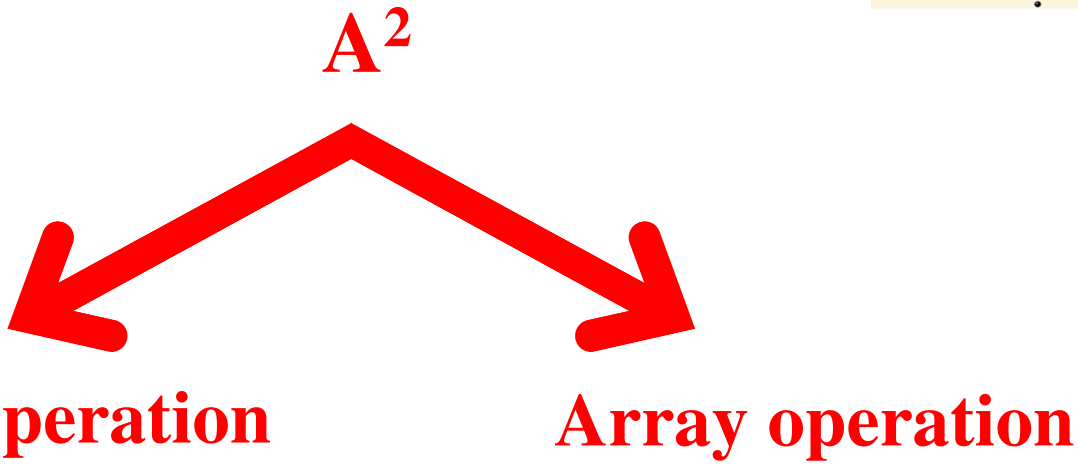
Results



```
>> b(4,1:4)=[31 54 13 11];
>> b(1:3,1:2)=0

b =

     0     0     7     8     0
     0     0     5    11    42
     0     0    15    13     0
    31    54    13    11     0

>> b(3,:)=[]

b =

     0     0     7     8     0
     0     0     5    11    42
    31    54    13    11     0

>> b(2,end)

ans =

    42
```

Arithmetic Operators

$$A = \begin{matrix} 3 & 4 & 5 \\ 1 & 4 & 0 \\ 1 & 8 & 2 \end{matrix}$$



A^2

Matrix operation

Array operation

$$\begin{matrix} 18 & 68 & 25 \\ 7 & 20 & 5 \\ 13 & 52 & 9 \end{matrix}$$

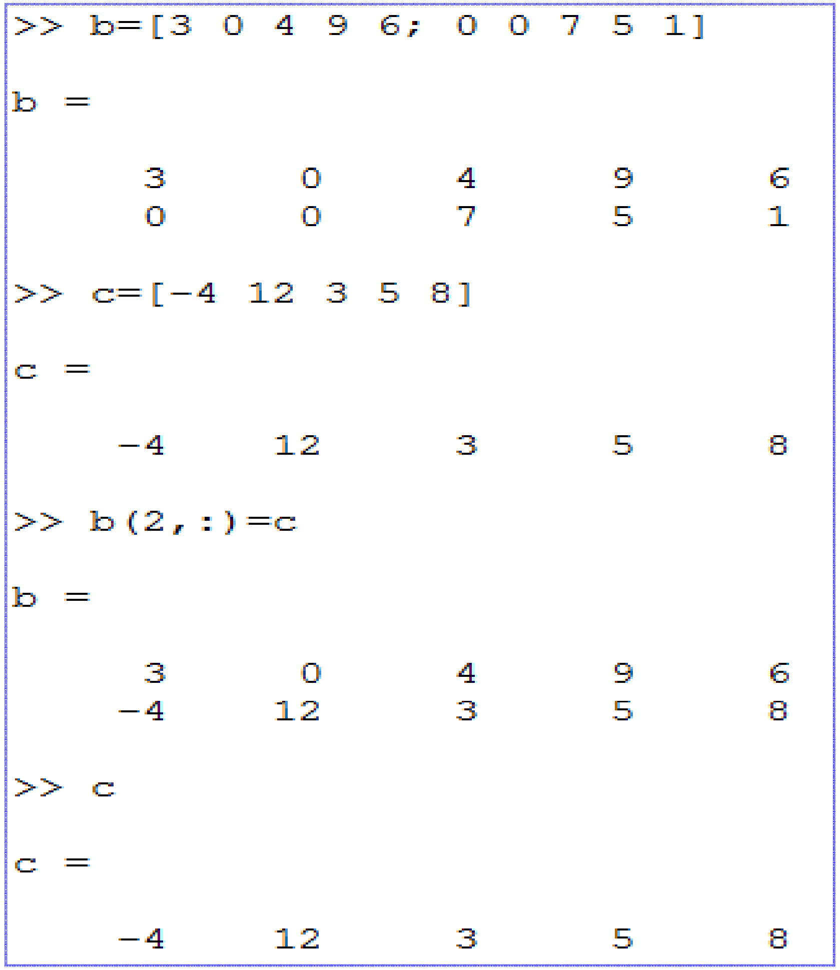
$$\begin{matrix} 9 & 16 & 25 \\ 1 & 16 & 0 \\ 1 & 64 & 4 \end{matrix}$$

Exercise

- Let b=[3 0 4 9 6; 0 0 7 5 1]
- Let c=[-4 12 3 5 8]

1. What is the output of following command b(2,:)=c ?
2. Which of following change ‘b’ or ‘c’?

Results



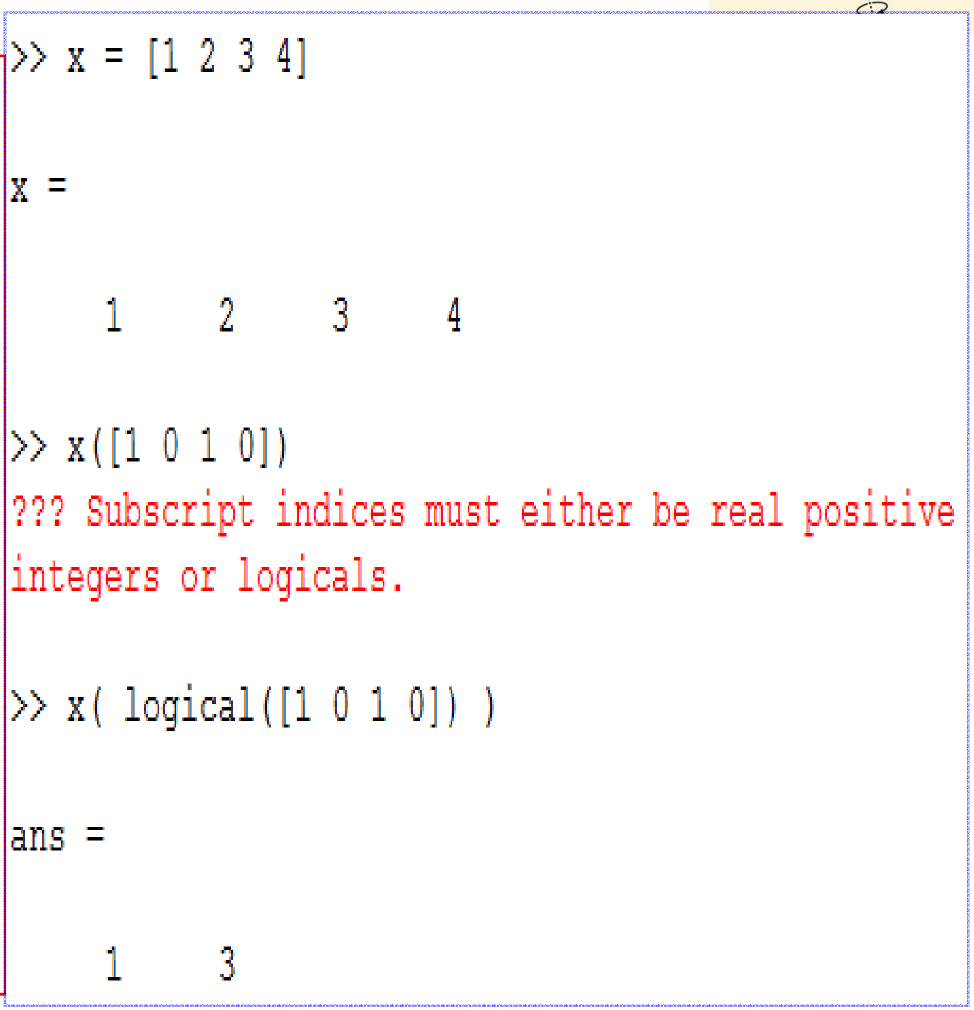
```
>> b=[3 0 4 9 6; 0 0 7 5 1]

b =

     3     0     4     9     6
     0     0     7     5     1

>> c=[-4 12 3 5 8]

c =

    -4    12     3     5     8

>> b(2,:)=c

b =

     3     0     4     9     6
    -4    12     3     5     8

>> c

c =

    -4    12     3     5     8
```

Logical

```
# Let x = [1 2 3 4]
```

Explain the difference between the result of the following two code lines:

```
x([1 0 1 0])
x( logical([1 0 1 0]) )
```



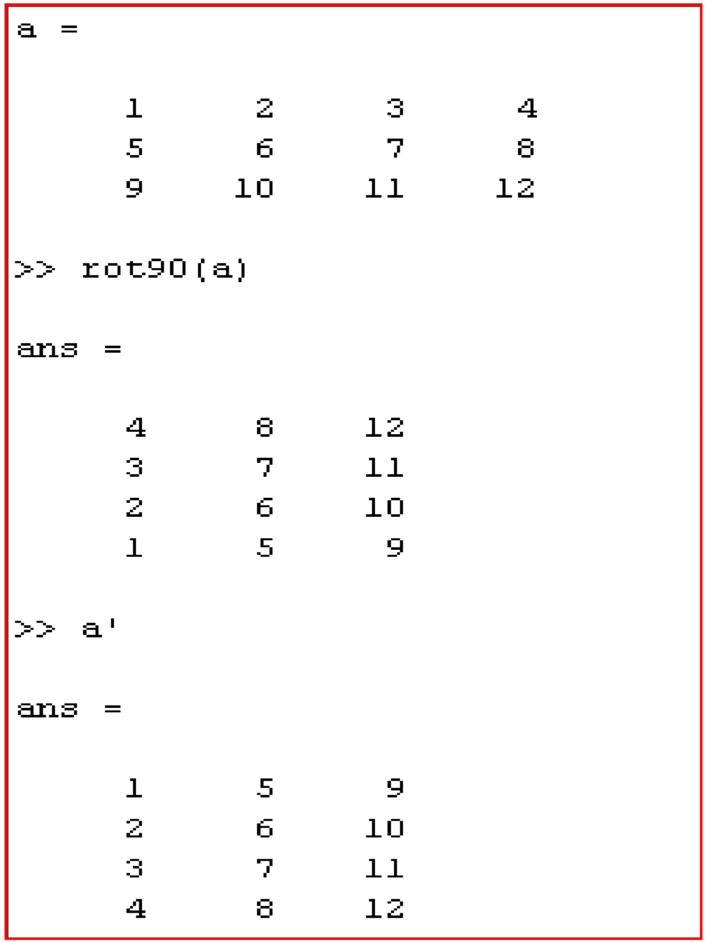
```
>> x = [1 2 3 4]

x =

     1     2     3     4

>> x([1 0 1 0])
??? Subscript indices must either be real positive
integers or logicals.

>> x( logical([1 0 1 0]) )

ans =

     1     3
```

Rotation

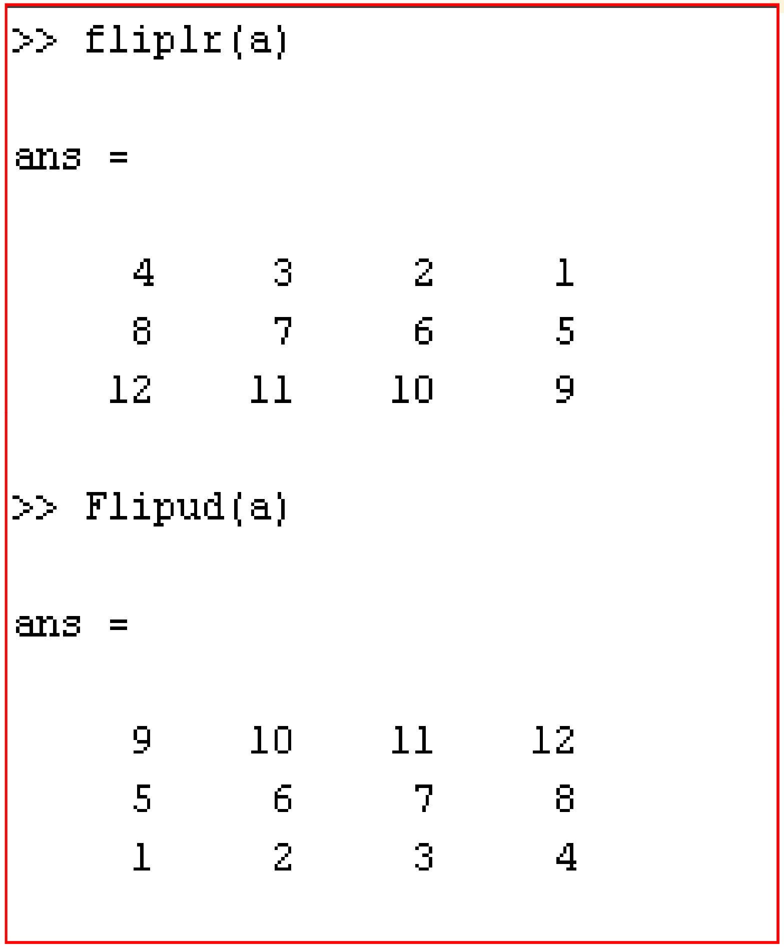
- Let a=[1 2 3;4 5 6;7 8 9]

1. rot90(a)
2. a’
3. fliplr(a)
4. Flipud(a)

Results



```
a =

     1     2     3     4
     5     6     7     8
     9    10    11    12

>> rot90(a)

ans =

     4     8    12
     3     7    11
     2     6    10
     1     5     9

>> a'

ans =

     1     5     9
     2     6    10
     3     7    11
     4     8    12
```



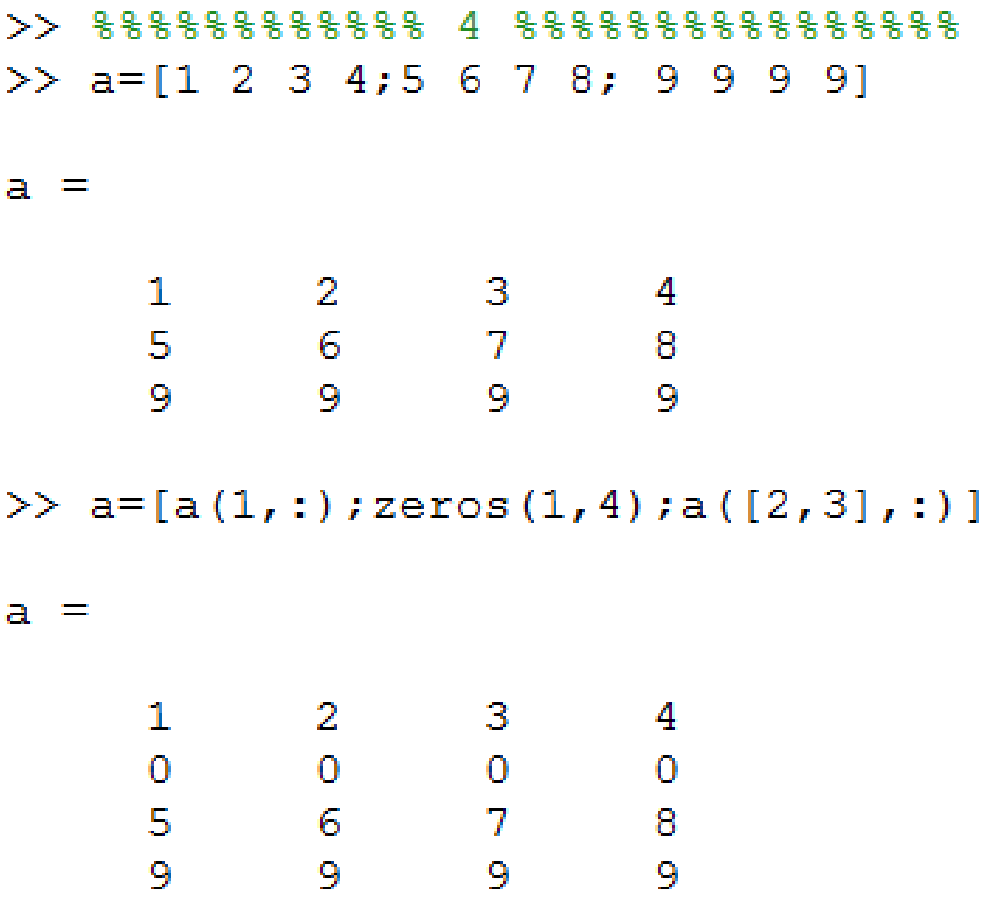
```
>> fliplr(a)

ans =

     4     3     2     1
     8     7     6     5
    12    11    10     9

>> Flipud(a)

ans =

     9    10    11    12
     5     6     7     8
     1     2     3     4
```

Question

```
which of following give the matrix
   1 2 3 4
   0 0 0 0
   5 6 7 8
   9 9 9 9
1. a=[1 2 3 4,zeros(1,4),[5,6,7,8],9*ones(1,4)]
2. b=[5 6 7 8 9 9 9 9]; a=[1 2 3 4;[0 0 0 0]; [b]]
3. a=[1 2 3 4;5 6 7 8; 9 9 9 9]
   a=[a(1,:);zeros(1,4);a((2,3),:)]
4. a=[1 2 3 4;5 6 7 8; 9 9 9 9]
   a=[a(1,:);zeros(1,4);a([2,3],:)]
```

```
>> a=[1 2 3 4,zeros(1,4),[5,6,7,8],9*ones(1,4)]

a =

  Columns 1 through 9

     1     2     3     4     0     0     0     0     5

  Columns 10 through 16

     6     7     8     9     9     9     9

>> %%%%%%%%%%%%%%%2%%%%%%%%%%%%%%%
>> b=[5 6 7 8 9 9 9 9]

b =

     5     6     7     8     9     9     9     9

>> a=[1 2 3 4;[0 0 0 0]; [b]]
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.

>> %%%%%%%%%%% 3  %%%%%%%%%%%%%%%%%
>> a=[1 2 3 4;5 6 7 8; 9 9 9 9]

a =

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

>> a=[a(1,:);zeros(1,4);a((2,3),:)]
??? a=[a(1,:);zeros(1,4);a((2,3),:)]
                             |
Error: Expression or statement is incorrect--possibly
unbalanced (, {, or [.
```

Results



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

a =

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

>> a=[a(1,:);zeros(1,4);a([2,3],:)]

a =

     1     2     3     4
     0     0     0     0
     5     6     7     8
     9     9     9     9
```

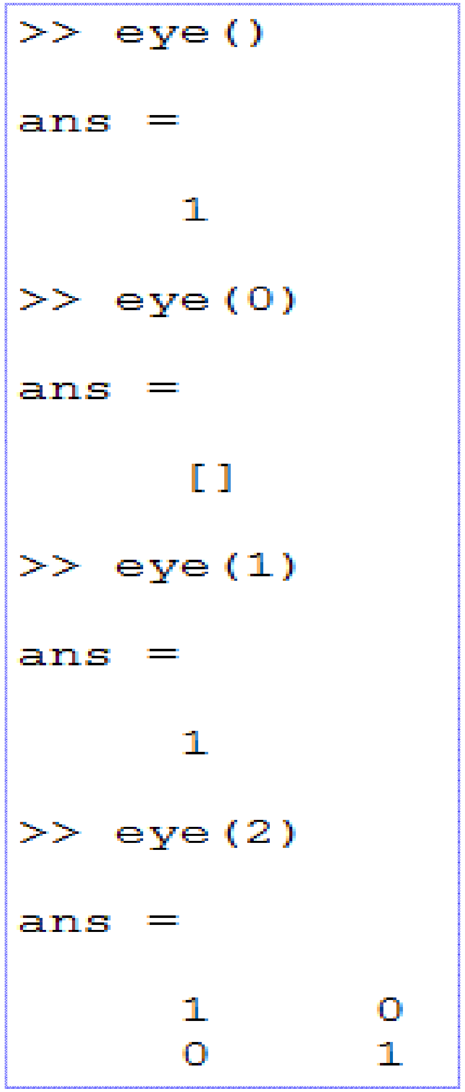
Identity matrix

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

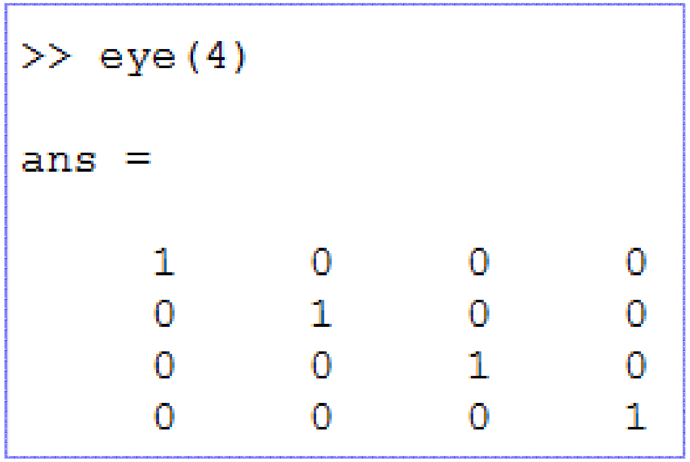
Y = eye(*m*,*n*) or Y = eye([*m n*]) returns an *m*-by-*n* matrix with 1's on the diagonal and 0's elsewhere.

The size inputs *m* and *n* should be nonnegative integers. Negative integers are treated as 0.

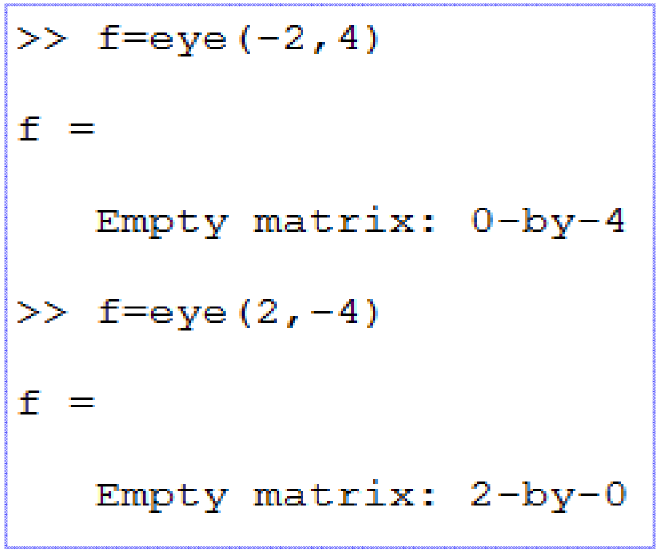
Identity matrix



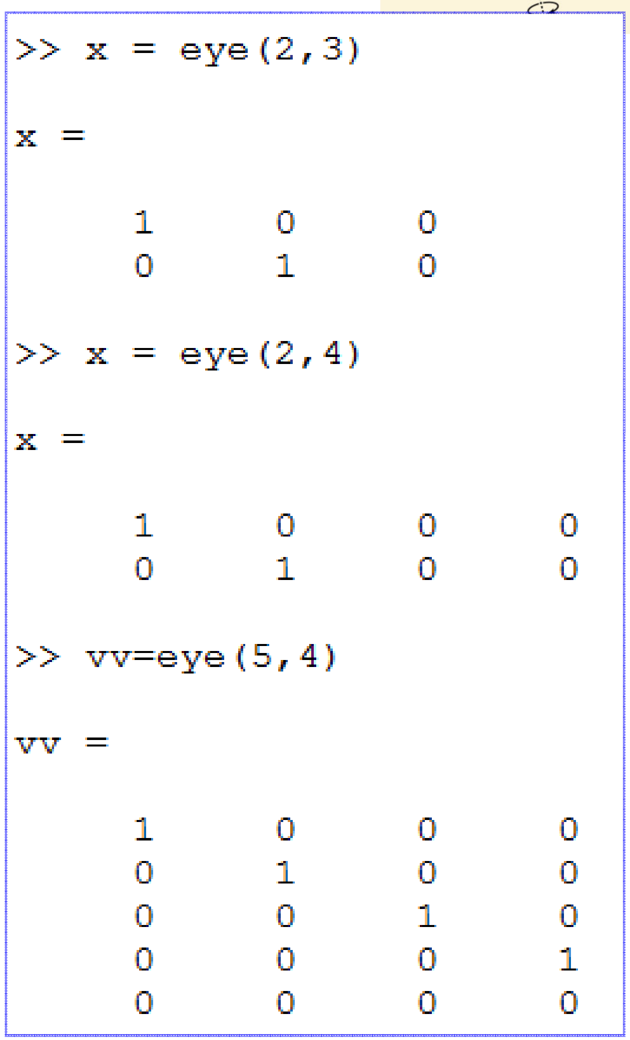
```
>>  eye()

ans =

     1

>>  eye(0)

ans =

     []

>>  eye(1)

ans =

     1

>>  eye(2)

ans =

     1     0
     0     1
```



```
>> eye(4)

ans =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```



```
>>  f=eye(-2,4)

f =

   Empty matrix: 0-by-4

>>  f=eye(2,-4)

f =

   Empty matrix: 2-by-0
```



```
>> x = eye(2,3)

x =

     1     0     0
     0     1     0

>> x = eye(2,4)

x =

     1     0     0     0
     0     1     0     0

>> vv=eye(5,4)

vv =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
     0     0     0     0
```

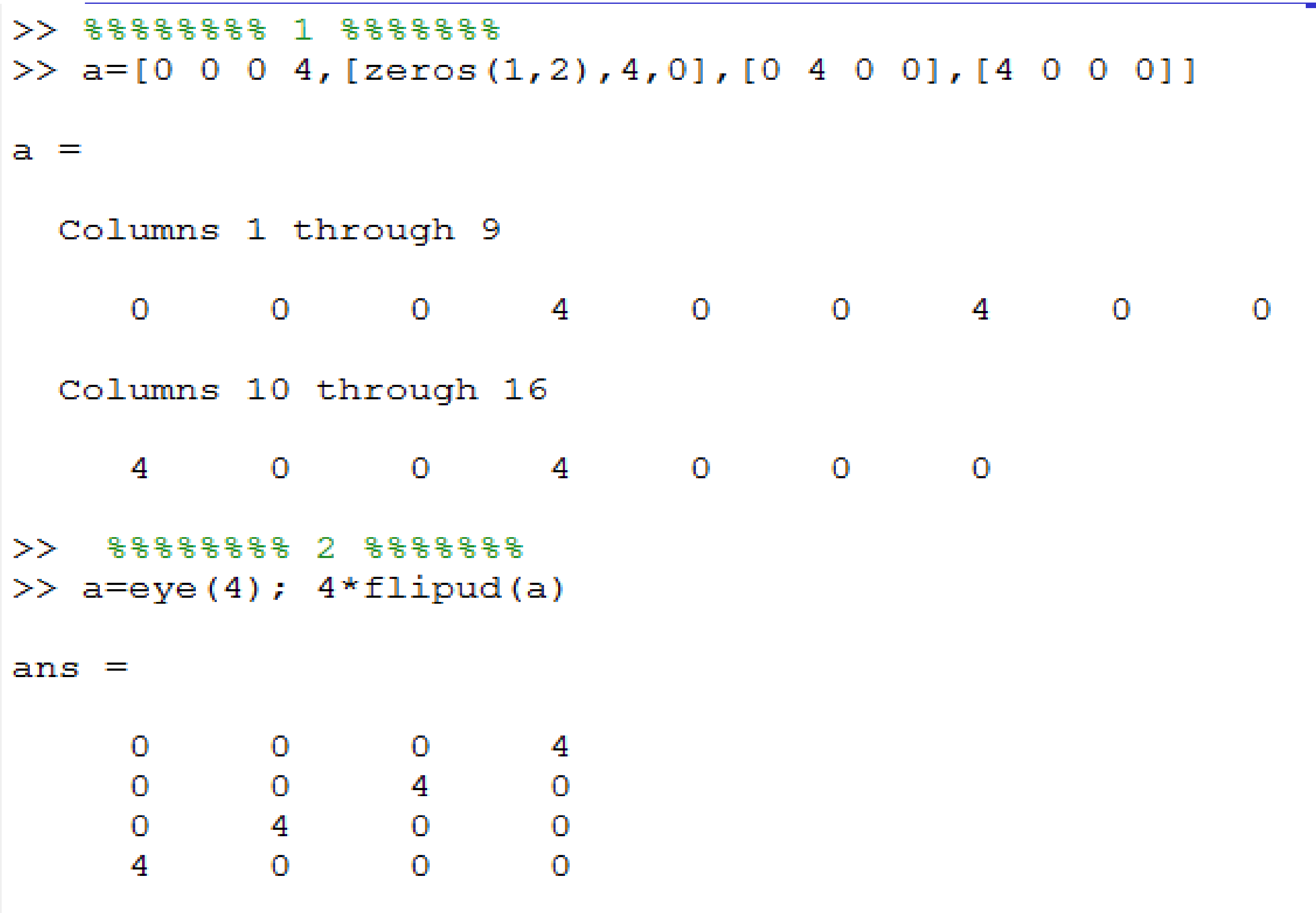
Question

which of following give the matrix

```
    0  0  0  4
    0  0  4  0
    0  4  0  0
    4  0  0  0
```

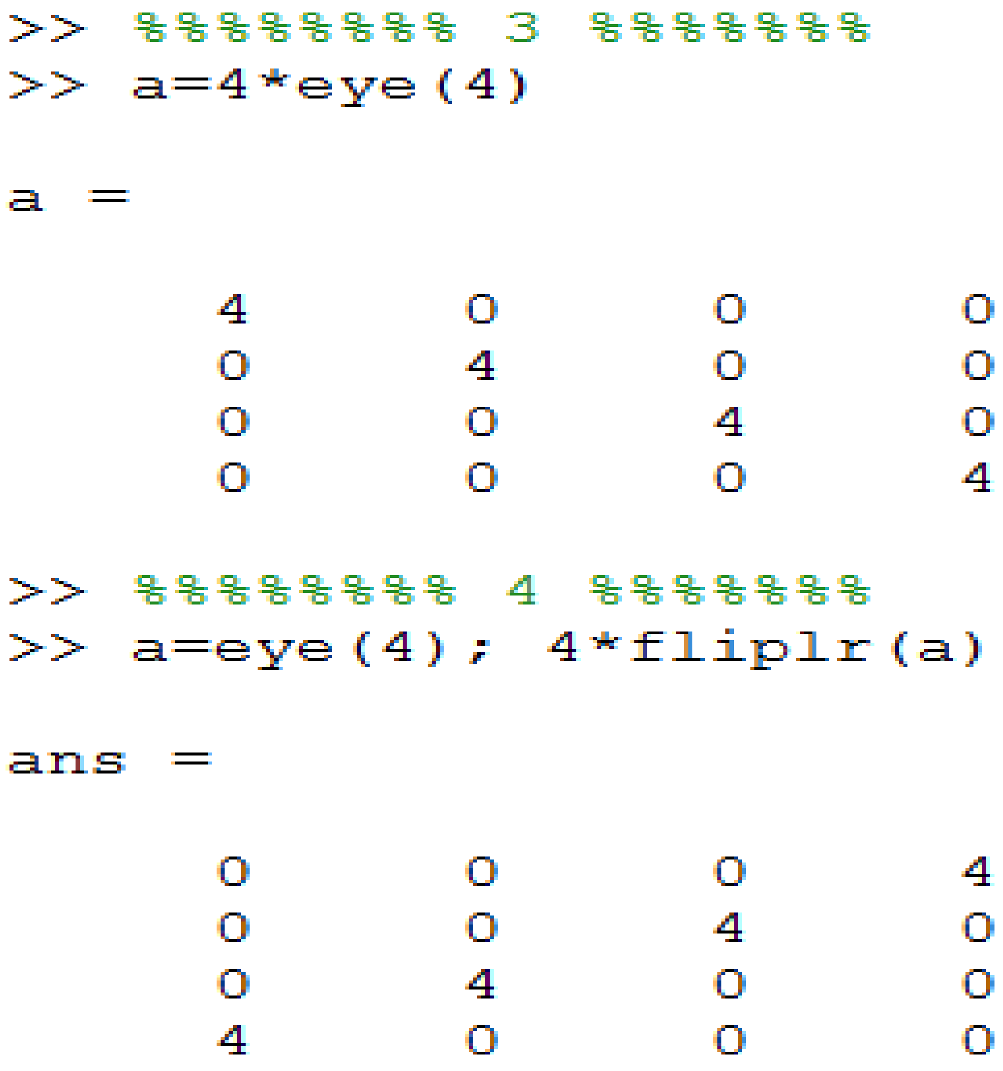
```
1. a=[0 0 0 4,[zeros(1,2),4,0],[0 4 0 0],[4 0 0 0]]
2. a=eye(4); 4*flipud(a)
3. a=4*eye(4)
4. a=eye(4); 4*fliplr(a)
```

Results



```
>> %%%%%%%% 1 %%%%%%%
>> a=[0 0 0 4,[zeros(1,2),4,0],[0 4 0 0],[4 0 0 0]]

a =

  Columns 1 through 9

     0     0     0     4     0     0     4     0     0

  Columns 10 through 16

     4     0     0     4     0     0     0

>>  %%%%%%%% 2 %%%%%%%
>> a=eye(4); 4*flipud(a)

ans =

     0     0     0     4
     0     0     4     0
     0     4     0     0
     4     0     0     0
```

Results



```
>> %%%%%%%% 3 %%%%%%%
>> a=4*eye(4)

a =

     4     0     0     0
     0     4     0     0
     0     0     4     0
     0     0     0     4

>> %%%%%%%% 4 %%%%%%%
>> a=eye(4); 4*fliplr(a)

ans =

     0     0     0     4
     0     0     4     0
     0     4     0     0
     4     0     0     0
```