Introduction to MATLAB®

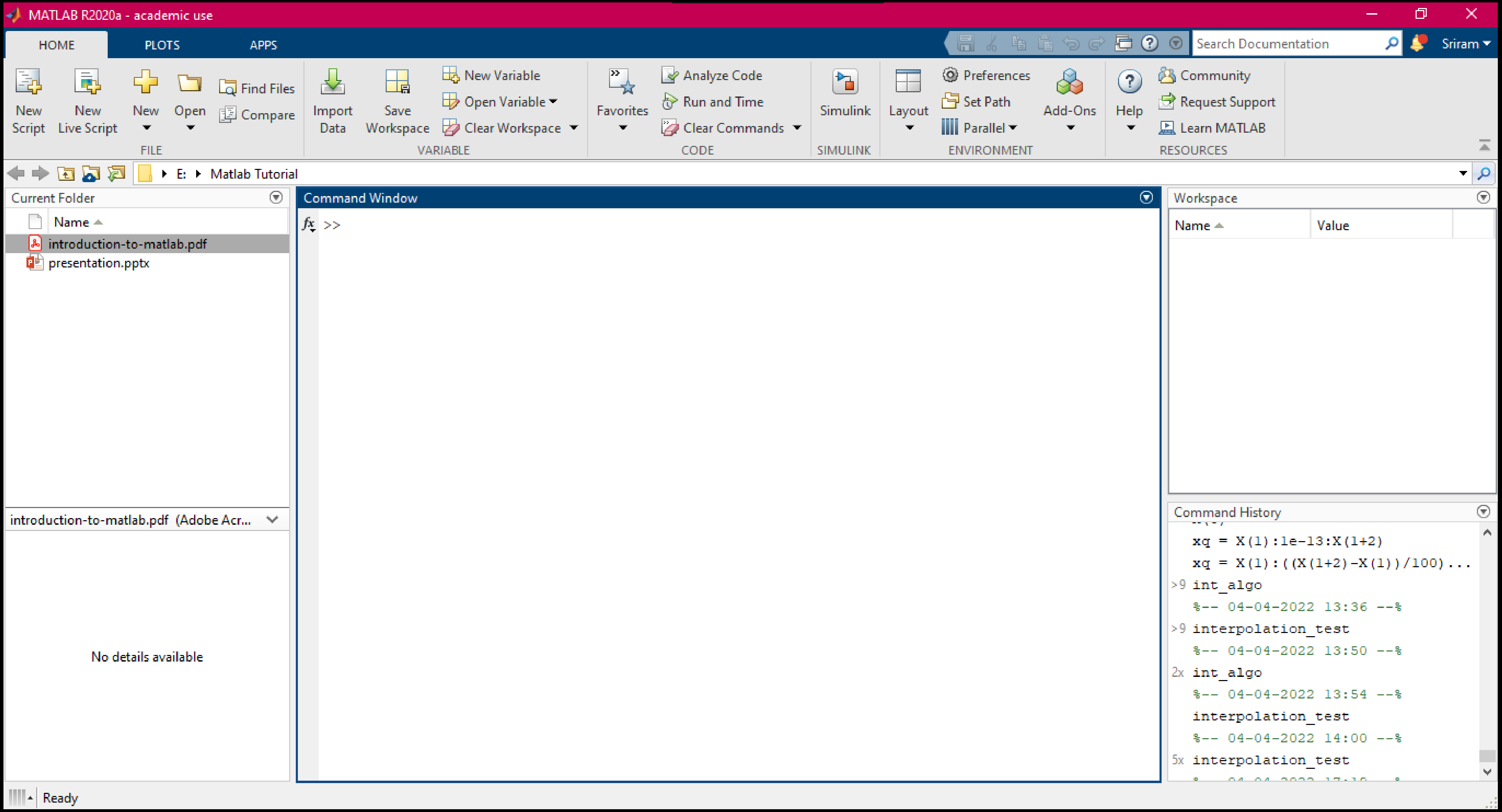
Introduction to Matrices

Sriram Krishnamurthy

Basics

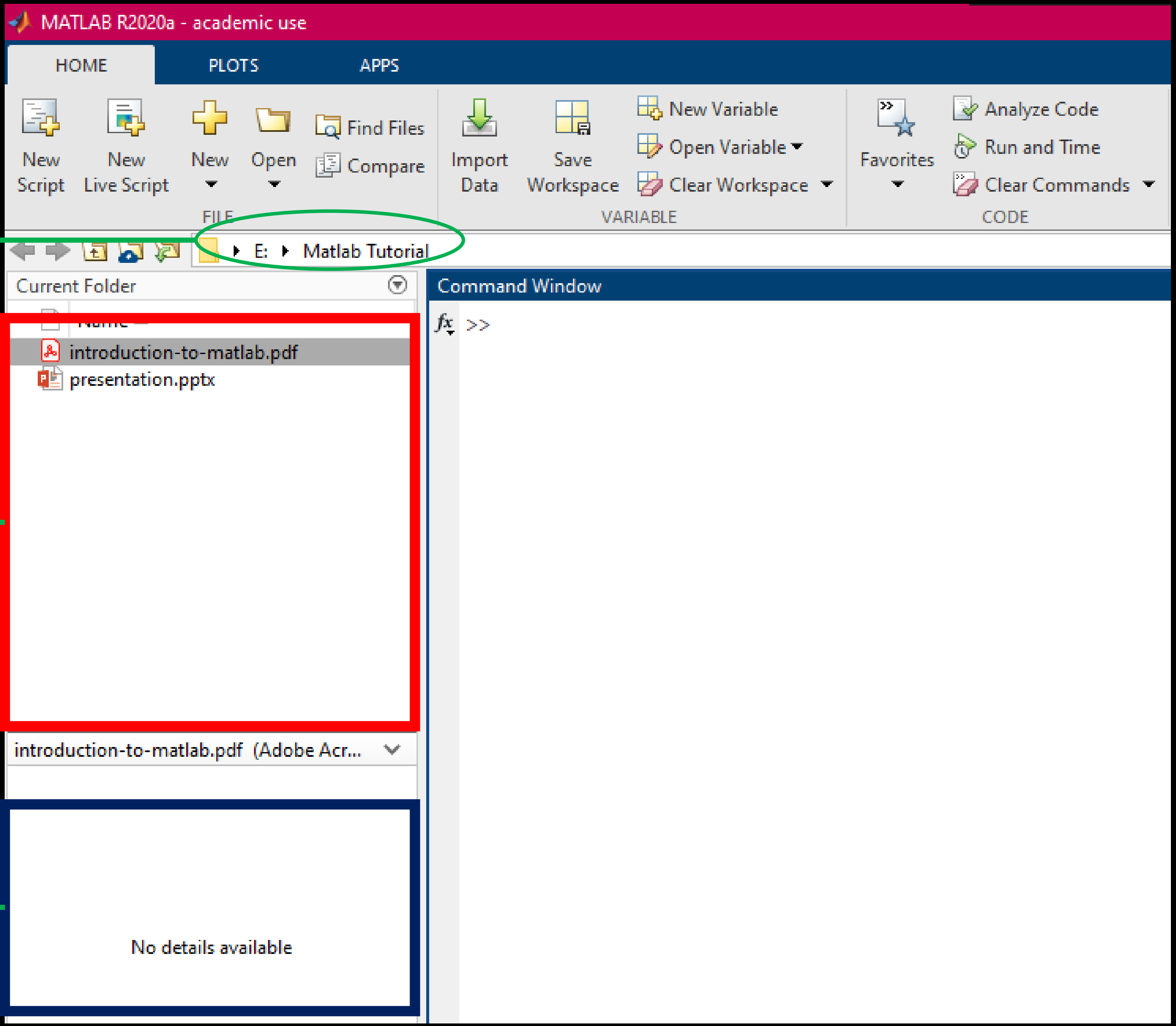
- Matlab is a GUI based computing software generally used for Engineering applications.
- Similar to Wolfram Mathematica
- MATLAB stands for MATrix LABoratory
- Open Source alternatives:

The Command Window



```
   xq = X(1):1e-13:X(1+2)
   xq = X(1):((X(1+2)-X(1))/100)...
>9 int_algo
   %-- 04-04-2022 13:36 --%
>9 interpolation_test
   %-- 04-04-2022 13:50 --%
2x int_algo
   %-- 04-04-2022 13:54 --%
   interpolation_test
   %-- 04-04-2022 14:00 --%
5x interpolation_test
```

The Command Window



introduction-to-matlab.pdf

presentation.pptx

introduction-to-matlab.pdf (Adobe Acr...

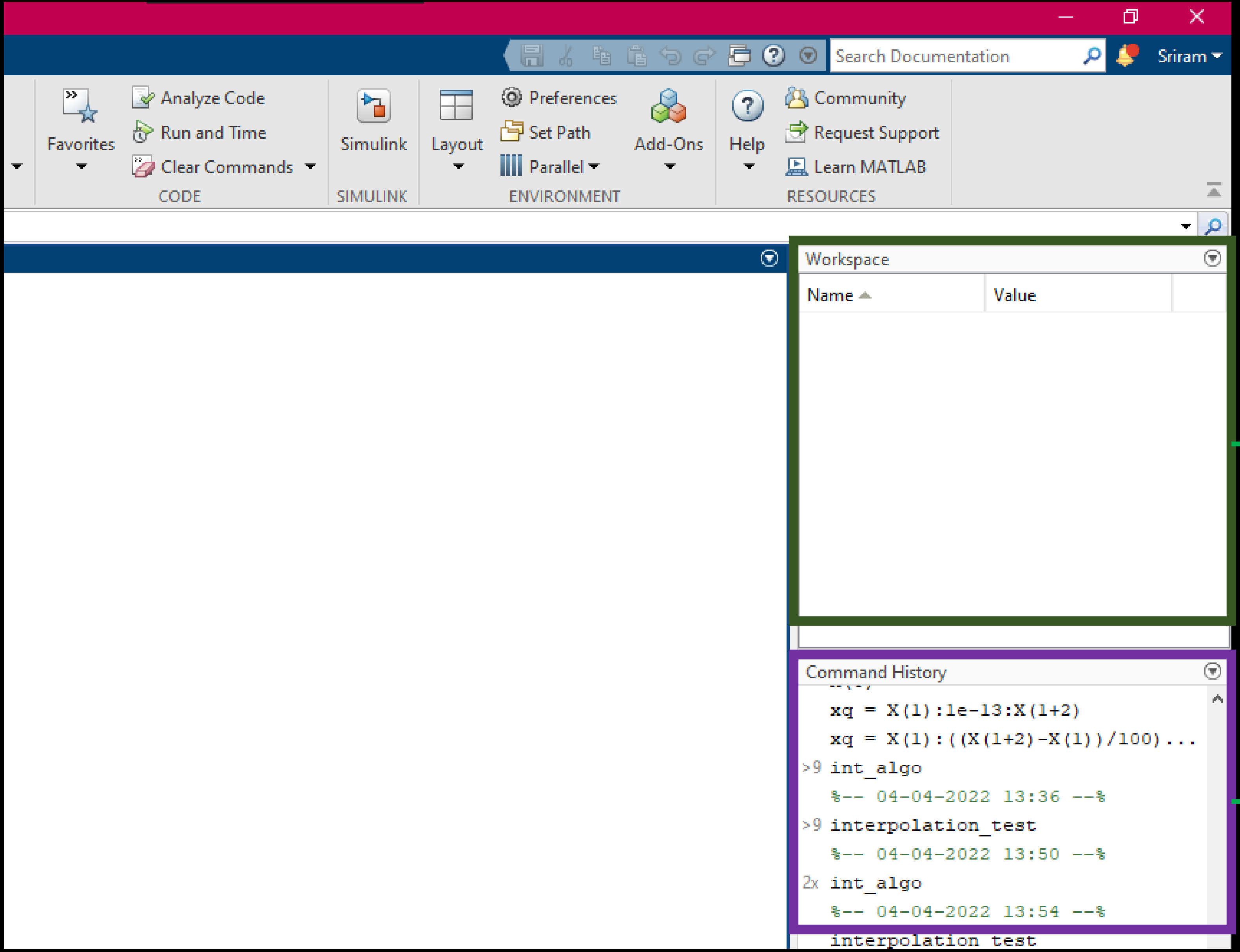
No details available

Path

Contents in the current working folder

Details of the Selected file

The Command Window



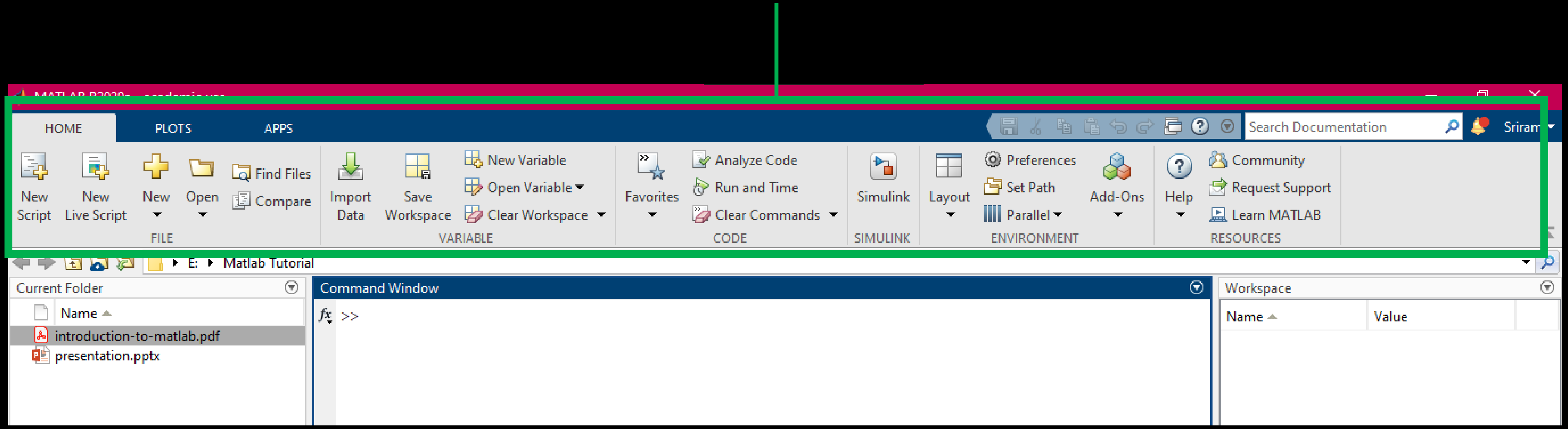
```
  xq = X(1):1e-13:X(1+2)
  xq = X(1):((X(1+2)-X(1))/100)...
>9 int_algo
  %-- 04-04-2022 13:36 --%
>9 interpolation_test
  %-- 04-04-2022 13:50 --%
2x int_algo
  %-- 04-04-2022 13:54 --%
  interpolation_test
```

Work Space – shows all the variables – very Important

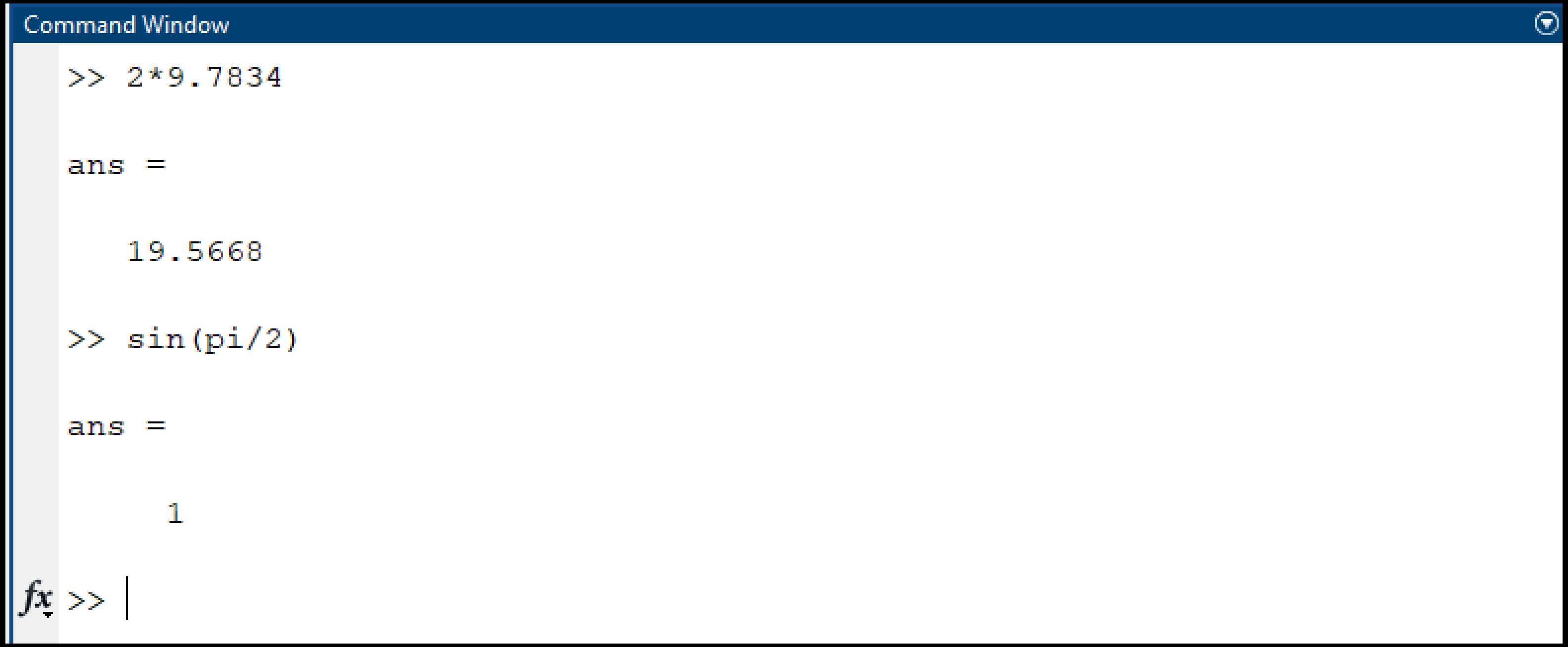
Command History

The Command Window

Commonly Used tools - ToolBar

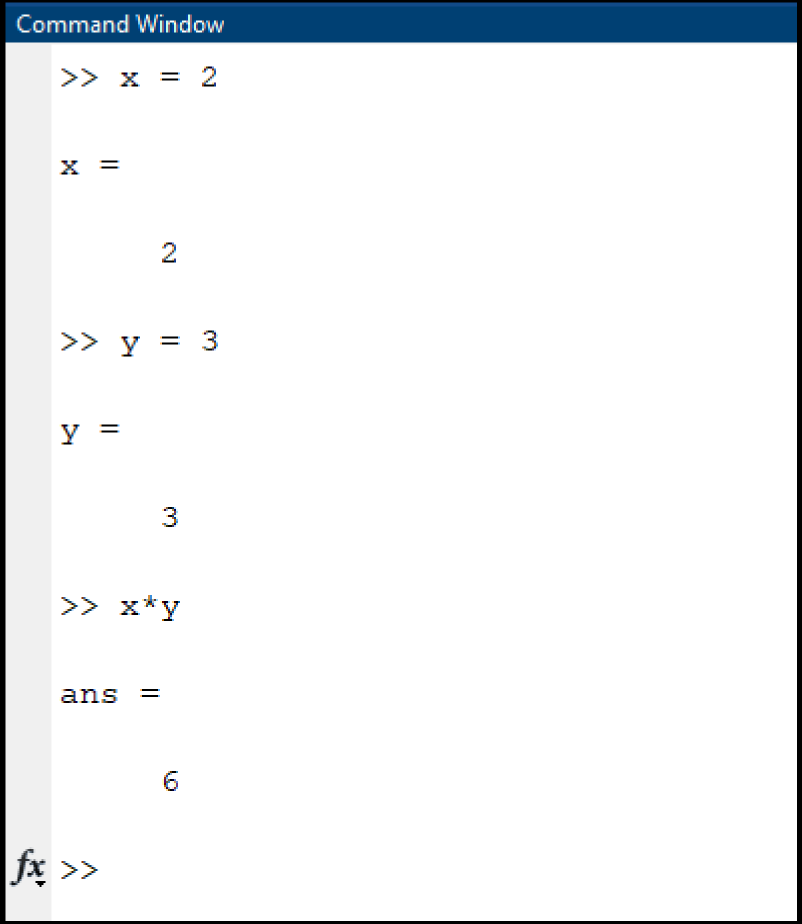


MATLAB – A high performance calculator

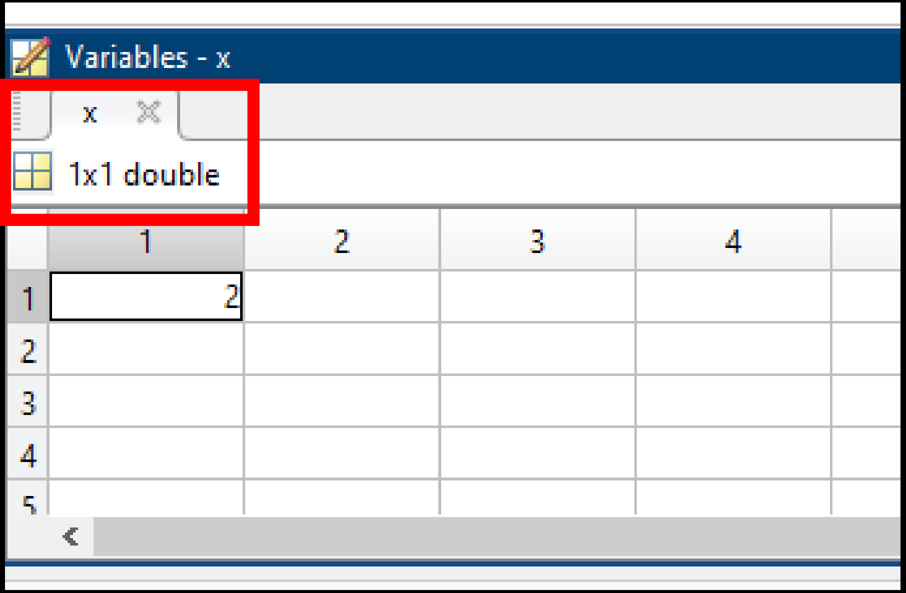


```
>> 2*9.7834

ans =

   19.5668

>> sin(pi/2)

ans =

     1

>> 
```

Variables



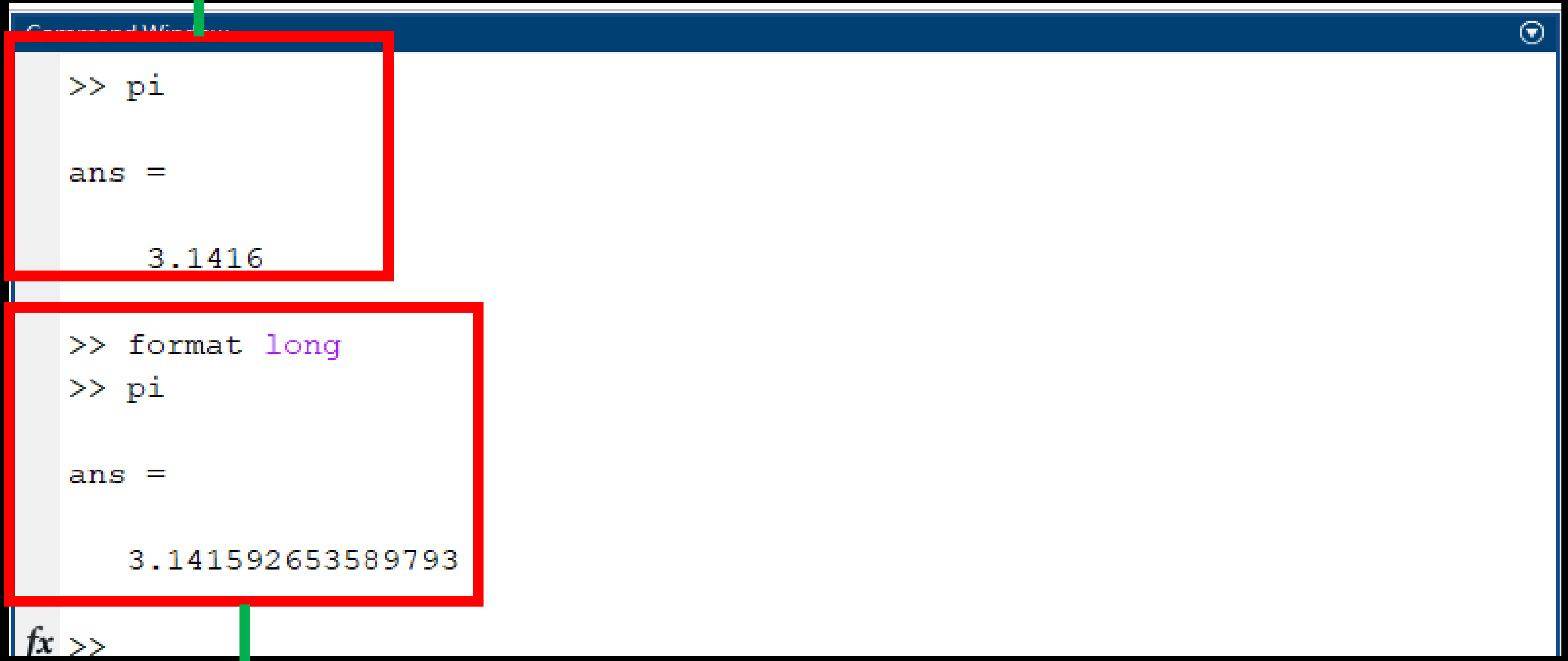
```
>> x = 2

x =

     2

>> y = 3

y =

     3

>> x*y

ans =

     6

>>
```



1x1 double

	1	2	3	4
1	2			
2				
3				
4				
5				

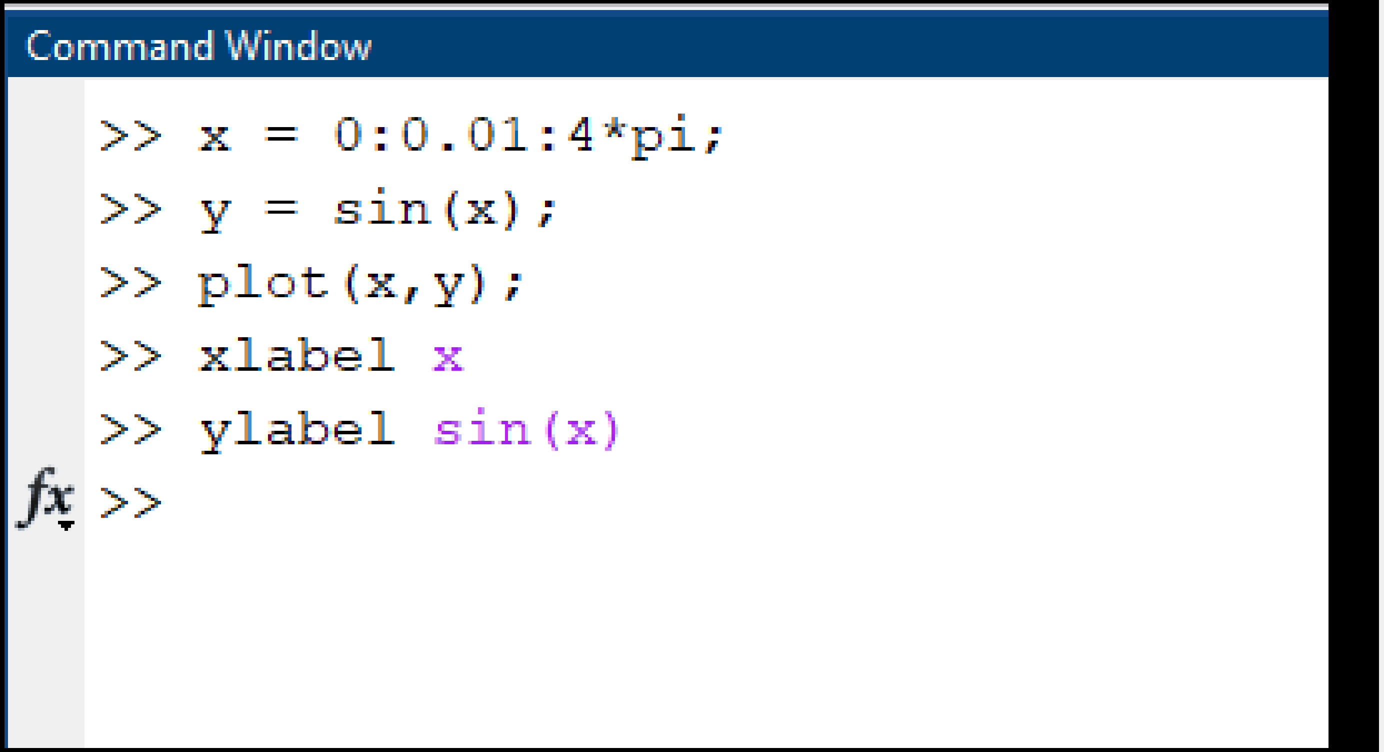
Default format – 4 digits after the decimal (just on the screen!) [format short]



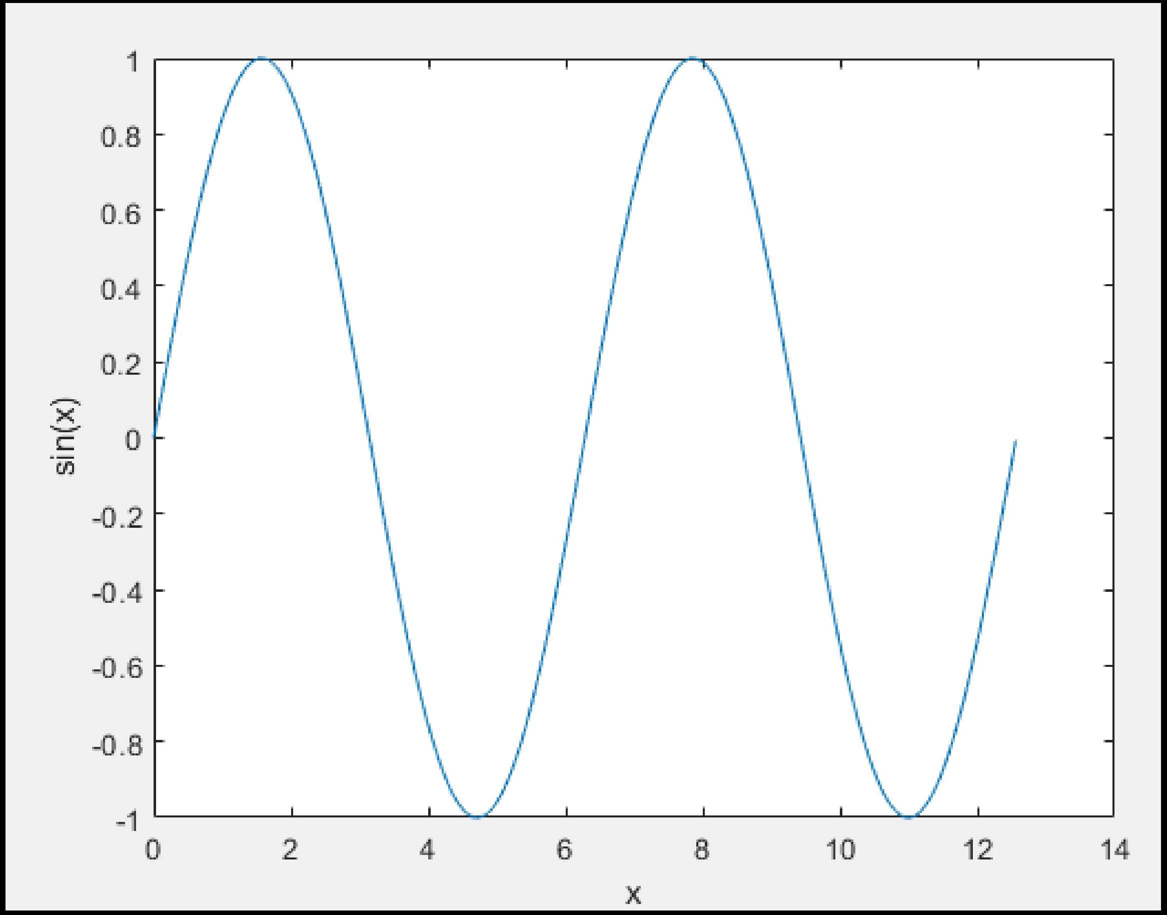
```
>> pi

ans =

    3.1416

>> format long
>> pi

ans =

   3.141592653589793

>>
```

format long – 15 digits after the decimal (actual value used for calculation)

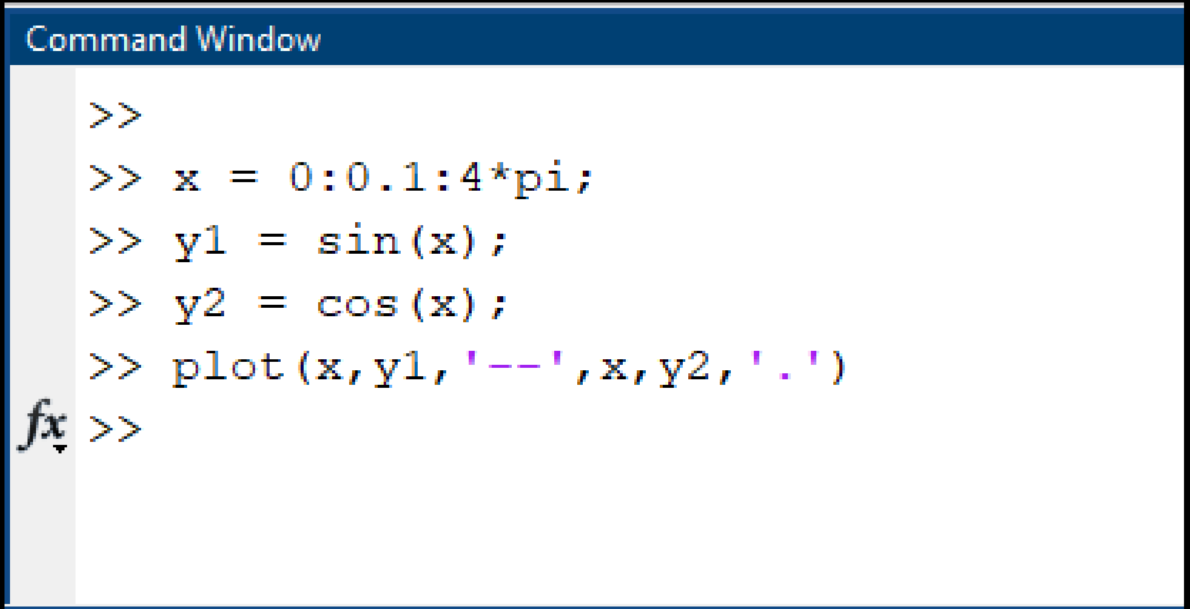
Basic Plotting



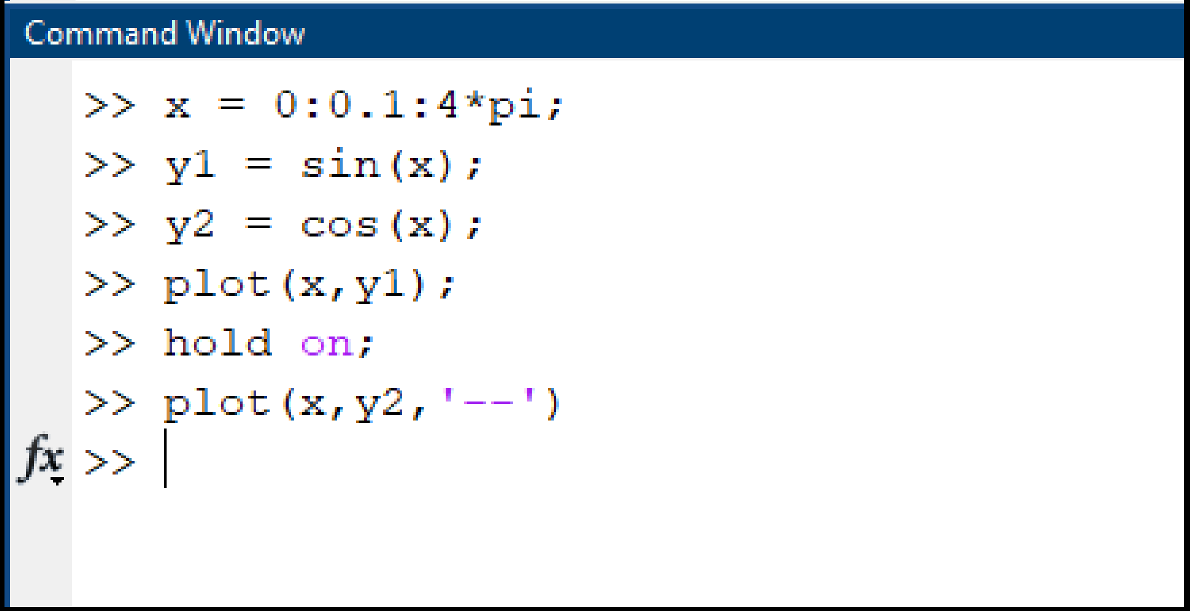
```
>> x = 0:0.01:4*pi;
>> y = sin(x);
>> plot(x,y);
>> xlabel x
>> ylabel sin(x)
>>
```



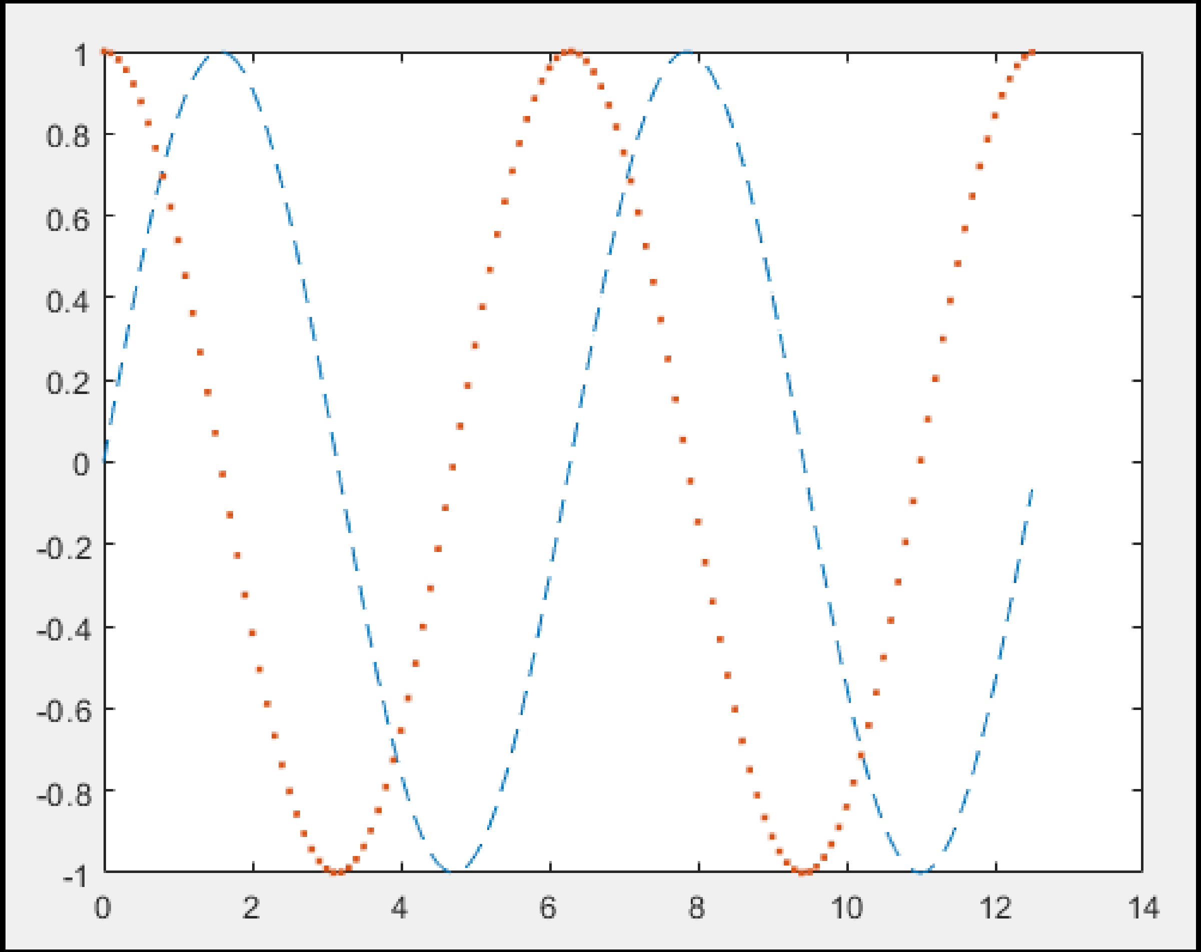
Basic Plotting



```
>>
>> x = 0:0.1:4*pi;
>> y1 = sin(x);
>> y2 = cos(x);
>> plot(x,y1,'--',x,y2,'.')
>>
```

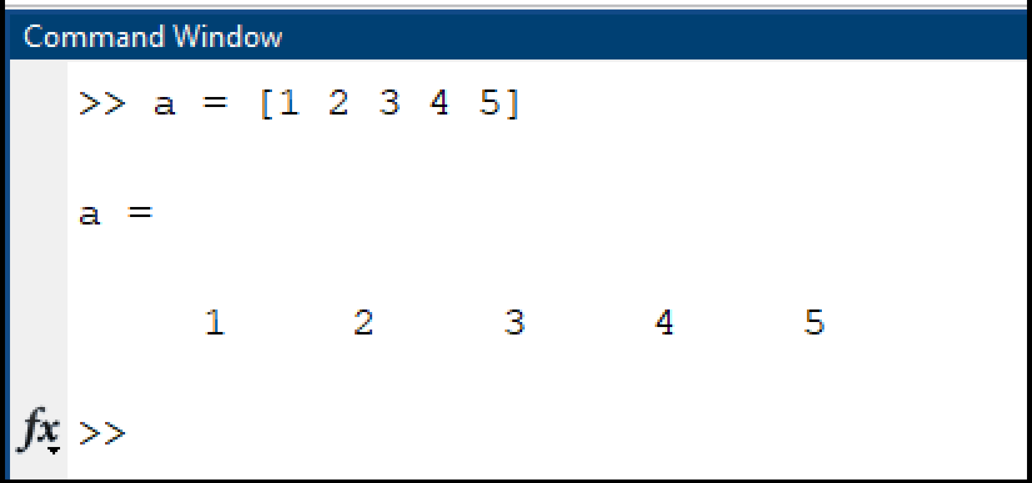


```
>> x = 0:0.1:4*pi;
>> y1 = sin(x);
>> y2 = cos(x);
>> plot(x,y1);
>> hold on;
>> plot(x,y2,'--')
>>
```



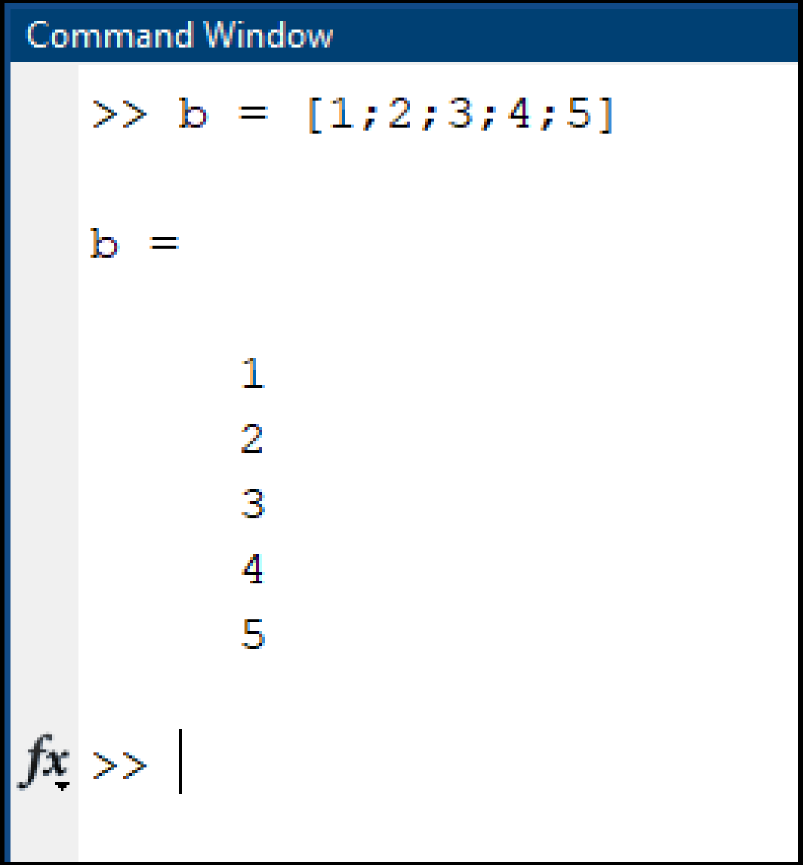
Matrices

Row Vector



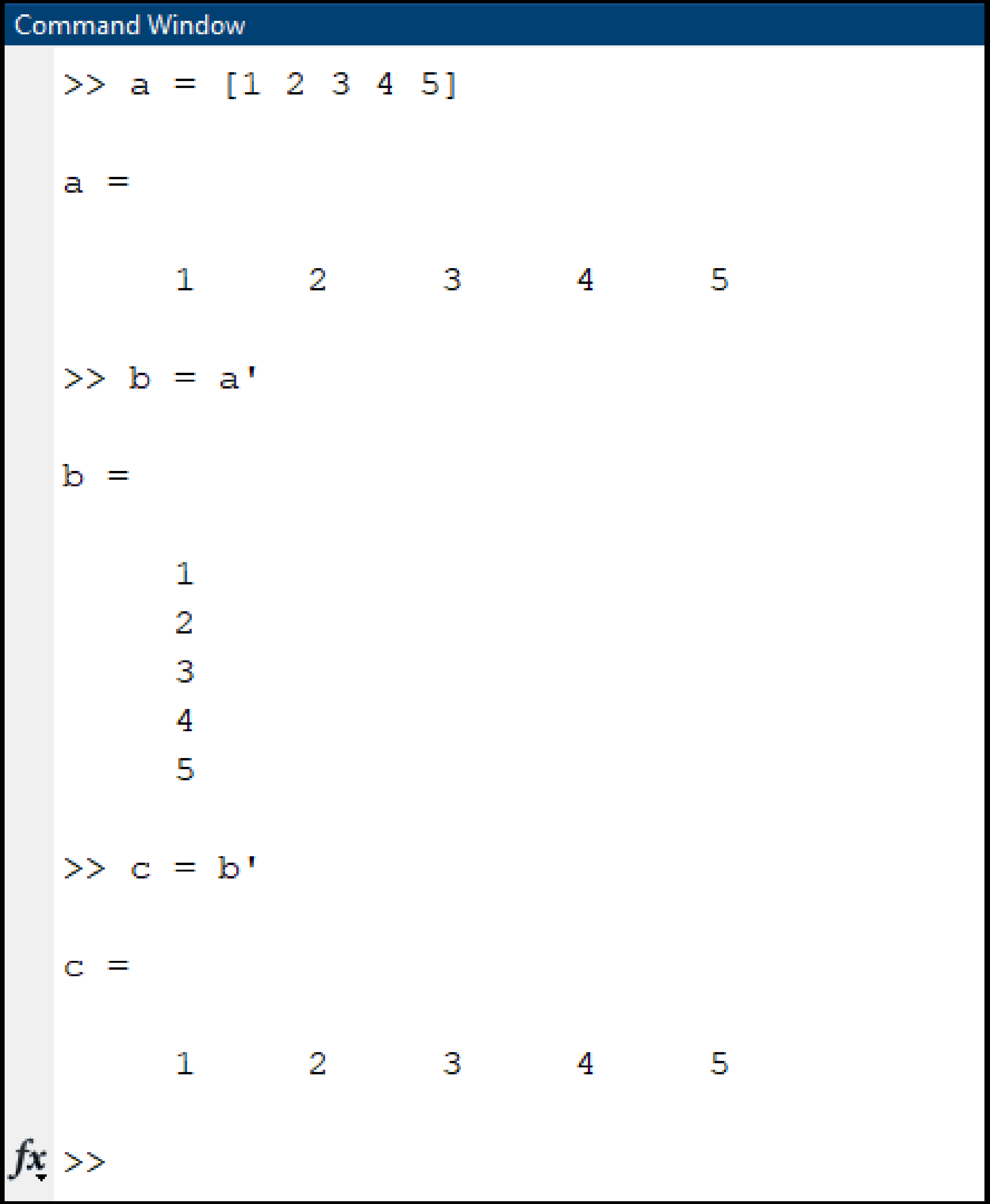
```
>> a = [1 2 3 4 5]

a =

     1     2     3     4     5

>>
```

Column Vector



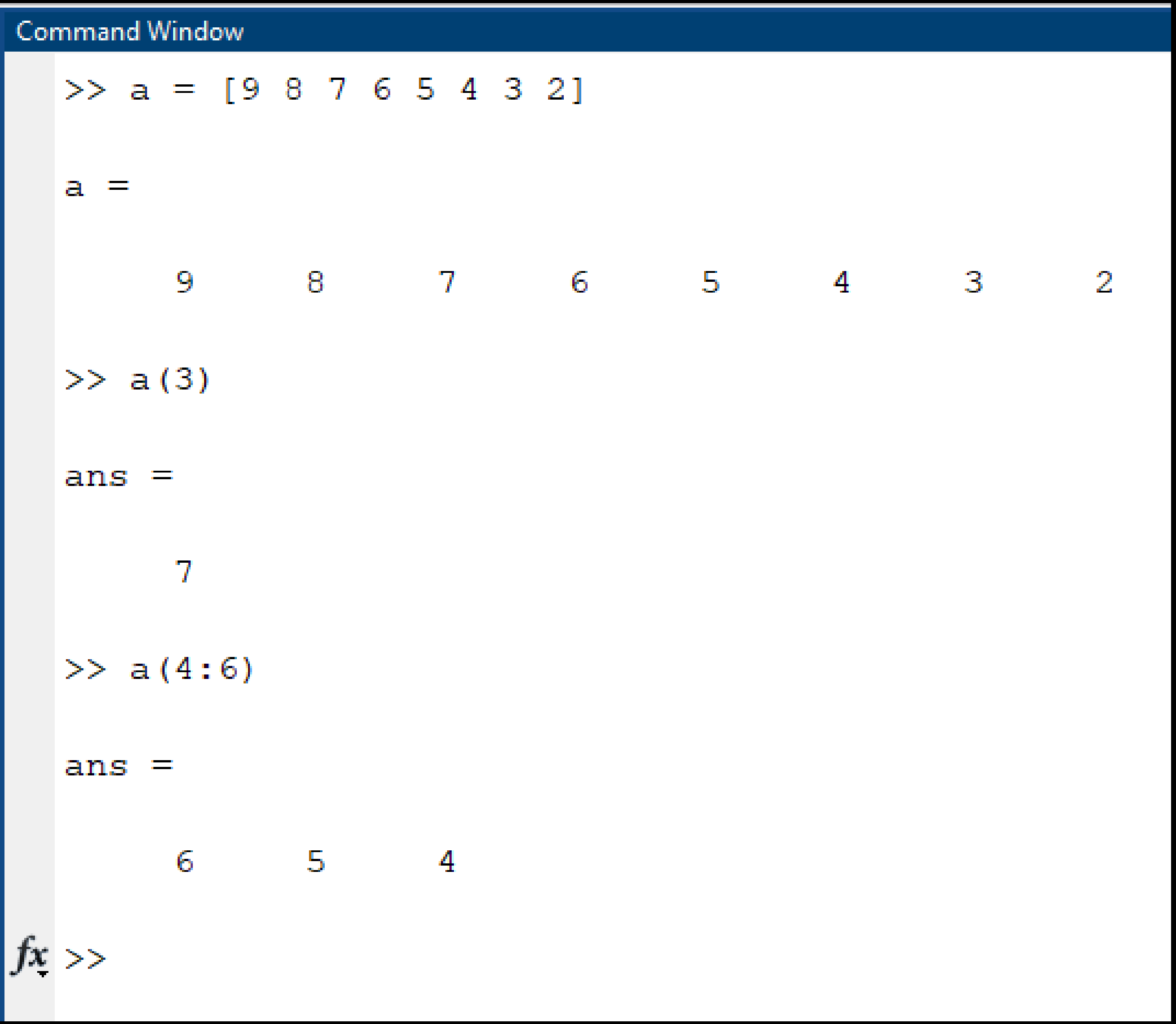
```
>> b = [1;2;3;4;5]

b =

     1
     2
     3
     4
     5

>>
```

Vector Transpose

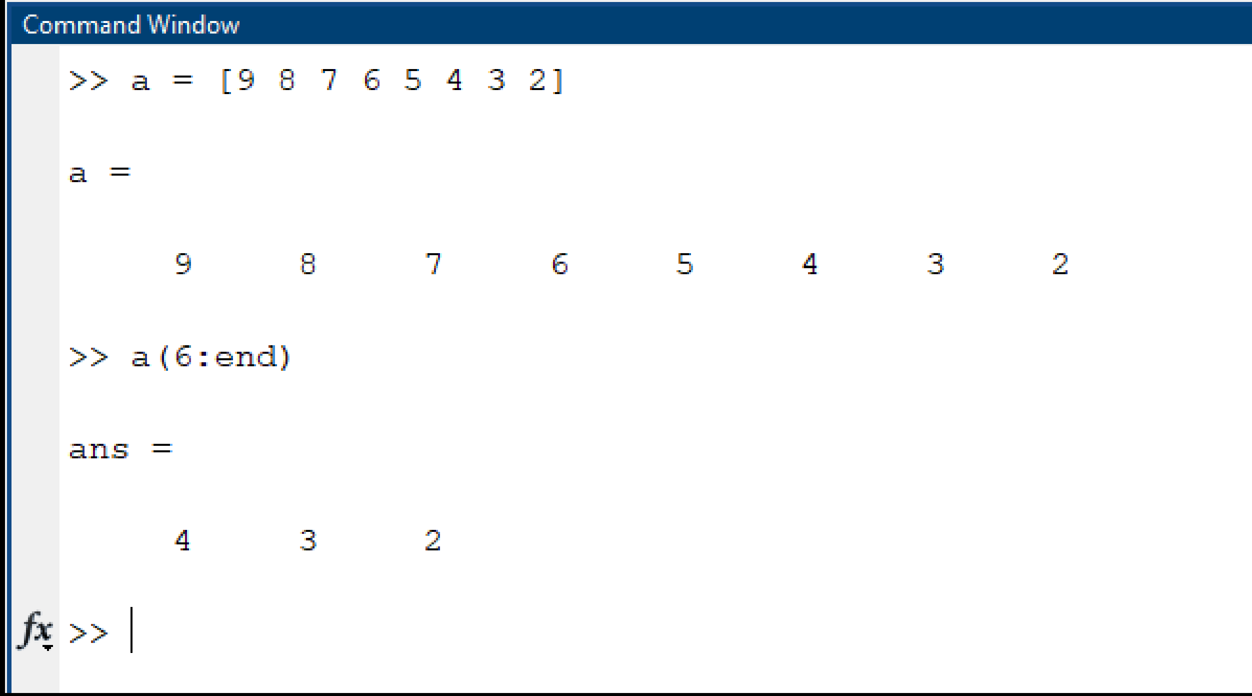


```
>> a = [1 2 3 4 5]

a =

     1     2     3     4     5

>> b = a'

b =

     1
     2
     3
     4
     5

>> c = b'

c =

     1     2     3     4     5

>>
```

Accessing elements

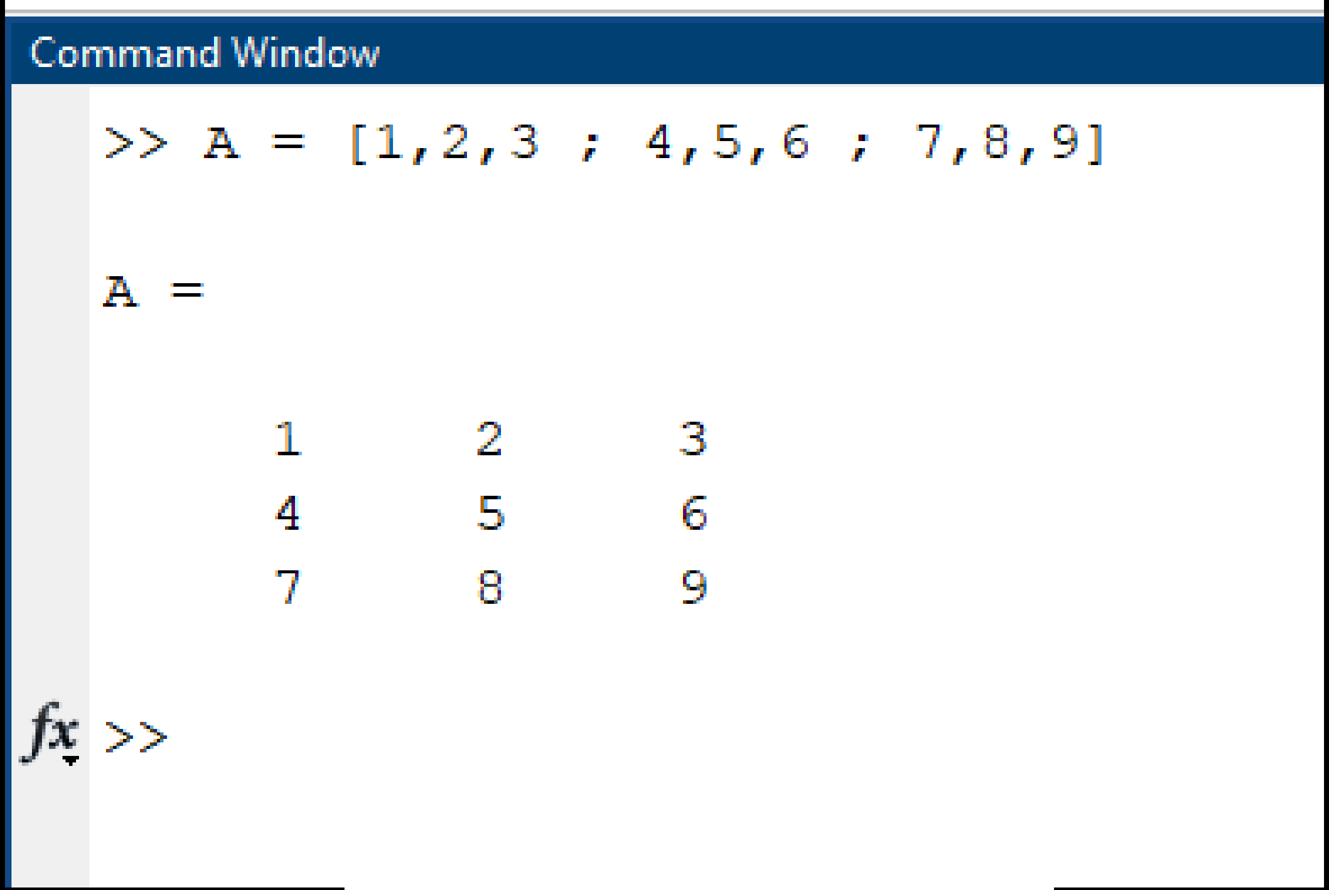


```
>> a = [9 8 7 6 5 4 3 2]

a =

     9     8     7     6     5     4     3     2

>> a(3)

ans =

     7

>> a(4:6)

ans =

     6     5     4

>>
```

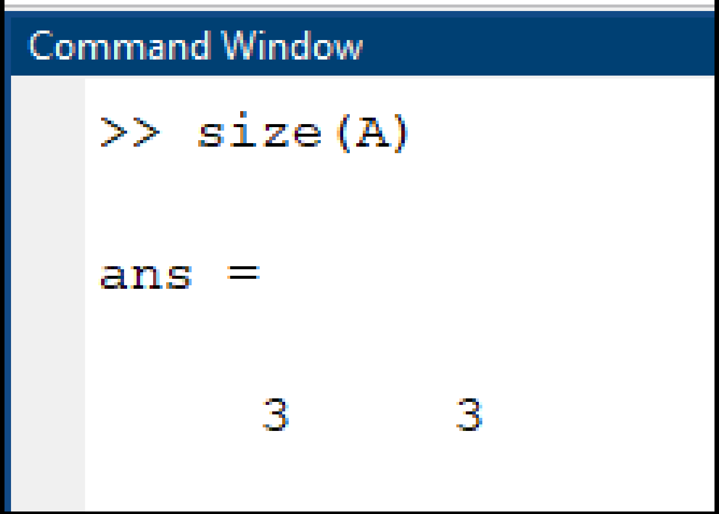


```
>> a = [9 8 7 6 5 4 3 2]

a =

     9     8     7     6     5     4     3     2

>> a(6:end)

ans =

     4     3     2

>>
```

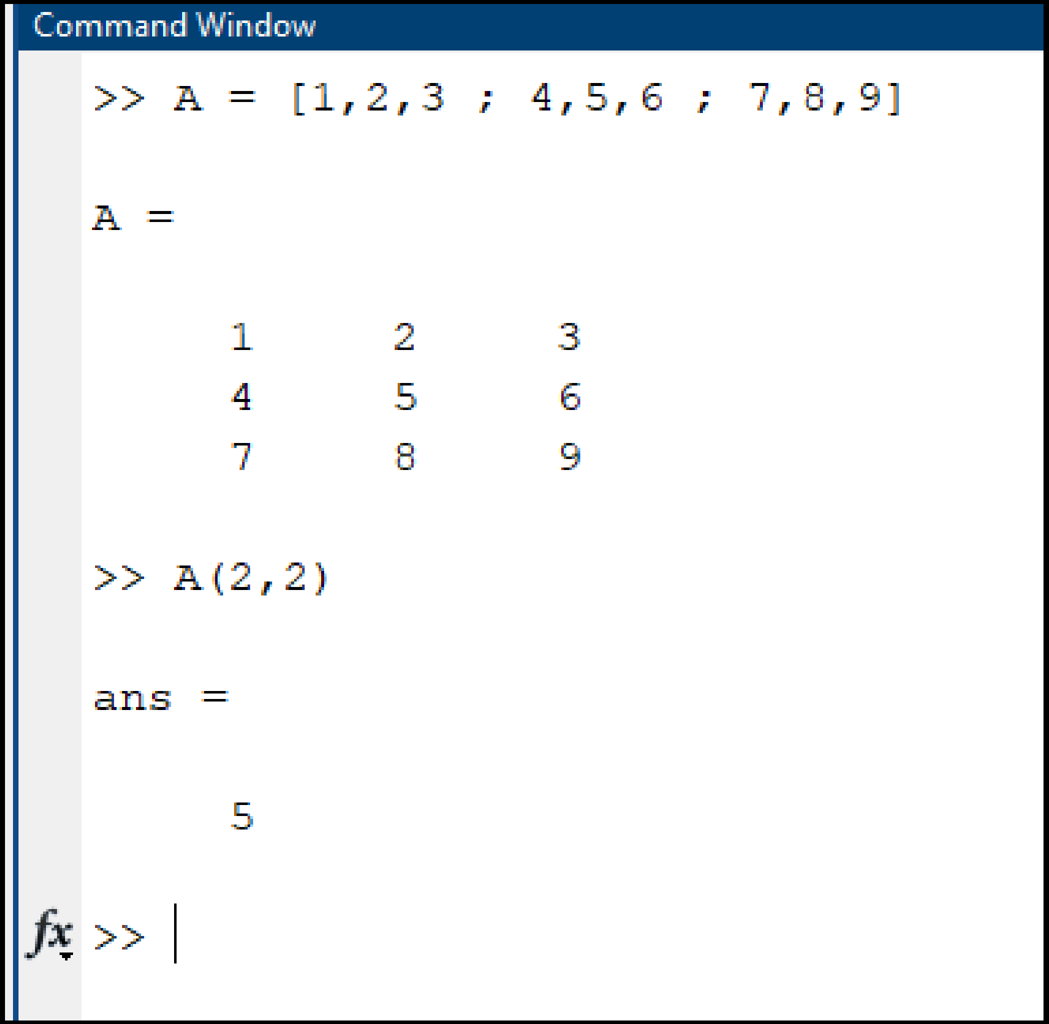
- Begin with a square bracket, **[**
- Separate elements in a row with spaces or commas **(,)**
- Use a semicolon **(;)** to separate rows
- End the matrix with another square bracket, **]**.



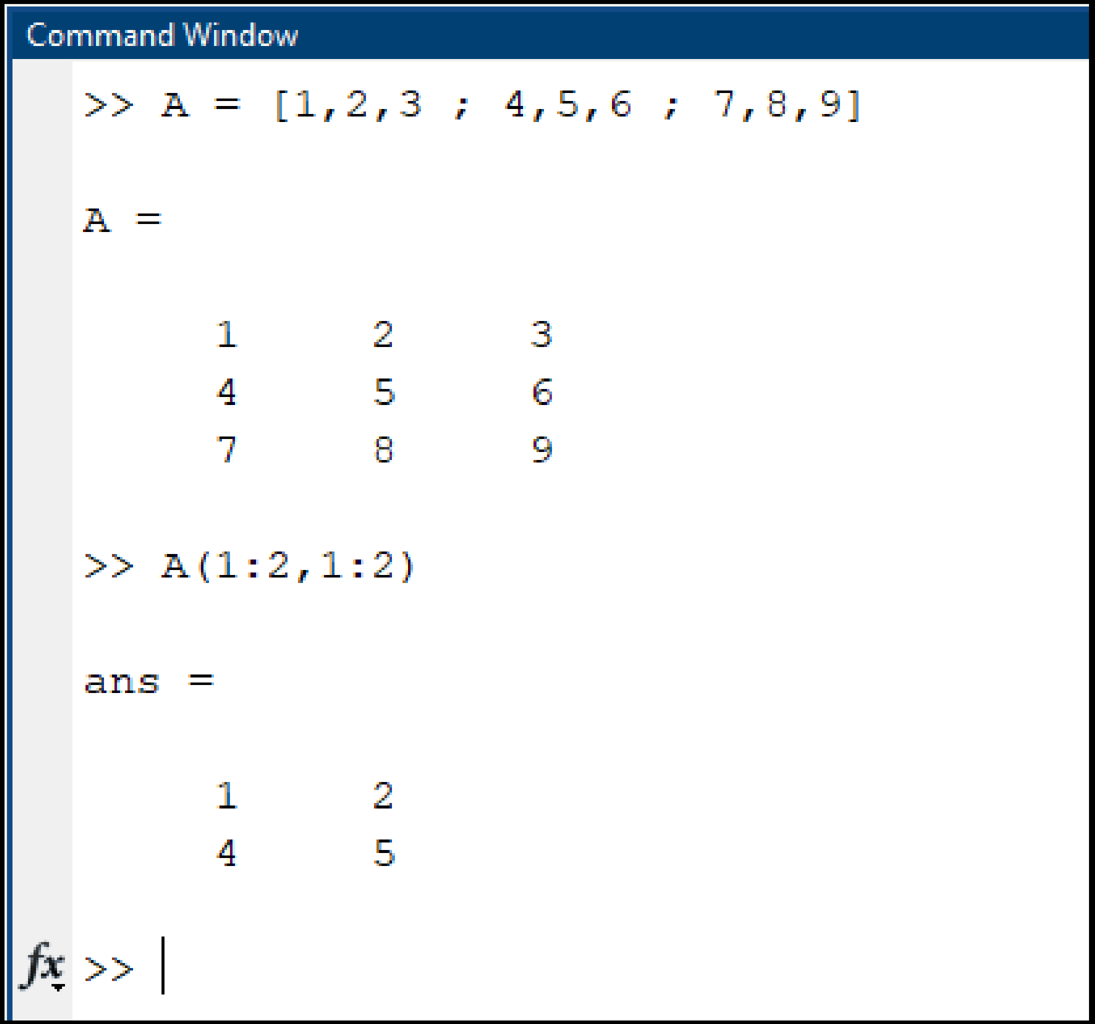
```
>> A = [1,2,3 ; 4,5,6 ; 7,8,9]

A =

     1     2     3
     4     5     6
     7     8     9

>>
```



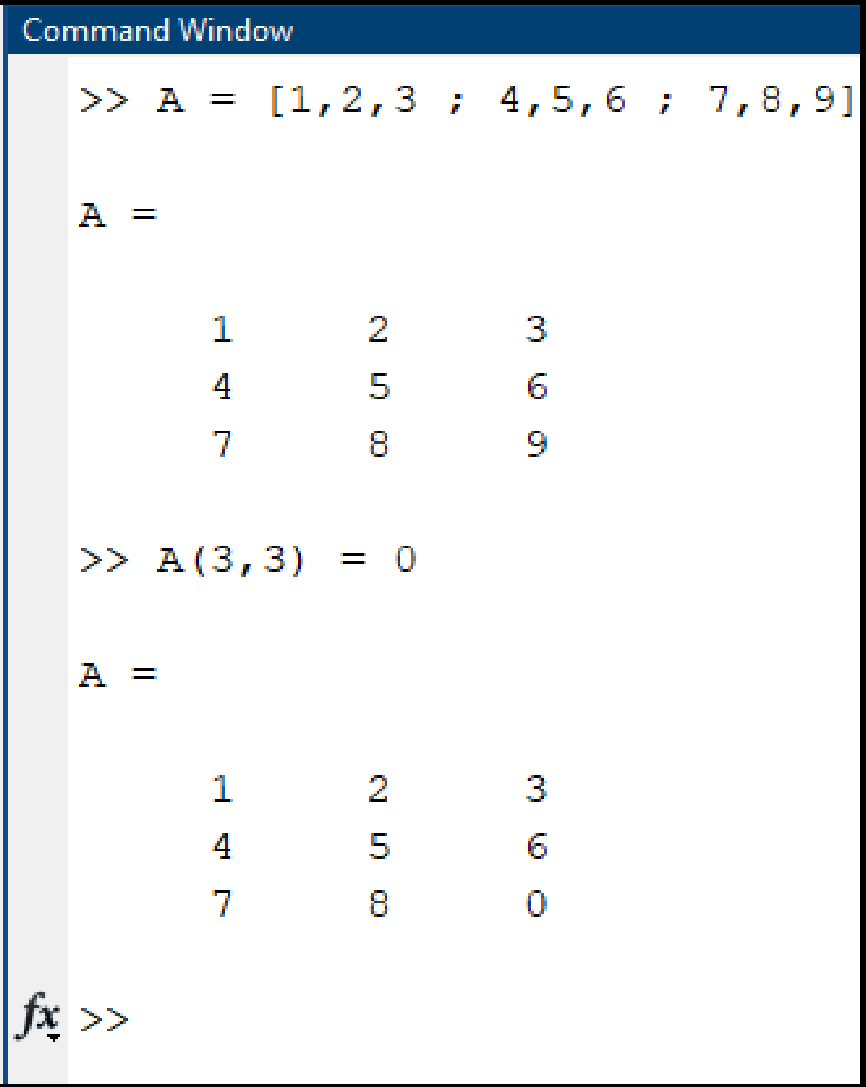
```
>> size(A)

ans =

     3     3
```



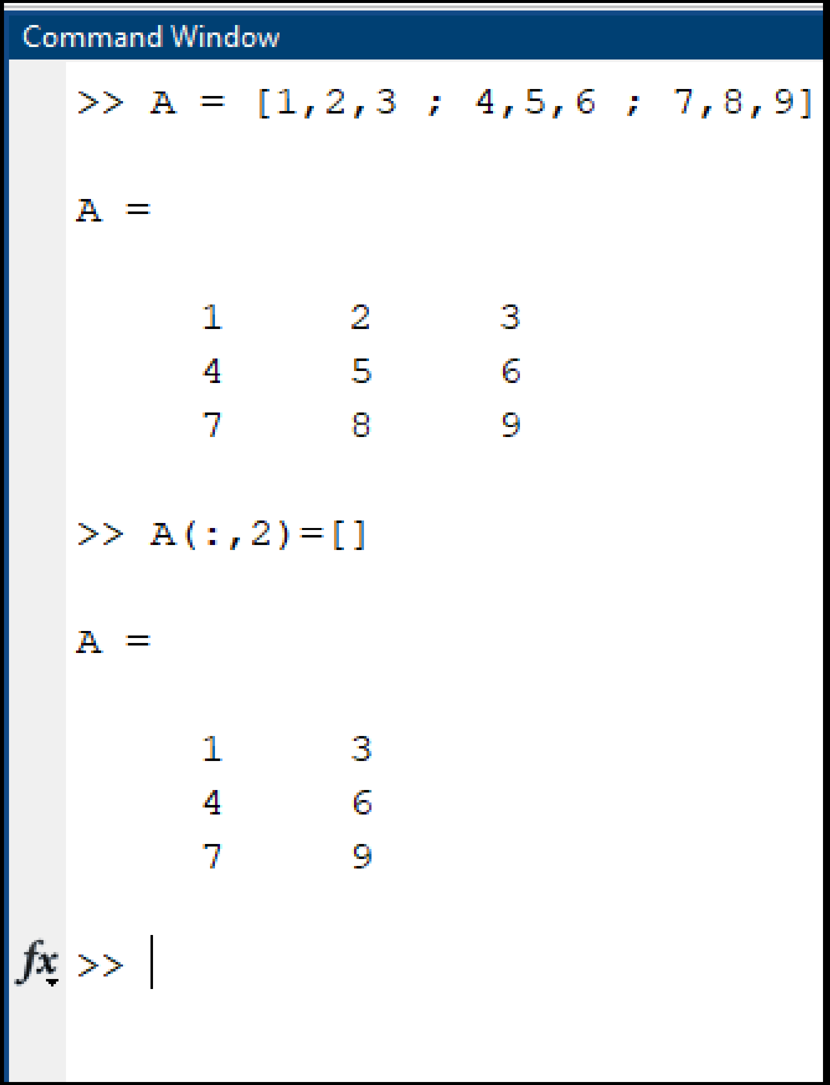
```
>> A = [1,2,3 ; 4,5,6 ; 7,8,9]

A =

     1     2     3
     4     5     6
     7     8     9

>> A(2,2)

ans =

     5

>> 
```



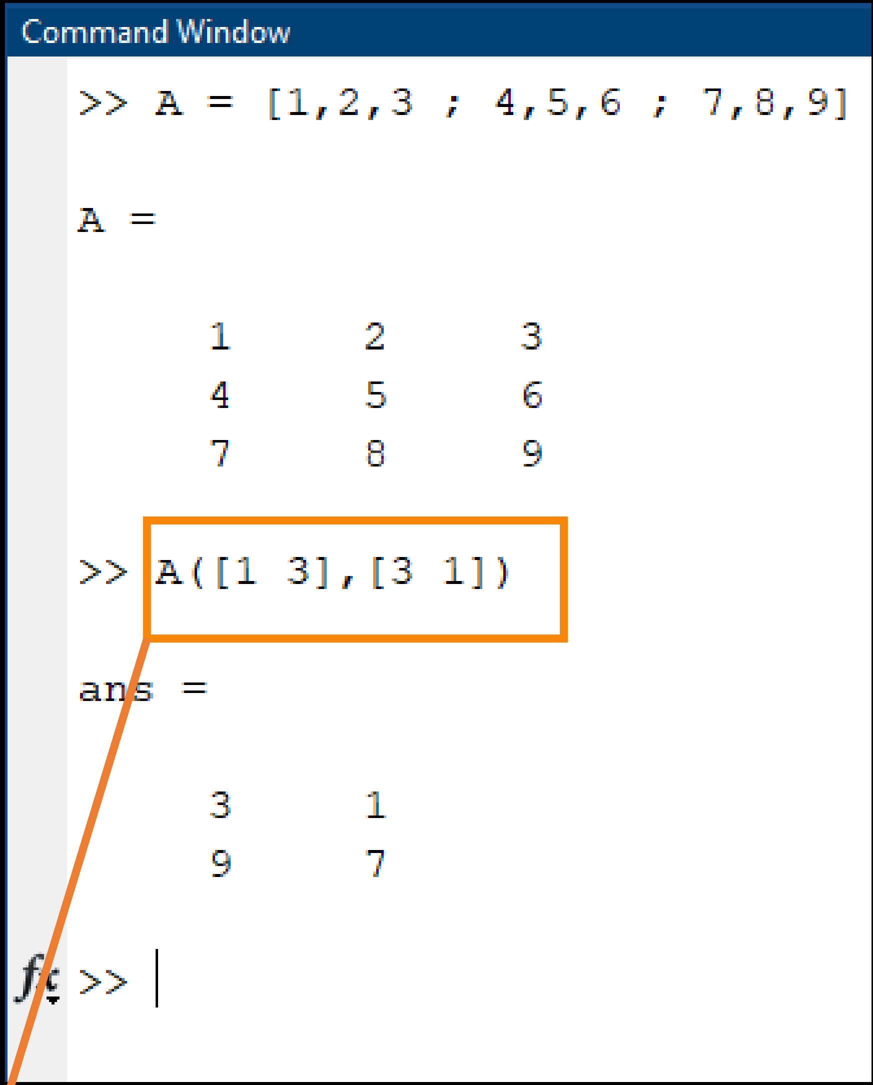
```
>> A = [1,2,3 ; 4,5,6 ; 7,8,9]

A =

     1     2     3
     4     5     6
     7     8     9

>> A(1:2,1:2)

ans =

     1     2
     4     5

>> 
```



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

A =

     1     2     3
     4     5     6
     7     8     9

>> A(3,3) = 0

A =

     1     2     3
     4     5     6
     7     8     0

fx >>
```



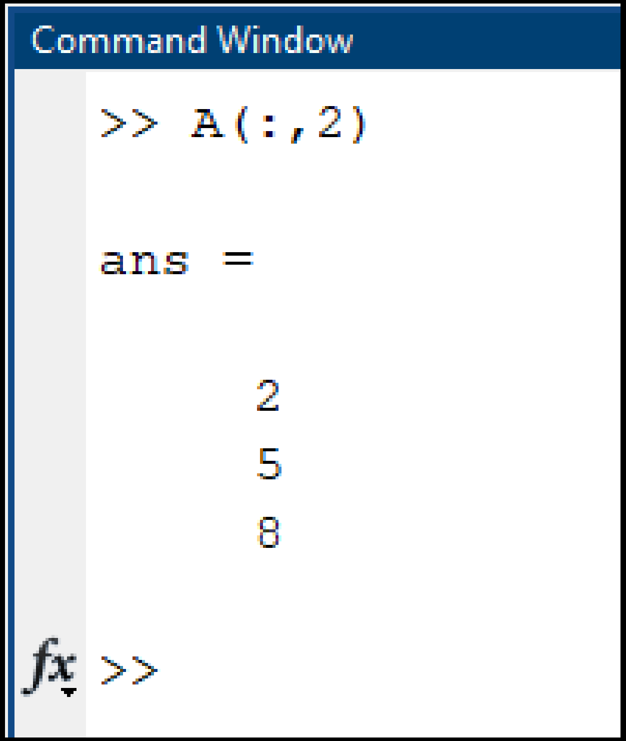
```
>> A = [1,2,3 ; 4,5,6 ; 7,8,9]

A =

     1     2     3
     4     5     6
     7     8     9

>> A(:,2)=[]

A =

     1     3
     4     6
     7     9

fx >>
```



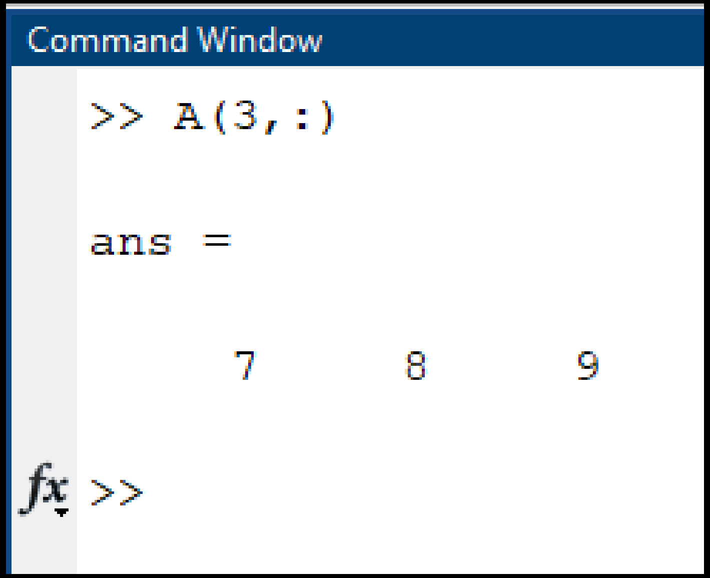
```
>> A = [1,2,3 ; 4,5,6 ; 7,8,9]

A =

     1     2     3
     4     5     6
     7     8     9

>> A([1 3],[3 1])

ans =

     3     1
     9     7

fx >>
```

Row 1&3, Column 3&1

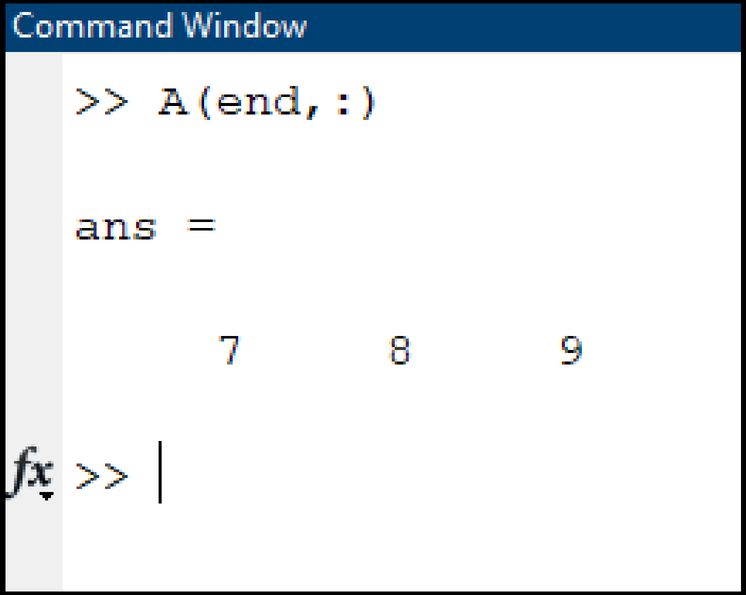
- A(:,j) is the jth column of A
- A(i,:) is the ith row, and
- A(end,:) picks out the last row of A



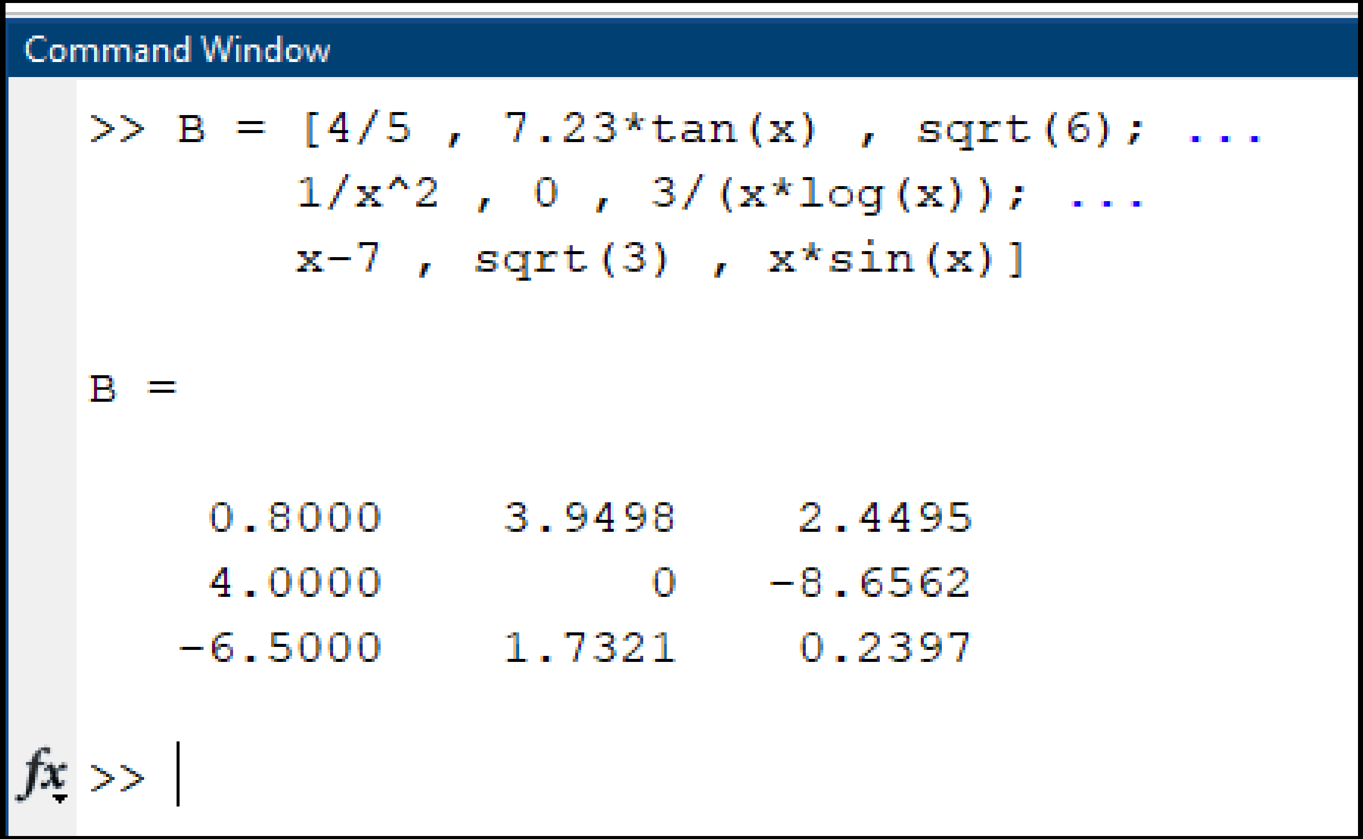
```
>> A(:,2)

ans =

     2
     5
     8

>>
```



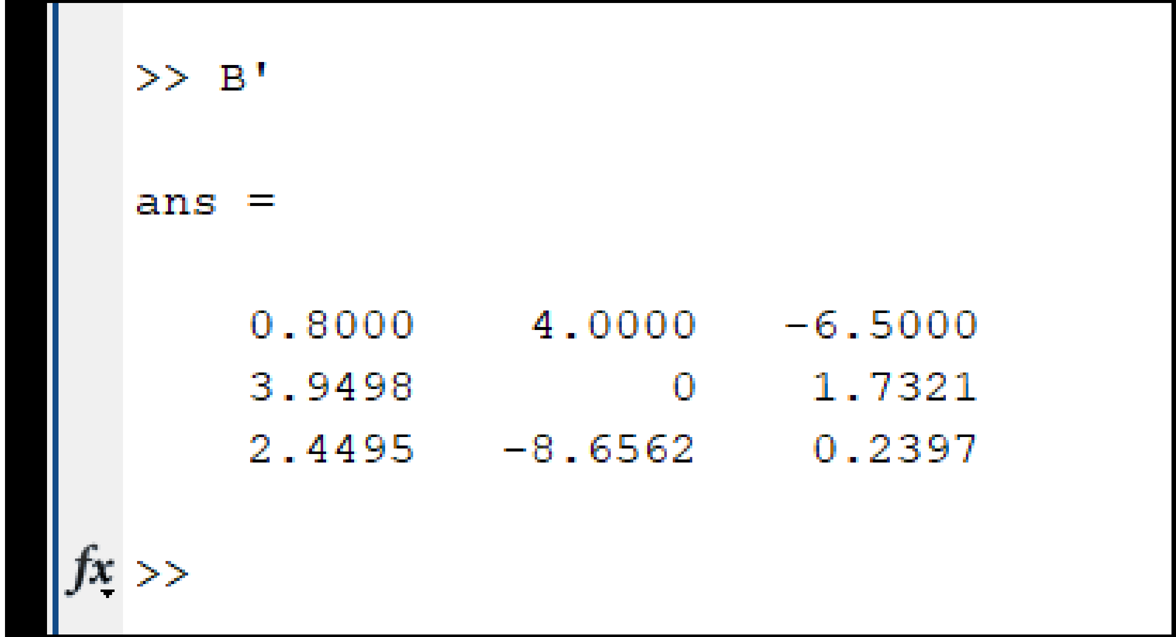
```
>> A(3,:)

ans =

     7     8     9

>>
```



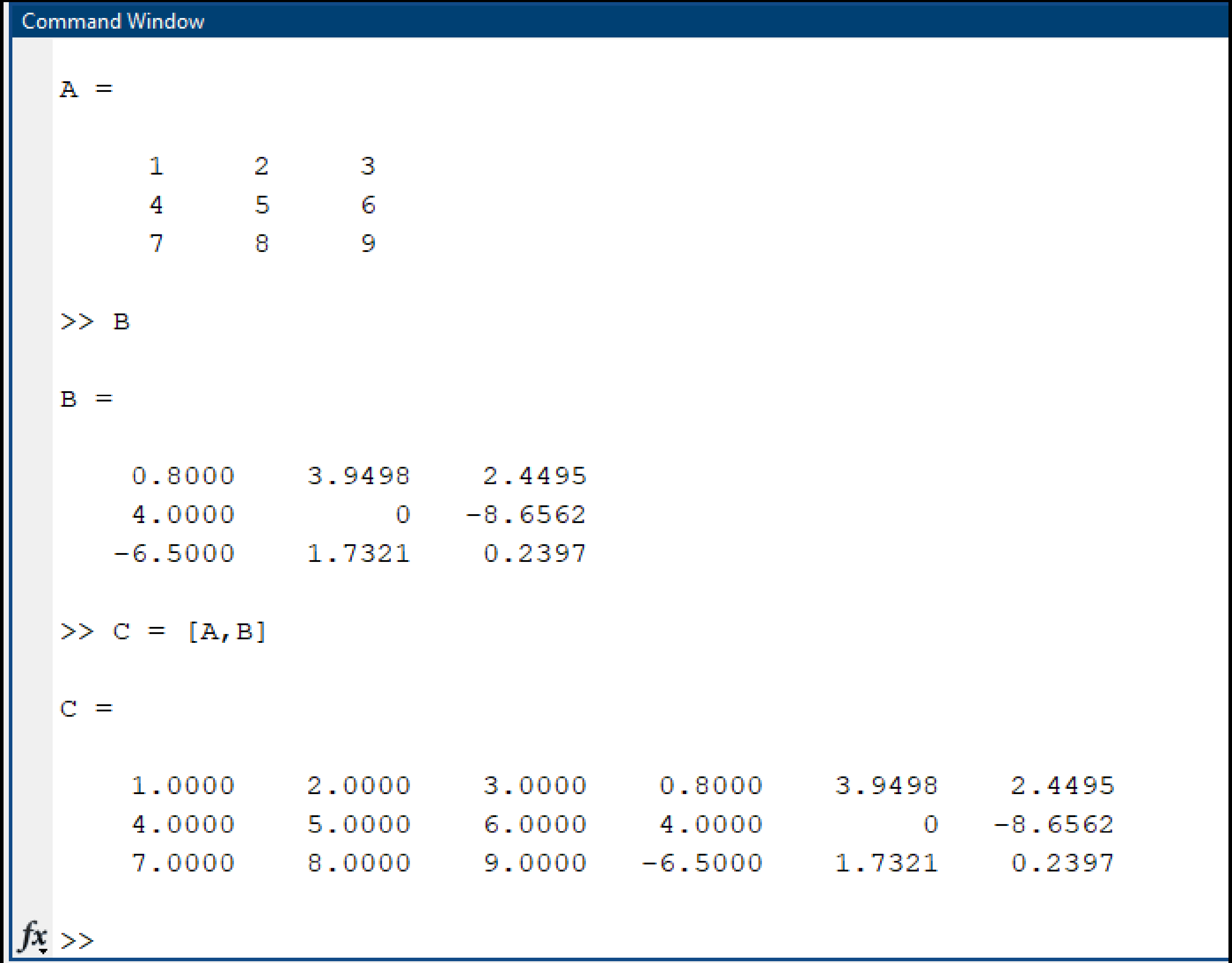
```
>> A(end,:)

ans =

     7     8     9

>>
```



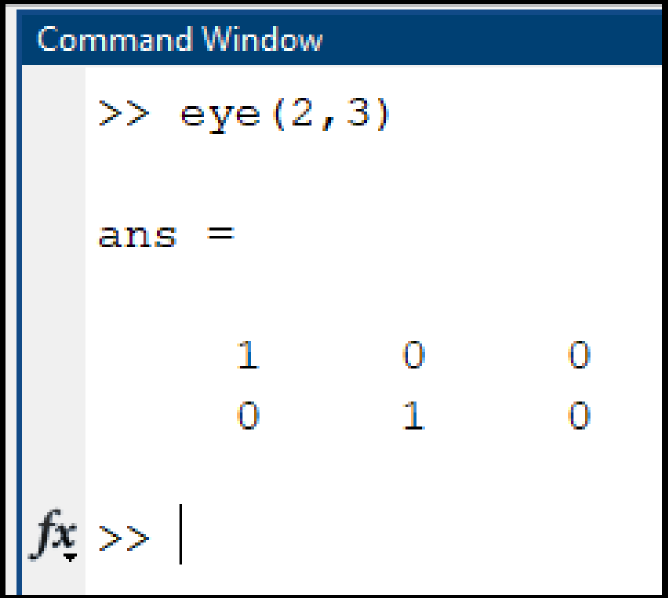
```
>> B = [4/5 , 7.23*tan(x) , sqrt(6); ...
       1/x^2 , 0 , 3/(x*log(x)); ...
       x-7 , sqrt(3) , x*sin(x)]

B =

    0.8000    3.9498    2.4495
    4.0000         0   -8.6562
   -6.5000    1.7321    0.2397

>> 
```



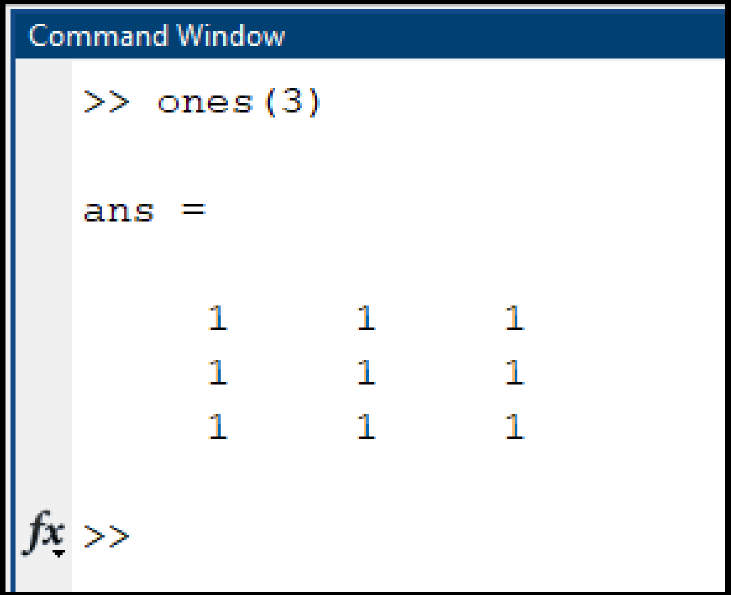
```
>> B'

ans =

    0.8000    4.0000   -6.5000
    3.9498         0    1.7321
    2.4495   -8.6562    0.2397

>> 
```



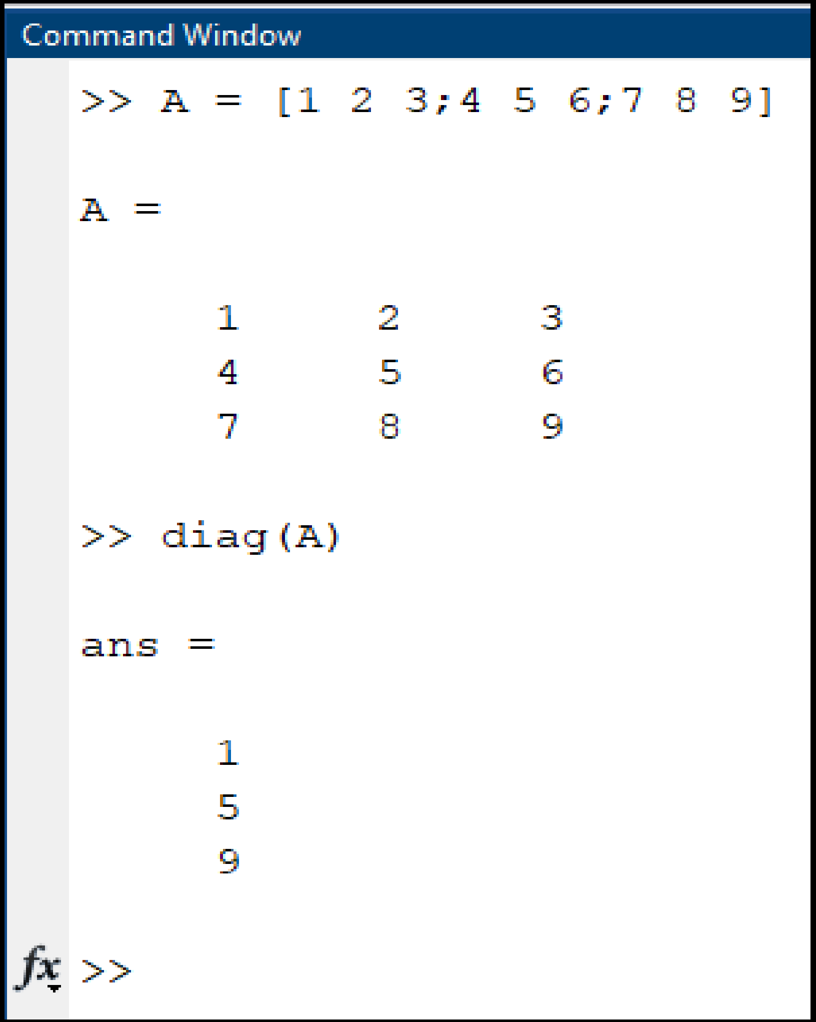
```
A =

     1     2     3
     4     5     6
     7     8     9

>> B

B =

    0.8000    3.9498    2.4495
    4.0000         0   -8.6562
   -6.5000    1.7321    0.2397

>> C = [A,B]

C =

    1.0000    2.0000    3.0000    0.8000    3.9498    2.4495
    4.0000    5.0000    6.0000    4.0000         0   -8.6562
    7.0000    8.0000    9.0000   -6.5000    1.7321    0.2397

>>
```



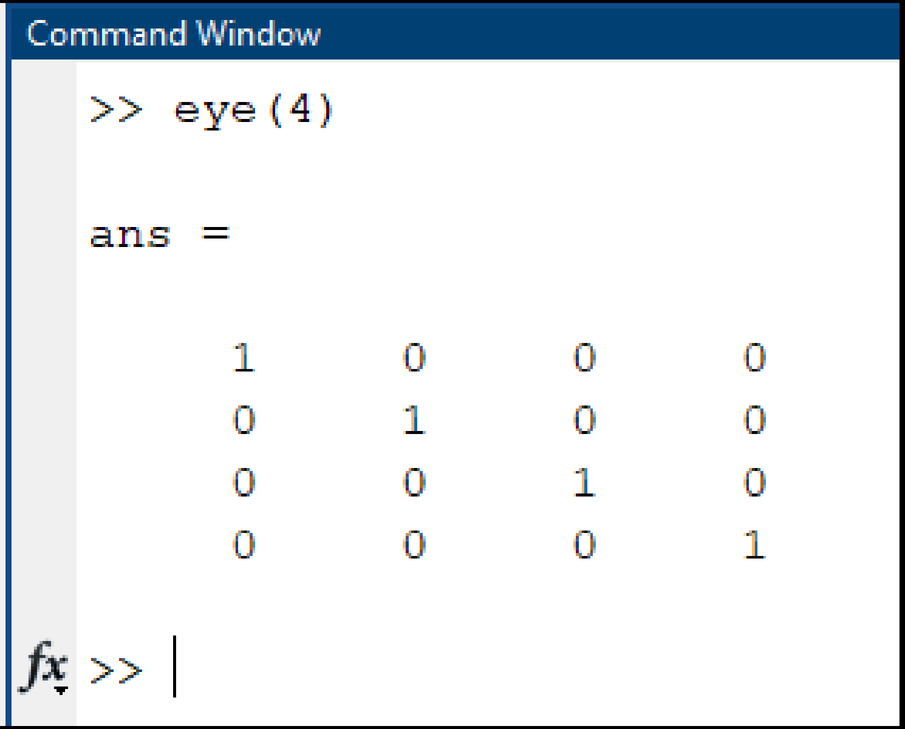
```
>> eye(2,3)

ans =

     1     0     0
     0     1     0

>> 
```



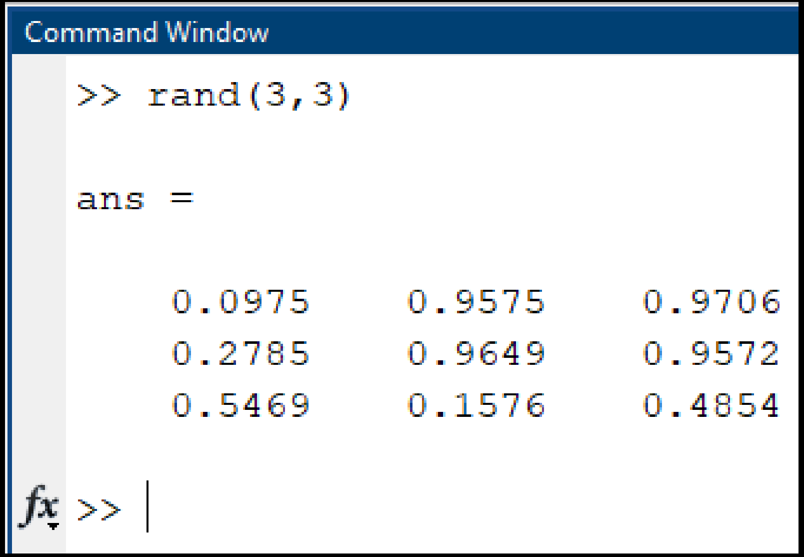
```
>> ones(3)

ans =

     1     1     1
     1     1     1
     1     1     1

>> 
```



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

A =

     1     2     3
     4     5     6
     7     8     9

>> diag(A)

ans =

     1
     5
     9

>> 
```

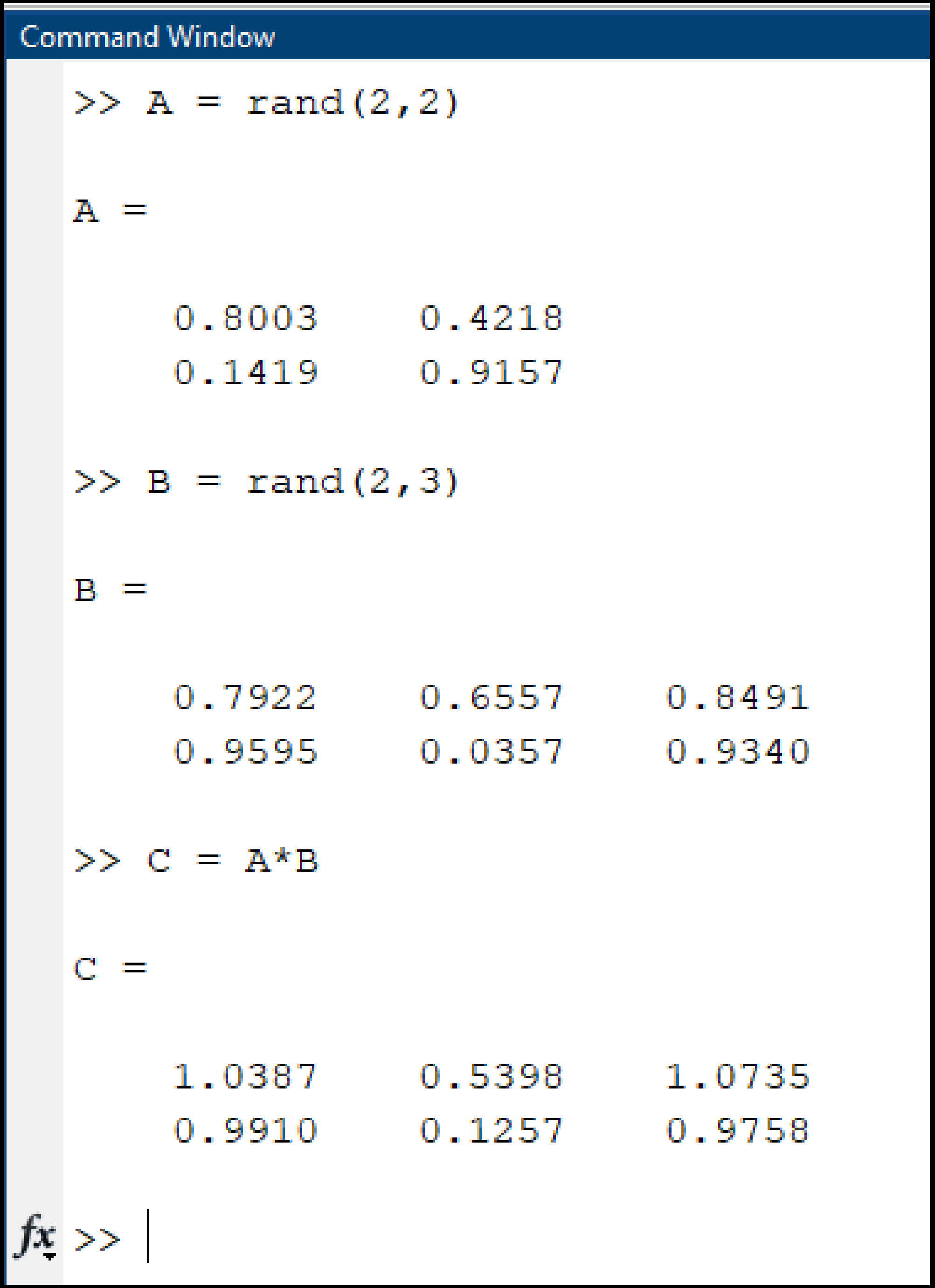


```
>> eye(4)

ans =

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

>> 
```

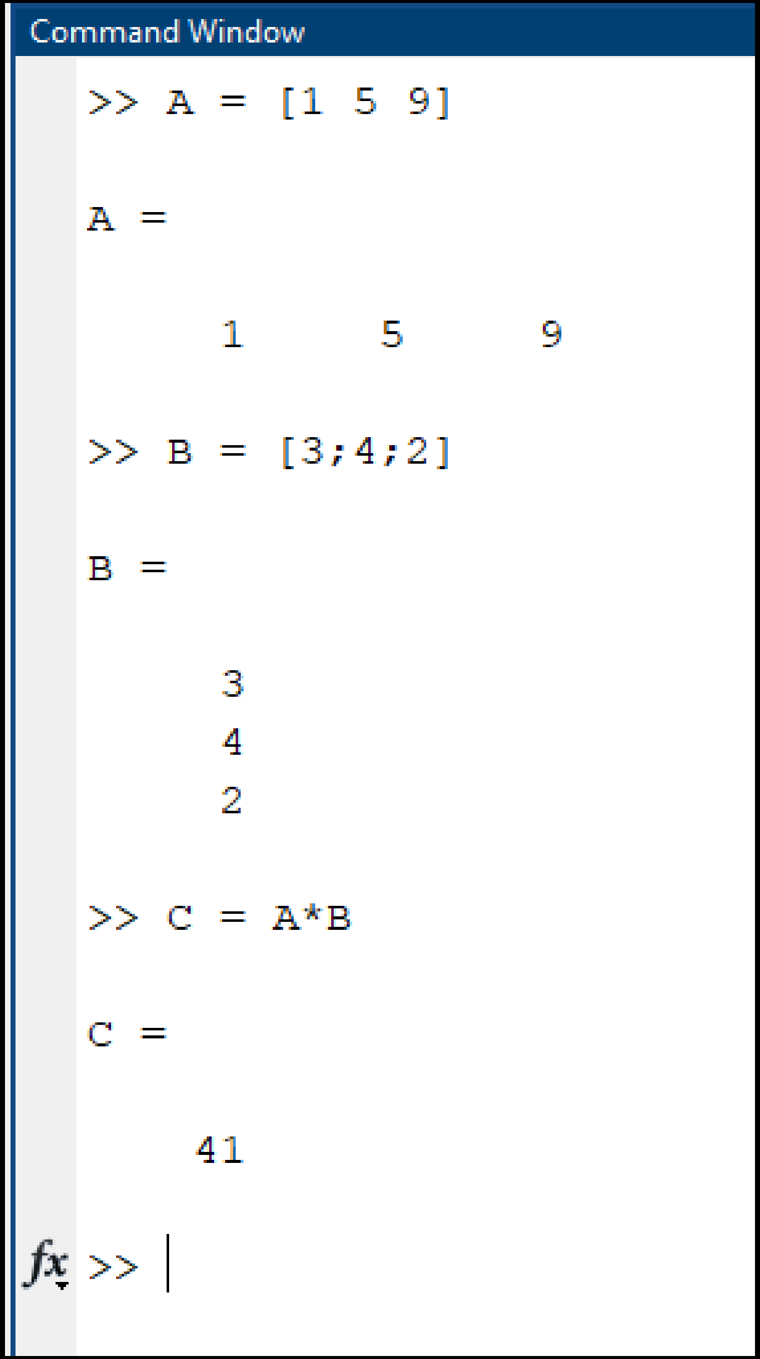


```
>> rand(3,3)

ans =

    0.0975    0.9575    0.9706
    0.2785    0.9649    0.9572
    0.5469    0.1576    0.4854

>> 
```

Arithmetic Operations

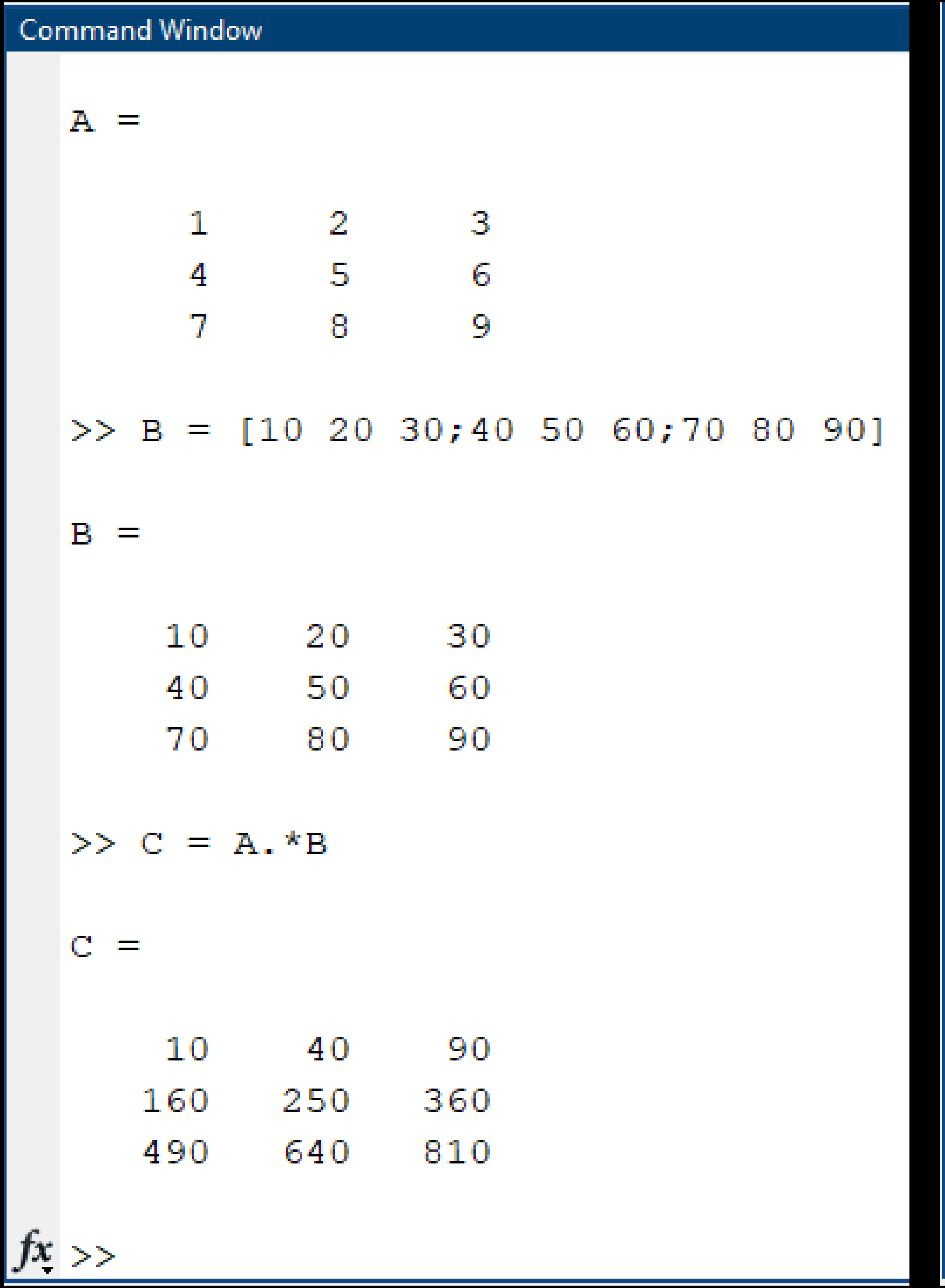


```
>> A = rand(2,2)

A =

    0.8003    0.4218
    0.1419    0.9157

>> B = rand(2,3)

B =

    0.7922    0.6557    0.8491
    0.9595    0.0357    0.9340

>> C = A*B

C =

    1.0387    0.5398    1.0735
    0.9910    0.1257    0.9758

>> 
```

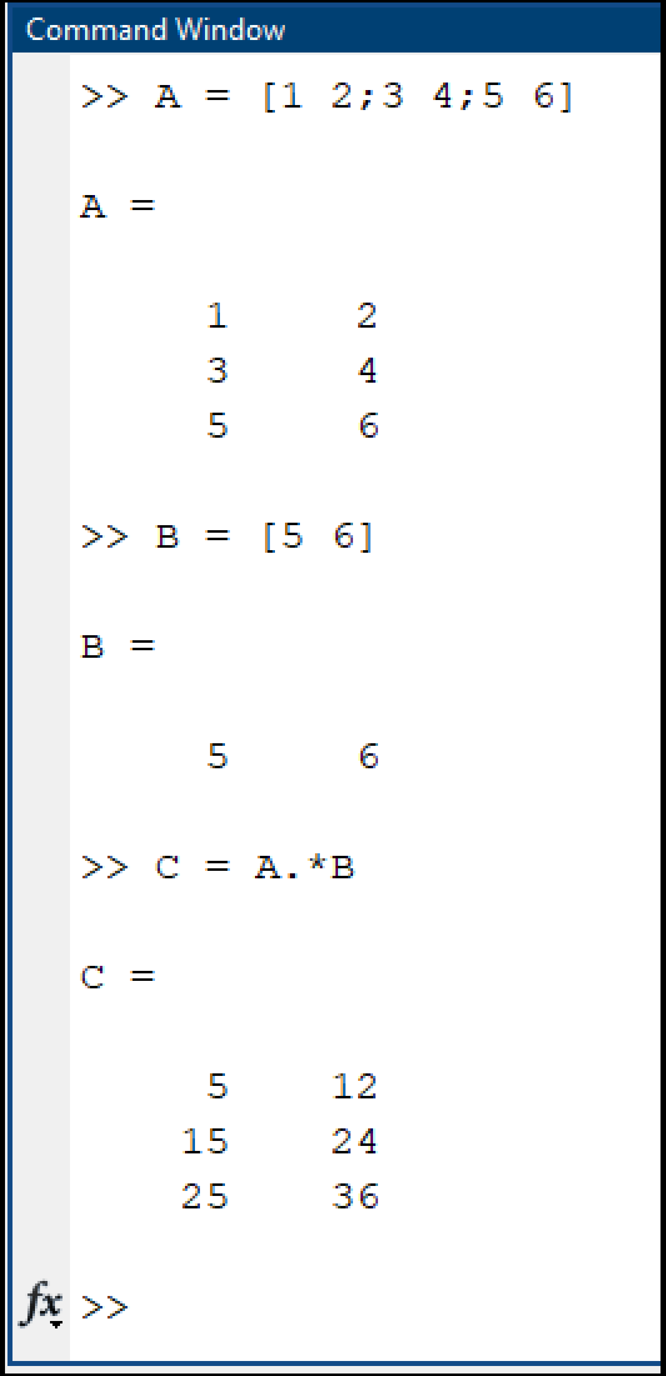


```
>> A = [1 5 9]

A =

     1     5     9

>> B = [3;4;2]

B =

     3
     4
     2

>> C = A*B

C =

    41

>> 
```

Element by element multiplication



```
A =

     1     2     3
     4     5     6
     7     8     9

>> B = [10 20 30;40 50 60;70 80 90]

B =

    10    20    30
    40    50    60
    70    80    90

>> C = A.*B

C =

    10    40    90
   160   250   360
   490   640   810

>>
```



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

A =

     1     2
     3     4
     5     6

>> B = [5 6]

B =

     5     6

>> C = A.*B

C =

     5    12
    15    24
    25    36

>>
```

Solving Linear Equations

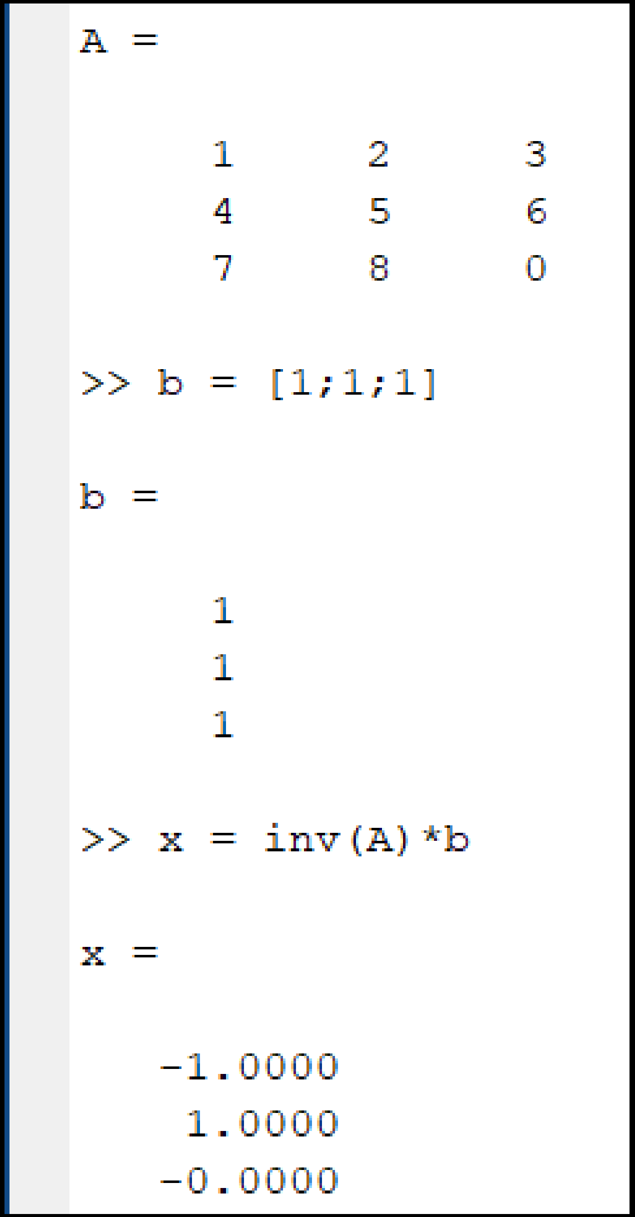
$$Ax = b$$

$$\begin{gathered} x + 2y + 3z = 1 \\ 4x + 5y + 6z = 1 \\ 7x + 8y = 1 \end{gathered}$$

Solution is given by:

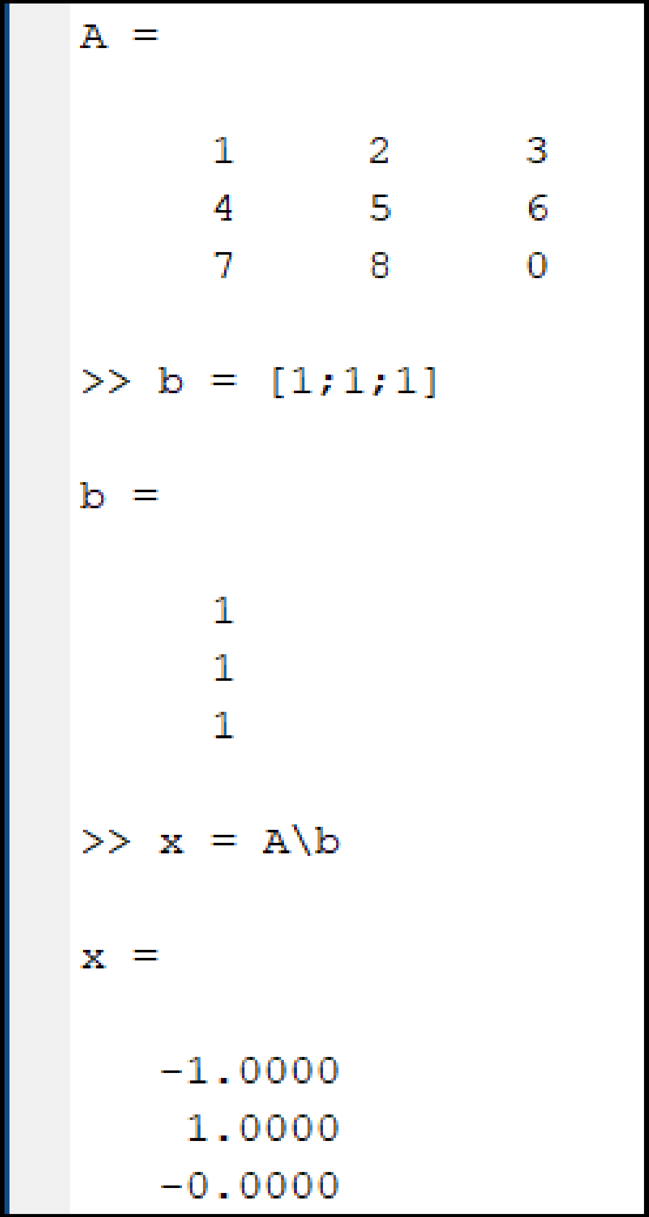
$$x = A^{-1}b$$

Method 1

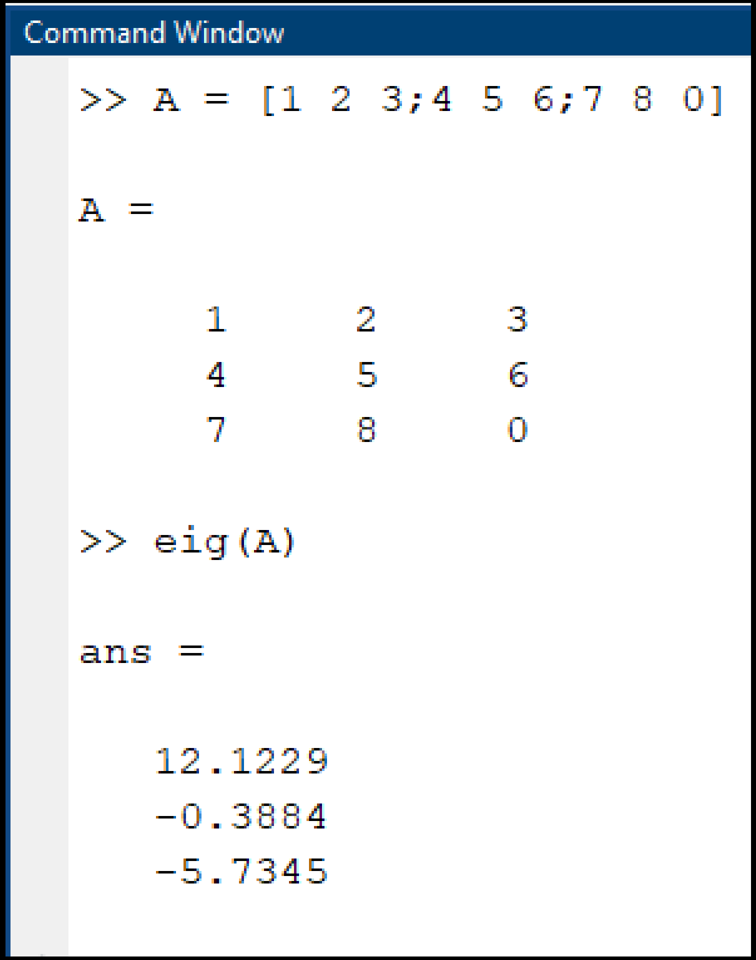


```
A =

     1     2     3
     4     5     6
     7     8     0

>> b = [1;1;1]

b =

     1
     1
     1

>> x = inv(A)*b

x =

   -1.0000
    1.0000
   -0.0000
```

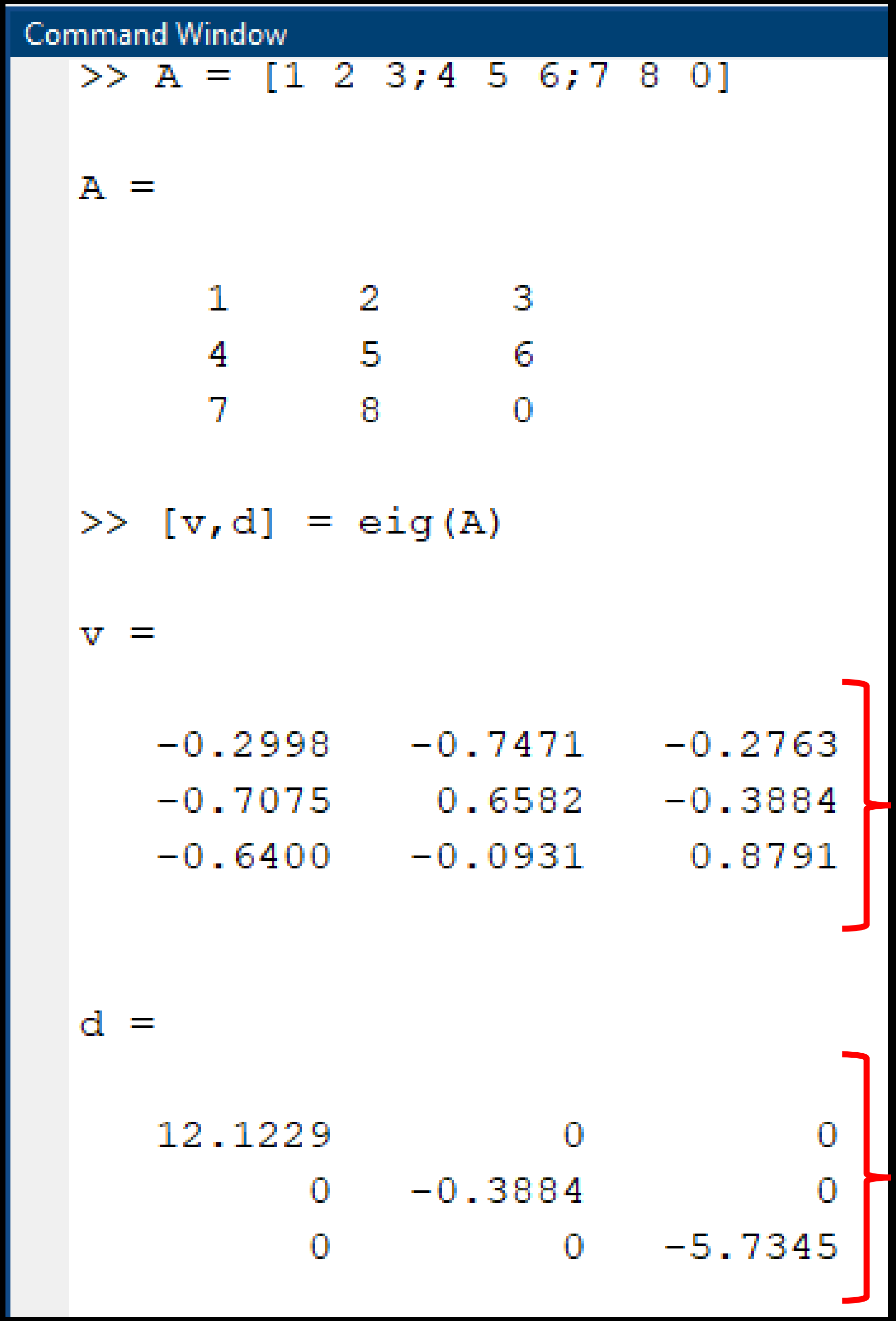
Method 2 Gaussian elimination



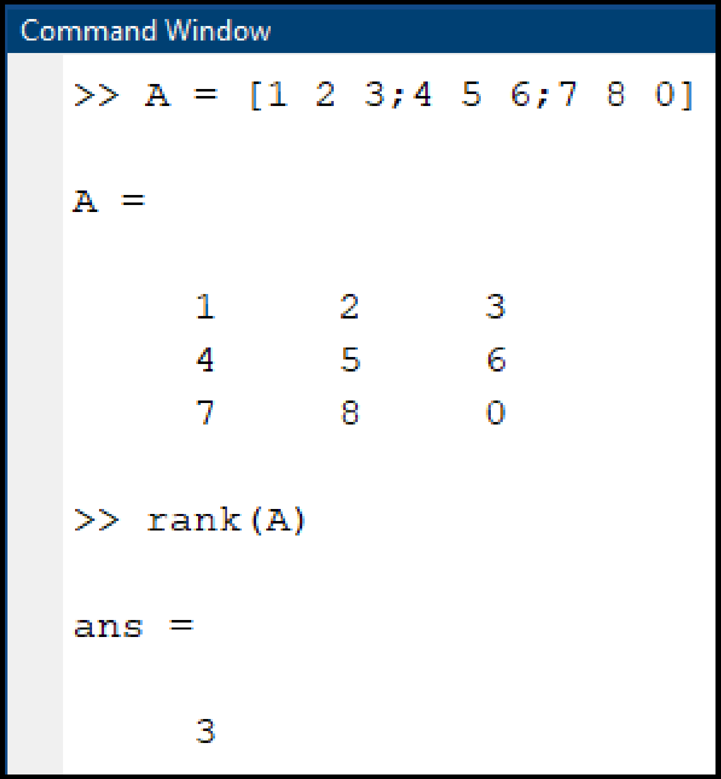
```
A =

     1     2     3
     4     5     6
     7     8     0

>> b = [1;1;1]

b =

     1
     1
     1

>> x = A\b

x =

   -1.0000
    1.0000
   -0.0000
```



```
>> A = [1 2 3;4 5 6;7 8 0]

A =

     1     2     3
     4     5     6
     7     8     0

>> eig(A)

ans =

   12.1229
   -0.3884
   -5.7345
```



```
>> A = [1 2 3;4 5 6;7 8 0]

A =

     1     2     3
     4     5     6
     7     8     0

>> [v,d] = eig(A)

v =

   -0.2998   -0.7471   -0.2763
   -0.7075    0.6582   -0.3884
   -0.6400   -0.0931    0.8791


d =

   12.1229         0         0
         0   -0.3884         0
         0         0   -5.7345
```



```
>> A = [1 2 3;4 5 6;7 8 0]

A =

     1     2     3
     4     5     6
     7     8     0

>> rank(A)

ans =

     3
```

Eigen Vectors

Eigen values