MATLAB Programming

Joe

July 3, 2022

Part I

Basic Knowledge

Outline

1 Linear Algebra
- Linear equations
- Matrix operations
- Exercise

2 Introduction, window and help
- Introduction
- Window
- Help

3 Commands, statements and files
- Basic commands
- Simple drawing
- Function file

Linear equations

$$\begin{cases} 2x_1 + 3x_2 + 1x_3 = 4 \\ 4x_1 + 2x_2 + 3x_3 = 9 \\ 7x_1 + 0x_2 + 1x_3 = 1 \end{cases}$$

$\mathbf{Ax} = \mathbf{b}$

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 1 \\ 4 & 2 & 3 \\ 7 & 0 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 4 \\ 17 \\ 1 \end{bmatrix}$$

Transpose

$$\mathbf{A}^{\mathrm{T}} = \begin{bmatrix} 2 & 4 & 7 \\ 3 & 2 & 0 \\ 1 & 3 & 1 \end{bmatrix}, \quad \mathbf{x}^{\mathrm{T}} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}, \quad \mathbf{b}^{\mathrm{T}} = \begin{bmatrix} 4 & 17 & 1 \end{bmatrix}$$

Linear equations

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2 \\ \vdots \qquad\quad \vdots \\ a_{m,1}x_1 + a_{m,2}x_2 + \cdots + a_{m,n}x_n = b_m \end{cases}$$

$\mathbf{Ax} = \mathbf{b}$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Matrices and vectors

Row vector and column vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}, \quad \mathbf{x}^{\mathrm{T}} = \begin{bmatrix} x_1 & x_2 & \vdots & x_n \end{bmatrix}$$

Matrix and transpose

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}, \quad \mathbf{A}^{\mathrm{T}} = \begin{bmatrix} a_{1,1} & a_{2,1} & \cdots & a_{m,1} \\ a_{1,2} & a_{2,2} & \cdots & a_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1,n} & a_{2,n} & \cdots & a_{m,n} \end{bmatrix}$$

Add, subtract and multiply

Addition and subtraction

$$\begin{bmatrix}1 & 3 & 1\\ 1 & 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0 & 5\\ 7 & 5 & 0\end{bmatrix} = \begin{bmatrix}1+0 & 3+0 & 1+5\\ 1+7 & 0+5 & 0+0\end{bmatrix} = \begin{bmatrix}1 & 3 & 6\\ 8 & 5 & 0\end{bmatrix}$$

Multiply

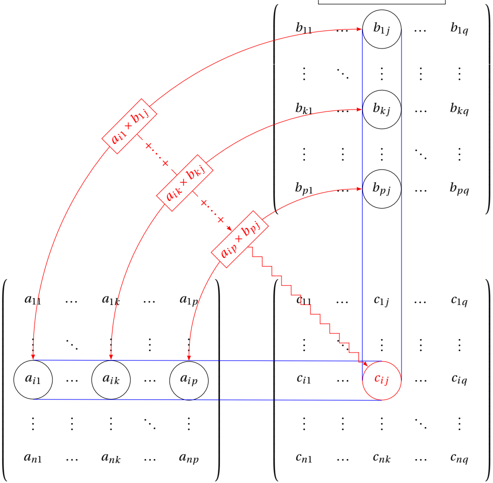
$$2 \cdot \begin{bmatrix}1 & 8 & -3\\ 4 & -2 & 5\end{bmatrix} = \begin{bmatrix}2\cdot 1 & 2\cdot 8 & 2\cdot(-3)\\ 2\cdot 4 & 2\cdot(-2) & 2\cdot 5\end{bmatrix} = \begin{bmatrix}2 & 16 & -6\\ 8 & -4 & 10\end{bmatrix}$$

Matrix multiplication

$$\begin{bmatrix}1 & 0 & 2\\ -1 & 3 & 1\end{bmatrix} \times \begin{bmatrix}3 & 1\\ 2 & 1\\ 1 & 0\end{bmatrix} = \begin{bmatrix}5 & 1\\ 4 & 2\end{bmatrix}$$

Matrix multiplication

B : p rows q columns


b_{11} … b_{1j} … b_{1q}
b_{k1} … b_{kj} … b_{kq}
b_{p1} … b_{pj} … b_{pq}
$a_{i1} \times b_{1j}$
$a_{ik} \times b_{kj}$
$a_{ip} \times b_{pj}$
+ … +
+ … +
a_{11} … a_{1k} … a_{1p}
a_{i1} … a_{ik} … a_{ip}
a_{n1} … a_{nk} … a_{np}
c_{11} … c_{1j} … c_{1q}
c_{i1} … c_{ij} … c_{iq}
c_{n1} … c_{nk} … c_{nq}

A : n rows p columns

C = A × B : n rows q columns

Exercise

Write the system of equations in matrix multiplication form

$$\begin{cases} 4x + 5y = 3 \\ 1x + 2y = 15 \\ 3x + 1y = 12 \end{cases} \qquad \mathbf{Ax} = \mathbf{b} \quad \mathbf{A} = ?\ \mathbf{x} = ?\ \mathbf{b} = ?$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = ?, \qquad \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = ?$$

Outline

1 Linear Algebra

- Linear equations
- Matrix operations
- Exercise

2 Introduction, window and help

- Introduction
- Window
- Help

3 Commands, statements and files

- Basic commands
- Simple drawing
- Function file

Introduction

History

- MATLAB **MAT**rix **LAB**oratory;
- Clever Moler 1980, the original intention was to solve the matrix operation problem of the "linear algebra" course;
- MathWorks 1984

characteristic

- Algorithm development, data visualization, data analysis, and numerical calculation.
- User interface and calling programs written in other languages.
- Numerous additional toolboxes are suitable for applications in different fields.

Window

- Command window
- Script window
- Graphics window

Help

Help

- `>> help functionname`
- `>> lookfor keyword`

Internet resources

- Mathworks 文件交流中心: ▸ Mathworks
- Github 代码托管网站: ▸ Github
- Octive 在线练习网站: ▸ Octive-online

Outline

1 Linear Algebra

- Linear equations
- Matrix operations
- Exercise

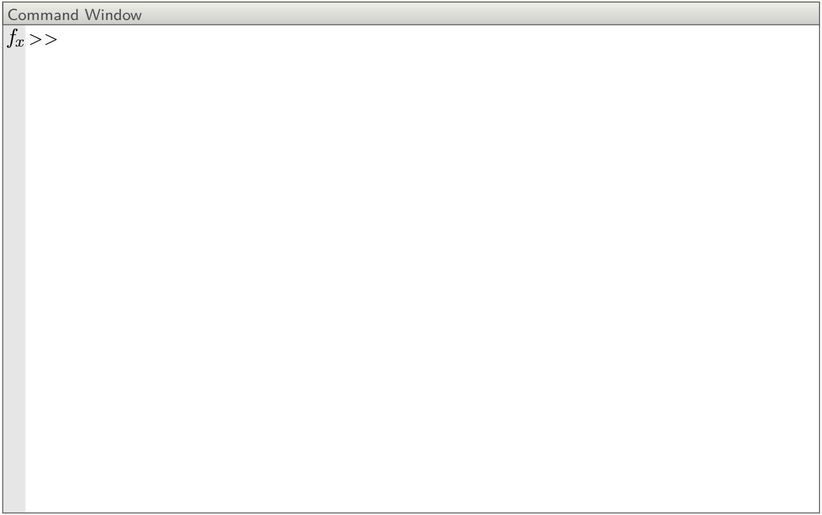
2 Introduction, window and help

- Introduction
- Window
- Help

3 Commands, statements and files

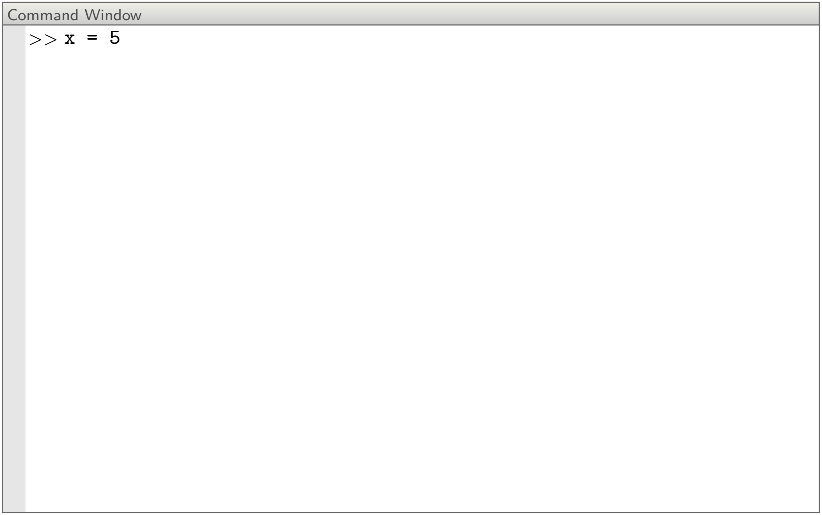
- Basic commands
- Simple drawing
- Function file

Real, complex, row vector, column vector, matrix assignment



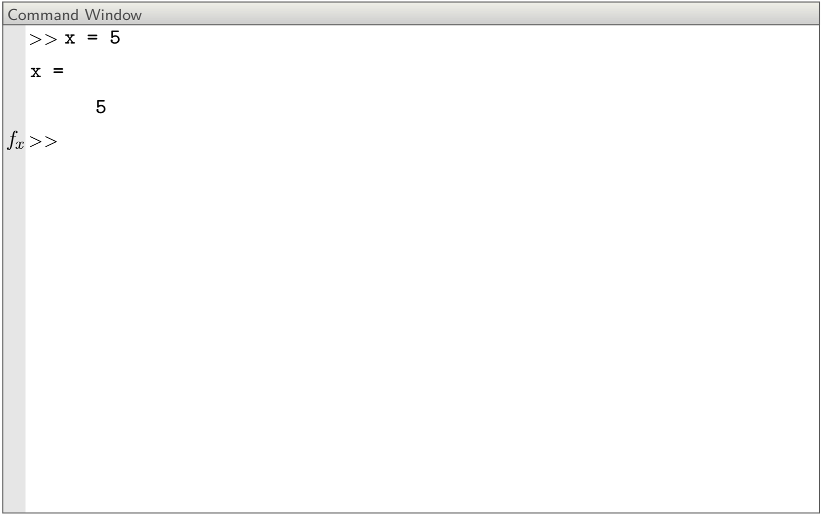
```
>>
```

Real, complex, row vector, column vector, matrix assignment



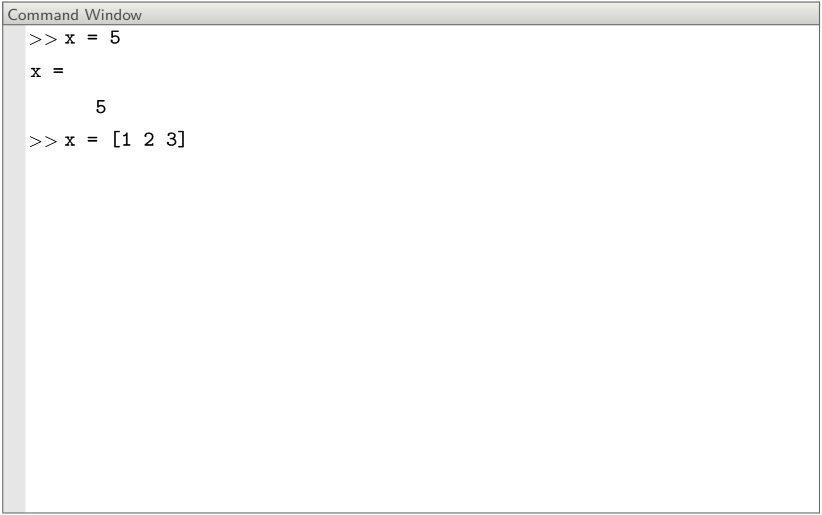
```
>> x = 5
```

Real, complex, row vector, column vector, matrix assignment



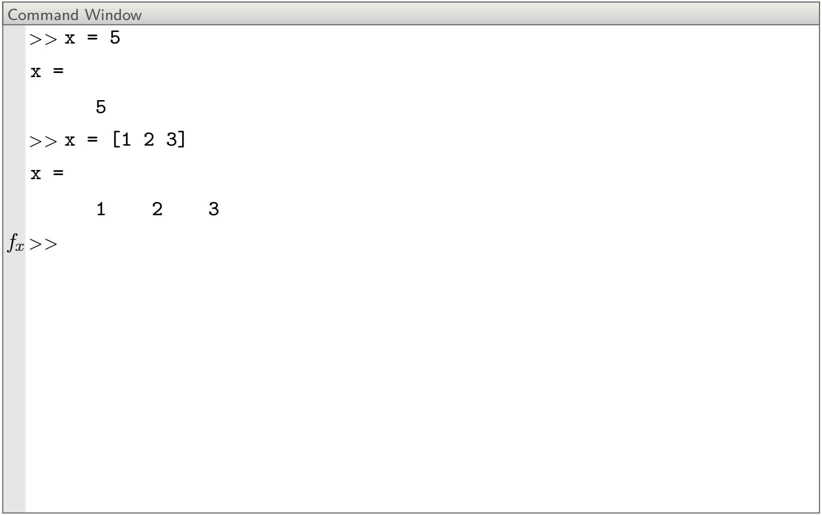
```
>> x = 5

x =

       5

fx >>
```

Real, complex, row vector, column vector, matrix assignment



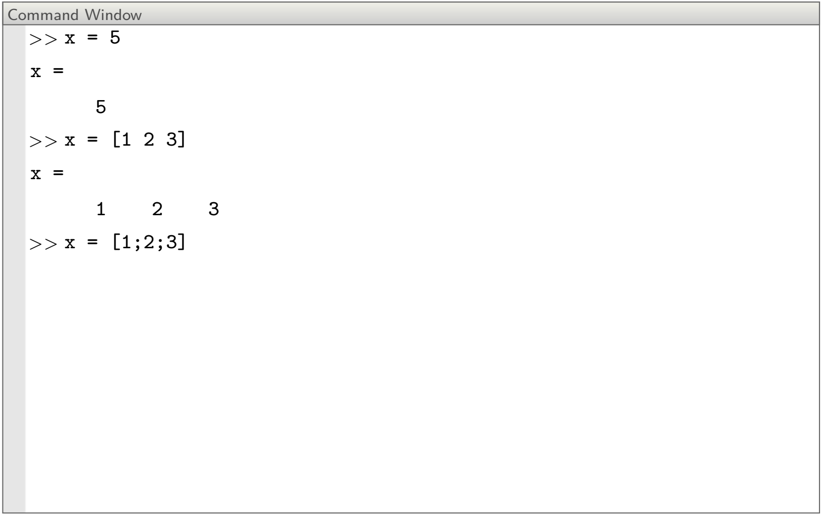
```
>> x = 5

x =

     5

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

Real, complex, row vector, column vector, matrix assignment



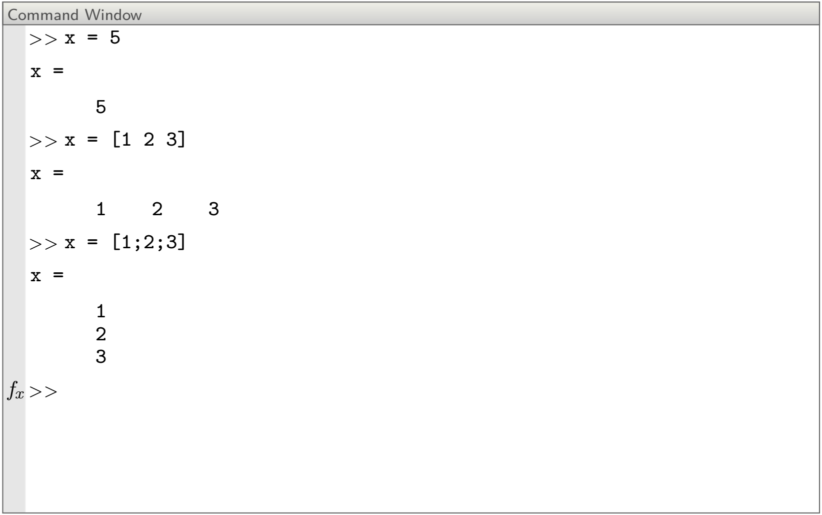
```
>> x = 5

x =

     5

>> x = [1 2 3]

x =

     1     2     3

fx >>
```

Real, complex, row vector, column vector, matrix assignment



```
>> x = 5

x =

     5

>> x = [1 2 3]

x =

     1     2     3

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

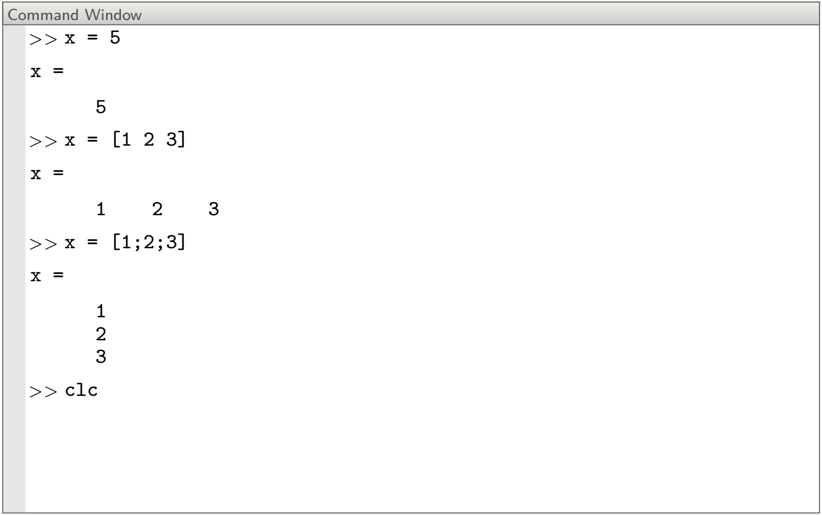
Real, complex, row vector, column vector, matrix assignment



Command Window

```
>> x = 5

x =

     5

>> x = [1 2 3]

x =

     1     2     3

>> x = [1;2;3]

x =

     1
     2
     3

fx >>
```

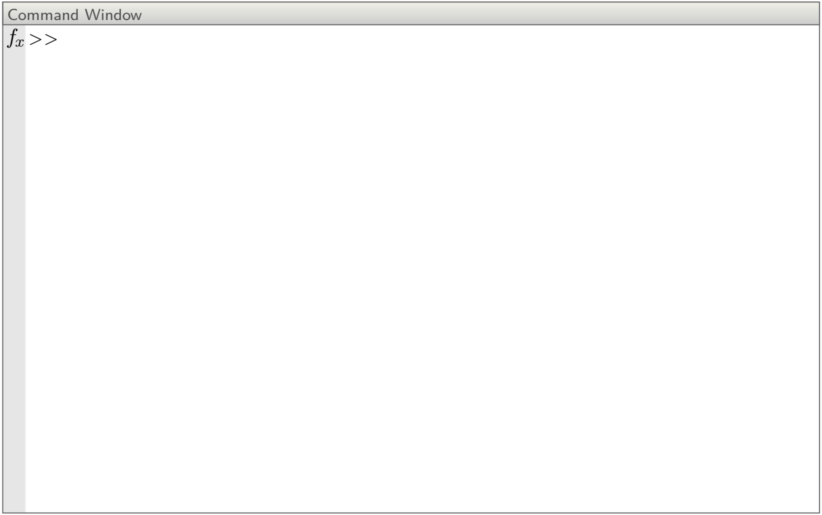
Real, complex, row vector, column vector, matrix assignment



Command Window

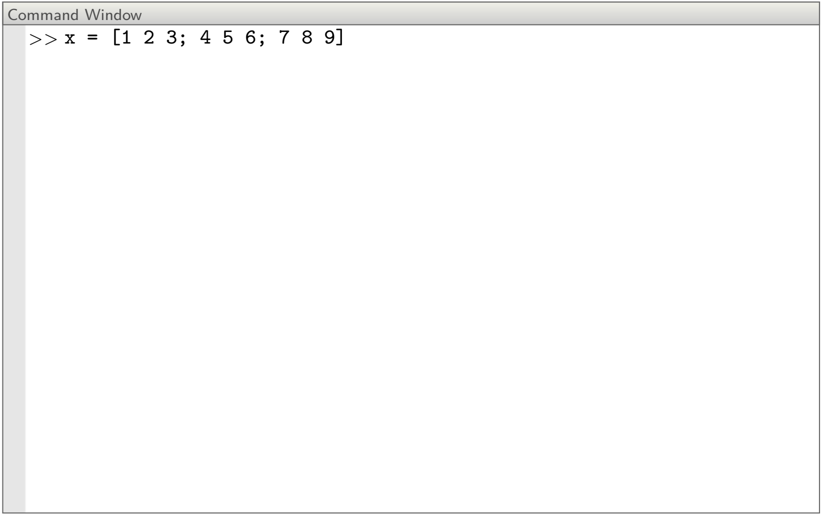
```
>> x = 5

x =

     5

>> x = [1 2 3]

x =

     1     2     3

>> x = [1;2;3]

x =

     1
     2
     3

>> clc
```

Real, complex, row vector, column vector, matrix assignment



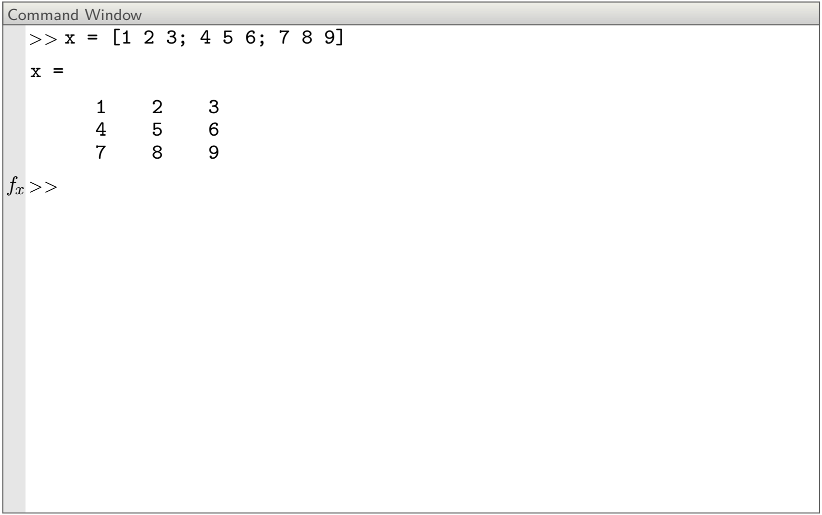
```
fx >>
```

Real, complex, row vector, column vector, matrix assignment



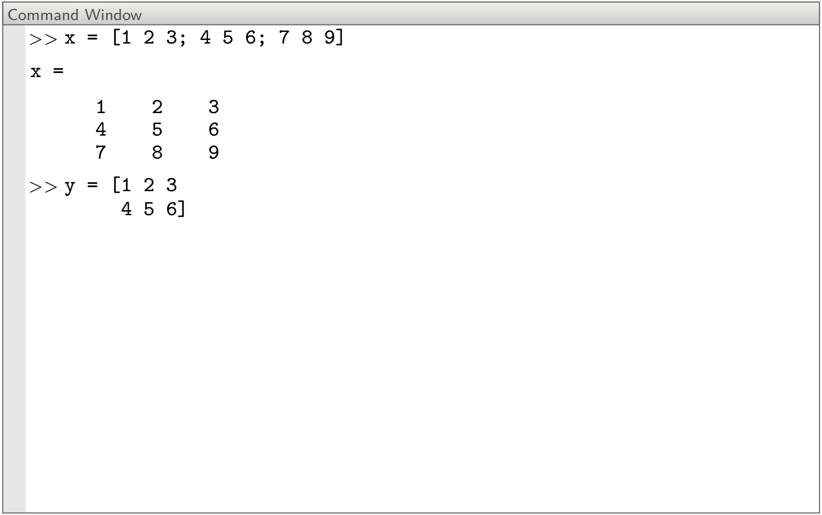
```
>> x = [1 2 3; 4 5 6; 7 8 9]
```

Real, complex, row vector, column vector, matrix assignment



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

x =

     1     2     3
     4     5     6
     7     8     9

fx >>
```

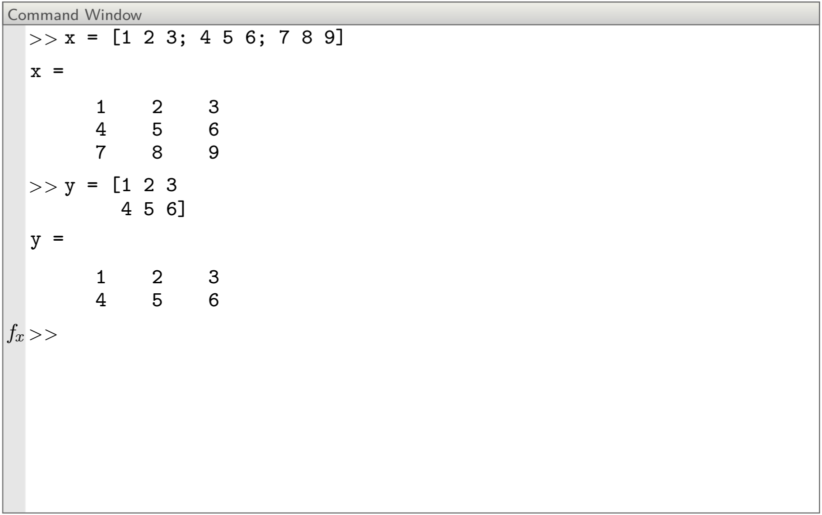
Real, complex, row vector, column vector, matrix assignment



Command Window

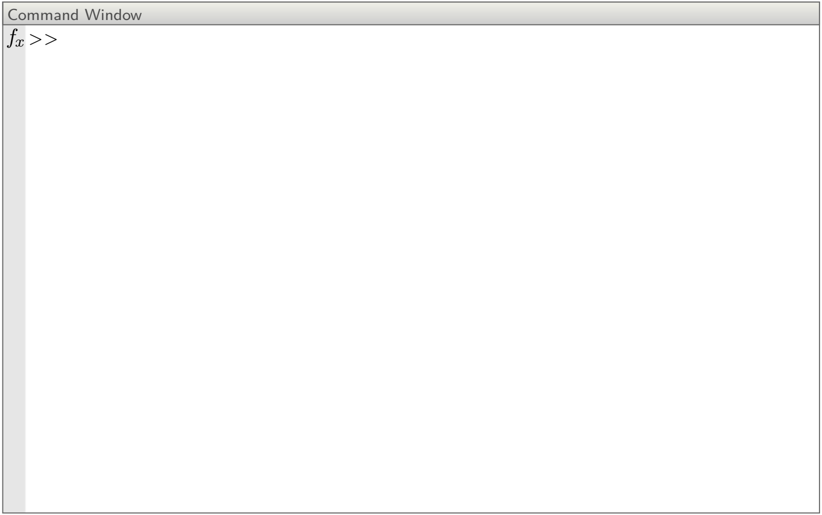
```
>> x = [1 2 3; 4 5 6; 7 8 9]

x =

     1     2     3
     4     5     6
     7     8     9

>> y = [1 2 3
        4 5 6]
```

Real, complex, row vector, column vector, matrix assignment



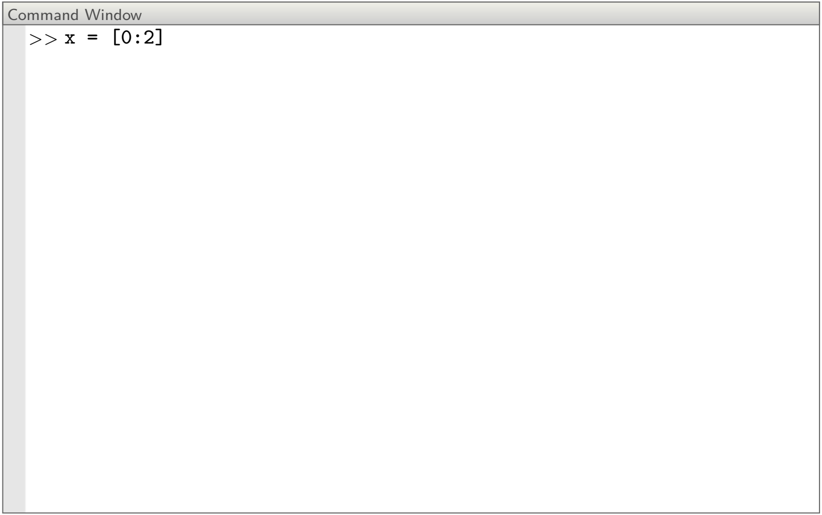
```
>> x = [1 2 3; 4 5 6; 7 8 9]

x =

     1     2     3
     4     5     6
     7     8     9

>> y = [1 2 3
        4 5 6]

y =

     1     2     3
     4     5     6

fx >>
```

General assignment method of vector



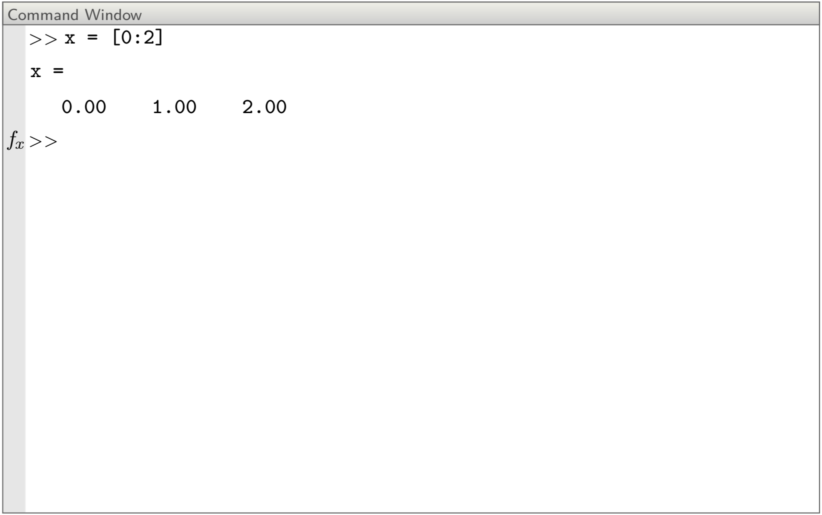
```
fx >>
```

General assignment method of vector



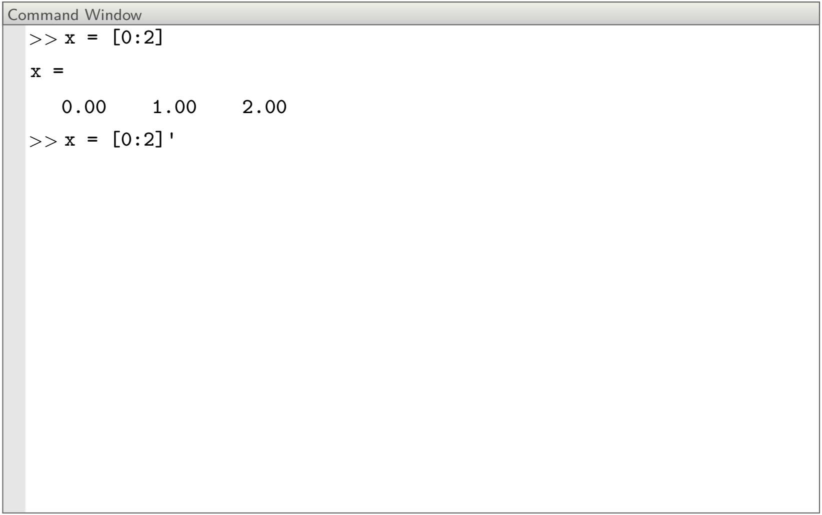
```
>> x = [0:2]
```

General assignment method of vector



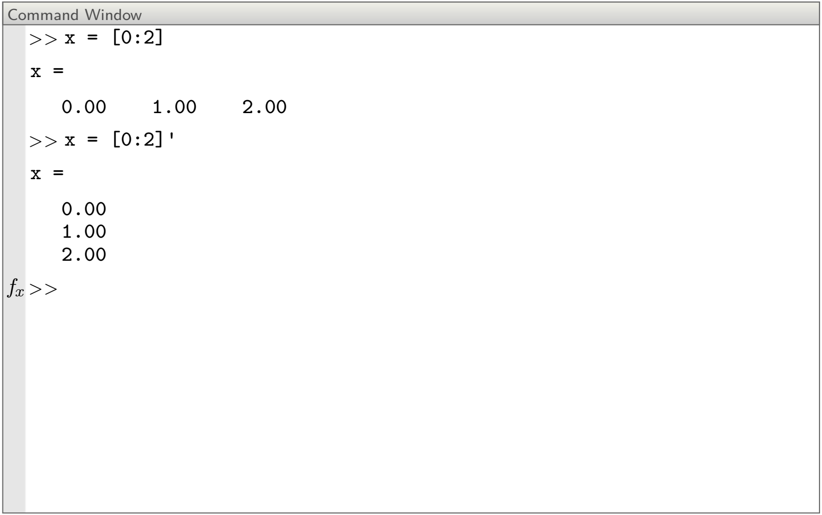
```
>> x = [0:2]

x =

    0.00     1.00     2.00

fx >>
```

General assignment method of vector



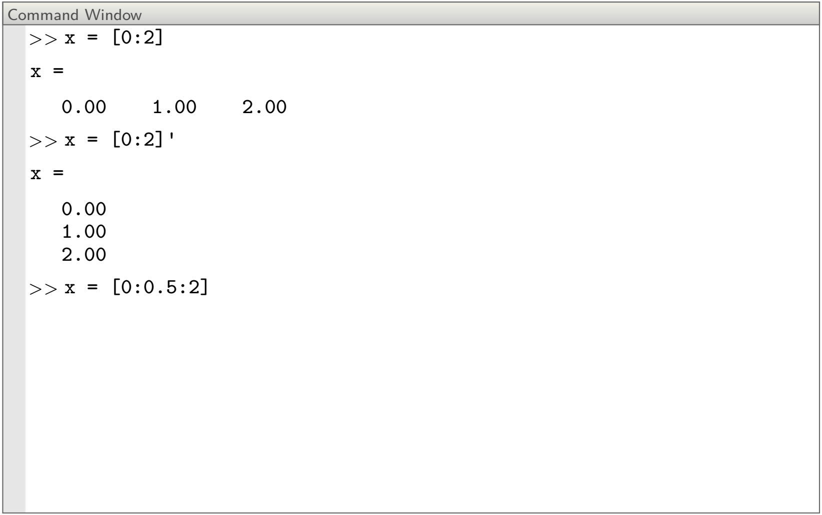
```
>> x = [0:2]

x =

   0.00     1.00     2.00

>> x = [0:2]'
```

General assignment method of vector



```
>> x = [0:2]

x =

   0.00     1.00     2.00

>> x = [0:2]'

x =

   0.00
   1.00
   2.00

fx >>
```

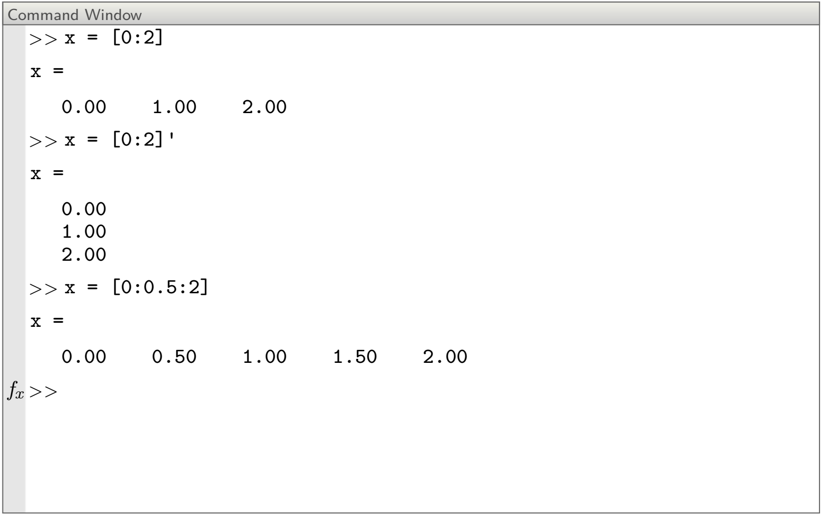
General assignment method of vector



Command Window

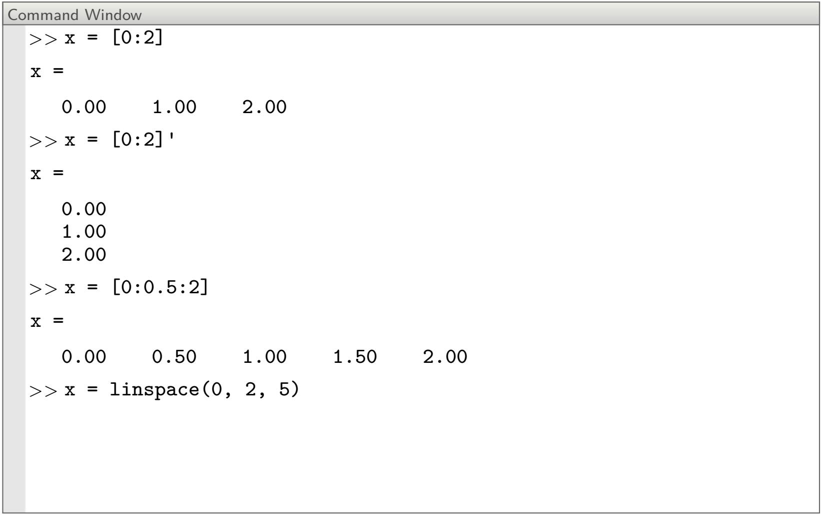
```
>> x = [0:2]

x =

   0.00     1.00     2.00

>> x = [0:2]'

x =

   0.00
   1.00
   2.00

>> x = [0:0.5:2]
```

General assignment method of vector



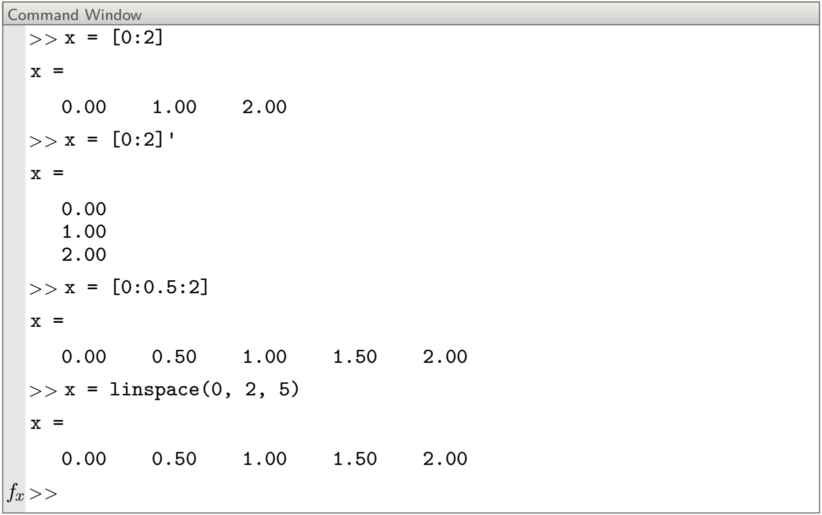
```
>> x = [0:2]

x =

   0.00     1.00     2.00

>> x = [0:2]'

x =

   0.00
   1.00
   2.00

>> x = [0:0.5:2]

x =

   0.00     0.50     1.00     1.50     2.00

fx >>
```

General assignment method of vector



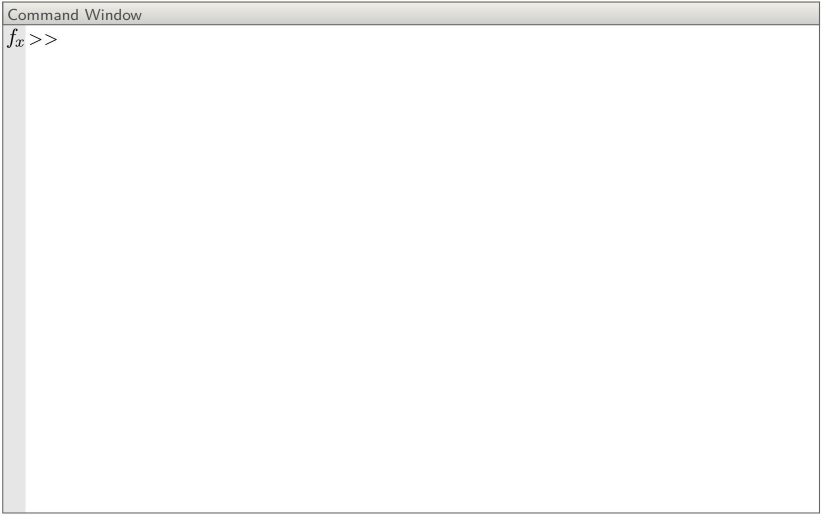
```
>> x = [0:2]

x =

   0.00     1.00     2.00

>> x = [0:2]'

x =

   0.00
   1.00
   2.00

>> x = [0:0.5:2]

x =

   0.00     0.50     1.00     1.50     2.00

>> x = linspace(0, 2, 5)
```

General assignment method of vector



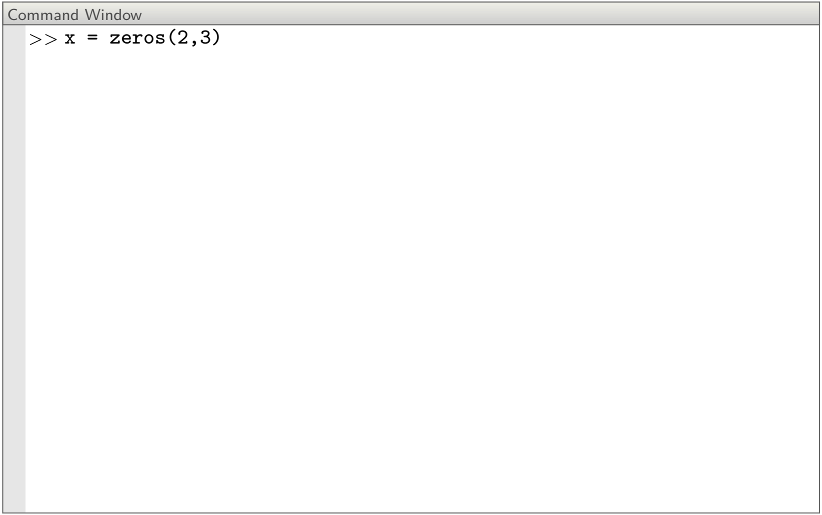
```
>> x = [0:2]

x =

   0.00     1.00     2.00

>> x = [0:2]'

x =

   0.00
   1.00
   2.00

>> x = [0:0.5:2]

x =

   0.00     0.50     1.00     1.50     2.00

>> x = linspace(0, 2, 5)

x =

   0.00     0.50     1.00     1.50     2.00

fx >>
```

Common matrix



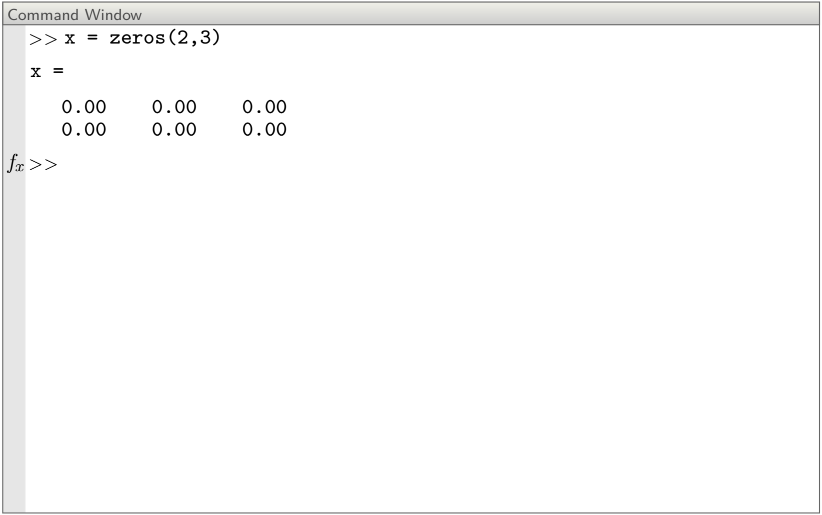
```
fx >>
```

Common matrix



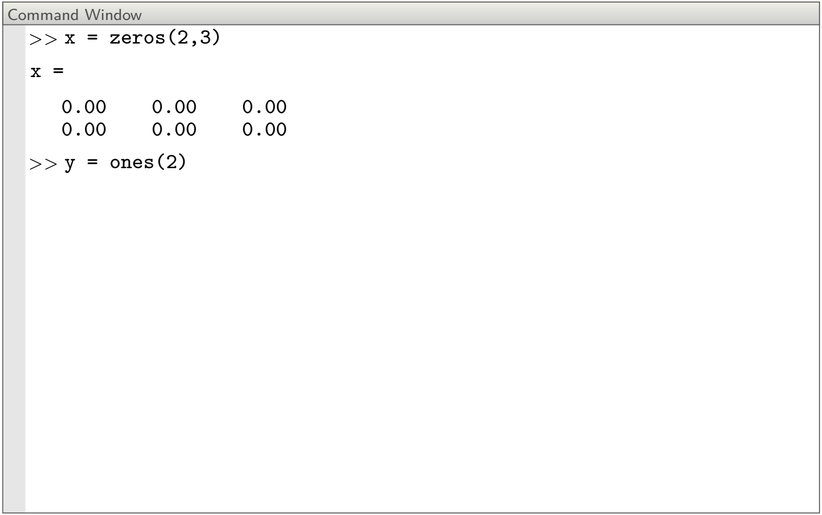
```
>> x = zeros(2,3)
```

Common matrix



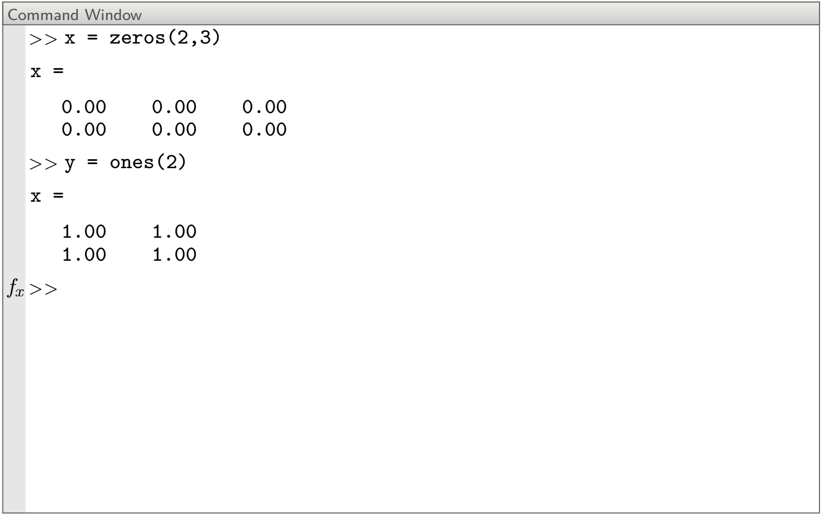
```
>> x = zeros(2,3)

x =

   0.00     0.00     0.00
   0.00     0.00     0.00

fx >>
```

Common matrix



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

x =

   0.00     0.00     0.00
   0.00     0.00     0.00

>> y = ones(2)
```

Common matrix



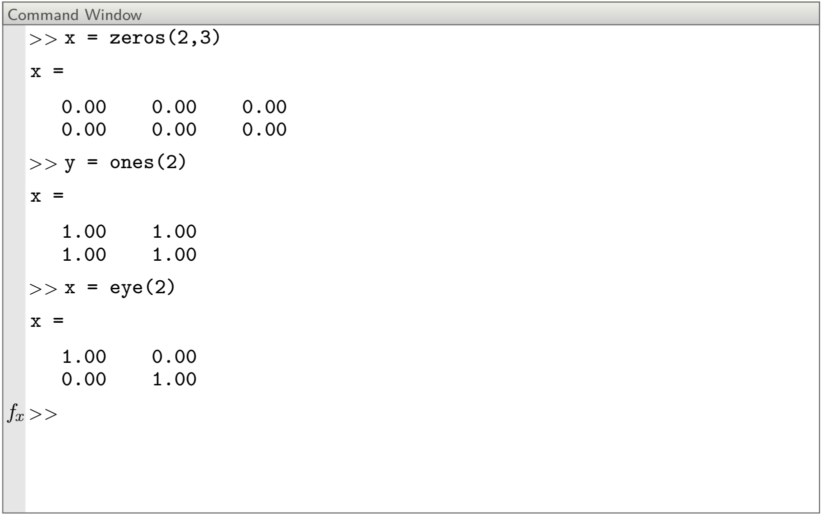
```
>> x = zeros(2,3)

x =

   0.00    0.00    0.00
   0.00    0.00    0.00

>> y = ones(2)

x =

   1.00    1.00
   1.00    1.00

fx >>
```

Common matrix

Command Window

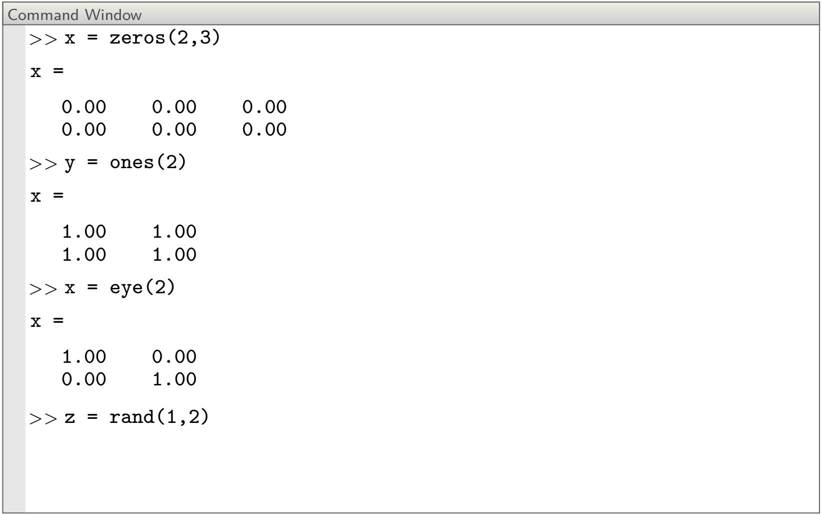
```
>> x = zeros(2,3)

x =

   0.00     0.00     0.00
   0.00     0.00     0.00

>> y = ones(2)

x =

   1.00     1.00
   1.00     1.00

>> x = eye(2)
```

Common matrix



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

x =

   0.00    0.00    0.00
   0.00    0.00    0.00

>> y = ones(2)

x =

   1.00    1.00
   1.00    1.00

>> x = eye(2)

x =

   1.00    0.00
   0.00    1.00

fx >>
```

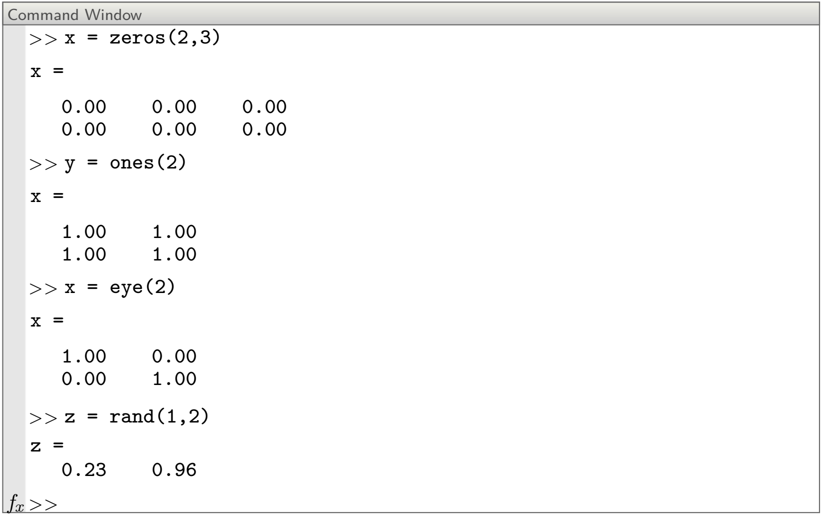
Common matrix



Command Window

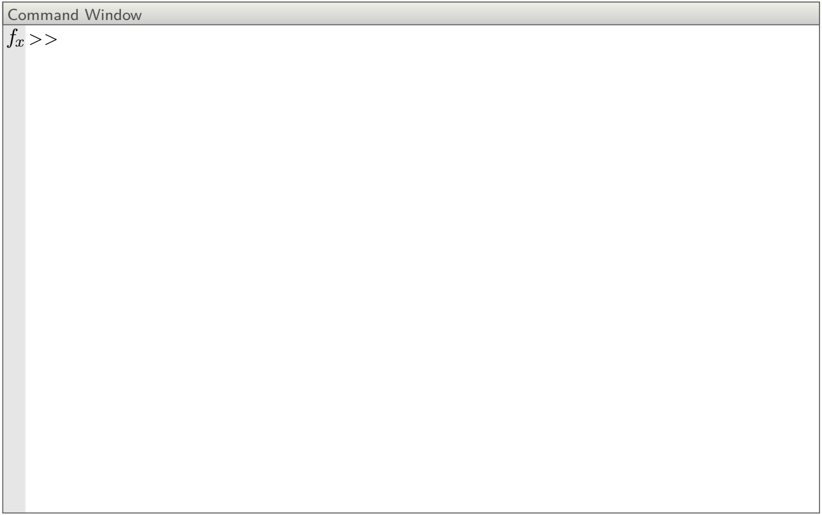
```
>> x = zeros(2,3)

x =

   0.00    0.00    0.00
   0.00    0.00    0.00

>> y = ones(2)

x =

   1.00    1.00
   1.00    1.00

>> x = eye(2)

x =

   1.00    0.00
   0.00    1.00

>> z = rand(1,2)
```

Common matrix



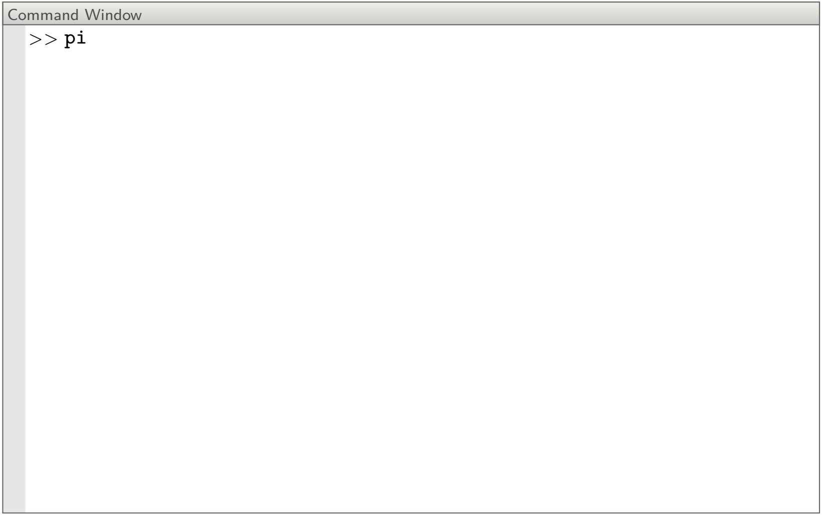
```
>> x = zeros(2,3)

x =

   0.00     0.00     0.00
   0.00     0.00     0.00

>> y = ones(2)

x =

   1.00     1.00
   1.00     1.00

>> x = eye(2)

x =

   1.00     0.00
   0.00     1.00

>> z = rand(1,2)

z =
   0.23     0.96

fx >>
```

Fixed variable



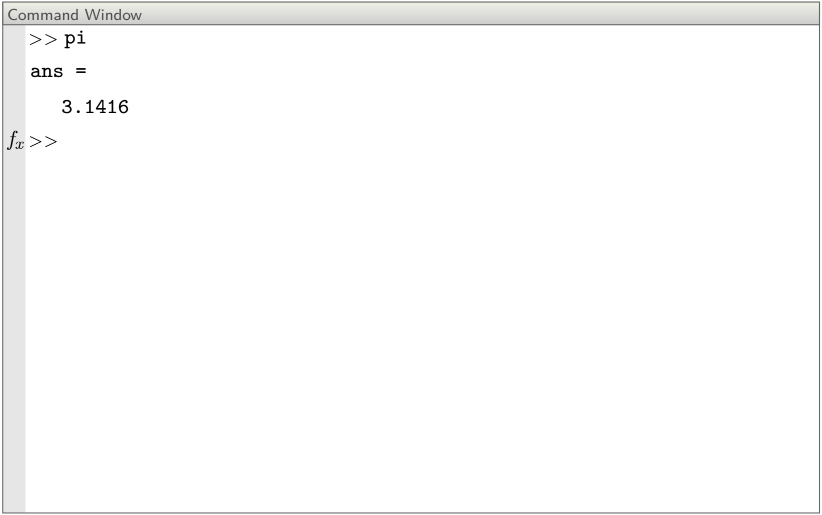
```
>>
```

Fixed variable



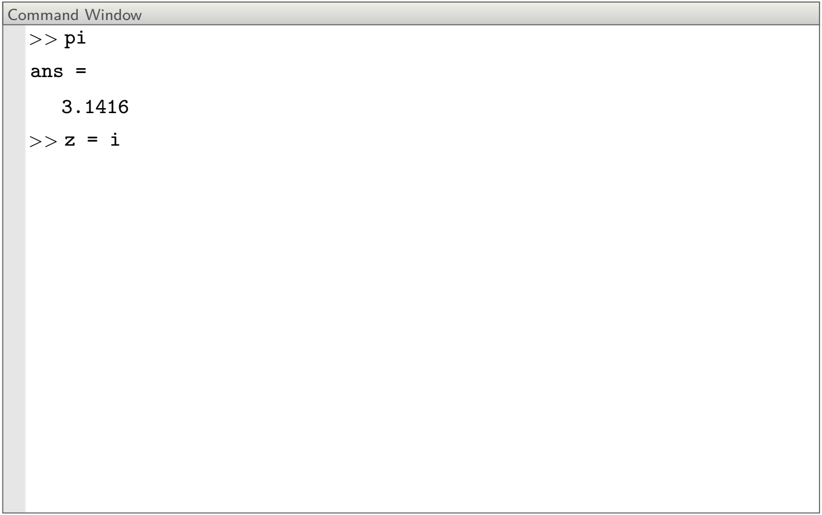
```
>> pi
```

Fixed variable



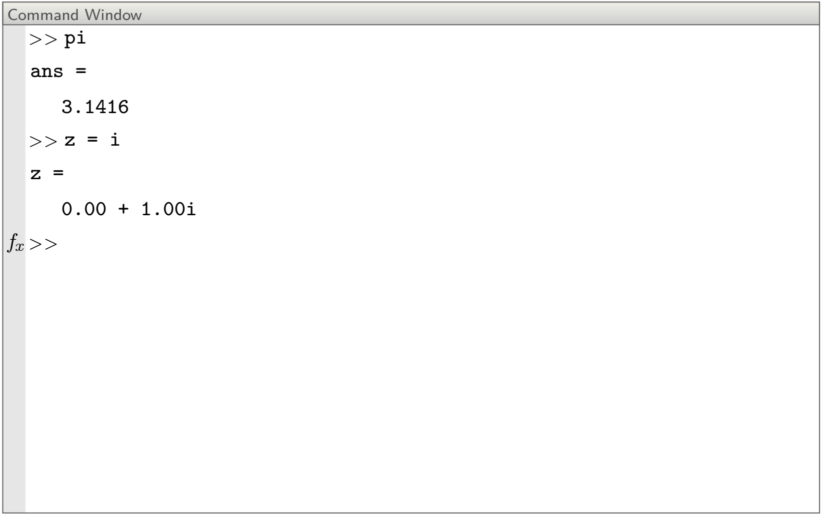
```
>> pi
ans =
   3.1416
fx >>
```

Fixed variable



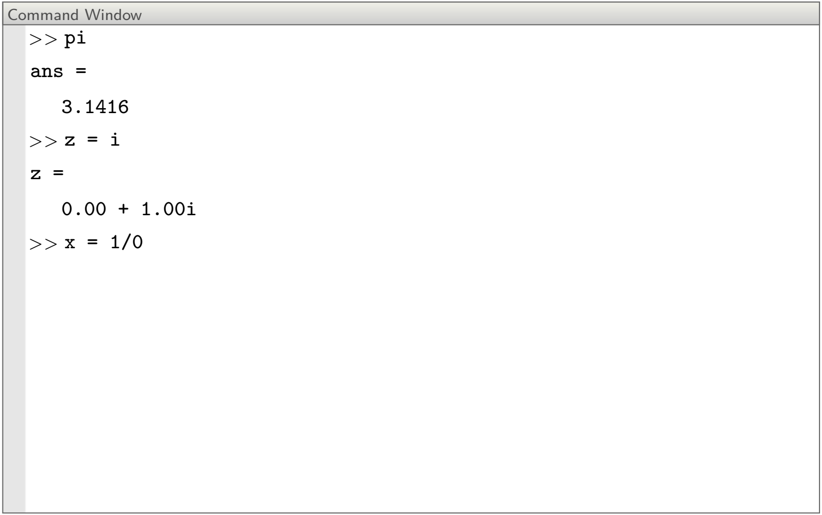
```
>> pi
ans =
   3.1416
>> z = i
```

Fixed variable



```
>> pi
ans =
   3.1416
>> z = i
z =
   0.00 + 1.00i
fx >>
```

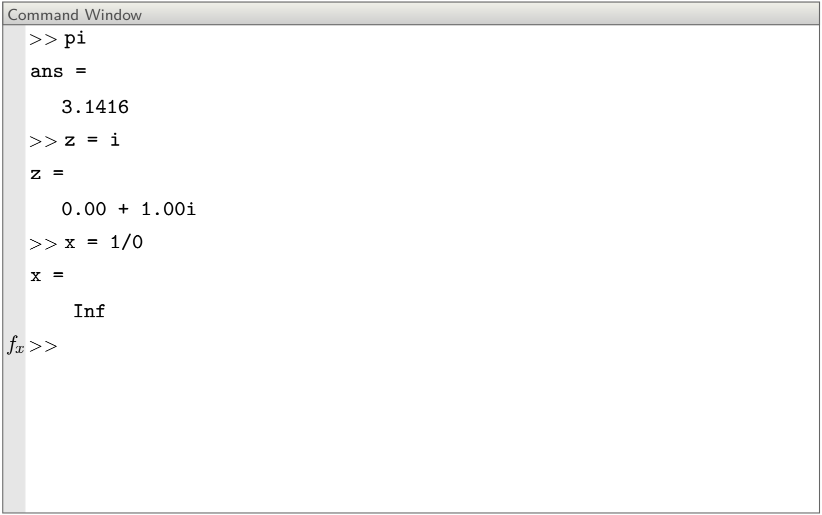
Fixed variable



Command Window

```
>> pi
ans =
   3.1416
>> z = i
z =
   0.00 + 1.00i
>> x = 1/0
```

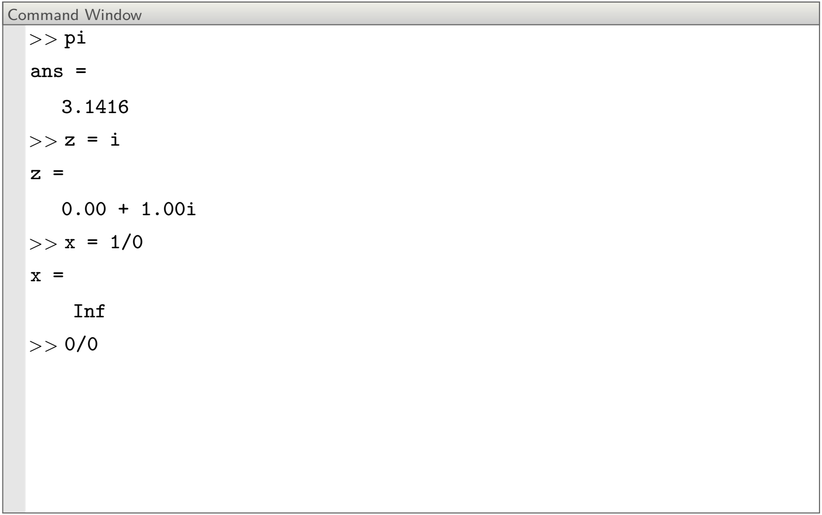
Fixed variable



Command Window

```
>> pi
ans =

   3.1416

>> z = i
z =

   0.00 + 1.00i

>> x = 1/0
x =

    Inf

fx >>
```

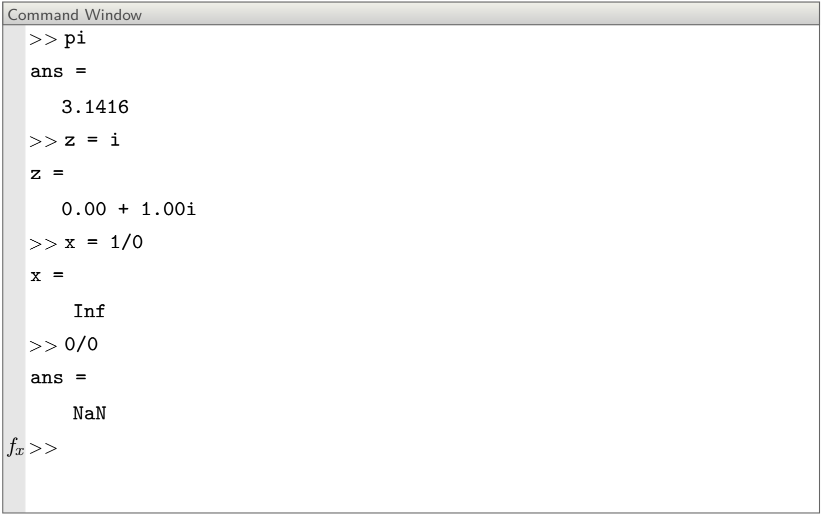
Fixed variable



Command Window

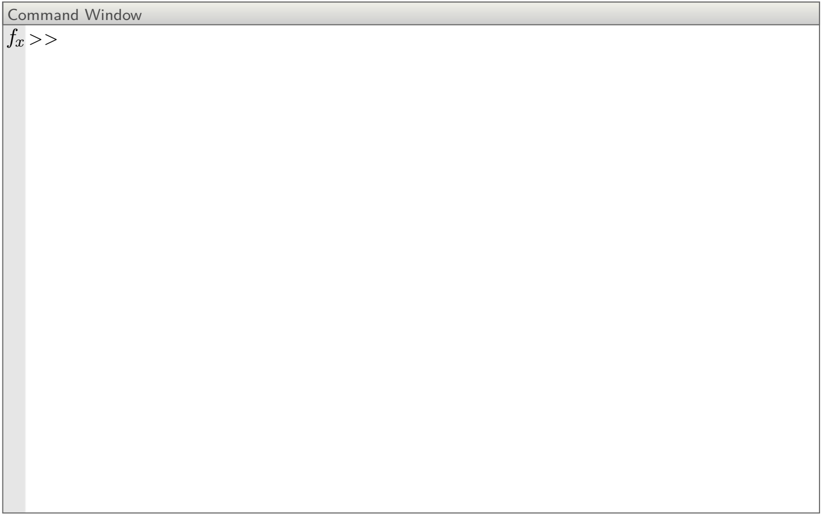
```
>> pi
ans =

   3.1416
>> z = i
z =

   0.00 + 1.00i
>> x = 1/0
x =

    Inf
>> 0/0
```

Fixed variable



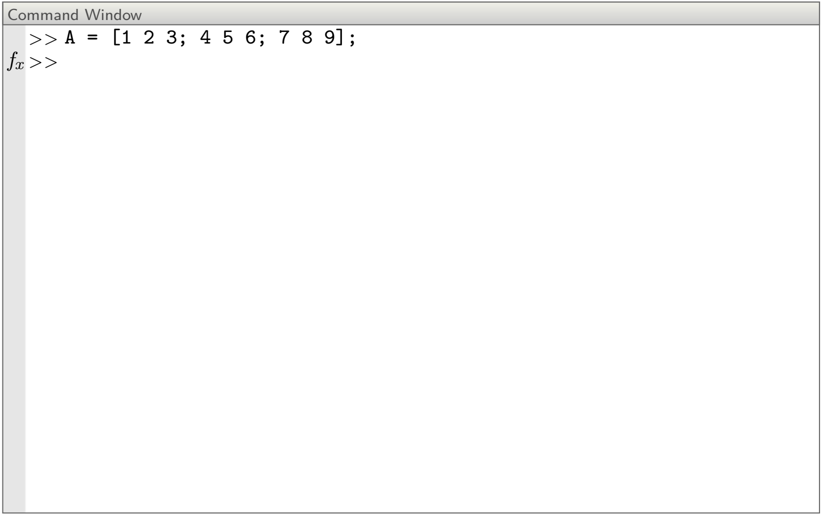
```
>> pi
ans =
   3.1416
>> z = i
z =
   0.00 + 1.00i
>> x = 1/0
x =
    Inf
>> 0/0
ans =
    NaN
fx >>
```

Matrix and array operations



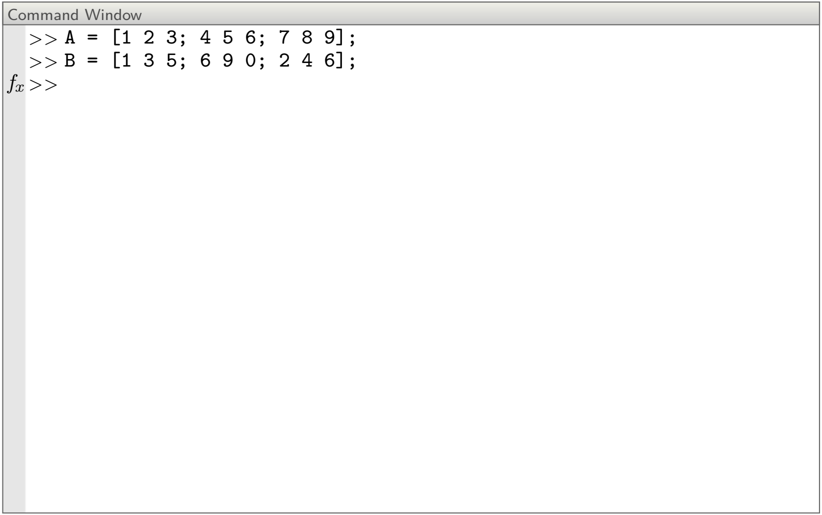
```
>>
```

Matrix and array operations



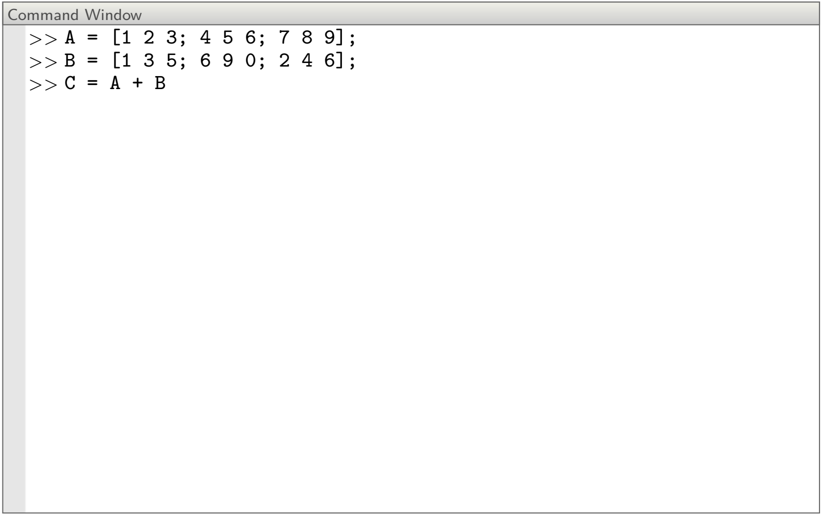
```
>> A = [1 2 3; 4 5 6; 7 8 9];
fx >>
```

Matrix and array operations



```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
fx >>
```

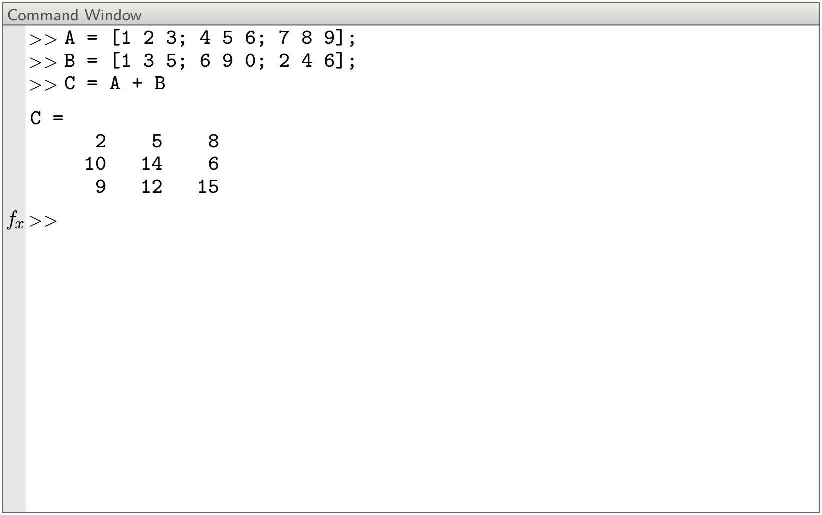
Matrix and array operations



Command Window

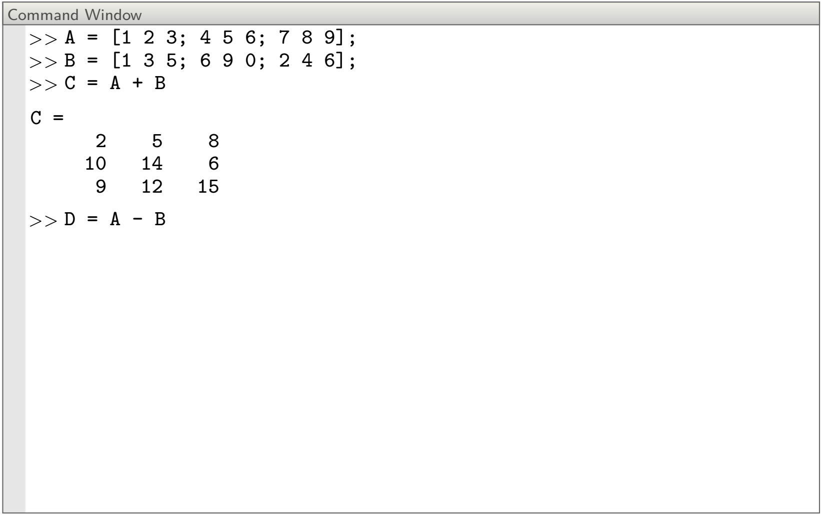
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> C = A + B
```

Matrix and array operations



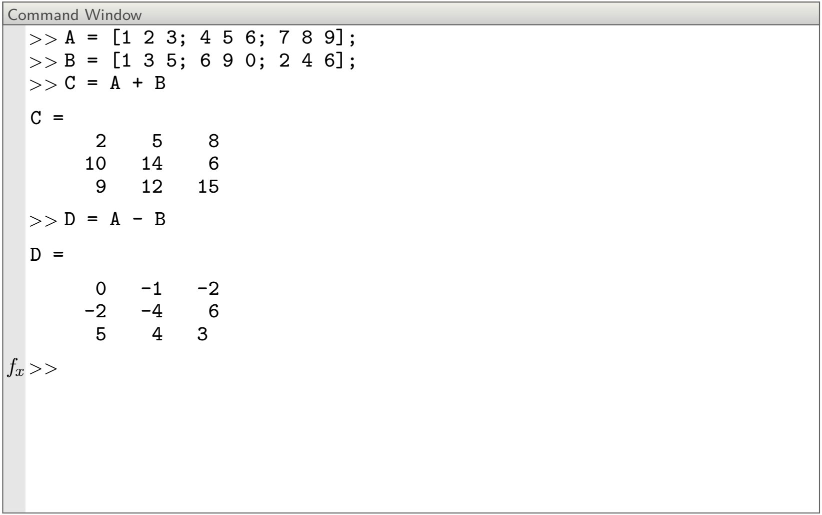
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> C = A + B

C =
      2    5    8
     10   14    6
      9   12   15

fx >>
```

Matrix and array operations



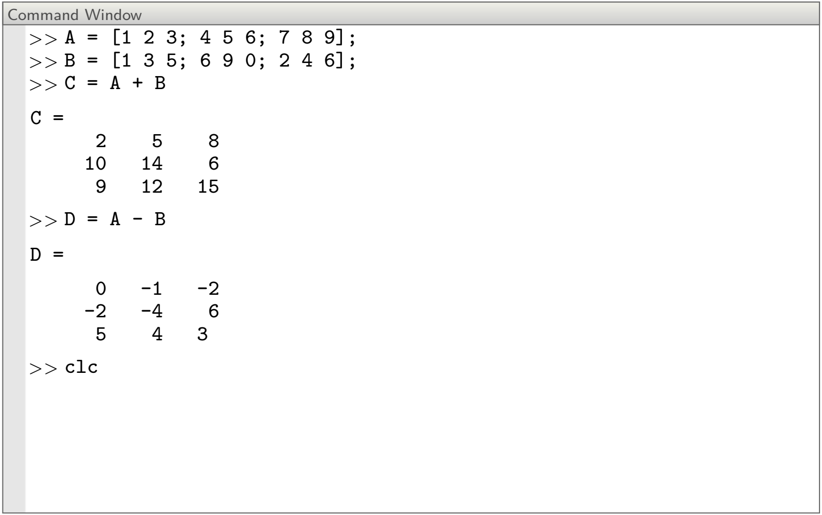
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> C = A + B

C =
      2     5     8
     10    14     6
      9    12    15

>> D = A - B
```

Matrix and array operations



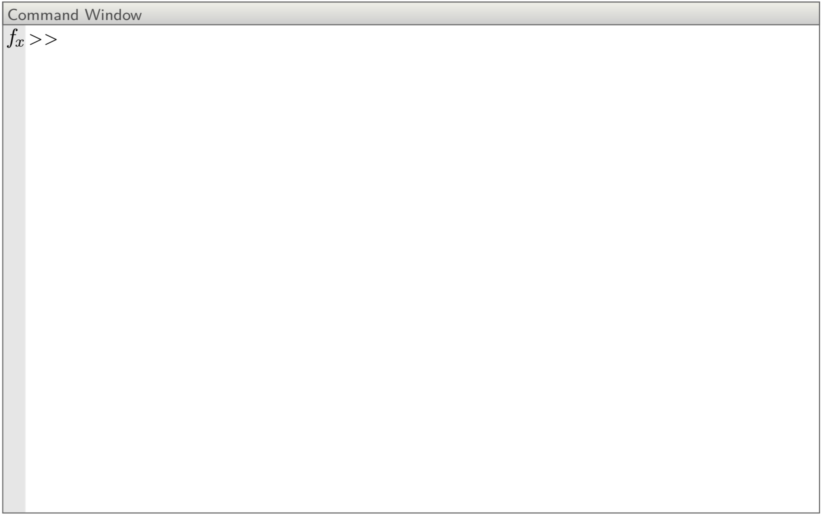
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> C = A + B

C =
      2    5    8
     10   14    6
      9   12   15

>> D = A - B

D =

      0   -1   -2
     -2   -4    6
      5    4   3

fx >>
```

Matrix and array operations



```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> C = A + B

C =
      2    5    8
     10   14    6
      9   12   15

>> D = A - B

D =

      0   -1   -2
     -2   -4    6
      5    4   3

>> clc
```

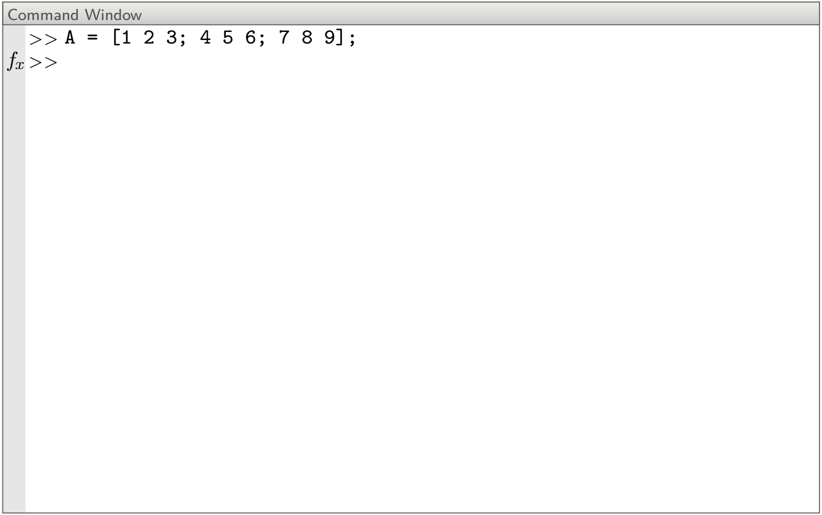
Matrix and array operations



Command Window

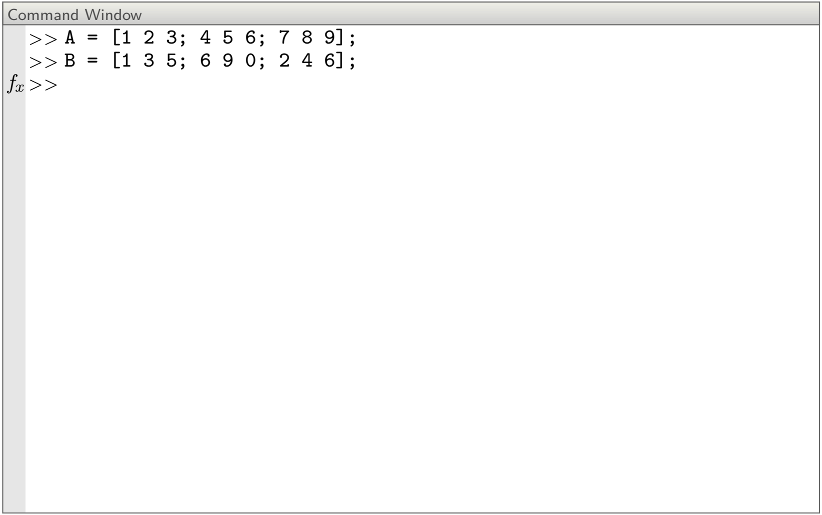
```
fx >>
```

Matrix and array operations



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

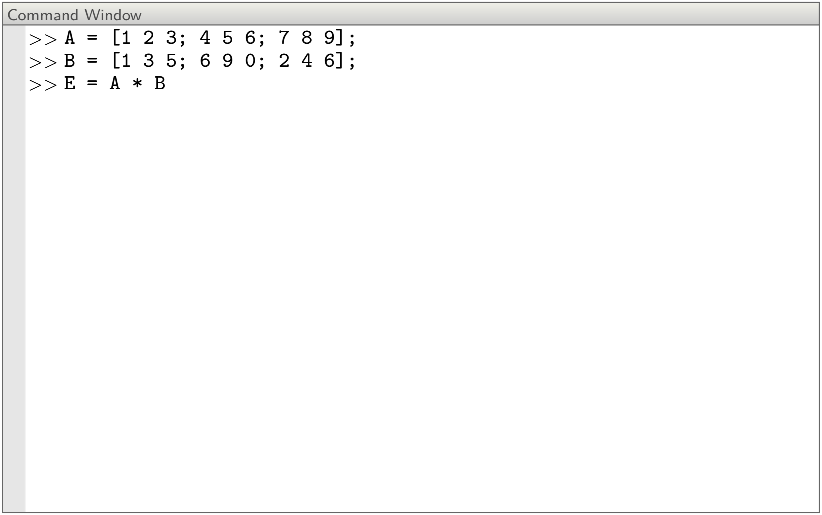
Matrix and array operations



Command Window

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
fx >>
```

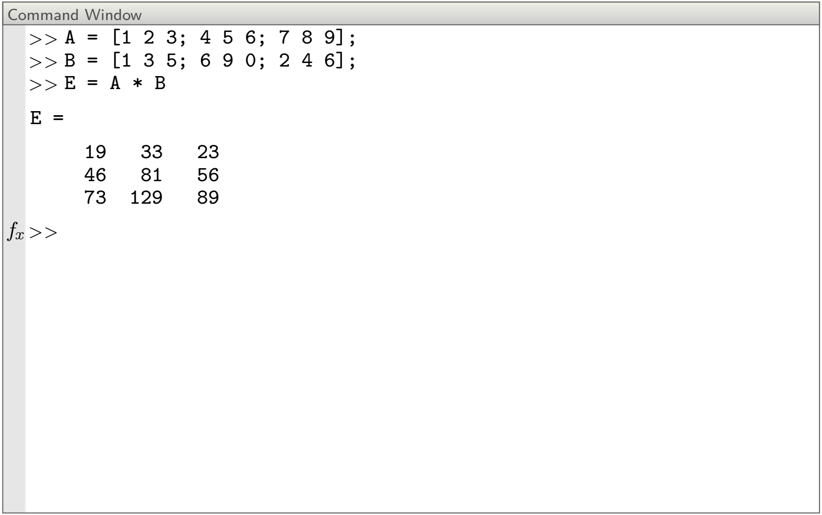
Matrix and array operations



Command Window

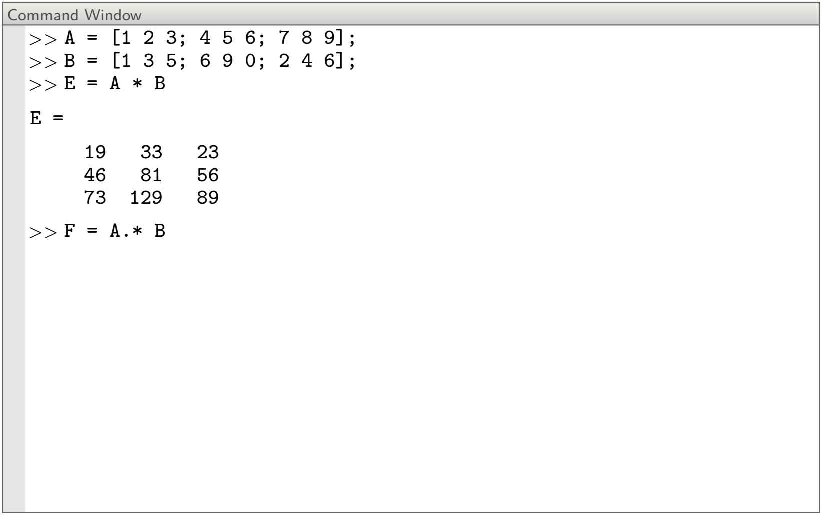
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> E = A * B
```

Matrix and array operations



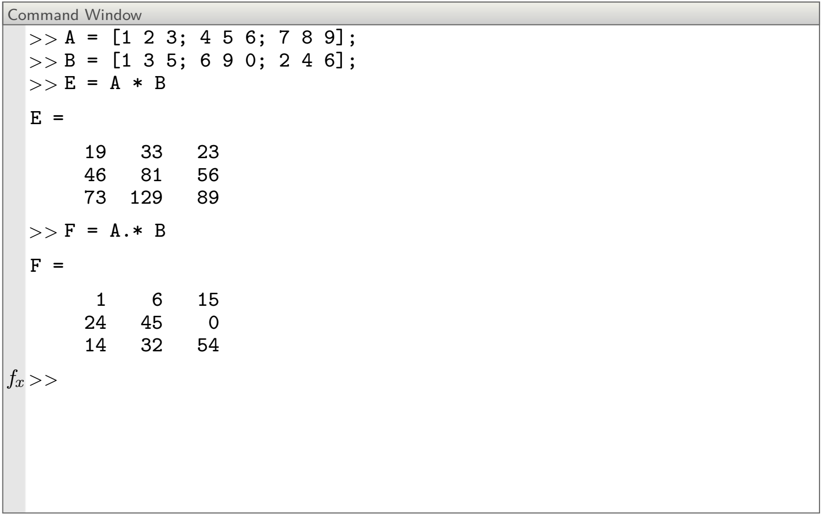
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> E = A * B

E =

    19    33    23
    46    81    56
    73   129    89

fx >>
```

Matrix and array operations



```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> E = A * B

E =

    19    33    23
    46    81    56
    73   129    89

>> F = A.* B
```

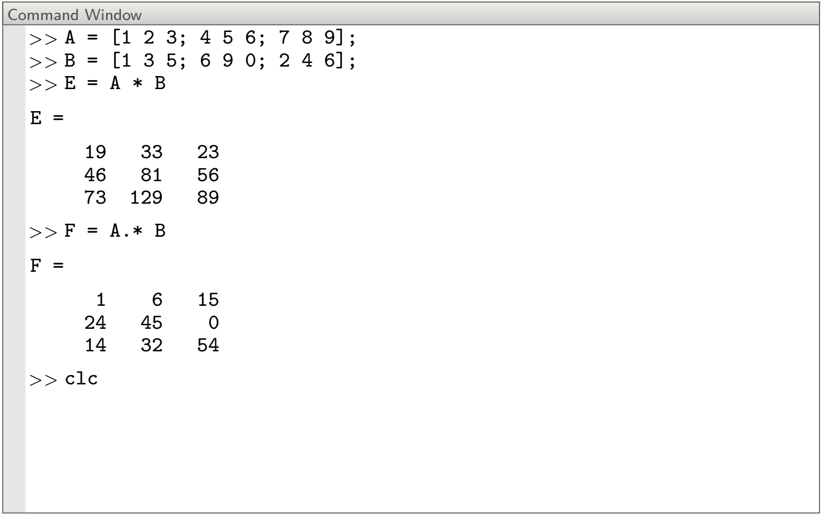
Matrix and array operations



Command Window

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> E = A * B

E =

    19    33    23
    46    81    56
    73   129    89

>> F = A.* B

F =

     1     6    15
    24    45     0
    14    32    54

fx >>
```

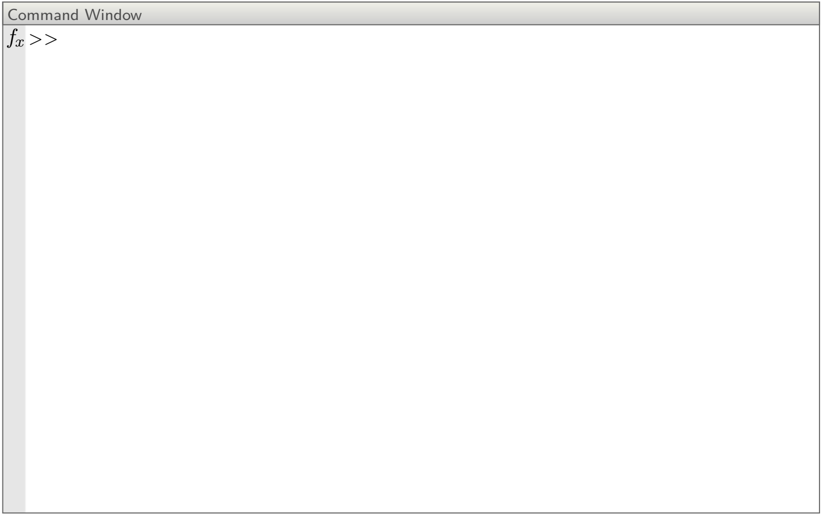
Matrix and array operations



Command Window

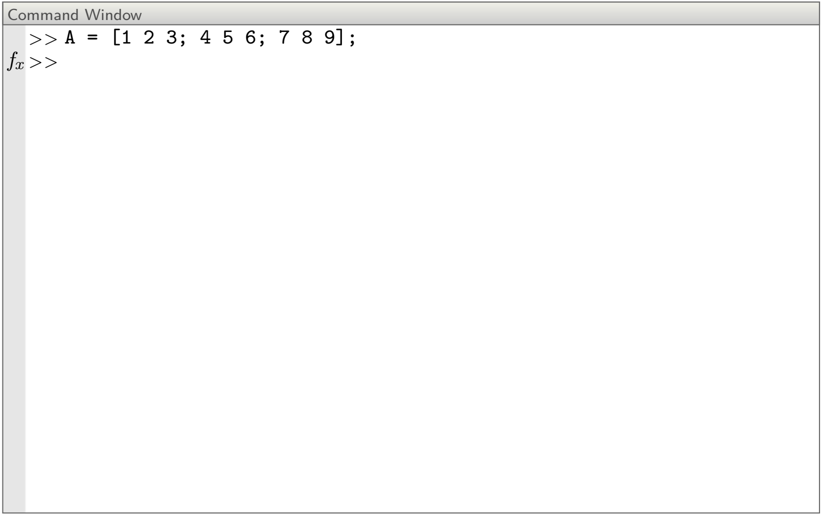
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> E = A * B

E =

    19    33    23
    46    81    56
    73   129    89

>> F = A.* B

F =

     1     6    15
    24    45     0
    14    32    54

>> clc
```

Matrix and array operations



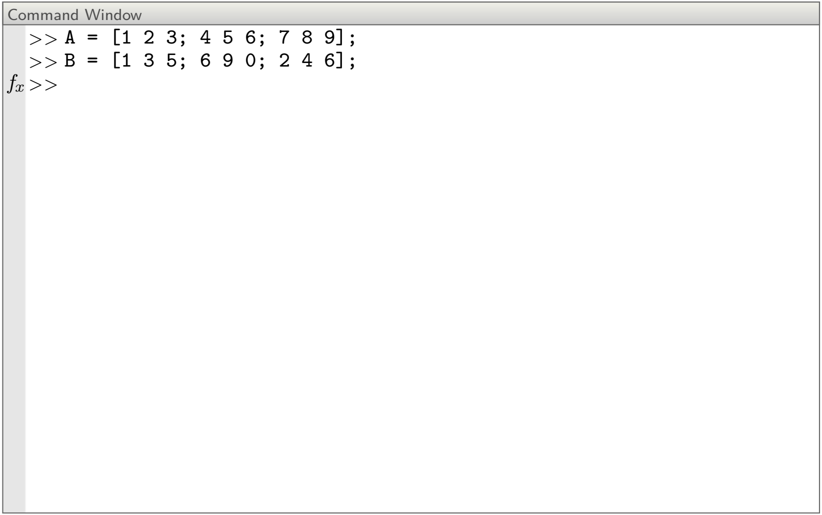
```
fx >>
```

Matrix and array operations



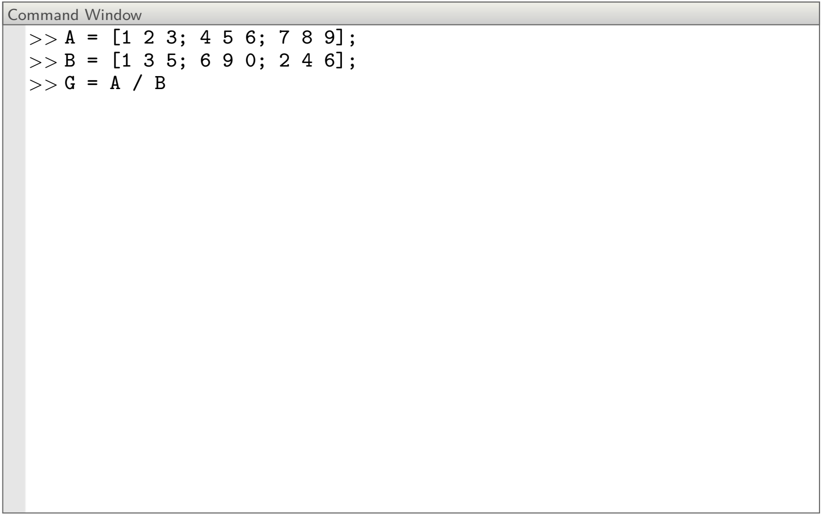
```
>> A = [1 2 3; 4 5 6; 7 8 9];
fx >>
```

Matrix and array operations



```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
fx >>
```

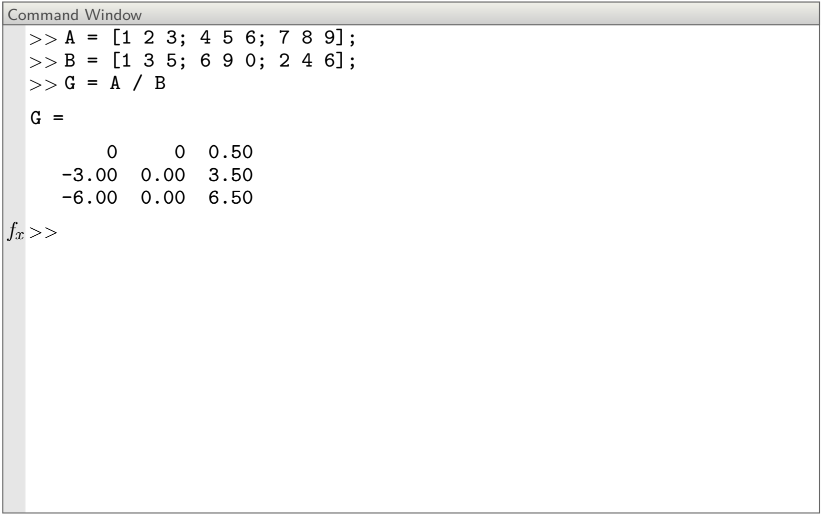
Matrix and array operations



Command Window

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> G = A / B
```

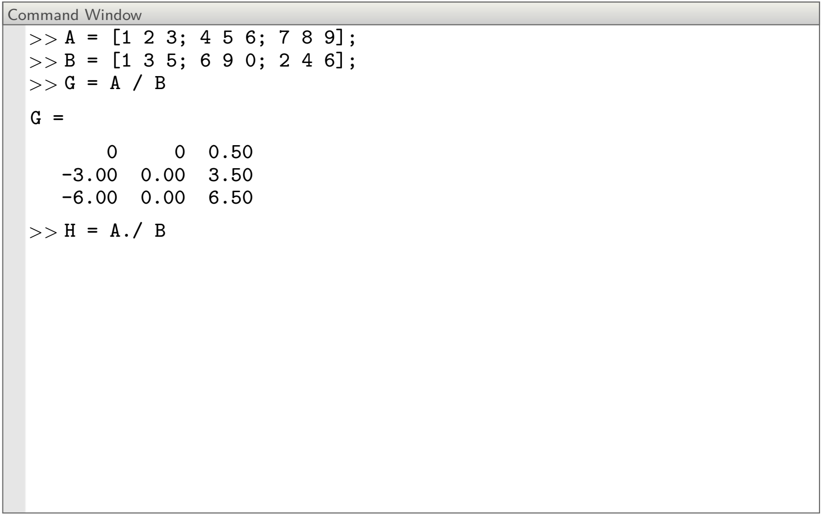
Matrix and array operations



Command Window

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> G = A / B

G =

        0      0   0.50
    -3.00   0.00   3.50
    -6.00   0.00   6.50

fx >>
```

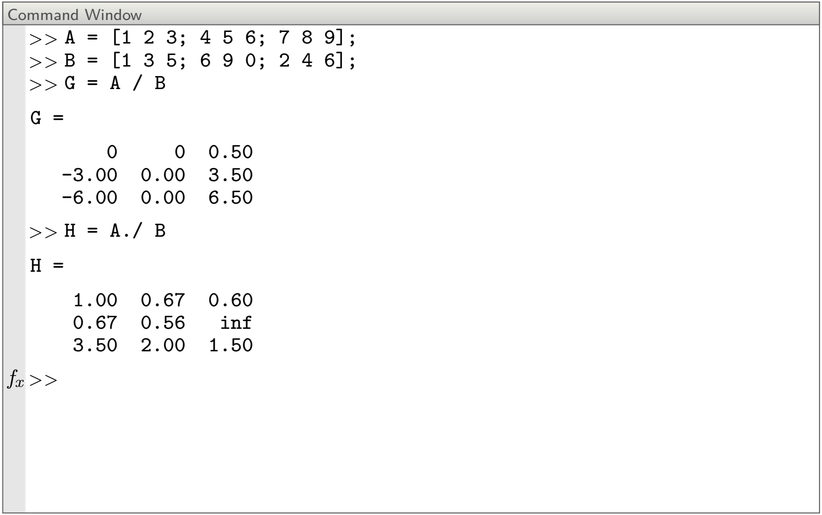
Matrix and array operations



Command Window

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> G = A / B

G =

        0      0   0.50
   -3.00   0.00   3.50
   -6.00   0.00   6.50

>> H = A./ B
```

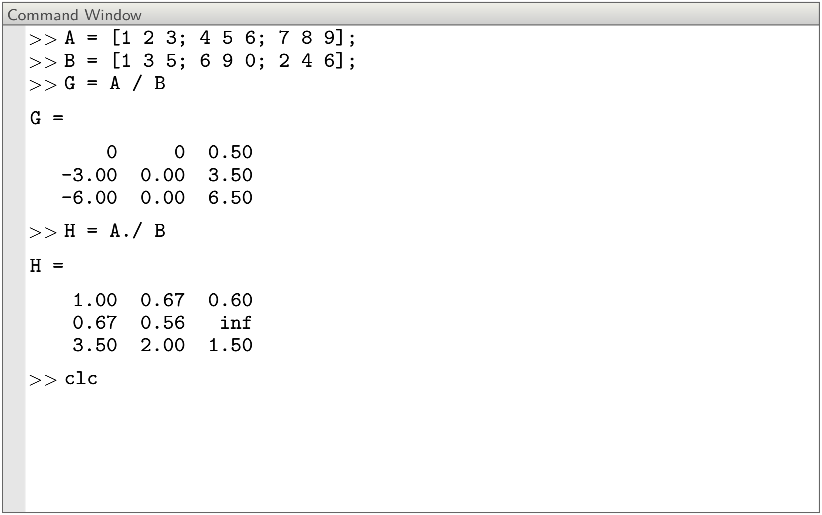
Matrix and array operations



Command Window

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> G = A / B

G =

      0      0   0.50
  -3.00   0.00   3.50
  -6.00   0.00   6.50

>> H = A./ B

H =

   1.00   0.67   0.60
   0.67   0.56    inf
   3.50   2.00   1.50

fx >>
```

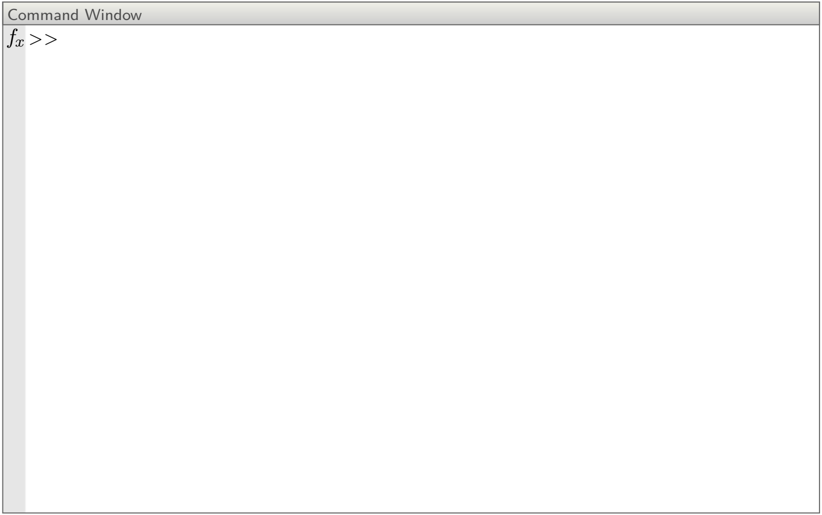
Matrix and array operations



Command Window

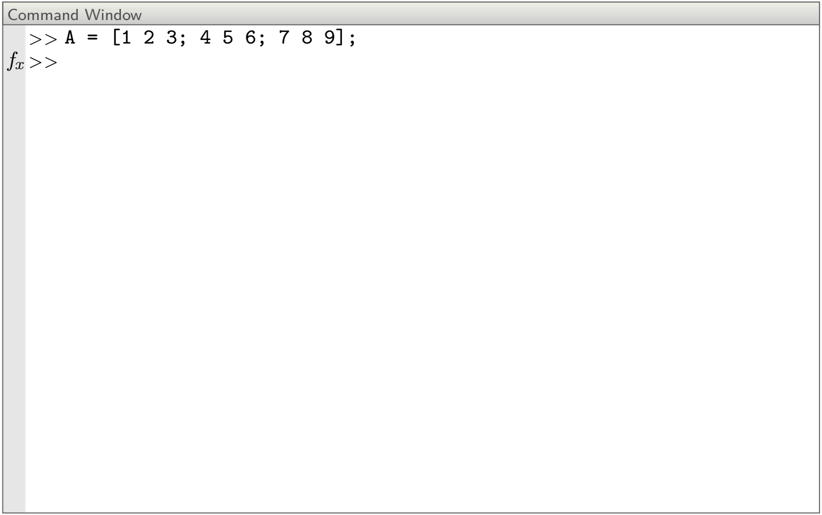
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> G = A / B

G =

      0      0  0.50
  -3.00  0.00  3.50
  -6.00  0.00  6.50

>> H = A./ B

H =

   1.00  0.67  0.60
   0.67  0.56   inf
   3.50  2.00  1.50

>> clc
```

Matrix and array operations



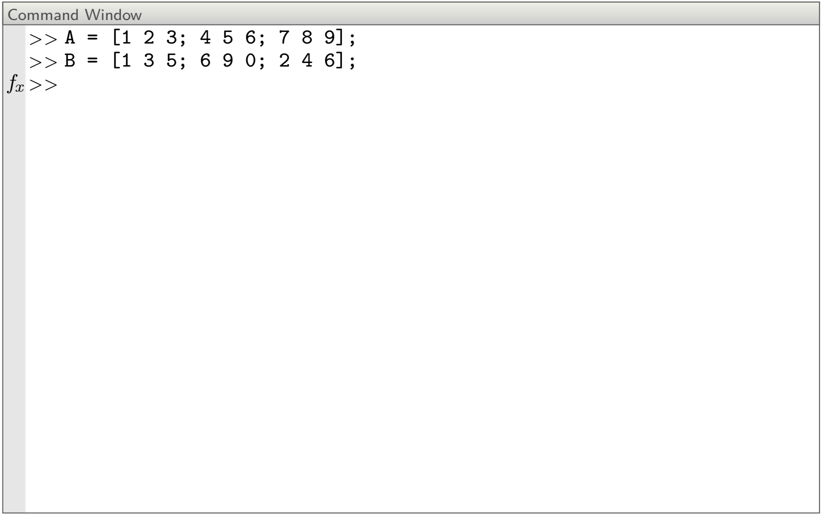
```
fx >>
```

Matrix and array operations



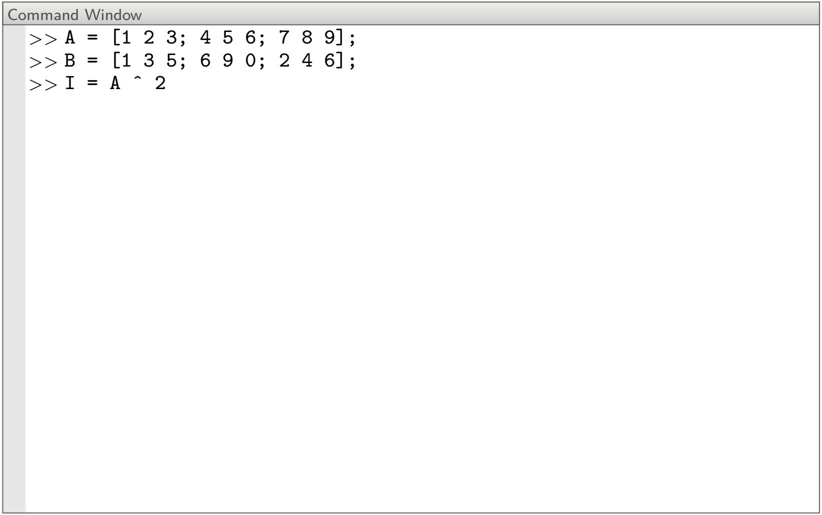
```
>> A = [1 2 3; 4 5 6; 7 8 9];
fx >>
```

Matrix and array operations



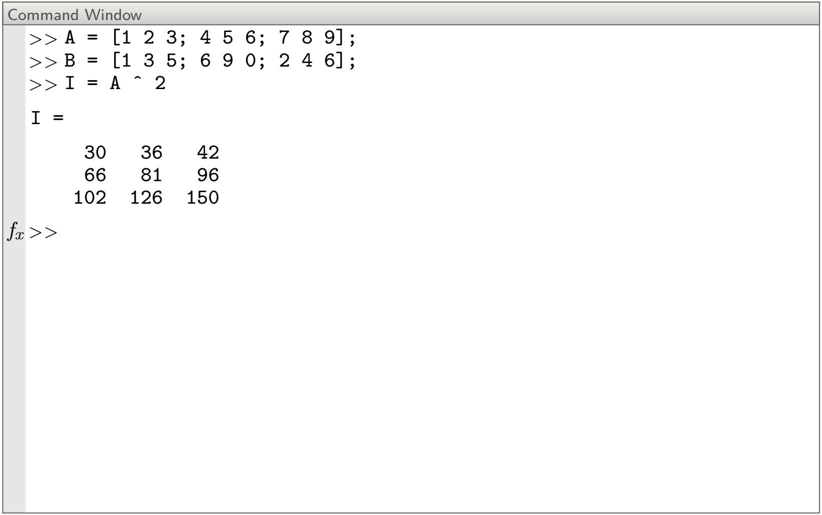
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
fx >>
```

Matrix and array operations



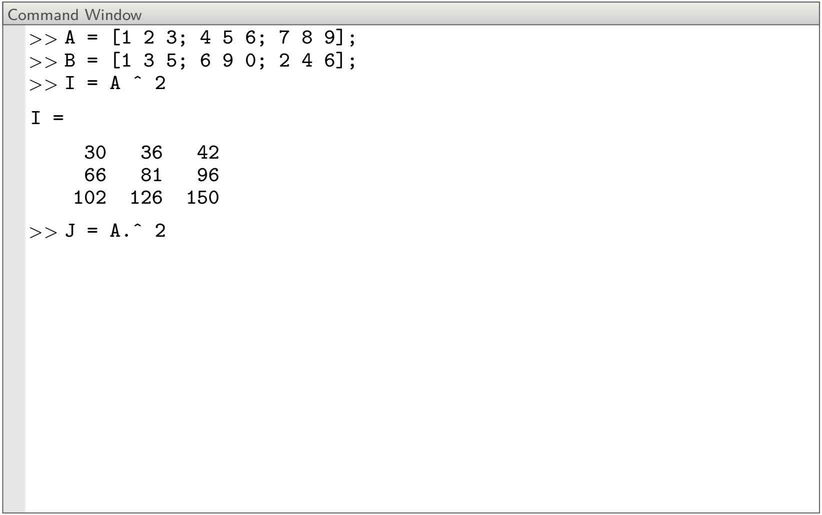
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> I = A ˆ 2
```

Matrix and array operations



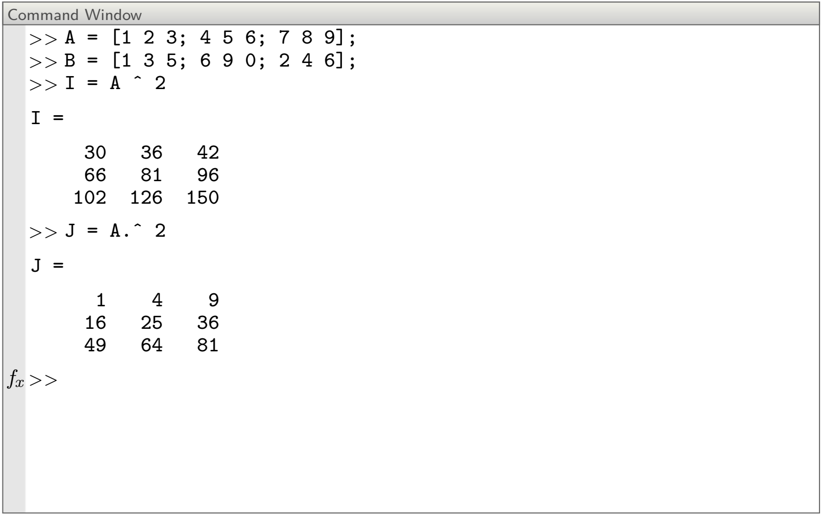
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> I = A ˆ 2

I =

     30    36    42
     66    81    96
    102   126   150

fx >>
```

Matrix and array operations



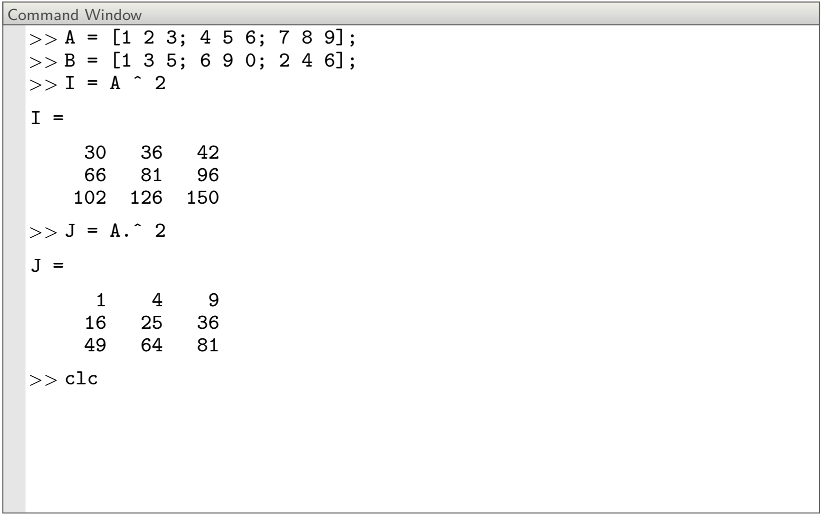
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> I = A ^ 2

I =

     30    36    42
     66    81    96
    102   126   150

>> J = A.^ 2
```

Matrix and array operations



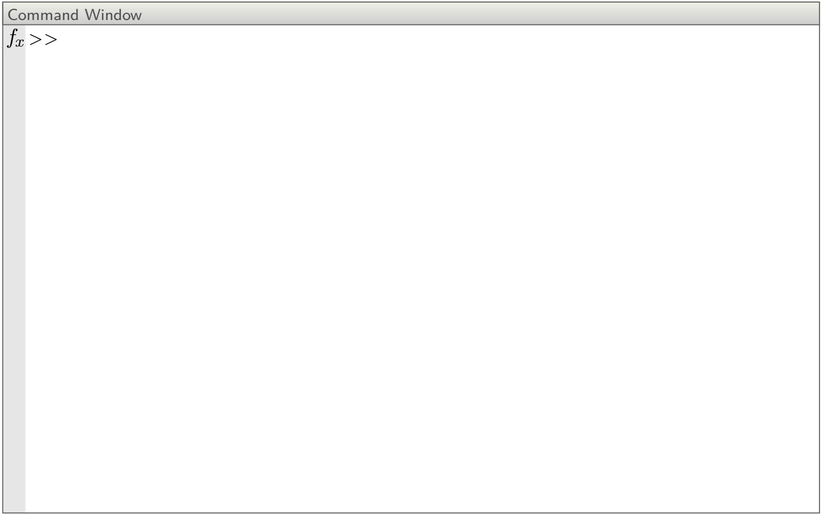
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> I = A ˆ 2

I =

     30    36    42
     66    81    96
    102   126   150

>> J = A.ˆ 2

J =

      1     4     9
     16    25    36
     49    64    81

fx >>
```

Matrix and array operations



```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = [1 3 5; 6 9 0; 2 4 6];
>> I = A ^ 2

I =

     30    36    42
     66    81    96
    102   126   150

>> J = A.^ 2

J =

      1     4     9
     16    25    36
     49    64    81

>> clc
```

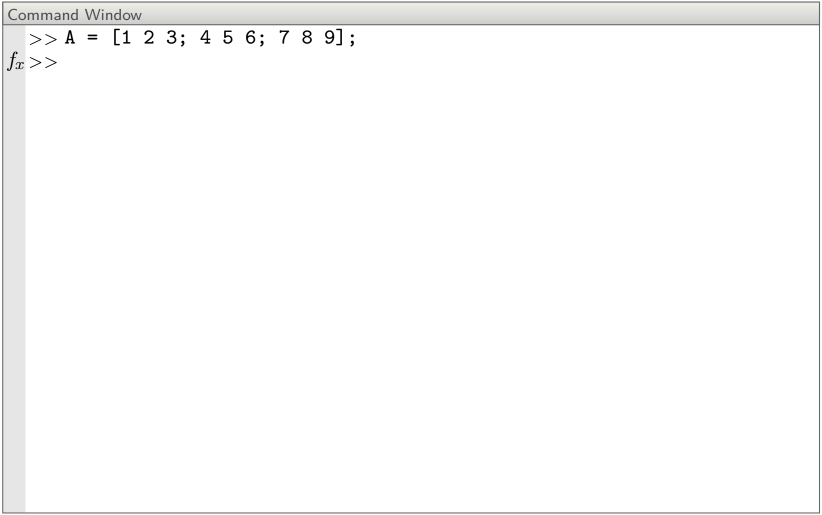
Array and array row and column block operation: value



Command Window

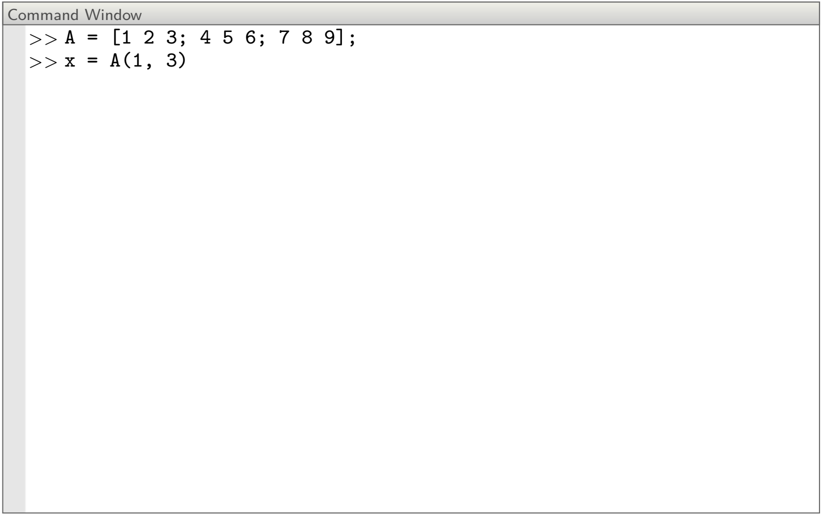
```
fx >>
```

Array and array row and column block operation: value



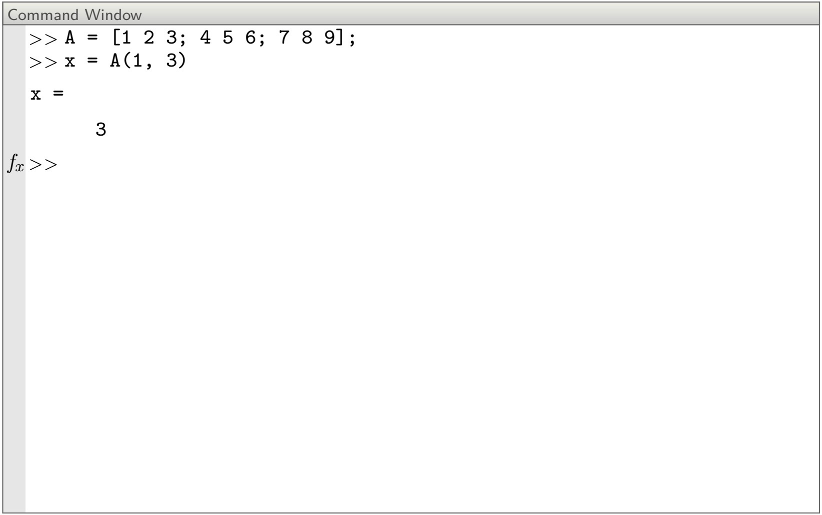
```
>> A = [1 2 3; 4 5 6; 7 8 9];
fx >>
```

Array and array row and column block operation: value



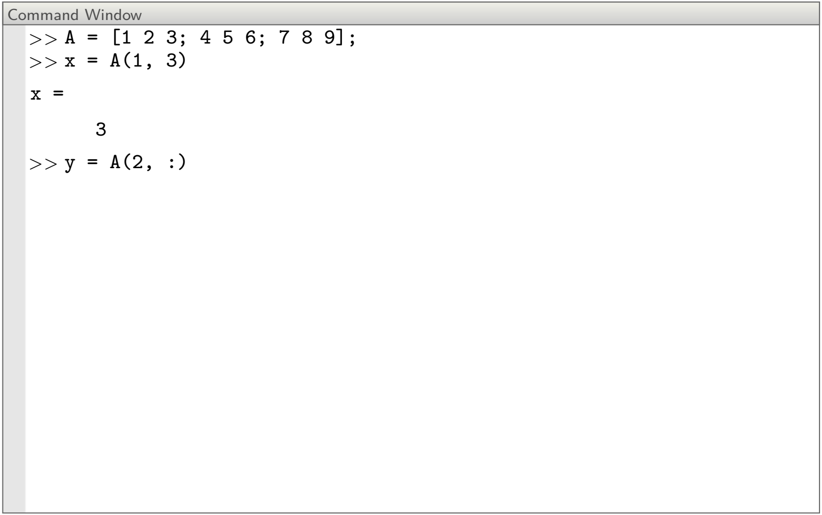
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> x = A(1, 3)
```

Array and array row and column block operation: value



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

x =

     3

fx >>
```

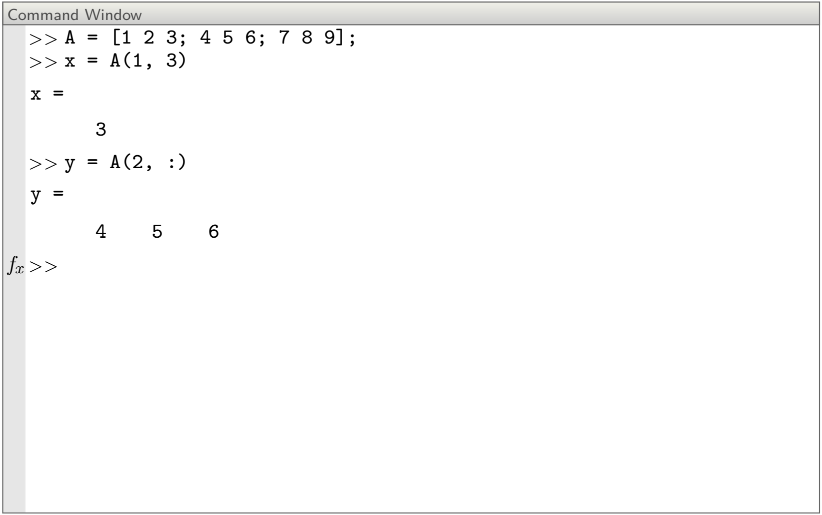
Array and array row and column block operation: value



Command Window

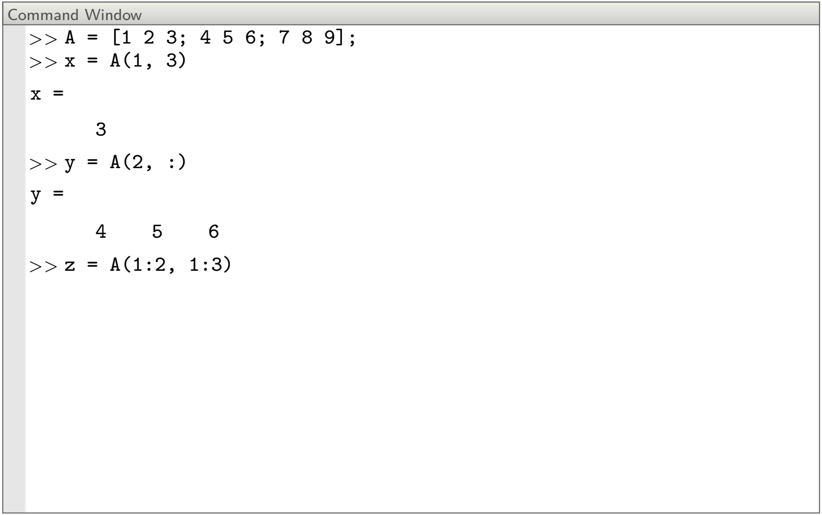
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> x = A(1, 3)

x =

     3

>> y = A(2, :)
```

Array and array row and column block operation: value



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

x =

     3

>> y = A(2, :)

y =

     4     5     6

fx >>
```

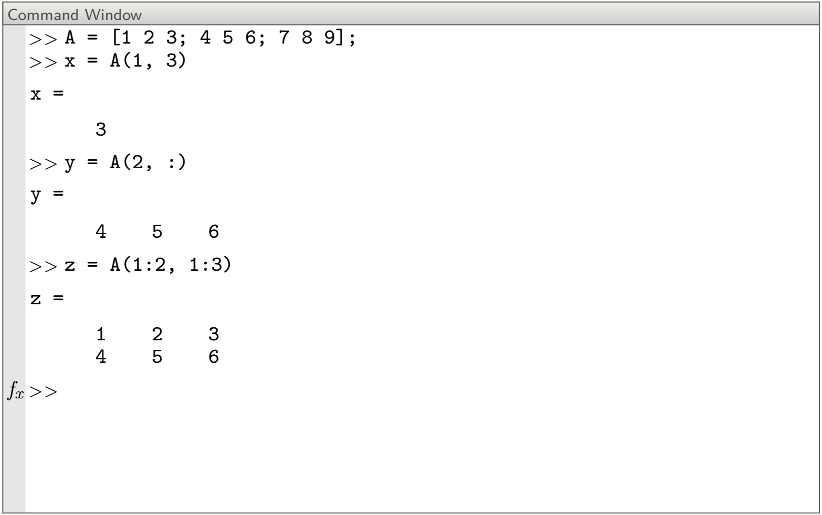
Array and array row and column block operation: value



Command Window

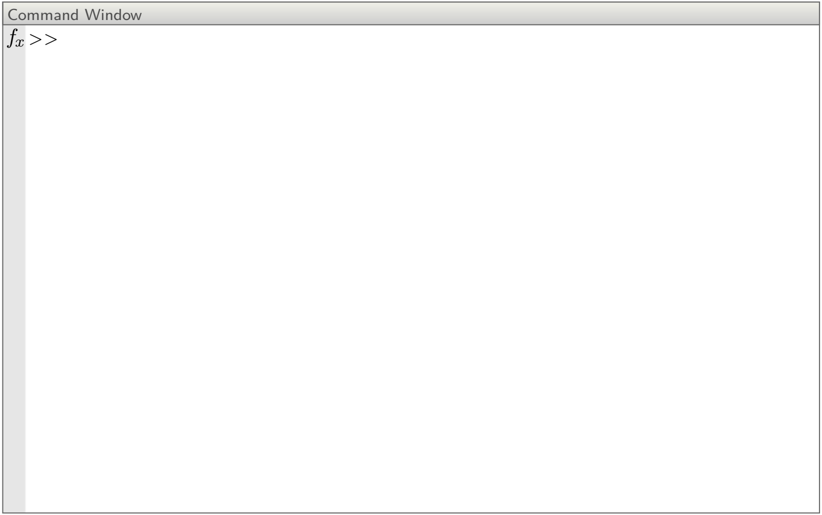
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> x = A(1, 3)

x =

     3

>> y = A(2, :)

y =

     4     5     6

>> z = A(1:2, 1:3)
```

Array and array row and column block operation: value



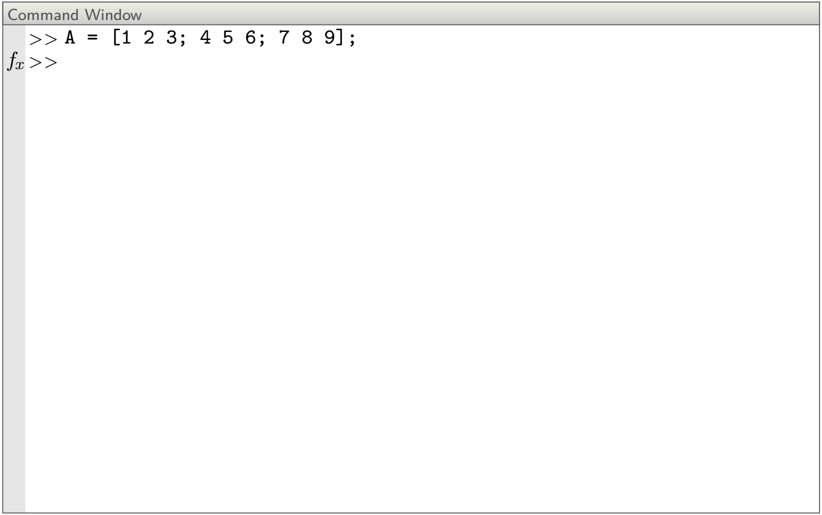
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> x = A(1, 3)

x =

     3

>> y = A(2, :)

y =

     4     5     6

>> z = A(1:2, 1:3)

z =

     1     2     3
     4     5     6

fx >>
```

Array and array row and column block operation: value



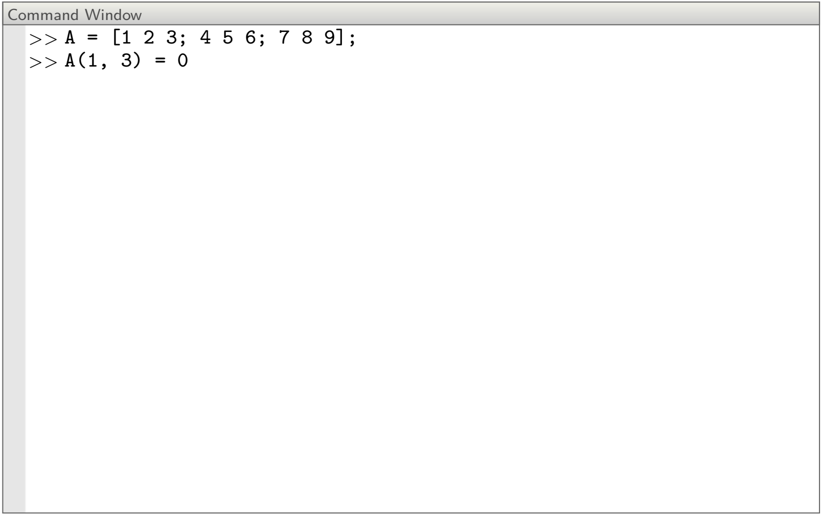
```
>>
```

Array and array row and column block operation: value



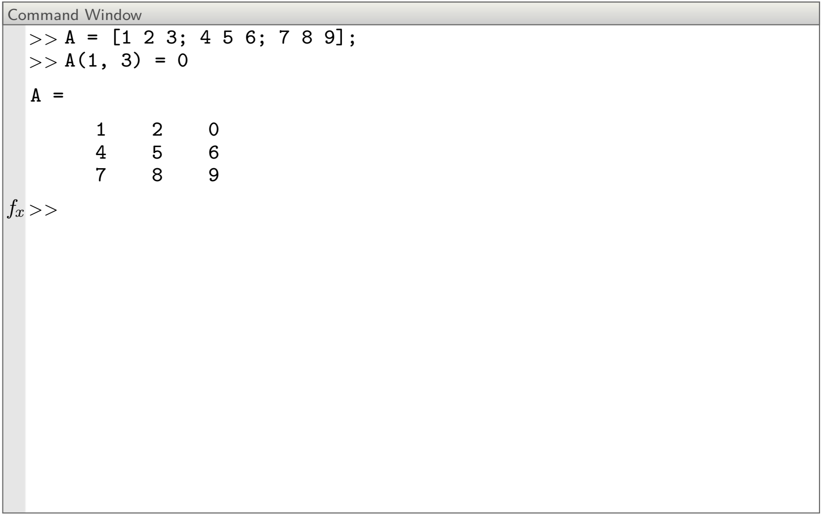
```
>> A = [1 2 3; 4 5 6; 7 8 9];
fx >>
```

Array and array row and column block operation: value



```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(1, 3) = 0
```

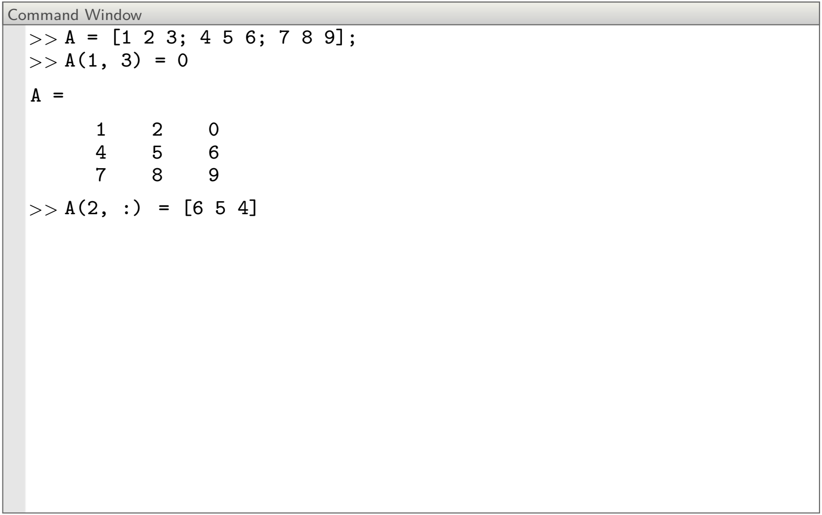
Array and array row and column block operation: value



Command Window

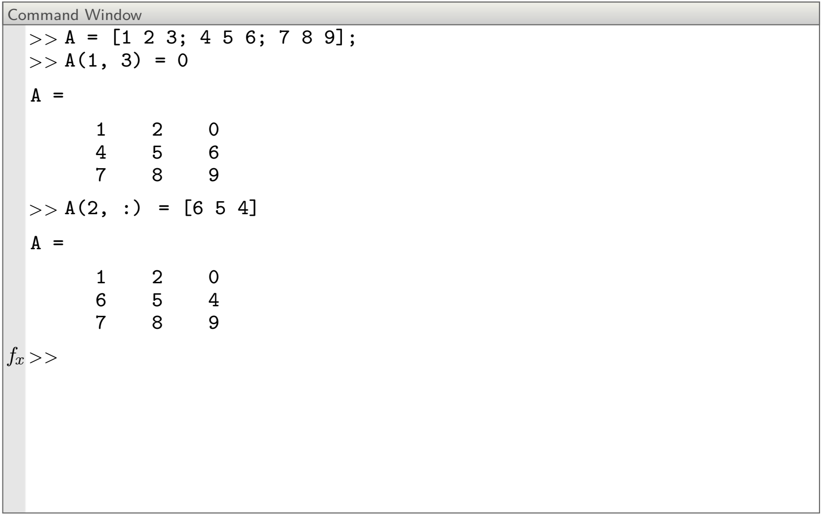
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(1, 3) = 0

A =

     1     2     0
     4     5     6
     7     8     9

fx >>
```

Array and array row and column block operation: value



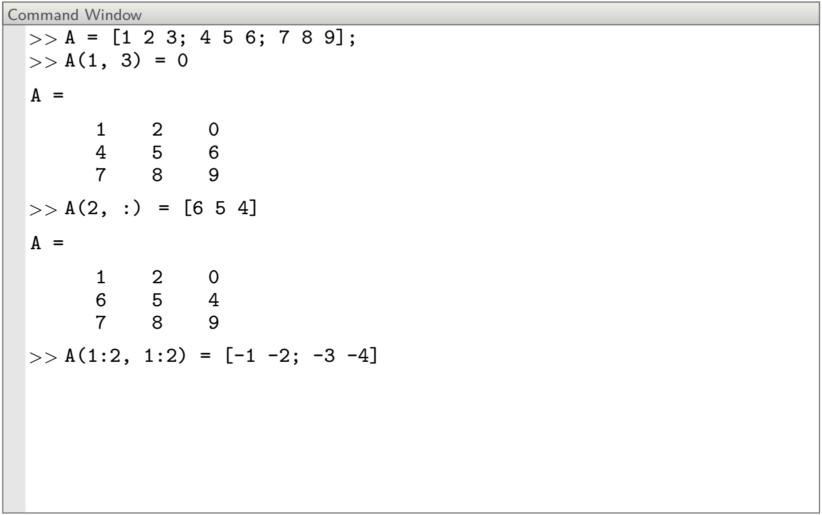
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(1, 3) = 0

A =

     1     2     0
     4     5     6
     7     8     9

>> A(2, :)  = [6 5 4]
```

Array and array row and column block operation: value



```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(1, 3) = 0

A =

     1     2     0
     4     5     6
     7     8     9

>> A(2, :)  = [6 5 4]

A =

     1     2     0
     6     5     4
     7     8     9

fx >>
```

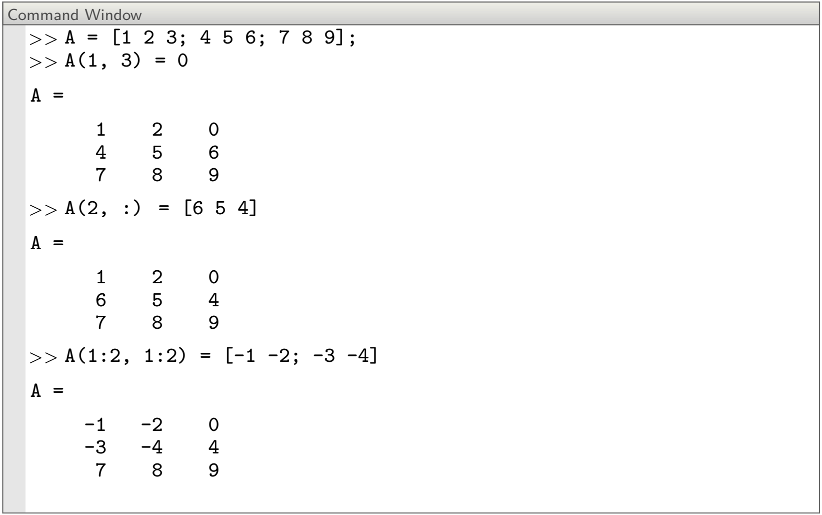
Array and array row and column block operation: value



Command Window

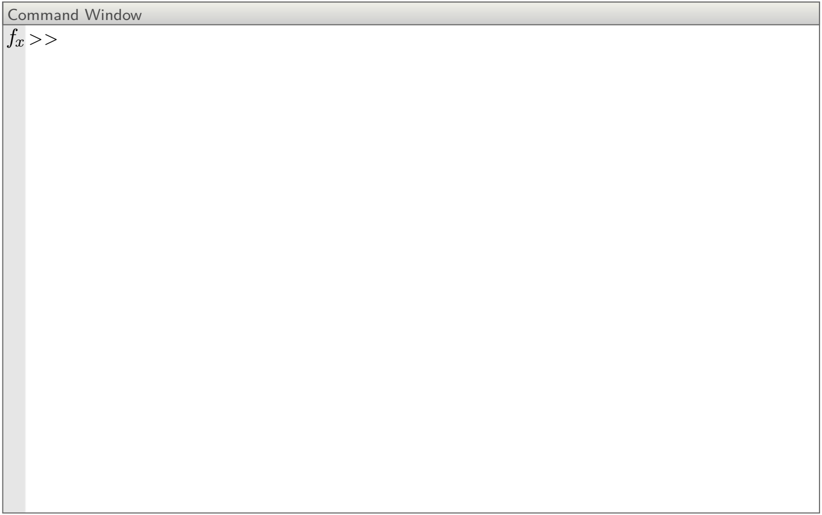
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(1, 3) = 0

A =

     1     2     0
     4     5     6
     7     8     9

>> A(2, :)  = [6 5 4]

A =

     1     2     0
     6     5     4
     7     8     9

>> A(1:2, 1:2) = [-1 -2; -3 -4]
```

Array and array row and column block operation: value



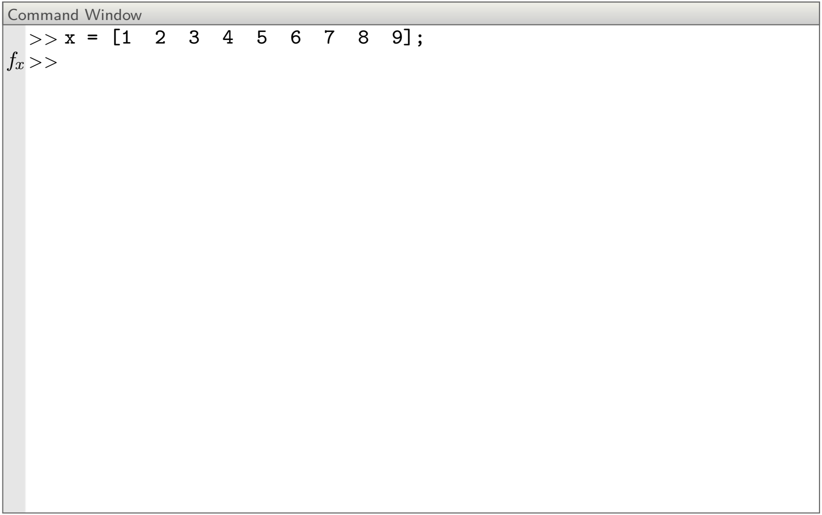
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> A(1, 3) = 0

A =

     1     2     0
     4     5     6
     7     8     9

>> A(2, :)  = [6 5 4]

A =

     1     2     0
     6     5     4
     7     8     9

>> A(1:2, 1:2) = [-1 -2; -3 -4]

A =

    -1    -2     0
    -3    -4     4
     7     8     9
```

Comparison and logical operations



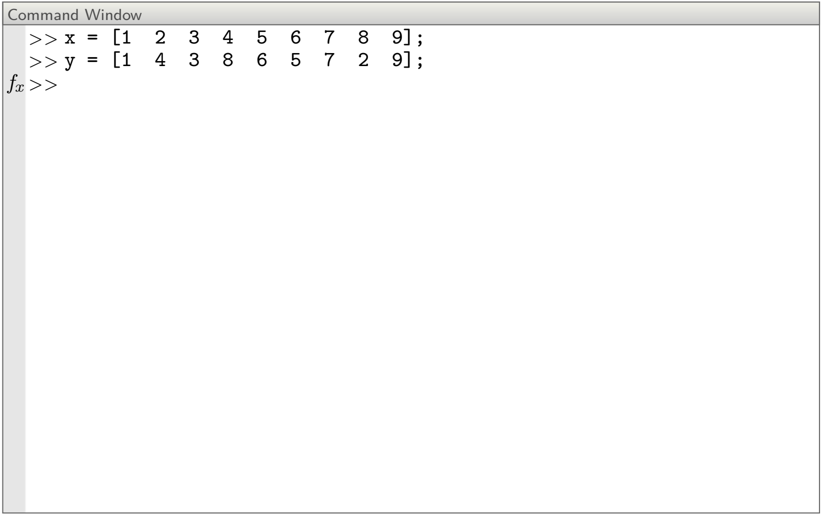
$f_x >>$

Comparison and logical operations



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

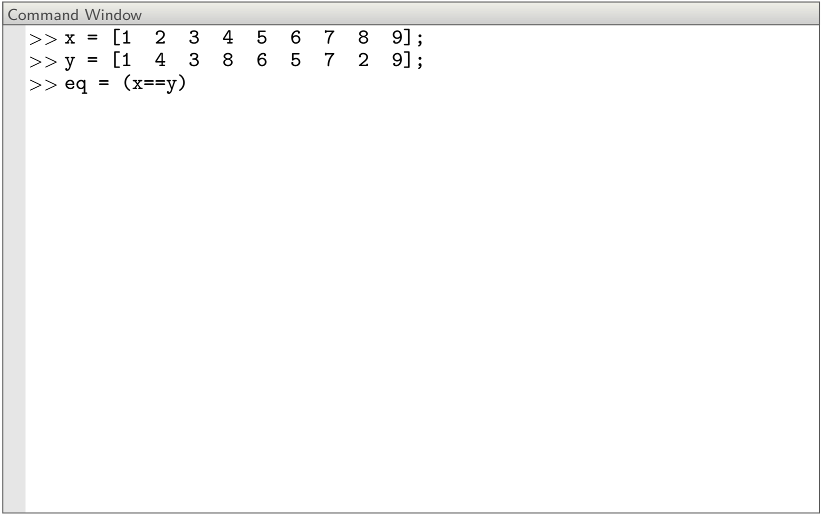
Comparison and logical operations



Command Window

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

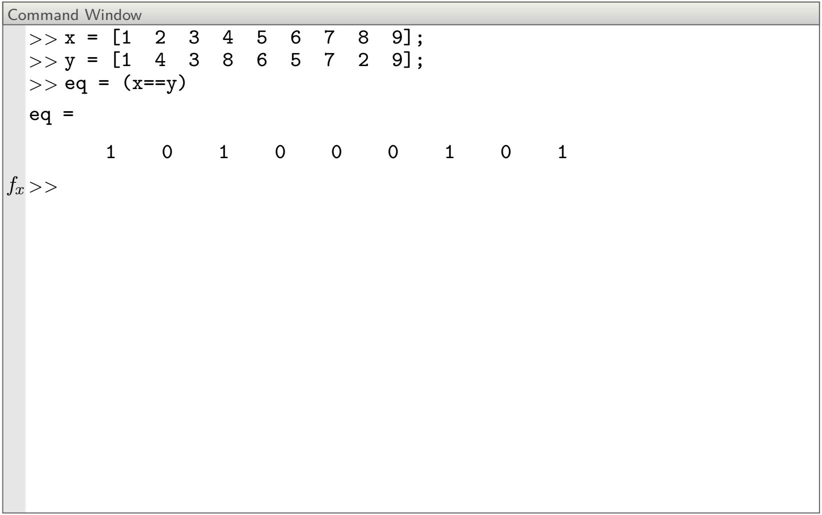
Comparison and logical operations



Command Window

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

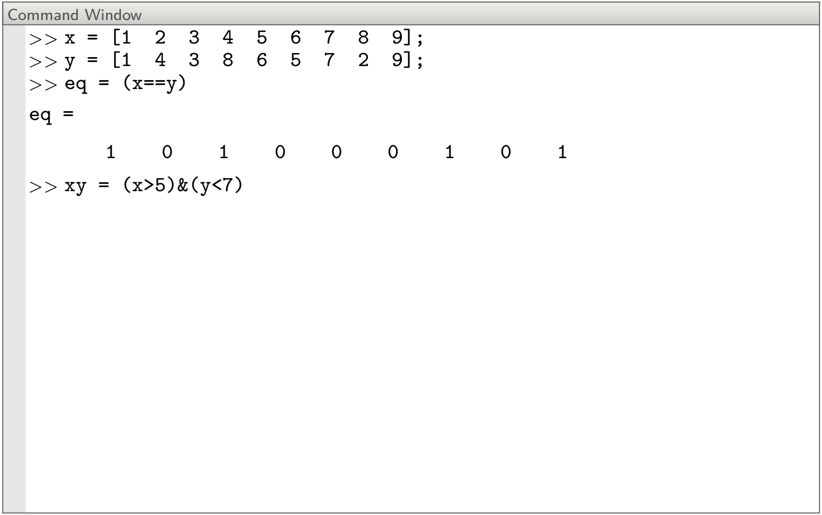
Comparison and logical operations



Command Window

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

eq =

     1     0     1     0     0     0     1     0     1

fx >>
```

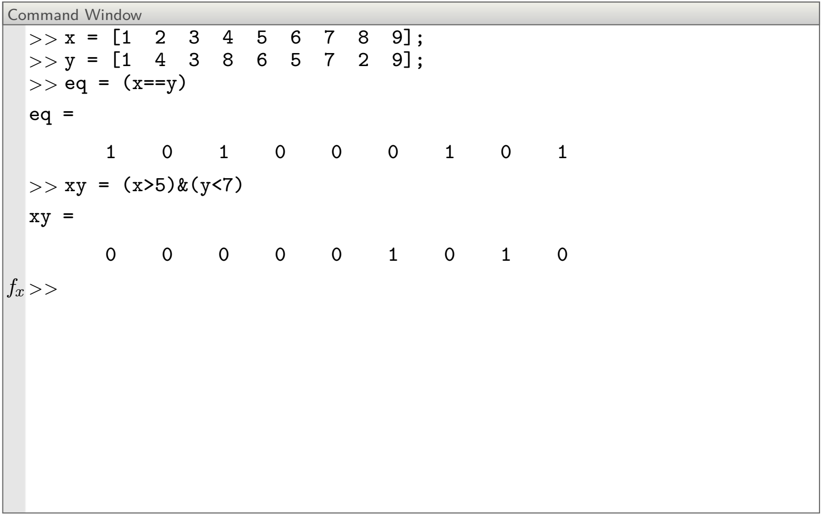
Comparison and logical operations



Command Window

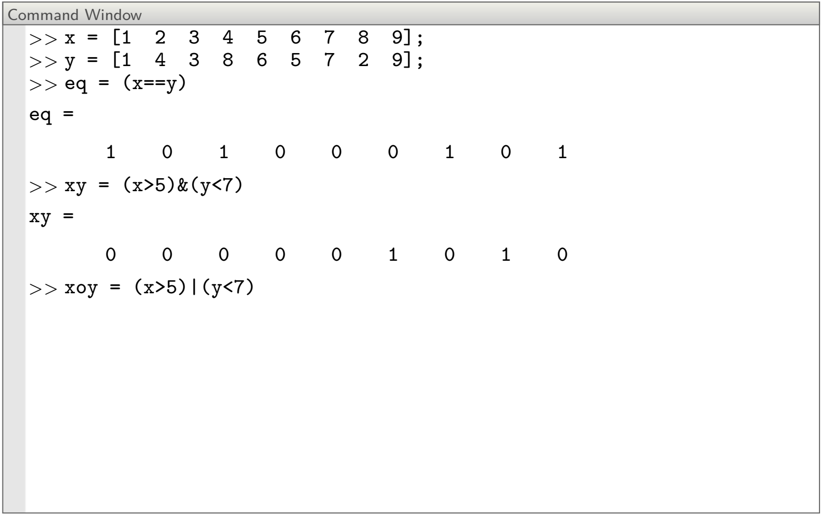
```
>> x = [1  2  3  4  5  6  7  8  9];
>> y = [1  4  3  8  6  5  7  2  9];
>> eq = (x==y)

eq =

     1     0     1     0     0     0     1     0     1

>> xy = (x>5)&(y<7)
```

Comparison and logical operations



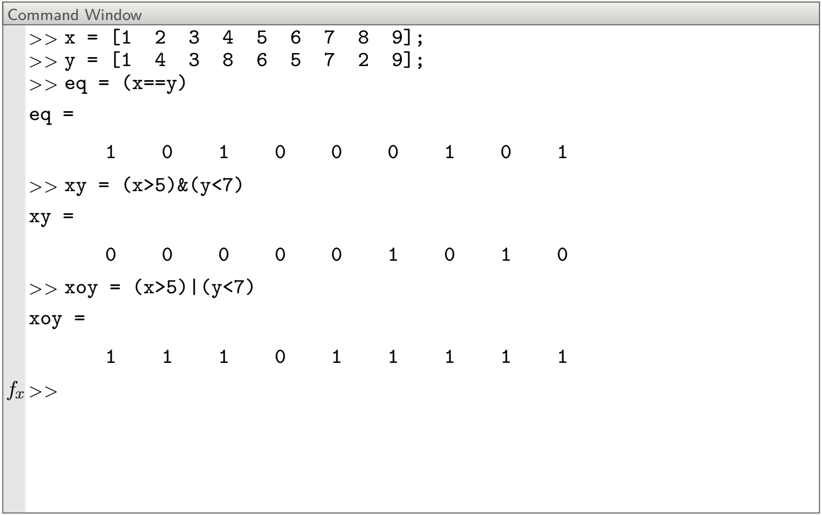
```
>> x = [1  2  3  4  5  6  7  8  9];
>> y = [1  4  3  8  6  5  7  2  9];
>> eq = (x==y)
eq =

      1     0     1     0     0     0     1     0     1

>> xy = (x>5)&(y<7)
xy =

      0     0     0     0     0     1     0     1     0

fx >>
```

Comparison and logical operations



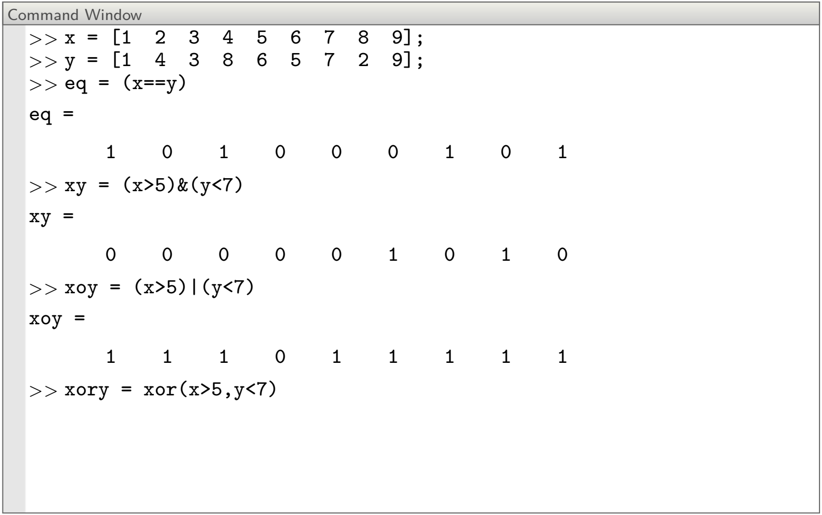
```
>> x = [1  2  3  4  5  6  7  8  9];
>> y = [1  4  3  8  6  5  7  2  9];
>> eq = (x==y)

eq =

     1     0     1     0     0     0     1     0     1

>> xy = (x>5)&(y<7)

xy =

     0     0     0     0     0     1     0     1     0

>> xoy = (x>5)|(y<7)
```

Comparison and logical operations



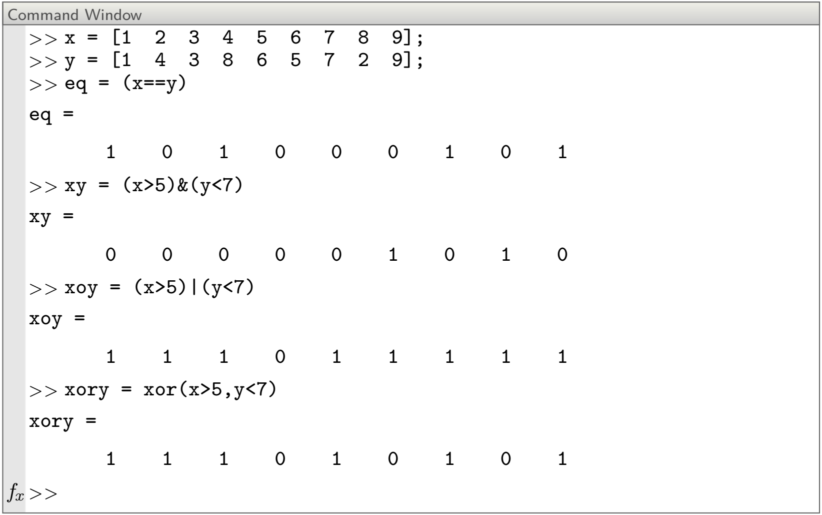
```
>> x = [1  2  3  4  5  6  7  8  9];
>> y = [1  4  3  8  6  5  7  2  9];
>> eq = (x==y)

eq =

      1     0     1     0     0     0     1     0     1

>> xy = (x>5)&(y<7)

xy =

      0     0     0     0     0     1     0     1     0

>> xoy = (x>5)|(y<7)

xoy =

      1     1     1     0     1     1     1     1     1

fx >>
```

Comparison and logical operations



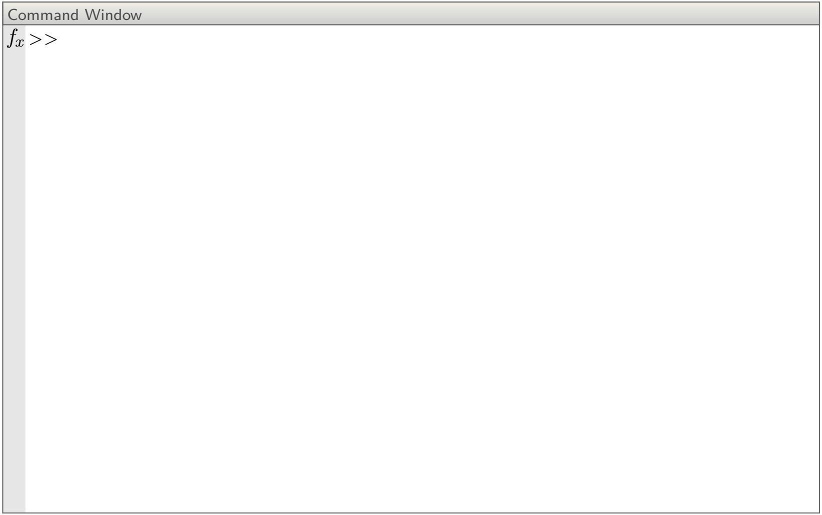
```
>> x = [1  2  3  4  5  6  7  8  9];
>> y = [1  4  3  8  6  5  7  2  9];
>> eq = (x==y)

eq =

     1     0     1     0     0     0     1     0     1

>> xy = (x>5)&(y<7)

xy =

     0     0     0     0     0     1     0     1     0

>> xoy = (x>5)|(y<7)

xoy =

     1     1     1     0     1     1     1     1     1

>> xory = xor(x>5,y<7)
```

Comparison and logical operations



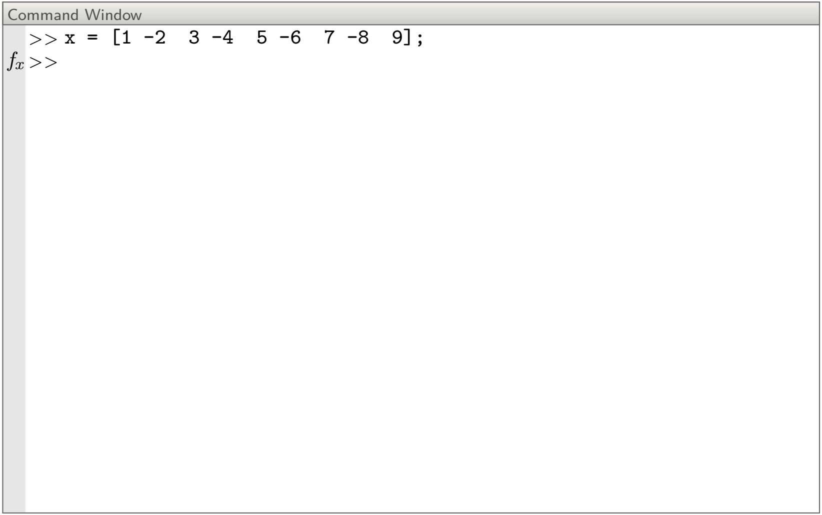
```
>> x = [1  2  3  4  5  6  7  8  9];
>> y = [1  4  3  8  6  5  7  2  9];
>> eq = (x==y)

eq =

     1     0     1     0     0     0     1     0     1

>> xy = (x>5)&(y<7)

xy =

     0     0     0     0     0     1     0     1     0

>> xoy = (x>5)|(y<7)

xoy =

     1     1     1     0     1     1     1     1     1

>> xory = xor(x>5,y<7)

xory =

     1     1     1     0     1     0     1     0     1

fx >>
```

Comparison and logical operations



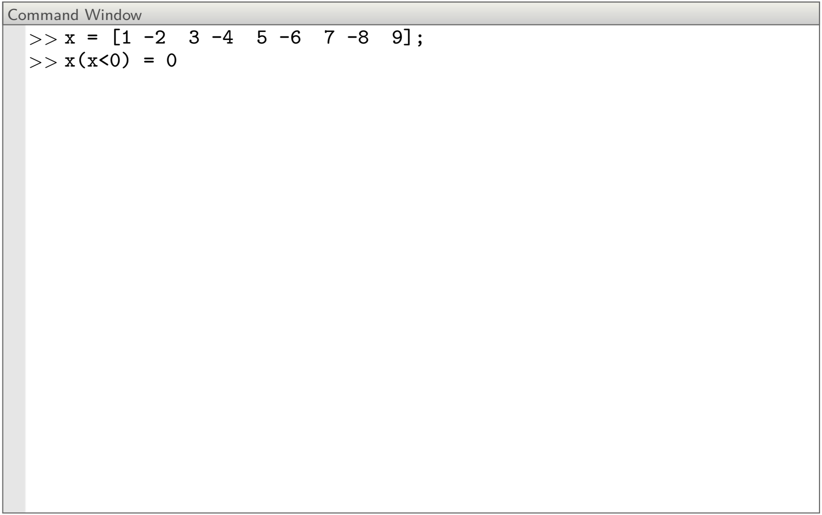
```
>>
```

Comparison and logical operations



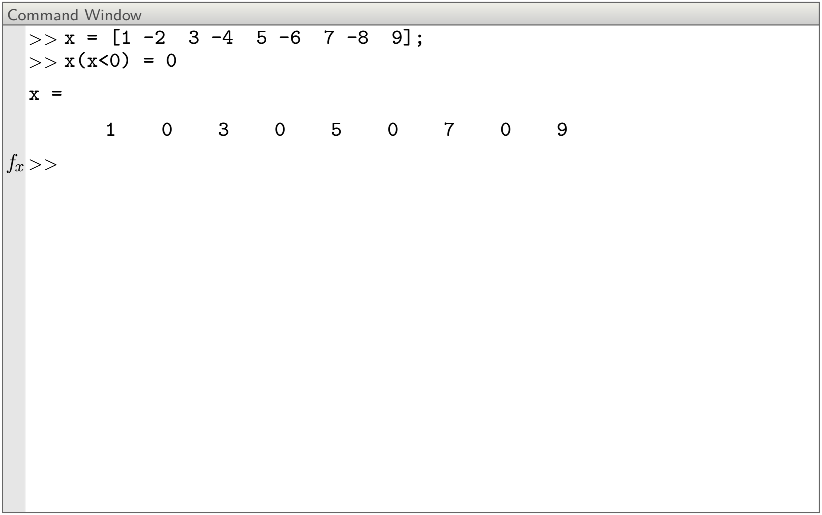
```
>> x = [1 -2  3 -4  5 -6  7 -8  9];
fx >>
```

Comparison and logical operations



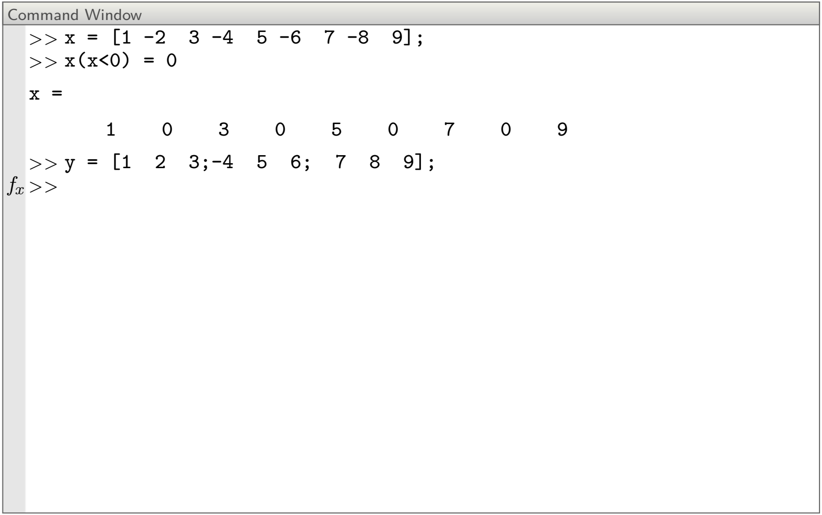
```
>> x = [1 -2  3 -4  5 -6  7 -8  9];
>> x(x<0) = 0
```

Comparison and logical operations



```
>> x = [1 -2  3 -4  5 -6  7 -8  9];
>> x(x<0) = 0

x =

       1     0     3     0     5     0     7     0     9

fx >>
```

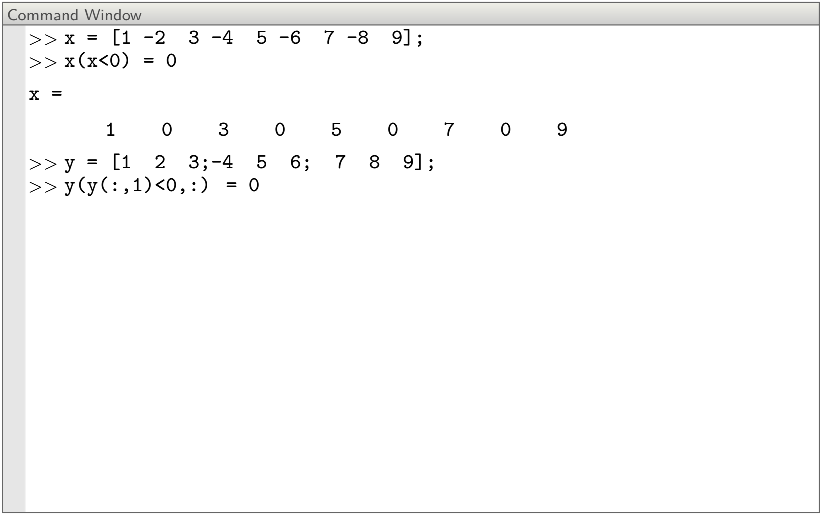
Comparison and logical operations



Command Window

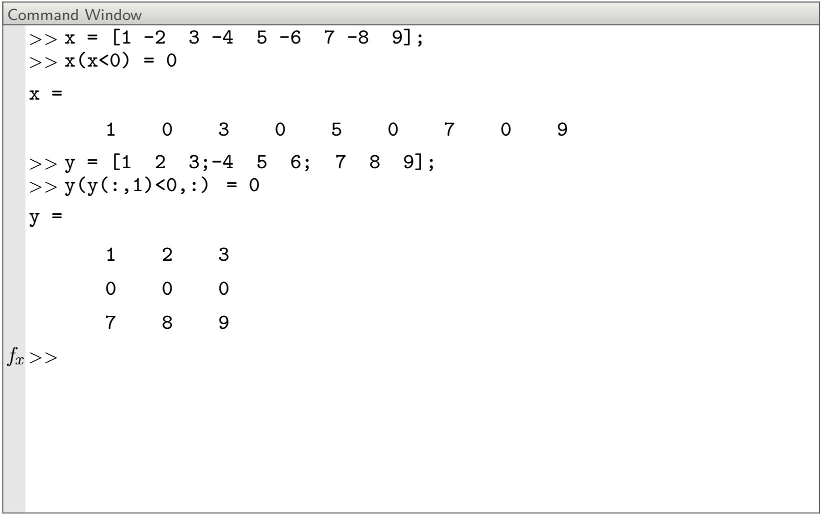
```
>> x = [1 -2  3 -4  5 -6  7 -8  9];
>> x(x<0) = 0

x =

      1     0     3     0     5     0     7     0     9

>> y = [1  2  3;-4  5  6;  7  8  9];
fx >>
```

Comparison and logical operations



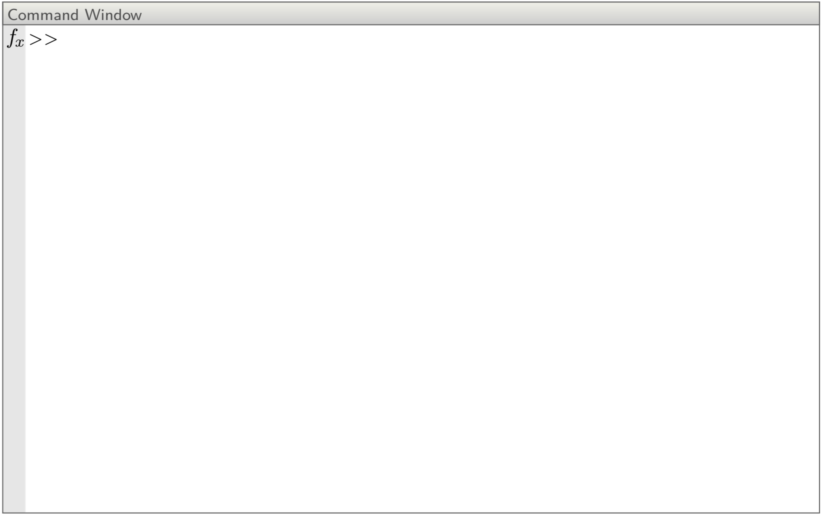
```
>> x = [1 -2  3 -4  5 -6  7 -8  9];
>> x(x<0) = 0

x =

      1     0     3     0     5     0     7     0     9

>> y = [1  2  3;-4  5  6;  7  8  9];
>> y(y(:,1)<0,:)  = 0
```

Comparison and logical operations



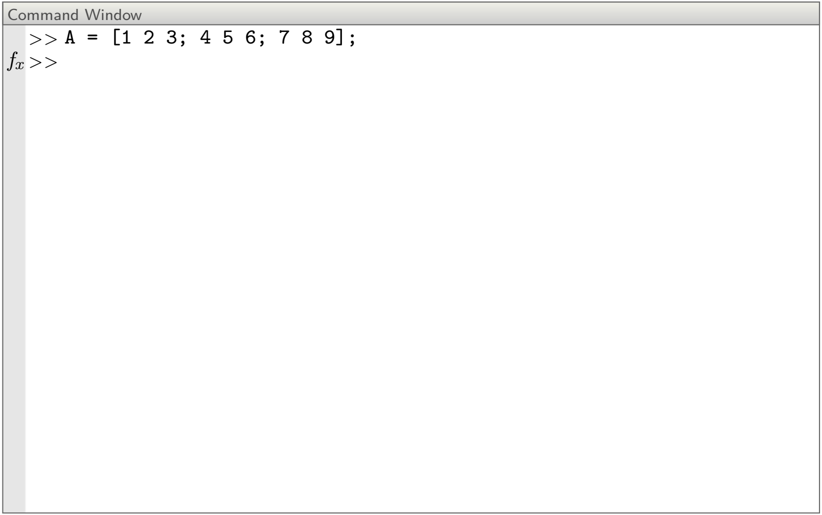
```
>> x = [1 -2  3 -4  5 -6  7 -8  9];
>> x(x<0) = 0

x =

      1     0     3     0     5     0     7     0     9

>> y = [1  2  3;-4  5  6;  7  8  9];
>> y(y(:,1)<0,:)  = 0

y =

      1     2     3

      0     0     0

      7     8     9

fx >>
```

Array manipulation functions: flipud, fliplr, rot90



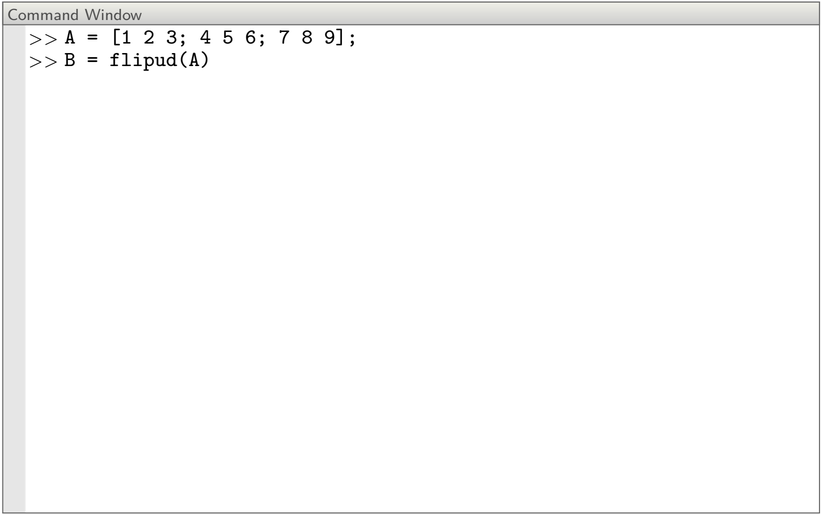
```
>>
```

Array manipulation functions: flipud, fliplr, rot90



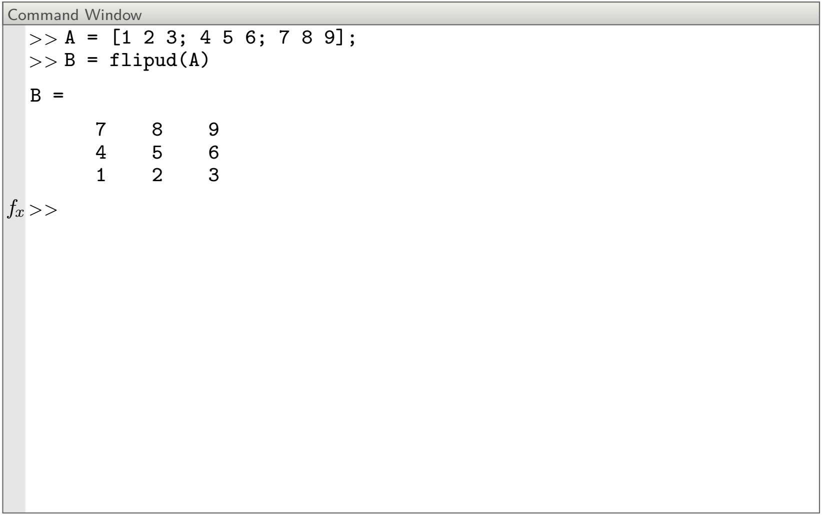
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>>
```

Array manipulation functions: flipud, fliplr, rot90



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

Array manipulation functions: flipud, fliplr, rot90



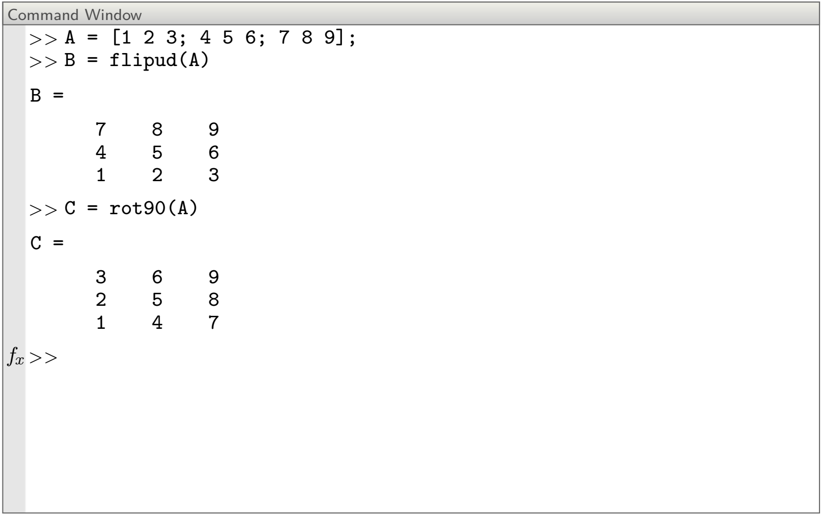
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = flipud(A)

B =

     7     8     9
     4     5     6
     1     2     3

fx >>
```

Array manipulation functions: flipud, fliplr, rot90

Command Window

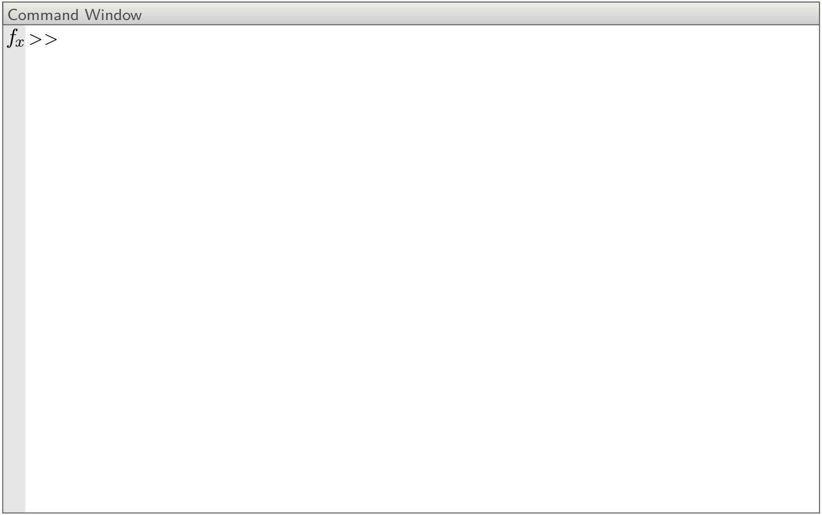
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = flipud(A)

B =

     7     8     9
     4     5     6
     1     2     3

>> C = rot90(A)
```

Array manipulation functions: flipud, fliplr, rot90



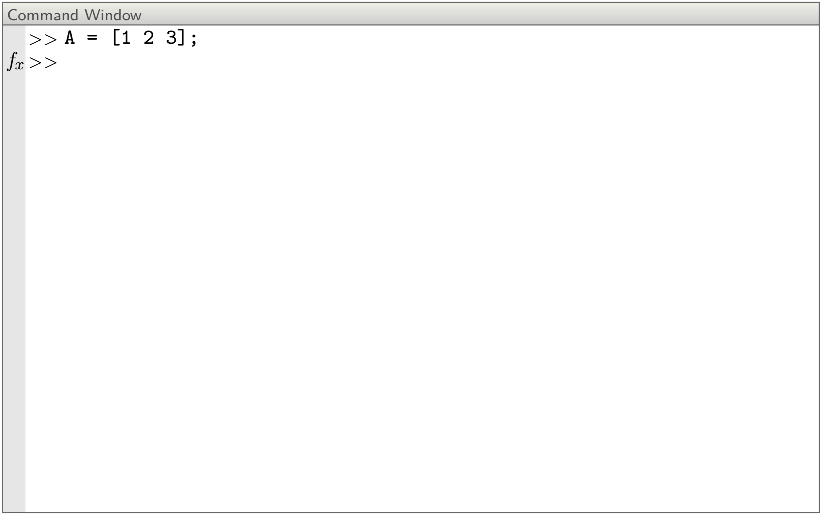
```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> B = flipud(A)

B =

     7     8     9
     4     5     6
     1     2     3

>> C = rot90(A)

C =

     3     6     9
     2     5     8
     1     4     7

fx >>
```

Array manipulation functions: sum



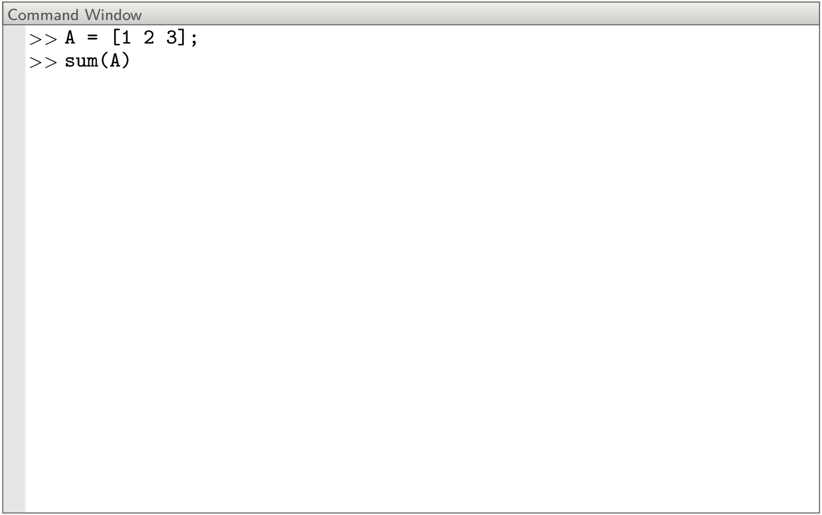
```
fx >>
```

Array manipulation functions: sum



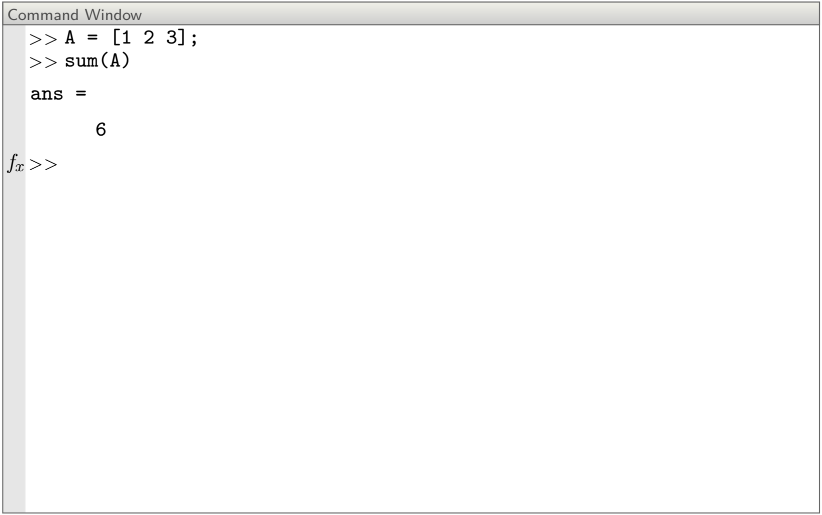
```
>> A = [1 2 3];
fx >>
```

Array manipulation functions: sum



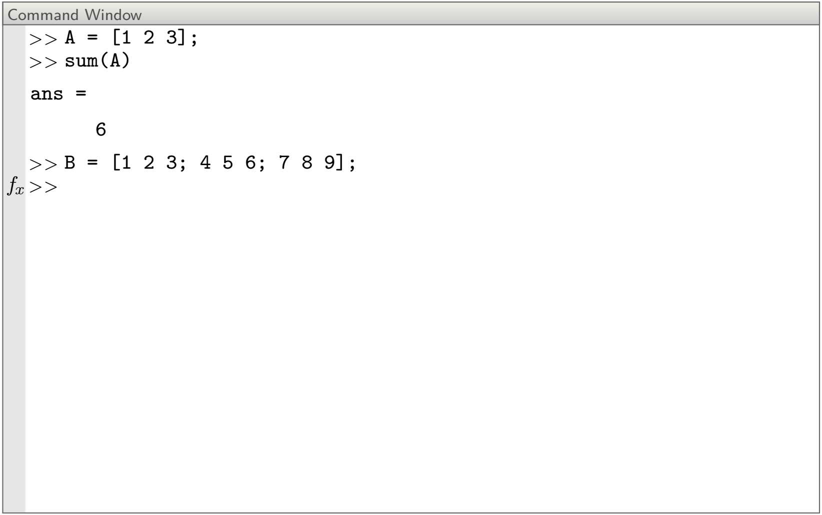
```
>> A = [1 2 3];
>> sum(A)
```

Array manipulation functions: sum



```
>> A = [1 2 3];
>> sum(A)

ans =

     6

fx >>
```

Array manipulation functions: sum



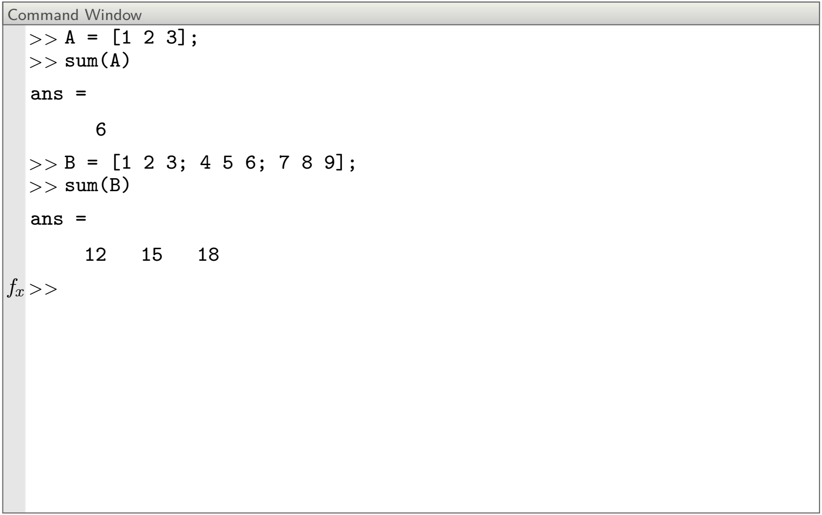
```
>> A = [1 2 3];
>> sum(A)

ans =

     6

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

Array manipulation functions: sum

Command Window

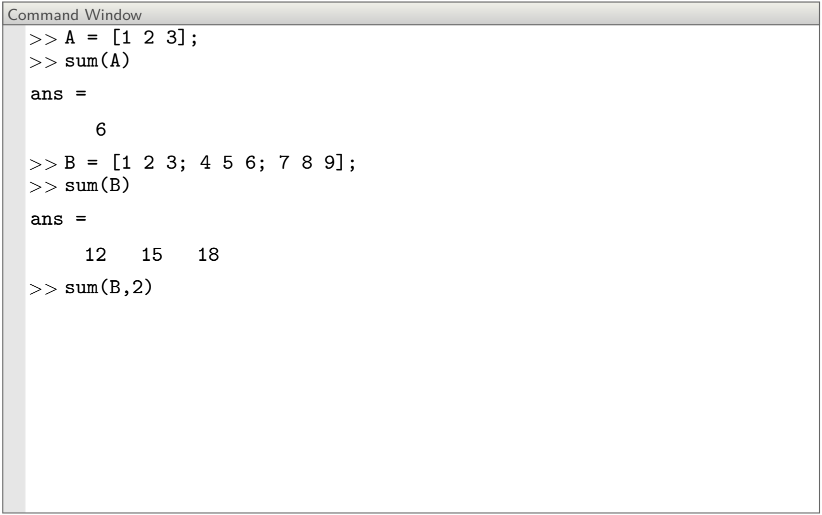
```
>> A = [1 2 3];
>> sum(A)

ans =

     6

>> B = [1 2 3; 4 5 6; 7 8 9];
>> sum(B)
```

Array manipulation functions: sum



```
>> A = [1 2 3];
>> sum(A)

ans =

     6

>> B = [1 2 3; 4 5 6; 7 8 9];
>> sum(B)

ans =

    12    15    18

fx >>
```

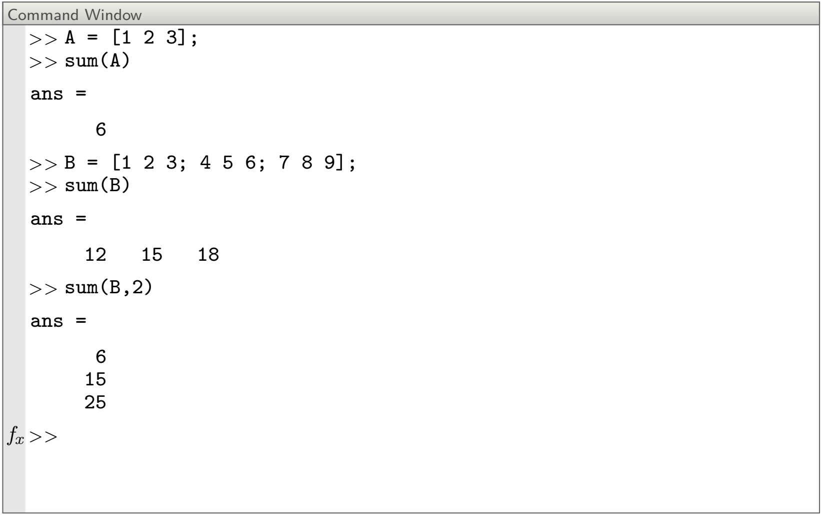
Array manipulation functions: sum



Command Window

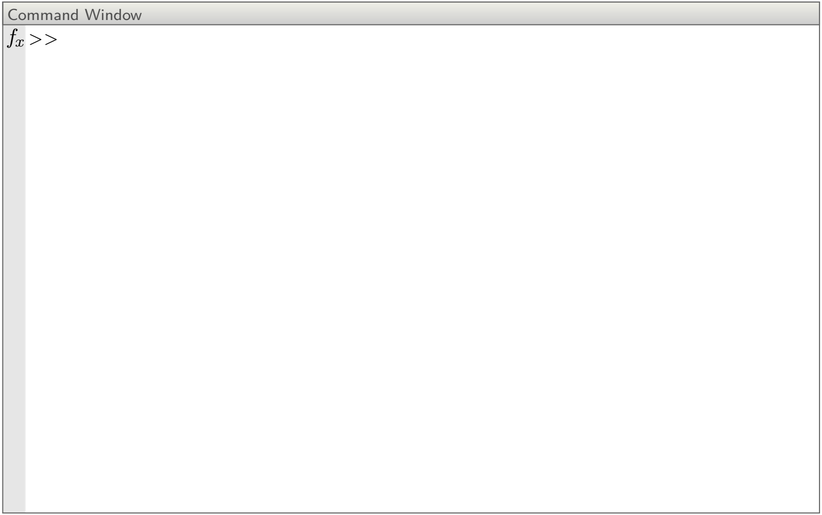
```
>> A = [1 2 3];
>> sum(A)

ans =

     6

>> B = [1 2 3; 4 5 6; 7 8 9];
>> sum(B)

ans =

    12    15    18

>> sum(B,2)
```

Array manipulation functions: sum



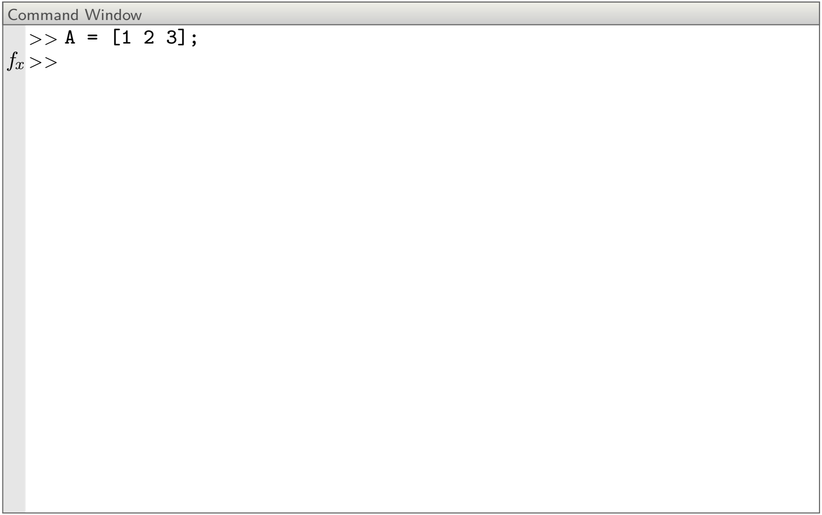
```
>> A = [1 2 3];
>> sum(A)

ans =

      6

>> B = [1 2 3; 4 5 6; 7 8 9];
>> sum(B)

ans =

     12    15    18

>> sum(B,2)

ans =

      6
     15
     25

fx >>
```

Array manipulation functions: max, min



```
>>
```

Array manipulation functions: max, min



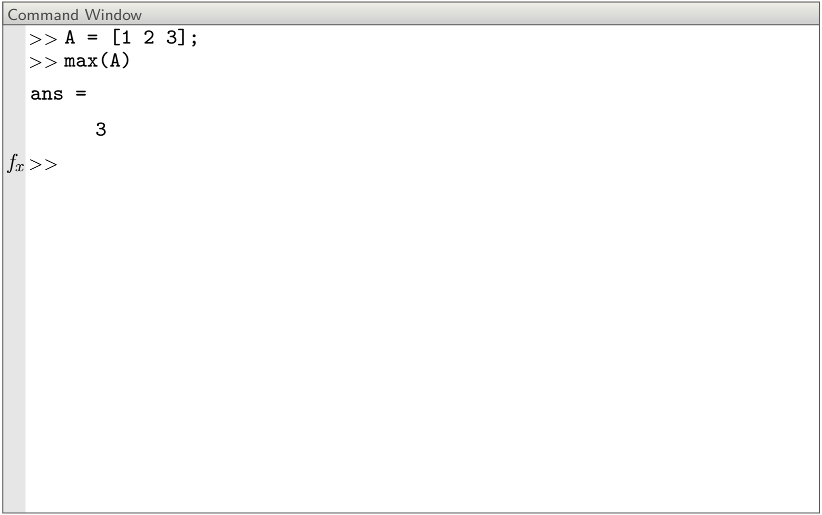
```
>> A = [1 2 3];
fx >>
```

Array manipulation functions: max, min

Command Window

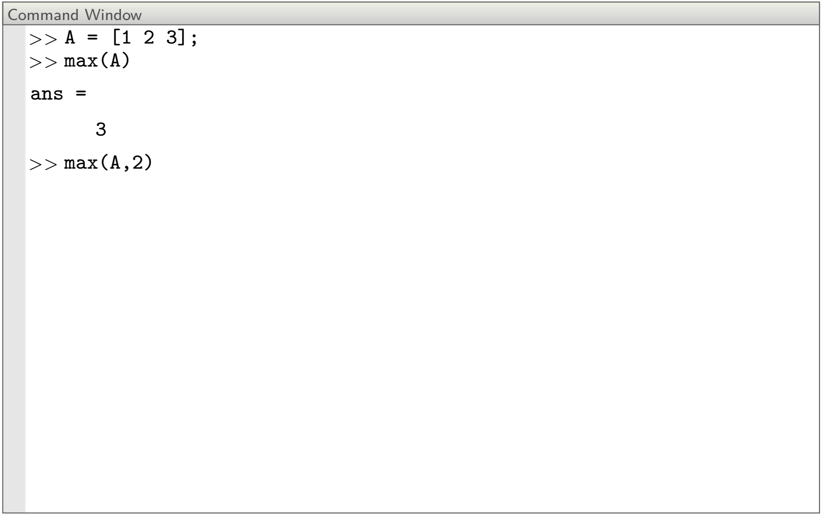
```
>> A = [1 2 3];
>> max(A)
```

Array manipulation functions: max, min



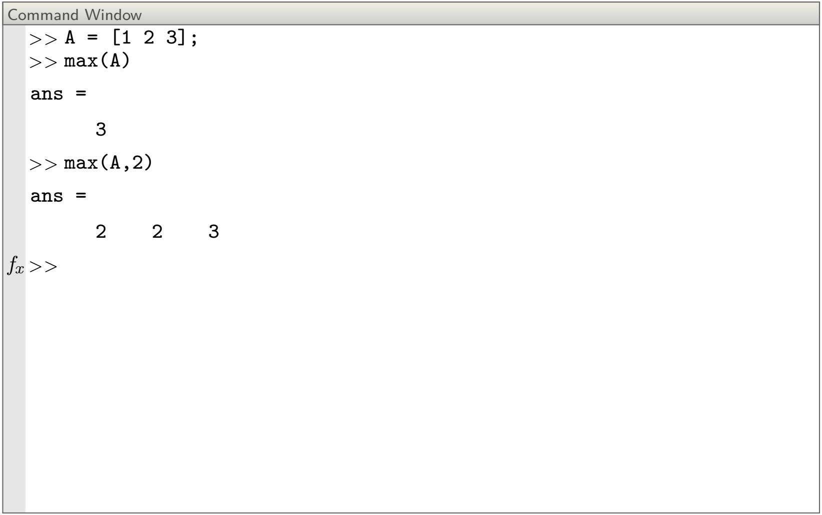
```
>> A = [1 2 3];
>> max(A)

ans =

     3

fx >>
```

Array manipulation functions: max, min



```
>> A = [1 2 3];
>> max(A)

ans =

     3

>> max(A,2)
```

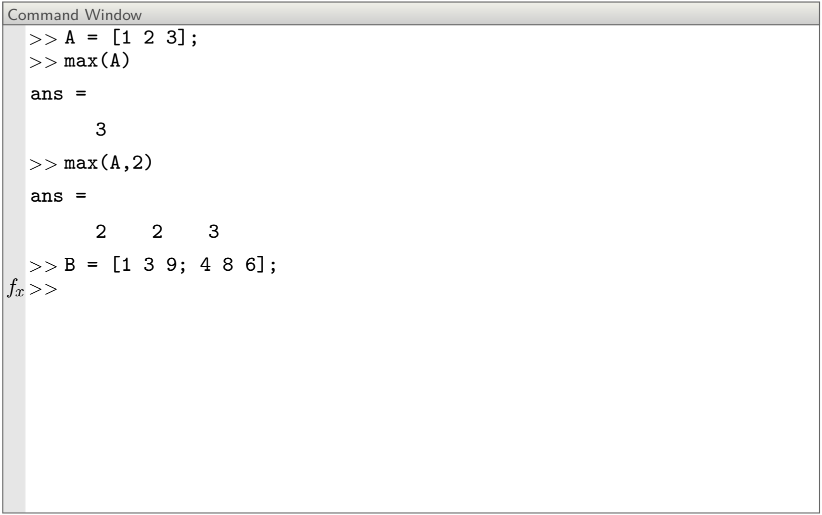
Array manipulation functions: max, min



Command Window

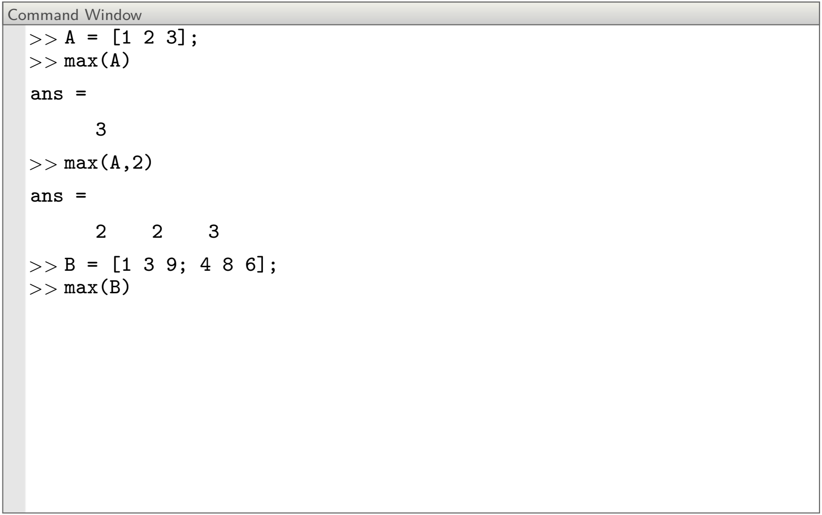
```
>> A = [1 2 3];
>> max(A)
ans =

     3

>> max(A,2)
ans =

     2     2     3

fx >>
```

Array manipulation functions: max, min



```
>> A = [1 2 3];
>> max(A)

ans =

     3

>> max(A,2)

ans =

     2     2     3

>> B = [1 3 9; 4 8 6];
>>
```

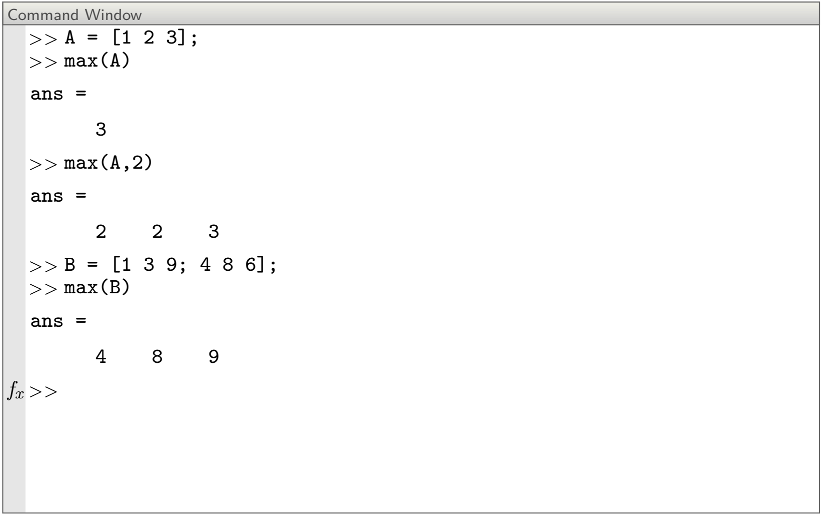
Array manipulation functions: max, min



Command Window

```
>> A = [1 2 3];
>> max(A)

ans =

     3

>> max(A,2)

ans =

     2     2     3

>> B = [1 3 9; 4 8 6];
>> max(B)
```

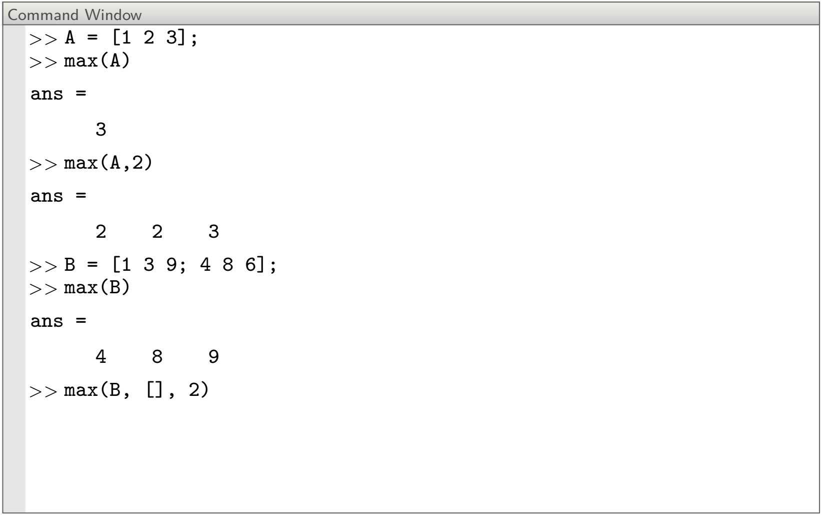
Array manipulation functions: max, min



Command Window

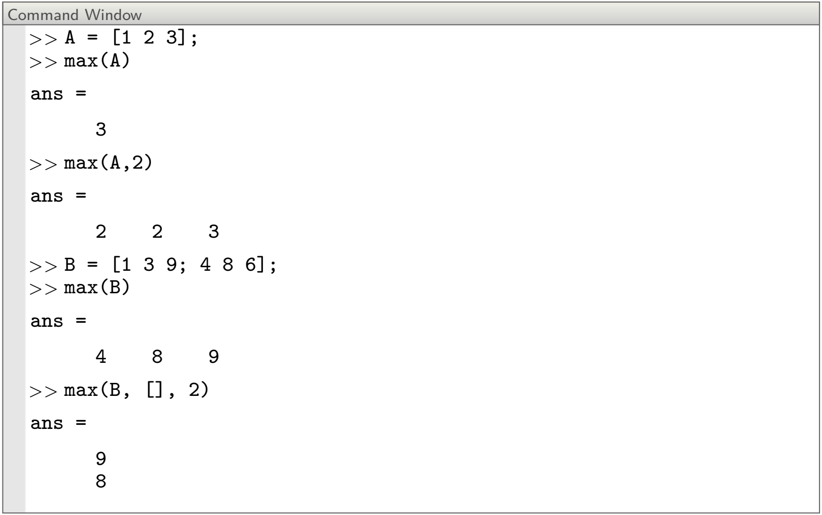
```
>> A = [1 2 3];
>> max(A)

ans =

     3

>> max(A,2)

ans =

     2     2     3

>> B = [1 3 9; 4 8 6];
>> max(B)

ans =

     4     8     9

fx >>
```

Array manipulation functions: max, min



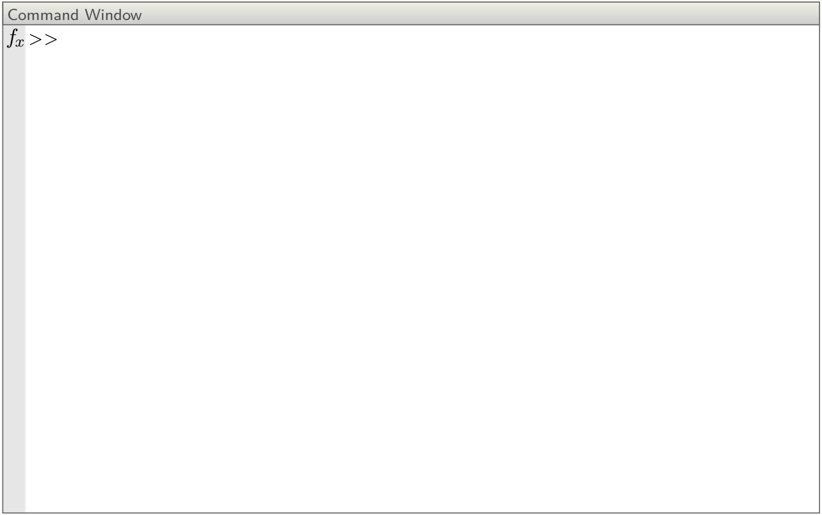
```
>> A = [1 2 3];
>> max(A)

ans =

     3

>> max(A,2)

ans =

     2     2     3

>> B = [1 3 9; 4 8 6];
>> max(B)

ans =

     4     8     9

>> max(B, [], 2)
```

Array manipulation functions: max, min



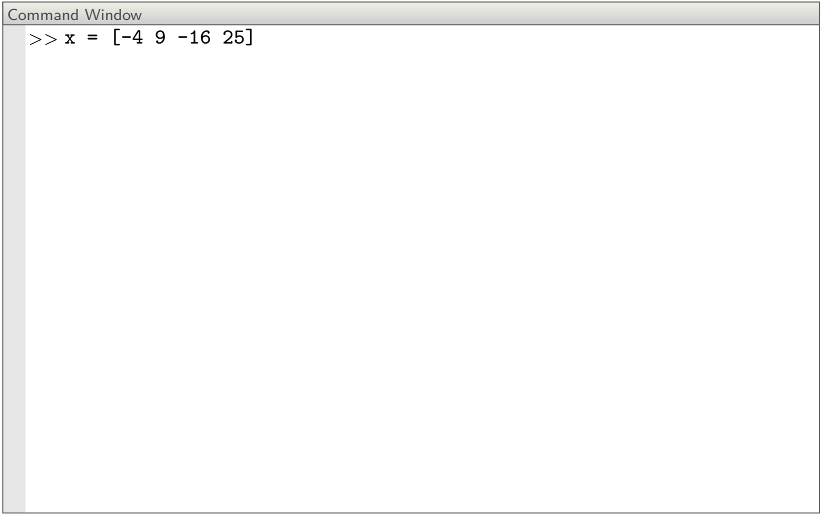
```
>> A = [1 2 3];
>> max(A)

ans =

     3

>> max(A,2)

ans =

     2     2     3

>> B = [1 3 9; 4 8 6];
>> max(B)

ans =

     4     8     9

>> max(B, [], 2)

ans =

     9
     8
```

Common mathematical functions: abs, sqrt



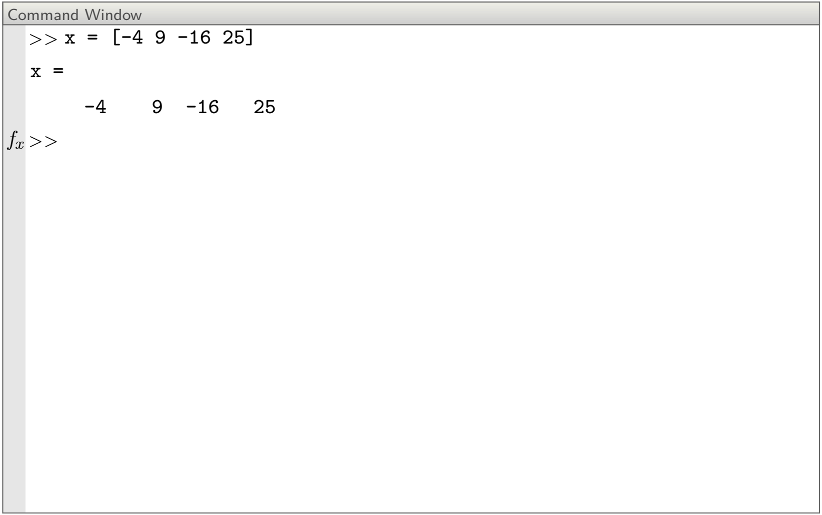
```
fx >>
```

Common mathematical functions: abs, sqrt



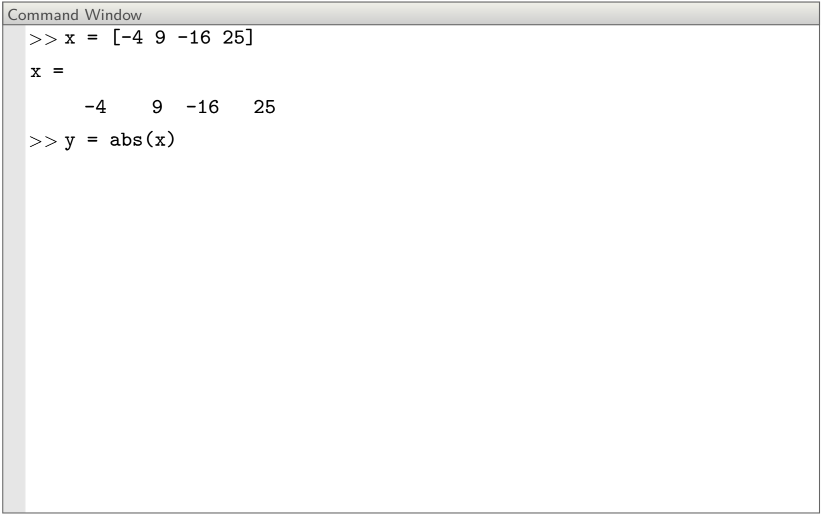
```
>> x = [-4 9 -16 25]
```

Common mathematical functions: abs, sqrt



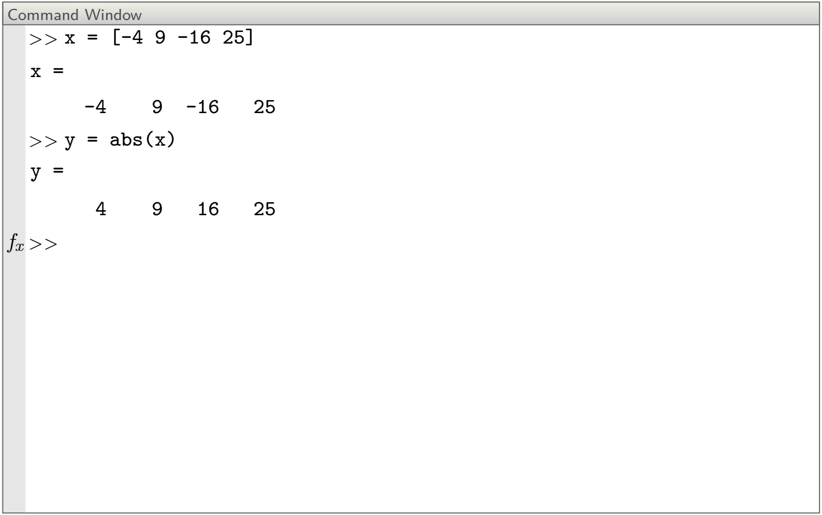
```
>> x = [-4 9 -16 25]

x =

    -4     9   -16    25

>>
```

Common mathematical functions: abs, sqrt



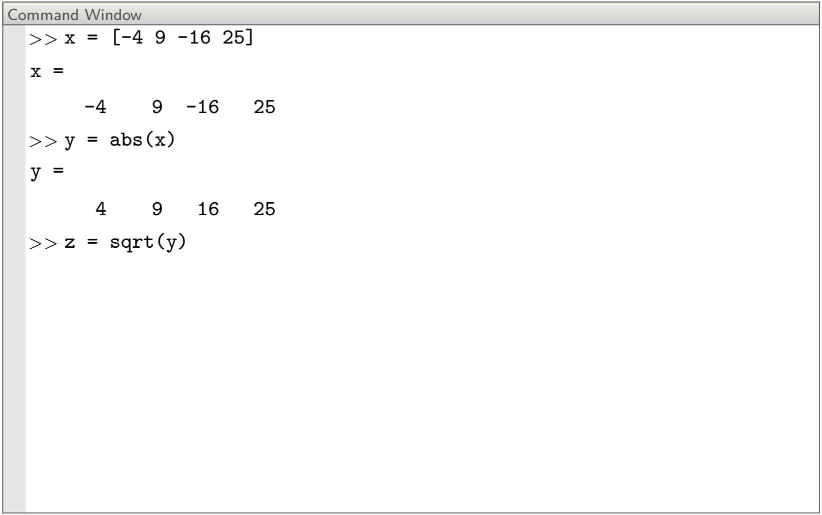
```
>> x = [-4 9 -16 25]

x =

    -4     9   -16    25

>> y = abs(x)
```

Common mathematical functions: abs, sqrt



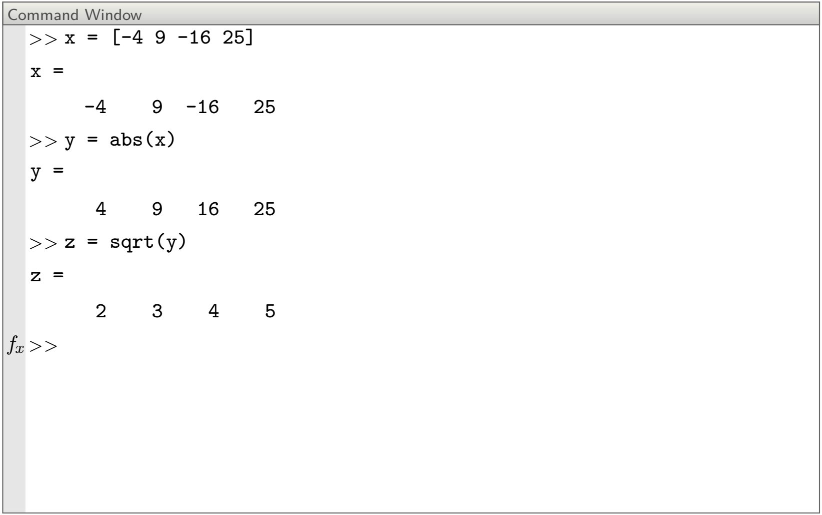
```
>> x = [-4 9 -16 25]
x =

    -4     9   -16    25
>> y = abs(x)
y =

     4     9    16    25

fx >>
```

Common mathematical functions: abs, sqrt



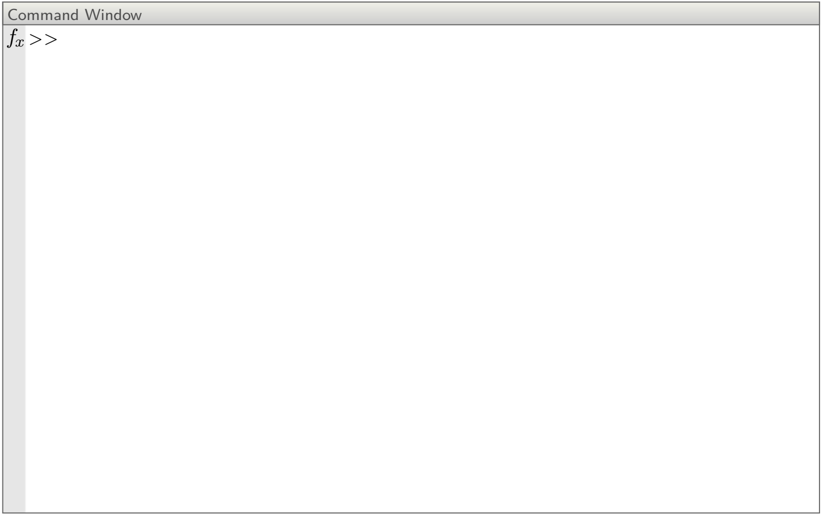
```
>> x = [-4 9 -16 25]
x =

    -4     9   -16    25
>> y = abs(x)
y =

     4     9    16    25
>> z = sqrt(y)
```

Common mathematical functions: abs, sqrt



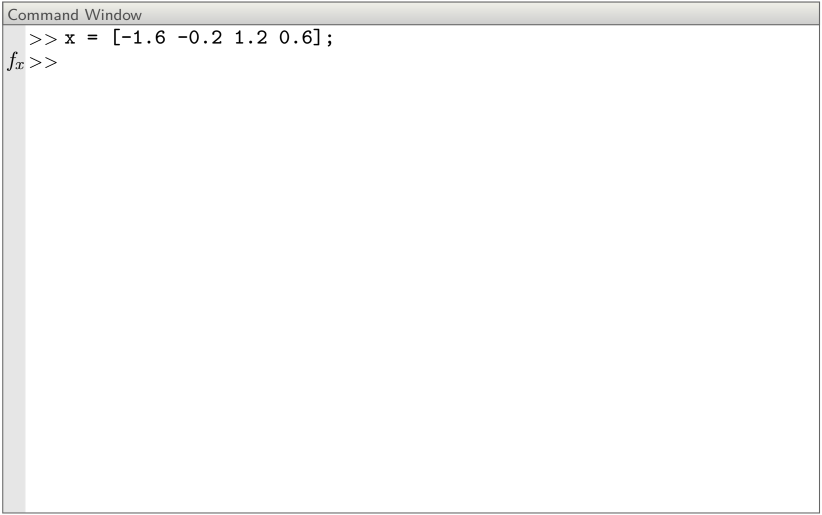
```
>> x = [-4 9 -16 25]
x =

    -4     9   -16    25

>> y = abs(x)
y =

     4     9    16    25

>> z = sqrt(y)
z =

     2     3     4     5

fx >>
```

Common mathematical functions: ceil, fix, floor, round



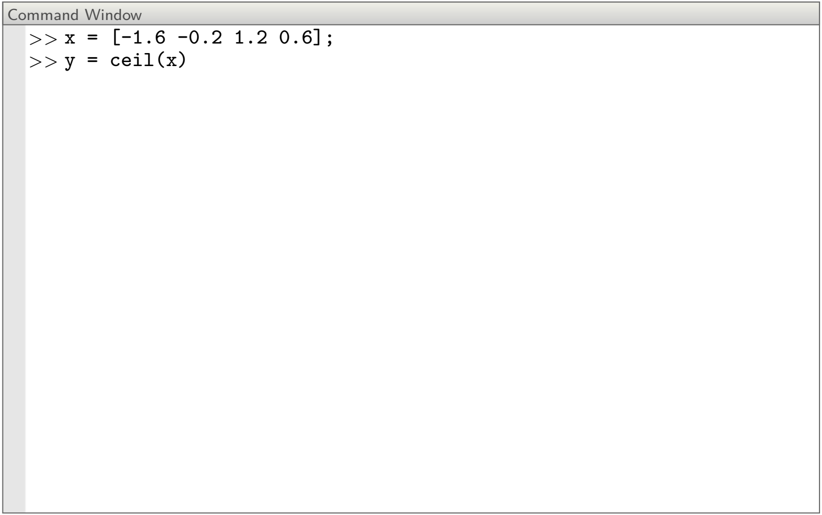
```
fx >>
```

Common mathematical functions: ceil, fix, floor, round



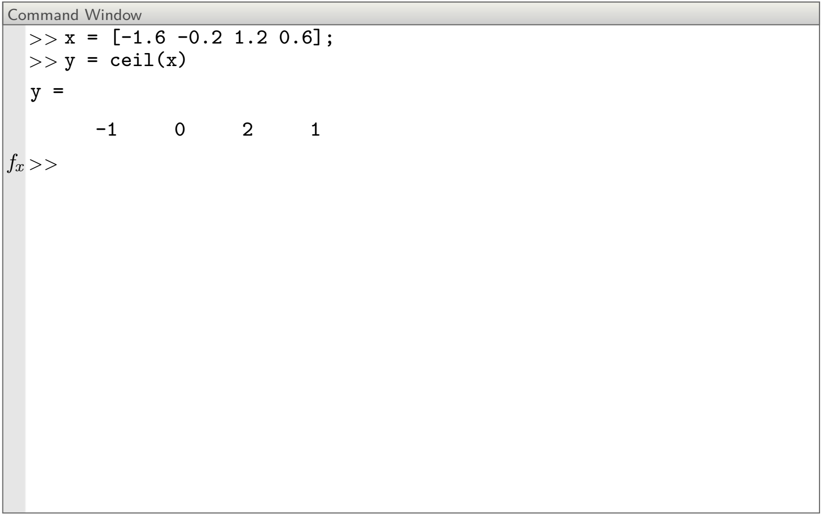
```
>> x = [-1.6 -0.2 1.2 0.6];
fx >>
```

Common mathematical functions: ceil, fix, floor, round



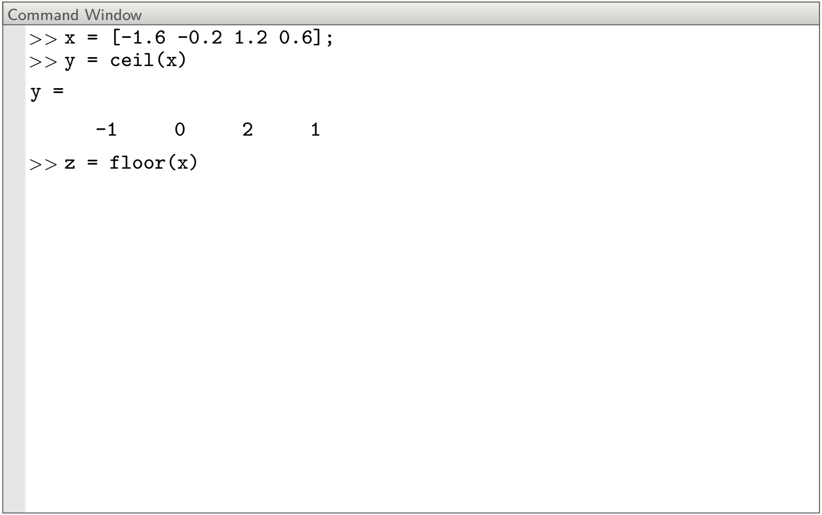
```
>> x = [-1.6 -0.2 1.2 0.6];
>> y = ceil(x)
```

Common mathematical functions: ceil, fix, floor, round



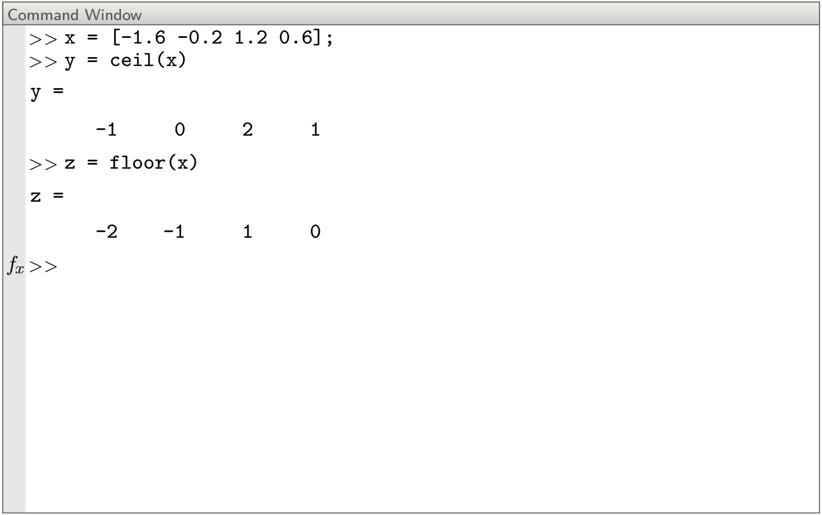
```
>> x = [-1.6 -0.2 1.2 0.6];
>> y = ceil(x)

y =

    -1     0     2     1

fx >>
```

Common mathematical functions: ceil, fix, floor, round



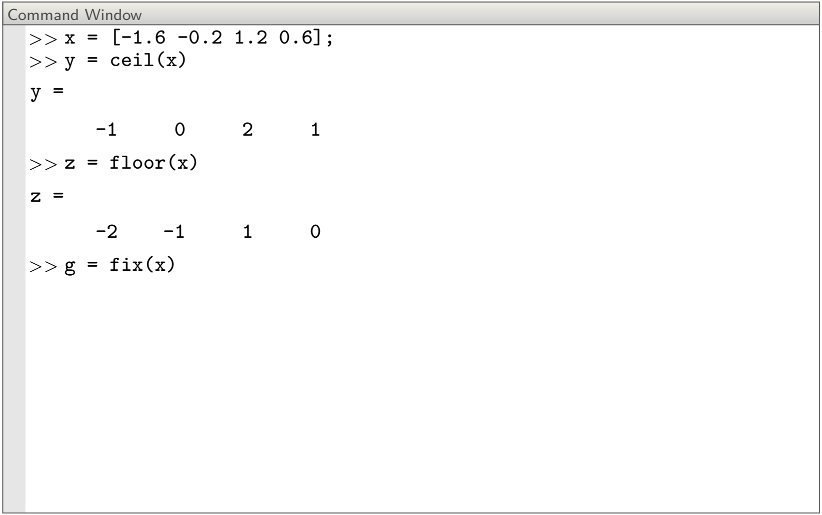
```
>> x = [-1.6 -0.2 1.2 0.6];
>> y = ceil(x)
y =

    -1     0     2     1

>> z = floor(x)
```

Common mathematical functions: ceil, fix, floor, round



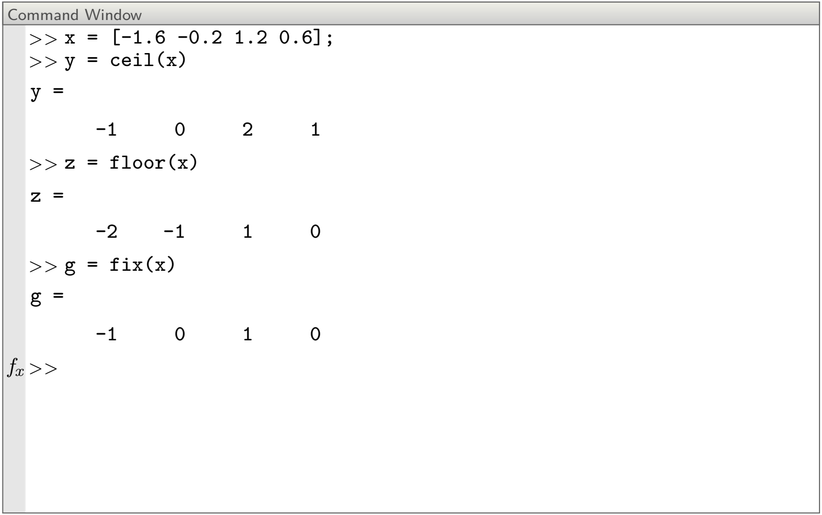
```
>> x = [-1.6 -0.2 1.2 0.6];
>> y = ceil(x)

y =

    -1     0     2     1

>> z = floor(x)

z =

    -2    -1     1     0

fx >>
```

Common mathematical functions: ceil, fix, floor, round



```
>> x = [-1.6 -0.2 1.2 0.6];
>> y = ceil(x)
y =

     -1      0      2      1

>> z = floor(x)
z =

     -2     -1      1      0

>> g = fix(x)
```

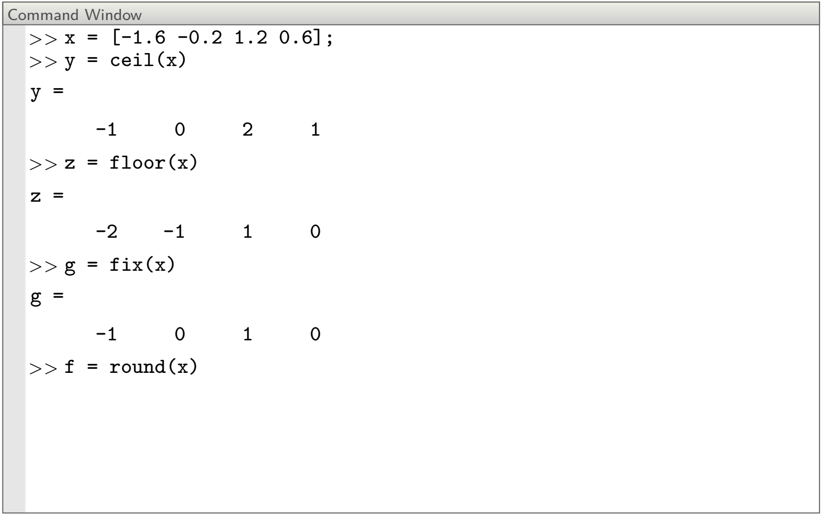
Common mathematical functions: ceil, fix, floor, round



Command Window

```
>> x = [-1.6 -0.2 1.2 0.6];
>> y = ceil(x)
y =

    -1     0     2     1

>> z = floor(x)
z =

    -2    -1     1     0

>> g = fix(x)
g =

    -1     0     1     0

>>
```

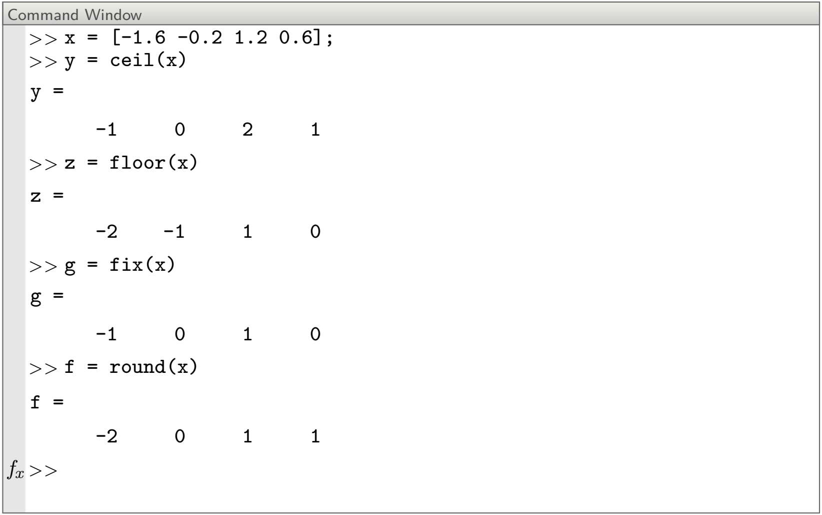
Common mathematical functions: ceil, fix, floor, round



Command Window

```
>> x = [-1.6 -0.2 1.2 0.6];
>> y = ceil(x)
y =

    -1     0     2     1

>> z = floor(x)
z =

    -2    -1     1     0

>> g = fix(x)
g =

    -1     0     1     0

>> f = round(x)
```

Common mathematical functions: ceil, fix, floor, round



```
>> x = [-1.6 -0.2 1.2 0.6];
>> y = ceil(x)
y =

     -1     0     2     1

>> z = floor(x)

z =

     -2    -1     1     0

>> g = fix(x)

g =

     -1     0     1     0

>> f = round(x)

f =

     -2     0     1     1

fx >>
```

Basic sentence

Basic sentence

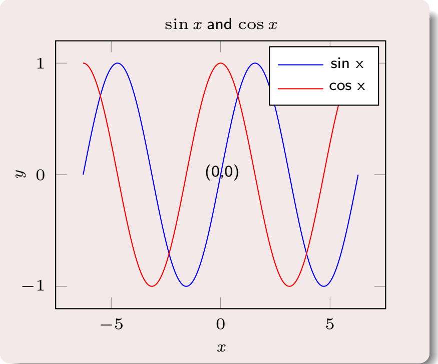
- for .. end
- if .. else .. end
- while .. end
- switch .. case .. end

Example: Find odd sums within 1-10

```
% sum of the odd numbers between 1 and 10
x = 0;
for i = 1:10
    if mod(i,2)
       x= x + i;
    end
end
```

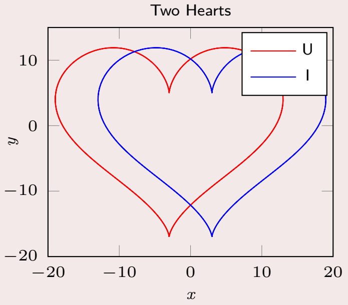
Two-dimensional curve: sin & cos

```
x = -2*pi:0.1:2*pi;
y1 = sin(x);
y2 = cos(x);
plot(x, y1, '-b');
hold on
plot(x, y2, '-r');

xlabel('x')
ylabel('y')
text(0,0, '(0,0)')
legend('sin x', 'cos x')
title('sin x and cos x')
```



Two-dimensional curve: Two Hearts

```
t = 0:pi/180:4*pi;
x = 16*sin(t).^3;
y = 13*cos(t)-5*cos(2*t)...
    -2*cos(3*t)-cos(4*t);

plot(x-3,y,'-r', x+3,y,'-b');
xlabel('x');
ylabel('y');
axis([-20, 20, -20, 15]);
title('Two Heart')
legend('U', 'I')
```



Two-dimensional curve: summary of plot usage

- plot: plot(x,y); plot(x,y,s), plot(x1,y1,s1,x2,y2,s2,...)

1	b	blue	.	point	-	solid
2	g	green	o	circle	:	dotted
3	r	red	x	x-mark	-.	dashdot
4	c	cyan	+	plus	--	dashed
5	m	magenta	*	star	(none)	no line
6	y	yellow	s	square		
7	k	black	d	diamond		
8	w	white	v	triangle (down)		
9			ˆ	triangle (up)		
10			<	triangle (left)		
11			>	triangle (right)		
12			p	pentagram		
13			h	hexagram		

Two-dimensional curve: graphic control sentence

- title
- xlabel; ylabel
- text
- legend
- grid on / grid off / g rid minor
- axis([xmin xmax ymin ymax]), xlim([xmin, xmax])

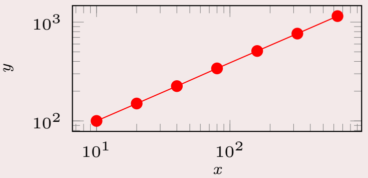
2D curve: logarithmic and polar coordinate system

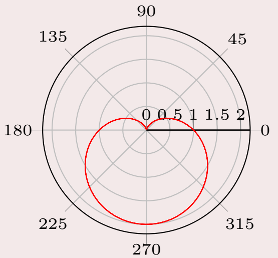
loglog, semilogx

```
x = 10*2.^[0:6];
y = [100 150 225 340 ...
     510 765 1150];
loglog(x,y,'.-r')

xlim([0.5e1,0.8e3])
ylim([0.8e2,1.4e3])
xlabel('x')
ylabel('y')
```

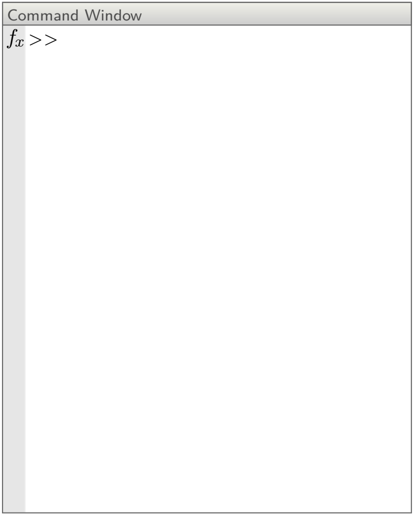
polar

```
theta = 0:pi/180:4*pi;
r = 1-sin(theta);
polar(theta,r,'-r');
```



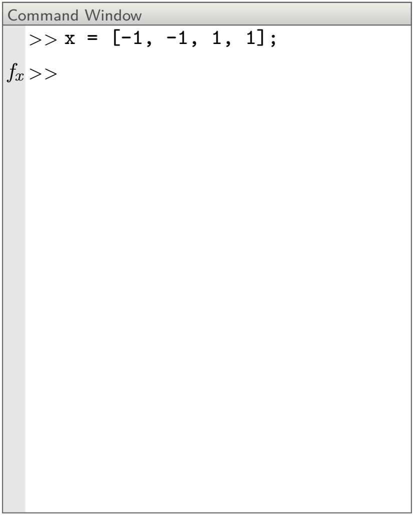


Two-dimensional filling



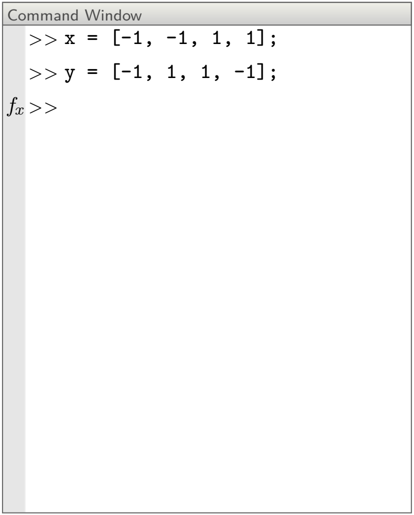
```
>>
```

Two-dimensional filling



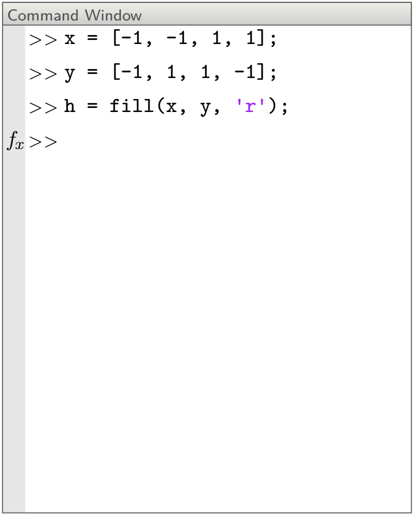
```
>> x = [-1, -1, 1, 1];
>>
```

Two-dimensional filling



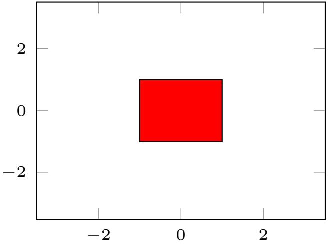
```
>> x = [-1, -1, 1, 1];
>> y = [-1, 1, 1, -1];
fx >>
```

Two-dimensional filling

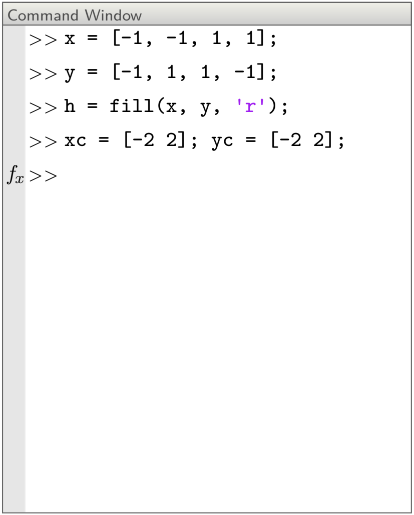


Command Window

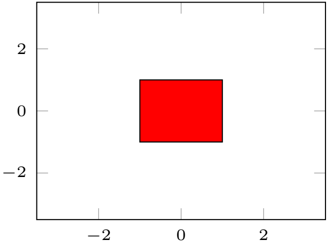
```
>> x = [-1, -1, 1, 1];
>> y = [-1, 1, 1, -1];
>> h = fill(x, y, 'r');
fx >>
```



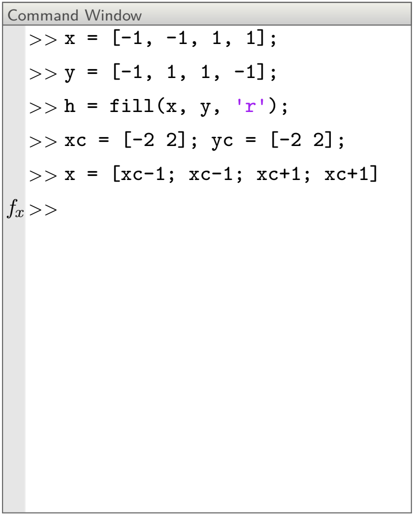
Two-dimensional filling



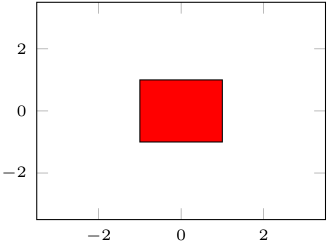
```
>> x = [-1, -1, 1, 1];
>> y = [-1, 1, 1, -1];
>> h = fill(x, y, 'r');
>> xc = [-2 2]; yc = [-2 2];
fx >>
```



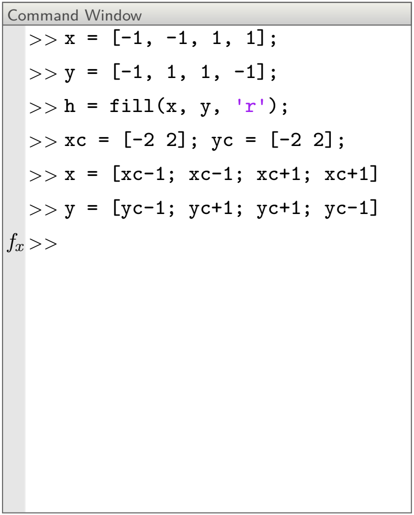
Two-dimensional filling



```
>> x = [-1, -1, 1, 1];
>> y = [-1, 1, 1, -1];
>> h = fill(x, y, 'r');
>> xc = [-2 2]; yc = [-2 2];
>> x = [xc-1; xc-1; xc+1; xc+1]
fx >>
```

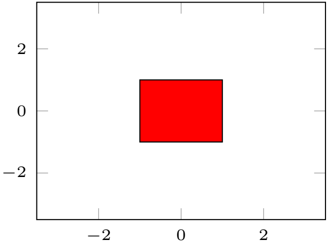


Two-dimensional filling

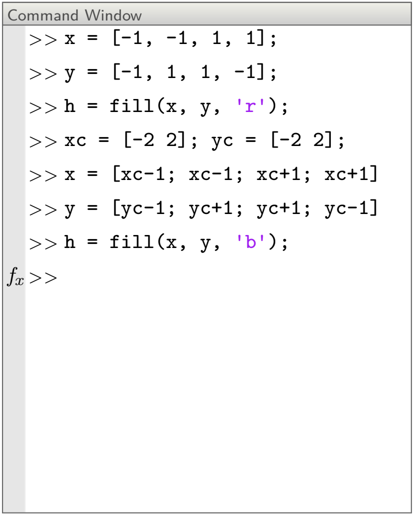


Command Window

```
>> x = [-1, -1, 1, 1];
>> y = [-1, 1, 1, -1];
>> h = fill(x, y, 'r');
>> xc = [-2 2]; yc = [-2 2];
>> x = [xc-1; xc-1; xc+1; xc+1]
>> y = [yc-1; yc+1; yc+1; yc-1]
fx >>
```

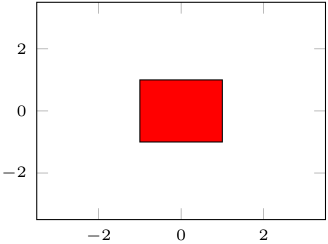


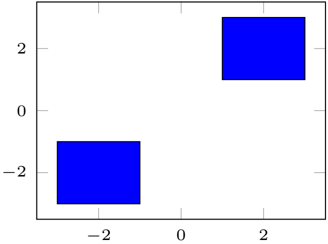
Two-dimensional filling



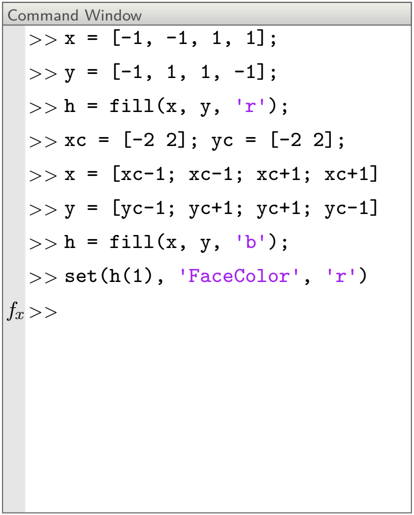
Command Window

```
>> x = [-1, -1, 1, 1];
>> y = [-1, 1, 1, -1];
>> h = fill(x, y, 'r');
>> xc = [-2 2]; yc = [-2 2];
>> x = [xc-1; xc-1; xc+1; xc+1]
>> y = [yc-1; yc+1; yc+1; yc-1]
>> h = fill(x, y, 'b');
fx >>
```

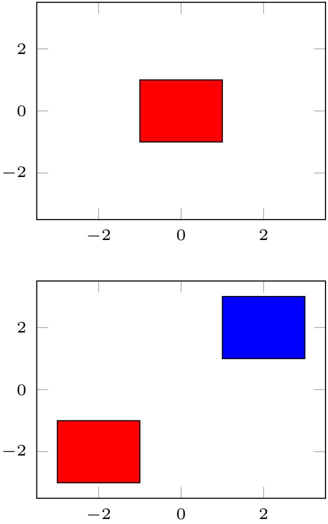




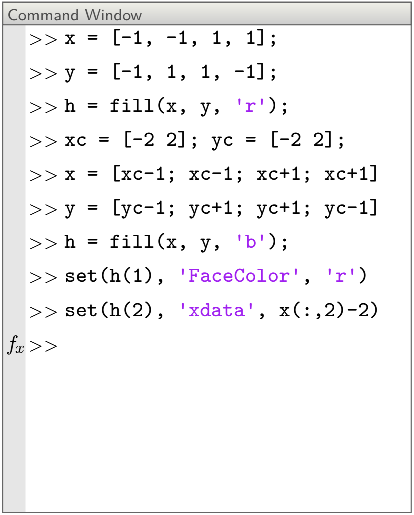
Two-dimensional filling



```
>> x = [-1, -1, 1, 1];
>> y = [-1, 1, 1, -1];
>> h = fill(x, y, 'r');
>> xc = [-2 2]; yc = [-2 2];
>> x = [xc-1; xc-1; xc+1; xc+1]
>> y = [yc-1; yc+1; yc+1; yc-1]
>> h = fill(x, y, 'b');
>> set(h(1), 'FaceColor', 'r')
fx >>
```

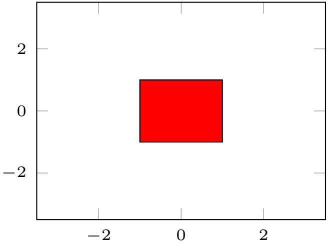


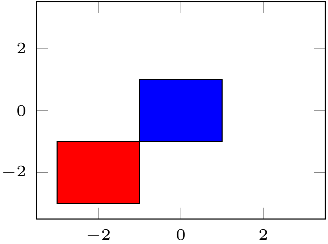
Two-dimensional filling



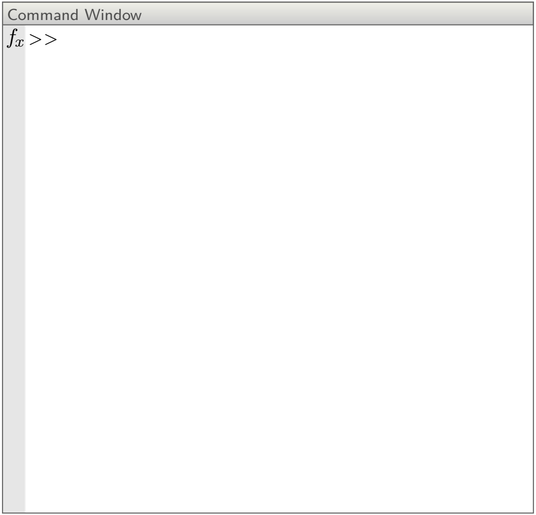
Command Window

```
>> x = [-1, -1, 1, 1];
>> y = [-1, 1, 1, -1];
>> h = fill(x, y, 'r');
>> xc = [-2 2]; yc = [-2 2];
>> x = [xc-1; xc-1; xc+1; xc+1]
>> y = [yc-1; yc+1; yc+1; yc-1]
>> h = fill(x, y, 'b');
>> set(h(1), 'FaceColor', 'r')
>> set(h(2), 'xdata', x(:,2)-2)
fx >>
```



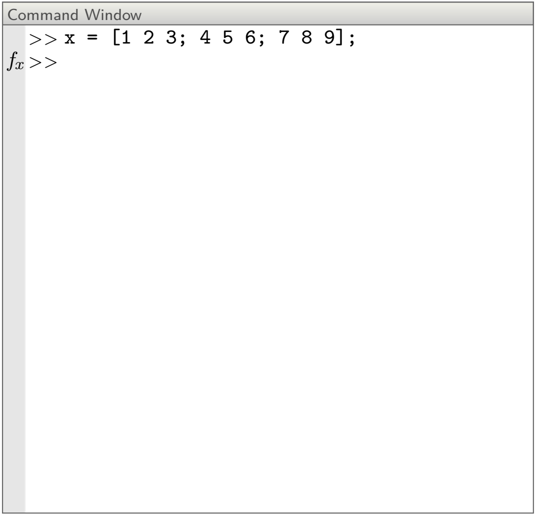


Array display



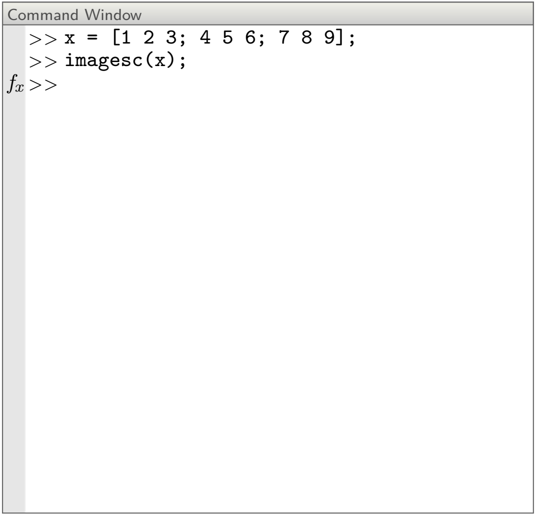
```
fx >>
```

Array display



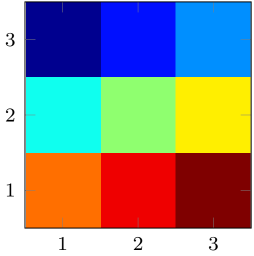
```
>> x = [1 2 3; 4 5 6; 7 8 9];
>>
```

Array display

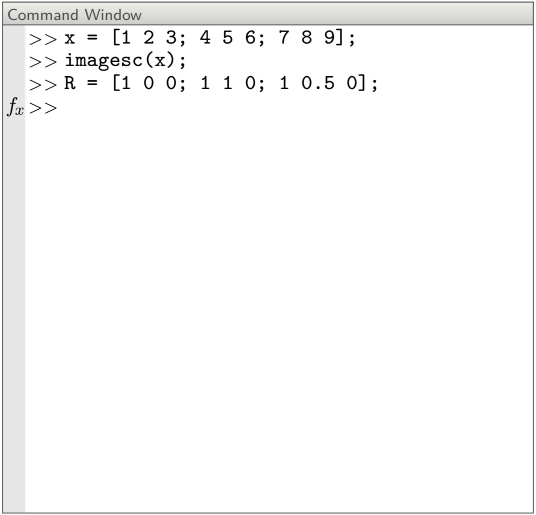


Command Window

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

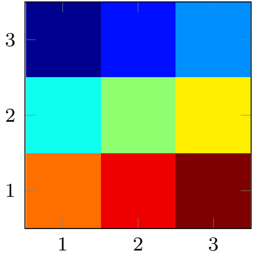


Array display

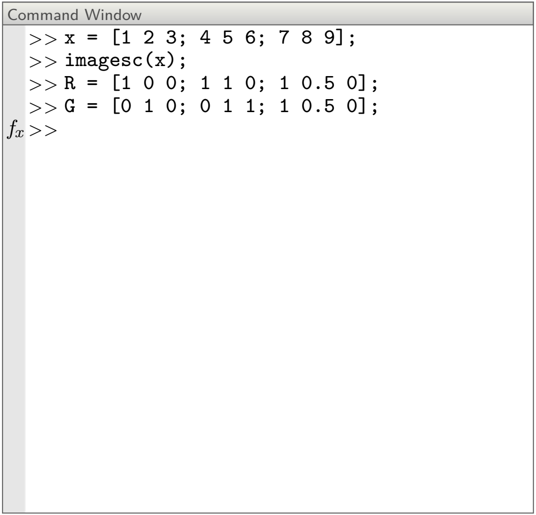


Command Window

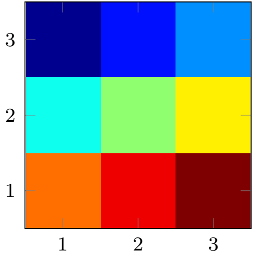
```
>> x = [1 2 3; 4 5 6; 7 8 9];
>> imagesc(x);
>> R = [1 0 0; 1 1 0; 1 0.5 0];
fx >>
```



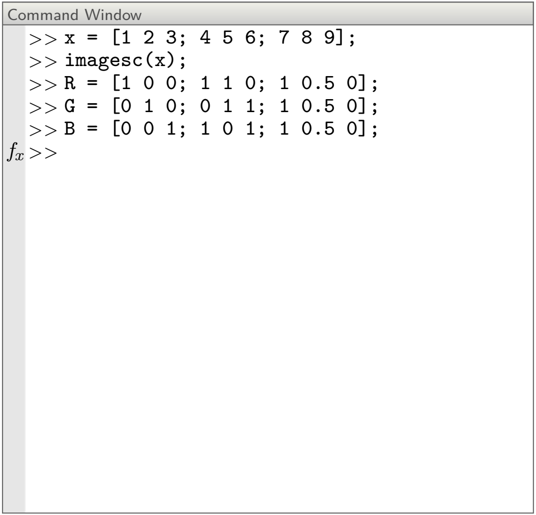
Array display



```
>> x = [1 2 3; 4 5 6; 7 8 9];
>> imagesc(x);
>> R = [1 0 0; 1 1 0; 1 0.5 0];
>> G = [0 1 0; 0 1 1; 1 0.5 0];
fx >>
```

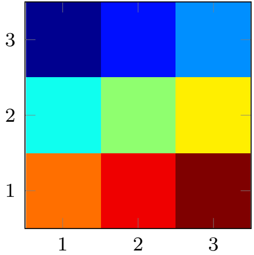


Array display

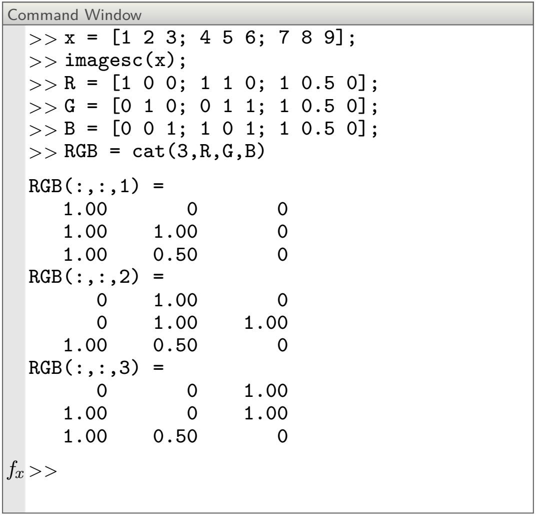


Command Window

```
>> x = [1 2 3; 4 5 6; 7 8 9];
>> imagesc(x);
>> R = [1 0 0; 1 1 0; 1 0.5 0];
>> G = [0 1 0; 0 1 1; 1 0.5 0];
>> B = [0 0 1; 1 0 1; 1 0.5 0];
fx >>
```

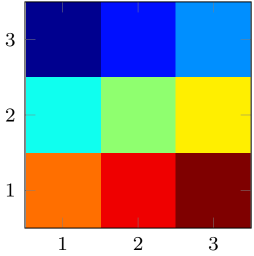


Array display

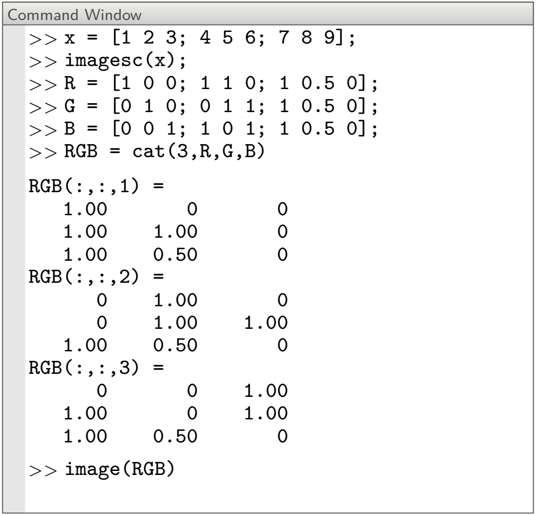


Command Window

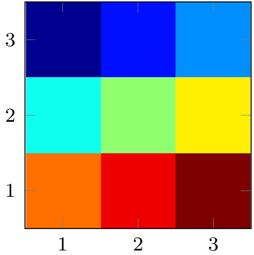
```
>> x = [1 2 3; 4 5 6; 7 8 9];
>> imagesc(x);
>> R = [1 0 0; 1 1 0; 1 0.5 0];
>> G = [0 1 0; 0 1 1; 1 0.5 0];
>> B = [0 0 1; 1 0 1; 1 0.5 0];
>> RGB = cat(3,R,G,B)

RGB(:,:,1) =
   1.00        0        0
   1.00     1.00        0
   1.00     0.50        0
RGB(:,:,2) =
      0     1.00        0
      0     1.00     1.00
   1.00     0.50        0
RGB(:,:,3) =
      0        0     1.00
   1.00        0     1.00
   1.00     0.50        0

fx >>
```

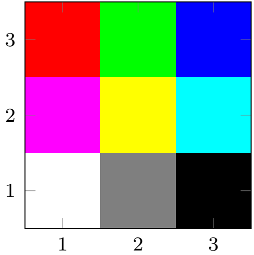


Array display



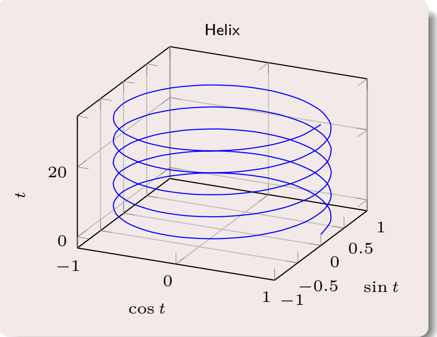
```
>> x = [1 2 3; 4 5 6; 7 8 9];
>> imagesc(x);
>> R = [1 0 0; 1 1 0; 1 0.5 0];
>> G = [0 1 0; 0 1 1; 1 0.5 0];
>> B = [0 0 1; 1 0 1; 1 0.5 0];
>> RGB = cat(3,R,G,B)

RGB(:,:,1) =
   1.00        0        0
   1.00     1.00        0
   1.00     0.50        0
RGB(:,:,2) =
      0     1.00        0
      0     1.00     1.00
   1.00     0.50        0
RGB(:,:,3) =
      0        0     1.00
   1.00        0     1.00
   1.00     0.50        0

>> image(RGB)
```



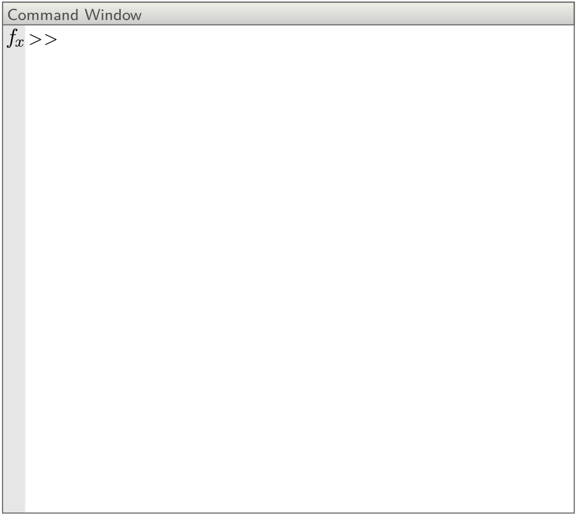


Three-dimensional curve: Helix

```
t = 0:pi/50:10*pi;
x = sin(t);
y = cos(t);
z = t;
plot3(x,y,z)

title('Helix')
xlabel('sin t')
ylabel('cos t')
zlabel('t')
grid on
```



3D surface: supplementary function meshgrid



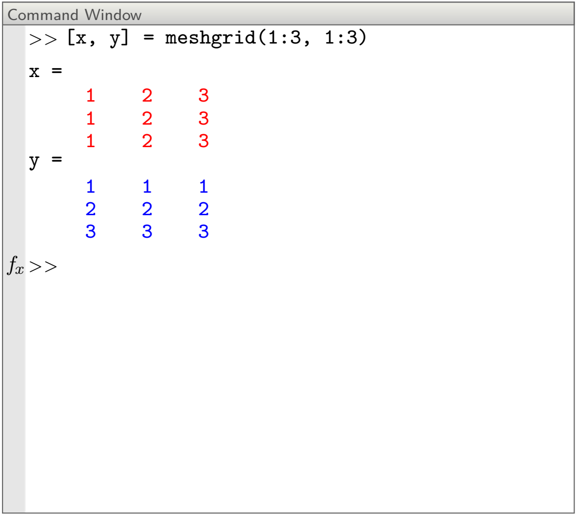
```
fx >>
```

$(1, 1)$	$(2, 1)$	$(3, 1)$
$(1, 2)$	$(2, 2)$	$(3, 2)$
$(1, 3)$	$(2, 3)$	$(3, 3)$

2	1	2
1	0	1
2	1	2

$\sqrt{2}$	1	$\sqrt{2}$
1	0	1
$\sqrt{2}$	1	$\sqrt{2}$

3D surface: supplementary function meshgrid



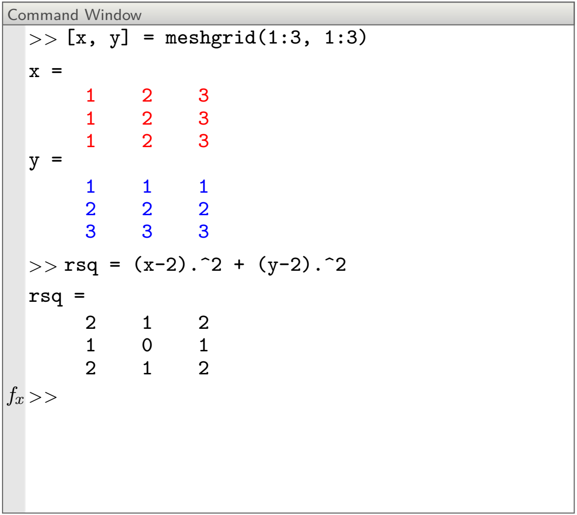
```
>> [x, y] = meshgrid(1:3, 1:3)

x =

     1     2     3
     1     2     3
     1     2     3

y =

     1     1     1
     2     2     2
     3     3     3

fx >>
```

$(1, 1)$	$(2, 1)$	$(3, 1)$
$(1, 2)$	$(2, 2)$	$(3, 2)$
$(1, 3)$	$(2, 3)$	$(3, 3)$

2	1	2
1	0	1
2	1	2

$\sqrt{2}$	1	$\sqrt{2}$
1	0	1
$\sqrt{2}$	1	$\sqrt{2}$

3D surface: supplementary function meshgrid



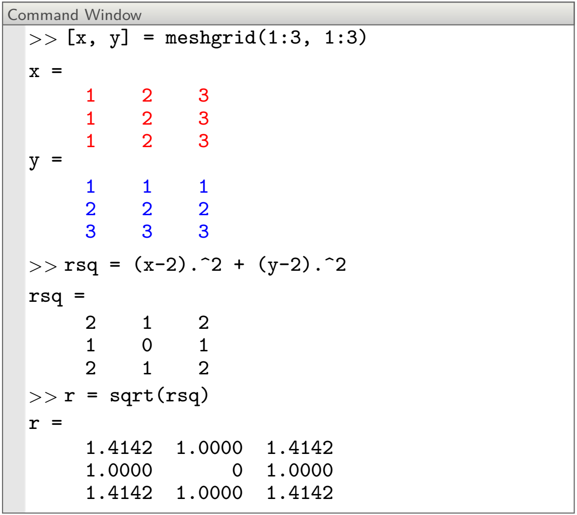
```
>> [x, y] = meshgrid(1:3, 1:3)

x =
        1     2     3
        1     2     3
        1     2     3
y =
        1     1     1
        2     2     2
        3     3     3

>> rsq = (x-2).^2 + (y-2).^2

rsq =
        2     1     2
        1     0     1
        2     1     2

>>
```

(1, 1)	(2, 1)	(3, 1)
(1, 2)	(2, 2)	(3, 2)
(1, 3)	(2, 3)	(3, 3)

2	1	2
1	0	1
2	1	2

$\sqrt{2}$	1	$\sqrt{2}$
1	0	1
$\sqrt{2}$	1	$\sqrt{2}$

3D surface: supplementary function meshgrid



Command Window

```
>> [x, y] = meshgrid(1:3, 1:3)
x =
     1     2     3
     1     2     3
     1     2     3
y =
     1     1     1
     2     2     2
     3     3     3
>> rsq = (x-2).^2 + (y-2).^2
rsq =
     2     1     2
     1     0     1
     2     1     2
>> r = sqrt(rsq)
r =
     1.4142   1.0000   1.4142
     1.0000        0   1.0000
     1.4142   1.0000   1.4142
```

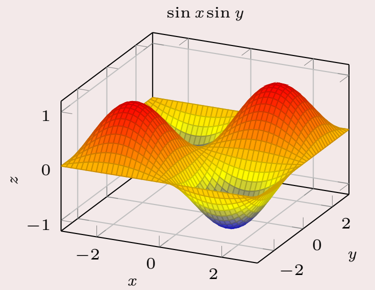
$(1, 1)$	$(2, 1)$	$(3, 1)$
$(1, 2)$	$(2, 2)$	$(3, 2)$
$(1, 3)$	$(2, 3)$	$(3, 3)$

2	1	2
1	0	1
2	1	2

$\sqrt{2}$	1	$\sqrt{2}$
1	0	1
$\sqrt{2}$	1	$\sqrt{2}$

Three-dimensional surface: sin (x) cos (y)

```
[x,y] = meshgrid(-pi:0.1:pi);
z = sin(x).*cos(y);
mesh(x,y,z) % meshc(x,y,z)

surf(x,y,z) % surfc(x,y,z)

xlabel('x')
ylabel('y')
zlabel('z')
title('sin x sin y')
```



M function format

M function format

```
function [output1, ..] = functionname(input1, ..)
% comment of this function
MatLab command 1;
MatLab command 2;
```

Example: Find the rectangular area

```
function area = rectarea(L, W)
%RECTAREA  area of a rectangle
%   rectarea(l, w) calculate the area of a rectangle
%   with a length of L and a width of W

area = L .* W
```

Part II

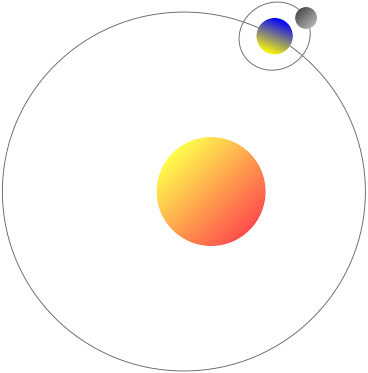
Practice

Outline

4. Multi-planet problem
   - Problem
   - Program
   - Result

5. Monte Carlo
   - Problem

Multi-planet problem



Consider a system of multiple celestial bodies (such as the "Sun, Earth, Moon" three celestial body system), and seek the motion law of each celestial body.

- The distance between celestial bodies is much larger than the size and all celestial bodies are regarded as particles.
- Each celestial body has a fixed mass, and gives the initial position and initial velocity.
- There is only gravitational force between any two celestial bodies.

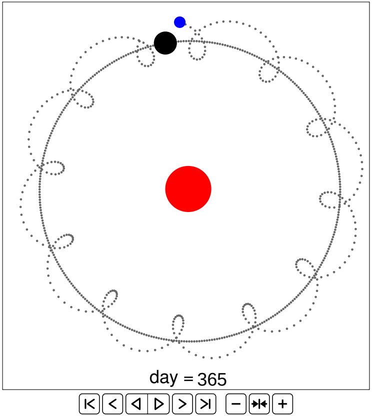
$$\mathbf{F_{ij}} = \frac{Gm_i m_j}{r_{ij}^2}\widehat{\mathbf{r}}_{\mathbf{ij}}$$

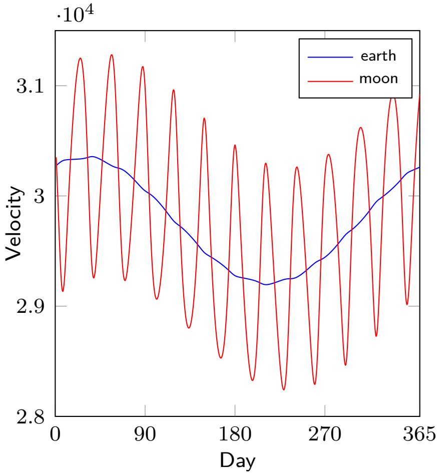
Multi-satellite problem simulation program

main.m

```
G = 6.67e-11;  dt = 24*3600;  N = 3;
M = [sun.mass    ; earth.mass    ; moon.mass    ];% N X 1
R = [sun.position; earth.position; moon.position];% N X 3
V = [sun.velocity; earth.velocity; moon.velocity];% N X 3
for t = 1:365
    F = zeros(N,3);                % F(i,:) = [fx, fy, fz]
    for i = 1 : N
        mi = M(i); ri = R(i,:);    % 第i个天体的质量和位置
        for j = (i+1):N;
            mj = M(j); rj = R(j,:);% 第j个天体的质量和位置
            rij =  rj - ri;
            fij = G*mi*mj./(norm(rij).^3).*rij;% 万有引力
            F([i,j],:) = F([i,j],:) + [fij; -fij];
        end
    end
    V = V + F./repmat(M,1,3)*dt;   % v(t+dt)=v(t)+a(t+dt)dt
    R = R + V*dt;                  % r(t+dt)=r(t)+v(t+dt)dt
end
```

Simulation results of the multi-satellite problem



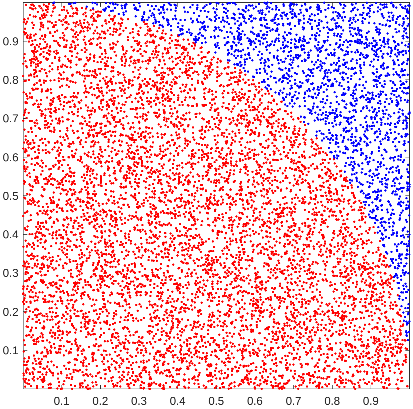


Outline

4. Multi-planet problem
   - Problem
   - Program
   - Result

5. Monte Carlo
   - Problem

Problem: Monte Carlo method for calculating pi



$$\frac{\pi}{4} \approx \frac{n_{\text{red}}}{n_{\text{red}} + n_{\text{blue}}}$$

Thank You!!!