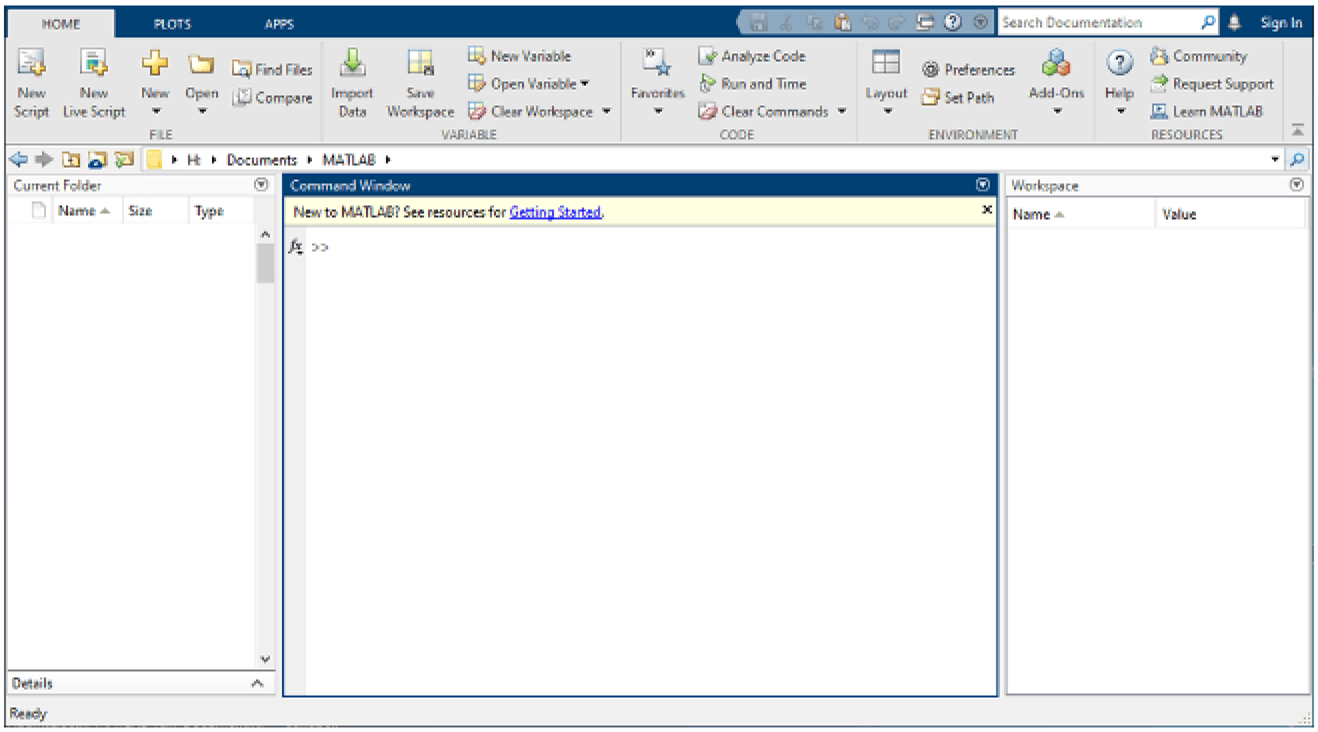

New to MATLAB? See resources for Getting Started.

```
>>
```

Ready

Desktop Basics

When you start MATLAB®, the desktop appears in its default layout.

The desktop includes these panels:

The desktop includes these panels:

Current Folder — Access your files.

Command Window — Enter commands at the command line, indicated by the prompt (>>).

Workspace — Explore data that you create or import from files.

As you work in MATLAB, you issue commands that create variables and call functions. For example, create a variable named a by typing this statement at the command line:

```
A = 1
```

MATLAB adds variable a to the workspace and displays the result in the Command Window.

```
A =


     1
```

Create a few more variables.

```
B = 2

B =


     2

C = a + b

C =


     3

D = cos(a)

D =


   0.5403
```

When you do not specify an output variable, MATLAB uses the variable ans, short for answer, to store the results of your calculation.

```
Sin(a)

Ans =


   0.8415
```

If you end a statement with a semicolon, MATLAB performs the computation, but suppresses the display of output in the Command Window.

```
E = a*b;
```

You can recall previous commands by pressing the up- and down-arrow keys, ↑ and ↓. Press the arrow keys either at an empty command line or after you type the first few characters of a command. For example, to recall the command b = 2, type b, and then press the up-arrow key.

Matrices and Arrays

MATLAB is an abbreviation for “matrix laboratory.” While other programming languages mostly work with numbers one at a time, MATLAB® is designed to operate primarily on whole matrices and arrays.

All MATLAB variables are multidimensional arrays, no matter what type of data. A matrix is a two-dimensional array often used for linear algebra.

Array Creation

To create an array with four elements in a single row, separate the elements with either a comma (,) or a space.

```
A = [1 2 3 4]

A = 1×4


   1    2    3    4
```

This type of array is a row vector.

To create a matrix that has multiple rows, separate the rows with semicolons.

```
A = [1 3 5; 2 4 6; 7 8 10]

A = 3×3


   1    3    5

   2    4    6

   7    8   10
```

Another way to create a matrix is to use a function, such as ones, zeros, or rand. For example, create a 5-by-1 column vector of zeros.

```
Z = zeros(5,1)

Z = 5×1


    0

    0

    0

    0

    0
```

Matrix and Array Operations

MATLAB allows you to process all of the values in a matrix using a single arithmetic operator or function.

```
A + 10

Ans = 3×3


   11    13    15

   12    14    16

   17    18    20


Sin(a)

Ans = 3×3


   0.8415    0.1411   -0.9589

   0.9093   -0.7568   -0.2794

   0.6570    0.9894   -0.5440
```

To transpose a matrix, use a single quote (‘):

```
A’

Ans = 3×3


    1     2     7

    3     4     8

    5     6    10
```

You can perform standard matrix multiplication, which computes the inner products between rows and columns, using the * operator. For example, confirm that a matrix times its inverse returns the identity matrix:

```
P = a*inv(a)

P = 3×3


   1.0000    0.0000   -0.0000

        0    1.0000   -0.0000

        0    0.0000    1.0000
```

Notice that p is not a matrix of integer values. MATLAB stores numbers as floating-point values, and arithmetic operations are sensitive to small differences between the actual value and its floating-point representation. You can display more decimal digits using the format command:

```
Format long

P = a*inv(a)

P = 3×3


  0.999999999999996   0.000000000000007  -0.000000000000002

                  0   1.000000000000000  -0.000000000000003
```

```
        0    0.000000000000014   0.999999999999995
        1
```

Reset the display to the shorter format using

```
Format short
```

Format affects only the display of numbers, not the way MATLAB computes or saves them.

To perform element-wise multiplication rather than matrix multiplication, use the .* operator:

```
P = a.*a

P = 3×3

   1    9   25

   4   16   36

  49   64  100
```

The matrix operators for multiplication, division, and power each have a corresponding array operator that operates element-wise. For example, raise each element of a to the third power:

```
a.^3

ans = 3×3

        1        27        125

        8        64        216

      343        512        1000
```

Concatenation

Concatenation is the process of joining arrays to make larger ones. In fact, you made your first array by concatenating its individual elements. The pair of square brackets [] is the concatenation operator.

```
A = [a,a]

A = 3×6

    1     3     5     1     3     5
    2     4     6     2     4     6
    7     8    10     7     8    10
```

Concatenating arrays next to one another using commas is called horizontal concatenation. Each array must have the same number of rows. Similarly, when the arrays have the same number of columns, you can concatenate vertically using semicolons.

```
A = [a; a]

A = 6×3

    1     3     5
    2     4     6
    7     8    10
    1     3     5
    2     4     6
    7     8    10
```

Complex Numbers

Complex numbers have both real and imaginary parts, where the imaginary unit is the square root of -1.

```
Sqrt(-1)

Ans = 0.0000 + 1.0000i
```

To represent the imaginary part of complex numbers, use either i or j.

```
C = [3+4i, 4+3j; -i, 10j]
```

```
C = 2×2 complex

   3.0000 + 4.0000i   4.0000 + 3.0000i
   0.0000 – 1.0000i   0.0000 +10.0000i
```

Array Indexing

Every variable in MATLAB® is an array that can hold many numbers. When you want to access selected elements of an array, use indexing.

For example, consider the 4-by-4 matrix A:

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

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

There are two ways to refer to a particular element in an array. The most common way is to specify row and column subscripts, such as

```
A(4,2)
Ans = 14
```

Less common, but sometimes useful, is to use a single subscript that traverses down each column in order:

```
A(8)
Ans = 14
```

Using a single subscript to refer to a particular element in an array is called linear indexing.

If you try to refer to elements outside an array on the right side of an assignment statement, MATLAB throws an error.

```
Test = A(4,5)

Index in position 2 exceeds array bounds (must not exceed 4).
```

However, on the left side of an assignment statement, you can specify elements outside the current dimensions. The size of the array increases to accommodate the newcomers.

```
A(4,5) = 17

A = 4×5


    1     2     3     4     0

    5     6     7     8     0

    9    10    11    12     0

   13    14    15    16    17
```

To refer to multiple elements of an array, use the colon operator, which allows you to specify a range of the form start:end. For example, list the elements in the first three rows and the second column of A:

```
A(1:3,2)

Ans = 3×1


    2

    6

   10
```

The colon alone, without start or end values, specifies all of the elements in that dimension. For example, select all the columns in the third row of A:

```
A(3,☺
```

```
Ans = 1×5


    9    10    11    12     0
```

The colon operator also allows you to create an equally spaced vector of values using the more general form start:step:end.

```
B = 0:10:100
```

```
B = 1×11


    0    10    20    30    40    50
```