clear x y;
who;

It is generally good programming style to write only one command per line; however, MATLAB does let you put multiple commands on a line.
x = 5; y = 13; w = 2*x + y; who;

More commonly one wishes to continue a single command across multiple lines due to the length of the syntax. This can be accomplished by using three dots.
z = 2*x + ...
y

Finally, when using clear we can get rid of all of the variables at once with the command "clear all".
clear all;
who; It does not print out anything because there are no variables.

1.2. Basic vector operations

The simplest, but NOT RECOMMENDED, way to declare a variable is by entering the components one-by-one.
x(1) = 1;
x(2) = 4;
x(3) = 6;
x display contents of x

It is generally better to declare a vector all at once, because then MATLAB knows how much memory it needs to allocate from the start. For large vectors, this is much more efficient.
y = [1 4 6] does same job as code above

Note that this declares a row vector. To get a column vector, we can either use the transpose (adjoint for complex x) operator **xT = x';** takes the transpose of the real row vector x or, we can make it a column vector right from the beginning
yT = [1; 4; 6];

To see the difference in the dimensions of a row vs. a column vector, use the command "size" that returns the dimensions of a vector or matrix.
size(xT)
size(y)
size(yT)
The command length works on both row and column vectors.
length(x), length(xT)

Adding or subtracting two vectors is similar to scalars.
z = x + y
w = xT - yT

Multiplying a vector by a scalar is equally straight-forward.
v = 2*x
c = 4;
v2 = c*x

We can also use the . operator to tell MATLAB to perform a given operation on an element-by-element basis. Let us say we want to set each value of y such that y(i) = 2*x(i) + z(i)^2 + 1. We can do this using the code
y = 2.*x + z.^2 + 1

The dot and cross products of two vectors are calculated by
dot(x,y)
z=cross(x,y)

We can define a vector also using the notation [a : d : b]. This produces a vector a, a + d, a + 2*d, a + 3*d, ... until we get to an integer n where a + n*d > b. Look at the two examples.
v = [0 : 0.1: 0.5];
v2 = [0 : 0.1: 0.49];

If we want a vector with N evenly spaced points from a to b, we use the command "linspace(a,b,N)".
v2 = linspace(0,1,5)

Sometimes, we will use a vector later in the program, but want to initialize it at the beginning to zero and by so doing allocate a block of memory to store it. This is done by
v = linspace(0,0,100)'; allocate memory for column vectors of zero

Finally, we can use integer counting variables to access one or more elements of a matrix.
v2 = [0 : 0.01 : 100];
c=v2(49)
w = v2(65:70)

clear all

1.3. Basic matrix operations

We can declare a matrix and give it a value directly.
A = [1 2 3; 4 5 6; 7 8 9]
We can use commas to separate the elements on a line as well.
B = [1,2,3; 4,5,6; 7,8,9]

We can build a matrix from row vectors
row1 = [1 2 3]; row2 = [4 5 6]; row3 = [7 8 9];
C = [row1; row2; row3]

or from column vectors.
column1 = [1; 4; 7];
column2 = [2; 5; 8];
column3 = [3; 6; 9];
D = [column1 column2 column3]

Several matrices can be joined to create a larger one.
M = [A B; C D]

We can extract row or column vectors from a matrix.
row1 = C(1,:)
column2 = D(:,2)

Or, we make a vector or another matrix by extracting a subset of the elements.
v = M(1:4,1)
w = M(2,2:4)
C = M(1:4,2:5)

The transpose of a real matrix is obtained using the ' operator
D = A'
C, C'

For a complex matrix, ' returns the adjoint (transpose and conjugate. The conjugation operation is removed by using the "transpose only" command .'
E = D;
E(1,2) = E(1,2) + 3*i;
E(2,1) = E(2,1) - 2*i;
E', E.'

The "who" command lists the matrices in addition to scalar and vector variables.
who

If in addition we want to see the dimensions of each variable, we use the "whos" command. This tells use the size of each variable and the amount of memory storage that each requires.
whos

The command "size" tells us the size of a matrix.
M = [1 2 3 4; 5 6 7 8; 9 10 11 12];
size(M)
num_rows = size(M,1)
num_columns = size(M,2)

Adding, subtracting, and multiplying matrices is straight-forward.
D = A + B
D = A - B
D = A*B

We can declare matrices in a number of ways.

We can create a matrix with m rows and n columns, all containing zeros by
m=3; n=4;
C = zeros(m,n)

If we want to make an N by N square matrix, we only need to use one index.
C = zeros(n)

We create an Identity matrix, where all elements are zero except for those on the principle diagonal, which are one.
D = eye(5)

Finally, we can use the . operator to perform element-by-element operations just as we did for vectors. The following command creates a matrix C, such that C(i,j) = 2*A(i,j) + (B(i,j))^2.
C = 2.*A + B.^2

Matrices are cleared from memory along with all other variables.
clear A B
whos
clear all
who