

# MATLAB - VARIABLES

[http://www.tutorialspoint.com/matlab/matlab\\_variables.htm](http://www.tutorialspoint.com/matlab/matlab_variables.htm)

Copyright © tutorialspoint.com

In MATLAB environment, every variable is an array or matrix.

You can assign variables in a simple way. For example,

```
x = 3 % defining x and initializing it with a value
```

MATLAB will execute the above statement and return the following result –

```
x = 3
```

It creates a 1-by-1 matrix named x and stores the value 3 in its element. Let us check another example,

```
x = sqrt(16) % defining x and initializing it with an expression
```

MATLAB will execute the above statement and return the following result –

```
x = 4
```

Please note that –

- Once a variable is entered into the system, you can refer to it later.
- Variables must have values before they are used.
- When an expression returns a result that is not assigned to any variable, the system assigns it to a variable named ans, which can be used later.

For example,

```
sqrt(78)
```

MATLAB will execute the above statement and return the following result –

```
ans = 8.8318
```

You can use this variable **ans** –

```
sqrt(78);  
9876/ans
```

MATLAB will execute the above statement and return the following result –

```
ans = 1118.2
```

Let's look at another example –

```
x = 7 * 8;  
y = x * 7.89
```

MATLAB will execute the above statement and return the following result –

```
y = 441.84
```

## Multiple Assignments

You can have multiple assignments on the same line. For example,

```
a = 2; b = 7; c = a * b
```

MATLAB will execute the above statement and return the following result –

```
c = 14
```

## I have forgotten the Variables!

The **who** command displays all the variable names you have used.

```
who
```

MATLAB will execute the above statement and return the following result –

```
Your variables are:  
a    ans  b    c
```

The **whos** command displays little more about the variables –

- Variables currently in memory
- Type of each variables
- Memory allocated to each variable
- Whether they are complex variables or not

```
whos
```

MATLAB will execute the above statement and return the following result –

Attr	Name	Size	Bytes	Class
====	====	====	====	====
	a	1x1	8	double
	ans	1x70	757	cell
	b	1x1	8	double
	c	1x1	8	double

```
Total is 73 elements using 781 bytes
```

The **clear** command deletes all *orthespecified* variables from the memory.

```
clear x    % it will delete x, won't display anything  
clear     % it will delete all variables in the workspace  
          % peacefully and unobtrusively
```

## Long Assignments

Long assignments can be extended to another line by using an ellipses `...`. For example,

```
initial_velocity = 0;  
acceleration = 9.8;  
time = 20;  
final_velocity = initial_velocity + acceleration * time
```

MATLAB will execute the above statement and return the following result –

```
final_velocity = 196
```

## The format Command

By default, MATLAB displays numbers with four decimal place values. This is known as **short format**.

However, if you want more precision, you need to use the **format** command.

The **format long** command displays 16 digits after decimal.

For example –

```
format long  
x = 7 + 10/3 + 5 ^ 1.2
```

MATLAB will execute the above statement and return the following result–

```
x = 17.2319816406394
```

Another example,

```
format short  
x = 7 + 10/3 + 5 ^ 1.2
```

MATLAB will execute the above statement and return the following result –

```
x = 17.232
```

The **format bank** command rounds numbers to two decimal places. For example,

```
format bank  
daily_wage = 177.45;  
weekly_wage = daily_wage * 6
```

MATLAB will execute the above statement and return the following result –

```
weekly_wage = 1064.70
```

MATLAB displays large numbers using exponential notation.

The **format short e** command allows displaying in exponential form with four decimal places plus the exponent.

For example,

```
format short e  
4.678 * 4.9
```

MATLAB will execute the above statement and return the following result –

```
ans = 2.2922e+01
```

The **format long e** command allows displaying in exponential form with four decimal places plus the exponent. For example,

```
format long e  
x = pi
```

MATLAB will execute the above statement and return the following result –

```
x = 3.141592653589793e+00
```

The **format rat** command gives the closest rational expression resulting from a calculation. For example,

```
format rat  
4.678 * 4.9
```

MATLAB will execute the above statement and return the following result –

```
ans = 34177/1491
```

## Creating Vectors

A vector is a one-dimensional array of numbers. MATLAB allows creating two types of vectors:

- Row vectors
- Column vectors

**Row vectors** are created by enclosing the set of elements in square brackets, using space or comma to delimit the elements.

For example,

```
r = [7 8 9 10 11]
```

MATLAB will execute the above statement and return the following result –

```
r =  
    7     8     9    10    11
```

Another example,

```
r = [7 8 9 10 11];  
t = [2, 3, 4, 5, 6];  
res = r + t
```

MATLAB will execute the above statement and return the following result –

```
res =  
     9     11     13     15     17
```

**Column vectors** are created by enclosing the set of elements in square brackets, using semicolon ; to delimit the elements.

```
c = [7; 8; 9; 10; 11]
```

MATLAB will execute the above statement and return the following result –

```
c =  
     7  
     8  
     9  
    10  
    11
```

## Creating Matrices

A matrix is a two-dimensional array of numbers.

In MATLAB, a matrix is created by entering each row as a sequence of space or comma separated elements, and end of a row is demarcated by a semicolon. For example, let us create a 3-by-3 matrix as:

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

MATLAB will execute the above statement and return the following result –

```
m =  
    1         2         3  
    4         5         6  
    7         8         9
```

Loading [Mathjax]/jax/output/HTML-CSS/jax.js