

# MATLAB

numerical computing

**tutorialspoint**

SIMPLY EASY LEARNING

[www.tutorialspoint.com](http://www.tutorialspoint.com)



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

## About the Tutorial

---

MATLAB is a programming language developed by MathWorks. It started out as a matrix programming language where linear algebra programming was simple. It can be run both under interactive sessions and as a batch job.

This tutorial gives you aggressively a gentle introduction of MATLAB programming language. It is designed to give students fluency in MATLAB programming language. Problem-based MATLAB examples have been given in simple and easy way to make your learning fast and effective.

## Audience

---

This tutorial has been prepared for the beginners to help them understand basic to advanced functionality of MATLAB. After completing this tutorial you will find yourself at a moderate level of expertise in using MATLAB from where you can take yourself to next levels.

## Prerequisites

---

We assume you have a little knowledge of any computer programming and understand concepts like variables, constants, expressions, statements, etc. If you have done programming in any other high-level language like C, C++ or Java, then it will be very much beneficial and learning MATLAB will be like a fun for you.

## Copyright & Disclaimer Notice

---

© Copyright 2014 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at [contact@tutorialspoint.com](mailto:contact@tutorialspoint.com)

# Table of Contents

---

About the Tutorial.....	i
Audience.....	i
Prerequisites.....	i
Copyright & Disclaimer Notice.....	i
Table of Contents.....	ii
<b>1. OVERVIEW.....</b>	<b>1</b>
<b>MATLAB's Power of Computational Mathematics.....</b>	<b>1</b>
<b>Features of MATLAB.....</b>	<b>1</b>
<b>Uses of MATLAB.....</b>	<b>2</b>
<b>2. ENVIRONMENT.....</b>	<b>3</b>
<b>Local Environment Setup.....</b>	<b>3</b>
<b>Understanding the MATLAB Environment.....</b>	<b>4</b>
<b>3. BASIC SYNTAX.....</b>	<b>7</b>
<b>Hands on Practice.....</b>	<b>7</b>
<b>Use of Semicolon (;) in MATLAB.....</b>	<b>8</b>
<b>Adding Comments.....</b>	<b>8</b>
<b>Commonly used Operators and Special Characters.....</b>	<b>9</b>
<b>Special Variables and Constants.....</b>	<b>10</b>
<b>Naming Variables.....</b>	<b>11</b>
<b>Saving Your Work.....</b>	<b>11</b>
<b>4. VARIABLES.....</b>	<b>12</b>
<b>Multiple Assignments.....</b>	<b>13</b>
<b>I have forgotten the Variables!.....</b>	<b>13</b>
<b>Long Assignments.....</b>	<b>14</b>

	The format Command .....	15
	Creating Vectors.....	17
	Creating Matrices .....	18
5.	COMMANDS.....	20
	Commands for Managing a Session .....	20
	Commands for Working with the System .....	20
	Input and Output Commands .....	22
	Vector, Matrix, and Array Commands .....	23
	Plotting Commands.....	25
6.	M-FILES .....	27
	The M Files.....	27
	Creating and Running Script File.....	27
7.	DATA TYPES.....	30
	Data Types Available in MATLAB.....	30
	Data Type Conversion .....	32
	Determination of Data Types .....	34
8.	OPERATORS.....	39
	Arithmetic Operators .....	39
	Functions for Arithmetic Operations .....	42
	Relational Operators .....	46
	Logical Operators .....	49
	Functions for Logical Operations .....	50
	Bitwise Operations.....	55
	Set Operations .....	57
9.	DECISION MAKING.....	60

if... end Statement .....	61
if...else...end Statement .....	63
if...elseif...elseif...else...end Statements.....	64
The Nested if Statements .....	66
The switch Statement .....	67
The Nested Switch Statements.....	69
<b>10. LOOP TYPES.....</b>	<b>71</b>
The while Loop.....	72
The for Loop.....	73
The Nested Loops.....	76
Loop Control Statements.....	78
The break Statement.....	79
The continue Statement.....	80
<b>11. VECTORS .....</b>	<b>83</b>
Row Vectors.....	83
Column Vectors.....	83
Referencing the Elements of a Vector .....	84
Vector Operations.....	85
Addition and Subtraction of Vectors .....	85
Scalar Multiplication of Vectors .....	86
Transpose of a Vector .....	86
Appending Vectors.....	87
Magnitude of a Vector .....	89
Vector Dot Product .....	90
Vectors with Uniformly Spaced Elements.....	90

12. MATRIX.....	92
Referencing the Elements of a Matrix .....	92
Deleting a Row or a Column in a Matrix .....	94
Matrix Operations.....	96
Addition and Subtraction of Matrices .....	96
Division (Left, Right) of Matrix .....	97
Scalar Operations of Matrices .....	98
Transpose of a Matrix .....	99
Concatenating Matrices .....	99
Matrix Multiplication .....	101
Determinant of a Matrix .....	102
Inverse of a Matrix .....	102
13. ARRAYS.....	104
Special Arrays in MATLAB .....	104
A Magic Square .....	106
Multidimensional Arrays.....	106
Array Functions.....	109
Sorting Arrays .....	112
Cell Array .....	113
Accessing Data in Cell Arrays.....	114
14. COLON NOTATION.....	116
15. NUMBERS.....	119
Conversion to Various Numeric Data Types.....	119
Smallest and Largest Integers.....	121
Smallest and Largest Floating Point Numbers .....	123

16. STRINGS.....	125
Rectangular Character Array .....	126
Combining Strings into a Cell Array .....	128
String Functions in MATLAB .....	129
17. FUNCTIONS .....	134
Anonymous Functions .....	135
Nested Functions .....	138
Private Functions .....	139
Global Variables .....	140
18. DATA IMPORT.....	142
Low-Level File I/O.....	146
Import Text Data Files with Low-Level I/O .....	147
19. DATA OUTPUT .....	152
Writing to Diary Files.....	154
Exporting Data to Text Data Files with Low-Level I/O.....	154
20. PLOTTING .....	156
Adding Title, Labels, Grid Lines, and Scaling on the Graph.....	158
Drawing Multiple Functions on the Same Graph .....	159
Setting Colors on Graph .....	160
Setting Axis Scales .....	161
Generating Sub-Plots .....	162
21. GRAPHICS.....	164
Drawing Bar Charts .....	164
Drawing Contours .....	165
Three-Dimensional Plots .....	167

22. ALGEBRA .....	169
Solving Basic Algebraic Equations in MATLAB .....	169
Solving Quadratic Equations in MATLAB .....	171
Expanding and Collecting Equations in MATLAB .....	176
Expanding and Collecting Equations in Octave .....	177
Factorization and Simplification of Algebraic Expressions .....	179
23. CALCULUS.....	181
Calculating Limits .....	181
Verification of Basic Properties of Limits using Octave .....	184
Left and Right Sided Limits .....	185
24. DIFFERENTIAL .....	188
Verification of Elementary Rules of Differentiation .....	189
Derivatives of Exponential, Logarithmic, and Trigonometric Functions .....	193
Computing Higher Order Derivatives.....	198
Finding the Maxima and Minima of a Curve.....	200
Solving Differential Equations .....	204
25. INTEGRATION .....	206
Finding Indefinite Integral Using MATLAB .....	206
Finding Definite Integral Using MATLAB.....	210
26. POLYNOMIALS.....	216
Evaluating Polynomials .....	216
Polynomial Curve Fitting .....	217
27. TRANSFORMS.....	219
The Laplace Transform .....	219
The Inverse Laplace Transform .....	220



<b>The Fourier Transforms .....</b>	<b>222</b>
<b>Inverse Fourier Transforms .....</b>	<b>224</b>
<b>28. GNU OCTAVE TUTORIAL .....</b>	<b>225</b>
<b>MATLAB vs Octave .....</b>	<b>225</b>
<b>29. SIMULINK .....</b>	<b>229</b>
<b>Using Simulink.....</b>	<b>230</b>

# 1. OVERVIEW

MATLAB (matrix laboratory) is a fourth-generation high-level programming language and interactive environment for numerical computation, visualization and programming.

MATLAB is developed by MathWorks.

It allows matrix manipulations; plotting of functions and data; implementation of algorithms; creation of user interfaces; interfacing with programs written in other languages, including C, C++, Java, and FORTRAN; analyze data; develop algorithms; and create models and applications.

It has numerous built-in commands and math functions that help you in mathematical calculations, generating plots, and performing numerical methods.

## **MATLAB's Power of Computational Mathematics**

---

MATLAB is used in every facet of computational mathematics. Following are some commonly used mathematical calculations where it is used most commonly:

- Dealing with Matrices and Arrays
- 2-D and 3-D Plotting and graphics
- Linear Algebra
- Algebraic Equations
- Non-linear Functions
- Statistics
- Data Analysis
- Calculus and Differential Equations
- Numerical Calculations
- Integration
- Transforms
- Curve Fitting
- Various other special functions

## Features of MATLAB

---

Following are the basic features of MATLAB:

- It is a high-level language for numerical computation, visualization and application development.
- It also provides an interactive environment for iterative exploration, design and problem solving.
- It provides vast library of mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration and solving ordinary differential equations.
- It provides built-in graphics for visualizing data and tools for creating custom plots.
- MATLAB's programming interface gives development tools for improving code quality, maintainability, and maximizing performance.
- It provides tools for building applications with custom graphical interfaces.
- It provides functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET and Microsoft Excel.

## Uses of MATLAB

---

MATLAB is widely used as a computational tool in science and engineering encompassing the fields of physics, chemistry, math and all engineering streams. It is used in a range of applications including:

- signal processing and Communications
- image and video Processing
- control systems
- test and measurement
- computational finance
- computational biology

# 2. ENVIRONMENT

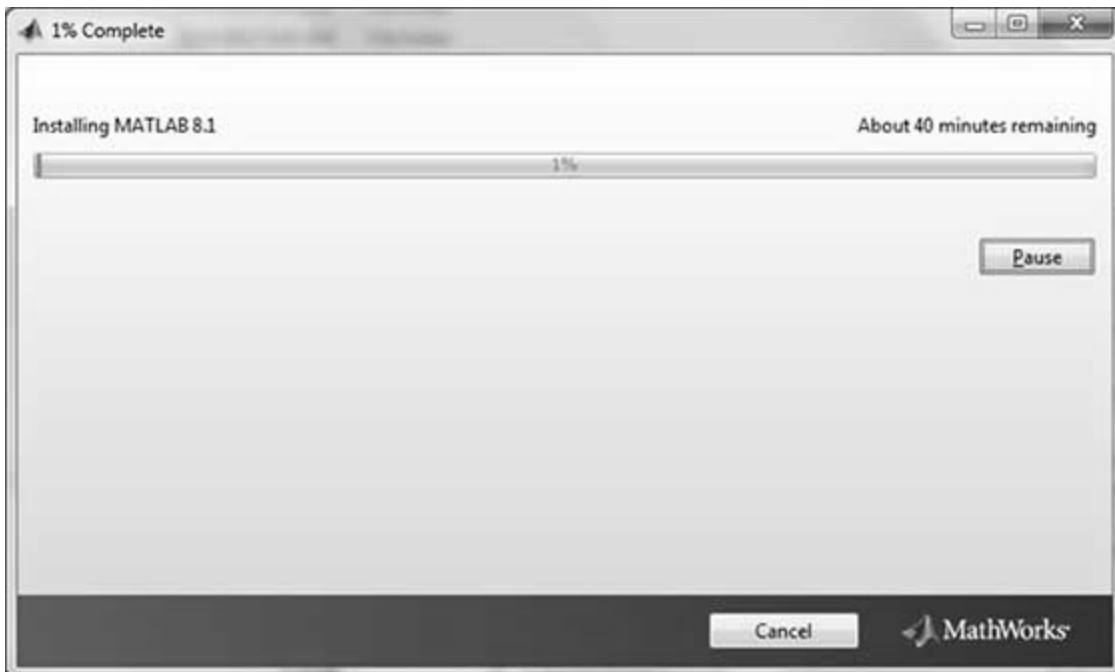
## Local Environment Setup

Setting up MATLAB environment is a matter of few clicks. The installer can be downloaded from [http://in.mathworks.com/downloads/web\\_downloads](http://in.mathworks.com/downloads/web_downloads):

MathWorks provides the licensed product, a trial version and a student version as well. You need to log into the site and wait a little for their approval.

After downloading the installer the software can be installed through few clicks.





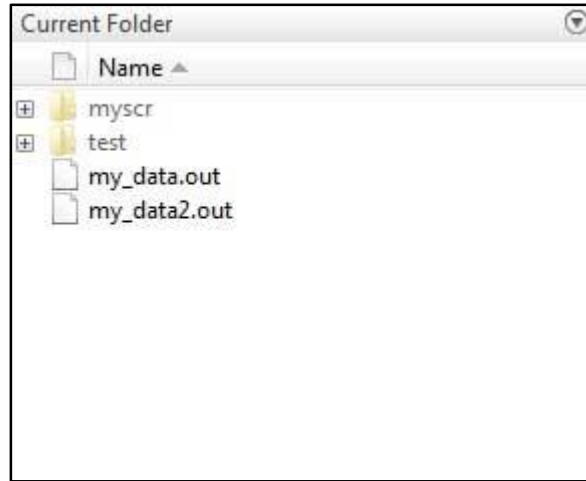
## Understanding the MATLAB Environment

MATLAB development IDE can be launched from the icon created on the desktop. The main working window in MATLAB is called the desktop. When MATLAB is started, the desktop appears in its default layout:

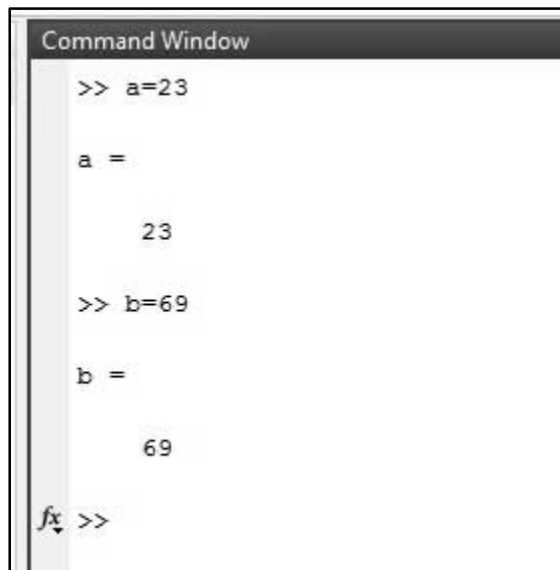


The desktop has the following panels:

**Current Folder** - This panel allows you to access the project folders and files.



**Command Window** - This is the main area where commands can be entered at the command line. It is indicated by the command prompt (>>).



**Workspace** - The workspace shows all the variables created and/or imported from files.

Workspace	
Name ▲	Value
a	23
b	69

**Command History** - This panel shows or rerun commands that are entered at the command line.

```

%-- 7/14/2013 5:58 PM --%
%-- 7/15/2013 9:01 AM --%
| simulink
%-- 7/15/2013 6:09 PM --%
| simulink
%-- 7/25/2013 7:57 AM --%
%-- 7/25/2013 7:58 AM --%
| chdir test
| prog4
%-- 7/29/2013 8:55 AM --%
| a=23
| b=69

```

## Set up GNU Octave

If you are willing to use Octave on your machine (Linux, BSD, OS X or Windows), then kindly download latest version from <http://www.gnu.org/software/octave/download.html>. You can check the given installation instructions for your machine

# 3. BASIC SYNTAX

MATLAB environment behaves like a super-complex calculator. You can enter commands at the >> command prompt.

MATLAB is an interpreted environment. In other words, you give a command and MATLAB executes it right away.

## Hands on Practice

---

Type a valid expression, for example,

```
5 + 5
```

And press ENTER

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
ans = 10
```

Let us take up few more examples:

```
3 ^ 2      % 3 raised to the power of 2
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
ans = 9
```

Another example,

```
sin(pi / 2)  % sine of angle 90o
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:



```
ans = 1
```

Another example,

```
7/0           % Divide by zero
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
ans = Inf
warning: division by zero
```

Another example,

```
732 * 20.3
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
ans = 1.4860e+04
```

MATLAB provides some special expressions for some mathematical symbols, like pi for  $\pi$ , Inf for  $\infty$ , i (and j) for  $\sqrt{-1}$  etc. **Nan** stands for 'not a number'.

## Use of Semicolon (;) in MATLAB

Semicolon (;) indicates end of statement. However, if you want to suppress and hide the MATLAB output for an expression, add a semicolon after the expression.

For example,

```
x = 3;
y = x + 5
```

When you click the Execute button, or type Ctrl+E, MATLAB executes it immediately and the result returned is:

```
y = 8
```

## Adding Comments

The percent symbol (%) is used for indicating a comment line. For example,

```
x = 9    % assign the value 9 to x
```

You can also write a block of comments using the block comment operators % { and % }.

The MATLAB editor includes tools and context menu items to help you add, remove, or change the format of comments.

## Commonly used Operators and Special Characters

MATLAB supports the following commonly used operators and special characters:

Operator	Purpose
+	Plus; addition operator.
-	Minus; subtraction operator.
*	Scalar and matrix multiplication operator.
.*	Array multiplication operator.
^	Scalar and matrix exponentiation operator.
.^	Array exponentiation operator.
\	Left-division operator.
/	Right-division operator.

.\	Array left-division operator.
./	Array right-division operator.
:	Colon; generates regularly spaced elements and represents an entire row or column.
( )	Parentheses; encloses function arguments and array indices; overrides precedence.
[ ]	Brackets; enclosures array elements.
.	Decimal point.
...	Ellipsis; line-continuation operator
,	Comma; separates statements and elements in a row
;	Semicolon; separates columns and suppresses display.
%	Percent sign; designates a comment and specifies formatting.
—	Quote sign and transpose operator.
.—	Non-conjugated transpose operator.
=	Assignment operator.

## Special Variables and Constants

---

MATLAB supports the following special variables and constants:

Name	Meaning
<b>ans</b>	Most recent answer.
<b>eps</b>	Accuracy of floating-point precision.
<b>i,j</b>	The imaginary unit $\sqrt{-1}$ .
<b>Inf</b>	Infinity.
<b>NaN</b>	Undefined numerical result (not a number).
<b>pi</b>	The number $\pi$

## Naming Variables

---

Variable names consist of a letter followed by any number of letters, digits or underscore.

MATLAB is **case-sensitive**.

Variable names can be of any length, however, MATLAB uses only first N characters, where N is given by the function **namelengthmax**.

## Saving Your Work

---

The **save** command is used for saving all the variables in the workspace, as a file with .mat extension, in the current directory.

For example,

```
save myfile
```

You can reload the file anytime later using the **load** command.

```
load myfile
```

# 4. VARIABLES

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***:

```
9876/ans
```

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

```
ans =  
1.1182e+03
```

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 =  
441.8400
```

## 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    x    y
```

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:

Name	Size	Bytes	Class	Attributes
a	1x1	8	double	
ans	1x1	8	double	
b	1x1	8	double	
c	1x1	8	double	
x	1x1	8	double	
y	1x1	8	double	

The **clear** command deletes all (or the specified) variable(s) 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 ellipsis (...). 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 =
    17.231981640639408
```



Another example,

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

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

```
x =
    17.2320
```

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 =  
  
2063/90
```

## 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 =
Columns 1 through 4
    7     8     9    10
Column 5
    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 =
Columns 1 through 4
    9    11    13    15
Column 5
    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
```

# 5. COMMANDS

MATLAB is an interactive program for numerical computation and data visualization. You can enter a command by typing it at the MATLAB prompt '>>' on the **Command Window**.

In this section, we will provide lists of commonly used general MATLAB commands.

## Commands for Managing a Session

---

MATLAB provides various commands for managing a session. The following table provides all such commands:

Command	Purpose
<b>clc</b>	Clears command window.
<b>clear</b>	Removes variables from memory.
<b>exist</b>	Checks for existence of file or variable.
<b>global</b>	Declares variables to be global.
<b>help</b>	Searches for a help topic.
<b>lookfor</b>	Searches help entries for a keyword.
<b>quit</b>	Stops MATLAB.
<b>who</b>	Lists current variables.
<b>whos</b>	Lists current variables (long display).

## Commands for Working with the System

---

MATLAB provides various useful commands for working with the system, like saving the current work in the workspace as a file and loading the file later.

It also provides various commands for other system-related activities like, displaying date, listing files in the directory, displaying current directory, etc.

The following table displays some commonly used system-related commands:

Command	Purpose
<b>cd</b>	Changes current directory.
<b>date</b>	Displays current date.
<b>delete</b>	Deletes a file.
<b>diary</b>	Switches on/off diary file recording.
<b>dir</b>	Lists all files in current directory.
<b>load</b>	Loads workspace variables from a file.
<b>path</b>	Displays search path.
<b>pwd</b>	Displays current directory.
<b>save</b>	Saves workspace variables in a file.
<b>type</b>	Displays contents of a file.
<b>what</b>	Lists all MATLAB files in the current directory.
<b>wkload</b>	Reads .wk1 spreadsheet file.

## Input and Output Commands

---

MATLAB provides the following input and output related commands:

Command	Purpose
<b>disp</b>	Displays contents of an array or string.
<b>fscanf</b>	Read formatted data from a file.
<b>format</b>	Controls screen-display format.
<b>fprintf</b>	Performs formatted writes to screen or file.
<b>input</b>	Displays prompts and waits for input.
<b>;</b>	Suppresses screen printing.

The **fscanf** and **fprintf** commands behave like C scanf and printf functions. They support the following format codes:

Format Code	Purpose
<b>%s</b>	Format as a string.
<b>%d</b>	Format as an integer.
<b>%f</b>	Format as a floating point value.

<b>%e</b>	Format as a floating point value in scientific notation.
<b>%g</b>	Format in the most compact form: %f or %e.
<b>\n</b>	Insert a new line in the output string.
<b>\t</b>	Insert a tab in the output string.

The format function has the following forms used for numeric display:

<b>Format Function</b>	<b>Display up to</b>
<b>format short</b>	Four decimal digits (default).
<b>format long</b>	16 decimal digits.
<b>format short e</b>	Five digits plus exponent.
<b>format long e</b>	16 digits plus exponents.
<b>format bank</b>	Two decimal digits.
<b>format +</b>	Positive, negative, or zero.
<b>format rat</b>	Rational approximation.
<b>format compact</b>	Suppresses some line feeds.
<b>format loose</b>	Resets to less compact display mode.



## Vector, Matrix, and Array Commands

---

The following table shows various commands used for working with arrays, matrices and vectors:

Command	Purpose
<b>cat</b>	Concatenates arrays.
<b>find</b>	Finds indices of nonzero elements.
<b>length</b>	Computes number of elements.
<b>linspace</b>	Creates regularly spaced vector.
<b>logspace</b>	Creates logarithmically spaced vector.
<b>max</b>	Returns largest element.
<b>min</b>	Returns smallest element.
<b>prod</b>	Product of each column.
<b>reshape</b>	Changes size.
<b>size</b>	Computes array size.
<b>sort</b>	Sorts each column.
<b>sum</b>	Sums each column.
<b>eye</b>	Creates an identity matrix.
<b>ones</b>	Creates an array of ones.

<b>zeros</b>	Creates an array of zeros.
<b>cross</b>	Computes matrix cross products.
<b>dot</b>	Computes matrix dot products.
<b>det</b>	Computes determinant of an array.
<b>inv</b>	Computes inverse of a matrix.
<b>pinv</b>	Computes pseudoinverse of a matrix.
<b>rank</b>	Computes rank of a matrix.
<b>rref</b>	Computes reduced row echelon form.
<b>cell</b>	Creates cell array.
<b>celldisp</b>	Displays cell array.
<b>cellplot</b>	Displays graphical representation of cell array.
<b>num2cell</b>	Converts numeric array to cell array.
<b>deal</b>	Matches input and output lists.
<b>iscell</b>	Identifies cell array.

## Plotting Commands

---

MATLAB provides numerous commands for plotting graphs. The following table shows some of the commonly used commands for plotting:

<b>Command</b>	<b>Purpose</b>
<b>axis</b>	Sets axis limits.
<b>fplot</b>	Intelligent plotting of functions.
<b>grid</b>	Displays gridlines.
<b>plot</b>	Generates xy plot.
<b>print</b>	Prints plot or saves plot to a file.
<b>title</b>	Puts text at top of plot.
<b>xlabel</b>	Adds text label to x-axis.
<b>ylabel</b>	Adds text label to y-axis.
<b>axes</b>	Creates axes objects.
<b>close</b>	Closes the current plot.
<b>close all</b>	Closes all plots.
<b>figure</b>	Opens a new figure window.
<b>gtext</b>	Enables label placement by mouse.
<b>hold</b>	Freezes current plot.
<b>legend</b>	Legend placement by mouse.
<b>refresh</b>	Redraws current figure window.

<b>set</b>	Specifies properties of objects such as axes.
<b>subplot</b>	Creates plots in sub windows.
<b>text</b>	Places string in figure.
<b>bar</b>	Creates bar chart.
<b>loglog</b>	Creates log-log plot.
<b>polar</b>	Creates polar plot.
<b>semilogx</b>	Creates semi log plot. (logarithmic abscissa).
<b>semilogy</b>	Creates semi log plot. (logarithmic ordinate).
<b>stairs</b>	Creates stairs plot.
<b>stem</b>	Creates stem plot.

# 6. M-FILES

So far, we have used MATLAB environment as a calculator. However, MATLAB is also a powerful programming language, as well as an interactive computational environment.

In previous chapters, you have learned how to enter commands from the MATLAB command prompt. MATLAB also allows you to write series of commands into a file and execute the file as complete unit, like writing a function and calling it.

## The M Files

---

MATLAB allows writing two kinds of program files:

**Scripts** - script files are program files with **.m extension**. In these files, you write series of commands, which you want to execute together. Scripts do not accept inputs and do not return any outputs. They operate on data in the workspace.

**Functions** - functions files are also program files with **.m extension**. Functions can accept inputs and return outputs. Internal variables are local to the function.

You can use the MATLAB editor or any other text editor to create your **.m** files. In this section, we will discuss the script files. A script file contains multiple sequential lines of MATLAB commands and function calls. You can run a script by typing its name at the command line.

## Creating and Running Script File

---

To create scripts files, you need to use a text editor. You can open the MATLAB editor in two ways:

Using the command prompt

Using the IDE

If you are using the command prompt, type **edit** in the command prompt. This will open the editor. You can directly type **edit** and then the filename (with **.m** extension)

```
edit
```

```
Or
```

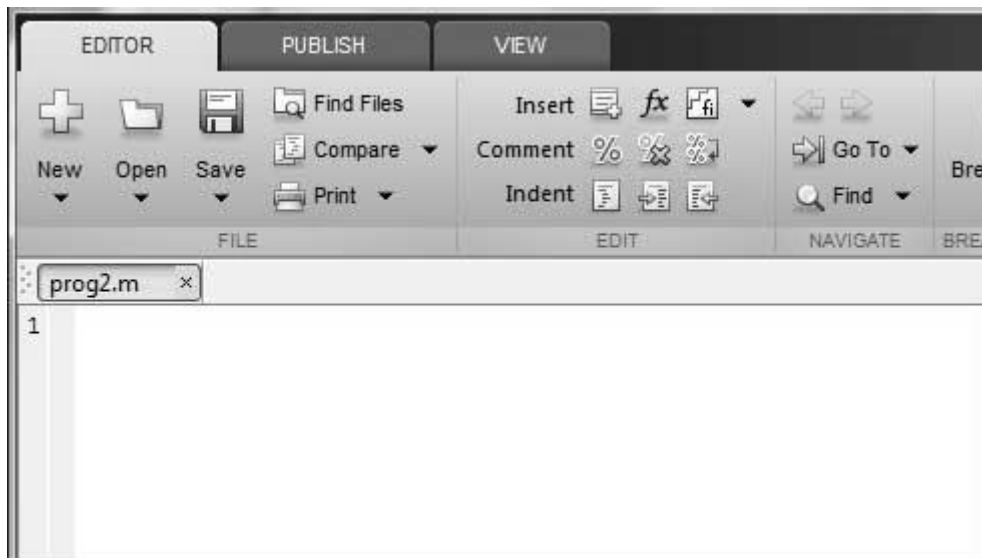
```
edit <filename>
```

The above command will create the file in default MATLAB directory. If you want to store all program files in a specific folder, then you will have to provide the entire path.

Let us create a folder named progs. Type the following commands at the command prompt(>>):

```
mkdir progs    % create directory progs under default directory
chdir progs    % changing the current directory to progs
edit prog1.m   % creating an m file named prog1.m
```

If you are creating the file for first time, MATLAB prompts you to confirm it. Click Yes.



Alternatively, if you are using the IDE, choose NEW -> Script. This also opens the editor and creates a file named Untitled. You can name and save the file after typing the code.

Type the following code in the editor:

```
NoOfStudents = 6000;
TeachingStaff = 150;
NonTeachingStaff = 20;
```

```
Total = NoOfStudents + TeachingStaff ...  
    + NonTeachingStaff;  
disp(Total);
```

After creating and saving the file, you can run it in two ways:

Clicking the **Run** button on the editor window or

Just typing the filename (without extension) in the command prompt: >> prog1

The command window prompt displays the result:

```
6170
```

### Example

Create a script file, and type the following code:

```
a = 5; b = 7;  
c = a + b  
d = c + sin(b)  
e = 5 * d  
f = exp(-d)
```

When the above code is compiled and executed, it produces the following result:

```
c =  
    12  
d =  
    12.6570  
e =  
    63.2849  
f =
```

3.1852e-06



# 7. DATA TYPES

MATLAB does not require any type declaration or dimension statements. Whenever MATLAB encounters a new variable name, it creates the variable and allocates appropriate memory space.

If the variable already exists, then MATLAB replaces the original content with new content and allocates new storage space, where necessary.

For example,

```
Total = 42
```

The above statement creates a 1-by-1 matrix named 'Total' and stores the value 42 in it.

## Data Types Available in MATLAB

MATLAB provides 15 fundamental data types. Every data type stores data that is in the form of a matrix or array. The size of this matrix or array is a minimum of 0-by-0 and this can grow up to a matrix or array of any size.

The following table shows the most commonly used data types in MATLAB:

Data Type	Description
<b>int8</b>	8-bit signed integer
<b>uint8</b>	8-bit unsigned integer
<b>int16</b>	16-bit signed integer
<b>uint16</b>	16-bit unsigned integer
<b>int32</b>	32-bit signed integer

<b>uint32</b>	32-bit unsigned integer
<b>int64</b>	64-bit signed integer
<b>uint64</b>	64-bit unsigned integer
<b>single</b>	single precision numerical data
<b>double</b>	double precision numerical data
<b>logical</b>	logical values of 1 or 0, represent true and false respectively
<b>char</b>	character data (strings are stored as vector of characters)
<b>cell array</b>	array of indexed cells, each capable of storing an array of a different dimension and data type
<b>structure</b>	C-like structures, each structure having named fields capable of storing an array of a different dimension and data type
<b>function handle</b>	pointer to a function
<b>user classes</b>	objects constructed from a user-defined class
<b>java classes</b>	objects constructed from a Java class

### Example

Create a script file with the following code:

```
str = 'Hello World!'
n = 2345
d = double(n)
```

```
un = uint32(789.50)
rn = 5678.92347
c = int32(rn)
```

When the above code is compiled and executed, it produces the following result:

```
str =
Hello World!

n =
    2345

d =
    2345

un =
    790

rn =
    5.6789e+03

c =
    5679
```

## Data Type Conversion

---

MATLAB provides various functions for converting a value from one data type to another. The following table shows the data type conversion functions:

Function	Purpose
----------	---------

<b>Char</b>	Convert to character array (string)
<b>int2str</b>	Convert integer data to string
<b>mat2str</b>	Convert matrix to string
<b>num2str</b>	Convert number to string
<b>str2double</b>	Convert string to double-precision value
<b>str2num</b>	Convert string to number
<b>native2unicode</b>	Convert numeric bytes to Unicode characters
<b>unicode2native</b>	Convert Unicode characters to numeric bytes
<b>base2dec</b>	Convert base N number string to decimal number
<b>bin2dec</b>	Convert binary number string to decimal number
<b>dec2base</b>	Convert decimal to base N number in string
<b>dec2bin</b>	Convert decimal to binary number in string
<b>dec2hex</b>	Convert decimal to hexadecimal number in string
<b>hex2dec</b>	Convert hexadecimal number string to decimal number
<b>hex2num</b>	Convert hexadecimal number string to double-precision number
<b>num2hex</b>	Convert singles and doubles to IEEE hexadecimal strings

<b>cell2mat</b>	Convert cell array to numeric array
<b>cell2struct</b>	Convert cell array to structure array
<b>cellstr</b>	Create cell array of strings from character array
<b>mat2cell</b>	Convert array to cell array with potentially different sized cells
<b>num2cell</b>	Convert array to cell array with consistently sized cells
<b>struct2cell</b>	Convert structure to cell array

## Determination of Data Types

---

MATLAB provides various functions for identifying data type of a variable.

Following table provides the functions for determining the data type of a variable:

<b>Function</b>	<b>Purpose</b>
<b>is</b>	Detect state
<b>isa</b>	Determine if input is object of specified class
<b>iscell</b>	Determine whether input is cell array
<b>iscellstr</b>	Determine whether input is cell array of strings
<b>ischar</b>	Determine whether item is character array
<b>isfield</b>	Determine whether input is structure array field
<b>isfloat</b>	Determine if input is floating-point array

<b>ishandle</b>	True for Handle Graphics object handles
<b>isinteger</b>	Determine if input is integer array
<b>isjava</b>	Determine if input is Java object
<b>islogical</b>	Determine if input is logical array
<b>isnumeric</b>	Determine if input is numeric array
<b>isobject</b>	Determine if input is MATLAB object
<b>isreal</b>	Check if input is real array
<b>isscalar</b>	Determine whether input is scalar
<b>isstr</b>	Determine whether input is character array
<b>isstruct</b>	Determine whether input is structure array
<b>isvector</b>	Determine whether input is vector
<b>class</b>	Determine class of object
<b>validateattributes</b>	Check validity of array
<b>whos</b>	List variables in workspace, with sizes and types

### Example

Create a script file with the following code:

```
x = 3
isinteger(x)
```

```
isfloat(x)
isvector(x)
isscalar(x)
isnumeric(x)

x = 23.54
isinteger(x)
isfloat(x)
isvector(x)
isscalar(x)
isnumeric(x)

x = [1 2 3]
```

End of ebook preview  
If you liked what you saw...  
Buy it from our store @ <https://store.tutorialspoint.com>