

# MATLAB Cheat Sheet

## Basic Commands

---

<code>%</code>	Indicates rest of line is commented out.
<code>;</code>	If used at end of command it suppresses output. If used within matrix definitions it indicates the end of a row.
<code>save filename</code>	Saves all variables currently in workspace to file <code>filename.mat</code> .
<code>save filename x y z</code>	Saves $x$ , $y$ , and $z$ to file <code>filename.mat</code> .
<code>save -append filename x</code>	Appends file <code>filename.mat</code> by adding $x$ .
<code>load filename</code>	Loads variables from file <code>filename.mat</code> to workspace.
<code>!</code>	Indicates that following command is meant for the operating system.
<code>...</code>	Indicates that command continues on next line.
<code>help function/command</code>	Displays information about the function/command.
<code>clear</code>	Deletes all variables from current workspace.
<code>clear all</code>	Basically same as <code>clear</code> .
<code>clear x y</code>	Deletes $x$ and $y$ from current workspace.
<code>home</code>	Moves cursor to top of command window.
<code>clc</code>	Homes cursor and clears command window.
<code>close</code>	Closes current figure window.
<code>close all</code>	Closes all open figure windows.
<code>close(H)</code>	Closes figure with handle $H$ .
<code>global x y</code>	Defines $x$ and $y$ as having global scope.
<code>keyboard</code>	When placed in an M-file, stops execution of the file and gives control to the user's keyboard. Type <code>return</code> to return control to the M-file or <code>dbquit</code> to terminate program.
<code>A=xlsread('data',... 'sheet1','a3:b7')</code>	Sets $A$ to be a 5-by-2 matrix of the data contained in cells A3 through B7 of sheet <code>sheet1</code> of excel file <code>data.xls</code>
<code>Success=xlswrite(... 'results',A,'sheet1','c7')</code>	Writes contents of $A$ to sheet <code>sheet1</code> of excel file <code>results.xls</code> starting at cell C7. If successful <code>success=1</code> .
<code>path</code>	Display the current search path for <code>.m</code> files
<code>addpath c:\my_functions</code>	Adds directory <code>c:\my_functions</code> to top of current search path.
<code>rmpath c:\my_functions</code>	Removes directory <code>c:\my_functions</code> from current search path.
<code>disp('random statement')</code>	Prints <code>random statement</code> in the command window.
<code>disp(x)</code>	Prints only the value of $x$ on command window.
<code>disp(['x=',num2str(x,5)])</code>	Displays <code>x=</code> and first 5 digits of $x$ on command window. Only works when $x$ is scalar or row vector.
<code>fprintf(... 'The %g is %4.2f.\n', x,sqrt(x))</code>	Displays <code>The 3 is 1.73.</code> on command window.
<code>format short</code>	Displays numeric values in floating point format with 4 digits after the decimal point.
<code>format long</code>	Displays numeric values in floating point format with 15 digits after the decimal point.

## Plotting Commands

---

<code>figure(H)</code>	Makes $H$ the current figure. If $H$ does not exist is creates $H$ .
------------------------	--

<code>plot(x, y)</code>	Note that $H$ must be a positive integer.
<code>plot(y)</code>	Cartesian plot of $x$ versus $y$ .
<code>plot(x, y, 's')</code>	Plots columns of $y$ versus their index.
<code>semilogx(x, y)</code>	Plots $x$ versus $y$ according to rules outlined by $s$ .
<code>semilogy(x, y)</code>	Plots $\log(x)$ versus $y$ .
<code>loglog(x, y)</code>	Plots $x$ versus $\log(y)$ .
<code>grid</code>	Plots $\log(x)$ versus $\log(y)$ .
<code>title('text')</code>	Adds grid to current figure.
<code>xlabel('text')</code>	Adds title <code>text</code> to current figure.
<code>ylabel('text')</code>	Adds x-axis label <code>text</code> to current figure.
<code>hold on</code>	Adds y-axis label <code>text</code> to current figure.
<code>hold off</code>	Holds current figure as is so subsequent plotting commands add to existing graph.
	Restores hold to default where plots are overwritten by new plots.

## Creating Matrices/Special Matrices

---

<code>A=[1 2;3 4]</code>	Defines $A$ as a 2-by-2 matrix where the first row contains the numbers 1, 2 and the second row contains the number 3, 4.
<code>B=[1:1:10]</code>	Defines $B$ as a vector of length 10 that contains the numbers 1 through 10.
<code>A=zeros(n)</code>	Defines $A$ as an $n$ -by- $n$ matrix of zeros.
<code>A=zeros(m,n)</code>	Defines $A$ as an $m$ -by- $n$ matrix of zeros.
<code>A=ones(n)</code>	Defines $A$ as an $n$ -by- $n$ matrix of ones.
<code>A=ones(n,m)</code>	Defines $A$ as an $m$ -by- $n$ matrix of ones.
<code>A=eye(n)</code>	Defines $A$ as an $n$ -by- $n$ identity matrix.
<code>A=repmat(x,m,n)</code>	Defines $A$ as an $m$ -by- $n$ matrix in which each element is $x$ .

`linspace(x1, x2, n)`

Generates  $n$  points between  $x1$  and  $x2$ .

## Matrix Operations

---

<code>A*B</code>	Matrix multiplication. Number of columns of $A$ must equal number of rows of $B$ .
<code>A^n</code>	$A$ must be a square matrix. If $n$ is an integer and $n > 1$ then $A^n$ is $A$ multiplied with itself $n$ times. Otherwise, $A^n$ is the solution to $A^n v_i = l_i v_i$ where $l_i$ is an eigenvalue of $A$ and $v_i$ is the corresponding eigenvector.
<code>A/B</code>	This is equivalent to <code>A*inv(B)</code> but computed more efficiently.
<code>A\B</code>	This is equivalent to <code>inv(A)*B</code> but computed more efficiently.
<code>A.*B, A./B, A.\B, A.^n</code>	Element-by-element operations.
<code>A'</code>	Returns the transpose of $A$ .
<code>inv(A)</code>	Returns the inverse of $A$ .
<code>length(A)</code>	Returns the larger of the number of rows and columns of $A$ .
<code>size(A)</code>	Returns of vector that contains the dimensions of $A$ .
<code>size(A,1)</code>	Returns the number of rows in $A$ .
<code>reshape(A,m,n)</code>	Reshapes $A$ into an $m$ -by- $n$ matrix.

`kron(A,B)`

`A = [A X]`

`A = [A; Y]`

Computes the Kronecker tensor product of  $A$  with  $B$ .

Concatenates the m-by-n matrix  $A$  by adding the m-by-k matrix  $X$  as additional columns.

Concatenates the m-by-n matrix  $A$  by adding the k-by-n vector  $Y$  as additional rows.

## Data Analysis Commands

---

`rand(m,n)`

`randn(m,n)`

`max(x)`

`min(x)`

`mean(x)`

`sum(x)`

`prod(x)`

`std(x)`

`var(x)`

Generates an m-by-n matrix of uniformly distributed random numbers.

Generates an m-by-n matrix of normally distributed random numbers.

If  $x$  is a vector it returns the largest element of  $x$ .

If  $x$  is a matrix it returns a row vector of the largest element in each column of  $x$ .

Same as `max` but returns the smallest element of  $x$ .

If  $x$  is a vector it returns the mean of the elements of  $x$ .

If  $x$  is a matrix it returns a row vector of the means for each column of  $x$ .

If  $x$  is a vector it returns the sum of the elements of  $x$ .

If  $x$  is a matrix it returns a row vector of the sums for each column of  $x$ .

Same as `sum` but returns the product of the elements of  $x$ .

If  $x$  is a vector it returns the standard deviation of the elements of  $x$ .

If  $x$  is a matrix it returns a row vector of the standard deviations for each column of  $x$ .

Same as `std` but returns the variance of the elements of  $x$ .

## Conditionals and Loops

---

`for i=1:10`

`procedure`

`end`

Iterates over `procedure` incrementing  $i$  from 1 to 10 by 1.

`while(criteria)`

`procedure`

`end`

Iterates over `procedure` as long as `criteria` is true.

`if(criteria 1)`

`procedure 1`

`elseif(criteria 2)`

`procedure 2`

`else`

`procedure 3`

`end`

If `criteria 1` is true do `procedure 1`, else if `criteria 2` is true do `procedure 2`, else do `procedure 3`.