

Starting and Stopping

<code>octave</code> <code>[--gui]</code>	start Octave CLI/GUI session
<code>octave</code> <code>file</code>	run Octave commands in <code>file</code>
<code>octave</code> <code>--eval</code> <code>code</code>	evaluate <code>code</code> using Octave
<code>octave</code> <code>--help</code>	describe command line options
<code>quit</code> or <code>exit</code>	exit Octave
<code>Ctrl-C</code>	terminate current command and return to top-level prompt

Getting Help

<code>help</code> <code>command</code>	briefly describe <code>command</code>
<code>doc</code>	use Info to browse Octave manual
<code>doc</code> <code>command</code>	search for <code>command</code> in Octave manual
<code>lookfor</code> <code>str</code>	search for <code>command</code> based on <code>str</code>

Command Completion and History

<code>TAB</code>	complete a command or variable name
<code>Alt-?</code>	list possible completions
<code>Ctrl-r</code> <code>Ctrl-s</code>	search command history

Directory and Path Commands

<code>cd</code> <code>dir</code>	change working directory to <code>dir</code>
<code>pwd</code>	print working directory
<code>ls</code> <code>[options]</code>	print directory listing
<code>what</code>	list <code>.m/.mat</code> files in the current directory
<code>path</code>	search path for Octave functions
<code>pathdef</code>	default search path
<code>addpath</code> (<code>dir</code>)	add a directory to the path
<code>getenv</code> (<code>var</code>)	value of environment variable

Package Management

Add-on packages are independent of core Octave, listed at <https://packages.octave.org/>

<code>pkg</code> <code>install</code> <code>-forge</code> <code>pkg</code>	download and install <code>pkg</code>
<code>pkg</code> <code>install</code> <code>file.tar.gz</code>	install pre-downloaded package file
<code>pkg</code> <code>list</code>	show installed packages
<code>pkg</code> <code>load</code> / <code>pkg</code> <code>unload</code>	load/unload installed package
<code>statistics</code> <code>optimization</code>	various common packages
<code>control</code> <code>signal</code> <code>image</code> <code>symbolic</code> etc.	

Matrices

Square brackets delimit literal matrices. Commas separate elements on the same row. Semicolons separate rows. Commas may be replaced by spaces, and semicolons may be replaced by newlines. Elements of a matrix may be arbitrary expressions, assuming all the dimensions agree.

<code>[x, y, ...]</code>	enter a row vector
<code>[x; y; ...]</code>	enter a column vector
<code>[w, x; y, z]</code>	enter a 2×2 matrix
<code>rows</code> <code>columns</code>	number of rows/columns of matrix
<code>zeros</code> <code>ones</code>	create matrix of zeros/ones
<code>eye</code> <code>diag</code>	create identity/diagonal matrix
<code>rand</code> <code>randi</code> <code>randn</code>	create matrix of random values
<code>sparse</code> <code>spalloc</code>	create a sparse matrix
<code>all</code>	true if all elements nonzero

<code>any</code>	true if at least one element nonzero
<code>nnz</code>	number of nonzero elements

Multi-dimensional Arrays

<code>ndims</code>	number of dimensions
<code>reshape</code> <code>squeeze</code>	change array shape
<code>resize</code>	change array shape, lossy
<code>cat</code>	join arrays along a given dimension
<code>permute</code> <code>ipermute</code>	like N-dimensional transpose
<code>shiftdim</code> <code>circshift</code>	cyclically shift array elements
<code>meshgrid</code>	matrices useful for vectorization

Ranges

Create sequences of real numbers as row vectors.

<code>base</code> : <code>limit</code>	
<code>base</code> : <code>incr</code> : <code>limit</code>	
<code>incr</code> == 1 if not specified. Negative ranges allowed.	

Numeric Types and Values

Integers saturate in Octave. They do not roll over.

<code>int8</code> <code>int16</code> <code>int32</code> <code>int64</code>	signed integers
<code>uint8</code> <code>uint16</code> <code>uint32</code> <code>uint64</code>	unsigned integers
<code>single</code> <code>double</code>	32-bit/64-bit IEEE floating point
<code>intmin</code> <code>intmax</code> <code>flintmax</code> <code>realmin</code> <code>realmax</code>	integer limits of given type floating point limits of given type
<code>inf</code> <code>nan</code> <code>NA</code>	IEEE infinity, NaN, missing value
<code>eps</code>	machine precision
<code>pi</code> <code>e</code>	3.14159..., 2.71828...
<code>i</code> <code>j</code>	$\sqrt{-1}$

Strings

A *string constant* consists of a sequence of characters enclosed in either double-quote or single-quote marks. Strings in double-quotes allow the use of the escape sequences below.

<code>\\</code>	a literal backslash
<code>\"</code>	a literal double-quote character
<code>\'</code>	a literal single-quote character
<code>\n</code>	newline, ASCII code 10
<code>\t</code>	horizontal tab, ASCII code 9
<code>sprintf</code> <code>sscanf</code>	formatted IO to/from string
<code>strcmp</code>	compare strings
<code>strcat</code>	join strings
<code>strfind</code> <code>regexp</code>	find matching patterns
<code>strrep</code> <code>regexprep</code>	find and replace patterns

Index Expressions

<code>var(idx)</code>	select elements of a vector
<code>var(idx1, idx2)</code>	select elements of a matrix
<code>var([1 3], :)</code>	rows 1 and 3
<code>var(:, [2 end])</code>	the second and last columns
<code>var(1:2:end, 2:2:end)</code>	get odd rows and even columns
<code>var1(var2 == 0)</code>	elements of <code>var1</code> corresponding to zero elements of <code>var2</code>
<code>var(:)</code>	all elements as a column vector

Cells, Structures, and Classdefs

<code>var{idx}</code> = ...	set an element of a cell array
<code>cellfun</code> (<code>f</code> , <code>c</code>)	apply a function to elements of cell array
<code>var.field</code> = ...	set a field of a structure
<code>fieldnames</code> (<code>s</code>)	returns the fields of a structure
<code>structfun</code> (<code>f</code> , <code>s</code>)	apply a function to fields of structure
<code>classdef</code>	define new classes for OOP

Assignment Expressions

<code>var</code> = <code>expr</code>	assign value to variable
<code>var(idx)</code> = <code>expr</code>	only the indexed elements are changed
<code>var(idx)</code> = []	delete the indexed elements

Arithmetic Operators

If two operands are of different sizes, scalars and singleton dimensions are automatically expanded. Non-singleton dimensions need to match.

<code>x + y</code> , <code>x - y</code>	addition, subtraction
<code>x * y</code>	matrix multiplication
<code>x .* y</code>	element-by-element multiplication
<code>x / y</code>	right division, conceptually equivalent to <code>(inverse (y') * x')</code>
<code>x ./ y</code>	element-by-element right division
<code>x \ y</code>	left division, conceptually equivalent to <code>inverse (x) * y</code>
<code>x .\ y</code>	element-by-element left division
<code>x ^ y</code>	power operator
<code>x .^ y</code>	element-by-element power operator
<code>+=</code> <code>-=</code> <code>*=</code> <code>.*=</code> <code>/=</code> <code>./=</code> <code>\=</code> <code>.\=</code> <code>^=</code> <code>.^=</code>	in-place equivalents of the above operators
<code>-x</code>	negation
<code>+x</code>	unary plus (a no-op)
<code>x'</code>	complex conjugate transpose
<code>x.'</code>	transpose
<code>++x</code> <code>--x</code>	increment / decrement, return <i>new</i> value
<code>x++</code> <code>x--</code>	increment / decrement, return <i>old</i> value

Comparison and Boolean Operators

These operators work on an element-by-element basis. Both arguments are always evaluated.

<code><</code> <code><=</code> <code>==</code> <code>>=</code> <code>></code>	relational operators
<code>!=</code> <code>~=</code>	not equal to
<code>&</code>	logical AND
<code> </code>	logical OR
<code>!</code> <code>~</code>	logical NOT

Short-circuit Boolean Operators

Operators evaluate left-to-right. Operands are only evaluated if necessary, stopping once overall truth value can be determined. Non-scalar operands are converted to scalars with `all`.

<code>x && y</code>	logical AND
<code>x y</code>	logical OR

Operator Precedence		
Table of Octave operators, in order of decreasing precedence.		
() {} .	array index, cell index, structure index	
' ' ^ .^	transpose and exponentiation	
+ - ++ -- !	unary minus, increment, logical “not”	
* / \ .* ./ .\	multiplication and division	
+ -	addition and subtraction	
:	colon	
< <= == > > !=	relational operators	
&	element-wise “and” and “or”	
&&	logical “and” and “or”	
= += -= *= /= etc.	assignment, groups left to right	
; ,	statement separators	

General programming

endfor, endwhile, endif etc. can all be replaced by end.

for x = 1:10	for loop
endfor	
while (x <= 10)	while loop
endwhile	
do	do-until loop
until (x > 10)	
if (x < 5)	if-then-else
elseif (x < 6)	
else	
endif	
switch (tf)	switch-case
case "true"	
case "false"	
otherwise	
endswitch	
break	exit innermost loop
continue	go to start of innermost loop
return	jump back from function to caller
try	cleanup only on exception
catch	
unwind_protect	cleanup always
unwind_protect_cleanup	

Functions

```
function [ret-list =] function-name [(arg-list)]
    function-body
endfunction
```

ret-list may be a single identifier or a comma-separated list of identifiers enclosed by square brackets.

arg-list is a comma-separated list of identifiers and may be empty.

Function Handles and Evaluation

```
@func create a function handle to func
@(vars) expr define an anonymous function
str2func func2str convert function to/from string
functions (handle) Return information about a function handle
```

```
f (args) Evaluate a function handle f
feval Evaluate a function handle or string
eval (str) evaluate str as a command
system (cmd) execute arbitrary shell command string
```

Anonymous function handles make a copy of the variables in the current workspace at the time of creation.

Global and Persistent Variables

```
global var = ... declare & initialize global variable
persistent var = ... persistent/static variable
```

Global variables may be accessed inside the body of a function without having to be passed in the function parameter list provided that they are declared global when used.

Common Functions

```
disp display value of variable
printf formatted output to stdout
input scanf input from stdin
who whos list current variables
clear pattern clear variables matching pattern
exist check existence of identifier
find return indices of nonzero elements
sort return a sorted array
unique discard duplicate elements
sortrows sort whole rows in numerical or
lexicographic order

sum prod sum or product
mod rem remainder functions
min max range meanbasic statistics
median std
```

Error Handling, Debugging, Profiling

```
error (message) print message and return to top level
warning (message) print a warning message
debug guide to all debugging commands
profile start/stop/clear/resume profiling
profshow show the results of profiling
profexplore
```

File I/O, Loading, Saving

```
save load save/load variables to/from file
save -binary save in binary format (faster)
dlmread dlmwrite read/write delimited data
csvread csvwrite read/write CSV files
xlsread xlswrite read/write XLS spreadsheets
```

```
fopen fclose open/close files
fprintf fscanf formatted file I/O
textscan
fflush flush pending output
```

Math Functions

Run doc <function> to find related functions.

```
cov corrcoef covariance, correlation coefficient
tan tanh atan2 trig and hyperbolic functions
cross curl del2 vector algebra functions
```

```
det inv determinant matrix inverse
eig eigenvalues and eigenvectors
norm vector norm, matrix norm
```

```
rank matrix rank
qr QR factorization
chol Cholesky factorization
svd singular value decomposition
```

```
fsolve solve nonlinear algebraic equations
lsode ode45 integrate nonlinear ODEs
dassl integrate nonlinear DAEs
integral integrate nonlinear functions
```

```
union set union
intersection set intersection
setdiff set difference
```

```
roots polynomial roots
poly matrix characteristic polynomial
polyder polyint polynomial derivative or integral
polyfit polyval polynomial fitting and evaluation
residue partial fraction expansion
legendre bessell special functions
```

```
conv conv2 convolution, polynomial multiplication
deconv deconvolution, polynomial division
```

```
fft fft2 ifft(a) FFT / inverse FFT
freqz FIR filter frequency response
filter filter by transfer function
```

Plotting and Graphics

```
plot plot3 2D / 3D plot with linear axes
line 2D or 3D line
patch fill 2D patch, optionally colored
semilogx semilogy logarithmic axes
loglog
bar hist bar chart, histogram
stairs stem stairsteps and stem graphs
contour contour plot
mesh trimesh surf plot 3D surfaces
```

```
figure new figure
hold on add to existing figure
title set plot title
axis set axis range and aspect
xlabel ylabel zlabel set axis labels
text add text to a plot
grid legend draw grid or legend
```

```
image imagesc spy display matrix as image
imwrite saveas print save figure or image
imread load an image
colormap get or set colormap
```

Quick reference for Octave 8.0.0. Copyright 1996-2024 The Octave Project Developers. The authors assume no responsibility for any errors on this card. This card may be freely distributed under the terms of the GNU General Public License.

Octave license and copyright: <https://octave.org/copyright/>

TpX Macros for this card by Roland Pesch (pesch@cygnus.com), originally for the GDB reference card