```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Introducing Matlab (adapted from http://www.cns.nyu.edu/~eero and
% http://www.cs.dartmouth.edu/~farid/teaching/cs88/matlab.intro.html)
% via http://www-cse.ucsd.edu/%7Esjb/classes/matlab/matlab.intro.html
% obtained from http://www.cs.unc.edu/~lazebnik/spring11/matlab.intro.html
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (1) Help and basics

% The symbol "%" is used in front of a comment.

% To get help type "help" (will give list of help topics) or "help topic"

% If you don't know the exact name of the topic or command you are looking for,
% type "lookfor keyword" (e.g., "lookfor regression")

% When writing a long matlab statement that exceeds a single row use ...
% to continue statement to next row.

% When using the command line, a ";" at the end means matlab will not
% display the result. If ";" is omitted then matlab will display result.

% Use the up-arrow to recall commands without retyping them (and down
% arrow to go forward in commands).

% Other commands borrowed from emacs and/or tcsh:
% C-a moves to beginning of line (C-e for end), C-f moves forward a
% character (C-b moves back), C-d deletes a character, C-k deletes
% the line to the right of the cursor, C-p goes back through the
% command history and C-n goes forward (equivalent to up and down arrows),
% tab command completion.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (2) Objects in matlab -- the basic objects in matlab are scalars,
% vectors, and matrices...


N       = 5                             % a scalar
v       = [1 0 0]                       % a row vector
v       = [1;2;3]                       % a column vector
v       = v'                            % transpose a vector
                                                (row to column or column to row)
v       = [1:.5:3]                      % a vector in a specified range:
v       = pi*[-4:4]/4                   %       [start:end] or [start:stepsize:end]
v       = []                            % empty vector


m       = [1 2 3; 4 5 6]                % a matrix: 1ST parameter is ROWS
                                        %           2ND parameter is COLS
m       = zeros(2,3)                    % a matrix of zeros
v       = ones(1,3)                     % a matrix of ones
m       = eye(3)                        % identity matrix
v       = rand(3,1)                     % random matrix with values in [0,1] (see also
randn)

load matrix_data                        % read data from a file:
                                        % create a file 'matrix_data' containing:
                                        %       2     3     4
                                        %       5     6     7
                                        %       1     2     3
matrix_data
```

```
v       = [1 2 3];                      % access a vector element
v(3)                                    %       vector(number)
                                        % Index starts from 1


m       = [1 2 3; 4 5 6]
m(1,3)                                  % access a matrix element
                                        %       matrix(rownumber, columnnumber)
m(2,:)                                  % access a matrix row (2nd row)
m(:,1)                                  % access a matrix column (1st row)


size(m)                                 % size of a matrix
size(m,1)                               % number rows
size(m,2)                               % number of columns

m1      = zeros(size(m))                % create a new matrix with size of m

who                                     % list of variables
whos                                    % list/size/type of variables


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (3) Simple operations on vectors and matrices

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (A) Pointwise (element by element) Operations:

% addition of vectors/matrices and multiplication by a scalar
% are done "element by element"
a       = [1 2 3 4];                    % vector
2 * a                                   % scalar multiplication
a / 4                                   % scalar multiplication
b       = [5 6 7 8];                    % vector
a + b                                   % pointwise vector addition
a - b                                   % pointwise vector addition
a .^ 2                                  % pointise vector squaring (note .)
a .* b                                  % pointwise vector multiply (note .)
a ./ b                                  % pointwise vector divide (note .)

log( [1 2 3 4] )                        % pointwise arithmetic operation
round( [1.5 2; 2.2 3.1] )               % pointwise arithmetic operation


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% (B) Vector Operations (no for loops needed)
% Built-in matlab functions operate on vectors, if a matrix is given,
% then the function operates on each column of the matrix

a       = [1 4 6 3]                     % vector
sum(a)                                  % sum of vector elements
mean(a)                                 % mean of vector elements
var(a)                                  % variance
std(a)                                  % standard deviation
max(a)                                  % maximum


a       = [1 2 3; 4 5 6]                % matrix
a(:)                                    % vectorized version of the matrix
mean(a)                                 % mean of each column
max(a)                                  % max of each column
max(max(a))                             % to obtain max of matrix
max(a(:))                               %       or...
```

```
%%%%%%%%%%%%%%%%%%%%%%%%
% (C) Matrix Operations:

[1 2 3] * [4 5 6]'                      % row vector 1x3 times column vector 3x1
                                        % results in single number, also
                                        % known as dot product or inner product

[1 2 3]' * [4 5 6]                      % column vector 3x1 times row vector 1x3
                                        % results in 3x3 matrix, also
                                        % known as outer product

a       = rand(3,2)                     % 3x2 matrix
b       = rand(2,4)                     % 2x4 matrix
c       = a * b                         % 3x4 matrix

a       = [1 2; 3 4; 5 6]               % 3 x 2 matrix
b       = [5 6 7];                      % 1 x 3 vector
b * a                                   % matrix multiply
a' * b'                                 % matrix multiply


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%(4) Saving your work

save mysession                          % creates mysession.mat with all variables
save mysession a b                      % save only variables a and b

clear all                               % clear all variables
clear a b                               % clear variables a and b

load mysession                          % load session
a
b


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%(5) Relations and control statements

% Example: given a vector v, create a new vector with values equal to
% v if they are greater than 0, and equal to 0 if they less than or
% equal to 0.


v       = [3 5 -2 5 -1 0]               % 1: FOR LOOPS
u       = zeros( size(v) );             % initialize
for i = 1:size(v,2)                     % size(v,2) is the number of columns
        if( v(i) > 0 )
                u(i) = v(i);
        end
end
u

v       = [3 5 -2 5 -1 0]               % 2: NO FOR LOOPS
u2      = zeros( size(v) );             % initialize
ind     = find( v>0 )                   % index into >0 elements
u2(ind) = v( ind )


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%(6) Creating functions using m-files:
% Functions in matlab are written in m-files. Create a file called
```

```
% 'thres.m' In this file put the following 4 lines:

function res = thres( v )
u       = zeros( size(v) );           % initialize
ind     = find( v>0 )                 % index into >0 elements
u(ind)  = v( ind )



v       = [3 5 -2 5 -1 0]
thres( v )                            % call from command line



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%(7) Plotting

x       = [0 1 2 3 4];                % basic plotting
plot( x );
plot( x, 2*x );
axis( [0 8 0 8] );

x       = pi*[-24:24]/24;
plot( x, sin(x) );
xlabel( 'radians' );
ylabel( 'sin value' );
title( 'dummy' );
gtext( 'put cursor where you want text and press mouse' );

figure;                               % multiple functions in separate graphs
subplot( 1,2,1 );
plot( x, sin(x) );
axis square;
subplot( 1,2,2 );
plot( x, 2.*cos(x) );
axis square;

figure;                               % multiple functions in single graph
plot( x,sin(x) );
hold on;                              % hold on tells matlab to write on top
plot (x, 2.*cos(x), '--' );           % of the current plot
legend( 'sin', 'cos' );
hold off;

figure;                               % matrices as images
m = rand(64,64);
imagesc(m)
colormap gray;
axis image
axis off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%(8) Working with the Images and the Matlab Image Processing Toolbox

[I,map]=imread('trees.tif');            % use as it is, Matlab has pre-stored images

figure
imshow(I,map)                           % display it as indexed image w/colormap

I2=ind2gray(I,map);                     % convert it to grayscale

figure
imagesc(I2,[0 1])                       % scale data to use full colormap
```

```
                                        %  for values between 0 and 1
colormap('gray')                        % use gray colormap
axis('image')                           % make displayed aspect ratio proportional
                                        %  to image dimensions

I=imread('football.jpg');               % read a JPEG image into 3D array

figure
imshow(I)
rect=getrect;                           % select rectangle
I2=imcrop(I,rect);                      % crop
I2=rgb2gray(I2);                        % convert cropped image to grayscale
imagesc(I2)                             % scale data to use full colormap
                                        %  between min and max values in I2
colormap('gray')
colorbar                                % turn on color bar
impixelinfo                             % display pixel values interactively
truesize                                % display at resolution of one screen pixel
                                        %  per image pixel
truesize(2*size(I2))                    % display at resolution of two screen pixels
                                        %  per image pixel

I3=imresize(I2,0.5,'bil');              % resize by 50% using bilinear
                                        %  interpolation
I3=imrotate(I2,45,'bil','crop');        % rotate 45 degrees and crop to
                                        %  original size
I3=double(I2);                          % convert from uint8 to double, to allow
                                        %  math operations
imagesc(I3.^2)                          % display squared image (pixel-wise)
imagesc(log(I3))                        % display log of image

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```