```
%  Matlab_intro:   An attempt at a Quick tutorial for Matlab (Matlab Intro-1/4)
%  ------------------------------------------------------------
%           The lines that start with the percent sign are comments.
%           You don't have to type them.  Matlab ignores anything after a %.

% starting up Matlab  -- use the icon on the "Dock" at the bottom of the screen.
%  A window should open with four "pane"s.  The main one is the command window.
%  Useful information here is:
%  1) What folder is matlab looking in for files?  (Current Folder)
%  2) What variables are already stored in memory? (Workspace)
%  NOTE: you might want to make the Command Window larger and the others smaller.

%%%%%  The Command Window
%  Try it Out!   In the command window, type
2+2       % notice that matlab can simply be a calculator
x=10         % set x to 10
x=10;        % the semicolon suppresses the output
x            % to see what x is equal to

% special variables:
ans      % the answer of the last unassigned expression.
pi       % 3.1415.....
eps      % the smallest positive number on this computer

% Vectors of numbers
x=[2 3 6]
2*x           % can multiply many numbers by a constant
2+x           % can add a constant to many numbers
x=[2:2:10]    % this notation is the same as [2 4 6 8 10]
% it works as     start:step:end
y=sin(x)      % you can create a new list of values from the other list.
x=[0:.01:1]         % This will give you a LONG output
%  When the output scrolls off the top of your window, use the command "more on"
%  and it will stop at each page and wait for a spacebar (or q to quit).
more on
%  Arrows go through your history: use the up arrow to repeat or edit a command.
x=[0:0.01:1]    % try the "more" feature--press q to skip to the end.
more off        % Use the up arrow and edit "on" to "off"

x=[-pi:.01:pi]; % Don't forget the semicolon!!!  (it supresses output)
plot(x,sin(x))      % Plot two lists of values (they must be the same length)
x=linspace(-pi,pi,5)  % 5 evenly spaced points between -pi and pi
%  this is an alternative to start:step:end
%  (there is also a function "logspace" which gives log spacing for log plots.)
plot(x,sin(x))
%  Use the arrow keys to go back to the previous commands and change
%  the number of points in the plot to 200. Use ; to suppress output.
%% Getting Help     at the command line, type
help linspace   % or "doc linspace" or use the Help Menu (question mark icon)

% MATRICES
A= [ 1 2; 3 4]
B= [ 1 1; 1 1]
A*B             % * means MATRIX MULTIPLICATION
A.*B            % .* means componentwise multiplication (each entry)
A^2             % Exponents work the same way!!!
A.^2
eye(5)          % create the "identity matrix" of size 5
ones(5)         % create a square matrix of ones
ones(5,1)       % create a list of ones
zeros(5,1)      % create a list of zeros
x=[2 3 6]
average = x*ones(3,1)/3  % note: scalar division because 3 is a scalar (not matrix)
%   Try the same line as above with ones(1,3) replacing ones(3,1).  What happens?

%Punctuation:      If you want to continue a line, use ... at the end
longone = 1+2+3+4+5+ ...
6+7+8+9+10
```

```
%Mathematical functions                              (Matlab Intro-2/4)
x=[0 1]
sqrt(x)
sin(x)
asin(x)   % arcsin
exp(x)
log(x)      %  ?? Log of zero?  What's that?

%%%%%  The Workspace Window
%look in the "Workspace" tab to see what variables are defined.
% Double-click on an array--- this should open an array editor.
% you can update it and close the editor (use the "x" in upper left)
%   OR you can use comands:
who      % lists all variables
whos     % tells how much memory they take up
clear x  % removes the value of x
clear    % removes the definitions of all variables (start over)

%%%%% Display  Format
pi
format long
pi
format short
pi
format longE  % long exponential
pi
format compact
pi
format loose
pi
% Which is the default way to show numbers?
% To change the default, create a new script (use icon) containing two lines:
format compact
format long
% Save the script as startup.m.  This script (in your Documents/Matlab Folder)
% will be run each time you start Matlab.
%    For this class always use format compact; format long  (or format longE)

%%%%%%  Working With Files  %%%%%%
% You can traverse the Folder "tree" in the window pane named Current Folder.
% You can also use the command window if you prefer:
pwd  % display the "present work directory" (current folder).
% Other useful commands to explore:  mkdir, cd, ls, delete

% Matlab starts "in" your Documents/Matlab folder.
% Matlab looks in your current Folder to find your files.
% Create a subfolder named Unit0 for this tutorial. Go to that subfolder.

%%%%%  Create a script file %%%%%%
% All commands can be entered into a file and run as a script.
% Try entering the commands needed to plot sin(x) from 0 to pi
% with grid spacings of .01 into a new file called plotsin.m
% You name it when you save it. Name it plotsin.m (Matlab needs the .m)
plotsin      % (you don't need to type the .m, matlab only looks for .m files)

% While Loops:
n=10;
while n>1  % The "loop" of code between while and end repeats while n>1
    n=n-1
end        % Try replacing > with >=
% There are also repeat loops (repeat...until) and for loops (for i=1:10)

% If conditions:
if x>1
  disp('x is greater than 1')
else
  disp('x is too small')
end
```

```
%%%%%  Creating Your Own Functions                    (Matlab Intro-3/4)
% Now make a file called "squared.m" which defines the following function
%  The file should look something like:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function f = squared(x)
%
%    Comments go here to say that it is a function to return x^2.
%    These comments are shown when you type "help squared" in the
%    command window.
%
f=x.^2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Now see if your function works:   in the command window, type
squared(10)
%%  ??Did it work?       (Yeah!)
% Check if "f" is now defined -- Functions protect their local variables.
% See if your help text works
help squared
%Try it on a vector:
squared([2 3 6])
%      Go back into the editor and change the f=x.^2; line to f=x^2;
squared([2 3 6])  % ???  what is the error?

%     Go back to the editor window and change the function so that it
%     returns the CUBE of a number  (using .^ for exponentiation).
squared(10)   % Despite the name, you should get 1000
%   Now Quickly change the file back to be a real square function, or we'll
%   get really confused! :)

% You can also define simple one-line functions like this in the Command Window:
f = @(x) x.^2;
f([2 3 6])
% f is a "function handle". It can be passed into other functions if needed.

% Download the file newton.m from the class webpage.
% Move it so Matlab can find it. (You can also cut and paste it into the editor.)
% Look at the file, the help text. In the command window define the functions
% and call newton to solve x^2=0.

%%%%%%%%  Timing programs   (tic and toc)
% Now copy squared.m and rename it slowfunction.m to create a "slow function".
% We'll simply mimic a slow process by pausing the computation for 1 second.
% Add a line: pause(1)    % and save
% Use the tic and toc commands to time the slow function
tic; slowfunction(10); toc         % try with 3 separate lines too

%%%%  Assignment scripts
% Create a script called unit0.m. The script should use Newton's method to solve
% x^2=0 starting at the guess x=1 and time how long that takes.
% You will need to pass in function handles. Use @squared for the f(x) function
% and use anonymous functions to define fprime(x).

% Run the script file to make sure it works:
unit0
% now run it again and store the output in a file
diary('output.txt')
unit0
diary off
%%%%%%%% WARNING: diary appends the output to the named file. You can write
% a lot of stuff there if you aren't paying attention. To create the file to
% hand in, delete old versions before running the diary command.

%%%%  Handing in assignments
% Outside of Matlab, in a Finder window navigate to the folder for your assignment.
% Put the files you wish to hand in into a Folder and Control-click on that folder.
% Select the option to Compress the folder. This will create a zip file that you can
% rename using your email address and assignment (e.g. dschultU1.zip) and attach to email.
```

```
%%%%%  Other tips for Matlab                    (Matlab Intro-4/4)

%  Printing Numbers:  How to format numbers and mix them with text.
fprintf('Iteration %2i: x=%18.12f, f(x)=%18.12f \n', 1,3.2, 3.2^2)

% In Matlab, "print" sends figures to the printer.
% fprintf() prints formatted numbers to the screen or to a file.
%   The first argument of fprintf() is a string (characters between single quotes)
%   Each time a %-sign appears in the string, the value of an input variable
%   is printed and formatted according to the code after the %-sign. If you have
%   3 %-signs you normally need to put three variables as arguments to fprintf().
% The code after the %-sign is described in the help system under "formatspec".
% Basically  %2i      integer with space for at least 2 digits.
%            %18.12f  floating point number with space for 18 characters and
%                     12 digits to right of decimal place.
%            %.12e    exponential notation with 12 digits to right of decimal place.
%            %g       "general": the more compact of 'e' or 'f' removing trailing zeros
% exponential notation is perhaps more exact but harder to read sometimes.
% These same codes are used in Python, C, Java, etc. If you program you will see them.



%  Function Handles:  How to pass a function as an argument of a function.
% Both Newton's method and Bisection can be applied to arbitrary functions.
% So we implement them with the desired function as input to the routine.
% That separates (abstracts) the choice of function from the root-finding method.
%
% To pass the function into e.g. newton.m you refer to it using a function handle.
% In these examples, f is the resulting function handle.
%  anonymous functions:  f=@(x) x^2;  or  f=@(x,y) sqrt(x^2+y^2);
%  existing Matlab function:   f=@sin     or f=@newton  (no space between @ and name)
%
% Function handles can be called as if they are Matlab functions:    f(10.2)
% and they can be passed into other functions:          bisection(f, 0.2, 1.2)

% The @() notation only allows a single command. If your function is more complex
% you MUST create a separate file for each such function. Most languages let you
% define more than one function in a file.

% when you write  myf(2) Matlab looks for a function handle "myf" first.
% Then it searches for files named myf.m in the current Folder.
% Then it looks for Matlab commands with the name "myf".
```