TODAY:
  - ROLL
  - SURVEY
  - SYLLABUS
  - INTRODUCTION TO OCTAVE

NEXT TIME:
  - INTERPOLATION
  - HAND OUT HW 1 (HW NORMALLY DUE WEDNESDAYS)

YOUR TASK BETWEEN NOW AND FRIDAY:
  - GET ACCESS TO OCTAVE OR MATLAB
  - PLAY WITH MY EXAMPLES (OR YOUR OWN!)
octave
a = [5, 7, sqrt(2)]  % Example of an array.
a.*a  % Product of arrays, element-by-element.
a./a  % Division element-by-element.
b = [1, 2, 3; 4, 5, 6]  % Example of a matrix (rank 2 tensor).
b'  % b' is the transpose of b.
b'*b  % Matrix product.
a*a'  % Dot product.
x = 0:0.2:2  % An array going from 0 to 2 in increments of 0.2.
pi  % What a surprise!
x *= pi  % This means the same as x = x*pi.
pi = 3  % Constants can be overwritten.
y = sin(x);  % The semicolon suppresses output.
z = cos(x);
plot(x,y)  % Just what you might expect.
plot(x,z)  % Nice, but the original plot disappeared.
hold on  % Subsequent plots will appear on the same axes.
plot(x,y)
hold off  % Subsequent plots will replace the existing figure.
plot(z,y)
M = [x, y, z];  % What happens when we do this?
size(M)  % The result has 1 row and 33 columns.
M  % Aha! they were concatenated into a single array.
M = [x', y', z'];  % A matrix (rank 2 tensor) with 3 columns.
size(M)
M
M(1:5, 1)  % Octave addresses arrays in "row, column" order.
dlmwrite("myfile.txt", M, "\t")  % Write M to a tab-delimited text file.
N = dlmread("myfile.txt", "\t")  % Read data back in to array N.
P = dlmread("myfile.txt", "\t", 2, 1)  % 2, 1 are row, column offsets.
sqrt(-1)  % Octave variables can take complex values.
i  % j is also defined.
exp(i*x')  % complex numbers work as they should.
c = [1, 0.1; 0.1, 1]  % A square matrix.
c^(-1)  % Matrix inverse. (What would c.^(−1) mean?)
eye(2)  % Identity matrix (2x2).
eye(2)/c  % Another way to express the inverse of c.
help plot  % Accessing Octave's built-in documentation.

% Below is the complete example with input and output, starting
% from the Unix prompt (in Mac OS X; should look identical, or
% very nearly so, in Linux).

kankel% octave
GNU Octave, version 3.0.3
% A file with the .m suffix can contain a script of commands to be run % in sequence. To execute this script, type 'example2' at the octave prompt. % Note that the .m file must exist either in the current working directory % or in octave's search path. To learn what is in octave's search path, % type the 'path' command at the octave prompt.

T = dlmread('myfile.txt','	');

x = T(:,2); % The : means all valid values of the (row) index.
y = T(:,3);

figure(1); % This controls which window the figure (plot) will appear in...
hold off % Just in case there's an existing figure on the screen.
plot(x,y)
hold on

% Now overplot a very nice circle with fine resolution.
theta = 0:0.01:2*pi; % You'll be sorry if you forget the semicolon...
xbig = cos(theta);
ybig = sin(theta);
plot(xbig,ybig)
plot(0.75*xbig, 1.5*ybig) % an ellipse.
% Note how the figure resized itself to accommodate a wider range of
% values.

% A nicer way to make a figure.
figure(2) % You can have lots of numbered figures on the screen at once.
hold off
plot(x, y, 'bo', 'markersize',15, xbig, ybig, 'g', 0.75*xbig, 1.5*ybig, %
 'k');
legend('points from myfile.txt', 'smooth circle', 'big smooth circle');
title('Plot of circles (from example2.m)');
xlabel('x (furlongs)');
ylabel('y (furlongs)');
axis('square'); % Make circles look like circles, and ellipses like
% ellipses!
% This command specifies that 1 unit in x is the same size as 1 unit in
y.
print('myplot.pdf', '-dpdf'); % High quality PDF output!

% One more plot...
tantheta = divide(ybig, xbig); % A user-defined function. Look at
divide.m.
figure(3)
hold off
plot(theta, tantheta, 'b', 'linewidth', 5)
axis([0,2*pi,-5,10])
hold on
% Overplot some helpful fiducial lines:
plot([pi/2, pi/2], [-5,10], 'k', [3*pi/2, 3*pi/2], [-5,10], 'k', [0,2*pi], %
 [0,0], 'k')
title("Plot of sine over cosine")
% An example function
function result = divide(a,b)

result = a./b;
% An example of sorting some data.

% Consider a data set consisting of x,y pairs:
x = [82, 10, 43, 46, 34, 55, 15, 23, 50, 28, 59];
y = [47, 27, 28, 28, 29, 35, 31, 32, 30, 31, 41];

% Plot them.
figure(1)
hold off
plot(x, y, 'r^-', 'markersize', 15) % Ugly!

% What can we do to make the figure look right?

% Instead, plot data sorted by x.
s = cksort(x); % See cksort.m, my cheesy sorting program.
figure(2)
hold off
plot(x(s), y(s), 'b^-', 'markersize', 15)
% This program provides a list of indices that sorts the array x. The sorting
% algorithm is very inefficient, O(N^2).
function sort_index = cksort(x)

% NOTE: indenting means nothing to octave, but makes the code more readable.
N = numel(x);  % number of elements in x.
for i=1:N
    irank = 1;  % initial value of the ith element rank. 1 means smallest.
    for j=1:N
        if ( x(j) < x(i) )  % each value of x that is smaller than x(i)...
            irank ++;  % ...implies the rank of x(i) should be increased.
        endif
    endfor
    sort_index(irank) = i;  % i.e.: "The irank-th smallest value of x is x(i)."
endfor
% Now x(irank) is a sorted version of the array x!