```
% WHOI math review: Programming
% This script contains a series of examples used during the course of the
% lecture to illustrate basic matrix operations and control flow
% operations.

% First: Open MATLAB
% Describe Current folder, workspace, editor, path, command window

% Remember that % is the comment character in MATLAB
% Writing %% at the beginning of a line starts a new cell in a MATLAB
% script. A cell can be evaluated by scrolling into it with the cursor and
% typing Ctrl+Enter. Note that there is a MATLAB variable type 'cell' which
% is something completely unrelated!

% mention wrapping around
% mention pressing the up key after typing something
% mention tab to complete

%% Variables
% To create a variable, simply assign a value to a name
% The variable name must start with a letter, but can include numbers afterwards

dog = 'happy'
mynumber = 1000

% variables are easily overwritten
dog = 'hungry'

% keep track of variables with 'whos'
whos

% variables can be saved as a *.mat
save danstuff dog mynumber

% variables can be deleted from the workspace...
clear dog
whos
clear

% ...and loaded back in
load danstuff

% back to slides

% to see your MATLAB path
path

% Look up a function
help min

%% How can we construct vectors and matrices?

% First way: use square brackets [] to concatenate elements
% Column vector
a = [1;2]
% Row vector
b = [0,1]
% Null vector
v = []
% Matrix
v = [1, NaN, 4,5; 1, 2, 3, 4]
% Make a matrix out of two vectors
v = [a,a]
v = [b,b]
```

```
% Matrices can't have elements with unspecified entries! e.g.
v = [a,b] % (fail)
clc

% Second way: use the colon operator :
v = 1.5:1:4;
v = 1.5:4 % implied step by 1!

% We can do this with variables, not just numbers.
% For example, remembering that pi is a reserved matlab variable,
v = 0:0.5:pi
v = pi:-0.5:0

% Third way: using built-in functions!
clc
% These are all pretty syntactically similar:
ones(4,1)
ones(5)
zeros(3,4);
nan(3,4);
rand(3,4); % uniform on [0,1]
randn(3,4); % standard normal
eye(5); % identity matrix
true(3); % matrix of boolean 1s
false(3); % matrix of boolean 0s
A = magic(3); % magic squares!
A = magic(3);

% linspace is very useful:
linspace(0,2*pi,5)

%% a couple of special functions:
% repmat
a = [1;2];
aa = repmat(a,1,10)
% reshape - sort of a weird one but occasionally extremely useful
a = magic(4)
reshape(a,2,8)
% size
[nr,nc] = size(aa)

% length
length(aa)

% questions?

%% Useful bits of matrix arithmetic
% what does this produce?
a = [1:3;2:4]
b = ones(2)

a+a % matrix addition
a+b % doesn't work - must be the same size!
a+1 % addition of a scalar and a vector
a/2 % division by a scalar
a' % matrix transpose
a'*a %
a*a' %

% Produces an error because a is not square:
a*a

% but you can do this:
a.*a
% the '.' denotes element-wise operations. here are others:
```

```
a./a % quotient
a./b % #fail
a.^2 % exponentiation
2.^a % exp. by a vector
a.^a

% Questions?

% Example: how can we make a 100x100 matrix of 2s?

%% Accessing elements of vectors and matrices

% MATLAB indexes starting at 1! (not 0).
% Start with vector:

v = [4 16 9 1 25] % just another example vector
% How can we grab what it in a vector? e.g.

v(3) % the third element
v([3,4,5]) % the third through fifth elements
v(3:5) % the third through fifth elements written more compactly
v(4:end) % MATLAB treats 'end' to mean the value of the largest index of a column or row
v([1 1 1 4 3]) % you can provide an arbitrary list of indices!
% here's something you can't do:
v(6)
% a last way uses a vector of logicals. It's hard to overemphasize how
% useful this is!
ex_log_inds = logical([0 1 0 1 0]);
v(ex_log_inds)
% we'll see why this is so powerful soon!

A = magic(3)
A(3,2) % a single element
A(6) % matrices can be indexed like vectors!

A(2,[1,2]) % the first and second elements in the second row
A(1:2,2:end) % the 2x2 submatrix in the upper right
% The colon has another important use in MATLAB:
A(:,1) % all the rows in the first column
A(1,:) % all the columns in the first row
A(:) % all the elements in A 'stretched' out columnwise

%% Exercise: Isolate the Himalayas from MATLAB's topo file
load topo
% look at the matrix
% plot it using pcolor
% use the data cursor to find the NE and SW limits
% NE: x: 211, y: 115
% SW: x: 195, y: 105
% >> himalayas = topo(115:136,67:111);
% >> pcolor(himalayas)

% go back to slide - review ways of accessing elements
% questions?

%% Changing entries in a matrix

v(2) = 0 % ok
% The general rule: the number of entries specified on both sides of the
% equals sign MUST BE THE SAME!

v([1,5]) = [nan,nan] % ok

% this won't work!
v(1) = [nan,nan]; % #(fail)
```

```
% there is one exception: if you're setting multiple values equal to a
% SCALAR

v([3 5]) = pi;

% same lessons for matrices

%% Exercise: Flatten the Himalayas from MATLAB's topo file
load topo
% look at the matrix
% plot it using pcolor
% use the data cursor to find the NE and SW limits
% >> topo(115:136,67:111)=0;
% >> pcolor(himalayas)

% return to slides


%% Relational operators

% Is 3<4?
3<4
3==3
% nan behaves a little weirdly...
nan==nan

% logical indexing by range
A = magic(5)
A > 10
A < 20
A > 10 & A < 20

% A very useful practice:
A(A > 10 & A < 20);

% the find command gives a list of indices of a vector whose elements
% satisfy some condition. to wit:
v = [1 20 5 34 54];
ind = find(v>=20)
v(ind)

%% More with topo
surf(topo)
shading interp
colormap hot
axis off
set(gcf,'color','k')
% What's the average depth of the ocean?

% go to slides for if!
%% if statements

% a trivial one:
if true
    disp('chicken')
end

% useful for household chores:
if randn>0
    disp('Brian cleans the bathroom')
else
    disp('Dan cleans the bathroom')
end
```

```
% for the unscrupulous:
if rand>0 % rand is uniform on [0,1]...
    disp('Brian cleans the bathroom')
else
    disp('Dan cleans the bathroom')
end


% go to slides

%% the while loop
% a canary in a coal mine with a 5% chance of danger
canary_lives=true;
while canary_lives
    disp('keep on mining')
    if rand>0.95
        disp('get out!')
        canary_lives = false;
    end
end
% back to slides
%% for loops

% here we don't use the information about the index:
for ii = 1:5
    disp('chicken')
end

%here we do:
for ii = 1:5
    disp(['chicken' num2str(ii)])
end

%% A common use of for loops is to loop through the indices of a vector:
load carsmall
for ii = 1:length(Model) % this is very common!
    disp([Model(ii,:) ' had mpg of ' num2str(MPG(ii))])
end

%% Final example: How can we compute an arbitrarily long Fibonacci sequence?

N = 20; % number of entries

f = [0 1];

for ii = 2:N
    next_entry = f(ii) + f(ii-1);
    f = [f,next_entry]
end

% Does the ratio of consecutive entries converge?

f(2:end)./f(1:end-1)

% cf ((1 + sqrt(5))/2)

% questions?

%% Nested for loops can traverse matrices

A = magic(5)

for ii = 1:5
    for jj = 1:5
        if A(ii,jj) > 10
```

```
            disp(A(ii,jj)*A(ii,jj))
        end
    end
end

% avoid these when you can! your code will take longer to run and nobody
% will talk to you at parties. Vectorize!

%% Plotting
% just time for one example...
load sunspot.dat
yr = sunspot(:,1)
ac = sunspot(:,2)
plot(yr,ac)
xlabel('year')
ylabel('sunspot activity')
xlim([min(yr),max(yr)])
hold on
plot(randn(1,length(yr)))

% export_fig is very useful!

print('-dpdf','sunspots')
```