1 MATLAB built-in functions... fun with MATLAB

MATLAB provides numerous built-in variables and functions. Below is a collection of commands that can be typed in the *command window* to illustrate some of the built-in features:

```
% This is a comment

% Variables, constants, and simple calculations:
  a= 100                    % Assign the value 100 to variable a.
                            % Note that declaration is NOT necessary
  b= 99
  format compact            % No blank line between lines of output
  a/b                       % Floating point division ALWAYS
  ans                       % Variable to store result of most recent command
  y= ans
  format long               % Display numbers using 15 digits
  y
  format short              % Display numbers using 5 digits (default)
  y
  (3*2)^2
  (3*2)^2;                  % Semicolon suppress output
  x= 2;  y= x^x;  z= y^y    % Semicolon is a separator and suppresses output
  x= 2,  y= x^x,  z= y^y    % Comma is a separator that doesn't suppress output
  format loose              % Insert blank line between lines of output

% Functions:
  sqrt(x)
  pi                        % A built-in variable (constant)
  cos(pi)                   % Trigonometric functions have an argument in radians
  abs(ans)
  abs(cos(pi))
  exp(ans)
  rand(2)                   % Generate a 2-by-2 matrix of random numbers
                            %   Each random no. is in the OPEN interval (0,1)
  mod(5,2)                  % The remainder of 5 divided by 2
  help mod                  % Get documentation of function mod
  lookfor mod               % Look for a function related to mod
```

2 Input & output statements

Input: *variable* = `input('`*prompt*`')`

Output:

`disp('`*words to be displayed*`')`	Strings are enclosed in single quotes.

`x= 1.1;  y= 2;`
`s= sprintf('X is %6.2f, Y is %d\n', x, y);`
`disp(s)`
	Build string `s` using `sprintf`. Then use `disp` to output the string. `%6.2f` is a format sequence specifying that **6** characters will be used for printing a variable value as a floating point number with **2 decimal places**. `%d` is a format sequence specifying that an integer variable value is to be printed.

`x= 1.1;  y= 2;`
`fprintf('X is %6.2f, Y is %d\n', x, y);`
	Use `fprintf` to output the string directly.

3 Script File

A sequence of MATLAB commands stored in a file with the filename extension **.m** is called a script file, sometimes called M-files. When you type the filename *without* the extension in the command window, the commands stored in the file are executed.

4 1-Dimensional Array: Vector

In MATLAB, one dimensional arrays are called *vectors*. MATLAB distinguishes between *row* and *column* vectors. Use *square brackets* to delimit arrays. Using a space or a comma as the separator results in a *row* vector while using a semicolon as the separator results in a *column* vector.
MATLAB **array index starts at 1**, not zero. To access a value in an array, use parentheses to enclose the index value. For example, `x(2)` is the value in the 2nd cell of vector `x`.

```
x= [11 23 9]     % Row vector x
y= [11; 23; 9]   % Column vector y
y= y'            % Transpose y.  Now y is a row vector

y(3)             % 9, the value in cell 3 of vector y.
y(5)             % ERROR!  No such cell in vector y
y(5)= 8          % Put value 8 in cell 5 in vector y.  y is now [11 23 9 0 8]

length(y)        % 5, the length of vector y
[m,n]= size(y)   % m gets the number of rows of y (1);
                 % n gets the number of columns of y (5)

% Create vectors using functions or short-cut expressions:
  a= zeros(1,5)       % 1 row, 5 columns of 0s (a row vector)
  b= ones(5,1)        % 5 rows, 1 column of 1s (a column vector)
  c= rand(1,4)        % 1 row, 4 columns (a row vector) of random numbers
                      %   Each random number is in the OPEN interval (0,1)
  d= 1:4              % [1 2 3 4]
  e= 1:3:11           % [1 4 7 10]  1st expression is starting value
                      %             2nd expression is amount by which to add
                      %             3rd expression is highest possible value
  f= linspace(0,1,5)  % [0, .25 .5 .75. 1]
                      %   Vector of 5 values equally spaced from 0 to 1

% Build vectors by concatenation
  x= ones(1,3)
  y= [2 4]
  z= [x y]             % [1 1 1 2 4]
  z= [z 0]             % [1 1 1 2 4 0]
  z= [9 z z]           % [9 1 1 1 2 4 0 1 1 1 2 4 0]
  c= [zeros(1,2)'; 6]  % a column, same as [0; 0; 6]

% Sub-vectors
  x= [2 5 8 6 8]
  x(1:3)           % [2 5 8], cells 1, 2, and 3 of x as a vector
  x(3:length(x))   % [8 6 8], all cells from x(3) until the last cell, x(5)
  x(3:end)         % [8 6 8]. When used as an index value in an array x,
                   %          end equals length(x)
  x([1,3,4])       % [2 8 6], cells x(1), x(3), and x(4)
```

5 Vectorized Code in 1-D

MATLAB can operate (e.g., perform arithmetic operations) on entire vectors in one step (in one statement). Code that operate on vectors, instead of on scalars, in one statement is said to be *vectorized.*

```
x= [10 20 30]
y= [2  1  2]
z= [2; 3; 2]  % column

% Vectorized addition, subtraction
  x+y          % [12 21 32]
  x-y          % [8 19 28]
  x+5          % [15 25 35]

% Vectorized multiplication, division, power
% Need DOT OPERATOR (.)
  x.*y         % [20 20 60]
  x./y         % [5 10 15]
  x.^y         % [100 20 900]
  x.*5         % [50 100 150]

% Shape is important!
  x+z          % ERROR!  x is a row while z is column
  x+z'         % [12 23 32]
```

Note: MATLAB is built to support a field of mathematics called *linear algebra.* After you learn linear algebra (not in CS100!), you can really harness the power of MATLAB's matrix computation capabilities. In CS100, we will *not* use "matrix multiplication" as defined in linear algebra and coded in MATLAB as `m*n` where `m` and `n` are vectors (or matrices). Rather, we multiply the vectors "cell-by-cell" using the MATLAB code `m.*n`, which means to perform the operation `m(i)*n(i)` for all index i for vectors `m` and `n` (assuming that they have the same shape and length). The result of "dot multiply" is a vector the same shape and length as `m` and `n`, as illustrated in the example above.

6 The `if` Construct

```
if condition1
    statements to execute if condition1 is true
elseif condition2
    statements to execute if condition1 is false but condition2 is true
else
    statements to execute if all previous conditions are false
end
```

7 Relational and Logical Operators

MATLAB uses the value zero (0) to represent *false.* Any value that is *not* zero represents *true.*

Relational Operators		Logical Operators	
`==`	equal to		
`~=`	not equal to	`&&`	logical and
`<`	less than	`||`	logical or
`>`	greater than	`~`	not
`<=`	less than or equal to		
`>=`	greater than or equal to		