MATLAB QUICK TUTORIAL

- Use a semi-colon (;) at the end of each line to suppress output
- For an array `A(m,n)`, `m` is the row, `n` is the column
- Indexes start at 1
- The % symbol starts comments, use them often to easily understand your code
- (. . .) at the end of a line indicates expression continues on next line
- Get quick help with `>>help` *command* or `>>doc` *command*
- Find commands by typing first few letters then press tab

Define a scalar	`>> a = 4;`

Define an array (matrix or vector)	`>> b = [0 1 2 3];  %only columns`
	`>> c = [0;1;2;3];  %only rows`
	`>> d = [0 1; 2 3];  %rows and columns`

Define Ranges	`>> x = 1:5  %fill x with vector ranging from 1 to 5`
	`>> x = 5:-0.5:1  %fill x with increment of -0.5`
	`>> x= linspace(1,5,10) %fill x with 10 equal spaced points`

Basic Operations	`+ - * / ^ .* ./ .^` `dot(.) preceding any operation is done in every element of the array. A^2 = A*A =/= A.^`
Transpose	`'  %takes the transpose of the matrix` `>>b = [0 1 2 3]'  %makes b a row vector`

Define a function	`>> f = @(x)(x.^3 – 4*x);  %arguments followed by function`
	`>> g = inline('x.^3 – 4*x','x')  %same as above`
	`Create M-file (on back)`

Evaluate a function	`>> f(2.8)  %function with arguments in parenthesis`
	`>> feval(g,2.8)  %equivalent to above line`
	`>> myfcn(2.8)`
	`%% private function in M-file##`

`if` structure	`>> if condition`
	`  *statements*  %indent for ease of reading`
	`  else`
	`  *statements*`
	`  end`
	`%condition arguments are:`
	`== equal  ~= not equal`
	`<  less than  <= less than or equal to`
	`>  greater than  >= greater than or equal to`
	`&  and  ~ not  | or`

for structure	`for i = 1:5`
	`  *statements*  %indent for ease of reading`
	`end`
	`%loops over i incremented by 1`

Selecting data	`>> data = [5 10; 3 4; 7 1; 13 4; 20 17; 15 10; 12 11];`
	`>> data(:,1)  %returns entire 1st column`
	`>> data(2,:)  %returns entire 2nd row`
	`>> data(2:4,:)  %returns rows 2-4`
	`>> sub = data(3,:)  %sets 3rd row as new variable`

Plotting	`>> plot(x,y)  >>plot(x,f(x))`

##M-file myfcn.m

```
function var = myfcn(x)
%help comments
var = x.^3 - 4.*x;
```

Note:

`var =` return variable

`myfcn =` function name(saved as myfcn.m)

`x =` function argument

Example:

$$f(x) = \left(\frac{3}{x}\right)^3 - \sin(x)$$

```
>> f = @(x)((3./x).^3-sin(x))
Or
M-file: g.m
      function var = g(x)
      var = (3./x).^3-sin(x)
```

Evaluate functions

```
>> f(2)
>> g(2)
```

These return the same values

Passing functions to M-files

```
M-file: comp.m
  function [R A] = comp(fcn,a,b)
  Ta = feval(fcn,a);
  fcn(b);
  R = Tb - Ta
  A = (Ta+Tb)/2
```

Evaluate

```
>> [R1 A1] = comp(f,1,4)
>> [R2 A2] = comp(g,1,4)
```

Formatting Output

Use fprintf

```
>> m = 12.5
>>fprintf('The value of m is
%f\n',m)
```

or

```
>>display(m)
```

Inserts the value of m where % is located

`f` is the formatting option, there are many others that can dictate number of decimals returned

`\n` starts new line in the window

See `>>help/doc fprintf` for the many, many formatting options

Plot a function

```
>> x = 0:.1:2;
>> f1 = @(x)(x.^2+5);
>> f2 = @(x)(x.^3-3);
>> plot(x,f1(x),--)
>> hold %wont clear current plot
>> plot(x,f2(x),'r*')
>> figure %creates new figure
>> plot(x,x^2+5*x,'go')
```

`figure(#)` swaps active figure for plotting/formatting options

see `>>help/doc plot` for more options in plot formatting and options such as

```
xlabel
ylabel
legend
axes
```

DO NOT leave plots with unlabeled axis or blank legends