* Add ; after the command to not run the line
  + disp(a);
  + 3.1416
  + >> disp(sprintf('2 deciamls: %0.2f', a))
  + 2 deciamls: 3.14
  + >> disp(sprintf('6 decimals: %0.6f', a))
  + 6 decimals: 3.141593
  + >> format long
  + >> a
  + a = 3.14159265358979
  + >> format short
  + >> a
  + a = 3.1416
* A = [1 2; 3 4; 5 6]
  + Displays a 3 by 2 matrix
* V = [1; 2; 3]
  + Order 3 vector
* V = [1 2 3]
  + 1 by 3 matrix
* V = 1:0.1:2
  + 1 by 3 matrix starting from 1, going up to 2, incrementally by 0.1
* Ones(2,3)
  + 2 by 3 matrix of 1
* 2\*ones(2,3)
  + 2 by 3 matrix of 2
* Zeros(1,3)
  + 1 by 3 matrix of 0
* Rand(1,3)
  + 1 by 3 matrix of random numbers between 0 and 1
* Randn(1,3)
  + 1 by 3 matrix of random Gaussian variable
* Hist(w)
  + Plots a histogram of the matrix
* Eye(4)
  + Identity matrix of 4
* Size(A)
  + Size of a matrix
* Length(W)
  + Length of the largest dimension (could work on matrix as well, but mainly used in vectors)
* Load filename.dat
  + Loading files
* Who
  + Shows the variables currently in the program
* Whos
  + Detailed view
* Save filename.mat variableName
  + Saves the variable in a file in a binary format
* Clear
  + Clears all variables
* save hello.txt w -ascii
  + saves the variable in a file in readable format
* A(3,2)
  + Point in matrix
* A(2, : )
  + All elements 2nd row
* A(:, 1)
  + All elements 1st column
* Basically the : mean all
* A(:,2) = [10;11;12]
  + Assigning values to the second column of a matrix
* A = [A, [100; 101; 102]]
  + Append another column vector to the right
* A(:)
  + Put all elements of a in a single column vector
* C = [A B]
  + Concatenates two matrixes together
* A .\* B
  + Takes every element of A and multiplies it with every element of B (like addition, just that it multiplies them)
* The . signifies element-wide opperations
* A’
  + A transposed

>> t = [0: 0.01: 0.98];

>> y1 = sin(2\*pi\*4\*t);

>> plot(t, y1);

>> y2 = cos(2\*pi\*4\*t);

>> plot(t,y2);

>> plot(t, y1);

>> hold on;

>> plot(t,y2, 'r');

>> xlabel('time')

>> ylabel('value')

>> legend('sin', 'cos')

>> title('my plot')

>> print -dpng 'myPlot.png'

>> figure(1); plot(t,y1);

>> figure(2); plot(t,y2);

>> subplot(1,2,1); %divides plot a 1x2 grid, access first element

>> plot(t,y1);

>> subplot(1,2,2);

>> plot(t,y2);

>> axis([0.5 1 -1 1])

>> clf

imagesc(A)

>> imagesc(A), colorbar, colormap gray;

>> a=1, b=2, c=3

v = zeros(10,1)

v =

0

0

0

0

0

0

0

0

0

0

>> for i=1:10,

v(i) = 2^i;

end;

>> v

v =

2

4

8

16

32

64

128

256

512

1024

>> indices=1:10;

>> indices

indices =

1 2 3 4 5 6 7 8 9 10

>> for i=indices,

disp(i);

end;

1

2

3

4

5

6

7

8

9

10

v

v =

2

4

8

16

32

64

128

256

512

1024

>> i = 1;

>> while i <= 5,

Display all 1783 possibilities? (y or n)

v(i) = 100;

i = i + 1;

end;

>> v

v =

100

100

100

100

100

64

128

256

512

1024

i = 1;

>> while true,

v(i) = 999;

i = i+1;

if i == 6,

break;

end;

end;

>> v

v =

999

999

999

999

999

64

128

256

512

1024

>> v(1)

ans = 999

>> v(1) = 2

v =

2

999

999

999

999

64

128

256

512

1024

>> if v(1) == 1,

disp('The value is one');

elseif v(1) == 2,

disp('The value is two');

else

disp('The value is other);

parse error:

syntax error

>>> disp('The value is other);

^

>> whatevah

squareThisNumber(4)

ans = 16

>> % Octave search path (advanced/optional)

>> addpath('C:\Users\ionutiliescu\Desktop')

>> cd C:\

>> pwd

ans = C:\

>> squareThisNumber(4)

squareAndCubeThisNumber(3)

ans = 9

>> [a,b] = squareAndCubeThisNumber(5);

>> [a,b] = squareAndCubeThisNumber(5)

a = 25

b = 125

>> X = [1 1; 1 2; 1 3]

X =

1 1

1 2

1 3

>> Y = [1; 2; 3]

Y =

1

2

3

>> theta = [0;1]

theta =

0

1

j = costFunctionJ(X,Y, theta)

j = 0

j = costFunctionJ(X,Y, theta)

j = 2.3333

>> (1^2 + 2^2 + 3^2)/ (2\*m)

error: 'm' undefined near line 1 column 23

>> (1^2 + 2^2 + 3^2)/ (2\*3)

ans = 2.3333

Vectorization