**MATLAB/Octave - Computing on data**

**#create matrix A for later use**

>> A = [1 2; 3 4; 5 6]

A =

1 2

3 4

5 6

**#create matrix B for later use**

>> B = [11 12; 13 14; 15 16]

B =

11 12

13 14

15 16

**#create matrix C for later use**

>> C = [1 1; 2 2]

C =

1 1

2 2

**#multiply 2 matrix A and C, number of columns of first matrix should match with number of rows of second matrix**

>> A \* C

ans =

5 5

11 11

17 17

**#unsuccessful multiplication of 2 matrix A and B as above condition didn't meet**

>> A \* B

Error using \*

Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To perform elementwise multiplication, use '.\*'.

**#element wise multiplication of 2 matrix, here both the number of rows and columns should match between both the matrices**

>> A .\* B

ans =

11 24

39 56

75 96

**#element wise square**

>> A .^ 2

ans =

1 4

9 16

25 36

**#create vector V for later use**

>> V = [ 1; 2; 3]

V =

1

2

3

**#element wise division**

>> 1 ./ V

ans =

1.0000

0.5000

0.3333

**#log of the vector V**

>> log(V)

ans =

0

0.6931

1.0986

**#exp of the vector V**

>> exp(V)

ans =

2.7183

7.3891

20.0855

**#absolute values of random vector containing negative integers**

>> abs([-1; 2; -3])

ans =

1

2

3

**#multiply every element of vector V with -1**

>> -1 \* V

ans =

-1

-2

-3

**#same as above but with short cut approach**

>> -V

ans =

-1

-2

-3

**#add ones to every element of vector V**

>> V + ones(length(V),1)

ans =

2

3

4

**#same as above bur with short cut approach**

>> V + 1

ans =

2

3

4

**#print matrix A**

>> A

A =

1 2

3 4

5 6

**#transpose of A**

>> A'

ans =

1 3 5

2 4 6

**#double transpose of A gives back A only as per the transpose property**

>> (A')'

ans =

1 2

3 4

5 6

**#single row vector**

>> a = [1 15 2 0.5]

a =

1.0000 15.0000 2.0000 0.5000

**#find maximum value from any vector**

>> val = max(a)

val =

15

**#fetch maximum value from a vector with its index location**

>> [val, ind] = max(a)

val =

15

ind =

2

**#print matrix A**

>> A

A =

1 2

3 4

5 6

**#find maximum element from every column of matrix A**

>> max(A)

ans =

5 6

**#print row vector 'a'**

>> a

a =

1.0000 15.0000 2.0000 0.5000

**#element wise comparison, return 1 as true and 0 as false based on the given condition**

>> a < 3

ans =

1×4 logical array

1 0 1 1

**#return elements which satisfy the given condition**

>> find(a<3)

ans =

1 3 4

**#magic matrix, the sum of every row, column, diagonals are same**

>> A = magic(3)

A =

8 1 6

3 5 7

4 9 2

**#find row and column location of the element of a matrix which satisfy the given condition**

>> [r, c] = find(A >= 7)

r =

1

3

2

c =

1

2

3

**#fetch specific element of a matrix**

>> A(2,3)

ans =

7

**#print row vector 'a'**

>> a

a =

1.0000 15.0000 2.0000 0.5000

**#sum of all the elements of vector 'a'**

>> sum(a)

ans =

18.5000

**#prod of all the elements of vector 'a'**

>> prod(a)

ans =

15

**#get floor values of all the elements of vetor/matrix**

>> floor(a)

ans =

1 15 2 0

**#get ceil values of all the elements of vetor/matrix**

>> ceil(a)

ans =

1 15 2 1

**#fetch element wise maximum of 2 random matries**

>> max(rand(3), rand(3))

ans =

0.7458 0.6173 0.8021

0.8131 0.5755 0.9891

0.9861 0.5301 0.4516

**#print matrix A**

>> A

A =

8 1 6

3 5 7

4 9 2

**#column wise maximum values for a given matrix**

>> max(A, [], 1)

ans =

8 9 7

**#row wise maximum values for a given matrix**

>> max(A, [], 2)

ans =

8

7

9

**#column wise maximum values for a given matrix if no extra parameter given (default behaviour)**

>> max(A)

ans =

8 9 7

**#maximum element from the whole matrix**

>> max(max(A))

ans =

9

**#same as above with different approach**

>> max(A(:))

ans =

9

**#magic matrix**

>> A = magic(9)

A =

47 58 69 80 1 12 23 34 45

57 68 79 9 11 22 33 44 46

67 78 8 10 21 32 43 54 56

77 7 18 20 31 42 53 55 66

6 17 19 30 41 52 63 65 76

16 27 29 40 51 62 64 75 5

26 28 39 50 61 72 74 4 15

36 38 49 60 71 73 3 14 25

37 48 59 70 81 2 13 24 35

**#column wise sum**

>> sum(A)

ans =

369 369 369 369 369 369 369 369 369

**#column wise sum**

>> sum(A,1)

ans =

369 369 369 369 369 369 369 369 369

**#row wise sum**

>> sum(A,2)

ans =

369

369

369

369

369

369

369

369

369

**#print matrix A**

>> A

A =

47 58 69 80 1 12 23 34 45

57 68 79 9 11 22 33 44 46

67 78 8 10 21 32 43 54 56

77 7 18 20 31 42 53 55 66

6 17 19 30 41 52 63 65 76

16 27 29 40 51 62 64 75 5

26 28 39 50 61 72 74 4 15

36 38 49 60 71 73 3 14 25

37 48 59 70 81 2 13 24 35

**#fetch diagonal elements of any matrix using identity (eye) matrix**

>> A .\* eye(9)

ans =

47 0 0 0 0 0 0 0 0

0 68 0 0 0 0 0 0 0

0 0 8 0 0 0 0 0 0

0 0 0 20 0 0 0 0 0

0 0 0 0 41 0 0 0 0

0 0 0 0 0 62 0 0 0

0 0 0 0 0 0 74 0 0

0 0 0 0 0 0 0 14 0

0 0 0 0 0 0 0 0 35

**#sum of diagonal elements on a matrix**

>> sum(sum(A .\* eye(9)))

ans =

369

**#flip upside down any matrix, in this case eye matrix has been fliped upside down**

>> flipud(eye(9))

ans =

0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 1 0 0

0 0 0 0 0 1 0 0 0

0 0 0 0 1 0 0 0 0

0 0 0 1 0 0 0 0 0

0 0 1 0 0 0 0 0 0

0 1 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0

**#magic matrix**

>> A = magic(3)

A =

8 1 6

3 5 7

4 9 2

**#pseudo inverse of matrix A**

>> temp = pinv(A)

temp =

0.1472 -0.1444 0.0639

-0.0611 0.0222 0.1056

-0.0194 0.1889 -0.1028

**#multiply matrix A and pseudo inverse of A gives back identity matrix based on the matrix property**

>> temp \* A

ans =

1.0000 0.0000 -0.0000

-0.0000 1.0000 0.0000

0.0000 -0.0000 1.0000