ASSIGNMENT – 4

Q1. Write python code to display the matrix whose all entries are 10 and order is(4,6)

```
>>> from sympy import *
>>> A=ones(4,6)
>>> print(A*10)
Matrix([[10, 10, 10, 10, 10, 10], [10, 10, 10, 10, 10, 10], [10, 10, 10, 10, 10, 10], [10, 10, 10, 10, 10, 10]])
```

Q2. Using python code construct the following matrices. i.)An identity matrix of order 10x10 ii.)Zero matrix of order 7x3. iii.)Identity matrix of order 5x4.

```
>>> from sympy import *
>>> A=eye(10,10)
>>> print(A)
Matrix([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]])
>>> B=zeros(7,3)
>>> print(B)
Matrix([[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]])
>>> C=ones(5,4)
>>> print(C)
Matrix([[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]])
```

Q3. Using python code construct the following matrices. i.) Matrix of order 5x6 with all entries 1 ii.)Zero matrix of order 27x33. iii.)Identity matrix of order 5.

```
>>> from sympy import *
>>> A=ones(5,6)
>>> print(A)
Matrix([[1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1]])
>>> B=zeros(27,33)
>>> print(B)
Matrix([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0]])
```

```
>>> C=eye(5)

>>> print(C)

Matrix([[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 1, 0], [0, 0, 0, 0, 1]])
```

Q4. Using sympy module of python, find the following terms of vectors x=[1,-5,0] and y=[2,3,-1]. i.)5x ii.)x+y iii.) x-3y

```
>>> from sympy import *

>>> x=Matrix([[1],[-5],[0]])

>>> y=Matrix([[2],[3],[-1]])

>>> x+y

Matrix([

[ 3],

[-2],

[-1]])

 >>> 5*x

Matrix([

[  5],

[-25],

[  0]])

>>> x-3*y

Matrix([

[ -5],

[-14],

[  3]])
```

Q5. Using sympy module of python, find the eigenvalues and eigenvectors i.)of matrix A=4 2 2

2 4 2

2 2 4

ii.) B= 2 5

-1 4

```
>>> from sympy import *
>>> A=Matrix([[4,2,2],[2,4,2],[2,2,4]])
>>> B=Matrix([[3,-2],[6,-4]])
>>> A.eigenvals()
{8: 1, 2: 2}
>>> B.eigenvals()
{-1: 1, 0: 1}
>>> A.eigenvects()
[(2, 2, [Matrix([
[-1],
[ 1],
[ 0]]), Matrix([
[-1],
[ 0],
[ 1]])]), (8, 1, [Matrix([
[1],
[1],
[1]])])]
>>> B.eigenvects()
[(-1, 1, [Matrix([
[1/2],
[  1]])]), (0, 1, [Matrix([
[2/3],
[  1]])])]
```

Q6. Write python program to find the determinant of matrices i.) A=(1 ,0 ,5) ,(2,1,6),(3,4,0) ii.)B=(9,0,3),(1,4,1),(1,0,-1).

```
>>> from sympy import *
>>> A=Matrix([[1 ,0 ,5] ,[2,1,6],[3,4,0]])
>>> B=Matrix([[9,0,3],[1,4,1],[1,0,-1]])
>>> A.det()
```

1

>>> B.det()

-48

Q7. Using sympy module of python, find the following for matrices. A=(-1,1,0),(8,5,2),(2,-6,2) B=(9,0,3),(1,4,1),(1,0,-1). i.)2A+B ii.)3A-5B iii.)A-1 iv.)B**3 v.) At+Bt

```
>>> from sympy import *
>>> A=Matrix([[-1,1,0],[8,5,2],[2,-6,2]])
>>> B=Matrix([[9,0,3],[1,4,1],[1,0,-1]])
>>> 2*A+B
Matrix([
[ 7,   2, 3],
[17,  14, 5],
[ 5, -12, 3]])
>>> 3*A-5*B
Matrix([
[-48,   3, -15],
[ 19,  -5,   1],
[  1, -18,  11]])
>>> A.inv()
Matrix([
[-11/17, 1/17, -1/17],
[  6/17, 1/17, -1/17],
[ 29/17, 2/17, 13/34]])
>>> B**3
Matrix([
[780,  0, 228],
[148, 64,  52],
[ 76,  0,  20]])
>>> A.T+B.T
Matrix([
[8, 9,  3],
[1, 9, -6],
[3, 3,  1]]
```

Q8. Write python code to find the eigenvalues and eigenvectors of the matrix a.) A=(1,3,3),(2,2,3),(4,2,1) b.)B=(3,-2),(6,-4)

```
>>> from sympy import *

>>> A=Matrix([[1,3,3],[2,2,3],[4,2,1]])

>>> B=Matrix([[3,-2],[6,-4]])

>>> A.eigenvals()

{7: 1, -1: 1, -2: 1}

>>> B.eigenvals()

{-1: 1, 0: 1}

>>> A.eigenvects()

[(-2, 1, [Matrix([

[-1/2],

[-1/2],

[   1]])]), (-1, 1, [Matrix([

[ 0],

[-1],

[ 1]])]), (7, 1, [Matrix([

[1],

[1],

[1]])])]

>>> B.eigenvects()

[(-1, 1, [Matrix([

[1/2],

[  1]])]), (0, 1, [Matrix([

[2/3],

[  1]])])]
```

Q9. Using python code construct identity matrix of order 10 and hence find determinant, trace and transpose of it

```
>>> from sympy import *

>>> A=eye(10)

>>> print(A)

Matrix([[1, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 1, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 1, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
```

```
>>> A.det()
1
>>> A.trace()
10
>>> A.T
Matrix([
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]])
```

Q10. Using python code , find determinant and inverse of the matrix if exist. A=(4,2,2),(2,4,2),(2,2,4)

```
>>> from sympy import *
>>> A=Matrix([[4,2,2],[2,4,2],[2,2,4]])
>>> A.det()
32
>>> A.inv()
Matrix([
[ 3/8, -1/8, -1/8],
[-1/8,  3/8, -1/8],
[-1/8, -1/8,  3/8]])
```

Q11. Write python code to verify (AB)-1=B-1A.

```
>>> from sympy import *
>>> Q=Matrix([[2,3],[1,4]])
>>> W=Matrix([[5,6],[7,8]])
>>> E=Q*W
>>> E.inv()
Matrix([
```

```
[-19/5,   18/5],
[33/10, -31/10]])
>>> R=W.inv()
>>> Y=Q.inv()
>>> R*Y
Matrix([
[-19/5,   18/5],
[33/10, -31/10]])
```

Q12. Use linsolve command in python to solve the following system of linear equations. X-2y+3z=7,2x+y+z=4,-3x+2y-2z=-10.

```
>>> from sympy import*
>>> x,y,z=symbols("x,y,z")
>>> A=Matrix([[1,-2,3],[2,1,1],[-3,2,-2]])
>>> B=Matrix([[7],[4],[-10]])
>>> linsolve((A,B),[x,y,z])
{(2, -1, 1)}
```

Q13. For matrix A=(1,0,5,4),(2,1,6,-1),(3,4,0,2) apply the following using python i.)Delete 2nd row. Ii.)Delete 1st column. Iii.)Add column [9,9]as 2nd column.

```
>>> A=Matrix([[1,0,5,4],[2,1,6,-1],[3,4,0,2]])
>>> A.row_del(2)
>>> A
Matrix([
[1, 0, 5,  4],
[2, 1, 6, -1]])
>>> A.col_del(0)
>>> A
Matrix([
[0, 5,  4],
[1, 6, -1]])
```

Q14. Declare the matrix A=(5,2,5,4),(10,3,4,6),(2,0,-1,11) find a row echelon form and rank of matrix A

```
>>> A=Matrix([[5,2,5,4],[10,3,4,6],[2,0,-1,11]])
>>> A.rank()
3
>>> A.rref()
```

(Matrix([

[1, 0, 0,   77/9],

[0, 1, 0, -104/3],

[0, 0, 1,   55/9]]), (0, 1, 2))

Q16. Using python solve the following system of equation using LU-factorization method. 3x-7y-2z=-7,-3x+5y+z=5,6x-4y=2.

>>> from numpy import *

>>> from sympy import *

>>> from sympy.abc import x,y,z

>>> AB=Matrix([[3,-7,-2,-7],[[-3,5,1,5],[6,4,0,1]])

>>> AB=Matrix([[3,-7,-2,-7],[-3,5,1,5],[6,4,0,1]])

>>> solve_linear_system_LU(AB,[x,y,z])

{x: -3/10, y: 7/10, z: 3/5}

Q17. Using python solve the following system of equation using gauss elimination method. x+y+2z=-7,-x+2y+3z=6,3x-7y+6z=1.

>>> from sympy import *

>>> x,y,z=symbols("x,y,z")

>>> A=Matrix([[1,1,2],[-1,-2,3],[3,-7,6]])

>>> B=Matrix([[7,6,1]])

>>> linsolve((A,B),[x,y,z])

{(-1, 2, 3)}

Q18. Using python accept the matrix A=(1,-3,2,4),(-3,9,-1,5),(5,-2,6,-3),(-4,12,2,7) . find null space, column space and rank of the matrix.

>>> from sympy import *

>>> A=Matrix([[1,-3,2,4],[-3,9,-1,5],[5,-2,6,-3],[-4,12,2,7]])

>>> A.nullspace()

[]

>>> A.columnspace()

[Matrix([

[1],

[-3],

[5],

[-4]]), Matrix([

[-3],

[9],

[-2],

[12]]), Matrix([

[2],

[-1],

[6],

[2]]), Matrix([

[4],

[5],

[-3],

[7]])]

>>> A.rank()

4

Q19. Using python accept the matrix A=(1,2,3),(2,5,3),(1,0,8).Find the transpose ,determinant , inverse ,of the matrix and also reduce the matrix to row reduce echelon form and daigonalize it

>>> from sympy import *

>>> A=Matrix([[1,2,3],[2,5,3],[1,0,8]])

>>> A.T

Matrix([

[1, 2, 1],

[2, 5, 0],

[3, 3, 8]])

>>> A.det()

-1

>>> A.inv()

Matrix([

[-40, 16, 9],

[13, -5, -3],

[5, -2, -1]])

>>> A.rref()

(Matrix([

[1, 0, 0],

[0, 1, 0],

```
[0, 0, 1]]), (0, 1, 2))

>>> A.diagonalize()

(Matrix([

[
(464*2**(1/3) + (-40 + 2**(2/3)*(1 + sqrt(3)*I)*(335 + 3*sqrt(74247)*I)**(1/3))*(1 + sqrt(3)*I)*(335 +
3*sqrt(74247)*I)**(1/3))/(12*(1 + sqrt(3)*I)*(335 + 3*sqrt(74247)*I)**(1/3)),
(464*2**(1/3) + (-40 + 2**(2/3)*(1 - sqrt(3)*I)*(335 + 3*sqrt(74247)*I)**(1/3))*(1 - sqrt(3)*I)*(335 +
3*sqrt(74247)*I)**(1/3))/(12*(1 - sqrt(3)*I)*(335 + 3*sqrt(74247)*I)**(1/3)),
-10/3 - 2**(2/3)*(335 + 3*sqrt(74247)*I)**(1/3)/6 - 58*2**(1/3)/(3*(335 + 3*sqrt(74247)*I)**(1/3))],

[(-50112*2**(1/3)*(1 + sqrt(3)*I)*(335 + 3*sqrt(74247)*I)**(2/3) + (335 +
3*sqrt(74247)*I)**(1/3)*(464*2**(1/3) + (1 + sqrt(3)*I)*(56 + 2**(2/3)*(1 + sqrt(3)*I)*(335 +
3*sqrt(74247)*I)**(1/3))*(335 + 3*sqrt(74247)*I)**(1/3))**2 - 36*(1 + sqrt(3)*I)**2*(148 + 3*2**(2/3)*(1 +
sqrt(3)*I)*(335 + 3*sqrt(74247)*I)**(1/3))*(335 + 3*sqrt(74247)*I))/(288*(1 + sqrt(3)*I)**2*(335 +
3*sqrt(74247)*I)), ((335 + 3*sqrt(74247)*I)**(1/3)*(464*2**(1/3) + (1 - sqrt(3)*I)*(56 + 2**(2/3)*(1 -
sqrt(3)*I)*(335 + 3*sqrt(74247)*I)**(1/3))*(335 + 3*sqrt(74247)*I)**(1/3))**2 - 50112*2**(1/3)*(1 -
sqrt(3)*I)*(335 + 3*sqrt(74247)*I)**(2/3) + 36*(-148 + 3*2**(2/3)*(-1 + sqrt(3)*I)*(335 +
3*sqrt(74247)*I)**(1/3))*(1 - sqrt(3)*I)**2*(335 + 3*sqrt(74247)*I))/(288*(1 - sqrt(3)*I)**2*(335 +
3*sqrt(74247)*I)), (-42*sqrt(74247) - 2131*2**(2/3)*I*(335 + 3*sqrt(74247)*I)**(1/3) - 73*2**(1/3)*I*(335 +
3*sqrt(74247)*I)**(2/3) - 2**(2/3)*sqrt(74247)*(335 + 3*sqrt(74247)*I)**(1/3) + 4690*I +
2**(1/3)*sqrt(74247)*(335 + 3*sqrt(74247)*I)**(2/3))/(12*(3*sqrt(74247) - 335*I))],

[
1,
1,
1]]), Matrix([

[14/3 - 58/(3*(-1/2 - sqrt(3)*I/2)*(335/2 + 3*sqrt(74247)*I/2)**(1/3)) - (-1/2 - sqrt(3)*I/2)*(335/2 +
3*sqrt(74247)*I/2)**(1/3)/3,                                                                    0,
0],

[                                                            0, 14/3 - (-1/2 + sqrt(3)*I/2)*(335/2 +
3*sqrt(74247)*I/2)**(1/3)/3 - 58/(3*(-1/2 + sqrt(3)*I/2)*(335/2 + 3*sqrt(74247)*I/2)**(1/3)),
0],

[                                                            0,
0, 14/3 - (335/2 + 3*sqrt(74247)*I/2)**(1/3)/3 - 58/(3*(335/2 + 3*sqrt(74247)*I/2)**(1/3))]]))
```

Q20. Write the python code to perform the R2+2R1 row operation on the given matrix A=(1,1,1),(2,2,2),(3,3,3).

```
>>> from sympy import *

>>> from numpy import*

>>> A=Matrix([[1,1,1],[2,2,2],[3,3,3]])

>>> A[1,:] += 2*A[0,:]

>>> print(A)

Matrix([[1, 1, 1], [4, 4, 4], [3, 3, 3]])
```