In [ ]:
```
from main import *
```

Assignment 1

In [ ]:
```
# a
# m = 7 and n = 6. Here is an example:

A = Matrix([[1,0,0,0,0,0],[0,1,0,0,0,0],[0,0,1,0,0,0],[0,0,0,1,0,0],[0,0,0,0,1,0
b = Matrix([[0],[0],[0],[0],[0],[0],[1]])
A, b
```

Out[ ]:
$$\left(\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}\right)$$

In [ ]:
```
# b
```

The statement is false. While $\{x, y\}$ and $\{u, v\}$ are both linearly independent sets, it doesn't guarantee that the combined set $\{x, y, u, v\}$ is linearly independent.

In $\mathbb{R}^4$, the maximum number of linearly independent vectors you can have is 4 . So, it's possible for $\{x, y, u, v\}$ to be linearly independent, but it's not a certainty. For a counterexample, consider:

$$x = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$u = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad v = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Both $\{x, y\}$ and $\{u, v\}$ are linearly independent sets in $\mathbb{R}^4$. However, the combined set $\{x, y, u, v\}$ is not linearly independent because v can be expressed as a linear combination of x, y, and u :

$$v = x + y + u$$

So, even in $\mathbb{R}^4$, the statement " $\{x, y, u, v\}$ is always linearly independent" is false.

No, not all invertible matrices are diagonalizable. A matrix is diagonalizable if and only if it has enough linearly independent eigenvectors to form a basis for the space it acts on. This means that for an $n \times n$ matrix to be diagonalizable, it must have n linearly independent eigenvectors. Being invertible (having a non-zero determinant) does not

guarantee that a matrix will have n linearly independent eigenvectors. For example, consider the matrix:

$$A = \begin{bmatrix} 2 & 1 \\ 0 & 2 \end{bmatrix}$$

This matrix is invertible because its determinant is non-zero. However, it only has one distinct eigenvalue, $\lambda = 2$, with algebraic multiplicity 2 . The geometric multiplicity of this eigenvalue (i.e., the number of linearly independent eigenvectors associated with this eigenvalue) is 1 , which is less than its algebraic multiplicity. Thus, the matrix A is not diagonalizable. So, while many invertible matrices are diagonalizable, it's not a guarantee.

The statement is true. By definition, a basis for a subspace W is a set of linearly independent vectors that spans W. If W is a subspace of $\mathbf{R}^6$ with dimension 4 , then any set of 4 linearly independent vectors in W will span W and thus constitute a basis for W . So, if you have 4 linearly independent vectors in W, they will indeed form a basis for W.

In [ ]:
```
# c: Determinant is zero since row 3 is a multiple of row 1.
A = Matrix([[2,8,3,6,7],[1,2,4,5,5],[4,16,6,12,14],[5,7,8,9,0],[3,4,5,7,1]])
A.det()
```

Out[ ]: 0

In [ ]:
```
# d Roots are -3, and 2. since 2 is a global minimum, it has multiplicity 2 (sin
```

In [ ]:
```
A = Matrix([[2,8,3,6,7],[1,2,4,5,5],[4,16,6,12,14],[5,7,8,9,0],[3,4,5,7,1]])
display(A.rref())
A.columnspace()
```

$$\left(\begin{bmatrix} 1 & 0 & 0 & 0 & -\frac{387}{92} \\ 0 & 1 & 0 & 0 & \frac{25}{23} \\ 0 & 0 & 1 & 0 & \frac{22}{23} \\ 0 & 0 & 0 & 1 & \frac{59}{92} \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \ (0, 1, 2, 3) \right)$$

Out[ ]: $$\left[\begin{bmatrix} 2 \\ 1 \\ 4 \\ 5 \\ 3 \end{bmatrix}, \ \begin{bmatrix} 8 \\ 2 \\ 16 \\ 7 \\ 4 \end{bmatrix}, \ \begin{bmatrix} 3 \\ 4 \\ 6 \\ 8 \\ 5 \end{bmatrix}, \ \begin{bmatrix} 6 \\ 5 \\ 12 \\ 9 \\ 7 \end{bmatrix} \right]$$

In [ ]:
```
At = A.T
display(At.rref())
display(At.nullspace())
```

$$\left(\begin{bmatrix} 1 & 0 & 2 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \ (0, 1, 3, 4) \right)$$

$$\begin{bmatrix}\begin{bmatrix}-2\\0\\1\\0\\0\end{bmatrix}\end{bmatrix}$$

In [ ]:
```
# e
# dim col A = number of non-zero singular values, 1 (same as rank)
# dim nul A = n - rank, n is size of right singular vector so 5-1 = 4
# dim Nul A^T is the number of free variables in A^T - this will be 3 - 1 = 2
# dim Col A^T is the row space, same dimension as Col A = 1
```

Assignment 2

In [ ]:
```
A = Matrix([[1, 1], [4, 1]])
(Matrix.hstack(A.eigenvects()[0][2][0], A.eigenvects()[1][2][0], Matrix([15, -10
```

Out[ ]:
$$\left(\left(\begin{bmatrix}1 & 0 & -20\\0 & 1 & 10\end{bmatrix}, (0, 1)\right), \left[\left(-1, 1, \left[\begin{bmatrix}-\frac{1}{2}\\1\end{bmatrix}\right]\right), \left(3, 1, \left[\begin{bmatrix}\frac{1}{2}\\1\end{bmatrix}\right]\right)\right]\right)$$

Assignment 3

In [ ]:
```
A = 4
B = 8
n = 3

R((A*B)**-1), R(1/8)**n * A**3 * B**2, A*B, 4**n * R(A**-2), 3**3*2**3*4*8
```

Out[ ]:
$$\left(\frac{1}{32}, 8, 32, 4, 6912\right)$$

Assignment 4

In [ ]:
```
# A
A_aug = Matrix([[1,2,2,1,14],[-1,-2,0,-3,-4],[2,4,8,-2,48]])
A_aug
```

Out[ ]:
$$\begin{bmatrix}1 & 2 & 2 & 1 & 14\\-1 & -2 & 0 & -3 & -4\\2 & 4 & 8 & -2 & 48\end{bmatrix}$$

In [ ]:
```
# b)
pivot1 = A_aug.rref()[1][0] + 1
pivot2 = A_aug.rref()[1][1] + 1

pivot1, pivot2, 2, 4, A_aug.rank()
```

Out[ ]:
$$(1, 3, 2, 4, 2)$$

In [ ]:
```
# c From RREF we can see that x1 will be 4 and x3 will be 4 when x2 and x4 are 0
```

In [ ]:
```
# d
# Remember we are looking for the null space, so the first vector will be all ze
A = A_aug[:, 0:-1]
A.nullspace()
```

Out[ ]:
$$\left[\begin{bmatrix}-2\\1\\0\\0\end{bmatrix}, \begin{bmatrix}-3\\0\\1\\1\end{bmatrix}\right]$$

Assignment 5

In [ ]:
```
# So you would get the result by cofactor expanding on any row/column
x, y, z = symbols('x y z')
Matrix([[1,2,x],[2,1,y],[3,5,z]]).det()
```

Out[ ]:
$$7x + y - 3z$$

Assignment 6

In [ ]:
```
# a: Notice that v3 is just 2*v1 + v2 so v3 = 2v1 + 1v2
```

In [ ]:
```
# L1 is dependent since it has 5 vectors but spans R^4 so one will be dependent
# L2 is independent
# L3 is dependent since row 2 and 4 are the same and will produce a free variabl
# L4 includes L3 so is also dependent
```

In [ ]:
```
# c So you check by multiplying each vector with A
A = Matrix([[1,1,0],[1,4,3],[0,3,1]])
x = Matrix([1, -1, 3])
y = Matrix([1, 5, 3])
z = Matrix([5, 0, 1])
w = Matrix([[-3],[0],[1]])
display(A*x, A*y, A*z, A*w)
```

$$\begin{bmatrix}0\\6\\0\end{bmatrix}$$

$$\begin{bmatrix}6\\30\\18\end{bmatrix}$$

$$\begin{bmatrix}5\\8\\1\end{bmatrix}$$

$$\begin{bmatrix}-3\\0\\1\end{bmatrix}$$

In [ ]:
```
# We see that y and w are eigenvectors with eigvals 6 and 1, respectively.
```

In [ ]:
```
# d
# Since the Matrix is symmetric, the trace is the sum of the eigen values. The t
# on the main diagonal. The is 1 + 4 + 1 = 6. We have 1 + 6, so we can deduce th
```

Assignment 7

In [ ]:
```
# Find the null space of A^T
v1 = Matrix([[1],[3],[-1],[1]])
v2 = Matrix([[1],[4],[0],[2]])
A = Matrix.hstack(v1, v2)
B = A.T

display(A.T)

B.nullspace()
```

$$\begin{bmatrix} 1 & 3 & -1 & 1 \\ 1 & 4 & 0 & 2 \end{bmatrix}$$

Out[ ]:
$$\left[\begin{bmatrix} 4 \\ -1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \\ 0 \\ 1 \end{bmatrix}\right]$$

In [ ]:
```
B[1,:] = B[1,:] - B[0,:]
B
```

Out[ ]:
$$\begin{bmatrix} 1 & 3 & -1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

In [ ]:
```
B[0,:] = B[0,:] - 3*B[1,:]
B
```

Out[ ]:
$$\begin{bmatrix} 1 & 0 & -4 & -2 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Part 2

Assignment 1

In [ ]:
```
# a
A = Matrix([2, -1, -2])
v1 = (A.T).nullspace()[0]
v2 = (A.T).nullspace()[1]
p = Matrix([1, 1, 2])
proj = p.project(v1) + p.project(v2)
proj = proj.normalized()
v1, v2, p
```

Out[ ]:
$$\left(\begin{bmatrix} \frac{1}{2} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}\right)$$

In [ ]:
```
u1, u2 = GramSchmidt((2*v1, v2))
u1, u2
```

Out[ ]:
$$\left(\begin{bmatrix}1\\2\\0\end{bmatrix}, \begin{bmatrix}\frac{4}{5}\\-\frac{2}{5}\\1\end{bmatrix}\right)$$

In [ ]:
```
p.project(u1)+p.project(u2)
```

Out[ ]:
$$\begin{bmatrix}\frac{5}{3}\\\frac{2}{3}\\\frac{4}{3}\end{bmatrix}$$

In [ ]:
```
# You could also solve it using least squares
X = Matrix.hstack(v1, v2)
X
```

Out[ ]:
$$\begin{bmatrix}\frac{1}{2} & 1\\1 & 0\\0 & 1\end{bmatrix}$$

In [ ]:
```
XtX = X.T * X
Xtp = X.T * p

b = Matrix.hstack(XtX, Xtp).rref()[0][:,-1]
X*b
```

Out[ ]:
$$\begin{bmatrix}\frac{5}{3}\\\frac{2}{3}\\\frac{4}{3}\end{bmatrix}$$

In [ ]:
```
# b
q = Matrix([3,1,0])
XtX = X.T * X
Xtq = X.T * q

b = Matrix.hstack(XtX, Xtq).rref()[0][:,-1]
(q-X*b).norm()
```

Out[ ]:
$$\frac{5}{3}$$

In [ ]:
```
# same using projection
(q-(q.project(u1)+q.project(u2))).norm()
```

Out[ ]:
$$\frac{5}{3}$$

In [ ]:
```
# c
Matrix.vstack(u1.T, u2.T).nullspace()[0].normalized() * 3
```

Out[ ]:
$$\begin{bmatrix} -2 \\ 1 \\ 2 \end{bmatrix}$$

Assignment 2

In [ ]:
```
# We translate the problem to a matrix problem

A = Matrix([[-R(8,200), R(3,100)],[R(8,200), -R(8,100)]])
display(Math(r'A = ' + latex(A)))
```

$$A = \begin{bmatrix} -\frac{1}{25} & \frac{3}{100} \\ \frac{1}{25} & -\frac{2}{25} \end{bmatrix}$$

In [ ]:
```
# We find the eigenvalues and the corresponding eigenspaces.
l = symbols('l')
l1, l2 = solve(det(A-l*eye(np.shape(A)[0])))
display(Math(r'\lambda_1 =' +latex(l1) + r'\approx' + latex(round(l1, 2))))
display(Math(r'\lambda_2 =' +latex(l2) + r'\approx' + latex(round(l2, 2))))

v1 = (A-l1*eye(np.shape(A)[0])).nullspace()[0]
v2 = (A-l2*eye(np.shape(A)[0])).nullspace()[0]
display(Math(r'v_1 = '+ latex(v1) + r'=' + latex(v1.evalf(4))))
display(Math(r'v_2 = '+ latex(v2) + r'=' + latex(v2.evalf(4))))
```

$$\lambda_1 = -\frac{1}{10} \approx -0.1$$

$$\lambda_2 = -\frac{1}{50} \approx -0.02$$

$$v_1 = \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 1.0 \end{bmatrix}$$

$$v_2 = \begin{bmatrix} \frac{3}{2} \\ 1 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 1.0 \end{bmatrix}$$

In [ ]:
```
y0 = Matrix([100, 0])
c1 = Matrix.hstack(v1, v2, y0).rref()[0][0,-1]
c2 = Matrix.hstack(v1, v2, y0).rref()[0][1,-1]
c1, c2
```

Out[ ]:
$$(-50,\ 50)$$

In [ ]:
```
# a
def y(t): return c1*v1*exp(t*l1) + c2*v2*exp(t*l2)
y(25).evalf(10)
```

Out[ ]:
$$\begin{bmatrix} 47.54192444 \\ 26.22228305 \end{bmatrix}$$

In [ ]:
```
import math
t = symbols('t')
y1 = c1*v1[0]*math.e**(l1*t) + c2*v2[0]*math.e**(l2*t)
y2 = c1*v1[1]*math.e**(l1*t) + c2*v2[1]*math.e**(l2*t)
```

In [ ]:
```
# b
#z=limit(y2/y1,t,oo)
#z
```

In [ ]:
```
for x in range (100,201):
    f1 = c1*v1[0]*math.e**(l1*x) + c2*v2[0]*math.e**(l2*x)
    f2 = c1*v1[1]*math.e**(l1*x) + c2*v2[1]*math.e**(l2*x)
    if f1-f2 <= 1:
        print(x)
        break
```

```
161
```

Assignment 3

In [ ]:
```
u1 = Matrix([0,1,-1,-2])
u2 = Matrix([1,0,1,0])
v1 = Matrix([0,-1,0,1])
v2 = Matrix([1,0,0,-1])
Matrix.hstack(v1, v2, u1, u2).rref()
```

Out[ ]:
$$\left(\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \ (0, 1, 2)\right)$$

In [ ]:
```
# so dim U+V = 3
```

Assignment 4

In [ ]:
```
A = Matrix([[1,2,-4],[3,1,0],[2,3,-3]])
B = Matrix([[2,3],[1,0],[3,-1]])
X = A**-1 * B
X, A*X, A
```

Out[ ]:
$$\left(\begin{bmatrix} 0 & 1 \\ 1 & -3 \\ 0 & -2 \end{bmatrix}, \ \begin{bmatrix} 2 & 3 \\ 1 & 0 \\ 3 & -1 \end{bmatrix}, \ \begin{bmatrix} 1 & 2 & -4 \\ 3 & 1 & 0 \\ 2 & 3 & -3 \end{bmatrix}\right)$$

In [ ]:
```
k = symbols('k')
A = Matrix([[1,2,-4],[3,k,0],[2,3,-3]])
B = Matrix([[2,3],[1,0],[3,-1]])
L, U, _ = A.LUdecomposition()
s = solve(U[2,2], k)
s
```

Out[ ]:
$$\left[\frac{18}{5}\right]$$

S

Assignment 5

In [ ]:
```
import pandas as pd
x = pd.DataFrame([0.2, 0.3, 1.1, 1.3, 2.2, 2.4, 2.5])
y = pd.DataFrame([1.6, -1.2, -0.3, 2.1, 2.3, -0.2, -1.3])
```

In [ ]:
```
# a)
X1 = Matrix([ones(len(x), 1)]).row_join(Matrix(x**3))
X2 = Matrix((Matrix(x**2)).row_join(Matrix(x**3)))
X3 = Matrix([ones(len(x), 1)]).row_join(Matrix(x))

display(Math(r'X_1 = ' + latex(X1) + r'\quad X_2 = ' + latex(X2) + r'\quad X_3 =
             + latex(Matrix(y))))
```

$$X_1 = \begin{bmatrix} 1 & 0.008 \\ 1 & 0.027 \\ 1 & 1.331 \\ 1 & 2.197 \\ 1 & 10.648 \\ 1 & 13.824 \\ 1 & 15.625 \end{bmatrix} \quad X_2 = \begin{bmatrix} 0.04 & 0.008 \\ 0.09 & 0.027 \\ 1.21 & 1.331 \\ 1.69 & 2.197 \\ 4.84 & 10.648 \\ 5.76 & 13.824 \\ 6.25 & 15.625 \end{bmatrix} \quad X_3 = \begin{bmatrix} 1 & 0.2 \\ 1 & 0.3 \\ 1 & 1.1 \\ 1 & 1.3 \\ 1 & 2.2 \\ 1 & 2.4 \\ 1 & 2.5 \end{bmatrix} \quad y = \begin{bmatrix} 1.6 \\ -1.2 \\ -0.3 \\ 2.1 \\ 2.3 \\ -0.2 \\ -1.3 \end{bmatrix}$$

In [ ]:
```
X1tX1 = X1.T*X1
X1ty = X1.T*Matrix(y)
Mat, _ = X1tX1.row_join(X1ty).rref()
B1 = Mat[:,-1]


X2tX2 = X2.T*X2
X2ty = X2.T*Matrix(y)
Mat, _ = X2tX2.row_join(X2ty).rref()
B2 = Mat[:,-1]


X3tX3 = X3.T*X3
X3ty = X3.T*Matrix(y)
Mat, _ = X3tX3.row_join(X3ty).rref()
B3 = Mat[:,-1]
```

In [ ]:
```
# b)
display(Math(r'X_1^TX_1 = ' + latex(X1tX1)))
display(Math(r'X_1^Ty = ' + latex(X1ty)))
display(Math(r'X_2^TX_2 = ' + latex(X2tX2)))
display(Math(r'X_2^Ty = ' + latex(X2ty)))
display(Math(r'X_3^TX_3 = ' + latex(X3tX3.evalf(5))))
display(Math(r'X_3^Ty = ' + latex(X3ty)))
```

$$X_1^T X_1 = \begin{bmatrix} 7 & 43.66 \\ 43.66 & 555.222668 \end{bmatrix}$$

$$X_1^T y = \begin{bmatrix} 3.0 \\ 5.6079 \end{bmatrix}$$

$$X_2^T X_2 = \begin{bmatrix} 99.9956 & 234.145 \\ 234.145 & 555.222668 \end{bmatrix}$$

$$X_2^T y = \begin{bmatrix} 4.997 \\ 5.6079 \end{bmatrix}$$

$$X_3^T X_3 = \begin{bmatrix} 7.0 & 10.0 \\ 10.0 & 19.88 \end{bmatrix}$$

$$X_3^T y = \begin{bmatrix} 3.0 \\ 3.69 \end{bmatrix}$$

In [ ]:
```
display(Latex("$$y_1(t) = {}+{}t^3$$".format(round(B1[0],2), round(B1[0],4))))
display(Latex("$$y_2(t) = {}t^2+{}t^3$$".format(round(B2[0],2), round(B2[1], 4))))
display(Latex("$$y_3(t) = {}+{}t$$".format(round(B3[0],2), round(B3[1], 4))))
```

$$y_1(t) = 0.72 + 0.7175t^3$$

$$y_2(t) = 2.10t^2 + -0.8754t^3$$

$$y_3(t) = 0.58 + -0.1065t$$

In [ ]:
```
# c)
display(Latex("$$e_1 = {}$$".format(round((Matrix(y)-X1*B1).norm(), 2))))
display(Latex("$$e_2 = {}$$".format(round((Matrix(y)-X2*B2).norm(), 2))))
display(Latex("$$e_3 = {}$$".format(round((Matrix(y)-X3*B3).norm(), 2))))
```

$$e_1 = 3.69$$

$$e_2 = 3.15$$

$$e_3 = 3.76$$

Assignment 6

In [ ]:
```
AAt = Matrix([[2,4,4],[4,8,8],[4,8,8]])
AAt.eigenvects()
u1 = 2*AAt.eigenvects()[1][2][0]
u2 = AAt.eigenvects()[0][2][0]
u3 = AAt.eigenvects()[0][2][1]

u1, u2, u3 = GramSchmidt((u1, u2, u3), True)
U = Matrix.hstack(u1, u2, u3)

U
```

Out[ ]:
$$\begin{bmatrix} \frac{1}{3} & -\frac{2\sqrt{5}}{5} & -\frac{2\sqrt{5}}{15} \\ \frac{2}{3} & \frac{\sqrt{5}}{5} & -\frac{4\sqrt{5}}{15} \\ \frac{2}{3} & 0 & \frac{\sqrt{5}}{3} \end{bmatrix}$$

In [ ]:
```
# a)
AAt.eigenvects() # multiply the first by 2
```

Out[ ]:
$$\left[\left(0,\ 2,\ \left[\begin{bmatrix}-2\\1\\0\end{bmatrix},\ \begin{bmatrix}-2\\0\\1\end{bmatrix}\right]\right),\ \left(18,\ 1,\ \left[\begin{bmatrix}\frac{1}{2}\\1\\1\end{bmatrix}\right]\right)\right]$$

In [ ]:
```
# b)
s1 = sqrt(AAt.eigenvects()[1][0])
V = Matrix([[(sqrt(2))/(2),-(sqrt(2))/(2)],[(sqrt(2))/(2),(sqrt(2))/(2)]])
Vt = V.T
S = diag(s1, 0).col_join(zeros(1,2))
S

display(S)
display(U * S * Vt)
display(AAt)
display(Vt)
```

$$\begin{bmatrix}3\sqrt{2} & 0\\0 & 0\\0 & 0\end{bmatrix}$$

$$\begin{bmatrix}1 & 1\\2 & 2\\2 & 2\end{bmatrix}$$

$$\begin{bmatrix}2 & 4 & 4\\4 & 8 & 8\\4 & 8 & 8\end{bmatrix}$$

$$\begin{bmatrix}\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\\-\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2}\end{bmatrix}$$

In [ ]:
```
V = Matrix([[(sqrt(2))/(2),-(sqrt(2))/(2)],[(sqrt(2))/(2),(sqrt(2))/(2)]])
U * S * V.T
```

Out[ ]:
$$\begin{bmatrix}1 & 1\\2 & 2\\2 & 2\end{bmatrix}$$

In [ ]:
```
display(1+1+2+2+2+2)
display(Math('Sum  of  A  matrix  is= {}'.format(latex(np.sum(U * S * V.T)))))
```

10

$Sum of A matrix is = 10$

In [ ]: