**Name: Somit Jain**

**Reg. No. 20BDS0181**

**Course: Mathematical Modelling for Data Science(CSE3045 ELA)**

**Faculty: Dr. Ilanthenral K P S K**

**Slot: L49+L50**

**Lab-Assignment 1**

1. Using Python NumPy package linalg, explore all the possible routines along with different possible scenarios with specific reference to

import numpy as np

A = np.array([[1,2],

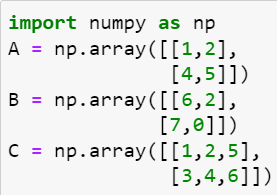
[4,5]])

B = np.array([[6,2],

[7,0]])

C = np.array([[1,2,5],

[3,4,6]])

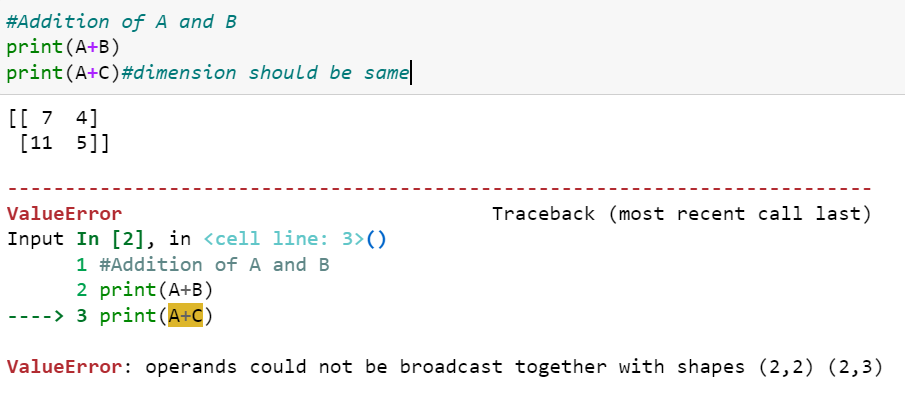


1. **Matrix addition**

**Code:** #Addition of A and B

print(A+B)

print(A+C)#dimension should be same



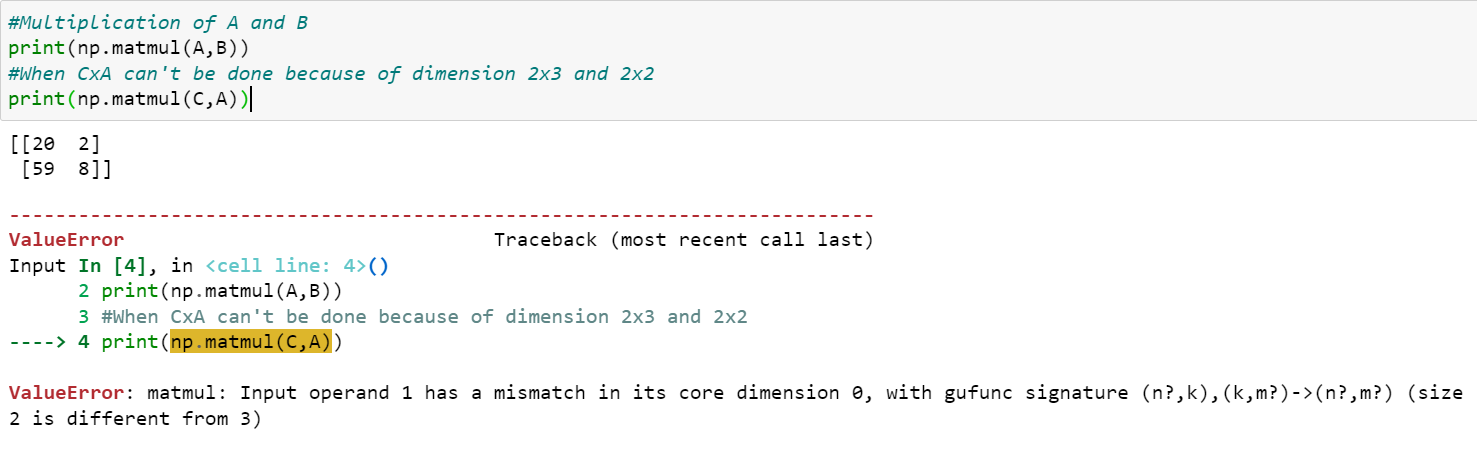
1. Matrix multiplication

Code: #Multiplication of A and B

print(np.matmul(A,B))

#When CxA can't be done because of dimension 2x3 and 2x2

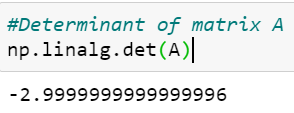
print(np.matmul(C,A))



1. Determinant of a matrix

Code: #Determinant of matrix A

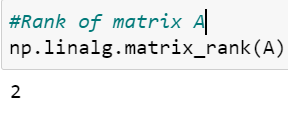
np.linalg.det(A)



1. Rank of a matrix

Code: #Rank of matrix A

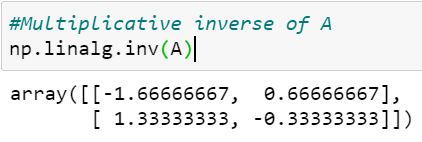
np.linalg.matrix\_rank(A)



1. Multiplicative inverse of a matrix

#Multiplicative inverse of A

np.linalg.inv(A)



Code: #Let D be a matrix of det = 0

D = np.matrix([[1,2],

[2,4]])

print(np.linalg.det(D))

print(np.linalg.inv(D))#Inverse can't be found for matrix with det 0



2. Without using inbuilt routines, solve a system of linear equations with two variables.

a. By substitution method

Code: """Let two equations be a1x+b1y=c1 and a2x+b2y=c2

We know that in substitution we substitute a variable in 1 equation using other equation. By doing this we get the following

solution x = (c1b2-b2c1)/(a1b2-a2b1) and y = x = (a1c2-a2c1)/(a1b2-a2b1) and if denominator is infinity the lines are

parallel"""

a1,b1,c1,a2,b2,c2 = map(int,input().split())

det = a1\*b2 - b1\*a2

if det==0:

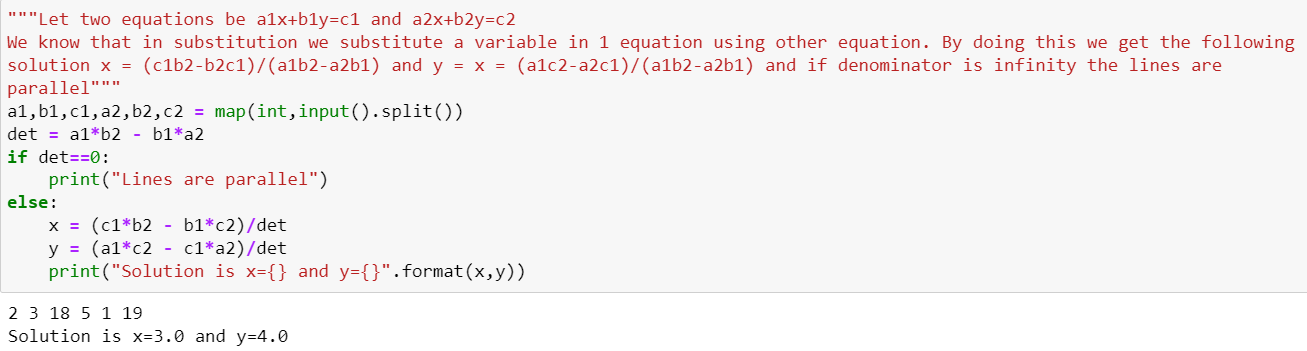
print("Lines are parallel")

else:

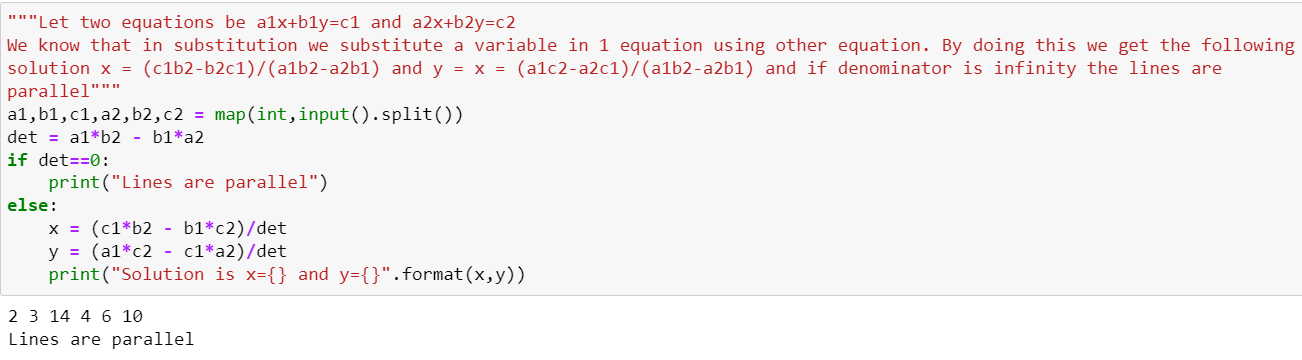
x = (c1\*b2 - b1\*c2)/det

y = (a1\*c2 - c1\*a2)/det

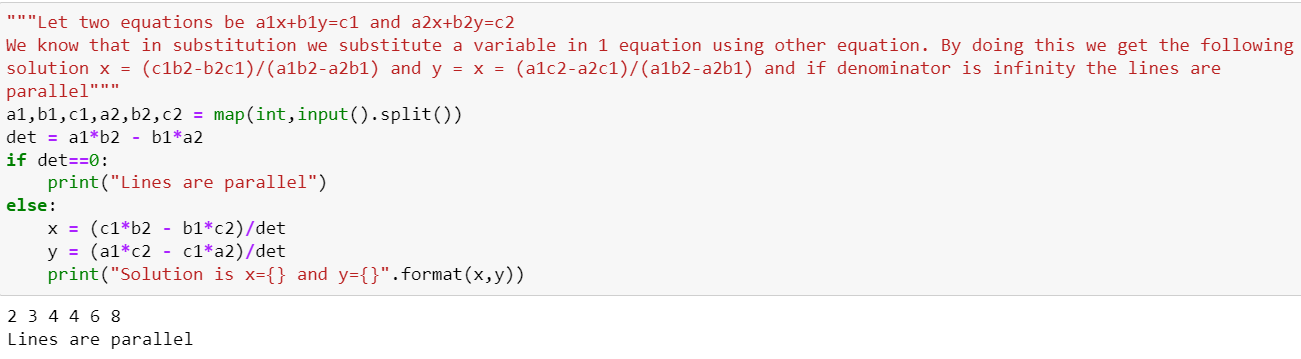
print("Solution is x={} and y={}".format(x,y))



No Solution:



Infinite Solution:



b. By Elimination method

Code: #Let two equations be a1x+b1y=c1 and a2x+b2y=c2

a1,b1,c1,a2,b2,c2 = map(int,input().split())

b1 = b1\*a2

c1 = c1\*a2

b2 = b2\*a1

c2 = c2\*a1

a1 = a1\*a2

a2 = a1

if b1-b2==0:

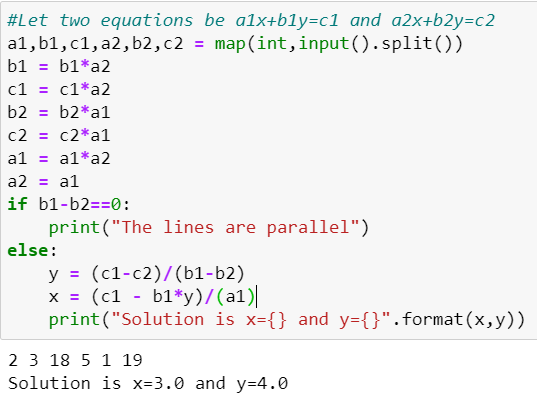
print("The lines are parallel")

else:

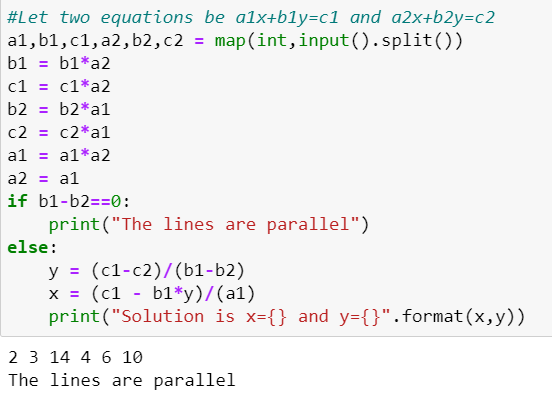
y = (c1-c2)/(b1-b2)

x = (c1 - b1\*y)/(a1)

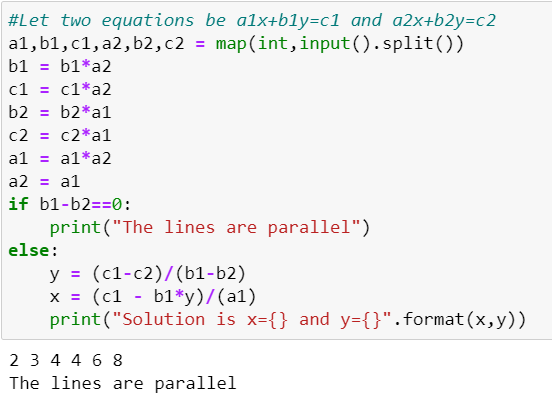
print("Solution is x={} and y={}".format(x,y))



No Solution:



Infinite Solution:



c. Gaussian Elimination and Gauss- Jordan Method

Code: #Gauss Elimination Method: Every 1st element of the row should be the only non zero element in its column

import numpy as np

a = np.zeros((2,3))

x = np.zeros(2)

for i in range(2):#taking input of augmented matrix

for j in range(3):

a[i][j] = float(input())

print(a)

rat = a[1][0]/a[0][0]

for k in range(3):

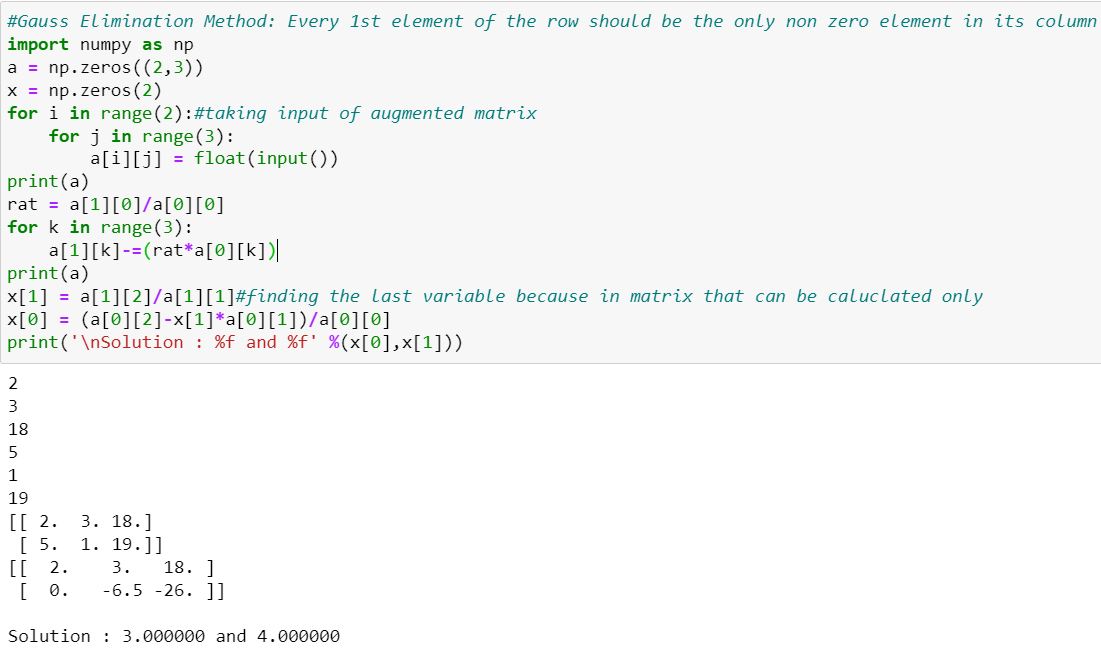
a[1][k]-=(rat\*a[0][k])

print(a)

x[1] = a[1][2]/a[1][1]#finding the last variable because in matrix that can be caluclated only

x[0] = (a[0][2]-x[1]\*a[0][1])/a[0][0]

print('\nSolution : %f and %f' %(x[0],x[1]))



Code: #Gauss Jordan Elimination: There can be only one non zero element in each row and column

import numpy as np

a = np.zeros((2,3))

x = np.zeros(2)

for i in range(2):#taking input of augmented matrix

for j in range(3):

a[i][j] = float(input())

print(a)

rat1 = a[1][0]/a[0][0]

for k in range(3):

a[1][k]-=(rat1\*a[0][k])

rat2 = a[0][1]/a[1][1]

for k in range(3):

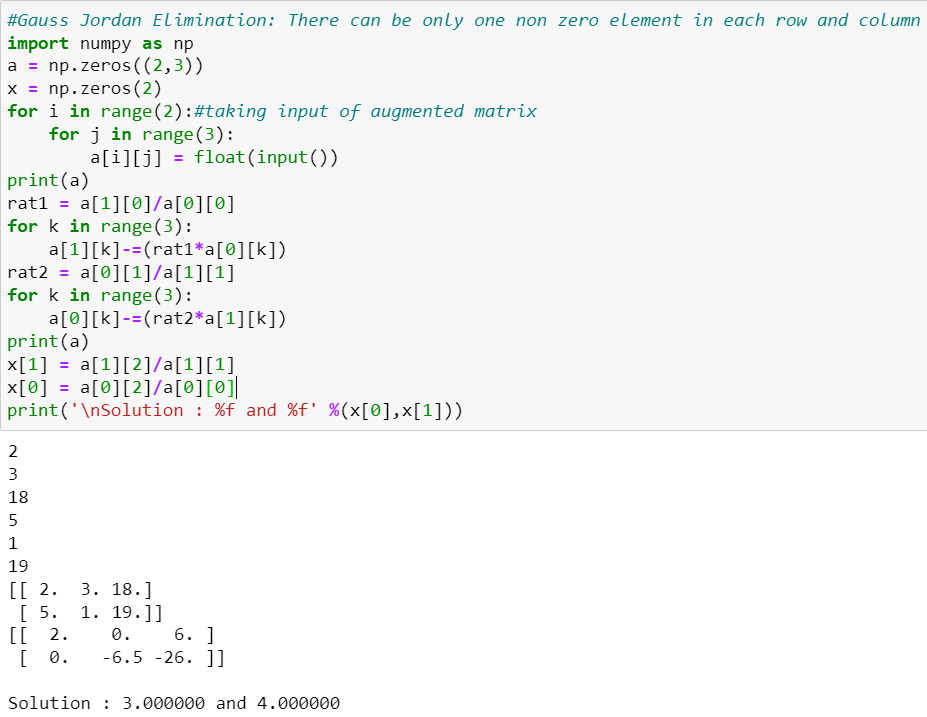
a[0][k]-=(rat2\*a[1][k])

print(a)

x[1] = a[1][2]/a[1][1]

x[0] = a[0][2]/a[0][0]

print('\nSolution : %f and %f' %(x[0],x[1]))



Code: import numpy as np

a = np.ones((2,2))

b = np.zeros((2,1))

x = np.zeros(2)

for i in range(2):#taking input of augmented matrix

a[i][0]= float(input())

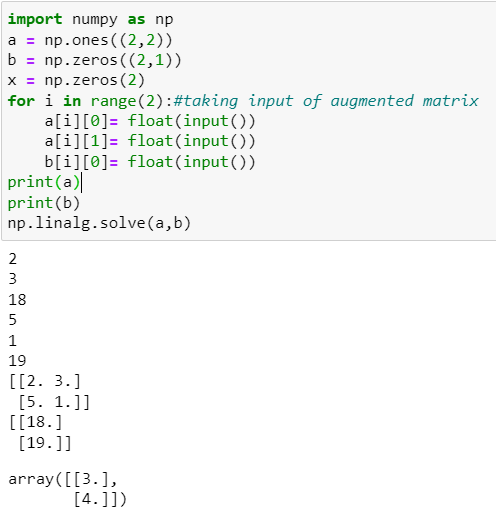
a[i][1]= float(input())

b[i][0]= float(input())

print(a)

print(b)

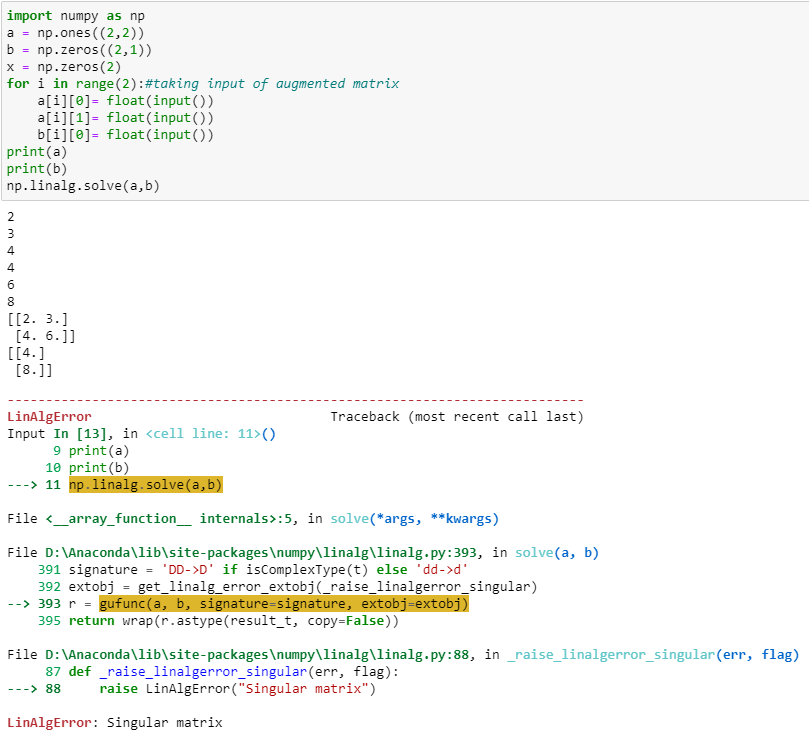
np.linalg.solve(a,b)



No Solution:



Infinite Solution:



Lab 2: Eigen Values and Eigen Vectors

1. Find Eigen values and eigen vectors using Python NumPy package linalg, explore all the possible routines along with different possible scenarios.

Code: import numpy as np

import numpy.linalg as alg

A = np.matrix([[2,-3,0],

[2,-5,0],

[0,0,3]])

evalu,evect = alg.eig(A)

evalu = np.round\_(evalu)

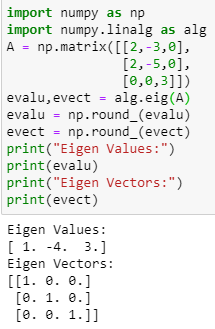
evect = np.round\_(evect)

print("Eigen Values:")

print(evalu)

print("Eigen Vectors:")

print(evect)



Not a Square Matrix:



4. Using R, find the eigen values and vectors.

Code: A = matrix(c(2,-3,0,2,-5,0,0,0,3),3,3,byrow = TRUE)

print(A)

print(eigen(A))

