**Home work 3&4 python**

1. Calculating the inverse of a 2x2 matrix without using numpy

Sol:

import numpy as np

def det(arr):

return arr[0,0]\*arr[1,1]-arr[0,1]\*arr[1,0]

def interchange(arr):

c=np.array([(arr[1,1],-arr[0,1]),(-arr[1,0],arr[0,0])])

return c

def inv\_matrix(arr):

if det(arr)==0:return "Error the matrix is singular"

return (1/det(arr))\*interchange(arr)

a=[]

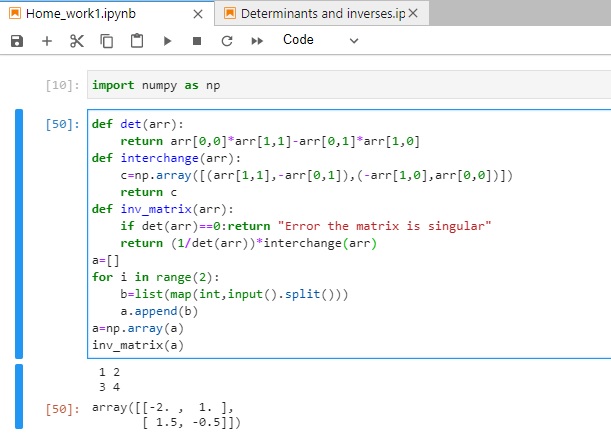
for i in range(2):

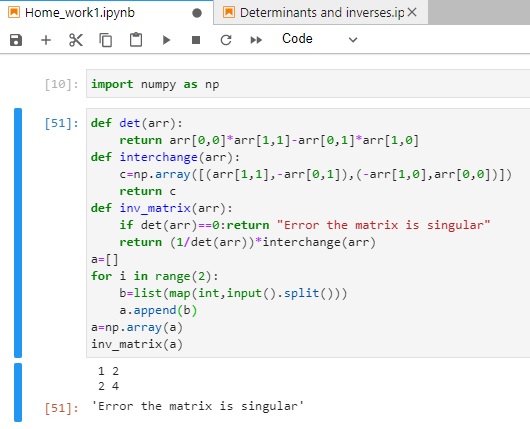
b=list(map(int,input().split()))

a.append(b)

a=np.array(a)

inv\_matrix(a)





1. Coding a Python code to inverse a 3x3 matrix:

Sol:

import numpy as np

def det(arr):

n=len(arr)

if n==1:return arr[0,0]

if n==2:return arr[0,0]\*arr[1,1]-arr[1,0]\*arr[0,1]

sum=0

for i in range(0,n):

m=minor(arr,0,i)

sum=sum+((-1)\*\*i)\*arr[0,i]\*det(m)

return sum

def minor(arr,i,j):

c=np.delete(arr,i,0)

c=np.delete(c,j,1)

return c

def cofactor(arr):

n=len(arr)

c=np.empty([3,3])

l=0

for i in range(0,n):

for j in range(0,n):

c[i,j]=((-1)\*\*l)\*det(minor(arr,i,j))

l+=1

return c

def transpose(arr):

b=np.array([])

for i in range(0,3):

b=np.concatenate((b,arr[:,i]))

return b.reshape([3,3])

def inv\_matrix(arr):

if det(arr)==0:return "Error the matrix is singular"

d= det(arr)

m=cofactor(arr)

adj=transpose(m)

return (1/d)\*adj

a=[]

for i in range(3):

b=list(map(int,input().split()))

a.append(b)

a=np.array(a)

c=inv\_matrix(a)

print(c)



