```
# -*- coding: utf-8 -*-
"""
Created on Tue Feb 12 16:00:28 2019
HW3
@author: Ryan Dalby
"""

import numpy as np

"""
Performs matrix operations for Exercise 1
"""
print("Exercise 1")
A = np.array([[4,7],[1, 2],[5, 6]])
B = np.array([[4,3,7],[1,2,7],[2,0,4]])
C = np.array([[3],[6],[1]])
D = np.array([[9,4,3,-6], [2,-1,7,5]])
E = np.array([[1,5,8],[7,2,3],[4,0,6]])
F = np.array([[3,0,1], [1,7,3]])
G = np.array([[7,6,4]])


ansA = E + B
ansB = np.matmul(A, F)
ansC = B - E
ansD = 7 * B
ansE = np.matmul(E, B)
ansF = np.transpose(C)
ansG = np.matmul(B, A)
ansH = np.transpose(D)
print("ansA = \n{}\nansB = \n{}\nansC = \n{}\nansD = \n{}\nansE = \n{}\nansF = \n{}\nansG
print("\n\n\n\n")
"""
Solves given system of equations (represented by Mx = b) for Exercise 2
"""
print("Exercise 2")
M2 = np.array([[-2.2,20], [-1, 8.7]])
b2 = np.array([[240],[87]])
ex2Ansa = np.linalg.det(M2)
ex2Ansb = np.linalg.solve(M2, b2)

print("ansA = \n{}\nansB = \nx1 = {}\nx2 = {}".format(ex2Ansa, ex2Ansb[0], ex2Ansb[1]))

print("\n\n\n\n")

"""
Exercise 3
"""
def gaussElimination(A, b):
    """
    Uses naive gauss elimination method without pivoting to determine solutions to Ax = b
    """
    n = np.shape(A)[0] #assuming nxn matrix

    print("Naive Gauss Elimination Steps:\n")
    print("Original Matrix:")
    print(A,"\n")
    print("Perform forward elimination:")
    #Perform forward elimination
    for k in range(n - 1):
        for i in range(k + 1, n):
            s = A[i,k] / A[k,k]
            for j in range(k, n):
                A[i,j] = A[i,j] - s * A[k,j]
                print("A = \n{}\nb = \n{}\n".format(A,b))
            b[i] = b[i] - s * b[k]

    print()
    print("Perform back substitution:")
    #Perform back substitution
    xSol = np.zeros(shape = (n,1))
    xSol[n-1] = b[n-1] / A[n-1,n-1] #index last element of xSol and set it to the solution
    print("A = \n{}\nb = \n{}\n".format(A,b))
    for i in range(n-1, -1, -1):
        s = 0
        for j in range(i+1, n):
            s = s + A[i,j] * xSol[j]
        xSol[i] = (b[i] - s) / A[i,i]
        print("x = \n{}\n".format(xSol))
    return xSol

print("Exercise 3")
#z = gaussElimination(M2, b2) #Test with last problem
#print(z)

M3 = np.array([[5,1,-.5], [-6,-12,4], [2,2,10]])
b3 = np.array([[13.5], [-123], [-43]])

ansEx3 = gaussElimination(M3, b3)
print("Final x = \n{}".format(ansEx3))
```