```python
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 26 21:13:09 2019
Naive Gauss Elimination from HW3 Exercise 3
@author: Ryan Dalby
"""

import numpy as np

def gaussElimination(A, b):
    """
    Uses naive gauss elimination method without pivoting to determine solutions to Ax = b
    Will directly modify A passed into upper triangular form
    """
    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("A = \n{}\nb = \n{}\n".format(A,b))


#    print()
#    print("Perform back substitution:")
    #Perform back substitution
    xSol = np.zeros(shape = (n,1), dtype='float')
    xSol[n-1] = b[n-1] / A[n-1,n-1] #index last element of xSol and set it to the solution value fo
#    print("A = \n{}\nb = \n{}\n".format(A,b))
    for i in range(n-1, -1, -1):
        s = 0.0
        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
```