Program that finds the Null space basis vectors - Akash Gupta

Basic Python functionality is implemented here.

```
# Finding NullSpace solutions to any square vector
import random
#A = [[1,5,2,7,5,4],
#     [6,11,44,6,9,6],
#     [2,10,4,14,10,8],
#     [33,77,2,67,89,21],
#     [12,22,88,12,18,12],
#     [3,76,22,89,123,5]]

A = [[1,5,2,7,5,4],
     [3,15,6,21,15,12],
     [2,10,4,14,10,8],
     [33,77,2,67,89,21],
     [12,22,88,12,18,12],
     [3,76,22,89,123,5]]

#A = [[random.randint(1,5) for x in range(10)] for y in range(10)]

rows = len(A)
cols = len(A[0])

def printMatrix(A, rows, cols):
    for i in range(rows):
        for j in range(cols):
            print(A[i][j], end = "\t")
        print()

def rowEchelonForm(A, rows, cols):
    for i in range(rows - 1):
        zeroPiv = False
        if A[i][i] == 0:
            zeroPiv == True
            for r in range(i+1,rows):
                if A[r][i] != 0:
                    for c in range(cols):
                        A[i][c], A[r][c] = A[r][c], A[i][c]
                    zeroPiv = False
                    break
                elif A[r][i] == 0:
                    zeroPiv = True

        if zeroPiv == True:
            return(A)
        elif zeroPiv == False:
            for j in range(i+1,rows):
                factor = A[j][i]/A[i][i]
                for k in range(cols):
                    A[j][k] = A[j][k] - (factor * A[i][k])
    return A

def Determinant(A,rows):
```

```
    determinant = 1
    for i in range(rows):
        determinant = determinant * A[i][i]
    return determinant


def countFreeVariables(B,rows,cols):
    zRow = [0.0 for z in range(cols)]
    count = 0
    for i in range(rows):
        if A[i] == zRow:
            count += 1
        else:
            count = count
    return count


B = rowEchelonForm(A,rows,cols)

def NullSol(B,rows,cols):
    count = countFreeVariables(B,rows,cols)
    fullSum = {}
    Y = []
    pivSol = []

    for i in range(cols-1,cols-(count+1),-1):
        X = [0 for x in range(rows)]
        X[i] = 1
        Y.append(X)
        sums = []
        for m in range(rows-(count+1),-1,-1):
            sumOfRow = 0
            sumOfRow = B[m][i]
            if B[m][m] != 0:
                sums.append(-sumOfRow/B[m][m])

        fullSum[i] = sums

    for vals in fullSum.values():
        pivSol.append(vals)

    for y1 in range(len(Y)):
        for y2 in range(len(Y[y1])-count):
            if Y[y1][y2] == 0:
                Y[y1][y2] = pivSol[y1][y2]
            else:
                pass
    if pivSol == []:
        print("null space contains only the zero vector")
    else:
        print("the Null space is formed by linear combinations of the following special solution vectors
        print("\n")
        print(Y)
```

```
printMatrix(B,rows,cols)
print("\n")
NullSol(B,rows,cols)
```

```
1   5   2   7   5   4
0.0 -88.0   -64.0   -164.0  -76.0   -111.0
0.0 0.0 91.63636363636364   -1.181818181818187  -9.18181818181818   11.93181818181818
0.0 -7.105427357601002e-15  0.0 -46.04761904761905  52.476190476190474  -80.25000000000001
0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0


the Null space is formed by linear combinations of the following special solution vectors


[[-1.7427611168562567, -0.13020833333333331, -1.2613636363636365, -4.0, 0, 1], [1.139607032057911, 0.10
```