Pivoting

The starting point of this tutorial is the LU decomposition code (`LU_scratch.py`) that we wrote in class. We found that it fails when the input matrix has a singular sub-matrix, even when the input matrix itself is invertible. For instance

$$A = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 2 & -1 \\ 1 & -1 & 0 \end{pmatrix}$$

has the sub-matrix

$$\begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix},$$

which is singular and therefore our LU decomposition code encounters a division by zero.
The solution to this problem is "pivoting", described in Lecture 6. In particular, we can do LU decomposition with partial pivoting, which involves finding a permutation matrix P such that $PA = LU$. The pseudo-code for LU decomposition with partial pivoting can be found in the Lecture 6 Slides. In this tutorial, we will implement this algorithm following the steps below.

- Write a function that swaps two rows of a matrix. Inputs should be a $n \times n$ array M and two indices i and j, where $0 \leq i, j \leq n-1$ (following the Python convention that indices start from 0). Output should be the array M with rows i and j swapped.

- The function you have created will be good for the swapping rows as required for P and U (i.e. all elements of the rows get moved). For L you must only swap the elements of the appropriate rows that are *before* the column with the pivot. Modify your row swapping function so that only the first k elements of the rows get swapped, where k will be an additional input to your function. Note, you can also use this function on P and U, with the appropriate value of k.

- There is a line in the pseudo-code that says "Select $k \geq j$ to maximize $|U_{k,j}|$." Read the documentation on the `argmax` function on `numpy.org` to figure out how to find the index of the largest element in an array. Work out how to use this function to implement the line in the pseudo-code mentioned above.

- Now implement the pseudo-code for LU decomposition with pivoting. Input should be a $n \times n$ array A, output should be $n \times n$ arrays P, L and U so that P is a permutation matrix and $PA = LU$. Note that the last loop in the pseudo-code with pivoting is the same as without. So you may find the `LU_scratch.py` code helpful.

- Test it by decomposing the matrix A above and verifying that $PA = LU$. Also, test the matrix from class (see `LU_scratch.py`) that didn't work when we didn't use pivoting.

- **Bonus**: Memory and time savers: (a) When you swap whole rows of U, you're actually swapping lots of zeros. The first $j-1$ elements of both of the rows you're swapping are zeros, and so it's better to avoid swapping these elements (note this is already written into the pseudo-code). Modify your row swapping function to incorporate this efficiency. (b) Modify your code so that you store the permutation matrix as a vector; see last part of Lecture 6 slides. This memory saver can be very significant when the size of your matrices is large.