On computing the determinant...

In this tutorial you will program a function for computing the determinant of a matrix. The determinant of a square matrix is a single number that gives you important information about the matrix and about the corresponding linear system. That is, for $A \in GL(n,n)$ $(n > 1)$ if $\det(A) \neq 0$, then

- A is invertible (i.e, A^{-1} exists)
- The linear system $Ax = b$ for $b \in \mathcal{R}^n$, has a unique solution.

The determinant of a square matrix can be calculated using this algorithm:
Let $A \in GL(n,n)$ $(n > 1)$ be a square matrix.

1. If $n = 2$ then $\det(A) = A_{11}A_{22} - A_{12}A_{21}$.
2. If $n > 2$ then construct the k^{th} submatrix $A^{(k)}$ by deleting the first column and k^{th} row of A. Then compute the determinant as

$$\det(A) = \sum_{k=1}^{n} (-1)^{k+1} A_{k1} \det(A^{(k)})$$

This is a recursive definition: only the determinant of a 2×2 matrix is defined explicitly, in all other cases the determinant is computed as a sum of determinants of smaller matrices. Therefore, we are going to program this recursively. You can find a simple example on slide 14 of lecture 3.

1. Write a pseudo-code for the recursive computation of the determinant. Remember the basic rules of pseudo-code: make it clear and transparent, avoid programming language-specific key words (like `range` of `np.copy`) and make sure it can be translated directly into Python (or C or C++ or any other reasonable language). As an exercise, consider exchanging your pseudo-code wiht a classmate and basing your code on their pseudo-code.
2. Implement your function in Python. Use the function `np.random.rand` to generate a random array of size $n \times n$ and check the result against the built-in function `np.linalg.det`. How large can you make the matrix size n?

Discussion: You will have noticed that the built-in function completed a lot faster than your recursively programmed function. Do you have an idea why?