

## TP n°8 - Matrices et array

1. Création d'array numpy

**Q1 :** Modéliser les matrices suivantes en utilisant des array numpy :

$$\text{a)} \quad A_1 = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 \\ 0 & 1 & \ddots & & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & & \ddots & 1 & 0 \\ 1 & 0 & \dots & 0 & 1 \end{pmatrix} \text{ avec 15 lignes et 15 colonnes.}$$

$$\text{b)} \quad A_2 = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ 1 & 2 & \dots & 29 & 30 \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix} \text{ et } A_3 = \begin{pmatrix} 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & \dots & 2 & 0 \\ 0 & 0 & \dots & 3 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 30 & 0 \end{pmatrix} \text{ avec 30 lignes et 30 colonnes.}$$

$$\text{c)} \quad A_4 = \begin{pmatrix} 3 & 0 & \dots & 0 & 0 \\ 0 & 6 & \ddots & & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & & \ddots & 87 & 0 \\ 0 & 0 & \dots & 0 & 90 \end{pmatrix}$$

**Q2 :** **a)** Définir une fonction `base(n)` qui prend en argument un entier naturel non nul  $n$  et qui renvoie la liste des array numpy correspondant aux  $n^2$  matrices avec  $n$  lignes et  $n$  colonnes remplies de 0 sauf pour un coefficient valant 1 à chacune des places possibles.

**b)** Définir une fonction `matrice(n)` qui prend en argument un entier naturel non nul  $n$  et qui renvoie l'array numpy correspondant à la matrice avec  $n$  lignes et  $n$  colonnes telle que :

$$A = \begin{pmatrix} 1 & 2 & \dots & n-1 & n \\ n+1 & n+2 & \dots & 2n-1 & 2n \\ \vdots & \vdots & & \vdots & \vdots \\ ? & ? & \dots & n^2-1 & n^2 \end{pmatrix}$$

2. Utilisation des outils de calculs matriciels sur les array numpy

Donner les instructions les plus courtes possibles permettant de renvoyer le résultat demandé - les matrices seront bien-sûr modélisées par des array numpy.

**Q1 :** Créer une matrice triangulaire supérieure de taille  $10 \times 10$  dont tous les termes sont nuls exceptés les 9 coefficients de la surdiagonale qui sont égaux à 1.  
Calculer ses puissances 2, 3, ..., 9, 10. Que dire de cette matrice ?

**Q2 :** Dans cette question, on importera le module `random`.

- a) Définir une fonction `mat_aleatoire()` qui renvoie une matrice de taille  $2 \times 2$  dont chaque coefficient est un nombre flottant tiré uniformément et aléatoirement entre 0 et 1.
- b) Définir une matrice `A=mat_aleatoire()` puis renvoyer :
  - sa deuxième colonne,
  - sa transposée,
  - son déterminant puis son inverse si elle existe.
- c) Faire plusieurs essais de calcul de déterminant avec des matrices "tirée aléatoirement" avec la fonction `mat_aleatoire()`. Qu'en pensez-vous ?

### 3. Programmation des principales opérations matricielles

Dans la suite, pour  $n, p \in \mathbb{N}^*$ , on désigne par  $M_{n,m}$  l'ensemble de matrices (à coefficients dans  $\mathbb{R}$  ou  $\mathbb{C}$ ) avec  $n$  lignes et  $m$  colonnes.

Pour rapprocher les formules matricielles de leur programmation informatique, pour  $A$  une matrice de  $M_{n,m}$ , on numérotera de 0 à  $n-1$  ses lignes et de 0 à  $m-1$  ses colonnes (en mathématiques, la numérotation commence traditionnellement en 1); puis on notera  $A_{i,j}$  le coefficient de la ligne  $i$ , colonne  $j$  de la matrice  $A$ .

Dans les questions de programmation qui suivent, par "matrice", on entendra "array numpy représentant une matrice" (ou encore "liste de listes représentant une matrice").

**Q1 :** Transposition de matrices

Pour  $A$  une matrice de  $M_{n,m}$ , on appelle **transposée** de  $A$  et on note  ${}^tA$  la matrice de  $M_{m,n}$  dont les colonnes sont les lignes de  $A$  dans le même ordre. Ainsi, pour une ligne  $i$  entre 0 et  $m-1$  et une colonne  $j$  entre 0 et  $n-1$  :

$${}^tA_{i,j} = A_{j,i}.$$

Écrire une fonction `transposee(A)` qui prend en argument une matrice  $A$  et qui renvoie sa transposée de deux façons :

- a) en utilisant la définition de la transposée en terme de coefficients et seulement des listes de listes pour représenter les matrices en jeu ;
- b) en utilisant la définition de la transposée en terme de transformations lignes/colonnes et des array numpy pour représenter les matrices en jeu.

**Q2 :** Matrices symétriques

On dit qu'une matrice carrée  $A$  de  $M_{n,n}$  est **symétrique** si  ${}^tA = A$ .

Écrire une fonction `est_symetrique(A)` qui prend en argument une matrice  $A$  et qui renvoie `True` si elle est symétrique et `False` sinon. Pour ce faire, on s'inspirera des algorithmes de recherche vu dans les TP précédent et on ne calculera pas la transposée de la matrice !

**Q3 :** Somme de matrices

Pour deux matrices  $A, B$  de  $M_{n,m}$  (et donc de **même taille**), la somme de  $A$  et  $B$  est la matrice notée  $A + B$  de coefficients :

$$(A + B)_{i,j} = A_{i,j} + B_{i,j}.$$

Écrire une fonction `addition(A,B)` qui prend en argument deux matrices  $A, B$  et qui renvoie la somme  $A+B$ .

*On rappelle que pour deux array numpy de nombres flottants et de même taille  $A$  et  $B$ , l'instruction  $A+B$  renvoie la somme matricielle des deux matrices... Bien-sûr, on n'utilisera pas cela pour écrire la fonction `addition` !.*

#### Q4 : Produit de matrices

Pour deux matrices  $A$  de  $M_{n,m}$  et  $B$  de  $M_{p,q}$ , lorsque  $m = p$  i.e. le nombre de colonnes de  $A$  est égal au nombre de lignes de  $B$ , le produit de  $A$  et  $B$  est la matrice de  $M_{n,q}$  notée  $AB$  de coefficients :

$$\begin{aligned}(AB)_{i,j} &= A_{i,0}B_{0,j} + A_{i,1}B_{1,j} \dots + A_{i,m-1}B_{m-1,j} \\ &= \sum_{k=0}^{m-1} A_{i,k}B_{k,j}\end{aligned}$$

Si  $m \neq p$ , le produit de  $A$  et  $B$  n'est pas possible.

Écrire une fonction `produit(A,B)` qui prend en argument deux matrices  $A, B$  et qui renvoie le produit  $AB$  lorsqu'il est possible et `None` sinon.