Lycée Marcelin BERTHELOT
BCPST 1C – Mme MOREL

Tableaux

`numpy` est une bibliothèque destinée à manipuler des tableaux (multidimensionnels) ainsi que les fonctions mathématiques qui opèrent sur ces tableaux.
Dans ce cours, nous considérerons les tableaux bidimensionnels dans le but de représenter des matrices, des images , etc ...
On charge la bibliothèque `numpy` sous l'alias `np` ; SYNTAXE :

1 Création

Un tableau est un ensemble d'éléments de **même type**, organisés en lignes et en colonnes.

SYNTAXE.
Pour créer un tableau bidimensionnel, on donne ses éléments sous forme d'une liste de listes , que l'on transforme en tableau en lui appliquant la fonction `array`.
Pour définir le tableau suivant :

x_{11}	x_{12}	$\cdots$	x_{1m}
x_{21}	x_{22}	$\cdots$	x_{2m}
$\vdots$	$\vdots$		$\vdots$
x_{n1}	x_{n2}	$\cdots$	x_{nm}

voici la syntaxe :

tab=np.array([[$x_{11},\ldots,x_{1m}$],[$x_{21},\ldots,x_{2m}$],...,[$x_{n1},\ldots,x_{nm}$]])

Chaque ligne est définie comme une liste; ces listes sont réunies dans une liste de listes. On applique ensuite la fonction `array`, d'où la lourdeur de la syntaxe ...

Remarque 1 Il y a une grande différence entre les tableaux et les listes: les listes peuvent contenir des objets de plusieurs types et peut changer de longueur, mais pas les tableaux. On rencontrera donc des tableaux d'entiers, de flottants (permettant de faire du calcul matriciel); de booléens; de chaînes de caractères (permettant de programmer des jeux); et toujours de même taille.

Exemple 1 :
(1) Écrire le tableau : $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$
(2) Écrire le tableau nul à deux lignes et quatre colonnes.
(3) Écrire la matrice identité de $\mathcal{M}_3(\mathbb{K})$, i.e. le tableau à trois lignes et trois colonnes ne contenant que des zéros, sauf des 1 sur sa diagonale.

Exemple 2 Pour créer un tableau de taille $n \times m$ rempli intégralement de la valeur 0:

```
Z=np.array(n*[m*[0]]) ou Z=np.zeros((n,m))
```

2 Accès aux éléments

On se donne un tableau T de taille $n \times m$, et i, j deux entiers appartenant respectivement à $[|0, n-1|]$ et $[|0, m-1|]$.

instruction Python	opérations sur les tableaux
`shape(T)`	donne les dimensions de T sous forme (nb lignes,nb colonnes)
`size(T)`	donne le nombre d'éléments de T
`size(T,0)`	donne le nombre de lignes de T
`size(T,1)`	donne le nombre de colonnes de T
`T[i,j]`	donne accès à l'élément de T situé sur la iè ligne et jè colonne
`T[i,:]`	donne accès à la ième ligne de T
`T[:,j]`	donne accès à la jème colonne de T

Remarque 2 Comme pour les listes, la numérotation des indices commence à 0 et s'arrête à $n-1$ pour les lignes et $m-1$ pour les colonnes.
Les indices négatifs permettent de parcourir les numéros de ligne ou colonne en partant de la fin.

Remarque 3 On se donne le tableau : $T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ En vous inspirant des listes, d'après vous, que renvoient les commandes suivantes ?

```
T[:, 1:3]
T[:2, 1:2]
T[:2, 1]
```

Rappel 1 :
Il est possible de modifier (ou d'effacer) un élément d'une liste au sein même de cette liste. On dit que les listes sont *mutables*.
La mutabilité des listes oblige à la prudence quand on souhaite dupliquer une liste. Par exemple , si on tappe dans la console:

x=[5,2,9] ; y=x

on dispose en fait d'une seule et même liste qui possède deux noms. Si l'on écrit alors dans la console :

x[2]=10 ; x ; y

la liste devient mais son adresse ne change pas en mémoire. Les variables x et y pointent donc vers la même liste, et ont donc la même valeur
Tout se passe donc comme si la mutation de x s'était répercutée sur y.
La syntaxe pour dupliquer la liste , c'est-à-dire créer une autre variable de même valeur que x , est :

y=x[:]

Remarque 4 Pour économiser de l'espace mémoire, `Python` évite autant que possible de recopier des listes ou des tableaux. Ainsi, si la variable **a** désigne une liste ou un tableau, l'instruction **b=a** ne crée pas une nouvelle copie de **a**; elle se contente d'attribuer un nouveau nom à la liste ou au tableau. Dès lors, toute modification de **a** affecte **b** et vice-versa. Il n'est donc pas possible, par simple affectation, de dupliquer une liste ou un tableau...
Pour les listes, on rappelle que l'instruction **newL=L[:]** crée une vraie copie de **L**. Si l'on procède ensuite à une modification de **newL**, cela ne change pas **L** et vice-versa.
Pour les tableaux: si **tab** désigne un tableau donné, l'instruction **newtab=tab[:,:]** agit comme une commande **newtab=tab**.

3 Opérations sur les tableaux

3.1 Opérations élémentaires

instruction Python	opérations sur les tableaux
`in`	appartenance
`sum`	somme

Exemple 3 : On donne les instructions suivantes:

```
T=np.array([[1,2,3],[4,5,6]])
1 in T
0 in T
s=sum(T)
print(s)
```

Que renvoie l'ordinateur?

Remarque 5 ATTENTION: le test d'égalité == ne permet pas, comme pour les listes, de comparer globalement deux tableaux.
Si les tableaux sont de même taille, la comparaison se fait case par case: on obtient donc un tableau de booléens et non un seul booléen; sinon, l'ordinateur renvoie `False`.

3.2 Opérations arithmétiques

Contrairement aux listes, on peut effectuer les opérations mathématiques (addition, multiplication, division, soustraction) sur les tableaux. Toutes ces opérations agissent case par case entre tableaux de même format.
SYNTAXE.

S+T	S-T	S*T	S/T	a*T

Remarque 6 L'addition et la multiplication par un `float` (dilatation) correspondent aux opérations connues sur les matrices, mais la multiplication ne correspond pas du tout au produit matriciel! Il s'agit d'une opération coefficient par coefficient entre deux tableaux de même taille.

Exemple 4 Effectuer les opérations entre les deux tableaux suivants:

```
T=np.array([[1,2,3],[4,5,6]])
S=np.array([[0,0,7],[6,6,6]])
```

Remarque 7 ATTENTION: concernant l'addition et la dilatation, il y a une grosse différence entre listes et tableaux.
En effet, pour les listes, + correspond à une fusion (concaténation). Pour les tableaux, c'est une simple addition case par case.
L'opération $n*$ crée une nouvelle liste constituée de n copies de la liste de départ. Mais pour les tableaux, chaque coefficient du tableau de départ est multiplié par n.

3.3 Opérations fonctionnelles

Toutes les fonctions mathématiques de référence peuvent être utilisées sur les tableaux. Elles agissent coefficient par coefficient sur les tableaux.
SYNTAXE. on conserve la notation habituelle d'une fonction, sauf que la variable est un tableau. On va donc chercher la fonction dans la bibliothèque adéquat !

Exemple 5 :

```
T=np.array([[1,2,3],[4,5,6]])
U=T**2
A=np.exp(T); B=np.sin(T)
```