Chapitre 1 : La bibliothèque numpy - les tableaux numpy

La bibliothèque numpy contient de nombreuses fonctions déjà écrites et très utiles pour les sciences : fonctions mathématiques usuelles, calcul matriciel, manipulations de tableaux, …. L'importation des fonctions du module numpy s'effectue par la commande suivante :

```
import numpy as np# Préfixe np à utiliser pour importer toutes les fonctions
```

a) Listes et tableaux numpy

A la différence des listes, un tableau généré sous numpy :

- Est de taille fixée à la création,
- Contient des éléments de même type (attention entiers ≠ flottants),
- Ne suit pas les mêmes règles de calcul (la manipulation des tableaux numpy se rapprochant de celle des vecteurs)

	Liste	Tableau numpy

Création	`#creation d'une liste`
	`L1=[0,1,2,3]`
	`#creation d'une liste de liste`
	`L2=[(0,1,2,3),(4,5,6,7)]`
		`#creation d'un tableau d'une ligne (équivalent à un vecteur)`
		`T1=np.array([0,1,2,3])`
		`L'exemple ci-dessus peut s'interpréter comme la conversion d'une liste en tableau :`
		`T4=np.array(L1)`
		`On peut utiliser un 2e argument optionnel pour fixer ou convertir le type de données (par défaut int64)`
		`T4=np.array(T1,dtype ="uint8")`
		`#creation d'un tableau 2D :`
		`T2=np.array([[0,1,2,3],`
		`[4,5,6,7]])`

Multiplier tous les éléments de la séquence par un entier	`L3=2*L1` `#L3 est une concaténation de L1 avec L1`	`T3=2*T1` `#Chaque élément de T1 est multiplié par 2 (on parle de vectorisation)`

b) Opérations « de bases » sur les tableaux numpy

Lignes de commandes	Interpréteur

`"""deux fonctions intéressantes pour créer des tableaux 1D en ligne"""`
`T5=np.linspace(0,4,5)#(départ,fin inclue,nbre de points)`
`T6=np.arange(0,5,1)#(départ,fin exclue,incrément)`
	`[ 0. 1. 2. 3. 4.]`
	`[0 1 2 3 4]`

`"""initialisation : création d'un tableau de zéros"""` `T7=np.zeros((2,3))#((nbre de lignes, nbre de colonnes))`	`[[ 0. 0. 0.]` `[ 0. 0. 0.]]`

`"""somme de tous les éléments d'un tableau"""` `s=np.sum(T1)`	6

`"""obtenir les dimensions d'un tableau"""`
`lignes,colonnes=np.shape((T7)) #renvoie un tuple(nbre lignes, nbre colonnes)`
`lignes= np.shape((T7))[0]`
	(2,3)
	2

`"""accès à une valeur ou création d'un sous tableau"""`
`T8=np.array([[0,1,2,3],[4,5,6,7],[8,9,10,11]])`
`print(T8[1])#renvoie la ligne d'index 1 (2e ligne)`
`print(T8[-1])#renvoie la dernière ligne`
`print(T8[1,2])#renvoie le motif en 2e ligne et 3e colonne`
`print(T8[1,:])#renvoie la ligne d'index 1`
`print(T8[1,:-1])#renvoie la ligne d'index 1 sans le dernier élément`
`print(T8[:,1])#renvoi tout le colonne d'index 1`
`print(T8[1:3,1:3])#[index le ligne:index de la dernière ligne exclue,index le colonne:index de la dernière colonne exclue]`
`print(T8[-1:-5:-1,-1:-5:-1])#début,fin, pas =-1 pour fixer le sens inverse`
	`[4 5 6 7]`
	`[8 9 10 11]`
	`6`
	`[4 5 6 7]`
	`[4 5 6]`
	`[1 5 9]`
	`[[ 5 6]`
	`[ 9 10]]`
	`[[11 10 9 8]`
	`[ 7 6 5 4]`
	`[ 3 2 1 0]]`

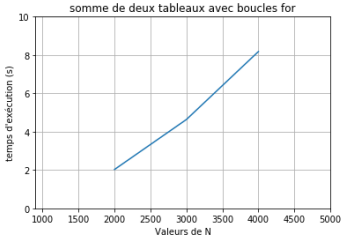
c) Notation vectorielle des tableaux numpy

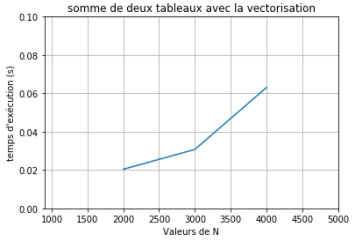
Soient trois tableaux numpy A, B et C de même dimension $N \times N$. On note i l'index associé à la ligne $i+1$ et j l'index associé à la colonne $j+1$ (avec $0 \leq i < N$ et $0 \leq i < N$).
On souhaite réaliser un algorithme permettant de retourner C pour que chaque élément $C[i,j]$ de C vérifie $C[i,j] = A[i,j] + B[i,j]$:

Avec boucles for	Avec la notation vectorielle

`A=np.array([[1,2,3],[4,5,6],[7,8,9]])`
`B=np.array([[9,8,7],[6,5,4],[3,2,1]])`
`C=np.zeros((3,3))`
`for i in range (3):`
`  for j in range (3):`
`  C[i,j]=A[i,j]+B[i,j]`
	`C[:]=A[:]+B[:]`

La complexité temporelle $T(N)$ de ces deux algorithmes est en $O(N^2)$ cependant les temps de calculs ne sont pas les mêmes (la version vectorielle étant optimisée).





Commenté [AM1]: Les séquences ou structures de données rencontrées depuis le 1e semestre sont :
-Les chaînes et les tuples (qui sont immuables : les valeurs stockées dans ces séquences ne peuvent être modifier par affectation après création- pour les chaînes une modification par la méthode replace est possible mais conduit à la création d'une nouvelle chaîne avec une autre adresse mémoire)
-Les listes (muables, on dit aussi variables)

Commenté [AM2]: Attention entier ≠ flottant !
Si on créée un tableau d'entier et que l'on souhaite :
-remplacer un élément par un flottant alors le flottant est converti en entier
-remplacer un élément par une chaîne alors l'interpréteur renvoie une erreur

Commenté [AM3]: Ce type de format est typiquement celui rencontré dans les tableaux associés aux valeur affectées aux pixels des photos. Attention aux éventuels problèmes d'overflow !

Commenté [AM4]: Autrement appelé découpage ou slicing

Commenté [AM5]: Ces résultats mettent en évidence le défaut principal du langage interprété : à chaque itération l'interpréteur « dialogue » avec la « machine » ce qui représente un coût temporel