

Vecteurs - Partie II

Rappels

Pour définir un vecteur, il faut importer la bibliothèque `numpy`.

On peut alors définir un petit vecteur "à la main" : `V = np.array([x1,...,xn])` ou bien définir des vecteurs particuliers avec `np.arange`, `np.linspace`, `np.zeros...`

Opérations sur les vecteurs

L'intérêt des vecteurs par rapport aux listes est la possibilité d'effectuer facilement des opérations sur leurs coefficients :

Opérations "coefficient par coefficient"

Soient $U = [u_0, u_1, \dots, u_n]$ et $V = [v_0, v_1, \dots, v_n]$ des vecteurs de même taille. Soit a un nombre réel.

- $U + a$ donne le vecteur : $[u_0 + a, u_1 + a, \dots, u_n + a]$
- $a * U$ donne le vecteur : $[a \cdot u_0, a \cdot u_1, \dots, a \cdot u_n]$
- $U + V$ donne le vecteur : $[u_0 + v_0, u_1 + v_1, \dots, u_n + v_n]$
- $U * V$ donne le vecteur : $[u_0 v_0, u_1 v_1, \dots, u_n v_n]$
- U / V donne le vecteur : $[\frac{u_0}{v_0}, \frac{u_1}{v_1}, \dots, \frac{u_n}{v_n}]$
- Enfin, si f est n'importe quelle fonction numérique, $f(U)$ donne le vecteur : $[f(u_0), f(u_1), \dots, f(u_n)]$

Exemples : $U ** 3 = [u_0^3, \dots, u_n^3]$, $3 ** U = [3^{u_0}, \dots, 3^{u_n}]$,
`np.sin(U)` = $[\sin(u_0), \dots, \sin(u_n)]$, etc...

Exercice 1

Définir dans la console les vecteurs :

`U = [0., 2., 3., -1.]` et `V = [2., 1., -1., 3.]`

Qu'affichent les instructions suivantes ?

```
print(U-3) : ..... print(U+2*V) : .....
print(U*V) : ..... print(U/V) : .....
print(U**2) : ..... print(1/V) : .....
```

Remarque 1

On rencontrera parfois des problèmes liés au type en voulant effectuer certaines opérations. Pour cette raison, il est souvent bon d'éviter les valeurs entières 1, 2, 3 ... (type `int`) et de préférer l'écriture 1., 2., 3. (type `float`).

Exemple : L'opération `U**(-1)` renvoie un message d'erreur pour le vecteur `U = [1,2,3]` mais fonctionne sans problème pour le vecteur `U = [1., 2., 3.]`

Méthode : Construire un vecteur avec les coefficients souhaités (#2)

- 1 Construire un premier vecteur A "particulier" à l'aide de `arange` ou de `linspace`. (Bien souvent, on construira $A = [1, 2, \dots, n]$ ou $[0, 1, \dots, n]$)
- 2 Créer le vecteur $V = f(A)$ en choisissant l'opération f pour avoir les coefficients voulus.

Exercice 2

Créer, avec des instructions très simples, le vecteur $V = [1, 2, 4, 8, 16, 64, 128]$.

```
import numpy as np

A = ..... ; V = .....
```

On peut appliquer cette nouvelle méthode aux vecteurs de l'Exercice 4 du TP précédent :

Exercice 3

1. Pour tout $n \in \mathbb{N}$, soit $u_n = \frac{n}{n+2}$.

Compléter le programme suivant pour qu'il affiche le vecteur $U = [u_0, u_1, \dots, u_9]$

```
import numpy as np

A = .....
U = .....

print(U)
```

2. Pour $n \in \mathbb{N}^*$ soit $v_n = \frac{\ln(n)}{n}$. Compléter la fonction suivante pour qu'elle renvoie un vecteur contenant les n premiers termes de la suite v .

```
import numpy as np

def preterm(n) :

    A = .....

    V = .....

    return(V)
```

Somme, produit, min, max, pour un vecteur (ou une liste)

📖 Sommes, produits, min, max des coefficients

Après avoir importé `numpy` as `np`, si `U` est un vecteur ou une liste :

- `np.sum(U)` donne la somme des coefficients de `U`.
- `np.prod(U)` donne le produit des coefficients de `U`.
- `np.min(U)` donne le minimum des coefficients de `U`.
- `np.max(U)` donne le maximum des coefficients de `U`.
- `np.mean(U)` donne la moyenne des coefficients de `U`.

✎ Exercice 4

Définir dans la console le vecteur (ou la liste) `U = [2, -5, 6, 3, 3, 1]`.

Qu'affichent les instructions suivantes ?

`np.sum(U)` : `np.prod(U)` : `np.min(U)` : `np.mean(U)` :

Ceci nous donne un nouveau moyen de calculer facilement des sommes et des produits :

✎ Exercice 5

On veut écrire une fonction qui calcule et renvoie la valeur de la somme $\sum_{k=1}^n \frac{1}{k^2}$.

1. Méthode "traditionnelle" avec une boucle `for` :

```
def somme1(n) :  
    S = ...  
    for k in ..... :  
        S = .....  
    return(S)
```

2. Méthode rapide en construisant un vecteur approprié :

```
import numpy as np  
def somme2(n) :  
    A = ..... ; V = .....  
    return np.sum(V)
```

3. Méthode "en une ligne", en construisant une liste appropriée :

```
import numpy as np  
def somme3(n) :  
    return .....
```

✎ Exercice 6

Sans utiliser de boucle `for`, définir une fonction `factorielle` qui prend en entrée un entier $n \in \mathbb{N}$ et renvoie la valeur de $n!$.