

# Vecteurs - Partie I

## Rappels sur les listes

Une liste (type `list`) est une variable qui prend la forme d'une succession de valeurs.

Exemple : `L = [2, 3.5, -2, 4]`

L'instruction `L[i]` permet d'accéder au  $(i + 1)$ -ème élément d'une liste L (pour l'afficher ou le modifier).

Exemple : Après avoir tapé `L[2] = 0`, la liste devient `L = [2, 3.5, 0, 4]`.

## Le type `array` et les vecteurs

Après importation de la bibliothèque `numpy`, l'instruction `np.array` permet de créer des tableaux (variable de type `array`, ou plus précisément de type `numpy.ndarray`).

En Python, on appellera "vecteur" un **tableau à une seule ligne** : `[x1, x2, ..., xn]`. (Cela peut s'identifier, mathématiquement, à un  $n$ -uplet  $(x_1, \dots, x_n) \in \mathbb{R}^n$ ).

### ■ Crée un vecteur "à la main"

Après avoir importé `numpy` as `np` :

`V = np.array([x1, ..., xn])`

Un vecteur ressemble donc énormément à une liste... mais le type n'est pas le même! Cette différence est importante : il est possible d'effectuer des "opérations" sur les vecteurs mais pas sur les listes...

### Exercice 1

Définir les variables suivantes dans la console :

```
>>> import numpy as np
>>> L = [1,2,3,4]      >>> V = np.array([1,2,3,4])
```

Qu'affichent les commandes suivantes ?

<code>print(L) : .....</code>	<code>print(V) : .....</code>
<code>type(L) : .....</code>	<code>type(V) : .....</code>
<code>L + 1 : .....</code>	<code>V + 1 : .....</code>
<code>L / 2 : .....</code>	<code>V / 2 : .....</code>

Ceci explique pourquoi, dans les énoncés, on préférera souvent les vecteurs aux listes.

## Créer des vecteurs particuliers

### ■ Equivalent de `range` pour les vecteurs : `arange`

Après avoir importé `numpy` as `np` :

- `np.arange(n)` crée un vecteur contenant les entiers successifs de 0 à  $n - 1$ .

Exemple : `U = np.arange(4)` stocke `[0, 1, 2, 3]` dans la variable U.

- `np.arange(a,b)` crée un vecteur contenant les entiers successifs de  $a$  à  $b - 1$ .

Exemple : `V = np.arange(5,10)` stocke `[5, 6, 7, 8, 9]` dans la variable V.

- `np.arange(a,b,r)` crée un vecteur en progression arithmétique de raison `r`, de premier terme `a` et dont le dernier terme de dépasse et n'atteint pas `b`.

Exemple : `W = np.arange(4,13,3)` stocke `[4, 7, 10]` dans la variable W.

### ▲ Attention !

C'est bien "`arange`" (le "a" de `array` suivi `range`) et non pas "`arrange`"!

### ■ Exercice 2

1. Quelle instruction permet de créer le vecteur `V = [-4, -3, -2, -1, 0, 1]` ?

2. Quelle instruction permet de créer le vecteur `W = [0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1]` ?

### ■ Crée d'autres vecteurs particuliers

Après avoir importé `numpy` as `np` :

- `np.linspace(a,b,n)` crée un vecteur contenant  $n$  points uniformément répartis entre `a` et `b` (inclus!).

Exemple : `V = np.linspace(1,2,5)` crée 5 points répartis entre 1 et 2 :

`V = [1, 1.25, 1.5, 1.75, 2]`

- `np.zeros(n)` crée un vecteur contenant  $n$  zéros.

Exemple : `V = np.zeros(4)` crée `V = [0, 0, 0, 0]`.

- `np.ones(n)` crée un vecteur contenant  $n$  uns.

Exemple : `V = np.ones(5)` crée `V = [1, 1, 1, 1, 1]`.

### Exercice 3

1. Quelle instruction permet de créer le vecteur  $V = [0,0,0,0,0,0,0]$  ?

2. Recréer le vecteur  $W$  de l'Exercice 2 avec la commande `linspace`.

## Extraction et modification de coefficients

### Accéder à un coefficient

Si  $V$  est un vecteur, `V[i]` permet d'accéder à la  $(i+1)$ -ème coordonnée de  $V$ .

On peut ainsi l'afficher : `print(V[i])` ou bien la modifier : `V[i] = a`.

### Attention !

Comme toujours, le premier coefficient d'un vecteur correspond à l'indice 0.

### Méthode : Construire un vecteur avec les coefficients souhaités

1 Créer un vecteur de la bonne "taille", contenant uniquement des zéros.

2 Remplir petit à petit les coefficients à l'aide d'une boucle `for`.

### Exercice 4

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

(a) Compléter le programme suivant pour qu'il affiche un vecteur  $U$  contenant les 10 premiers termes de la suite.

```
import numpy as np

U = .....
for k in ..... :
    U[k] = .....
print(U)
```

(b) Rappeler l'instruction permettant, en une seule ligne, de créer la liste  $L = [\frac{0}{0+2}, \frac{1}{1+2}, \dots, \frac{9}{9+2}]$  :

`L =`

Pour créer le vecteur  $U$  voulu, on aurait donc aussi pu poser directement :

`U =`

2. Pour tout  $n \in \mathbb{N}^*$  soit  $v_n = \frac{\ln(n)}{n}$ .

(a) Compléter la fonction suivante pour qu'elle renvoie un vecteur contenant les  $n$  premiers termes de la suite  $v$ .

```
import numpy as np

def premterm(n) :
    V = .....
    for k in ..... :
        V[k] = .....
    return(V)
```

(b) Autrement, pour construire  $V$ , on aurait pu poser directement :

`V =`

### Exercice 5

1. On considère la suite récurrente :  $u_0 = 1/2$  et  $\forall n \in \mathbb{N}$ ,  $u_{n+1} = \sqrt{u_n}$ .

Adapter le programme suivant pour qu'il affiche le vecteur  $U = [u_0, u_1, \dots, u_{20}]$

```
import numpy as np; n = 20

U = ..... ; U[0] = .....

for k in ..... :
    U[k] = .....

print(U)
```

2. On considère la suite de Fibonacci :  $v_0 = 0$ ,  $v_1 = 1$  et  $\forall n \in \mathbb{N}$ ,  $v_{n+2} = v_n + v_{n+1}$ . Compléter la fonction `fibo` suivante pour qu'elle affiche le vecteur  $V = [v_0, v_1, \dots, v_n]$  quand on lui fournit un  $n \in \mathbb{N}$ .

```
import numpy as np

def fibo(n) :

    V = ..... ; V[0] = .... ; V[1] = ....

    for k in ..... :
        V[k] = .....

    return(V)
```