[Méthode] Numpy

Numpy est une bibliothèque Python fournissant des outils de calcul numérique. La bibliothèque est habituellement appelée par la commande :

```
import numpy as np
```

L'objet de base est un **tableau** de valeurs (*nd.array*) qui a des caractéristiques proches d'une liste, mais de manipulation plus simple.

Création de tableau

Il est nécessaire de créer un tableau avec des valeurs initiales, contrairement à une liste.

Ces valeurs peuvent être complexes (avec la notation j). De plus π et e sont définis par **np.pi** et **np.e**.

```
a=np.array([1,2,3,4]) # tableau 1D de valeurs
a=np.array([[1,2,3],[4,5,6]]) # tableau 2D de valeurs
a=np.zeros(n) # tableau contenant n 0
a=np.ones(n) # tableau contenant n 1
a=np.arange(n1,n2,p) # comme list(range(n1,n2,p)) pour les tableaux
a=np.linspace(x1,x2,n) # tableau de n valeurs reparties lineairement entre x1 et x2
a=np.logspace(x1,x2,n) # tableau de n valeurs reparties logarithmiquement entre x1 et
    x2
```

Pour sélectionner une portion de données dans un tableau, le *slicing* fonctionne de la même manière que pour les listes.

Actions sur un tableau

Toute opération sur un tableau se fait directement sur l'ensemble des valeurs du tableau, terme à terme.

Par exemple : **np.array([1,2,3])+np.array([4,5,6])** donne **array([5,7,9])**.

Remarque : ceci est différent des listes, qui se concatènent.

En plus des opérations arithmétiques de base (+−*/ et ** pour une exponentiation), *numpy* fournit des fonctions mathématiques (avec des noms sans ambiguité).

- ❏ fonctions usuelles :

```
np.sqrt(), np.exp(), np.log(), np.log10()
```

- ❏ fonctions trigonométriques (ajout d'un h pour les fonctions hyperboliques) :

```
np.sin(), np.cos(), np.tan(), np.arcsin(), np.arccos(), np.arctan()
```

- ❏ fonctions spécifiques aux complexes :

```
np.real(), np.imag(), np.abs(), np.angle()
```

- ❏ fonctions statistiques :

```
np.average(a) # moyenne des elements de a
np.std(a,ddof=1) # ecart-type des elements de a
```

- ❏ Autres :

```
np.abs() # valeur absolue
np.sign() # signe
np.around() # arrondi a l'entier le plus proche
np.trunc() # partie entiere
```