Introduction à Numpy

Par

OUHAB Abdelkrim

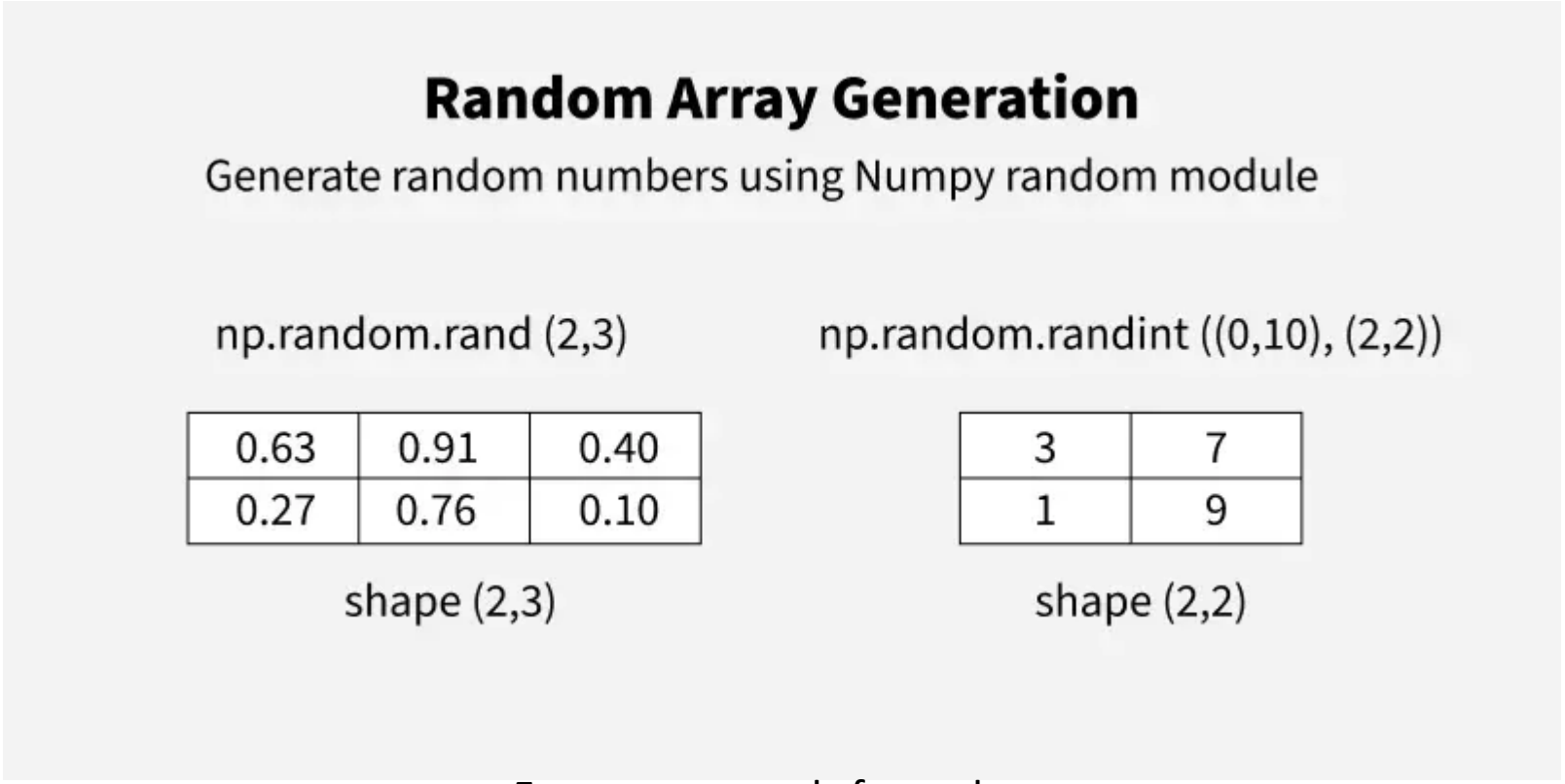
Bibliothèque NumPy

- NumPy est une bibliothèque Python essentielle pour le calcul numérique
- Elle manipule des tableaux à N dimensions nommées *ndarray*
- Elle est conçue pour manipuler efficacement des tableaux volumineux
- Elle est nettement plus rapide que les listes intégrées de Python
- Les opérations vectorisées dans NumPy (sans utilisation de boucles) peuvent être de 10 à 100 fois plus rapides que les boucles Python

À quoi sert NumPy ?

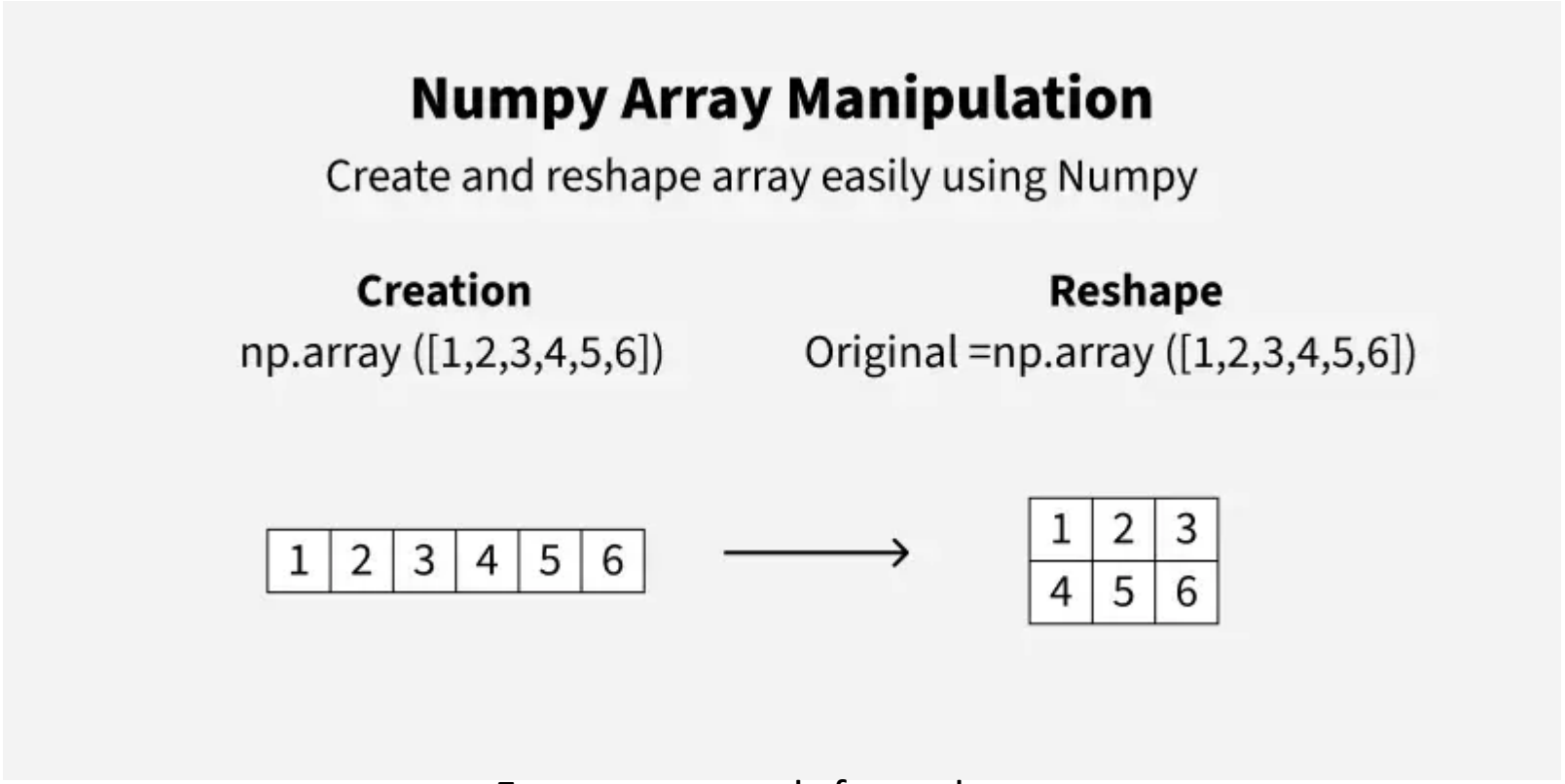
- Création et manipulation de tableaux
- Opérations entre éléments d’un tableaux et opérations matricielles
- Génération de nombres aléatoires
- Calculs statistiques
- Opérations d'algèbre linéaire
- Utilisation des transformations de Fourier
- Gestion efficace des valeurs manquantes dans les ensembles de données.

Génération de nombres aléatoires



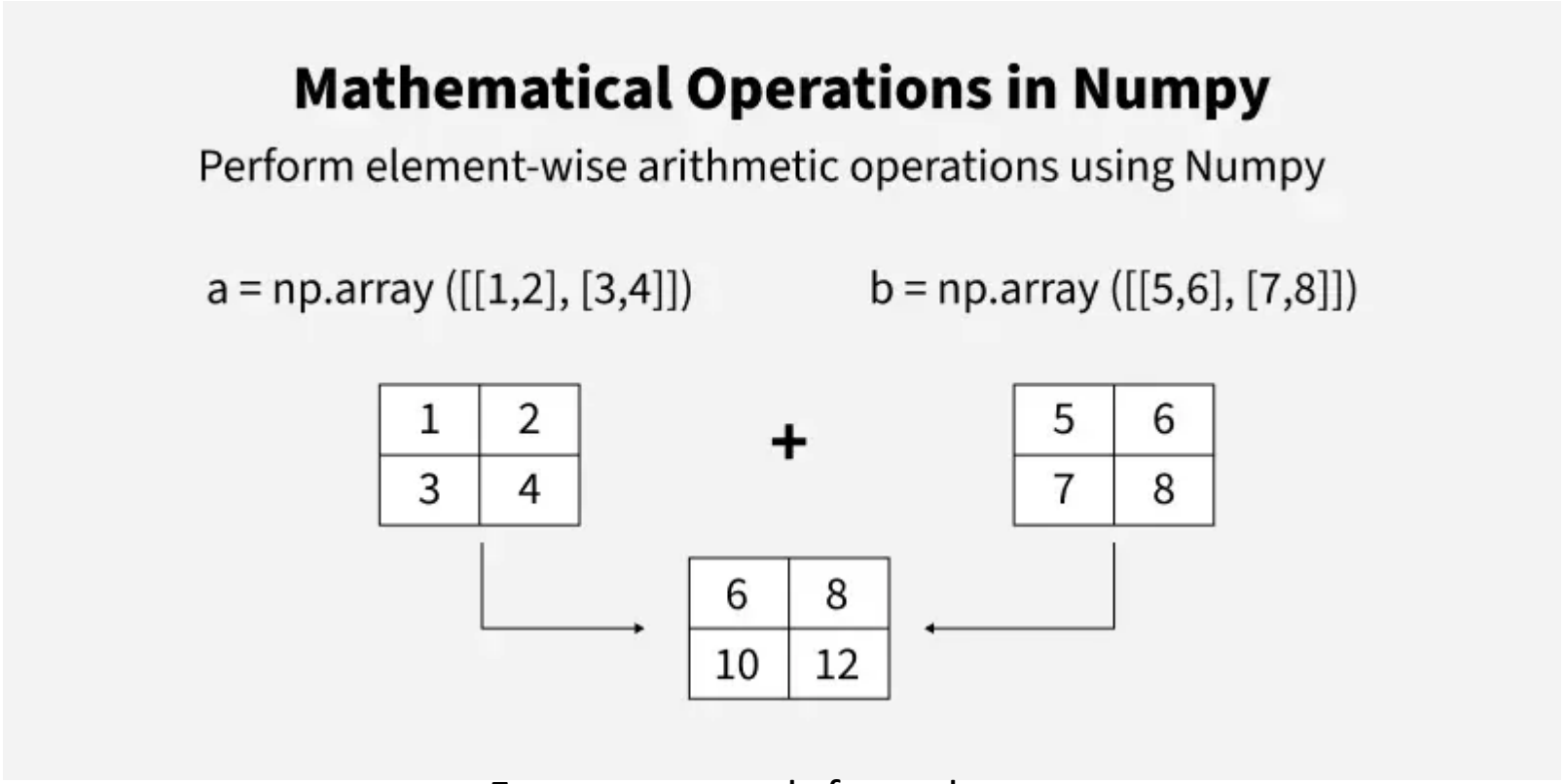
From www.geeksforgeeks.org

Création et manipulation de tableaux



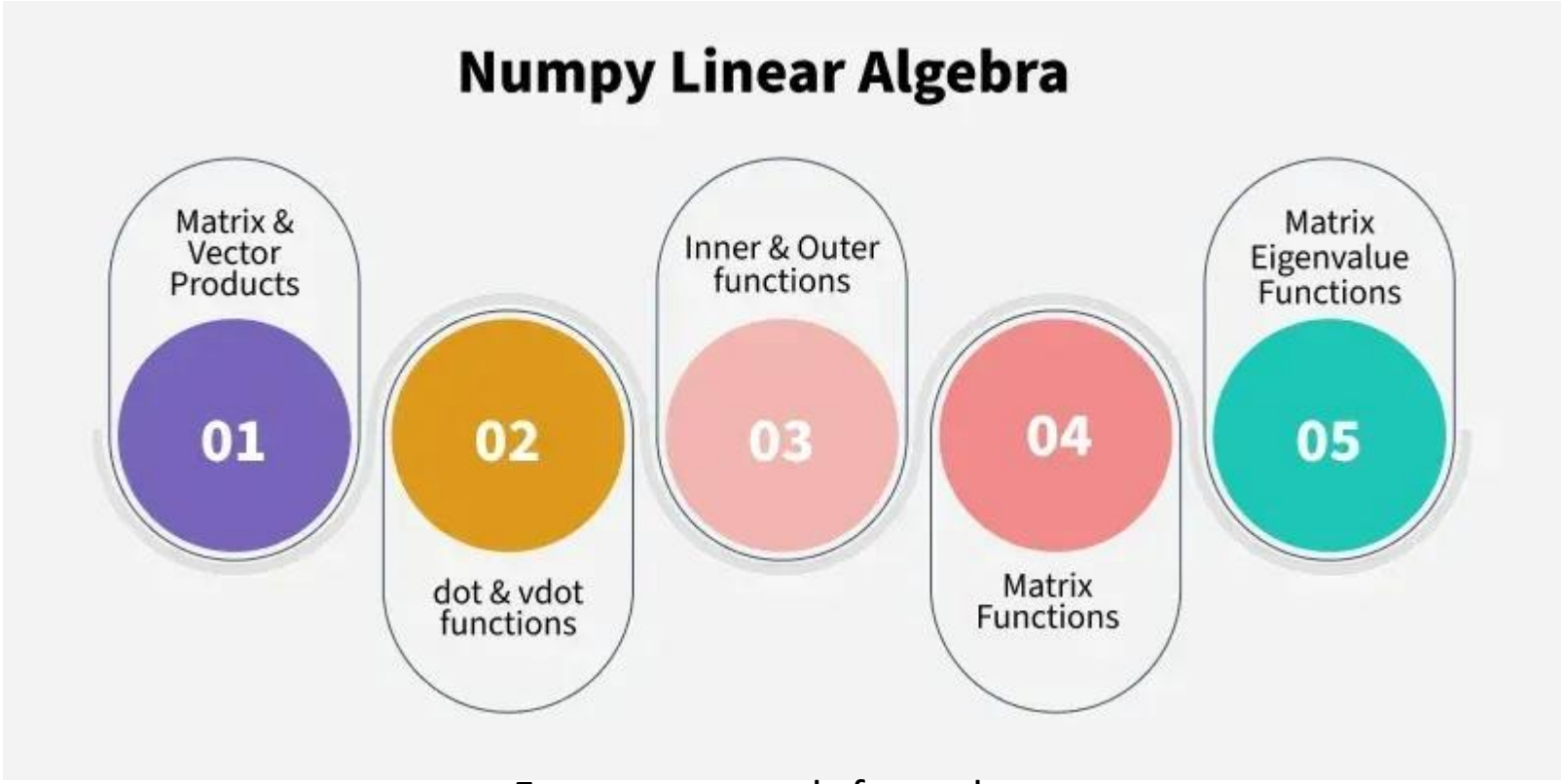
From www.geeksforgeeks.org

Opérations entre éléments d'un tableaux



From www.geeksforgeeks.org

Opérations d'algèbre linéaire



From www.geeksforgeeks.org

Utilisation de NumPy

- Installation

  pip install numpy

- Importation

  import numpy as np

Création de tableaux: Numpy.array()

❑ La fonction Numpy.array() permet de convertir des listes Python en tableaux NumPy

```
import numpy as np
# 1D array
liste1 = [1, 2, 3]
a1 = np.array(liste1)
# 2D array
liste2 =  [[1, 2], [3, 4]]
a2 = np.array(liste2)
print("a1= ",a1)
print("a2= ",a2)
```

Output

```
a1= [1 2 3]
a2= [[1 2] [3 4]]
```

Création de tableaux: np.zeros() et np.ones()

- numpy.zeros(shape, dtype = None, order = 'C'))
- numpy. ones(shape, dtype = None, order = 'C')
- **shape:** un entier unique ou un tuple
- **dtype:** facultatif, par défaut float
- **order:** {‘C’, ‘F’} spécifie l'ordre de disposition de la mémoire : C optimal pour les opérations sur les lignes, F optimal pour les opérations par colonne.

Création de tableaux: np.zeros() et np.ones()

```
import numpy as np
a1 = np.zeros(4)
a2 = np.ones(3)
a3 = np.zeros((2,3))
a4 = np.zeros((2,4))
a5 = np.zeros(4, dtype=int)
print("a1= ", a1)
print("a2= ", a2)
print("a3= ", a3)
print("a4= ", a4)
print("a5= ", a5)
```

Output

```
a1= [0. 0. 0. 0.]
a2= [1. 1. 1.]
a3= [[0. 0. 0.]
 [0. 0. 0.]]
a4= [[0. 0. 0. 0.]
[0. 0. 0. 0.]]
a5= [0 0 0 0]
```

Création de tableaux: numpy.arange()

- numpy.arange(start, stop, step, dtype)

- **start (optional):** valeur de départ par défaut est 0.
- **stop (required):** valeur finale , exclusif
- **step (optional):** espacement entre les valeurs consécutives par défaut 1
- **dtype (optional):** type de données pour les valeurs du tableau

Création de tableaux: numpy.arange()

```
import numpy as np
a1= np.arange(7) # de 0 à 6
a2= np.arange(2,7) # de 2 à 6
a3= np.arange(2,7,2) # de 2 à 6 avec espacement 2
print("a1= ", a1)
print("a2= ", a2)
print("a3= ", a3)
```

Output

```
a1= [0 1 2 3 4 5 6]
a2= [2 3 4 5 6]
a3= [2 4 6]
```

Attributs des tableaux NumPy

- **shape:** renvoie les dimensions du tableau (nombre lignes et colonnes)
- **dtype**: renvoie le type de données des éléments.
- **ndim**: renvoie le nombre de dimensions (1D, 2D, …)

```
import numpy as np
a = np.array([[1, 2, 3], [4, 5, 6]])
print(a.shape)
print(a.dtype)
print(a.ndim)
```

Output

```
(2, 3)
int64
2
```

Indexation des tableaux NumPy

- L'indexation désigne la méthode d'accès à des sous-ensembles de données au sein d'un tableau
- permet de récupérer, modifier et manipuler des données à des positions ou des intervalles précis

Tableaux 1D: accès par un indice

- Les positions des éléments d'un tableau sont appelées indices
- Les indices commencent par 0. Le premier élément est a[0], Le deuxième est a[1], …
- Les indices négatifs comptent à partir de la fin. Le dernier élément est a[-1], l'avant dernier est [-2], …

Tableaux 1D: accès par un indice

```
import numpy as np
a = np.array([10, 20, 30, 40, 50])
print("Premier élément = ",a[0])
print("Deuxième élément = ",a[1])
print("Dernier élément = ",a[-1])
print("Avant dernier élément = ",a[-2])
```

Output

```
Premier élément = 10
Deuxième élément = 20
Dernier élément = 50
Avant dernier élément = 40
```

Tableaux 1D: accès par plusieurs indices

```
import numpy as np
a = np.array([10, 20, 30, 40, 50])
indices = [0, 2, 4]
print(a[indices])
```

Output

```
[10 30 50]
```

Tableaux 1D: accès par intervalle

- permet d'extraire une plage d'éléments en utilisant le format ***start:end:step***

```
import numpy as np
a = np.array([0, 1, 2, 3, 4, 5, 6, 7])
print("indice 1 à indice 5")
print(a[1:6])
print("indice 1 à indice 5 avec step 2")
print(a[1:6:2])
```

Output

```
indice 1 à indice 5
[1 2 3 4 5]
indice 1 à indice 5 avec step 2
[1 3 5]
```

Tableaux 1D: utilisation de conditions

- Permet d'extraire des éléments en utilisant une ou plusieurs conditions

```
import numpy as np
a = np.array([40, 15, 10, 25, 30, 25])
print("éléments supérieurs à 20 ")
print(a[a > 20])
```

Output

```
éléments supérieurs à 20
[40 25 30 25]
```

Tableaux 2D: accès par indices

- matrice[ligne, colonne]

```
import numpy as np
m = np.array([[1, 2, 3], [4, 5, 6]])
print(m[0, 0])
print(m[0, 1])
print(m[1, 2])
```

Output

```
1
2
6
```

Modification d'éléments

```
import numpy as np
a = np.array([0, 1, 2, 3])
a[2] = 20
print("a = ",a)
a[1:3] = 100
print("a = ",a)
```

Output

```
a =  [ 0 1 20 3]
a =  [ 0 100 100 3]
```