

Python pour la Data Science



Section 1 : Remise à niveau rapide de Python



Section 2: La Data Science avec Python



Section 3: Python Pandas DataFrames et Séries



Section 4: Nettoyage de données

DATA SCIENCE AVEC PYTHON



Remise à niveau
rapide de Python

Data Science avec
Python

**Installation des
bibliothèques**

1

2

Installation de
Python

3

C'est quoi Jupyter ?

4

Installation d'Anaconda
sur Windows

5

Installation d'Anaconda
sur Mac OS

6

Installation d'Anaconda
Sur Ubuntu

7

Comment implémenter
Python dans Jupyter?

8

Gestion des répertoires
dans Jupyter Notebook

9

Entrée/sortie

10

Travailler avec différents
types de données

11

Variables

12

Opérateurs arithmétiques



Installation des bibliothèques

Si vous avez installé Anaconda, vous n'avez pas besoin de télécharger de bibliothèques, car il installe automatiquement toutes les bibliothèques de science des données(data science) les plus populaires, telles que Pandas, Numpy, Matplotlib, Seaborn, etc..



Remise à niveau
rapide de Python

Data Science avec
Python

Installation des
bibliothèques

1

2

Importation de
bibliothèques

3

Bibliothèque Pandas
pour la Data Science

4

Installation d'Anaconda
sur Windows

5

Installation d'Anaconda
sur Mac OS

6

Installation d'Anaconda
Sur Ubuntu

7

Comment implémenter
Python dans Jupyter?

8

Gestion des répertoires
dans Jupyter Notebook

9

Entrée/sortie

10

Travailler avec différents
types de données

11

Variables

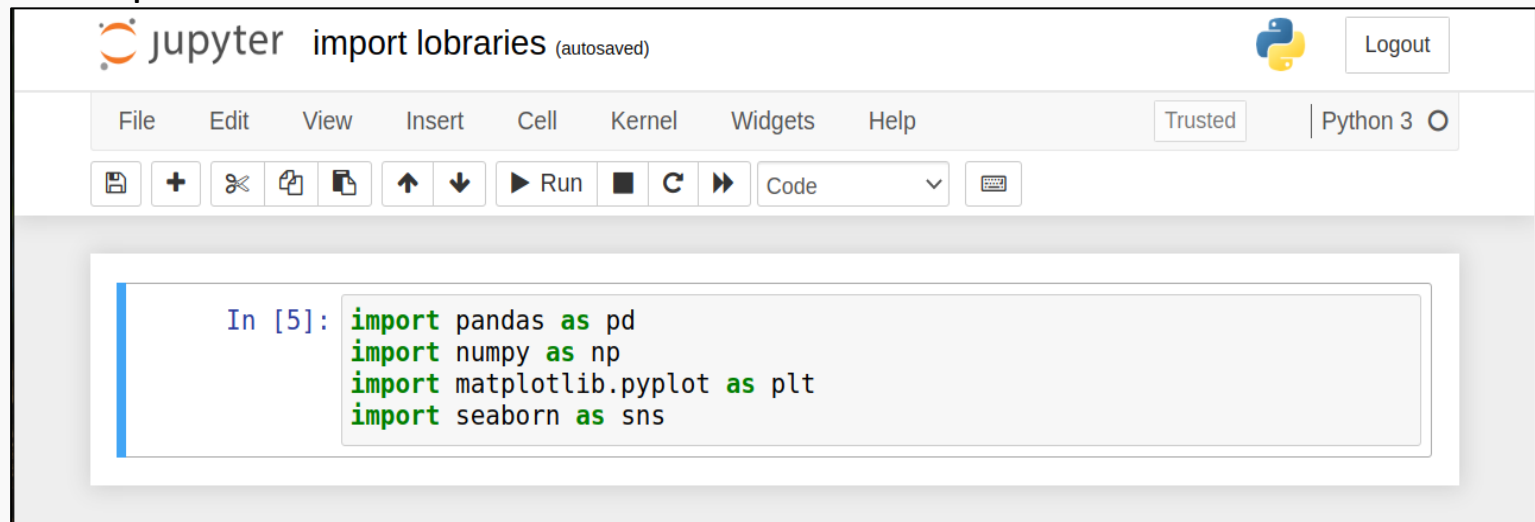
12

Opérateurs arithmétiques



Importation de bibliothèques

- Ouvrez votre Jupyter Notebook.
- Pour importer une bibliothèque, nous utilisons le mot-clé **import** suivi du nom de la bibliothèque.
- Nous pouvons utiliser le mot-clé **as** pour utiliser des abréviations pour nos noms de bibliothèques.
- Les abréviations courantes utilisées sont
 - pd for pandas
 - np pour numpy
 - plt pour matplotlib.pyplot
 - sns pour seaborn



The screenshot shows a Jupyter Notebook window titled "import lbraries (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for saving, adding, deleting, and running code, and a code editor. The code editor contains the following Python code:

```
In [5]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```



Remise à niveau
rapide de Python

Data Science avec
Python

Installation des
bibliothèques

1

2

Importation de
bibliothèques

3

Bibliothèque Pandas
pour la Data Science

4

Installation d'Anaconda
sur Windows

Installation d'Anaconda
sur Mac OS

5

6

Installation d'Anaconda
Sur Ubuntu

Comment implémenter
Python dans Jupyter?

7

8

Gestion des répertoires
dans Jupyter Notebook

Entrée/sortie

9

10

Travailler avec différents
types de données

Variables

11

12

Opérateurs arithmétiques



Bibliothèque Pandas pour la Data Science

- Pandas est une bibliothèque Python pour la manipulation et l'analyse de données.
- Elle permet d'explorer, de nettoyer et de traiter des données tabulaires.
- Elle offre deux façons de stocker les données ;
 - les séries, qui sont des structures de données unidimensionnelles
 - Data Frame, qui est une structure de données bidimensionnelle

DataFrame

	nom	calories	protéine	vitamines	note
0	Bran à 100%.	70	4	25	68.402973
1	Bran 100% naturel	120	3	0	33.983679
2	Tout-Bran	70	4	25	59.425505
3	Tout-Bran avec fibres supplémentaires	50	4	25	93.704912
4	Délice d'amandes	110	2	25	34.384843
5	Cheerios pomme-cannelle	110	2	25	29.509541
6	Pomme Jacks	110	2	25	33.174094
7	Basic 4	130	3	25	37.038562
8	Boulettes de Bran	90	2	25	49.120253
9	Flocons de Bran	90	3	25	53.313813

```
0    70
1   120
2    70
3    50
4   110
5   110
6   110
7   130
8    90
9    90
Nom    calories, dtype: int64
```

Séries



Remise à niveau
rapide de Python

Data Science avec
Python

Installation des
bibliothèques

1

2 Importation de
bibliothèques

2

Bibliothèque Pandas
pour la Data Science

3

4 Bibliothèque NumPy
pour Data Science

4

Installation d'Anaconda
sur Mac OS

5

6 Installation d'Anaconda
Sur Ubuntu

6

Comment implémenter
Python dans Jupyter?

7

8 Gestion des répertoires
dans Jupyter Notebook

8

Entrée/sortie

9

10 Travailler avec différents
types de données

10

Variables

11

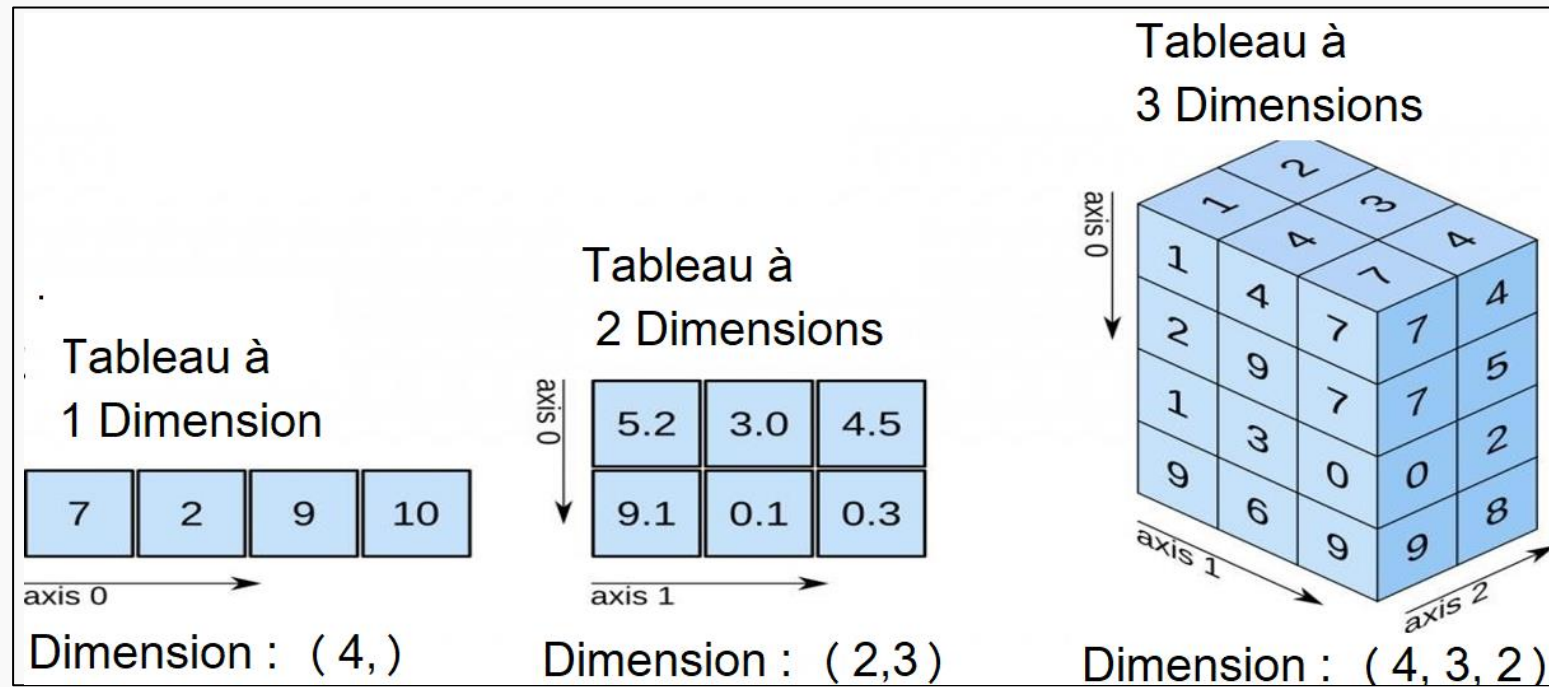
12 Opérateurs arithmétiques

12



Bibliothèque NumPy pour Data Science

- NumPy est l'abréviation de Numerical Python.
- Il fournit une structure de données appelée tableau NumPy, qui est une grille de valeurs.
- Il fournit également une collection de fonctions mathématiques de haut niveau qui peuvent être exécutées sur des tableaux NumPy multidimensionnels.





Remise à niveau
rapide de Python

Data Science avec
Python

Installation des
bibliothèques

1

2

Importation de
bibliothèques

3

Bibliothèque Pandas
pour la Data Science

4

Bibliothèque NumPy
pour Data Science

Pandas vs NumPy

5

6

Installation d'Anaconda
Sur Ubuntu

Comment implémenter
Python dans Jupyter?

7

8

Gestion des répertoires
dans Jupyter Notebook

Entrée/sortie

9

10

Travailler avec différents
types de données

Variables

11

12

Opérateurs arithmétiques



Pandas vs NumPy

NumPy	Pandas
NumPy and Pandas sont deux bibliothèques Python pour la Data Science	
Il est utilisé pour le calcul scientifique	Il est utilisé pour la manipulation des données, comme le stockage, l'exploration, le nettoyage et le traitement des données.
Il fournit des tableaux NumPy qui peuvent être multidimensionnels	Il fournit deux structures de données; <ul style="list-style-type: none">• Séries (unidimensionnelles)• Cadres de données (bidimensionnels)
Nous utilisons Pandas pour la manipulation des données et NumPy pour les calculs mathématiques.	
Puisque les séries Pandas et les Data Frames peuvent être considérées comme des tableaux NumPy unidimensionnels et bidimensionnels respectivement, nous pouvons également leur appliquer des fonctions mathématiques NumPy.	



Remise à niveau
rapide de Python

Data Science avec
Python

Installation des
bibliothèques

1

2

Importation de
bibliothèques

3

Bibliothèque Pandas
pour la Data Science

4

Bibliothèque NumPy
pour Data Science

Pandas vs NumPy

5

6

Bibliothèque Matplotlib
pour Data Science

Comment implémenter
Python dans Jupyter?

7

8

Gestion des répertoires
dans Jupyter Notebook

Entrée/sortie

9

10

Travailler avec différents
types de données

Variables

11

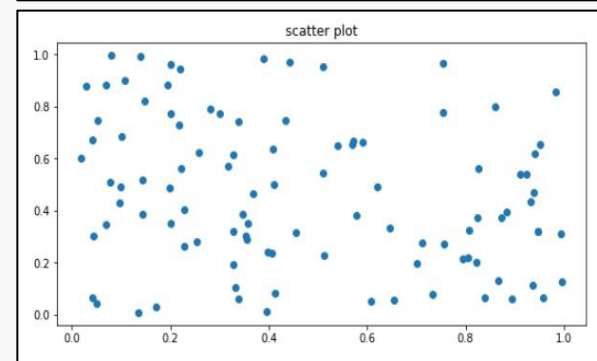
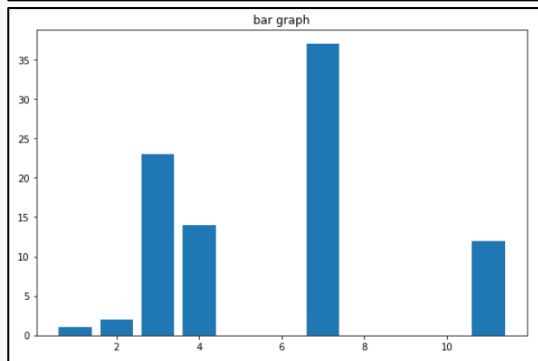
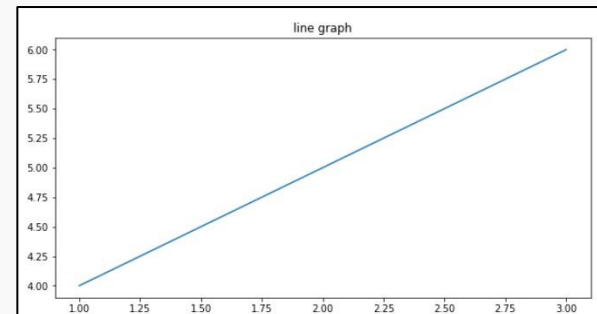
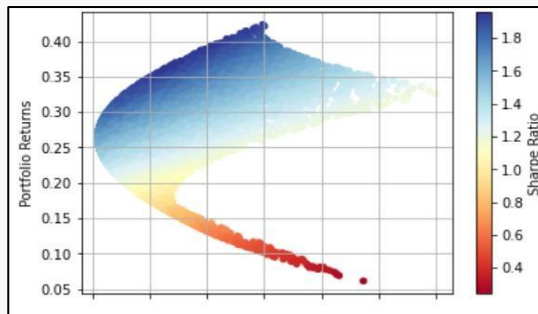
12

Opérateurs arithmétiques



Bibliothèque Matplotlib pour Data Science

- Matplotlib est une bibliothèque Python de visualisation, c'est-à-dire qu'elle est utilisée pour tracer des graphiques.
- Le module pyplot à l'intérieur de Matplotlib fournit l'interface à la fonctionnalité de traçage sous-jacente de Matplotlib.
- Matplotlib permet de créer un certain nombre de types de graphiques différents, tels que des histogrammes, des diagrammes de dispersion, des diagrammes de surface, des camemberts, etc.





Remise à niveau
rapide de Python

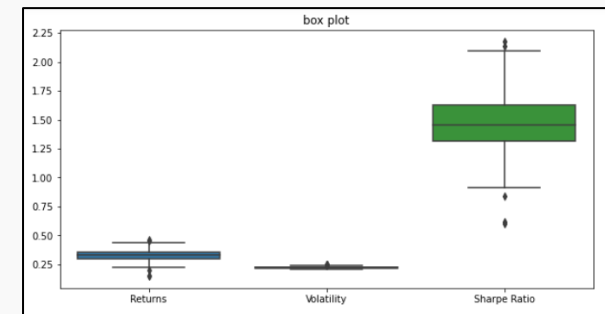
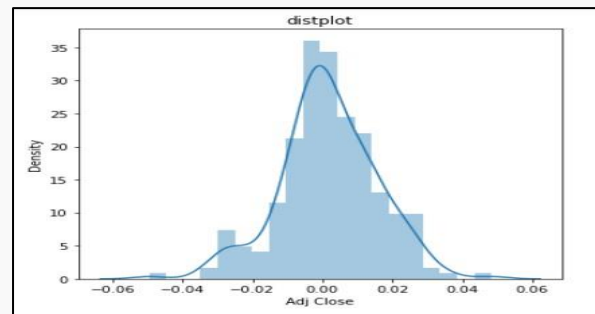
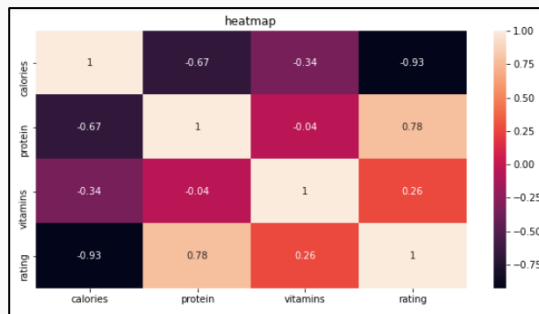
Data Science avec
Python

Installation des bibliothèques	1	
	2	Importation de bibliothèques
Bibliothèque Pandas pour la Data Science	3	
	4	Bibliothèque NumPy pour Data Science
Pandas vs NumPy	5	
	6	Bibliothèque Matplotlib pour Data Science
Bibliothèque Seaborn pour Data Science	7	
	8	Gestion des répertoires dans Jupyter Notebook
Entrée/sortie	9	
	10	Travailler avec différents types de données
Variables	11	
	12	Opérateurs arithmétiques



Bibliothèque Seaborn pour Data Science

- Seaborn est une autre bibliothèque Python de visualisation construite au-dessus de Matplotlib.
- Elle étend les fonctionnalités de Matplotlib et permet de créer une variété de graphiques différents avec moins de syntaxe.





Remise à niveau
rapide de Python

Data Science avec
Python

Installation des bibliothèques	1	
	2	Importation de bibliothèques
Bibliothèque Pandas pour la Data Science	3	
	4	Bibliothèque NumPy pour Data Science
Pandas vs NumPy	5	
	6	Bibliothèque Matplotlib pour Data Science
Bibliothèque Seaborn pour Data Science	7	
	8	Tableaux NumPy
Entrée/sortie	9	
	10	Travailler avec différents types de données
Variables	11	
	12	Opérateurs arithmétiques



Tableaux NumPy

C'est quoi les tableaux NumPy ?

Le tableau NumPy est une structure de données multidimensionnelle conçue pour gérer facilement de grands ensembles de données.

- Un tableau NumPy est appelé **ndarray**.
- Nous pouvons trouver le nombre de dimensions d'un tableau NumPy en utilisant **.ndim**.

Tableaux NumPy et listes Python

- Les tableaux NumPy offrent plus de fonctionnalités intégrées que les listes Python.
- Les tableaux NumPy permettent de travailler plus facilement avec d'énormes ensembles de données multidimensionnelles avec moins de syntaxe.
- Les tableaux NumPy sont également plus efficaces que les listes Python en termes d'occupation de la mémoire et de vitesse.



Création de tableaux NumPy (1/3)

Tableaux 1 dimension

- Un tableau NumPy à 1 dimension est un tableau où chaque élément du tableau le plus extérieur est un tableau à 0 dimension (scalaire).
- Nous pouvons créer un tableau NumPy à l'aide de la fonction `array()` de la bibliothèque NumPy.
- Nous pouvons créer un tableau NumPy en utilisant des listes ou des n-uplets Python.
- Pour créer un tableau NumPy à 1 dimension, nous fournissons une liste ou un n-uplet Python à la fonction `array()`.

```
jupyter 1-D Numpy Array Last Checkpoint: an hour ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
[1]: import numpy as np
      Tableau1D = np.array([1, 2, 3, 4, 5])
      print(Tableau1D)
      [1 2 3 4 5]
[2]: print(Tableau1D.ndim)
      1
[ ]: |
```



Création de tableaux NumPy (2/3)

Tableaux NumPy à 2 dimensions

- Un tableau NumPy à 2 dimensions est un tableau où chaque élément du tableau le plus extérieur est un tableau à 1 dimension.
- Pour créer un tableau NumPy à 2 dimensions, nous fournissons une liste ou un n-uplet Python à la fonction `array()`.

The image shows a Jupyter Notebook interface with the title "2-D Numpy Array". The notebook contains two code cells. The first cell, labeled [5], imports numpy as np and creates a 2D array named Tableau2D using np.array([[1, 2, 3], [4, 5, 6]]). The output of this cell is a 2x3 array: [[1 2 3], [4 5 6]]. The second cell, labeled [6], prints Tableau2D.ndim, which outputs 2, indicating the array has 2 dimensions.

```
jupyter 2-D Numpy Array Last Checkpoint: a few seconds ago (autosaved) Python 3
```

```
[5]: import numpy as np
Tableau2D = np.array([[1, 2, 3], [4, 5, 6]])
print(Tableau2D)

[[1 2 3]
 [4 5 6]]

[6]: print(Tableau2D.ndim)

2
```



Création de tableaux NumPy (3/3)

Tableaux NumPy à 3 dimensions

- Un tableau NumPy à 3 dimensions est un tableau où chaque élément du tableau le plus extérieur est un tableau à 2 dimensions.
- Pour créer un tableau NumPy à 3 dimensions, nous fournissons une liste ou un n-uplet Python à la fonction `array()`.

```
jupyter 3-D Numpy Array Last Checkpoint: 3 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
[7]: import numpy as np
      Tableau3D = np.array([[[1, 2, 3], [-1, -2, -3]], [[4, 5, 6], [-4, -5, -6]]])
      print(Tableau3D)

      [[ 1  2  3]
       [-1 -2 -3]]

       [[ 4  5  6]
       [-4 -5 -6]]

[8]: print(Tableau3D.ndim)

3
```



Quiz Time

1. Combien de dimensions possède le tableau `[[[1, 2, 3, 4]]]` ?
- a) 1
 - b) 2
 - c) 3



Quiz Time

1. Combien de dimensions possède le tableau [[[1, 2, 3, 4]]] ?

- a) 1
- b) 2
- c) 3



Indexation des tableaux NumPy(1/8)

Tableaux NumPy 1-D

- L'indexation d'un tableau NumPy 1-D est identique à l'indexation d'une liste Python 1-D.
- Indiquez l'index de l'élément à l'intérieur des crochets pour obtenir cet élément.

The screenshot shows a Jupyter Notebook window titled "1-D Numpy Array". The interface includes a top bar with "jupyter", the title, and a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". A toolbar contains icons for saving, adding cells, undo, redo, and running code. The notebook area displays two code cells. The first cell, labeled "[9]:", contains the code to import numpy as np, create a 1D array named "Tableau1D" with values [1, 2, 3, 4, 5], and print it. The output of this cell is "[1 2 3 4 5]". The second cell, labeled "[10]:", contains the code to print the element at index 2 of "Tableau1D". The output of this cell is "3". A third cell, labeled "[]:", is currently empty.

```
[9]: import numpy as np
Tableau1D = np.array([1, 2, 3, 4, 5])
print(Tableau1D)

[1 2 3 4 5]

[10]: print(Tableau1D[2])

3

[ ]:
```



Indexation des tableaux NumPy (2/8)

Tableaux NumPy 2-D

- Pour indexer un tableau NumPy 2-D, nous fournissons 2 valeurs à l'intérieur des crochets ([]).
 - La première valeur est l'index du tableau interne
 - La deuxième valeur est l'index de l'élément à l'intérieur du tableau interne
- Dans l'exemple suivant, nous obtenons le premier élément du second tableau.

The image shows a Jupyter Notebook interface with the title "2-D Numpy Array (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, undo, redo, and running code. The code cell contains the following Python code:

```
[11]: import numpy as np

[12]: Tableau2D = np.array([[1, 2, 3], [4, 5, 6]])
      print(Tableau2D)
```

The output of the code is a 2-D NumPy array:

```
[[1 2 3]
 [4 5 6]]
```

The first element of the second row, the value 4, is highlighted with a red square, demonstrating indexing with [1, 0].

```
[ ]:
```



Indexation des tableaux NumPy (3/8)

Tableaux NumPy 2-D

- La première dimension contient 2 tableaux.
- Si nous disons Tableau2D[1], nous obtenons le deuxième tableau

The image shows a Jupyter Notebook window titled "2-D Numpy Array (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for saving, adding cells, undo, redo, and running code, and a status bar indicating "Trusted" and "Python 3". The notebook contains three code cells:

```
[11]: import numpy as np
```

```
[12]: Tableau2D = np.array([[1, 2, 3], [4, 5, 6]])
      print(Tableau2D)
```

```
[[1 2 3]
 [4 5 6]]
```

```
[13]: print(Tableau2D[1])
```

```
[4 5 6]
```



Indexation des tableaux NumPy (4/8)

Tableaux NumPy 2-D

- La deuxième dimension contient 3 éléments.
- si nous disons `Tableau2D[1, 0]`, nous obtenons le premier élément du deuxième tableau.

The image shows a Jupyter Notebook interface with the title "2-D Numpy Array (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for saving, adding cells, undo, redo, and running code, and a status bar showing "Trusted" and "Python 3". The notebook contains two code cells. The first cell, labeled "[14]:", creates a 2D NumPy array named "Tableau2D" with the values [[1, 2, 3], [4, 5, 6]] and prints it. The output of this cell is a 2x3 array: [[1 2 3], [4 5 6]]. The second cell, labeled "[15]:", prints the element at index [1, 0] of the "Tableau2D" array. The output of this cell is the value 4.

```
[14]: Tableau2D = np.array([[1, 2, 3], [4, 5, 6]])
      print(Tableau2D)

      [[1 2 3]
       [4 5 6]]

[15]: print(Tableau2D[1, 0])

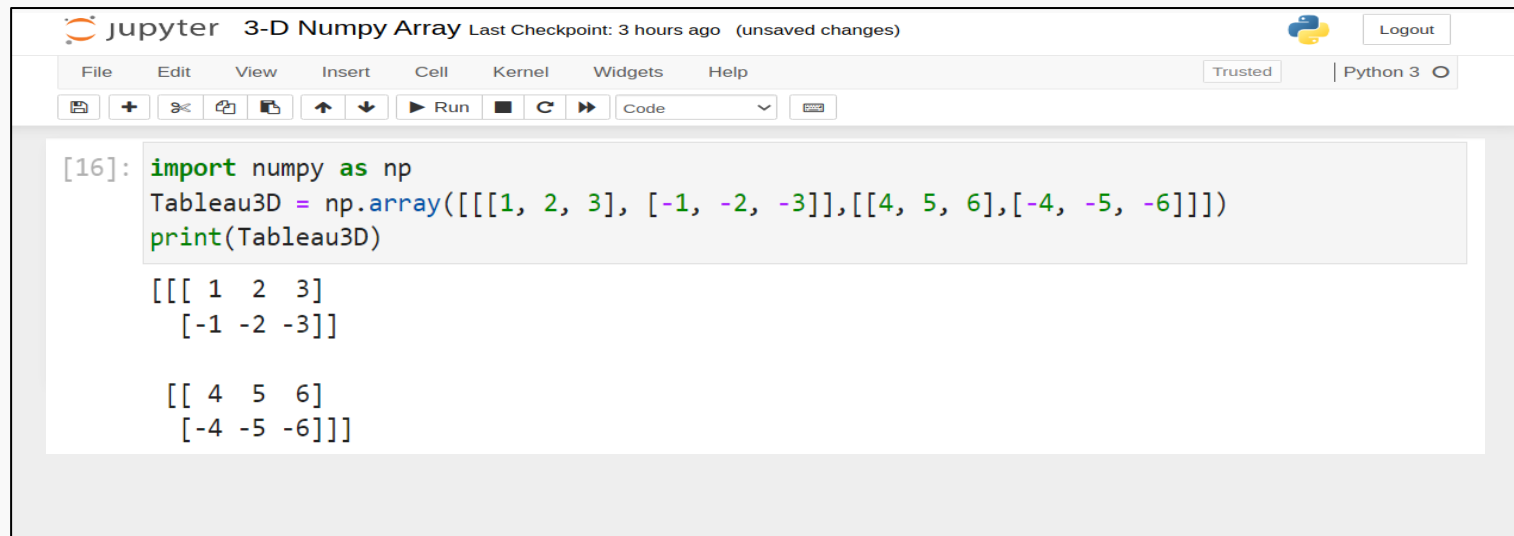
      4
```



Indexation des tableaux NumPy (5/8)

Tableaux NumPy 3-D

- Pour indexer un tableau NumPy 3-D, nous fournissons 3 valeurs à l'intérieur des crochets ([]).
 - La première valeur est l'index du tableau interne 2-D dans la première dimension.
 - La deuxième valeur est l'index du tableau interne 1-D dans la deuxième dimension.
 - La troisième valeur est l'indice de l'élément dans la troisième dimension.
- Dans l'exemple suivant, nous obtenons le premier élément du deuxième tableau du premier tableau.



```
[16]: import numpy as np
Tableau3D = np.array([[[1, 2, 3], [-1, -2, -3]], [[4, 5, 6], [-4, -5, -6]]])
print(Tableau3D)

[[[ 1  2  3]
  [-1 -2 -3]]

 [[ 4  5  6]
  [-4 -5 -6]]]
```



Indexation des tableaux NumPy (6/8)

Tableaux NumPy 3-D

- La première dimension contient 2 tableaux.
- Si nous disons Tableau3D[0], nous obtenons le premier tableau.

The image shows a Jupyter Notebook interface with the title "3-D Numpy Array" and a status bar indicating "Last Checkpoint: 3 hours ago (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and other functions. The code cell contains the following Python code:

```
[17]: import numpy as np
Tableau3D = np.array([[[1, 2, 3], [-1, -2, -3]], [[4, 5, 6], [-4, -5, -6]]])
print(Tableau3D[0])
```

The output of the code is displayed below the code cell:

```
[[ 1  2  3]
 [-1 -2 -3]]
```



Indexation des tableaux NumPy (7/8)

Tableaux NumPy 3-D

- La deuxième dimension contient à nouveau 2 tableaux.
- Si nous disons `Tableau3D[0, 1]`, nous obtenons le deuxième tableau du premier tableau.

```
jupyter 3-D Numpy Array Last Checkpoint: 3 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
[18]: import numpy as np
      Tableau3D = np.array([[[1, 2, 3], [-1, -2, -3]], [[4, 5, 6], [-4, -5, -6]]])
      print(Tableau3D[0, 1])
      [-1 -2 -3]
[ ]:
```



Indexation des tableaux NumPy (8/8)

Tableaux NumPy 3-D

- La troisième dimension contient 3 valeurs.
- Si nous disons Tableau3D[0, 1, 0], nous obtenons le premier élément du second tableau du premier tableau.

The image shows a Jupyter Notebook interface with the title "3-D Numpy Array". The notebook contains a code cell with the following Python code:

```
[19]: import numpy as np
Tableau3D = np.array([[1, 2, 3], [-1, -2, -3]], [[4, 5, 6], [-4, -5, -6]])
print(Tableau3D[0, 1, 0])
```

The output of the code is displayed below the cell:

```
-1
```

Below the output, there is an input prompt `[]:` followed by a text box for user input.



Quiz Time

1. Considérons un tableau numpy $x = [[[1, 2, 3, 4]]]$. Que donnera $x[0, 0, 0]$?
- a) 1
 - b) 3
 - c) 4



Quiz Time

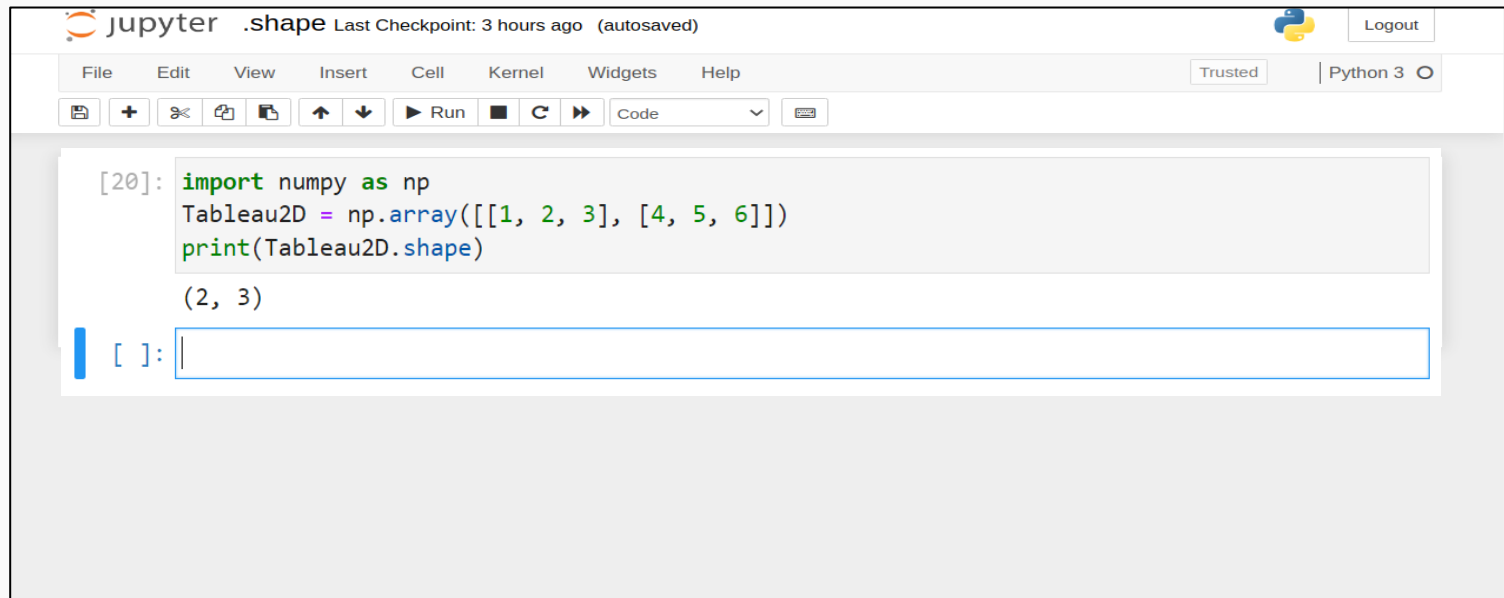
1. Considérons un tableau numpy $x = [[[1, 2, 3, 4]]]$. Que donnera $x[0, 0, 0]$?

- a) **1**
- b) 3
- c) 4



Forme du tableau

- Les tableaux NumPy ont un attribut ***shape*** qui renvoie un n-uplet.
 - La première valeur du n-uplet donne le nombre de dimensions du tableau.
 - La deuxième valeur du n-uplet donne le nombre d'éléments dans chaque dimension..

A screenshot of a Jupyter Notebook interface. The title bar shows ".shape" and "Last Checkpoint: 3 hours ago (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar contains icons for saving, adding cells, undo, redo, and running code. The code cell contains the following Python code:

```
[20]: import numpy as np
Tableau2D = np.array([[1, 2, 3], [4, 5, 6]])
print(Tableau2D.shape)
```

The output of the code is displayed below the code cell as

```
(2, 3)
```

. The input prompt

```
[ ]:
```

 is visible at the bottom of the code cell.



Itération sur des tableaux NumPy (1/8)

Tableaux NumPy 1-D

- Nous pouvons utiliser une boucle for pour itérer sur un tableau 1-D comme nous le faisons pour une liste 1-D Python.

The image shows a Jupyter Notebook interface with the title "iterating over arrays" and a status bar indicating "Last Checkpoint: 3 hours ago (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, execution, and code management. The code cell contains the following Python code:

```
[23]: import numpy as np
Tableau1D = np.array([1, 2, 3, 4, 5])
for i in Tableau1D:
    print(i)
```

The output of the code cell shows the numbers 1, 2, 3, 4, and 5, each on a new line.



Itération sur des tableaux NumPy (2/8)

Tableaux NumPy 2-D

- Nous pouvons utiliser une boucle for imbriquée pour itérer sur un tableau 2-D.
 - La boucle for extérieure itère sur le tableau extérieur.
 - La boucle for intérieure itère sur le tableau intérieur.



Itération sur des tableaux NumPy (3/8)

Tableaux NumPy 2-D

- Nous utilisons une boucle for pour itérer sur le tableau extérieur.
- Nous imprimons tous les tableaux intérieurs.

```
jupyter iterating over arrays Last Checkpoint: 3 hours ago (unsaved changes) Python 3
```

```
[24]: import numpy as np
      Tableau2D = np.array([[1, 2, 3], [4, 5, 6]])
      for i in Tableau2D:
          print(i)
```

```
[1 2 3]
[4 5 6]
```



Itération sur des tableaux NumPy (4/8)

Tableaux NumPy 2-D

- Nous utilisons une autre boucle for imbriquée dans la boucle for externe pour itérer sur le tableau interne.
- Nous affichons tous les éléments de chacun des tableaux internes.

```
jupyter iterating over arrays Last Checkpoint: 3 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
[25]: import numpy as np
      Tableau2D = np.array([[1, 2, 3], [4, 5, 6]])
      for i in Tableau2D:
          for j in i:
              print(j)

1
2
3
4
5
6
```



Itération sur des tableaux NumPy (5/8)

Tableaux NumPy 3-D

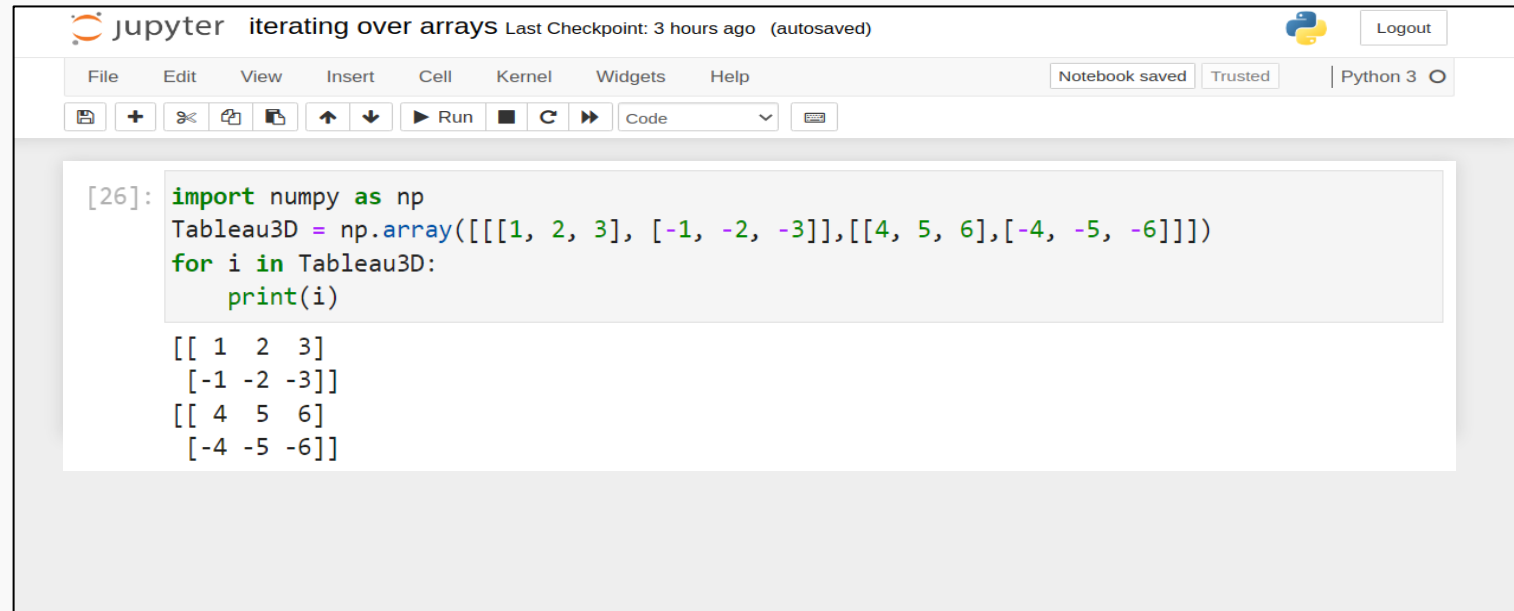
- Nous pouvons utiliser 3 boucles for imbriquées pour itérer sur un tableau 3D.
 - La boucle for la plus extérieure itère sur les tableaux de la première dimension.
 - La boucle for du milieu itère sur les tableaux de la deuxième dimension.
 - La boucle for la plus intérieure itère sur tous les éléments de la troisième dimension.



Itération sur des tableaux NumPy (6/8)

Tableaux NumPy 3-D

- Le tableau le plus à l'extérieur contient 2 tableaux, tous deux en 2D.
- Nous utilisons une boucle for pour afficher ces tableaux 2-D.



```
jupyter iterating over arrays Last Checkpoint: 3 hours ago (autosaved) Logout
File Edit View Insert Cell Kernel Widgets Help Notebook saved Trusted Python 3
[26]: import numpy as np
Tableau3D = np.array([[1, 2, 3], [-1, -2, -3]], [[4, 5, 6], [-4, -5, -6]])
for i in Tableau3D:
    print(i)

[[ 1  2  3]
 [-1 -2 -3]]
[[ 4  5  6]
 [-4 -5 -6]]
```



Itération sur des tableaux NumPy (7/8)

Tableaux NumPy 3-D

- Chacun des tableaux 2-D contient 2 tableaux dans la seconde dimension, chacun d'entre eux étant 1-D.
- Nous utilisons une autre boucle for imbriquée dans la première boucle for pour imprimer ces tableaux 1D.

The image shows a Jupyter Notebook interface with the title "iterating over arrays" and a status bar indicating "Last Checkpoint: 3 hours ago (unsaved changes)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The code cell contains the following Python code:

```
[27]: import numpy as np
Tableau3D = np.array([[[1, 2, 3], [-1, -2, -3]], [[4, 5, 6], [-4, -5, -6]]])
for i in Tableau3D:
    for j in i:
        print(j)
```

The output of the code is displayed below the code cell:

```
[1 2 3]
[-1 -2 -3]
[4 5 6]
[-4 -5 -6]
```



Itération sur des tableaux NumPy (8/8)

Tableaux NumPy 3-D

- Chacun des tableaux 1-D contient 3 éléments dans la troisième dimension.
- Nous utilisons une autre boucle for imbriquée dans les deux premières boucles for pour imprimer ces éléments.

The image shows a Jupyter Notebook window titled "iterating over arrays" with a "Last Checkpoint: 3 hours ago (unsaved changes)" status. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, undo, redo, and running code. The code cell contains the following Python code:

```
[28]: import numpy as np
Tableau3D = np.array([[[1, 2, 3], [-1, -2, -3]], [[4, 5, 6], [-4, -5, -6]]])
for i in Tableau3D:
    for j in i:
        for k in j:
            print(k)
```

The output of the code is displayed below the code cell, showing the elements of the 3D array in a single column:

```
1
2
3
-1
-2
-3
4
5
6
-4
-5
-6
```



Remise à niveau
rapide de Python

Data Science avec
Python

Installation des bibliothèques	1	
	2	Importation de bibliothèques
Bibliothèque Pandas pour la Data Science	3	
	4	Bibliothèque NumPy pour Data Science
Pandas vs NumPy	5	
	6	Bibliothèque Matplotlib pour Data Science
Bibliothèque Seaborn pour Data Science	7	
	8	Tableaux NumPy
Mathématiques pour Data Science	9	
	10	Travailler avec différents types de données
Variables	11	
	12	Opérateurs arithmétiques



Mathématiques pour Data Science

NumPy nous fournit une énorme collection de fonctions de haut niveau pour les tableaux multidimensionnels.

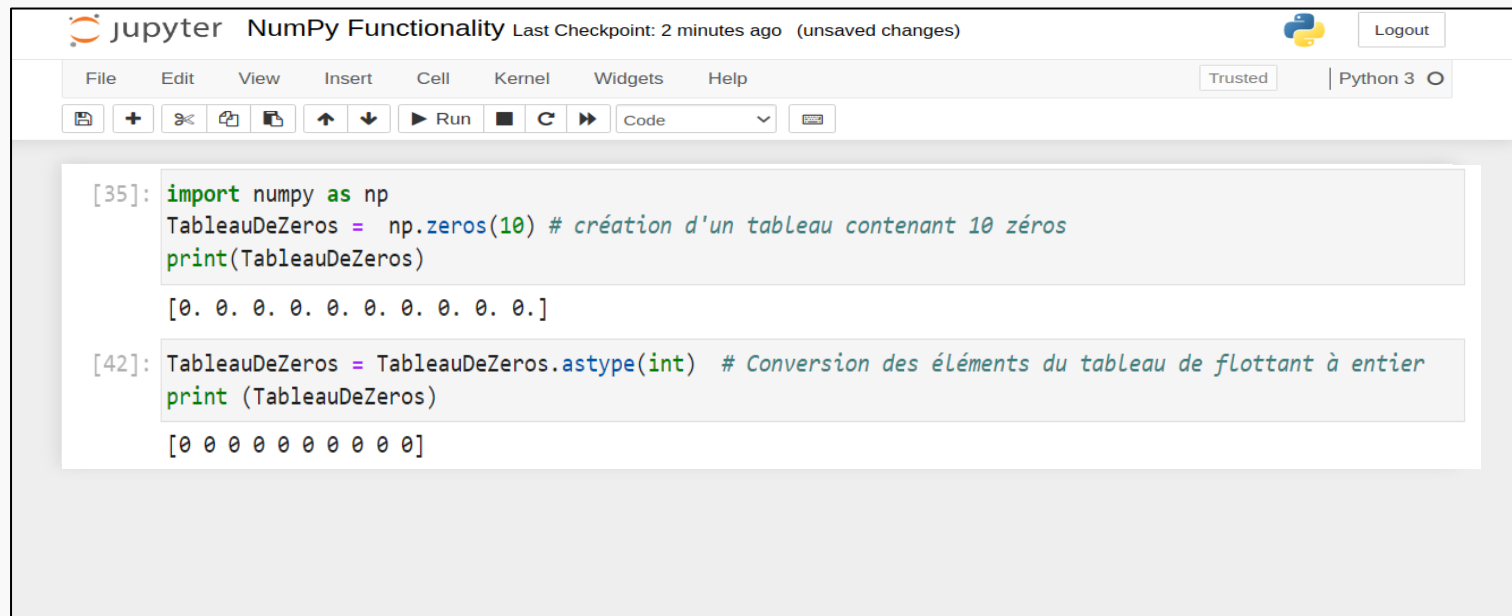
- Jetons un coup d'œil à certaines des fonctionnalités fournies par NumPy.



.zeros()

Pour créer un tableau NumPy pré-rempli de zéros, nous pouvons utiliser la fonction NumPy intégrée **.zeros()**.

.zeros() nous donne une liste pré-remplie de zéros flottants. Pour convertir cette liste en liste d'entiers, nous utilisons la fonction **.astype()**.



```
jupyter NumPy Functionality Last Checkpoint: 2 minutes ago (unsaved changes) Python 3
```

```
[35]: import numpy as np
      TableauDeZeros = np.zeros(10) # création d'un tableau contenant 10 zéros
      print(TableauDeZeros)

      [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

[42]: TableauDeZeros = TableauDeZeros.astype(int) # Conversion des éléments du tableau de flottant à entier
      print (TableauDeZeros)

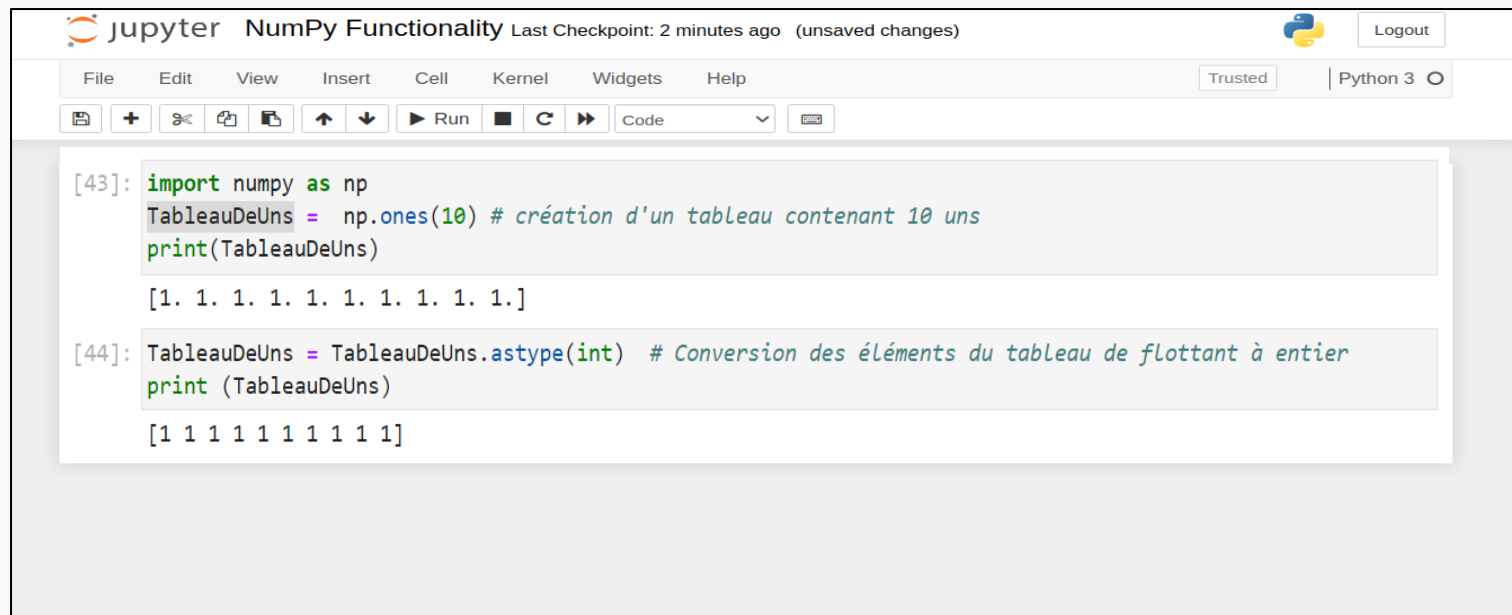
      [0 0 0 0 0 0 0 0 0 0]
```



.ones()

Pour créer un tableau NumPy pré-rempli de uns, nous pouvons utiliser la fonction NumPy intégrée `.ones()`.

`.ones()` nous donne une liste pré-remplie de flottants. Pour convertir cette liste en liste d'entiers, nous utilisons la fonction `.astype()`.



```
jupyter NumPy Functionality Last Checkpoint: 2 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
[43]: import numpy as np
      TableauDeUns = np.ones(10) # création d'un tableau contenant 10 uns
      print(TableauDeUns)

      [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]

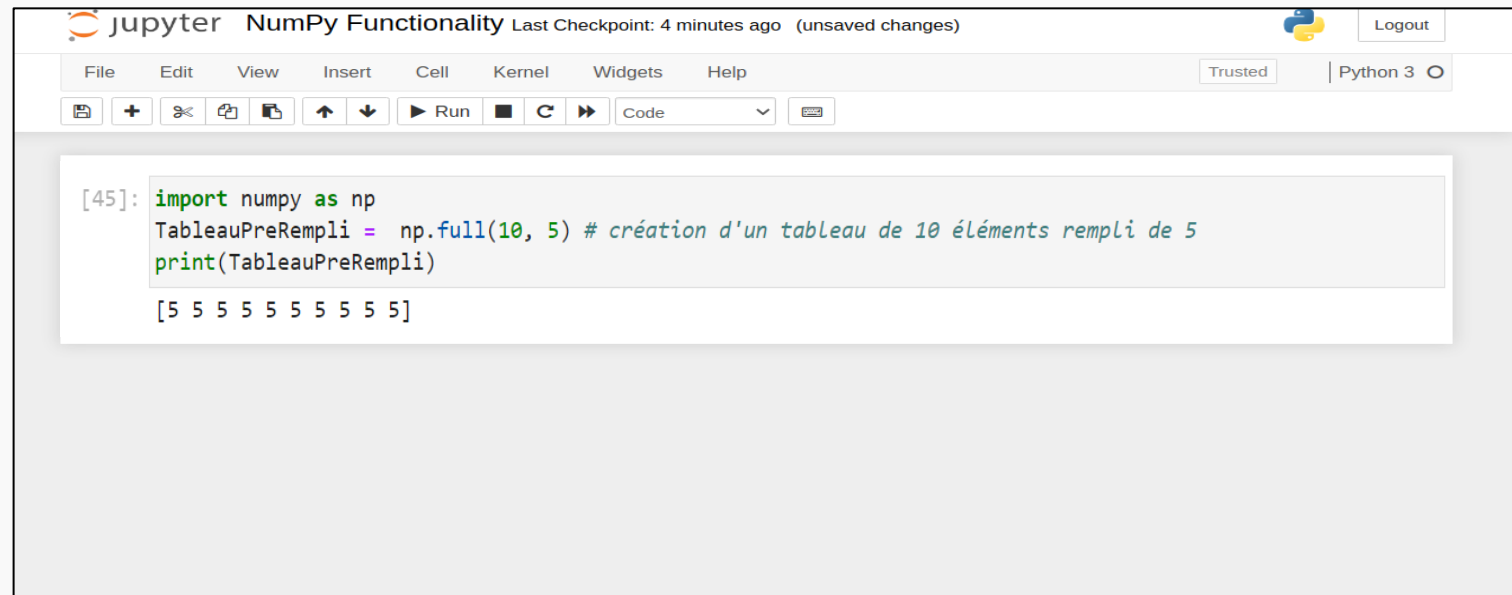
[44]: TableauDeUns = TableauDeUns.astype(int) # Conversion des éléments du tableau de flottant à entier
      print (TableauDeUns)

      [1 1 1 1 1 1 1 1 1 1]
```



.full()

- Pour créer un tableau NumPy pré-rempli avec un nombre spécifique, nous pouvons utiliser la fonction `.full()`, fonction intégrée de NumPy.
 - Le premier argument de la fonction `.full()` est la taille du tableau
 - Le deuxième argument de la fonction `.full()` est la valeur avec laquelle nous voulons que notre liste soit pré-remplie au préalable.



```
[45]: import numpy as np
      TableauPreRempli = np.full(10, 5) # création d'un tableau de 10 éléments rempli de 5
      print(TableauPreRempli)

      [5 5 5 5 5 5 5 5 5 5]
```




Quiz Time

1. Quelle est la syntaxe correcte pour créer un tableau numpy de 9 éléments rempli de tous les zéros (float) ?
- a) `np.zeros()`
 - b) `np.zeros(0)`
 - c) `np.zeros(9)`



Quiz Time

1. Quelle est la syntaxe correcte pour créer un tableau numpy de 9 éléments rempli de tous les zéros (float) ?
- a) `np.zeros()`
 - b) `np.zeros(0)`
 - c) `np.zeros(9)`



- Nous pouvons ajouter un scalaire à un tableau NumPy en utilisant simplement l'opérateur (+).
- La quantité scalaire est ajoutée à chacun des éléments du tableau.
- Notez que l'ajout d'un scalaire à une liste Python entraînera une erreur





Opérations scalaires (2/5)

Soustraction

- Nous pouvons soustraire un scalaire d'un tableau NumPy en utilisant simplement l'opérateur (-).
- La quantité scalaire est soustraite de chacun des éléments du tableau.
- Notez que la soustraction d'un scalaire d'une liste Python entraînera une erreur.

```
jupyter NumPy Functionality Last Checkpoint: 8 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

[76]: import numpy as np
      matrice = np.array([[1, 2, 3], [4, 5, 6]])
      print(matrice)

      [[1 2 3]
       [4 5 6]]

[77]: print(matrice - 2)

      [[-1  0  1]
       [ 2  3  4]]

[78]: ListePython = [[1, 2, 3], [4, 5, 6]]
      print(ListePython - 2)

      -----
      TypeError                                Traceback (most recent call last)
      <ipython-input-78-d643c030e30c> in <module>
          1 ListePython = [[1, 2, 3], [4, 5, 6]]
      ----> 2 print(ListePython - 2)

      TypeError: unsupported operand type(s) for -: 'list' and 'int'
```



Opérations scalaires (3/5)

Multiplication

- Nous pouvons multiplier un scalaire avec un tableau NumPy en utilisant simplement l'opérateur (*).
- La quantité scalaire est multipliée avec chacun des éléments du tableau.
- Notez que la multiplication d'un scalaire avec une liste Python entraînera une concaténation de listes.

The image shows a Jupyter Notebook interface with the title "NumPy Functionality". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and other notebook functions. The notebook is running on Python 3. The code is executed in three cells, showing the output of each operation.

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

[[1 2 3]
 [4 5 6]]

[74]: print(matrice * 2)

[[ 2  4  6]
 [ 8 10 12]]

[75]: ListePython = [[1, 2, 3], [4, 5, 6]]
      print(ListePython * 2)

[[1, 2, 3], [4, 5, 6], [1, 2, 3], [4, 5, 6]]
```



Opérations scalaires (4/5)

Division

Nous pouvons diviser un tableau NumPy par un scalaire en utilisant simplement l'opérateur (/) pour la division des flottants ou l'opérateur (//) pour la division des entiers.

- Chacun des éléments du tableau est divisé par le scalaire.
- Notez que la division d'une liste Python par un scalaire entraînera une erreur.

```
Jupyter NumPy Functionality Last Checkpoint: 10 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
[68]: import numpy as np
      matrice = np.array([[1, 2, 3], [4, 5, 6]])
      print(matrice)
      [[1 2 3]
       [4 5 6]]
[69]: print(matrice / 2)
      [[0.5 1.  1.5]
       [2.  2.5 3. ]]
[70]: print(matrice // 2)
      [[0 1]
       [2 2 3]]
[71]: ListePython = [[1, 2, 3], [4, 5, 6]]
      print(ListePython / 2)
      -----
      TypeError                                Traceback (most recent call last)
      <ipython-input-71-29ec4c095271> in <module>
          1 ListePython = [[1, 2, 3], [4, 5, 6]]
      ----> 2 print(ListePython / 2)
      TypeError: unsupported operand type(s) for /: 'list' and 'int'
```



Opérations scalaires (5/5)

Puissance

- Nous pouvons élever chaque élément d'un tableau NumPy à une puissance en utilisant simplement l'opérateur (**).
- Notez que l'augmentation des éléments d'une liste Python à l'aide de l'opérateur (**) entraînera une erreur.

The image shows a Jupyter Notebook interface titled "NumPy Functionality". The notebook contains three code cells. The first cell imports NumPy and creates a 2x3 array. The second cell squares the array. The third cell attempts to square a Python list, which results in a `TypeError`.

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

[[1 2 3]
 [4 5 6]]

[80]: print(matrice ** 2)

[[ 1  4  9]
 [16 25 36]]

[81]: ListePython = [[1, 2, 3], [4, 5, 6]]
print(ListePython ** 2)

-----
TypeError                                Traceback (most recent call last)
<ipython-input-81-0671c0972e1d> in <module>
      1 ListePython = [[1, 2, 3], [4, 5, 6]]
----> 2 print(ListePython ** 2)

TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
```



Transposée

- Nous pouvons prendre la transposée d'un tableau NumPy en mettant .T à la fin du tableau.
- Notez que la transposée d'une liste Python entraînera une erreur.

```
Jupyter NumPy Functionality Last Checkpoint: 12 minutes ago (unsaved changes) Python 3
```

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

[[1 2 3]
 [4 5 6]]

[83]: print(matrice.T)

[[1 4]
 [2 5]
 [3 6]]

[84]: ListePython = [[1, 2, 3], [4, 5, 6]]
print(ListePython.T)

-----
AttributeError                                Traceback (most recent call last)
<ipython-input-84-a679f2b4c16f> in <module>
      1 ListePython = [[1, 2, 3], [4, 5, 6]]
----> 2 print(ListePython.T)

AttributeError: 'list' object has no attribute 'T'
```




Opérations sur les éléments (1/4)

Addition

- Nous pouvons additionner les éléments de deux tableaux NumPy en utilisant simplement l'opérateur (+).
- Chaque élément du premier tableau est ajouté à l'élément correspondant du second tableau.
- Notez que l'addition de deux listes Python à l'aide de l'opérateur (+) n'est pas possible. Au lieu de cela, les listes sont concaténées si nous utilisons l'opérateur (+).

```
jupyter NumPy Functionality Last Checkpoint: 13 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
+ -> Run Code

[85]: import numpy as np
      matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
      print(matrice1)

      [[1 2 3]
       [4 5 6]]

[86]: import numpy as np
      matrice2 = np.array([[-1, -2, -3], [-4, -5, -6]])
      print(matrice2)

      [[-1 -2 -3]
       [-4 -5 -6]]

[87]: print(matrice1 + matrice2)

      [[0 0 0]
       [0 0 0]]

[88]: ListePython1 = [[1, 2, 3], [4, 5, 6]]
      ListePython2 = [[-1, -2, -3], [-4, -5, -6]]
      print(ListePython1 + ListePython2)

      [[1, 2, 3], [4, 5, 6], [-1, -2, -3], [-4, -5, -6]]
```



Opérations sur les éléments (2/4)

Soustraction

- Nous pouvons soustraire les éléments de deux tableaux NumPy en utilisant simplement l'opérateur (-).
- Chaque élément du second tableau est soustrait de l'élément correspondant du premier tableau.
- Notez que la soustraction des éléments de deux listes Python à l'aide de l'opérateur (-) entraînera une erreur.

```
jupyter NumPy Functionality Last Checkpoint: 14 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

[89]: import numpy as np
      matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
      print(matrice1)
      [[1 2 3]
       [4 5 6]]

[90]: import numpy as np
      matrice2 = np.array([[-1, -2, -3], [-4, -5, -6]])
      print(matrice2)
      [[-1 -2 -3]
       [-4 -5 -6]]

[91]: print(matrice1 - matrice2)
      [[ 2  4  6]
       [ 8 10 12]]

[92]: ListePython1 = [[1, 2, 3], [4, 5, 6]]
      ListePython2 = [[-1, -2, -3], [-4, -5, -6]]
      print(ListePython1 - ListePython2)

      -----
      TypeError                                 Traceback (most recent call last)
      <ipython-input-92-7330d015d772> in <module>
          1 ListePython1 = [[1, 2, 3], [4, 5, 6]]
          2 ListePython2 = [[-1, -2, -3], [-4, -5, -6]]
      ----> 3 print(ListePython1 - ListePython2)

      TypeError: unsupported operand type(s) for -: 'list' and 'list'
```



Opérations sur les éléments (3/4)

Multiplication

- Nous pouvons multiplier les éléments de deux tableaux NumPy en utilisant simplement l'opérateur (*).
- Chaque élément du premier tableau est multiplié par l'élément correspondant du second tableau.
- Notez que la multiplication des éléments de deux listes Python à l'aide de l'opérateur (*) entraînera une erreur.

```
[89]: import numpy as np
matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
print(matrice1)

[[1 2 3]
 [4 5 6]]

[90]: import numpy as np
matrice2 = np.array([[-1, -2, -3], [-4, -5, -6]])
print(matrice2)

[[-1 -2 -3]
 [-4 -5 -6]]

[93]: print(matrice1 * matrice2)

[[ -1  -4  -9]
 [-16 -25 -36]]

[94]: ListePython1 = [[1, 2, 3], [4, 5, 6]]
ListePython2 = [[-1, -2, -3], [-4, -5, -6]]
print(ListePython1 * ListePython2)

-----
TypeError                                Traceback (most recent call last)
<ipython-input-94-5302b9039179> in <module>
      1 ListePython1 = [[1, 2, 3], [4, 5, 6]]
      2 ListePython2 = [[-1, -2, -3], [-4, -5, -6]]
----> 3 print(ListePython1 * ListePython2)

TypeError: can't multiply sequence by non-int of type 'list'
```



Opérations sur les éléments (4/4)

Division

- Nous pouvons diviser les éléments de deux tableaux NumPy en utilisant simplement l'opérateur (/).
- Chaque élément du premier tableau est divisé par l'élément correspondant du second tableau.
- Notez que la division des éléments de deux listes Python à l'aide de l'opérateur (/) entraînera une erreur.

```
jupyter NumPy Functionality Last Checkpoint: 16 minutes ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
[95]: import numpy as np
      matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
      print(matrice1)
      [[1 2 3]
       [4 5 6]]

[96]: import numpy as np
      matrice2 = np.array([[1, -2, -3], [-4, -5, -6]])
      print(matrice2)
      [[1 -2 -3]
       [-4 -5 -6]]

[97]: print(matrice1 / matrice2)
      [[-1. -1. -1.]
       [-1. -1. -1.]]

[98]: print(matrice1 // matrice2)
      [[-1 -1 -1]
       [-1 -1 -1]]

[99]: ListePython1 = [[1, 2, 3], [4, 5, 6]]
      ListePython2 = [[1, -2, -3], [-4, -5, -6]]
      print(ListePython1 / ListePython2)

      -----
      Traceback (most recent call last)
      <ipython-input-99-bcb88f5e9fc5> in <module>
          1 ListePython1 = [[1, 2, 3], [4, 5, 6]]
          2 ListePython2 = [[1, -2, -3], [-4, -5, -6]]
      ----> 3 print(ListePython1 / ListePython2)

      TypeError: unsupported operand type(s) for /: 'list' and 'list'
```



Multiplication matricielle

Outre la multiplication par éléments, NumPy nous fournit également une fonction intégrée pour calculer la multiplication matricielle de deux tableaux. Nous utilisons la fonction `.matmul()` de la bibliothèque NumPy pour la multiplication matricielle de deux tableaux.

```
jupyter NumPy Functionality Last Checkpoint: 19 minutes ago (unsaved changes) Python 3
```

```
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
```

```
[100...] matrice1 = np.array([[1, 2, 3], [4, 5, 6], [0, 0, 0]])
print(matrice1)

[[1 2 3]
 [4 5 6]
 [0 0 0]]

[101...] matrice2 = np.array([[-1, -2, -3], [-4, -5, -6], [0, 0, 0]])
print(matrice2)

[[-1 -2 -3]
 [-4 -5 -6]
 [ 0  0  0]]

[103...] np.matmul(matrice1, matrice2)

[103...] array([[ -9, -12, -15],
               [-24, -33, -42],
               [  0,   0,   0]])
```



Quiz Time

1. Laquelle des affirmations suivantes est correcte?
- a) * est utilisé pour la multiplication de tableaux.
 - b) * est utilisé pour la multiplication par élément.
 - c) *est utilisé pour la puissance.



Quiz Time

1. Laquelle des affirmations suivantes est correcte?
- a) * est utilisé pour la multiplication de tableaux.
 - b) * est utilisé pour la multiplication par élément.**
 - c) * est utilisé pour la puissance.



Statistiques (1/7)

.min()

- La fonction .min() nous donne la valeur minimale dans un tableau NumPy.
- Cette fonction peut également être appliquée sur des listes Python.

The image shows a Jupyter Notebook interface with the title 'NumPy Functionality'. The top bar includes a 'Logout' button and a 'Python 3' selector. The menu bar contains 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for file operations, running, and code execution. The notebook contains two code cells. The first cell, labeled [47], imports numpy as np, creates a 2x3 array named 'matrice1' with values [[1, 2, 3], [4, 5, 6]], and prints it. The output shows the array structure. The second cell, labeled [48], calls np.min(matrice1), and the output shows the value 1, which is the minimum value in the array.

```
[47]: import numpy as np
matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
print(matrice1)

[[1 2 3]
 [4 5 6]]

[48]: np.min(matrice1)

[48]: 1
```




Statistiques (2/7)

.max()

- La fonction .max() nous donne la valeur maximale dans un tableau NumPy.
- Cette fonction peut également être appliquée sur des listes Python.

```
jupyter NumPy Functionality Last Checkpoint: 20 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Notebook saved Trusted Python 3

[49]: import numpy as np
      matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
      print(matrice1)

      [[1 2 3]
       [4 5 6]]

[50]: np.max(matrice1)

[50]: 6
```



Statistiques (3/7)

.sum()

- La fonction `.sum()` nous donne la somme de toutes les valeurs d'un tableau NumPy.
- Cette fonction peut également être appliquée sur des listes Python.

The image shows a Jupyter Notebook interface with the title "NumPy Functionality". The top bar includes a "Logout" button and a "Python 3" selector. The menu bar contains "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". The toolbar has icons for saving, adding cells, undo, redo, and running code. The notebook contains two code cells. The first cell, labeled [51], imports NumPy as np, creates a 2x2 array named 'matrice1' with values [[1, 2, 3], [4, 5, 6]], and prints it. The output shows the array structure. The second cell, labeled [52], calls np.sum(matrice1), and the output is 21.

```
[51]: import numpy as np
matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
print(matrice1)

[[1 2 3]
 [4 5 6]]

[52]: np.sum(matrice1)

[52]: 21
```



Statistiques (4/7)

.mean()

- La fonction `.mean()` nous donne la moyenne de toutes les valeurs d'un tableau NumPy.
- Cette fonction peut également être appliquée sur des listes Python.

```
[53]: import numpy as np
matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
print(matrice1)

[[1 2 3]
 [4 5 6]]

[54]: np.mean(matrice1)

[54]: 3.5
```



Statistiques (5/7)

.std()

- La fonction `.std()` nous donne l'écart type d'un tableau NumPy.
- Cette fonction peut également être appliquée sur des listes Python.

```
[55]: import numpy as np
matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
print(matrice1)

[[1 2 3]
 [4 5 6]]

[56]: np.std(matrice1)

[56]: 1.707825127659933
```



Statistiques (6/7)

.median()

- La fonction `.median()` nous donne la médiane d'un tableau NumPy.
- Cette fonction peut également être appliquée sur des listes Python.



```
[57]: import numpy as np
      matrice1 = np.array([[1, 2, 3], [4, 5, 6]])
      print(matrice1)

[[1 2 3]
 [4 5 6]]

[58]: np.median(matrice1)

[58]: 3.5
```



Statistiques (7/7)

- Une liste détaillée des fonctions statistiques de NumPy peut être trouvée sur le lien ci-dessous;

<https://numpy.org/doc/stable/reference/routines.statistics.html>



Documentation

- https://www.w3schools.com/python/numpy/numpy_intro.asp
- <https://www.tutorialspoint.com/numpy/index.htm>