



Séance 2

PYTHON FOR MACHINE LEARNING



Qu'est-ce que Python ?



Python est un langage de programmation :

- **interprété**
- **haut niveau**
- **polyvalent,**

conçu pour être :

- **simple**
- **lisible.**

Il est largement utilisé dans divers domaines, notamment :

- **Développement web**
- **Science des données et Machine Learning (NumPy, Pandas, Matplotlib)**
- **Automatisation et scripting**
- **Développement logiciel et jeux vidéo**
- **Cybersécurité et analyse de données**



Caractéristiques clés de Python

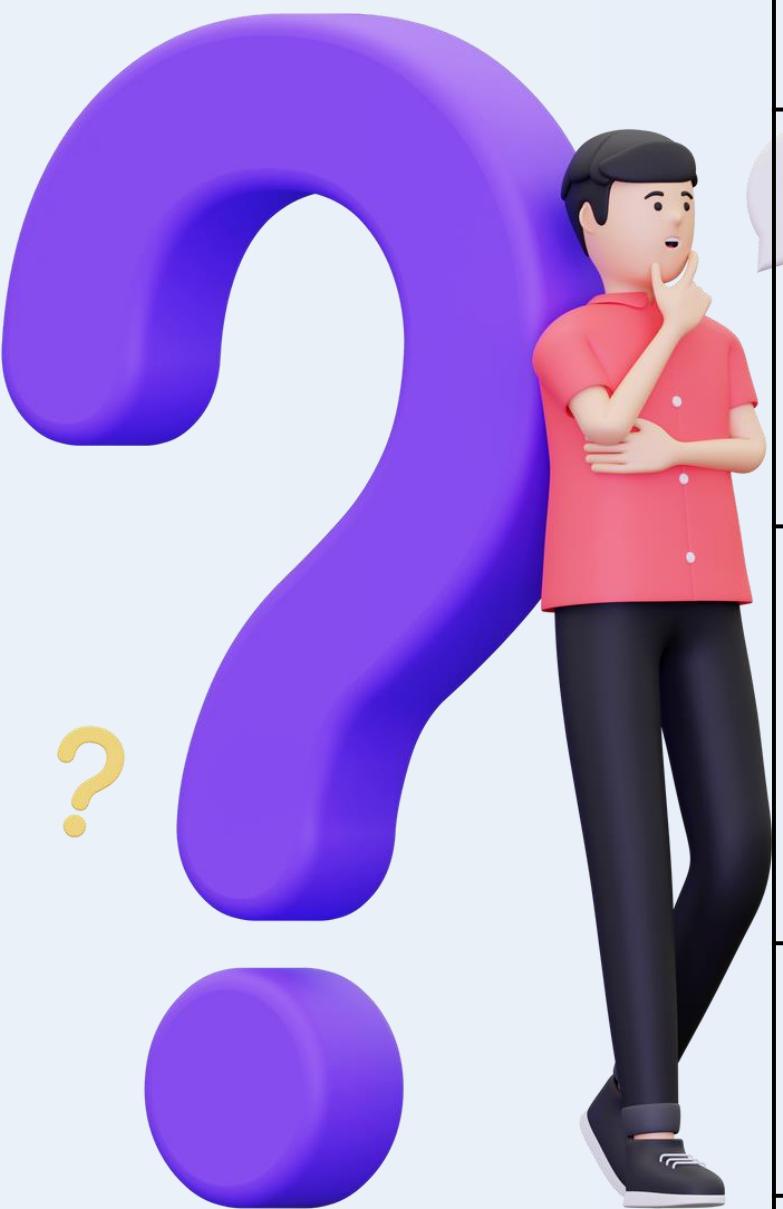
?



- **Facile à apprendre : syntaxe simple et lisible**
- **Multiplateforme : compatible avec Windows, macOS et Linux**
- **Extensible : intègre des modules en C/C++ pour plus de performances**
- **Orienté objet et fonctionnel : permet différents paradigmes de programmation**
- **Écosystème riche : des milliers de bibliothèques pour répondre à tous les besoins**



Comparaison de Python avec d'autres langages



Langage	Points forts	Limitations	Domaines d'application
Python	<ul style="list-style-type: none">Facilité d'apprentissage,vaste écosystème,support ML/IA	<ul style="list-style-type: none">Moins rapide que C/C++	<ul style="list-style-type: none">IA, Data Science, Web, Automatisation
C++	<ul style="list-style-type: none">Très performant,contrôle précis de la mémoire	<ul style="list-style-type: none">Complexité,temps de développement plus long	<ul style="list-style-type: none">Jeux vidéo,systèmes embarqués,haute performance
Java	<ul style="list-style-type: none">Portable (JVM),robuste,performant	<ul style="list-style-type: none">Plus verbeux que Python	<ul style="list-style-type: none">Développement :<ul style="list-style-type: none">d'applications webmobiles,entreprises
JavaScript	<ul style="list-style-type: none">Indispensable pour le web,interactif	<ul style="list-style-type: none">Pas adapté au calcul intensif	<ul style="list-style-type: none">Développement web,applications interactives
R	<ul style="list-style-type: none">Spécialisé en statistiques et data science	<ul style="list-style-type: none">Moins polyvalent	<ul style="list-style-type: none">Analyse de données,statistiques,Machine Learning



Pourquoi Python pour le Machine Learning ?

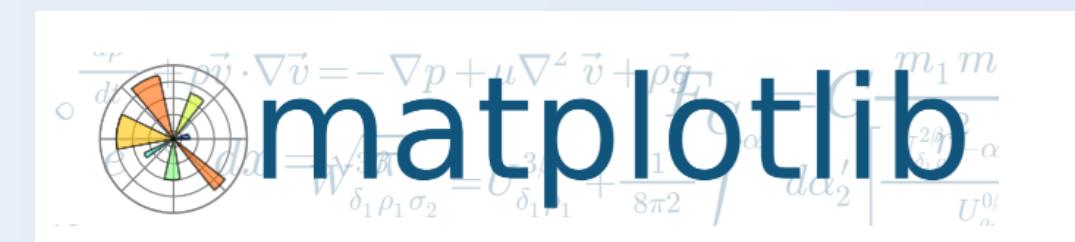
Simplicité et Facilité
d'Apprentissage

Un Écosystème de
Bibliothèques Puissant

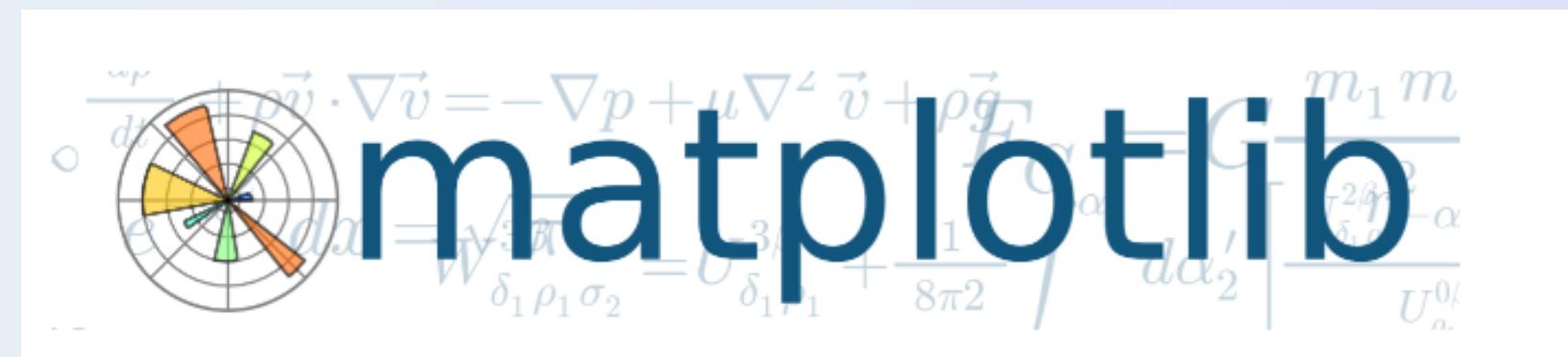
Puissant et flexible pour les
projets IA et ML



les 10 bibliothèques les plus utilisées



les trois librairies incontournables de l'écosystème scientifique de Python



La Bibliothèque NumPy

Qu'est-ce que NumPy ?



NumPy (Numerical Python) est une bibliothèque open-source qui permet de manipuler efficacement des données numériques sous forme de tableaux multidimensionnels (ndarrays).

- Optimisé pour les calculs scientifiques et mathématiques
- Plus rapide et plus efficace que les listes Python natives
- Essentiel en Data Science, Machine Learning et Deep Learning



La Bibliothèque NumPy

Pourquoi utiliser NumPy ?



Performance et rapidité

- NumPy est écrit en C et Fortran, ce qui le rend beaucoup plus rapide que les listes Python classiques.
- Gestion efficace de *grands ensembles de données* grâce à son stockage en tableaux multidimensionnels.

Outils puissants pour le Machine Learning

- Opérations mathématiques avancées (algèbre linéaire, statistiques, transformations de Fourier).
- Manipulation et transformation de données essentielles pour entraîner les modèles d'IA.

Interopérabilité avec d'autres bibliothèques

- Utilisé comme base pour Pandas, Scikit-learn, TensorFlow et PyTorch.
- Facilement intégrable avec Matplotlib pour la visualisation des données.



La Bibliothèque NumPy

Vérification et Installation de NumPy



Vérifier si NumPy est installé

Avant d'installer NumPy, vérifions s'il est déjà disponible dans l'environnement Python.

```
Entrée [5]: import numpy as np  
            print(np.__version__) # Vérifie la version de NumPy  
  
1.19.5
```

Si une erreur apparaît ("ModuleNotFoundError: No module named 'numpy'"),
==> **NumPy n'est pas installé.**

Installer NumPy si nécessaire

Avec pip (Python standard)

```
Entrée [7]: pip install numpy  
  
Requirement already satisfied: numpy in c:\us  
Note: you may need to restart the kernel to u
```

Avec conda (si vous utilisez Anaconda)

```
Entrée [*]: conda install numpy  
  
Entrée [ ]:
```



Manipulation des données sur NumPy



Création d'un Tableau à une dimension avec NumPy : vecteur

Définition : Un tableau 1D est une liste de nombres stockés sous forme de vecteur.

Utilisation : Données linéaires (séries temporelles, caractéristiques d'un modèle ML).

```
Entrée [6]: import numpy as np  
array_1d = np.array([10, 20, 30, 40, 50])  
print(array_1d)  
print("Shape :", array_1d.shape)
```

```
[10 20 30 40 50]  
Shape : (5,)
```

Un vecteur avec 5 éléments (1 seule ligne).



Manipulation des données sur NumPy



Création d'un Tableau à deux dimensions avec NumPy : Matrice

Définition : Un tableau 2D est une matrice (lignes × colonnes).

Utilisation : Il est utilisé en algèbre linéaire et Machine Learning.

```
Entrée [7]: array_2d = np.array([[10, 20, 30], [40, 50, 60]])
print("Tableau 2D :\n", array_2d)
print("Shape :", array_2d.shape)
```

```
Tableau 2D :
[[10 20 30]
 [40 50 60]]
Shape : (2, 3)
```

C'est une matrice avec 2 lignes et 3 colonnes.



Manipulation des données sur NumPy



Création d'un Tableau à trois dimensions avec NumPy : Tensor 3D

Définition : Un tableau 3D est une collection de matrices (stack de matrices)..

Utilisation : Utilisé en vision par ordinateur et Deep Learning (ex: images avec RGB)

```
Entrée [8]: array_3d = np.array([[ [1, 2], [3, 4]], [[5, 6], [7, 8]]])
print("Tableau 3D :\n", array_3d)
print("Shape :", array_3d.shape)
```

```
Tableau 3D :
[[[1 2]
 [3 4]]
```

```
[[5 6]
 [7 8]]]
```

```
Shape : (2, 2, 2)
```

2 matrices de 2 lignes et 2 colonnes.



Manipulation des données sur NumPy



Création d'un Tableau à n dimensions avec NumPy : Multidimensionnel

Définition : Un tableau multidimensionnel peut aller au-delà de 3 dimensions.

Utilisation : Utilisé en réseaux de neurones (Deep Learning), analyse volumétrique (IRM, 3D)...

Exemple en NumPy (tableau 4D) :

```
Entrée [9]: array_4d = np.random.rand(2, 3, 4, 5)
            print("Tableau 4D :\n", array_4d)
            print("Shape :", array_4d.shape)
```

Shape : (2, 3, 4, 5)

Tableau 4D :

```
[[[ [0.57624458 0.73971299 0.96526077 0.22352814 0.67383582]
    [0.9092169 0.69494369 0.0083912 0.75355541 0.15059646]
    [0.52209385 0.56757866 0.79711694 0.93321128 0.21685669]
    [0.09475552 0.1312955 0.18064072 0.70796476 0.87133991] ]
   [[0.69753 0.5820705 0.54190359 0.58475055 0.91875554]
    [0.78894925 0.41745725 0.33163511 0.45669479 0.44428422]
    [0.81813974 0.0574014 0.11639539 0.97678925 0.47448564]
    [0.50729743 0.28807966 0.79435438 0.92492317 0.54131603] ]
   [[0.49067466 0.32634219 0.13098302 0.02815857 0.61316346]
    [0.20675741 0.07218351 0.46974418 0.85902857 0.46256277]
    [0.14159179 0.11930362 0.39241828 0.42741735 0.35732183]
    [0.56510792 0.1184212 0.03317311 0.20483616 0.74471022] ]
   [[ [0.54260307 0.4240192 0.86827971 0.01045909 0.98228946]
    [0.99366391 0.3593687 0.25986244 0.56308643 0.98715917]
    [0.50202561 0.89909837 0.55133428 0.09589654 0.40759853]
    [0.31921598 0.20578199 0.48464229 0.34124058 0.32929555] ]
   [[0.56412391 0.89672658 0.12931855 0.2672649 0.93164277]
    [0.42859741 0.05803391 0.9408166 0.7614221 0.44657317]
    [0.24862625 0.51218753 0.59265233 0.28232191 0.67395291]
    [0.54674841 0.05249712 0.07399159 0.65058019 0.52736659] ]
   [[0.58010242 0.7435668 0.60674704 0.91291867 0.82294498]
    [0.30477108 0.98471362 0.03013598 0.93595081 0.6980362 ]
    [0.25155448 0.83398184 0.80902254 0.64012012 0.35186398]
    [0.5914567 0.81965364 0.51662407 0.37739243 0.80585926]]]
```

2 blocs contenant 3 matrices de 4 lignes et 5 colonnes.



Manipulation des données sur NumPy



Présentation des Tableaux Remplis de Zéros, de Uns et Aléatoires en NumPy

NumPy permet de créer des tableaux

:

- Remplis de 0 avec `np.zeros()`
- Remplis de 1 avec `np.ones()`
- Avec des valeurs aléatoires avec `np.random.rand()`



Manipulation des données sur NumPy



Création de Tableaux Remplis de Zéros, Uns et Valeurs Aléatoires Tableaux 1D (Vecteurs)

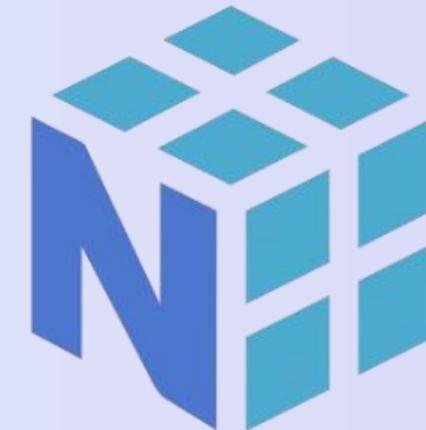
```
Entrée [10]: zeros_1d = np.zeros(5) # Vecteur de 5 éléments remplis de 0
              ones_1d = np.ones(5)   # Vecteur de 5 éléments remplis de 1
              random_1d = np.random.rand(5) # Vecteur de 5 éléments aléatoires entre 0 et 1

              print("Tableau de zéros 1D :", zeros_1d)
              print("Tableau de uns 1D :", ones_1d)
              print("Tableau aléatoire 1D :", random_1d)

Tableau de zéros 1D : [0. 0. 0. 0. 0.]
Tableau de uns 1D : [1. 1. 1. 1. 1.]
Tableau aléatoire 1D : [0.6719977 0.89853883 0.49653617 0.38867997 0.12396677]
```



Manipulation des données sur NumPy



Création de Tableaux Remplis de Zéros, Uns et Valeurs Aléatoires

Tableaux 2D : (lignes × colonnes) =

(Matrices)

```
Entrée [11]: zeros_2d = np.zeros((2, 3)) # Matrice 2x3 remplie de 0
ones_2d = np.ones((2, 3))      # Matrice 2x3 remplie de 1
random_2d = np.random.rand(2, 3) # Matrice 2x3 avec valeurs aléatoires

print("Matrice de zéros 2D :\n", zeros_2d)
print("Matrice de uns 2D :\n", ones_2d)
print("Matrice aléatoire 2D :\n", random_2d)
```

Matrice de zéros 2D :

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

Matrice de uns 2D :

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

Matrice aléatoire 2D :

```
[[0.365539  0.17853125 0.35943156]
 [0.22765879 0.74839479 0.69195571]]
```



Manipulation des données sur NumPy



Création de Tableaux Remplis de Zéros, Uns et Valeurs Aléatoires

Tableaux 3D : (tenseurs)

```
Entrée [12]: zeros_3d = np.zeros((2, 2, 3)) # 2 matrices 2x3 remplies de 0
ones_3d = np.ones((2, 2, 3))      # 2 matrices 2x3 remplies de 1
random_3d = np.random.rand(2, 2, 3) # 2 matrices 2x3 avec valeurs aléatoires

print("Tableau de zéros 3D :\n", zeros_3d)
print("Tableau de uns 3D :\n", ones_3d)
print("Tableau aléatoire 3D :\n", random_3d)
```

Tableau de zéros 3D :

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

 [[0. 0. 0.]
 [0. 0. 0.]])
```

Tableau de uns 3D :

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

 [[1. 1. 1.]
 [1. 1. 1.]])
```

Tableau aléatoire 3D :

```
[[[0.89576282 0.83055116 0.0691499 ]
 [0.433091   0.39398532 0.78102483]

 [[0.88804312 0.61190014 0.02225348]
 [0.09083578 0.90227474 0.30995113]]]
```



Manipulation des données sur NumPy



Création de Tableaux Remplis de Zéros, Uns et Valeurs Aléatoires

Tableaux 4D et plus : (Multidimensionnels)

```
Entrée [13]: zeros_4d = np.zeros((2, 2, 2, 3)) # Tableau 4D rempli de 0
ones_4d = np.ones((2, 2, 2, 3)) # Tableau 4D rempli de 1
random_4d = np.random.rand(2, 2, 2, 3) # Tableau 4D avec valeurs aléatoires
print("Tableau de zéros 4D :\n", zeros_4d)
print("Tableau de uns 4D :\n", ones_4d)
print("Tableau aléatoire 4D :\n", random_4d)
print("Shape du tableau 4D de zéros :", zeros_4d.shape)
print("Shape du tableau 4D de uns :", ones_4d.shape)
print("Shape du tableau 4D aléatoire :", random_4d.shape)
```

```
Tableau de zéros 4D :
[[[0. 0. 0.]
 [0. 0. 0.]]]
```

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

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

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

```
Tableau de uns 4D :
[[[1. 1. 1.]
 [1. 1. 1.]]]
```

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

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

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

```
Tableau aléatoire 4D :
[[[0.11143641 0.06613881 0.67268994]
 [0.82117586 0.01085353 0.79056702]]]
```

```
[[[0.10592643 0.70953783 0.93397612]
 [0.52878084 0.95560971 0.53778829]]]
```

```
[[[0.38007699 0.83132193 0.97679489]
 [0.49817799 0.99955626 0.90732438]]]
```

```
[[[0.40015659 0.12304455 0.30314821]
 [0.317856 0.12217656 0.83151587]]]]
```

```
Shape du tableau 4D de zéros : (2, 2, 2, 3)
Shape du tableau 4D de uns : (2, 2, 2, 3)
Shape du tableau 4D aléatoire : (2, 2, 2, 3)
```



Manipulation des données sur NumPy



Accéder aux éléments d'un tableau avec NumPy

```
Entrée [17]: A = np.array([[10, 20, 30], [40, 50, 60], [70, 80, 90]])  
print(A)  
print(A[1, 2]) # Accède à l'élément ligne 2, colonne 3
```

```
[[10 20 30]  
 [40 50 60]  
 [70 80 90]]  
60
```

Modifier un élément d'un tableau avec NumPy

```
Entrée [18]: print("Avant modification :")  
print(A)  
  
A[1, 2] = 99 # Modifier l'élément à la ligne 1, colonne 2  
  
print("Après modification :")  
print(A)
```

```
Avant modification :  
[[10 20 30]  
 [40 50 60]  
 [70 80 90]]
```

```
Après modification :  
[[10 20 30]  
 [40 50 99]  
 [70 80 90]]
```

NumPy permet de modifier les valeurs directement en réassignant un indice spécifique.



Manipulation des données sur NumPy



Obtenir la dimension et la taille d'un
shape --> tableau
size --> nombre total d'éléments

```
Entrée [19]: print(A.shape) # Affiche (3,3) → 3 Lignes, 3 colonnes
              print(A.size) # Affiche 9 → Nombre total d'éléments
(3, 3)
9
```

```
Entrée [21]: B = np.array([1, 2, 3, 4, 5, 6])
              print("Tableau 1D B : ", B)
              print(B.shape)

              B_reshaped = B.reshape(2, 3) # Transforme en matrice 2x3
              print("Tableau B après reshape :\n", B_reshaped)
              print("Shape après reshape : ", B_reshaped.shape) # (2,3)
```

```
Tableau 1D B : [1 2 3 4 5 6]
(6,)
Tableau B après reshape :
[[1 2 3]
 [4 5 6]]
Shape après reshape : (2, 3)
```

shape permet de vérifier la structure des données.

reshape() est essentiel pour réorganiser les données avant de les utiliser en Machine Learning.

Exemple concret : transformation d'une image 2D en un vecteur 1D pour l'apprentissage automatique.



Manipulation des données sur NumPy

Effectuer des opérations mathématiques avec NumPy
des opérations élément par élément (+, *).



Entrée [25]:

```
A = np.array([[1, 2, 3], [4, 5, 6]])
B = np.array([[1, 2, 3], [4, 5, 6]])

C = A + B # Addition élément par élément
D = A * B # Multiplication élément par élément

print("Somme des matrices A & B :\n", C)
print("Produit des matrices A & B :\n", D)
```

Somme des matrices A & B :

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

Produit des matrices A & B :

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

Condition : les dimensions de A et B doivent être identiques



Manipulation des données sur NumPy

Effectuer des opérations mathématiques avec NumPy



A * B → Multiplication élément par élément (produit d'Hadamard).

A @ B ou np.dot(A, B) → Produit matriciel classique (ligne × colonne).

```
Entrée [26]: A = np.array([[1, 2, 3],  
                         [4, 5, 6]])  
  
B = np.array([[1, 2, 3],  
                         [4, 5, 6]])  
D1 = A * B  
D2 = np.dot(A, B)  
print (D1)  
print (D2)
```

```
-----  
ValueError                                                 Traceback (most recent call last)  
<ipython-input-26-74f758c5946d> in <module>  
      5                                         [4, 5, 6])  
      6 D1 = A * B  
----> 7 D2 = np.dot(A, B)  
      8 print (D1)  
      9 print (D2)  
  
<__array_function__ internals> in dot(*args, **kwargs)  
  
ValueError: shapes (2,3) and (2,3) not aligned: 3 (dim 1) != 2 (dim 0)
```

```
Entrée [30]: A = np.array([[1, 2, 3],  
                         [4, 5, 6]])  
  
B1 = np.array([[1, 2, 3],  
                         [4, 5, 6]])  
B2 = np.array([[1, 2], [3,4] , [5,6]])  
D1 = A * B1  
D2 = np.dot(A, B2)  
print ("le produit element par element est :\n ", D1)  
print ("le produit matriciel est :\n ",D2)
```

```
le produit element par element est :  
[[ 1  4  9]  
 [16 25 36]]  
le produit matriciel est :  
[[22 28]  
 [49 64]]
```



Manipulation des données sur NumPy



- **Gestion des données hétérogènes limitée**
=> **NumPy fonctionne principalement avec des nombres, difficile pour du texte**
- **Manque de flexibilité pour la manipulation des tableaux**
=> **ajout/suppression de colonnes ou lignes complexe**
- **Pas de labels sur les colonnes et lignes**
=> **oblige à utiliser des index numériques,**
- **Gestion inefficace des données manquantes**
=> **NumPy ne gère pas bien les valeurs NaN,**
- **Peu d'outils intégrés pour l'analyse exploratoire des données**
=> **statistiques descriptives, regroupements, etc. (les outils EDA)**



La Bibliothèque pandas

Qu'est-ce que pandas ?



Pandas ; "**Panel Data**", faisant référence aux données multi-temporelles.
une bibliothèque open-source conçue pour la manipulation et l'analyse de données
Objectif : Offrir un outil puissant, flexible et performant pour l'analyse de données.

- Permet de travailler avec des données tabulaires (**DataFrames**), plus intuitif pour la **Data Science**.
- Gère les données hétérogènes (**nombres, textes, dates...**).
- Facilite le filtrage, le tri et la manipulation des colonnes et des lignes.
- Meilleure gestion des données manquantes (**NaN**).



Manipulation des Données avec Pandas



Charger un fichier CSV

Entrée [5]:

```
import pandas as pd  
df = pd.read_csv('tested.csv') # Lire un fichier CSV contenant le dataset Titanic  
print(df.head()) # Afficher les 5 premières lignes
```

```
PassengerId  Survived  Pclass  Name  \  
0  892,0,3,"Kelly, Mr. James",male,34.5,0,0,33091...  NaN  NaN  NaN  
1  893,1,3,"Wilkes, Mrs. James (Ellen Needs)",fem...  NaN  NaN  NaN  
2  894,0,2,"Myles, Mr. Thomas Francis",male,62,0,...  NaN  NaN  NaN  
3  895,0,3,"Wirz, Mr. Albert",male,27,0,0,315154,...  
4  896,1,3,"Hirvonen, Mrs. Alexander (Helga E Lin...
```

```
Sex  Age  SibSp  Parch  Ticket  Fare  Cabin  Embarked  
0  NaN  NaN  NaN  NaN  NaN  NaN  NaN  
1  NaN  NaN  NaN  NaN  NaN  NaN  NaN  
2  NaN  NaN  NaN  NaN  NaN  NaN  NaN
```

Aperçu du fichier tested.csv



	A	B	C	D	E	F	G	H
1	PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked							
2	892,0,3,"Kelly, Mr. James",male,34.5,0,0,330911,7.8292,Q							
3	893,1,3,"Wilkes, Mrs. James (Ellen Needs)",female,47,1,0,363272,7,,S							
4	894,0,2,"Myles, Mr. Thomas Francis",male,62,0,0,240276,9.6875,,Q							
5	895,0,3,"Wirz, Mr. Albert",male,27,0,0,315154,8.6625,,S							
6	896,1,3,"Hirvonen, Mrs. Alexander (Helga E Lindqvist)",female,22,1,1,3101298,12.2875,,S							
7	897,0,3,"Svensson, Mr. Johan Cervin",male,14,0,0,7538,9.225,,S							
8	898,1,3,"Connolly, Miss. Kate",female,30,0,0,330972,7.6292,,Q							
9	899,0,2,"Caldwell, Mr. Albert Francis",male,26,1,1,248738,29,,S							
10	900,1,3,"Abrahim, Mrs. Joseph (Sophie Halaut Easu)",female,18,0,0,2657,7.2292,,C							
11	901,0,3,"Davies, Mr. John Samuel",male,21,2,0,A/4 48871,24.15,,S							
12	902,0,3,"Illeff, Mr. Ylio",male,,0,0,349220,7.8958,,S							
13	903,0,1,"Jones, Mr. Charles Cresson",male,46,0,0,694,26,,S							
14	904,1,1,"Snyder, Mrs. John Pillsbury (Nelle Stevenson)",female,23,1,0,21228,82.2667,B45,S							
15	905,0,2,"Howard, Mr. Benjamin",male,63,1,0,24065,26,,S							
16	906,1,1,"Chaffee, Mrs. Herbert Fuller (Carrie Constance Toogood)",female,47,1,0,W.E.P. 5734,61.175,E31,S							
17	907,1,2,"del Carlo, Mrs. Sebastiano (Argenia Genovesi)",female,24,1,0,SC/PARIS 2167,27.7208,,C							
18	908,0,2,"Keane, Mr. Daniel",male,35,0,0,233734,12.35,,Q							
19	909,0,3,"Assaf, Mr. Gerios",male,21,0,0,2692,7.225,,C							
20	910,1,2,"Ulmakangas, Miss. Ida Livija",female,27,1,0,STON/O2 3101270,7.925,S							

lire un fichier CSV contenant le dataset
Titanic, qui a été téléchargé
à partir de Kaggle.



Manipulation des Données avec Pandas



Exploration des Données : Afficher les premières et dernières lignes

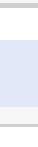
```
Entrée [7]: print(df.head(3)) # Affiche les 3 premières lignes
```



```
PassengerId  Survived  Pclass  Name  \
0  892,0,3,"Kelly, Mr. James",male,34.5,0,0,33091...
1  893,1,3,"Wilkes, Mrs. James (Ellen Needs)",fem...
2  894,0,2,"Myles, Mr. Thomas Francis",male,62,0,...
```

```
Sex  Age  SibSp  Parch  Ticket  Fare  Cabin  Embarked
0   NaN  NaN    NaN    NaN    NaN  NaN    NaN      NaN
1   NaN  NaN    NaN    NaN    NaN  NaN    NaN      NaN
2   NaN  NaN    NaN    NaN    NaN  NaN    NaN      NaN
```

```
Entrée [8]: print(df.tail(2)) # Affiche les 2 dernières lignes
```



```
PassengerId  Survived  Pclass  \
416  1308,0,3,"Ware, Mr. Frederick",male,,0,0,35930...
417  1309,0,3,"Peter, Master. Michael J",male,,1,1,...
```

```
Name  Sex  Age  SibSp  Parch  Ticket  Fare  Cabin  Embarked
416  NaN  NaN  NaN    NaN    NaN    NaN  NaN    NaN      NaN
417  NaN  NaN  NaN    NaN    NaN    NaN  NaN    NaN      NaN
```



Manipulation des Données avec Pandas

Exploration des Données : Obtenir des informations générales



Obtenir des informations sur le DataFrame

Entrée [10]: `print(df.info())` # Affiche la structure des données

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   PassengerId     418 non-null    object 
 1   Survived        0 non-null     float64
 2   Pclass          0 non-null     float64
 3   Name            0 non-null     float64
 4   Sex             0 non-null     float64
 5   Age             0 non-null     float64
 6   SibSp          0 non-null     float64
 7   Parch          0 non-null     float64
 8   Ticket          0 non-null     float64
 9   Fare            0 non-null     float64
 10  Cabin           0 non-null     float64
 11  Embarked        0 non-null     float64
dtypes: float64(11), object(1)
memory usage: 39.3+ KB
None
```

Manipulation des Données avec Pandas

Manipulation et Transformation des Exploration des Données



```
Entrée [14]: print(df ['PassengerId'])
```

```
0    892,0,3,"Kelly, Mr. James",male,34.5,0,0,33091...
1    893,1,3,"Wilkes, Mrs. James (Ellen Needs)",fem...
2    894,0,2,"Myles, Mr. Thomas Francis",male,62,0,....
3    895,0,3,"Wirz, Mr. Albert",male,27,0,0,315154,...
4    896,1,3,"Hirvonen, Mrs. Alexander (Helga E Lin...
          ...
413   1305,0,3,"Spector, Mr. Woolf",male,,0,0,A.5. 3...
414   1306,1,1,"Oliva y Ocana, Dona. Fermina",female...
415   1307,0,3,"Saether, Mr. Simon Sivertsen",male,3...
416   1308,0,3,"Ware, Mr. Frederick",male,,0,0,35930...
417   1309,0,3,"Peter, Master. Michael J",male,,1,1,...
Name: PassengerId, Length: 418, dtype: object
```

Accéder à une colonne

```
Entrée [16]: df['plus'] = '+1' # Toutes les lignes auront la valeur '+1'
print(df)
```

```
PassengerId  Survived  Pclass \
0    892,0,3,"Kelly, Mr. James",male,34.5,0,0,33091...    NaN    NaN
1    893,1,3,"Wilkes, Mrs. James (Ellen Needs)",fem...    NaN    NaN
2    894,0,2,"Myles, Mr. Thomas Francis",male,62,0,...    NaN    NaN
3    895,0,3,"Wirz, Mr. Albert",male,27,0,0,315154,...    NaN    NaN
4    896,1,3,"Hirvonen, Mrs. Alexander (Helga E Lin...    NaN    NaN
...
413   1305,0,3,"Spector, Mr. Woolf",male,,0,0,A.5. 3...    NaN    NaN
414   1306,1,1,"Oliva y Ocana, Dona. Fermina",female...    NaN    NaN
415   1307,0,3,"Saether, Mr. Simon Sivertsen",male,3...    NaN    NaN
416   1308,0,3,"Ware, Mr. Frederick",male,,0,0,35930...    NaN    NaN
417   1309,0,3,"Peter, Master. Michael J",male,,1,1,...
```

	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	plus
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1
...
413	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1
414	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1
415	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1
416	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1
417	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	+1

[418 rows x 13 columns]

Ajouter une nouvelle colonne



Manipulation des Données avec Pandas



Exploration des Données

```
Entrée [11]: print(df.loc[1]) # pour chaque colonne on affiche la valeur de la 1ère ligne
```

```
PassengerId    893, 1, 3, "Wilkes, Mrs. James (Ellen Needs)", fem...
Survived          NaN
Pclass            NaN
Name              NaN
Sex              NaN
Age              NaN
SibSp            NaN
Parch            NaN
Ticket           NaN
Fare              NaN
Cabin           NaN
Embarked         NaN
plus             +1
Name: 1, dtype: object
```

Accéder à une colonne ou une ligne



Manipulation des Données avec Pandas



Manipulation et Transformation des Données

```
Entrée [13]: data = {'Nom': ['Alice', 'Bob', 'Charlie'], 'Âge': [25, 30, 35], 'Score': [90, 85, 88]}\n            df = pd.DataFrame(data)\n            print(df)
```

	Nom	Âge	Score
0	Alice	25	90
1	Bob	30	85
2	Charlie	35	88

```
Entrée [14]: df['Ville'] = ['Paris', 'Lyon', 'Marseille']\n            print(df)
```

	Nom	Âge	Score	Ville
0	Alice	25	90	Paris
1	Bob	30	85	Lyon
2	Charlie	35	88	Marseille

Ajouter une nouvelle colonne



Manipulation des Données avec Pandas



Création et Chargement d'un DataFrame

```
Entrée [31]: import pandas as pd

data = {'Nom': ['Hassan', 'Fatima', 'Omar', 'Khadija', 'Youssef'],
        'Prénom': ['El Amrani', 'Bennani', 'Ouazzani', 'El Idrissi', 'Alaoui'],
        'Âge': [27, 34, 29, 42, 37],
        'Ville': ['Casablanca', 'Rabat', 'Marrakech', 'Fès', 'Tanger']}

df = pd.DataFrame(data)
print(df)
```

	Nom	Prénom	Âge	Ville
0	Hassan	El Amrani	27	Casablanca
1	Fatima	Bennani	34	Rabat
2	Omar	Ouazzani	29	Marrakech
3	Khadija	El Idrissi	42	Fès
4	Youssef	Alaoui	37	Tanger

**On a un DataFrame avec des noms
et villes marocaines, prêt à être manipulé**

Création d'un DataFrame à partir d'un dictionnaire : clés/valeurs



Manipulation des Données avec Pandas

Manipulation et Transformation des Données



Ajouter une nouvelle colonne (Profession)

```
Entrée [19]: df['Profession'] = ['Médecin', 'Ingénieur', 'Enseignant', 'Avocate', 'Commerçant']
print(df)
```

	Nom	Prénom	Âge	Ville	Profession
0	Hassan	El Amrani	27	Casablanca	Médecin
1	Fatima	Bennani	34	Rabat	Ingénieur
2	Omar	Ouazzani	29	Marrakech	Enseignant
3	Khadija	El Idrissi	42	Fès	Avocate
4	Youssef	Alaoui	37	Tanger	Commerçant



Manipulation des Données avec Pandas

Manipulation et Transformation des Données



Modifier une colonne

```
Entrée [20]: df[ 'Âge' ] = df[ 'Âge' ] + 2
               print(df)
```

	Nom	Prénom	Âge	Ville	Profession
0	Hassan	El Amrani	29	Casablanca	Médecin
1	Fatima	Bennani	36	Rabat	Ingénieur
2	Omar	Ouazzani	31	Marrakech	Enseignant
3	Khadija	El Idrissi	44	Fès	Avocate
4	Youssef	Alaoui	39	Tanger	Commerçant



Manipulation des Données avec Pandas

Manipulation et Transformation des Données



Filtrer les données

```
Entrée [21]: df_filtre = df[df['Âge'] > 30]
              print(df_filtre)
```

	Nom	Prénom	Âge	Ville	Profession
1	Fatima	Bennani	36	Rabat	Ingénieur
2	Omar	Ouazzani	31	Marrakech	Enseignant
3	Khadija	El Idrissi	44	Fès	Avocate
4	Youssef	Alaoui	39	Tanger	Commerçant



Manipulation des Données avec Pandas

Manipulation et Transformation des Données



Supprimer une colonne

```
Entrée [22]: df = df.drop(columns=['Profession'])  
print(df)
```

	Nom	Prénom	Âge	Ville
0	Hassan	El Amrani	29	Casablanca
1	Fatima	Bennani	36	Rabat
2	Omar	Ouazzani	31	Marrakech
3	Khadija	El Idrissi	44	Fès
4	Youssef	Alaoui	39	Tanger



Manipulation des Données avec Pandas

Manipulation et Transformation des Données



Aggregation et tri des données

```
Entrée [23]: df_grouped = df.groupby('Ville')['Âge'].mean()  
print(df_grouped)
```

```
Ville  
Casablanca    29  
Fès          44  
Marrakech   31  
Rabat        36  
Tanger       39  
Name: Âge, dtype: int64
```

Regrouper par ville et calculer l'âge moyen

Trier les individus par âge décroissant

```
Entrée [24]: df_sorted = df.sort_values(by='Âge', ascending=False)  
print(df_sorted)
```

	Nom	Prénom	Âge	Ville
3	Khadija	El Idrissi	44	Fès
4	Youssef	Alaoui	39	Tanger
1	Fatima	Bennani	36	Rabat
2	Omar	Ouazzani	31	Marrakech
0	Hassan	El Amrani	29	Casablanca





- **Manipuler des données réelles avec Pandas**
- **Appliquer des transformations et analyses avancées**
- **Faciliter l'organisation et le nettoyage des données en Data Science**

Mais Pandas présente certaines limitations

:

- **Visualisation limitée → Pandas permet de tracer des graphiques simples, mais manque de flexibilité.**
- **Manque d'interactivité → Impossible d'explorer visuellement les données de manière avancée.**
- **Difficulté pour représenter des tendances et des distributions complexes.**



Bibliothèques essentielles



Matplotlib est une bibliothèque Python utilisée pour la visualisation des données. Elle permet de créer des **graphes** et des **diagrammes** tels que des courbes, des histogrammes, des scatter plots, etc., de manière simple et flexible. 

 **Matplotlib est utilisé en Machine Learning pour visualiser les données et évaluer les résultats des modèles.**



A suivre ,,,,