

Fondements et applications de l'apprentissage automatique

B. HILALI - brahill79@gmail.com / T. SABRI – sabritarik@gmail.com
Filière: Ingénierie Logicielle et Intégration des Systèmes Informatiques (ILISI)
24-25

TP 1 : Découverte de Python pour l'apprentissage automatique

- **Objectif :** Maîtriser les bases de Python pour manipuler et visualiser des données.
- **Contenu :**
 - Syntaxe de base et structures de données : listes, dictionnaires, tuples.
 - Fonctions et programmation modulaire.
 - Introduction aux bibliothèques essentielles : Pandas, NumPy, Matplotlib.

INSTALLATION & CONFIGURATION

1. Introduction à Anaconda

- Qu'est-ce qu'Anaconda ?

Anaconda est une distribution Python spécialisée pour les sciences des données, l'apprentissage automatique et le calcul scientifique.

Il inclut :

- ✓ **Python** et ses bibliothèques essentielles.
- ✓ **Conda** : un gestionnaire de paquets et d'environnements.
- ✓ **Jupyter Notebook** : un outil interactif pour écrire et exécuter du code.
- ✓ **Bibliothèques** : pandas, numpy, matplotlib, scikit-learn, etc.

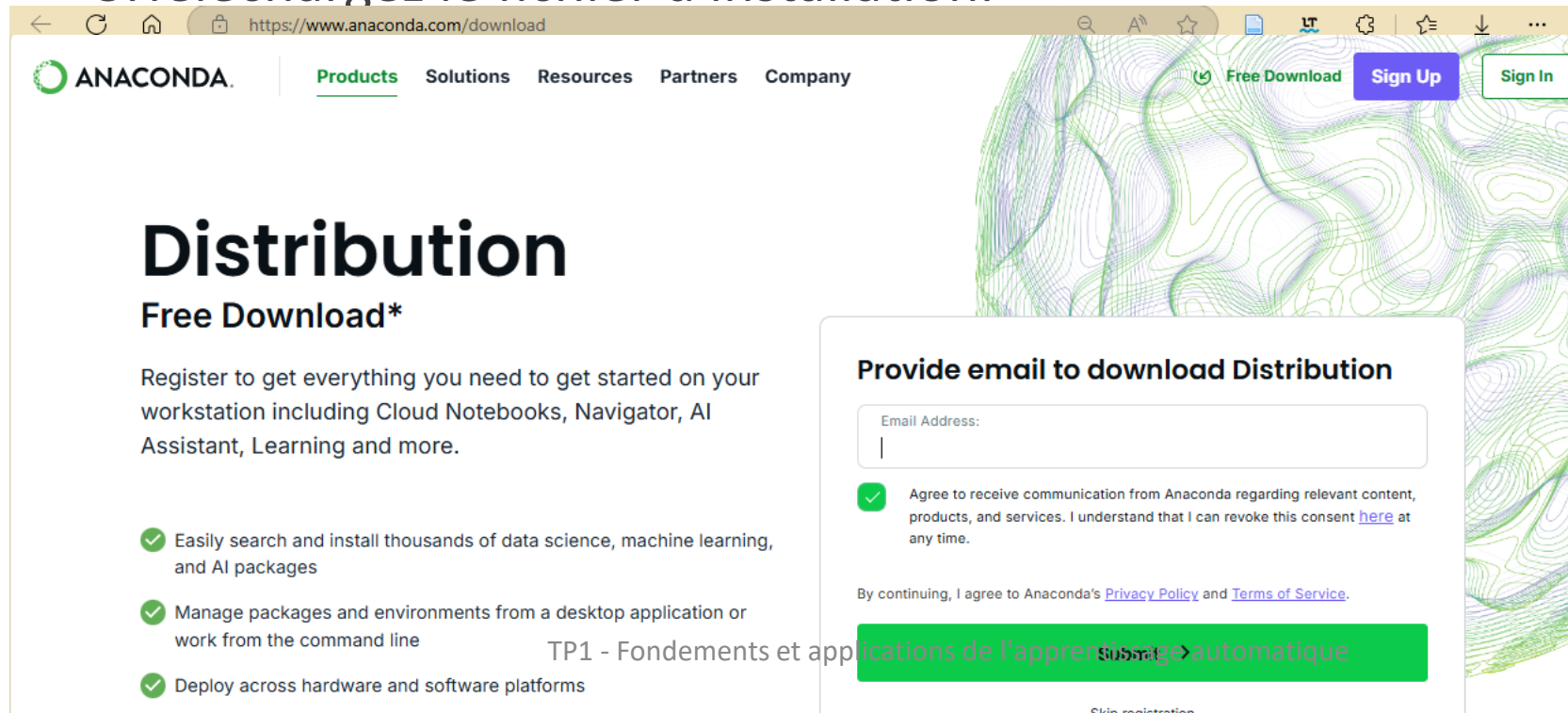
1. Introduction à Anaconda

- Pourquoi utiliser Anaconda ?
 - ✓ **Simplicité** : Installation facile de Python et des bibliothèques.
 - ✓ **Gestion des environnements** : Création d'environnements isolés pour différents projets.
 - ✓ **Compatibilité** : Optimisé pour les sciences des données et l'apprentissage automatique.

2. Téléchargement et Installation d'Anaconda

- **Étape 1 : Télécharger Anaconda**

1. Allez sur le site officiel d'Anaconda :
<https://www.anaconda.com/download>
2. Choisissez la version appropriée pour votre système d'exploitation (Windows, macOS, Linux).
3. Téléchargez le fichier d'installation.



2. Téléchargement et Installation d'Anaconda

- **Étape 2 : Installer et vérifier Anaconda**

- ✓ Exécutez le fichier d'installation téléchargé.
- ✓ Suivez les instructions de l'installateur.
- ✓ Ouvrez un terminal (ou l'invite de commandes sous Windows).
- ✓ Tapez la commande suivante pour vérifier l'installation.

`conda --version`

- ✓ Si Anaconda est correctement installé, la version de Conda s'affiche.

3. Ajouter des modules avec conda

- **Installer des modules**

- ✓ Anaconda inclut déjà de nombreuses bibliothèques, mais vous pouvez en ajouter d'autres avec Conda.
- ✓ Par exemple, pour installer pandas et matplotlib :

1. Ouvrez un terminal.
2. Tapez les commandes suivantes :

`conda install pandas`
`conda install matplotlib`

- ✓ Conda télécharge et installe automatiquement les modules et leurs dépendances..

3. Ajouter des modules avec conda

- **Installer des modules**

- ✓ Anaconda inclut déjà de nombreuses bibliothèques, mais vous pouvez en ajouter d'autres avec Conda.
- ✓ Par exemple, pour installer pandas et matplotlib :

1. Ouvrez un terminal.
2. Tapez les commandes suivantes :

`conda install pandas`
`conda install matplotlib`

- ✓ Conda télécharge et installe automatiquement les modules et leurs dépendances..

4. Utiliser Jupyter Notebook

Introduction :

Jupyter Notebook est un environnement interactif pour coder, visualiser des données et documenter votre travail.

Fonctionnalités clés :

- Exécution de code par cellules (Python, R, Julia, etc.).
- Support pour le texte enrichi (Markdown, LaTeX).
- Visualisation de données intégrée (matplotlib, seaborn, etc.).

Avantages :

- Idéal pour l'exploration de données et le prototypage.
- Collaboration facilitée via le partage de notebooks.
- Intégration avec des outils scientifiques (pandas, numpy, etc.).

LES BASES

1. Structures Conditionnelles

- Les conditions permettent d'exécuter du code en fonction de certaines conditions.

- **Syntaxe de base**

if condition1:

Bloc de code si condition1 est vraie

elif condition2:

Bloc de code si condition2 est vraie

else:

Bloc de code si aucune condition n'est vraie

1. Structures Conditionnelles

- Opérateurs utiles

- ✓ Comparaison : `==`, `!=`, `>`, `<`, `>=`, `<=`

- ✓ Logiques : `and`, `or`, `not`

1. Structures Conditionnelles

- Exemple

```
if statut == "manager":  
    print("Accès à la salle de réunion autorisé.")  
elif statut == "employé" and 9 <= heure <= 18:  
    print("Accès autorisé pendant les heures de travail.")  
elif statut == "employé":  
    print("Accès refusé en dehors des heures de travail.")  
elif statut == "stagiaire":  
    print("Accès à la salle de réunion refusé pour les stagiaires.")  
else:  
    print("Statut non reconnu. Accès refusé.")
```

2. Boucles

- Les boucles permettent de répéter des actions.
- Boucle for
- Utilisée pour itérer sur une séquence (liste, chaîne de caractères, etc.)..
- Syntaxe de base

for élément in séquence:
Bloc de code à répéter

```
for i in range(5):  
    print(i) # Affiche 0, 1, 2, 3, 4
```

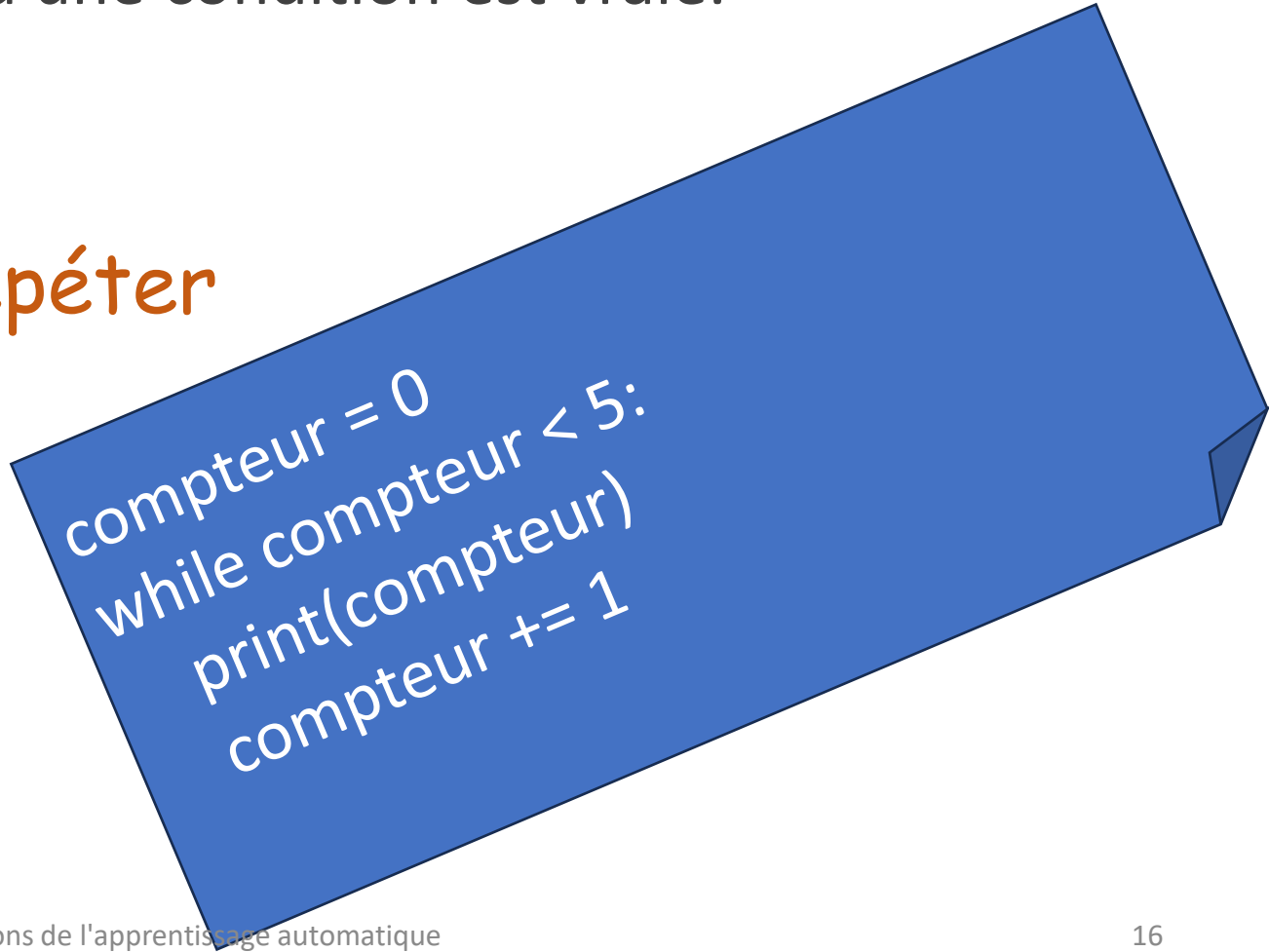
2. Boucles

- Boucle while

- Répète un bloc de code tant qu'une condition est vraie.
- Syntaxe de base

while condition:

Bloc de code à répéter

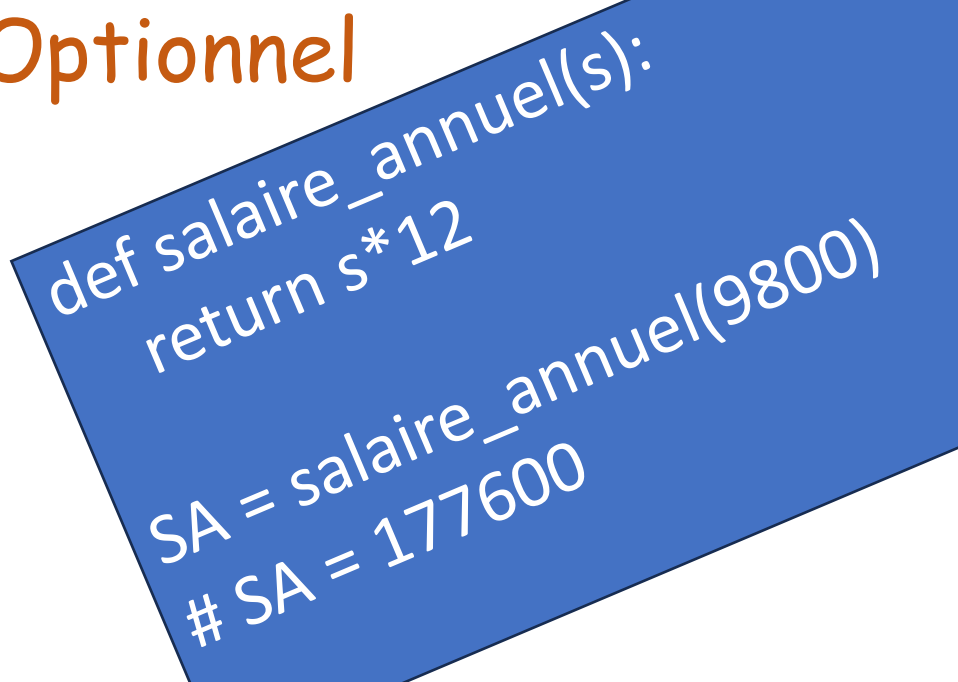


```
compteur = 0  
while compteur < 5:  
    print(compteur)  
    compteur += 1
```


3. Fonctions

- Les fonctions permettent de structurer et de réutiliser du code.
- **Définition d'une fonction**

```
def nom_fonction(param1, param2):  
    # Bloc de code  
    return résultat # Optionnel
```



```
def salaire_annuel(s):  
    return s*12  
  
SA = salaire_annuel(9800)  
# SA = 177600
```

4. Listes

- Les listes sont des collections ordonnées et modifiables d'éléments.

- **Création et accès**

```
ma_liste = [1, 2, 3, 4]  
print(ma_liste[0])
```

- **Fonctions principales**

- Ajouter un élément : `append()`, `insert()`
- Supprimer un élément : `remove()`, `pop()`
- Trier : `sort()`
- Longueur : `len()`
- Trouver un élément : `index()`

5. Dictionnaires

- Les dictionnaires stockent des paires clé-valeur.

- **Création et accès**

```
D = {"clé1": "valeur1", "clé2": "valeur2"}  
print(D["clé1"]) # Accès à "valeur1"
```

- **Fonctions principales**

- Ajouter/modifier une valeur : `mon_dictionnaire[clé] = valeur`
- Supprimer une clé : `del mon_dictionnaire[clé]`
- Vérifier l'existence d'une clé : `clé in mon_dictionnaire`
- Obtenir toutes les clés : `keys()`
- Obtenir toutes les valeurs : `values()`
- Obtenir les paires clé-valeur : `items()`

6. Tuples

- Les tuples sont des collections ordonnées et non modifiables d'éléments.

- **Création et accès**

```
mon_tuple = (1, 2, 3)
```

```
print(mon_tuple[0]) # Accès au premier élément (1)
```

- **Fonctions principales**

- Longueur : len()

- Compter les occurrences : count()

- Trouver un élément : index()

MODULE: PANDAS

1. Qu'est-ce que Pandas ?

- Bibliothèque Python open-source pour la manipulation et l'analyse de données.
- Conçue pour travailler avec des données structurées (tableaux, séries temporelles, etc.).
- **Utilité:**
 - Importation et exportation de données (CSV, Excel, SQL, etc.).
 - Nettoyage des données (valeurs manquantes, doublons, etc.).
 - Transformation des données (filtrage, agrégation, fusion, etc.).
 - Analyse exploratoire des données (statistiques, visualisation, etc.).

1. Qu'est-ce que Pandas ?

- Pourquoi Pandas ?
- Syntaxe simple et intuitive.
- Performant pour des datasets de taille moyenne.
- Intégration facile avec d'autres bibliothèques (NumPy, Matplotlib, Scikit-learn).

2. Structures de données de Pandas

- **Series :**
- Objet unidimensionnel similaire à un tableau ou une liste.
- Exemple : `s = pd.Series([1, 2, 3, 4])`
- **DataFrame :**
- Structure bidimensionnelle (comme un tableau Excel ou une table SQL).
- Exemple : `df = pd.DataFrame({'col1': [1, 2], 'col2': ['A', 'B']})`

3. Fonctions de base

1. Lecture des données :

- `pd.read_csv('fichier.csv')` : Lit un fichier CSV.
- `pd.read_excel('fichier.xlsx')` : Lit un fichier Excel.

2. Inspection des données :

- `df.head(n)` : Affiche les n premières lignes.
- `df.tail(n)` : Affiche les n dernières lignes.
- `df.info()` : Affiche les informations sur le DataFrame (types de données, valeurs manquantes).
- `df.describe()` : Résumé statistique des colonnes numériques.

3. Fonctions de base

3. Sélection des données :

- `df['colonne']` : Sélectionne une colonne.
- `df.loc[index]` : Sélectionne une ligne par son index.
- `df.iloc[indice]` : Sélectionne une ligne par son indice numérique.

4. Filtrage des données :

- `df[df['colonne'] > valeur]` : Filtre les lignes selon une condition.
- `df.query('colonne > valeur')` : Filtre avec une expression.

3. Fonctions de base

5. Manipulation des données :

- `df.dropna()` : Supprime les lignes avec des valeurs manquantes.
- `df.fillna(valeur)` : Remplace les valeurs manquantes par une valeur spécifiée.
- `df.drop_duplicates()` : Supprime les doublons.

6. Transformation des données :

- `df.sort_values('colonne')` : Trie les données par une colonne.
- `df.groupby('colonne').mean()` : Agrège les données par groupe.
- `df.pivot_table(values='A', index='B', columns='C')` : Crée un tableau croisé dynamique.

3. Fonctions de base

7. Fusion et concaténation :

- `pd.concat([df1, df2])` : Concatène deux DataFrames.
- `pd.merge(df1, df2, on='colonne')` : Fusionne deux DataFrames sur une colonne.

8. Exportation des données :

- `df.to_csv('fichier.csv')` : Exporte en CSV.
- `df.to_excel('fichier.xlsx')` : Exporte en Excel.

4. Application

- Vous allez travailler sur un dataset contenant les informations des employés d'une entreprise. Ce fichier CSV comprend plusieurs colonnes, comme l'ID de l'employé, le nom, le prénom, l'email, le numéro de téléphone, la date d'embauche, l'ID du poste, le salaire, le pourcentage de commission, l'ID du manager, et l'ID du département.
- **TAF:**
 1. Importer le fichier « **employees.csv** » - [Télécharger](#)
 2. Afficher les 5 premières lignes du dataset pour obtenir un aperçu initial des données.
 3. Obtenir les types de données de chaque colonne du dataset et identifier les colonnes qui nécessitent un changement de type.

4. Application

4. Afficher le nombre de lignes et de colonnes dans le dataset.
5. Afficher seulement les colonnes : **FIRST_NAME** et **LAST_NAME**
6. Afficher seulement les colonnes : **FIRST_NAME** et **LAST_NAME** pour les lignes 7 à 14
7. Créer une colonne **FULL_NAME** à partir des colonnes : **FIRST_NAME** et **LAST_NAME**
8. Créer une colonne **ANNUAL_SALARY** qui représente le salaire annuel
9. Lister les colonnes avec des valeurs manquantes et indiquer combien de valeurs sont manquantes par colonne.
10. Calculer le salaire moyen des employés.

4. Application

11. Afficher les employés dont le salaire est supérieur à la moyenne des salaires et savoir le nombre de résultats
12. Trier les employés par salaire en ordre décroissant.
13. Extraire les données des employés ayant le rôle '**JOB_ID**' spécifique, comme '**SH_CLERK**'.
14. Afficher les employés qui ont été embauchés après 2006.
15. Compter le nombre d'employés par département.
16. Compter les employés embauchés chaque année
17. Calculer le salaire moyen par département et afficher le résultat.

4. Application

18. Trouver les départements avec un salaire moyen supérieur à 7000.
19. Calculer le salaire annuel total de chaque département.
20. Afficher les 5 employés ayant le salaire le plus élevé.
21. Afficher les 5 employés ayant le salaire le plus bas.
22. Afficher le nombre d'employés par rôle (**JOB_ID**).
23. Modifier les pourcentages de commission (**COMMISSION_PCT**) par 0.1.
24. Calculer le salaire total (incluant la commission) pour chaque employé, en ajoutant la commission au salaire si applicable.
25. Afficher les employés dont le manager a l'ID 124.

4. Application

26. Calculer le salaire total moyen (en incluant la commission) de chaque rôle (**JOB_ID**).
27. Filtrer les employés ayant un numéro de téléphone qui commence par un certain préfixe (par exemple, '650').
28. Calculer l'ancienneté (en années) de chaque employé en soustrayant la date d'embauche de la date actuelle.
29. Afficher les employés ayant un salaire supérieur à la moyenne de leur département.
30. Afficher le salaire maximum par département
31. Afficher les employés qui n'ont pas de numéro de téléphone.

MODULE: NUMPY

1. Qu'est-ce que NumPy ?

- Bibliothèque Python pour le calcul numérique.
- Fournit des tableaux multidimensionnels (ndarrays) et des fonctions pour les manipuler.
- **Utilité:**
- Opérations mathématiques et statistiques performantes.
- Base pour de nombreuses bibliothèques scientifiques (Pandas, SciPy, Scikit-learn, etc.).

2. Structures de données de NumPy

- **ndarray** :
- Tableau multidimensionnel homogène (tous les éléments ont le même type).
- Exemple : `array = np.array([1, 2, 3])`.
- **Attributs importants:**
- `array.shape` : Dimensions du tableau.
- `array.dtype` : Type des éléments du tableau.
- `array.size` : Nombre total d'éléments.

3. Fonctions de base

1. Création de tableaux:

- `np.array([1, 2, 3])` : Crée un tableau à partir d'une liste.
- `np.zeros((3, 3))` : Crée un tableau rempli de zéros.
- `np.ones((2, 2))` : Crée un tableau rempli de uns.
- `np.arange(0, 10, 2)` : Crée un tableau avec des valeurs espacées régulièrement.

2. Manipulation des tableaux:

- `array.reshape((2, 3))` : Change la forme du tableau.
- `array.flatten()` : Aplatit un tableau en une dimension.
- `np.concatenate((array1, array2))` : Concatène des tableaux.

3. Fonctions de base

3. Opérations mathématiques :

- `np.sum(array)` : Somme des éléments.
- `np.mean(array)` : Moyenne des éléments.
- `np.max(array)` : Valeur maximale.
- `np.min(array)` : Valeur minimale.
- `np.std(array)` : Écart-type.

4. Indexation et découpage:

- `array[0, 1]` : Accède à un élément spécifique.
- `array[:, 1]` : Sélectionne une colonne.
- `array[1:3, :]` : Sélectionne une plage de lignes.

3. Fonctions de base

5. Algèbre linéaire :

- `np.dot(array1, array2)` : Produit matriciel.
- `np.linalg.inv(array)` : Inverse d'une matrice.
- `np.linalg.det(array)` : Déterminant d'une matrice..

6. Génération de nombres aléatoires :

- `np.random.rand(3, 3)` : Génère un tableau de nombres aléatoires entre 0 et 1.
- `np.random.randint(0, 10, size=(2, 2))` : Génère des entiers aléatoires

3. Fonctions de base

7. Manipulation de fichiers :

- `np.save('fichier.npy', array)` : Sauvegarde un tableau dans un fichier.
- `np.load('fichier.npy')` : Charge un tableau depuis un fichier

4. Application

Enoncé de l'exercice

MODULE: MATPLOTLIB

1. Qu'est-ce que Matplotlib ?

- **Définition** : Matplotlib est une bibliothèque Python open source pour la visualisation de données.
Elle permet de créer des graphiques statiques, animés et interactifs.
- **Principales fonctionnalités** :
 - Création de graphiques 2D et 3D.
 - Support pour une grande variété de types de graphiques (lignes, barres, histogrammes, etc.).
 - Personnalisation avancée (couleurs, styles, annotations).

1. Qu'est-ce que Matplotlib ?

- **Utilisations courantes :**

- Visualisation de données scientifiques.
- Analyse exploratoire des données (EDA).
- Création de rapports et présentations.

- **Exemple d'utilisation :**

```
import matplotlib.pyplot as plt  
plt.plot([1, 2, 3], [4, 5, 1])  
plt.title("Exemple simple")  
plt.show()
```

- **Avantages :**

- Intégration facile avec Jupyter Notebook.
- Compatible avec d'autres bibliothèques (pandas, numpy).
- Grande communauté et documentation complète.

3. Fonctions de base

1. Importation :

- `import matplotlib.pyplot as plt`

2. Créer un graphique simple :

- `plt.plot([1, 2, 3], [4, 5, 1])` # Tracer une ligne
- `plt.show()` # Afficher le graphique

3. Ajouter des titres et labels :

- `plt.title("Titre du graphique")`
- `plt.xlabel("Axe X")`
- `plt.ylabel("Axe Y")`

4. Types de graphiques courants

1. Graphique en ligne:

- `plt.plot(x, y)`

2. Graphique en barres:

- `plt.bar(x, height)`

3. Histogramme :

- `plt.hist(data, bins=10)`

4. Types de graphiques courants

4. Nuage de points:

- `plt.scatter(x, y)`

5. Personnalisation avancée :

- Légende : `plt.legend()`
- Grille : `plt.grid(True)`
- Limites des axes : `plt.xlim()` et `plt.ylim()`

5. Application

Enoncé de l'exercice