Bootcamp — Introduction à l'Intelligence Artificielle

Atelier (Jour 1) : Python pour l'IA — Pandas, NumPy, Matplotlib

---

Ibrahima SY
Chef du Département IA & Ingénierie des Données — ISI Dakar

Samedi 20 Septembre 2025 12h45–14h15

Objectifs de l'atelier

- Poser le rôle de **Python** dans un workflow IA moderne.
- Manipuler des données tabulaires avec **Pandas**.
- Calculer vite et bien avec **NumPy**.
- Visualiser proprement avec **Matplotlib**.

Agenda (90 minutes)

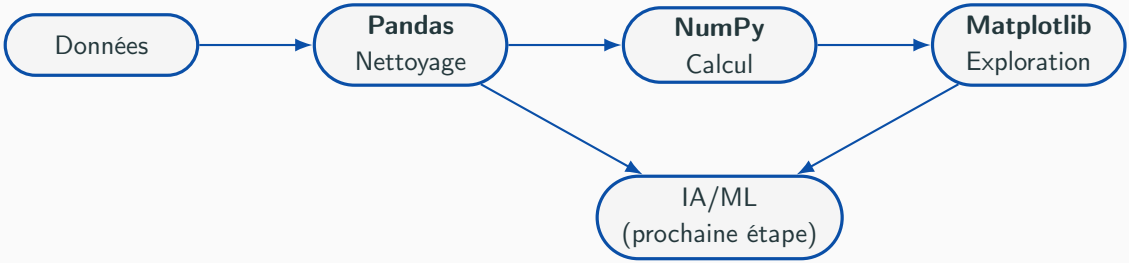
12 :45–12 :55	10’	Introduction & setup
SoftGray **12 :55–13 :15**	20’	Python utile : types, boucles, fonctions
13 :15–13 :35	20’	Pandas : DataFrame, exploration, transformation
SoftGray **13 :35–13 :55**	20’	NumPy : vecteurs, stats, corrélations
13 :55–14 :10	15’	Matplotlib : histogramme, scatter, courbe
SoftGray **14 :10–14 :15**	5’	Mini-atelier & conclusion

Contexte & Motivation

Pourquoi Python en IA ?

- **Simplicité** et lisibilité $\rightarrow$ focus sur le raisonnement.
- **Écosystème** : NumPy, Pandas, Matplotlib, scikit-learn, PyTorch.
- **Interopérabilité** (C/C++, R, Spark, Cloud).
- **Communauté** et documentation abondantes.

Workflow IA : du brut au modèle



Vérification environnement

```
import sys, numpy as np, pandas as pd, matplotlib
print("Python :", sys.version.split()[0])
print("NumPy  :", np.__version__)
print("Pandas :", pd.__version__)
```

Partie 1 — Python utile pour l'IA

Ce qu'il faut maîtriser (et pas plus)

- Types, structures (`list`, `dict`), slicing.
- Boucles, compréhensions, conditions.
- Fonctions claires, pures si possible.
- Lisibilité > "astuces".

Types & structures

```
age = 22; note = 14.5; nom = "Alice"; ok = True
notes = [12, 14, 16, 18]
etu = {"nom": "Alice", "age": 22, "note": 14.5}
```

Boucles & conditions

```
1 for n in notes:
2     print("Bonne note" if n>=15 else "Note moyenne:", n)
```

Fonctions (réutilisables)

```
1 def mention(n):
2     if n>=16: return "Excellent"
3     if n>=14: return "Bien"
4     return "Passable"
5
6 def moyenne(vals): return sum(vals)/len(vals)
```

Exercice rapide

Consignes (5')

Écrire une fonction `resume(notes)` qui renvoie `min`, `max`, `moy`.

Partie 2 — Pandas : manipuler les données

Pourquoi Pandas ?

- **DataFrame** = tableau programmable (Excel++).
- 80% du temps en IA = **préparation des données**.
- Outils clés : `head/info/describe`, filtres, `apply`, `groupby`.

Créer & explorer un DataFrame

```
import pandas as pd
df = pd.DataFrame({
  "nom": ["Alice","Bob","Charlie","Diarra","Eva","Fadel"],
  "campus": ["Dakar","Dakar","Kaolack","Dakar","Kaolack","
     Kaolack"],
  "age": [20,22,19,25,21,23],
  "note": [15,12,17,14,16,13]
})
print(df.head()); print(df.info()); print(df.describe())
```

Sélection & filtrage

```
1 bons = df[df["note"]>=15][["nom","note","campus"]]
2 ados = df[(df["age"]<21)]
```

Transformation & features

```
df["mention"] = df["note"].apply(
  lambda n: "Excellent" if n>=16 else ("Bien" if n>=14 else "
     Passable")
)
df["age2"] = df["age"]**2
```

GroupBy & agrégations

```
agg = df.groupby("campus")["note"].agg(["count","mean","min","max"]).reset_index()
print(agg)
```

Nettoyage (NaN, doublons)

```
df2 = df.copy()
df2.loc[2,"note"] = None
df2["note"] = df2["note"].fillna(df2["note"].mean())
df2 = df2.drop_duplicates()
```

À retenir — Pandas

- Pipeline reproductible (scripts/notebooks versionnés).
- `groupby` raconte des histoires (par campus, filière, etc.).
- `apply` pour des règles métier simples.

Exercice Pandas (5')

Consignes

Ajoutez `resultat` = Succès (note$\geq$14) sinon Échec. Comptez les Succès par campus.

Partie 3 — NumPy : calcul scientifique

Pourquoi NumPy ?

- Vecteurs/matrices rapides (C en dessous).
- Opérations vectorisées $\Rightarrow$ concision & performance.
- Base des tenseurs en Deep Learning.

Vecteurs & stats rapides

```
import numpy as np
a = np.array([1,2,3,4,5], dtype=float)
print("moy:", a.mean(), " var:", a.var(), " std:", a.std())
```

Opérations vectorisées

```
1 b = np.array([2,1,2,1,2], dtype=float)
2 print("somme:", a+b)
3 print("produit:", a*b)
```

Corrélation (intuition)

```
x = np.array([18,20,22,24,26,28])    #  ge
y = np.array([60,65,70,72,75,80])    # score
corr = np.corrcoef(x, y)[0,1]
print("corr(age,score):", round(corr,3))
```

À retenir — NumPy

- Vitesse & compacité (vs. listes Python).
- Outils statistiques de base prêts à l'emploi.
- Corrélation $\neq$ causalité (signal, pas preuve).

Exercice NumPy (5')

Consignes

Générez 50 valeurs aléatoires `score`, calculez moy/var/std, puis normalisez $(x - \mu)/\sigma$.

Partie 4 — Matplotlib : visualiser pour comprendre

Pourquoi visualiser ?

- Voir tendances, outliers, relations.
- Communiquer efficacement.
- Choisir le bon graphe pour la bonne question.

Histogramme : distribution

```
import matplotlib.pyplot as plt
plt.hist(df["note"], bins=5)
plt.title("Distribution des notes"); plt.xlabel("Note"); plt.
    ylabel("Fr quence")
plt.show()
```

Scatter : relation 2 variables

```
plt.scatter(df["age"], df["note"])
plt.title(" ge  vs Note"); plt.xlabel(" ge "); plt.ylabel("Note
    ")
plt.show()
```

Courbe : tendance ordonnée

```
df_sorted = df.sort_values("age")
plt.plot(df_sorted["age"], df_sorted["note"], marker="o")
plt.title("Tendance des notes avec l' ge ")
plt.xlabel(" ge "); plt.ylabel("Note"); plt.show()
```

Bonnes pratiques de graphes

- Titre explicite, axes labellisés, unités.
- Légende seulement si nécessaire.
- Simplicité $>$ effets visuels superflus.

À retenir — Viz

- Histogramme : distribution ; Scatter : relation ; Courbe : tendance.
- Un graphe = une question précise.

Mini-projet : de la donnée brute à la visualisation

Contexte du mini-projet

Scénario

Vous êtes Data Scientist junior au sein de l'ISI. On vous fournit un petit dataset d'étudiants contenant :

- leur âge,
- le nombre d'heures de révision,
- et leur score final.

Votre mission : explorer les données et produire une analyse visuelle rapide.

Étapes attendues

1. Créer un **DataFrame** avec les données.
2. Calculer des **statistiques descriptives** (moyenne, écart-type).
3. Vérifier la **corrélation** entre heures de révision et score.
4. Produire un **scatter plot** bien légendé.

Critères de réussite

- Code propre et sans erreur.
- Graphe lisible (titre, axes, unités).
- Interprétation courte (ex. : “Plus les heures de révision augmentent, plus le score s’améliore”).

Annexes utiles

Installation rapide (pip/venv)

```
python -m venv .venv
source .venv/bin/activate  # Windows: .venv\Scripts\activate
pip install --upgrade pip
pip install numpy pandas matplotlib
```

Alternative (Conda)

```
conda create -n bootcamp-ia python=3.11 -y
conda activate bootcamp-ia
conda install numpy pandas matplotlib -y
```

I/O fréquents

```
df.to_csv("etudiants.csv", index=False)
df2 = pd.read_csv("etudiants.csv")

plt.savefig("figure.png", dpi=150, bbox_inches="tight")
```

Erreurs courantes & solutions

- `KeyError` : nom de colonne erroné → `df.columns`.
- `NaN` inattendus → `isna()`, `fillna()`.
- Graphes vides → filtrage trop strict.

Qualité & éthique des données

- Documenter la source et les hypothèses.
- Limiter les biais, expliquer les variables.
- Sécurité : ne pas exposer de données sensibles.

Rappels clés

- Pandas prépare, NumPy calcule, Matplotlib révèle.
- Scripts/notebooks versionnés $\Rightarrow$ **reproductibilité**.

Conclusion & ouverture

Ce que vous savez faire maintenant

- Construire un DataFrame propre et informatif.
- Calculer des mesures clés et corrélations.
- Visualiser des distributions et relations.

Suite (14 :15–16 :00) — Régression linéaire

- Formulation, apprentissage, évaluation.
- Scikit-learn : pipeline simple.
- Mise en pratique $\rightarrow$ mini-projet.

Merci Questions ?

Ressources

- Documentation : pandas, numpy, matplotlib
- Tutoriels officiels & exemples Jupyter/Colab

Contacts

- **Ibrahima SY** — Département IA & Ingénierie des Données, ISI Dakar
- Email : ibrahima.sy@groupeisi.com