



Programmation Python

Séance 7 — pandas

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et
Techniques, Université Cheikh Anta Diop de Dakar

<https://ebtoure.sn/cours/programmation-python/chapitres/07-pandas.html>

Version du 11 octobre 2026



Version en ligne

Programmation Python

— Séance 7

pandas

Dr. El Hadji Bassirou TOURÉ

Département de Mathématiques et Informatique, Faculté des Sciences et Techniques, Université Cheikh Anta Diop de Dakar

Objectifs de la séance

Contenu

NumPy calcule sur des tableaux de nombres. [pandas](#), bâti sur NumPy, ajoute des **colonnes nommées** et des lignes étiquetées : le **DataFrame**, la table de données de la bio-informatique. On manipule notre jeu de morphométrie de pinsons, cette fois avec une colonne [espece](#).

- **Series** et **DataFrame** ; créer et charger ([read_csv](#)).
- **Explorer** : [shape](#), [info](#), [describe](#).
- **Sélectionner, filtrer, créer une colonne, trier.**
- **Valeurs manquantes** et **groupby** — le cœur de l'analyse.

L'entraînement se fait dans le **notebook**, avec une cellule « À vous » après chaque notion.

Un DataFrame

NaN = valeur manquante

colonnes nommees

	espece	bec_long	bec_haut	aile	masse
0	fuliginosa	9.1	8.0	67	15
1	fortis	13.2	10.9	NaN	22
2	magnirostris	16.1	14.2	80	28
3	fuliginosa	9.4	8.3	68	16

index

un filtre garde des lignes
`df[df["masse"] > 25]`

une colonne = Series
`df["masse"]`

Un DataFrame : des colonnes nommees et un index. On selectionne des colonnes, on filtre des lignes.

Un DataFrame : des colonnes nommées et un index. On sélectionne une colonne (une Series), on filtre des lignes ; une case manquante vaut NaN.

Series et DataFrame

L'idée : la colonne, puis la table

L'idée en une phrase

Une **Series** est une colonne : des valeurs avec une **étiquette** (l'index). Un **DataFrame** réunit plusieurs colonnes partageant le même index — une table. On importe pandas sous le nom `pd` ; tout est bâti sur NumPy.

Une Series : une colonne étiquetée

```
import pandas as pd

masse_moy = pd.Series([15.2, 21.5, 28.0],
                      index=["fuliginosa", "fortis", "magnirostris"])

print(masse_moy)
```

RÉSULTAT →

```
fuliginosa    15.2
fortis        21.5
magnirostris  28.0
dtype: float64
```

Chaque valeur porte une étiquette (le nom d'espèce) plutôt qu'un simple numéro de position.

Créer et charger

L'idée : d'un dictionnaire, ou d'un CSV

L'idée en une phrase

On construit un DataFrame à partir d'un **dictionnaire** (chaque clé devient une colonne) ; mais en pratique, les données arrivent dans un fichier **CSV** que l'on charge avec `read_csv`.

Créer un DataFrame depuis un dictionnaire

```
data = {
    "espece": ["fuliginosa"]*5 + ["fortis"]*6 + ["magnirostris"]*4,
    "bec_long": [9.1, 9.4, 8.9, ...],
    "bec_haut": [8.0, 8.3, 7.8, ...],
    "aile": [67, 68, 66, ...],
    "masse": [15, 16, 14, ...],
}
df = pd.DataFrame(data)
print(df.head())
```

RÉSULTAT →

	espece	bec_long	bec_haut	aile	masse
0	fuliginosa	9.1	8.0	67.0	15
1	fuliginosa	9.4	8.3	68.0	16
2	fuliginosa	8.9	7.8	66.0	14
3	fuliginosa	9.6	8.5	69.0	16
4	fuliginosa	9.2	8.1	67.0	15

`head()` montre les premières lignes. La table compte 15 spécimens.

Sauvegarder et recharger en CSV

```
df.to_csv("pinsons.csv", index=False)
df2 = pd.read_csv("pinsons.csv")
print(df2.head())
```

RÉSULTAT →

	espece	bec_long	bec_haut	ail	masse
0	fuliginosa	9.1	8.0	67.0	15
1	fuliginosa	9.4	8.3	68.0	16
2	fuliginosa	8.9	7.8	66.0	14
3	fuliginosa	9.6	8.5	69.0	16
4	fuliginosa	9.2	8.1	67.0	15

`read_csv` est le point de départ de presque toute analyse de données réelle.

Explorer

Dimensions et résumé statistique

```
print(df.shape)
print(df.describe().round(2))
```

RÉSULTAT →

```
(15, 5)
      bec_long  bec_haut   aile  masse
count      15.00     15.00   14.00   15.00
mean       12.55     10.75   73.64   21.13
std         2.75      2.41    5.36    5.18
min         8.90      7.80   66.00   14.00
25%         9.50      8.40   68.25   16.00
50%        12.90     10.60   74.50   21.00
75%        14.60     12.45   78.25   25.00
max        16.50     14.60   81.00   29.00
```

`describe` résume chaque mesure. `aile` n'a que **14** valeurs (`count`) : une manque.

Colonnes et types : `info()`

```
df.info()
```

RÉSULTAT →

```
<class 'pandas.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0   espece      15 non-null    str
1   bec_long    15 non-null    float64
2   bec_haut    15 non-null    float64
3   aile         14 non-null    float64
4   masse       15 non-null    int64
dtypes: float64(3), int64(1), str(1)
```

`info()` donne le type de chaque colonne et son nombre de valeurs présentes :
`aile` affiche 14, la mesure manquante s'y voit aussi.

Sélectionner

L'idée : colonnes et lignes

L'idée en une phrase

Une colonne se prend par son nom : `df["masse"]` (une Series). Plusieurs colonnes : une **liste** de noms, `df[["espece", "masse"]]`. Une ligne se prend par son étiquette avec `loc`.

Une colonne, plusieurs colonnes

```
print(df[["espece", "masse"]].head())
```

RÉSULTAT →

	espece	masse
0	fuliginosa	15
1	fuliginosa	16
2	fuliginosa	14
3	fuliginosa	16
4	fuliginosa	15

Crochets **simples** pour une colonne (Series) ; crochets **doubles** (une liste de noms) pour plusieurs colonnes (un DataFrame).

Une ligne : loc

```
print(df.loc[0])  
print(df.loc[0, "espece"])
```

RÉSULTAT →

```
espece      fuliginosa  
bec_long      9.1  
bec_haut      8.0  
aile         67.0  
masse         15  
Name: 0, dtype: object  
fuliginosa
```

`df.loc[0]` donne la ligne d'étiquette 0 (comme une Series) ; `df.loc[0, "espece"]` vise une case précise.

Filterer

Filtrer avec un masque booléen

Comme avec NumPy

Une comparaison produit un masque ; entre crochets, il ne garde que les lignes qui vérifient la condition.

```
print(df[df["masse"] > 25])
```

RÉSULTAT →

	espece	bec_long	bec_haut	aile	masse
11	magnirostris	16.1	14.2	80.0	28
12	magnirostris	15.7	13.8	79.0	27
13	magnirostris	16.5	14.6	81.0	29
14	magnirostris	15.9	14.0	80.0	28

Tous les spécimens de plus de 25 g sont des **magnirostris** : le filtre a isolé la plus grande espèce.

Créer une colonne

Ajouter une colonne calculée

```
df["rapport_bec"] = (df["bec_long"] / df["bec_haut"]).round(2)
print(df[["espece", "bec_long", "bec_haut", "rapport_bec"]].head())
```

RÉSULTAT →

	espece	bec_long	bec_haut	rapport_bec
0	fuliginosa	9.1	8.0	1.14
1	fuliginosa	9.4	8.3	1.13
2	fuliginosa	8.9	7.8	1.14
3	fuliginosa	9.6	8.5	1.13
4	fuliginosa	9.2	8.1	1.14

Le calcul est **vectorisé**, comme avec NumPy : la nouvelle colonne s'obtient à partir des colonnes existantes.

Trier

Trier les lignes

```
print(df.sort_values("masse").head())
```

RÉSULTAT →

	espece	bec_long	bec_haut	ail	masse	rapport_bec
2	fuliginosa	8.9	7.8	66.0	14	1.14
0	fuliginosa	9.1	8.0	67.0	15	1.14
4	fuliginosa	9.2	8.1	67.0	15	1.14
1	fuliginosa	9.4	8.3	68.0	16	1.13
3	fuliginosa	9.6	8.5	69.0	16	1.13

`sort_values("masse")` réordonne du plus léger au plus lourd : les plus petits spécimens sont des `fuliginosa`.

Valeurs manquantes

Repérer et combler les trous

L'idée en une phrase

Une mesure absente vaut `NaN`. `isna()` la repère, `.sum()` la compte ; on peut **supprimer** les lignes (`dropna`) ou **combler** le trou (`fillna`).

```
print(df[df["aile"].isna()])
```

RÉSULTAT →

	espece	bec_long	bec_haut	aile	masse	rapport_bec
7	fortis	12.5	10.2	NaN	20	1.23

```
aile_pleine = df["aile"].fillna(df["aile"].mean())  
print(aile_pleine.isna().sum())
```

0

Grouper

L'idée : compter, puis agréger par groupe

L'idée en une phrase

`value_counts()` compte les occurrences de chaque valeur d'une colonne.

`groupby("espece")` rassemble les lignes par espèce ; on applique ensuite une agrégation (`mean`, `sum`...) à **chaque groupe**. C'est le cœur de l'analyse de données.

Compter les effectifs

```
print(df["espece"].value_counts())
```

RÉSULTAT →

```
espece
fortis      6
fuliginosa  5
magnirostris 4
Name: count, dtype: int64
```

L'échantillon compte 6 *fortis*, 5 *fuliginosa* et 4 *magnirostris*.

Le profil moyen de chaque espèce

```
print(df.groupby("espece")[["bec_long", "aile", "masse"]].mean().round(2))
```

RÉSULTAT →

	bec_long	aile	masse
espece			
fortis	12.98	74.8	21.5
fuliginosa	9.24	67.4	15.2
magnirostris	16.05	80.0	28.0

En une ligne, `groupby` révèle le profil de chaque espèce : `magnirostris` a le bec le plus long, l'aile la plus grande et la masse la plus élevée ; `fuliginosa` est la plus petite sur les trois mesures.

Synthèse

Pièges fréquents

- **Une colonne est une Series** : `df["masse"]` (crochets simples). Avec une **liste** de noms, `df[["espece", "masse"]]`, on obtient un DataFrame (crochets doubles).
- **loc (étiquette) vs iloc (position)** : `df.loc[0]` cible l'étiquette 0 ; `df.iloc[0]` la première ligne.
- **NaN n'est pas 0** : une valeur manquante doit être traitée (`dropna`, `fillna`).
- **Un filtre ne modifie pas la table** : `df[df["masse"] > 25]` renvoie une nouvelle vue ; on l'affecte à une variable pour la garder.

À retenir — Séance 7

L'essentiel

DataFrame : une table à colonnes nommées, bâtie sur NumPy ; **Series** : une colonne.

Créer / charger : `pd.DataFrame(...)`, `read_csv`, `to_csv`.

Explorer : `shape`, `info()`, `describe()`, `head()`.

Sélectionner : `df["col"]`, `df[["c1", "c2"]]`, `df.loc[...]`.

Filtrer : masque booléen `df[df["col"] > v]`.

Créer une colonne : à partir des colonnes existantes (vectorisé). **Trier** : `sort_values`.

Manquantes : `isna`, `dropna`, `fillna`. **Grouper** : `value_counts`, `groupby("col").mean()`.

Prochaine séance

Séance 8 — Visualiser et faire le pont

On sait charger et résumer une table. La Séance 8 la **visualise** — histogrammes, nuages de points colorés par espèce — puis fait le **pont vers le Machine Learning** : la table devient une matrice X (les mesures) et un vecteur y (l'espèce à prédire).

À faire d'ici là : refaire les cellules « À vous » du notebook de la séance.

Références

- McKinney, W. — *Python for Data Analysis*, 3e éd., O'Reilly, 2022.
- VanderPlas, J. — *Python Data Science Handbook*, 2e éd., O'Reilly, 2023.
- pandas — *10 minutes to pandas* (documentation), 2024.
- pandas — *Getting started tutorials* (documentation), 2024.

Ressources de la séance

- **Notebook** · [TP 7 — pandas](#) (20 Ko)