Chapitre 3 : Apprentissage avancé d'Excel

Ce chapitre vise à approfondir la maîtrise d'Excel et à montrer comment les mêmes analyses peuvent être réalisées en **Python**, en utilisant des bibliothèques comme **pandas**, **matplotlib** et **openpyxl**.

L'objectif est de passer de l'analyse manuelle dans Excel à l'automatisation et la programmation scientifique.

1. Principes des macros et création d'une macro simple

- **Dans Excel :**

Une **macro** est une série d'actions enregistrées pour automatiser une tâche répétitive.

On peut créer une macro à l'aide de **l'enregistreur de macros** ou avec du **VBA (Visual Basic for Applications)**.

Exemple pratique :

Automatiser le formatage d'un tableau :

```
Sub MiseEnForme()
    Range("A1:D1").Font.Bold = True
    Range("A1:D1").Interior.Color = RGB(200, 200, 255)
    Columns("A:D").AutoFit
End Sub
```

Cette macro met en gras la première ligne, ajoute une couleur d'arrière-plan et ajuste la largeur des colonnes.

- **En Python (équivalent avec pandas + openpyxl) :**

```
import pandas as pd
from openpyxl import load_workbook
from openpyxl.styles import Font, PatternFill

# Charger un fichier Excel
wb = load_workbook("ventes.xlsx")
ws = wb.active

# Mettre en forme la première ligne
for cell in ws[1]:
    cell.font = Font(bold=True)
    cell.fill = PatternFill(start_color="C8C8FF", end_color="C8C8FF", fill_type="solid")
```

```
wb.save("ventes_formaté.xlsx")
```

Ce code fait exactement le même travail qu'une macro VBA, mais en Python.

2. **Tableaux croisés dynamiques (TCD)**

- **Dans Excel :**

Un **tableau croisé dynamique** permet de **résumer et analyser** un grand volume de données. Exemple : à partir d'une base de ventes (Date, Région, Produit, Montant), on veut connaître les ventes totales par produit.

- **En Python :**

```
import pandas as pd

# Charger les données
df = pd.read_excel("ventes.xlsx")

# Créer un tableau croisé dynamique
tcd = pd.pivot_table(df, values='Montant', index='Produit', columns='Région', aggfunc='sum', fill_value=0)
print(tcd)
```

Le résultat est identique à un TCD Excel : résumé automatique des ventes par produit et région.

3. **Histogrammes**

- **Dans Excel :**

Un **histogramme** montre la distribution d'une variable (par exemple, les notes d'un examen). Excel calcule automatiquement les classes et affiche la fréquence.

- **En Python :**

```
import matplotlib.pyplot as plt
import numpy as np

# Exemple de données : notes d'étudiants
notes = np.random.randint(0, 20, 50)

plt.hist(notes, bins=10, edgecolor='black')
plt.title("Distribution des notes")
```

```
plt.xlabel("Notes")
plt.ylabel("Nombre d'étudiants")
plt.show()
```

Python permet une personnalisation plus fine du graphique.

4. **Diagrammes en barres**

- **Dans Excel :**

Un **diagramme en barres** sert à comparer des catégories (ex : ventes par mois).

- **En Python :**

```
import matplotlib.pyplot as plt

mois = ['Jan', 'Fév', 'Mar', 'Avr']
ventes = [1200, 1800, 1600, 2100]

plt.bar(mois, ventes)
plt.title("Ventes mensuelles")
plt.xlabel("Mois")
plt.ylabel("Ventes (€)")
plt.show()
```

En Python, ce graphique peut être généré automatiquement à partir d'un DataFrame.

5. **Diagramme en araignée (ou radar)**

- **Dans Excel :**

Le **diagramme en araignée** permet de comparer plusieurs variables sur un même graphique (souvent utilisé pour l'évaluation de performance).

- **En Python :**

```
import matplotlib.pyplot as plt
import numpy as np

labels = ['Communication', 'Travail d'équipe', 'Créativité', 'Efficacité']
scores = [8, 7, 9, 6]
scores += scores[:1]  # fermer la forme

angles = np.linspace(0, 2 * np.pi, len(labels), endpoint=False).tolist()
angles += angles[:1]
```

```
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})
ax.plot(angles, scores, linewidth=2)
ax.fill(angles, scores, alpha=0.25)
ax.set_thetagrids(np.degrees(angles), labels)
plt.title("Évaluation de compétence")
plt.show()
```

Ce type de graphique est puissant pour visualiser les comparaisons multidimensionnelles.

6. Autres outils avancés d'Excel et Python

Quelques exemples :

- **Fonctions conditionnelles :**
  - Excel : =SI(A1>10; "OK"; "Non")
  - Python :
  - df["Résultat"] = df["Note"].apply(lambda x: "OK" if x > 10 else "Non")
- **Recherche de valeur :**
  - Excel : RECHERCHEV
  - Python : merge() ou loc[] dans pandas.
- **Analyse de scénarios / Solveur :**
  - Excel : outil de simulation.
  - Python : scipy.optimize pour la recherche du minimum/maximum.

7. Exercices Excel (et Python)

1. **Créer un tableau croisé dynamique** pour résumer des ventes par région et par produit.
2. **Créer un histogramme** de notes à partir d'un fichier Excel.
3. **Automatiser une tâche** de mise en forme à l'aide d'une macro ou d'un script Python.
4. **Construire un diagramme radar** pour comparer les compétences de plusieurs employés.
5. **Comparer les résultats entre Excel et Python** (exactitude, rapidité, flexibilité).