Chapitre 1

Apprentissage Avancé d'Excel avec Python

Objectifs pédagogiques (Durée : 2 semaines)

Ce chapitre vous apprendra à automatiser des tâches avancées dans Excel à l'aide de Python. Vous serez capable de :

— Manipuler des fichiers Excel avec Python (en utilisant `openpyxl` et `pandas`).

— Créer des graphiques comme des histogrammes, diagrammes en barres, et diagrammes en araignée avec `matplotlib`.

— Automatiser des tâches répétitives comme l'analyse des données, l'ajout de graphiques, et la création de tableaux croisés dynamiques.

— Utiliser des macros Excel avec Python via la bibliothèque `xlwings`.

1.1 Introduction : qu'est-ce qu'une macro Excel ?

Une **macro Excel** est une suite d'actions automatiques enregistrées ou programmées dans Excel, initialement en langage VBA (Visual Basic for Applications). Elle permet d'automatiser des tâches répétitives :

— formatage automatique,

— insertion de données,

— création de tableaux ou graphiques,

— génération de rapports.

1.1.1 Limites des macros VBA

— langage ancien et limité,

— difficile à maintenir dans des projets professionnels,

— faible compatibilité avec d'autres outils modernes,

— gestion complexe des données volumineuses.

C'est pourquoi Python devient aujourd'hui un **remplaçant naturel** des macros Excel.

1.2 Pourquoi utiliser Python pour automatiser Excel ?

Python permet de :
— manipuler très facilement des données avec `pandas`,

— lire et écrire des fichiers Excel avec `openpyxl`,

— contrôler Excel directement (comme une macro) avec `xlwings`,

— générer des graphiques avancés avec `matplotlib`,

— automatiser des rapports complets,

— traiter des milliers de lignes sans ralentissement.
Python remplace donc VBA grâce à sa puissance, sa simplicité et sa modernité.

1.3 Principes des macros et création d'une macro simple avec Python

Les macros Excel peuvent être automatisées avec **Python** grâce à des bibliothèques comme **‘xlwings‘**. Cette bibliothèque vous permet de contrôler Excel via Python, en créant des scripts qui exécutent des actions dans Excel, comme la création de macros ou la manipulation de données.

1.3.1 Exemple simple : Création d'une macro avec `xlwings`

```
1 import xlwings as xw
2
```

```python
# Ouvrir Excel
wb = xw.Book()

# Ajouter une feuille
ws = wb.sheets[0]
ws.range('A1').value = 'Bonjour Excel !'

# Enregistrer une macro dans Excel
def macro_exemple():
    ws.range('A1').value = 'Cela vient de Python !'

# Exécuter la macro
macro_exemple()
wb.save('mon_fichier_excel.xlsx')
wb.close()
```

Listing 1.1 – Créer une macro simple avec xlwings

Dans cet exemple, la macro modifie le contenu de la cellule A1 dans Excel. Vous pouvez ensuite sauvegarder ce fichier et l'exécuter à tout moment pour répéter l'opération.

1.4 Tableaux croisés dynamiques (TCD) avec Python

Un tableau croisé dynamique (TCD) est un excellent moyen d'analyser des données de manière interactive. En Python, vous pouvez manipuler les données et créer des TCD avec **Pandas**.

1.4.1 Exemple : Création d'un TCD avec `Pandas`

```python
import pandas as pd

# Créer un DataFrame
data = {
    'Produit': ['Pommes', 'Bananes', 'Oranges', 'Pommes', 'Bananes'],
    'Région': ['Nord', 'Sud', 'Nord', 'Sud', 'Nord'],
    'Ventes': [100, 150, 200, 250, 300],
}

df = pd.DataFrame(data)

# Créer un tableau croisé dynamique
tcd = pd.pivot_table(df, values='Ventes', index=['Produit'], columns=['R
    égion'], aggfunc='sum')

print(tcd)
```

Listing 1.2 – Créer un tableau croisé dynamique avec pandas

Dans cet exemple, nous avons un tableau croisé dynamique qui analyse les ventes par produit et par région.

1.5 Création de graphiques avec Python

Les graphiques permettent de visualiser les données de manière plus compréhensible. Avec **Matplotlib**, vous pouvez créer des histogrammes, des diagrammes en barres, et des diagrammes en araignée.

1.5.1 Exemple : Créer un histogramme avec `Matplotlib`

```python
import matplotlib.pyplot as plt

# Données
ventes = [100, 200, 150, 250, 300]
produits = ['Pommes', 'Bananes', 'Oranges', 'Fraises', 'Kiwis']

# Créer un histogramme
plt.bar(produits, ventes, color='blue')
plt.title('Ventes par produit')
plt.xlabel('Produit')
plt.ylabel('Ventes')

# Afficher le graphique
plt.show()
```

Listing 1.3 – Créer un histogramme avec matplotlib

Cet exemple crée un diagramme en barres montrant les ventes par produit.

1.5.2 Exemple : Créer un diagramme en araignée

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

# Données
labels = ['Ventes', 'Profit', 'Coût', 'Satisfaction']
values = [85, 75, 60, 90]

# Nombre de variables
num_vars = len(labels)

# Calculer les angles pour chaque axe
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

# Fermer le graphique
values += values[:1]
angles += angles[:1]

# Créer le graphique
fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))
ax.fill(angles, values, color='blue', alpha=0.25)
ax.plot(angles, values, color='blue', linewidth=2)

# Ajouter les labels
```

```
ax.set_yticklabels([])
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)

plt.title('Performance des produits', size=16)
plt.show()
```

Listing 1.4 – Créer un diagramme en araignée avec matplotlib

Le diagramme en araignée ci-dessus compare plusieurs critères pour différents produits ou départements.

1.6 Exercices pratiques

1.6.1 Exercices pour débutants

1. **Tableau croisé dynamique** : Créez un TCD pour analyser les ventes de produits par mois et par région.

2. **Macro VBA** : Créez une macro qui formate automatiquement un tableau Excel avec des bordures et un fond coloré.

3. **Histogramme** : Créez un histogramme représentant la distribution des notes des étudiants dans une classe.

4. **Diagramme en barres** : Créez un diagramme en barres comparant les performances de plusieurs départements dans une entreprise.

5. **Diagramme en araignée** : Créez un diagramme en araignée pour comparer les résultats de différents projets sur plusieurs critères.

1.7 Conclusion

Ce chapitre vous a permis d'explorer l'intégration de **Python avec Excel**, en automatisant des tâches comme la création de macros, l'analyse de données avec des tableaux croisés dynamiques, et la visualisation de données à l'aide de graphiques.

Vous pouvez maintenant utiliser Python pour automatiser et analyser efficacement vos données dans Excel, et créer des graphiques puissants pour mieux comprendre vos résultats.