Veille technologique : Pourquoi et comment les entreprises remplacent-elles progressivement les macros VBA par des scripts Python dans leurs outils métiers ?

I. Le contexte : automatisation et outils bureautiques en entreprise

Le 30 septembre 1985 sort la première version d'Excel pour les ordinateurs Apple Macintosh. Aujourd'hui, on estime à plus de 1 milliard le nombre d'utilisateurs d'Excel soit un tiers du marché des tableurs. Excel est un outil devenu essentiel dans le traitement des données en entreprise. Ses usages sont illimités, allant de la simple feuille de calcul permettant le suivi des stocks jusqu'à des outils financiers vitaux au bon fonctionnement de l'entreprise. Il est devenu rare de trouver une entreprise qui ne se sert pas d'Excel ou d'un autre tableur dans ses processus et pour cause, ses usages s'adressent à tout les publics, particuliers et professionnels, PME, TPE et même aux multinationales. Aujourd'hui sa présence est telle qu'un championnat du monde regroupant 128 participants a eu lieu en 2023.

Dans les années 1990, Microsoft intègre dans son logiciel Visual Basic for Applications (VBA), un langage de scripts pour standardiser les différentes macro-langues des applications individuelles Microsoft. VBA Excel va permettre d'automatiser les flux de travail réguliers ainsi que de générer des outils pratiques de gestion de projet ou de comptabilité. Bien que l'arrivée de VBA marque une révolution dans l'automatisation du traitement des données des entreprises, ce langage se retrouve petit à petit rattrapé par l'avancée des nouvelles technologies. Avec la présence de plus en plus accrue de l'informatique dans les entreprises, le nombre de données à traiter a explosé. Dans le cadre de petits fichiers, VBA est très optimal et pratique. Cependant, il devient de plus en plus fréquent pour les entreprises de posséder des fichiers de données très conséquents. De plus, l'automatisation de processus sur Excel est devenu courant dans les entreprises mais la recherche constante d'optimisation du traitement des données rend VBA désuet face à d'autres langages de programmation comme Python.

Pour mieux comprendre les limites croissantes de VBA, il faut s'intéresser à son fonctionnement. Excel permet deux utilisations de VBA. La première est l'enregistrement de macros. Par le biais d'un bouton dans l'onglet développeur de l'application, il est possible d'enregistrer nos actions. Une fois l'enregistrement de la macro terminé, il est possible de lire cette macro, les gestes enregistrés précédemment seront alors répétés. Je peux par exemple lancer l'enregistrement d'une macro nommée « Supprimer ». Une fois l'enregistrement lancé, je vais venir supprimer le contenu de trois cellules puis arrêter l'enregistrement. Une fois l'enregistrement coupé, chaque fois que je ferais appel à ma macro, le contenu des trois cellules sera supprimé. Ce premier usage est très pratique mais atteint vite ses limites lorsque l'on veut réaliser des actions en fonction de conditions. C'est dans ce cas là qu'intervient le second usage de VBA. En plus de permettre l'enregistrement de macros, Microsoft intègre dans Excel une interface de programmation. Celle-ci va permettre de coder des macros plus complexes apportant la possibilité de traiter les données de manière plus pointue. Cependant, là aussi les automatisations atteignent vite leurs limites lorsque la quantité de données devient trop importante ou que le traitement est trop conséquent. L'une des dernières grosses faiblesse de VBA est qu'il force l'utilisateur a rester sur le fichier tout le long de l'exécution du programme, l'empêchant d'avancer sur d'autres tâches pendant que le programme s'exécute. Face à ces inconvénients occupant une place de plus en plus importantes, il devient important de s'intéresser à d'autres technologies existantes.

II. Python : une réponse moderne aux enjeux de productivité

Python est un langage de programmation créé en 1989 et possédant de nombreux atouts parmi lesquels on retrouve principalement :

- Langage multiplateforme : Il fonctionne sur de nombreux systèmes d'exploitation tel que Linux, Microsoft, iOS, Android et même les supers supercalculateurs.
- C'est un langage gratuit, il peut être installé sur autant d'ordinateurs qu'on le souhaite et même sur téléphone.
- Il s'agit d'un langage de haut niveau, il demande donc relativement peu de connaissances sur le fonctionnement d'un ordinateur pour être utilisé.
- C'est un langage interprété. Cela signifie qu'un script Python n'a pas besoin d'être complice pour être exécuté contrairement à des langages comme le C ou le C++.
- Il est relativement simple à prendre en main.
- Il est orienté objet. Il est donc possible de concevoir en Python des entités avec un certain nombre de règles de fonctionnement et d'interaction.
- C'est un langage très utilisé en analyse de données ainsi que pour de nombreux d'autres usages.
- C'est le langage de programmation le plus utilisé au monde et possède la plus grande communauté.

Au delà de ses nombreux avantages listés précédemment, l'un des grands points forts de ce langage est sa syntaxe claire, moderne, lisible et proche du langage naturel. Cela facilite son apprentissage, sa maintenance ainsi que la collaboration entre les développeurs. Contrairement à VBA qui nécessite souvent plus de lignes de code pour des tâches simples, Python permet de faire plus avec moins.

Exemple d'addition des valeurs d'une colonne « Montant » dans un fichier Excel :

```
Dim total As Double
total = 0
For Each cell In Range("B2:B100")
    If IsNumeric(cell.Value) Then
        total = total + cell.Value
    End If
Next cell
MsgBox total
```

Automatisation en VBA

```
import pandas as pd
df = pd.read_excel("fichier.xlsx")
total = df["Montant"].sum()
print(total)
```

Automatisation en Python

La grande variété d'usages de Python lui permet de posséder un écosystème de bibliothèques spécialisées très riche. Elle permettent d'automatiser de tâches complexes de manière simple, rapide et fiable. Pour le traitement de données Excel ou Access, il existe plusieurs outils puissants mis à la disposition des développeurs comme :

- Pandas : il s'agit de la bibliothèque de référence pour la manipulation de données tabulaires. Elle permet de lire, filtrer, transformer, analyser et exporter des données avec une grande

efficacité. Elle est souvent utilisée pour remplacer les boucles complexes ou les tableaux croisés dynamiques.
- Openpyxl : spécialisée dans la lecture et l'écriture e fichiers Excel au format .xlsx. Elle permet de modifier les cellules, les formules, les formats ou encore e créer de nouveaux classeurs, sans avoir besoin d'ouvrir Excel.
- Pyodbc : une bibliothèque qui permet de se connecter à des bases e données via ODBC, dont Microsoft Access. Elle facilite l'extraction de données depuis des fichiers .mdb ou .accdb, ce qui est essentiel dans les entreprises qui utilisent encore Access comme base intermédiaire.
- Xlwings : elle permet d'interagir directement avec Excel installé sur l'ordinateur (comme le ferait VBA), tout en utilisant la puissance de Python. Idéal pour les projets hybrides qui doivent manipuler des feuilles existantes avec des macros.

Grâce à ces bibliothèques, Python peut s'adapter à une grande variété de cas d'usage allant du simple tri de données à des automatisations complexes multi-fichiers avec des performances souvent supérieures à VBA.

Python bénéficie d'une grande communauté active à l'échelle mondiale. Cela signifie que les développeurs débutants comme expérimentés peuvent facilement accéder à des ressources pour apprendre, résoudre un problème ou optimiser un script. La documentation officielle de Python ainsi que celles des bibliothèques populaires sont claires, bien structurées et régulièrement mises à jour. En complément, de nombreuses plateformes proposent des tutoriels, des exemples concrets et des réponses à des erreurs fréquentes. Parmi ces plateformes on retrouve notamment :

- Stack Overflow : il s'agit d'une vaste base de questions/réponses très utile pour résoudre des bugs.
- GitHub : il contient des milliers de projets open-source illustrant l'usage de Python dans des contextes professionnels.
- Youtube, Medium, Real Python, OpenClassrooms, DataCamp : qui contiennent des tutoriels complets allant de l'initiation à l'automatisation avancée.

Cette accessibilité permet aux entreprises de réduire les coûts de formation et de favoriser une montée en compétences rapide des équipes. Elle facilite également le partage des bonnes pratiques et la collaboration autour des scripts Python, ce qui est plus difficile avec du code VBA souvent fermé et mal documenté.

Dans un environnement où Excel et Access restent largement utilisés pour le suivi des activités, la gestion des données ou la production de tableaux de bord, Python apporte des réponses concrètes à plusieurs limites rencontrées avec VBA. Tout d'abord, Python permet de traiter de gros volumes de données plus rapidement notamment grâce à la bibliothèque pandas bien plus performante que les boucles classiques en VBA. De plus, Python peut manipuler des fichiers Excel sans avoir besoin d'ouvrir Excel lui-même, ce qui évite les blocages liés à l'interface graphique ou aux plantâmes fréquents de l'application. Concernant Access, la bibliothèque pyodbc permet de lire et interroger directement les bases Access (.mdb ou .accdb) comme s'il s'agissait d'une base SQL classique. Cela rend possible l'extraction automatique de données, leur traitement, puis leur restitution dans un format Excel ou CSV, le tout sans interaction manuelle. Enfin, Python offre la possibilité de centraliser des traitement automatisés : un même script peut parcourir des dossiers contenant des fichiers Excel et Access, en extraire des informations clés, les analyser puis générer un rapport global. Ce type d'automatisation est difficile à maintenir en VBA, surtout quand il faut interagir avec plusieurs fichiers, formats ou structures. En résumé, Python rend les processus plus robustes, évolutifs et adaptés ux besoins d'automatisation modernes, tout en restant compatible avec les outils bureautique encore largement utilisés.

L'un des plus grands atouts de Python dans le traitement de données Excel réside dans sa capacité à lire, modifier et créer des fichiers .xlsx sans nécessiter l'installation de Microsoft Excel sur le poste. Grâce aux bibliothèques abordées précédemment, il est possible de manipuler directement les fichiers Excel, en arrière-plan, depuis 'importe quel environnement compatible avec Python (Windows, Mac, Linux, serveur, etc…). Cela représente un avantage majeur par

rapport à VBA qui dépend entièrement de la suite Office. Avec Python, on peut automatiser des traitements sur des fichiers Excel en ligne de commande, sur un serveur ou même un script planifié sans aucune intervention humaine ni interface graphique. Cette indépendance permet aux entreprises de déployer des automatisations ur des serveurs ou des machines distantes, là où l'installation Microsoft face est soit impossible, soit inutilement coûteuse.

1. **Ouvrir le fichier Excel** contenant les factures.
2. Aller dans l'onglet "Données" et utiliser un **filtre automatique** pour trier les lignes par **nom de client**.
3. Copier les données par client dans un nouveau tableau ou utiliser un **tableau croisé dynamique** pour :
   - grouper les lignes par client,
   - calculer le **total des montants** facturés.
4. Créer une **nouvelle feuille ou un nouveau fichier Excel** et y coller les résultats.
5. Enregistrer ce nouveau fichier sous le nom **"synthèse_facturation.xlsx"**.

Exemple de script en no-code

```python
import pandas as pd

# Étape 1 : lecture du fichier Excel
df = pd.read_excel("factures.xlsx")

# Étape 2 : nettoyage éventuel des données (ex: enlever les lignes vides)
df = df.dropna(subset=["Client", "Montant"])

# Étape 3 : regroupement par client + calcul du total
synthese = df.groupby("Client")["Montant"].sum().reset_index()

# Étape 4 : tri par montant décroissant (optionnel)
synthese = synthese.sort_values(by="Montant", ascending=False)

# Étape 5 : écriture dans un nouveau fichier Excel
synthese.to_excel("synthese_facturation.xlsx", index=False)
```

Script en Python

Microsoft Access est encore largement utilisé dans les PME pour stocker des données métiers sous forme de bases .mdb ou .accdb. Toutefois, l'automatisation avec VBA u les requêtes internes à Access montrent rapidement leurs limites en terme de performance, de maintenance ou d'interopérabilité avec d'autres outils. Grâce à la bibliothèque Python pyodbc, il est possible de se connecter à une base Access via un pilote ODBC (Open Database Connectivity), exactement comme on le ferait avec une base SQL classique. Cette approche permet à Python :

- D'interroger es tables Access avec des requêtes SQL;
- De récupérer les données dans un DataFrame pandas pour les analyser ou les retraiter;
- De mettre à jour ou d'insérer des données dans la base si besoin.

Cette méthode permet donc d'exploiter les données contenues dans Access avec la puissance de Python, en dehors de l'environnement Access lui-même. C'est particulièrement utile pour créer des automatisations robustes ou extraire des données depuis des répertoires contenant des dizaines voir des centaines de fichiers Access.

Contrairement à VBA qui nécessite l'installation de Microsoft Office et s'exécute uniquement dans un contexte bureautique, les scripts Python peuvent être déployés et exécutés en dehors de l'environnement Office, ce qui ouvre la voie à des automatisations plus puissantes, flexibles et centralisées. Un script Python peut par exemple s'exécuter automatiquement chaque nuit sur un serveur pour traiter des fichiers Excel et Access déposés dans un dossier réseau. Il peut aussi être déclenché par une tâche planifiée (cron ou Task Scheduler). On peut par exemple aussi faire fonctionner le script dans le cloud via Azure, AWS ou encore Google Cloud pour automatiser la gestion de fichiers partagés ur un drive synchronisé depuis Microsoft 365. L'automatisation en dehors de l'environnement Office permet de centraliser les scripts sur un seul poste ou serveur et d'éviter les doublons de macros réparties entre plusieurs fichiers. De plus, cela permet une meilleure stabilité car l'environnement d'exécution est isolé et il n'y a donc pas de conflits avec d'autres utilisateurs. Enfin, on a une meilleure facilité de mise en production, de surveillance des logs et de scalabilité (exécution sur plusieurs fichiers et automatisation en masse).

III. Transition : comment les entreprises remplacent VBA par Python

La migration des automatisations Excel ou Access de VBA vers Python ne se fait pas du jour au lendemain. Il s'agit d'un processus progressif souvent mené en parallèle de l'évolution des outils et des besoins métiers. Nous allons aborder les étapes clés et les enjeux associés à cette transition.

---

a. Étapes typiques de migration

1. Analyse des macros existantes
2. Réécriture progressive avec Python
3. Tests, validation et documentation
4. Déploiement dans l'environnement cible

---

b. Outils et environnements utilisés

- Visual Studio Code
- Jupyter Notebook
- Anaconda
- PythonAnywhere ou Heroku

---

c. Difficultés rencontrées

- Résistance au changement
- Formation des équipes
- Compatibilité avec des formats anciens

IV. Études de cas et retours d'expérience

---

a. Exemple 1 : Cabinet d'expertise comptable

Un cabinet comptable utilisait un ensemble de macros VBA pour agréger des données client stockées dans des fichiers Excel répartis dans 4 000 dossiers. Le traitement prenait entre 2 et 4 heures et bloquait l'ordinateur. En migrant le processus en Python avec Pandas et Openpyxl, le

temps d’exécution est passé à 10 minutes sans bloquer la machine, tout en permettant une analyse plus poussée.

b. Exemple 2 : Service logistique d’un groupe industriel

Le service logistique utilisait Access pour gérer les livraisons et Excel pour les rapports. La migration des requêtes Access vers Python + Pandas a permis une automatisation complète des rapports hebdomadaires, intégrés à une interface Streamlit. Résultat : un gain de 5 heures par semaine.

c. Enseignements

- Python améliore les performances et la maintenabilité
- La documentation et la structuration du code sont facilitées
- Les scripts Python sont plus robustes et évolutifs

V. Conclusion

La transition de VBA vers Python n’est pas qu’un effet de mode : elle s’inscrit dans une dynamique plus large de modernisation des outils métiers, portée par des besoins croissants en automatisation, en traitement de gros volumes de données et en interopérabilité avec d'autres systèmes. Python s’impose comme un levier essentiel de transformation numérique, notamment dans les TPE, PME et services comptables.

Pour les développeurs en BTS SIO SLAM, la maîtrise de Python, de ses bibliothèques et de son écosystème représente un atout majeur pour répondre aux enjeux actuels des entreprises. C’est aussi une compétence recherchée dans les environnements DevOps et DataOps de demain, où automatisation et analyse de données vont de pair.

Sources

- Stack Overflow Developer Survey 2023 – https://survey.stackoverflow.co/2023
- Real Python – https://realpython.com
- OpenClassrooms – Formations Python et automatisation Excel
- Towards Data Science (Medium) – Articles sur les automatisations en Python
- Documentation officielle Python – https://docs.python.org
- Pandas – https://pandas.pydata.org
- Openpyxl – https://openpyxl.readthedocs.io
- GitHub – Exemples de projets de migration VBA → Python