Chapitre 1

Programmation et Automatisation

Objectifs pédagogiques (Durée : 4 semaines)

Ce chapitre vous apprendra à automatiser des tâches répétitives et à manipuler des fichiers avec Python. À la fin de ce chapitre, vous serez capable de :

— Comprendre ce qu'est l'automatisation et pourquoi c'est utile

— Manipuler des fichiers et dossiers avec Python

— Lire et écrire des fichiers Excel et CSV

— Utiliser Pandas pour analyser des données simples

— Créer des scripts utiles pour la vie quotidienne

— Développer de petites interfaces graphiques

1.1 Introduction à l'automatisation

1.1.1 Pourquoi automatiser ?

L'automatisation permet de faire faire à l'ordinateur les tâches répétitives et ennuyeuses à notre place. C'est comme avoir un assistant personnel numérique !

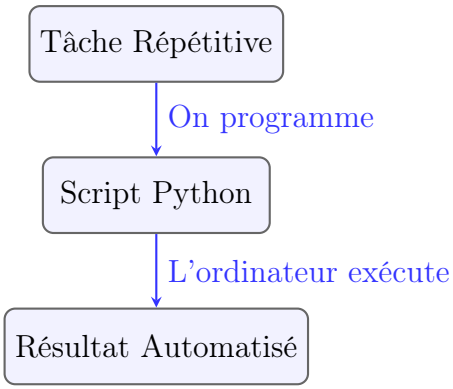

FIGURE 1.1 – L'automatisation nous évite de faire les mêmes tâches encore et encore

1.1.2 Exemples concrets d'automatisation

Voici des situations où l'automatisation est très utile :

— **Renommer 100 photos** : Au lieu de cliquer sur chaque photo, un script peut le faire en 1 seconde

— **Trier des fichiers** : Classer automatiquement les documents PDF, images et musiques dans différents dossiers

— **Envoyer des emails** : Envoyer le même message à plusieurs personnes personnalisé avec leur nom

— **Analyser des données** : Calculer automatiquement des statistiques sur des ventes ou des notes

1.2 Manipulation de fichiers et dossiers

1.2.1 Les bases avec le module `os`

```python
import os

# Créer un dossier
dossier = "mes_documents"
if not os.path.exists(dossier):
    os.mkdir(dossier)
    print(f"Dossier '{dossier}' créé !")

# Vérifier si un fichier existe
fichier = "mon_fichier.txt"
if os.path.exists(fichier):
    print("Le fichier existe")
else:
```

```
    print("Le fichier n'existe pas")

# Lister les fichiers d'un dossier
print("Fichiers dans le dossier courant:")
for element in os.listdir("."):
    if os.path.isfile(element):
        print(f"Fichier: {element}")
    elif os.path.isdir(element):
        print(f"Dossier: {element}")

# Obtenir le chemin absolu
chemin_absolu = os.path.abspath(".")
print(f"Je suis dans le dossier: {chemin_absolu}")
```

Listing 1.1 – Manipulation simple de fichiers et dossiers

1.2.2 Copier et déplacer des fichiers avec `shutil`

```
import shutil
import os

# Créer un fichier de test
with open("fichier_test.txt", "w") as f:
    f.write("Ceci est un fichier de test")

# Copier un fichier
shutil.copy("fichier_test.txt", "fichier_copie.txt")
print("Fichier copié !")

# Copier un dossier entier
if os.path.exists("backup"):
    shutil.rmtree("backup")  # Supprimer s'il existe
shutil.copytree(".", "backup")
print("Dossier backup créé !")

# Déplacer/renommer un fichier
shutil.move("fichier_test.txt", "fichier_renomme.txt")
print("Fichier déplacé/renommé !")

# Supprimer un fichier
os.remove("fichier_copie.txt")
print("Fichier supprimé !")
```

Listing 1.2 – Opérations sur les fichiers

1.2.3 Exercice pratique : Organisateur de fichiers

```
import os
import shutil

def organiser_fichiers(dossier="."):
    """Organise les fichiers par type dans des dossiers"""

```

```
    # Dossiers à créer
    categories = {
        'Images': ['.jpg', '.jpeg', '.png', '.gif', '.bmp'],
        'Documents': ['.pdf', '.docx', '.txt', '.xlsx'],
        'Musique': ['.mp3', '.wav', '.aac'],
        'Videos': ['.mp4', '.avi', '.mov'],
        'Scripts': ['.py', '.js', '.html', '.css']
    }

    # Créer les dossiers s'ils n'existent pas
    for categorie in categories:
        if not os.path.exists(categorie):
            os.mkdir(categorie)
            print(f"Dossier '{categorie}' créé")

    # Parcourir tous les fichiers
    for fichier in os.listdir(dossier):
        if os.path.isfile(fichier):
            # Obtenir l'extension du fichier
            _, extension = os.path.splitext(fichier)
            extension = extension.lower()

            # Trouver la catégorie
            deplace = False
            for categorie, extensions in categories.items():
                if extension in extensions:
                    shutil.move(fichier, os.path.join(categorie, fichier
))
                    print(f"Déplacé: {fichier} -> {categorie}/")
                    deplace = True
                    break

            if not deplace:
                print(f"Non déplacé: {fichier}")

# Utilisation
if __name__ == "__main__":
    print("Début de l'organisation...")
    organiser_fichiers()
    print("Organisation terminée !")
```

Listing 1.3 – Script pour organiser automatiquement les fichiers

1.3 Travail avec les fichiers de données

1.3.1 Lire et écrire des fichiers CSV

```
import csv

# Écrire dans un fichier CSV
with open('eleves.csv', 'w', newline='', encoding='utf-8') as fichier:
    writer = csv.writer(fichier)
    # Écrire l'en-tête
    writer.writerow(['Nom', 'Age', 'Note'])
```

```
    # Écrire les données
    writer.writerow(['Alice', 18, 15])
    writer.writerow(['Bob', 17, 12])
    writer.writerow(['Charlie', 19, 16])

print("Fichier CSV créé !")

# Lire un fichier CSV
with open('eleves.csv', 'r', encoding='utf-8') as fichier:
    reader = csv.reader(fichier)
    for ligne in reader:
        print(ligne)

# Méthode plus simple avec pandas
import pandas as pd

# Lire le CSV
df = pd.read_csv('eleves.csv')
print("Données du CSV:")
print(df)

# Afficher des informations
print(f"\nNombre d'élèves: {len(df)}")
print(f"Note moyenne: {df['Note'].mean():.2f}")
print(f"Meilleure note: {df['Note'].max()}")
```

Listing 1.4 – Manipulation des fichiers CSV

1.3.2 Travailler avec Excel

```
import pandas as pd

# Créer un DataFrame (tableau de données)
donnees = {
    'Produit': ['Pommes', 'Bananes', 'Oranges', 'Fraises'],
    'Prix': [2.50, 1.80, 3.20, 4.50],
    'Quantite': [100, 150, 80, 60],
    'Vendu': [85, 120, 65, 55]
}

df = pd.DataFrame(donnees)

# Ajouter une colonne calculée
df['Chiffre_affaires'] = df['Prix'] * df['Vendu']
df['Reste'] = df['Quantite'] - df['Vendu']

print("Tableau des ventes:")
print(df)

# Sauvegarder en Excel
df.to_excel('ventes_fruits.xlsx', index=False)
print("Fichier Excel créé !")

# Lire un fichier Excel
df_lu = pd.read_excel('ventes_fruits.xlsx')
print("\nDonnées lues depuis Excel:")
```

```
print(df_lu)

# Faire des calculs simples
total_ventes = df_lu['Chiffre_affaires'].sum()
print(f"\nChiffre d'affaires total: {total_ventes:.2f}    ")

# Trier par chiffre d'affaires
df_trie = df_lu.sort_values('Chiffre_affaires', ascending=False)
print("\nProduits triés par CA:")
print(df_trie[['Produit', 'Chiffre_affaires']])
```

Listing 1.5 – Manipulation simple de fichiers Excel

1.4 Création de scripts utiles

1.4.1 Script : Renommeur de photos

```
import os
import glob

def renommer_photos(dossier=".", prefixe="vacances_"):
    """Renomme toutes les photos d'un dossier"""

    # Types de fichiers image
    extensions = ['*.jpg', '*.jpeg', '*.png', '*.gif']

    photos = []
    for ext in extensions:
        photos.extend(glob.glob(os.path.join(dossier, ext)))

    print(f"Found {len(photos)} photos to rename")

    # Renommer chaque photo
    for i, ancien_nom in enumerate(photos, 1):
        # Obtenir l'extension
        _, extension = os.path.splitext(ancien_nom)

        # Nouveau nom
        nouveau_nom = f"{prefixe}{i:03d}{extension}"
        nouveau_chemin = os.path.join(dossier, nouveau_nom)

        # Renommer
        os.rename(ancien_nom, nouveau_chemin)
        print(f"Renommé: {os.path.basename(ancien_nom)} -> {nouveau_nom}
    ")

    print("Renommage terminé !")

# Utilisation
if __name__ == "__main__":
    renommer_photos(".", "mes_photos_")
```

Listing 1.6 – Script pour renommer des photos en lot

1.4.2 Script : Calculateur de moyennes

```python
def calculer_moyennes():
    """Calcule les moyennes des élèves"""

    eleves = []

    print("=== Calculateur de Moyennes ===")
    print("Entrez les notes des élèves (tapez 'fin' pour terminer)")

    while True:
        nom = input("\nNom de l'élève (ou 'fin'): ")
        if nom.lower() == 'fin':
            break

        try:
            note1 = float(input("Note 1: "))
            note2 = float(input("Note 2: "))
            note3 = float(input("Note 3: "))

            moyenne = (note1 + note2 + note3) / 3
            eleves.append({
                'nom': nom,
                'notes': [note1, note2, note3],
                'moyenne': moyenne
            })

            print(f"Moyenne de {nom}: {moyenne:.2f}")

        except ValueError:
            print("Erreur: Entrez des nombres valides")

    # Afficher le classement
    if eleves:
        print("\n" + "="*40)
        print("CLASSEMENT FINAL")
        print("="*40)

        # Trier par moyenne (meilleure d'abord)
        eleves_tries = sorted(eleves, key=lambda x: x['moyenne'],
    reverse=True)

        for i, eleve in enumerate(eleves_tries, 1):
            print(f"{i}. {eleve['nom']}: {eleve['moyenne']:.2f}")

        # Statistiques
        moyennes = [e['moyenne'] for e in eleves]
        print(f"\nMoyenne de la classe: {sum(moyennes)/len(moyennes):.2f
    }")
        print(f"Meilleure moyenne: {max(moyennes):.2f}")
        print(f"Moins bonne moyenne: {min(moyennes):.2f}")

# Lancer le programme
if __name__ == "__main__":
    calculer_moyennes()
```

Listing 1.7 – Script pour calculer des moyennes scolaires

1.5 Introduction aux interfaces graphiques

1.5.1 Créer une fenêtre simple avec Tkinter

```python
import tkinter as tk
from tkinter import messagebox

def creer_interface_simple():
    """Crée une interface graphique simple"""

    # Créer la fenêtre principale
    fenetre = tk.Tk()
    fenetre.title("Ma Première Application")
    fenetre.geometry("300x200")

    # Ajouter un titre
    titre = tk.Label(fenetre, text="Bienvenue dans mon application!",
                     font=("Arial", 14))
    titre.pack(pady=20)

    # Champ de saisie
    label_nom = tk.Label(fenetre, text="Entrez votre nom:")
    label_nom.pack()

    champ_nom = tk.Entry(fenetre, width=30)
    champ_nom.pack(pady=5)

    def dire_bonjour():
        """Fonction appelée quand on clique sur le bouton"""
        nom = champ_nom.get()
        if nom:
            messagebox.showinfo("Bonjour", f"Bonjour {nom} !")
        else:
            messagebox.showwarning("Attention", "Veuillez entrer votre
    nom")

    # Bouton
    bouton = tk.Button(fenetre, text="Dire bonjour",
                       command=dire_bonjour, bg="lightblue")
    bouton.pack(pady=10)

    # Bouton pour quitter
    bouton_quitter = tk.Button(fenetre, text="Quitter",
                               command=fenetre.quit, bg="lightcoral")
    bouton_quitter.pack(pady=5)

    # Lancer l'application
    fenetre.mainloop()

if __name__ == "__main__":
    creer_interface_simple()
```

Listing 1.8 – Première interface graphique

1.5.2 Application : Convertisseur de températures

```
import tkinter as tk
from tkinter import ttk

class ConvertisseurTemperature:
    def __init__(self):
        self.fenetre = tk.Tk()
        self.fenetre.title("Convertisseur de Température")
        self.fenetre.geometry("400x300")

        self.creer_interface()

    def creer_interface(self):
        """Crée l'interface du convertisseur"""

        # Titre
        titre = tk.Label(self.fenetre, text="Convertisseur de Tempé
rature",
                         font=("Arial", 16, "bold"))
        titre.pack(pady=20)

        # Frame pour la saisie
        frame_saisie = tk.Frame(self.fenetre)
        frame_saisie.pack(pady=10)

        # Champ de température
        tk.Label(frame_saisie, text="Température:").grid(row=0, column
=0, padx=5)
        self.champ_temp = tk.Entry(frame_saisie, width=10)
        self.champ_temp.grid(row=0, column=1, padx=5)

        # Menu déroulant pour l'unité source
        tk.Label(frame_saisie, text="De:").grid(row=0, column=2, padx=5)
        self.unite_source = ttk.Combobox(frame_saisie,
                                         values=["Celsius", "Fahrenheit",
"Kelvin"],
                                         state="readonly")
        self.unite_source.set("Celsius")
        self.unite_source.grid(row=0, column=3, padx=5)

        # Menu déroulant pour l'unité cible
        tk.Label(frame_saisie, text="Vers:").grid(row=1, column=2, padx
=5)
        self.unite_cible = ttk.Combobox(frame_saisie,
                                        values=["Celsius", "Fahrenheit", "
Kelvin"],
                                        state="readonly")
        self.unite_cible.set("Fahrenheit")
        self.unite_cible.grid(row=1, column=3, padx=5)

        # Bouton de conversion
        bouton_convertir = tk.Button(self.fenetre, text="Convertir",
                                     command=self.convertir, bg="
lightgreen")
        bouton_convertir.pack(pady=10)

        # Zone de résultat
```

```
        self.label_resultat = tk.Label(self.fenetre, text="Résultat: ",
                                       font=("Arial", 12), fg="blue")
        self.label_resultat.pack(pady=20)

        # Bouton pour effacer
        bouton_effacer = tk.Button(self.fenetre, text="Effacer",
                                   command=self.effacer, bg="lightyellow")
        bouton_effacer.pack(pady=5)

    def convertir(self):
        """Effectue la conversion"""
        try:
            temperature = float(self.champ_temp.get())
            source = self.unite_source.get()
            cible = self.unite_cible.get()

            # Conversion en Celsius d'abord
            if source == "Celsius":
                celsius = temperature
            elif source == "Fahrenheit":
                celsius = (temperature - 32) * 5/9
            elif source == "Kelvin":
                celsius = temperature - 273.15

            # Conversion de Celsius vers l'unité cible
            if cible == "Celsius":
                resultat = celsius
            elif cible == "Fahrenheit":
                resultat = (celsius * 9/5) + 32
            elif cible == "Kelvin":
                resultat = celsius + 273.15

            self.label_resultat.config(
                text=f"Résultat: {resultat:.2f}   {cible[0]}"
            )

        except ValueError:
            self.label_resultat.config(text="Erreur: Entrez un nombre
    valide")

    def effacer(self):
        """Efface les champs"""
        self.champ_temp.delete(0, tk.END)
        self.label_resultat.config(text="Résultat: ")

    def lancer(self):
        """Lance l'application"""
        self.fenetre.mainloop()

# Utilisation
if __name__ == "__main__":
    app = ConvertisseurTemperature()
    app.lancer()
```

Listing 1.9 – Convertisseur avec interface graphique

1.6 Exercices pratiques

1.6.1 Exercices pour débutants

1. **Organisateur de fichiers amélioré** : Améliore le script d'organisation pour qu'il crée aussi des sous-dossiers par année pour les photos.

2. **Calculateur de budget** : Crée un script qui aide à gérer un budget mensuel. Il doit pouvoir ajouter des dépenses et des revenus, puis afficher le solde.

3. **Liste de courses** : Développe une application qui permet de gérer une liste de courses. On doit pouvoir ajouter, supprimer et cocher des articles.

4. **Générateur de mots de passe** : Crée un programme qui génère des mots de passe sécurisés avec une longueur choisie par l'utilisateur.

5. **Convertisseur d'unités** : Étends le convertisseur de température pour qu'il puisse aussi convertir des longueurs (mètres, pieds, pouces) et des poids (kg, livres).

1.6.2 Projet final : Gestionnaire de contacts

```python
import json
import os

class GestionnaireContacts:
    def __init__(self, fichier="contacts.json"):
        self.fichier = fichier
        self.contacts = self.charger_contacts()

    def charger_contacts(self):
        """Charge les contacts depuis le fichier"""
        if os.path.exists(self.fichier):
            with open(self.fichier, 'r', encoding='utf-8') as f:
                return json.load(f)
        return []

    def sauvegarder_contacts(self):
        """Sauvegarde les contacts dans le fichier"""
        with open(self.fichier, 'w', encoding='utf-8') as f:
            json.dump(self.contacts, f, indent=2, ensure_ascii=False)

    def ajouter_contact(self):
        """Ajoute un nouveau contact"""
        print("\n--- Ajouter un contact ---")
        nom = input("Nom: ")
        prenom = input("Prénom: ")
        telephone = input("Téléphone: ")
        email = input("Email: ")

        contact = {
```

```
            'nom': nom,
            'prenom': prenom,
            'telephone': telephone,
            'email': email
        }

        self.contacts.append(contact)
        self.sauvegarder_contacts()
        print("Contact ajouté avec succès !")

    def lister_contacts(self):
        """Affiche tous les contacts"""
        print("\n--- Liste des contacts ---")
        if not self.contacts:
            print("Aucun contact enregistré")
            return

        for i, contact in enumerate(self.contacts, 1):
            print(f"{i}. {contact['prenom']} {contact['nom']}")
            print(f"   Téléphone: {contact['telephone']}")
            print(f"   Email: {contact['email']}")
            print()

    def rechercher_contact(self):
        """Recherche un contact"""
        print("\n--- Rechercher un contact ---")
        recherche = input("Entrez un nom ou prénom: ").lower()

        resultats = []
        for contact in self.contacts:
            if (recherche in contact['nom'].lower() or
                recherche in contact['prenom'].lower()):
                resultats.append(contact)

        if resultats:
            print(f"\n{len(resultats)} contact(s) trouvé(s):")
            for contact in resultats:
                print(f"- {contact['prenom']} {contact['nom']}")
                print(f"  Téléphone: {contact['telephone']}")
                print(f"  Email: {contact['email']}")
        else:
            print("Aucun contact trouvé")

    def menu_principal(self):
        """Affiche le menu principal"""
        while True:
            print("\n" + "="*40)
            print("GESTIONNAIRE DE CONTACTS")
            print("="*40)
            print("1. Lister les contacts")
            print("2. Ajouter un contact")
            print("3. Rechercher un contact")
            print("4. Quitter")

            choix = input("\nVotre choix (1-4): ")

            if choix == "1":
                self.lister_contacts()
```

```
            elif choix == "2":
                self.ajouter_contact()
            elif choix == "3":
                self.rechercher_contact()
            elif choix == "4":
                print("Au revoir !")
                break
            else:
                print("Choix invalide. Veuillez choisir entre 1 et 4.")

# Lancement du programme
if __name__ == "__main__":
    gestionnaire = GestionnaireContacts()
    gestionnaire.menu_principal()
```

Listing 1.10 – Projet complet - Gestionnaire de contacts

1.7 Conclusion et prochaines étapes

1.7.1 Ce que vous avez appris

Félicitations ! Vous savez maintenant :

— **Automatiser des tâches** : Faire travailler l'ordinateur à votre place

— **Manipuler des fichiers** : Créer, copier, déplacer et organiser des fichiers

— **Travailler avec des données** : Lire et écrire des fichiers CSV et Excel

— **Créer des interfaces** : Faire des programmes avec des fenêtres et des boutons

— **Développer des scripts utiles** : Résoudre des problèmes concrets

1.7.2 Conseils pour continuer

— **Pratiquez régulièrement** : Essayez d'automatiser une petite tâche chaque jour

— **Commencez simple** : Ne cherchez pas à tout automatiser d'un coup

— **Utilisez ce que vous créez** : Les meilleurs programmes sont ceux qu'on utilise soi-même

— **Partagez vos créations** : Montrez vos scripts à des amis ou collègues

1.7.3 Prochaines aventures

Dans les prochains chapitres, vous découvrirez :

— Comment créer des programmes encore plus intelligents

— Comment connecter vos programmes à internet

— Comment analyser des données plus complexes

— Comment créer des sites web avec Python

1.7.4 Petit quiz

1. Quelle commande permet de créer un dossier en Python ?

2. Comment lire un fichier CSV avec pandas ?

3. Quelle est la différence entre `shutil.copy()` et `shutil.move()` ?

4. Comment créer un bouton dans Tkinter ?

5. Pourquoi est-il important de sauvegarder souvent vos données ?

```python
# 1. Créer un dossier
import os
os.mkdir("mon_dossier")

# 2. Lire un CSV avec pandas
import pandas as pd
donnees = pd.read_csv("fichier.csv")

# 3. Différence copy/move
# copy: fait une copie, l'original reste
# move: déplace, l'original n'existe plus à l'emplacement initial

# 4. Créer un bouton Tkinter
import tkinter as tk
bouton = tk.Button(fenetre, text="Mon Bouton")

# 5. Sauvegarder souvent
# Pour ne pas perdre son travail en cas de problème !
```

Listing 1.11 – Réponses du quiz

Vous êtes maintenant prêt à automatiser votre vie avec Python !