TP N°2 : Automatisation de l'envoi automatique de rapports par email avec Python

Dr. Mohammed Nadjib HAMLAOUI

Filière Hydraulique
4ème année Hydraulique
Université Abdelhafid Boussouf -Mila-

Objectif du TP

L'objectif de ce travail pratique est de mettre en œuvre un mini-projet complet d'automatisation en Python. L'étudiant devra développer un script capable de :

1. Charger des données expérimentales depuis un fichier,
2. Réaliser des statistiques simples (moyenne, écart-type),
3. Générer automatiquement un graphique,
4. Envoyer par email un rapport contenant les résultats et le graphique.

1. Cahier des charges du mini-projet

1.1 Contexte

Une équipe de recherche enregistre quotidiennement des mesures issues d'expériences (température, tension, vitesse du vent, etc.). Afin d'automatiser la génération et la diffusion de rapports, on souhaite développer un script Python réalisant automatiquement le traitement des données et l'envoi d'un rapport.

1.2 Objectifs fonctionnels

Le programme doit :

- Lire un fichier CSV contenant les mesures expérimentales ;
- Calculer la moyenne et l'écart-type des valeurs mesurées ;
- Générer un graphique synthétique (courbe ou histogramme) ;
- Rédiger un rapport textuel contenant les résultats ;
- Envoyer automatiquement ce rapport (texte + image) par email.

1.3 Objectifs techniques

Le script devra :

- Être écrit en Python 3,
- Utiliser les bibliothèques `pandas`, `matplotlib`, `smtplib` et `email`,
- Être structuré en modules ou fonctions claires,
- Permettre de modifier facilement le fichier source ou le destinataire.

1.4 Livrables

- Le script Python final (`tp2_rapport_auto.py`),
- Le fichier de données (`mesures.csv`),
- Un exemple du rapport reçu par email,
- Un fichier `README.txt` expliquant l'utilisation.

2. Réalisation pratique

2.1 Étape 1 : Chargement des données

Le fichier `mesures.csv` contient, par exemple :

```
date,temperature
2025-11-01,22.5
2025-11-02,23.1
2025-11-03,21.9
2025-11-04,22.7
```

Lecture du fichier en Python :

```
import pandas as pd

data = pd.read_csv("mesures.csv")
print(data.head())
```

2.2 Étape 2 : Calculs statistiques

```
moyenne = data['temperature'].mean()
ecart_type = data['temperature'].std()

print(f"Moyenne = {moyenne:.2f} °C")
print(f"Ecart-type = {ecart_type:.2f} °C")
```

Interprétation :

- Un écart-type faible signifie que les valeurs sont stables.
- Un écart-type élevé traduit une grande variabilité.

2.3 Étape 3 : Génération du graphique

```
import matplotlib.pyplot as plt

plt.plot(data['date'], data['temperature'], marker='o')
plt.title("Évolution de la température")
plt.xlabel("Date")
```

```
plt.ylabel("Température (°C)")
plt.grid(True)
plt.savefig("rapport_graphique.png")
plt.close()
```

2.4 Étape 4 : Rédaction d'un rapport automatique

```
rapport = f"""
Rapport automatique des mesures

Moyenne : {moyenne:.2f} °C
Ecart-type : {ecart_type:.2f} °C

Interprétation :
Les mesures sont relativement {"stables" if ecart_type < 1 else "variables"}.

Voir le graphique joint pour plus de détails.
"""

with open("rapport.txt", "w") as f:
    f.write(rapport)
```

2.5 Étape 5 : Envoi automatique par email

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders

expediteur = "votre_adresse@gmail.com"
destinataire = "destinataire@gmail.com"
mot_de_passe = "mot_de_passe_app"

msg = MIMEMultipart()
msg['From'] = expediteur
msg['To'] = destinataire
msg['Subject'] = "Rapport automatique des mesures"
```

```
msg.attach(MIMEText(rapport, 'plain'))

with open("rapport_graphique.png", "rb") as f:
    piece = MIMEBase('application', 'octet-stream')
    piece.set_payload(f.read())
encoders.encode_base64(piece)
piece.add_header('Content-Disposition',
                 'attachment; filename="rapport_graphique.png"')
msg.attach(piece)

with smtplib.SMTP('smtp.gmail.com', 587) as serveur:
    serveur.starttls()
    serveur.login(expediteur, mot_de_passe)
    serveur.send_message(msg)

print("Email envoyé avec succès !")
```

Explication des lignes de code Python

Les lignes suivantes permettent de construire et d'envoyer un email automatique en Python :

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email import encoders
```

Nous allons expliquer chacune d'elles en détail.

1. `import smtplib`

Cette instruction importe le module `smtplib`, qui fait partie de la bibliothèque standard de Python. Ce module permet de se connecter à un serveur SMTP (par exemple Gmail ou Outlook) afin d'envoyer un courriel.

- **Rôle :** gérer la connexion au serveur d'envoi d'emails.

- **Exemple d'utilisation :**

```
server = smtplib.SMTP("smtp.gmail.com", 587)
```

2. `from email.mime.text import MIMEText`

Cette ligne importe la classe `MIMEText` (MIME = "Multipurpose Internet Mail Extensions"), utilisée pour créer le contenu textuel d'un email. Elle permet d'insérer un texte simple ou du HTML dans le message.

- **Rôle :** définir le corps du message.
- **Exemple :**

```
corps = MIMEText("Bonjour, voici votre rapport.", "plain")
```

3. `from email.mime.multipart import MIMEMultipart`

Cette instruction importe la classe `MIMEMultipart`. Elle est nécessaire lorsque l'email comporte plusieurs éléments : texte, pièces jointes, images, etc.

- **Rôle :** créer un email composé de plusieurs parties.
- **Exemple :**

```
message = MIMEMultipart()
message["Subject"] = "Rapport Automatique"
```

4. `from email.mime.base import MIMEBase`

Cette ligne importe la classe `MIMEBase`, utilisée pour préparer une pièce jointe brute (comme un PDF, un fichier Excel ou une image) afin qu'elle puisse être envoyée dans un email.

- **Rôle :** charger et traiter les fichiers à joindre.
- **Exemple :**

```
piece = MIMEBase("application", "octet-stream")
piece.set_payload(open("rapport.pdf", "rb").read())
```

5. `from email import encoders`

Cette instruction importe les outils d'encodage nécessaires pour convertir les pièces jointes en un format compatible avec les emails (base64).

- **Rôle :** encoder les fichiers avant l'envoi.
- **Exemple :**

```
encoders.encode_base64(piece)
```

Résumé

Ligne	Fonction
`smtplib`	Connexion au serveur SMTP et envoi du mail
`MIMEText`	Corps du message (texte ou HTML)
`MIMEMultipart`	Email contenant plusieurs éléments (texte + pièces jointes)
`MIMEBase`	Gestion des fichiers joints
`encoders`	Encodage des pièces jointes (base64)

Explication

- `MIMEMultipart()` permet d'attacher texte + fichiers.
- Les champs `From`, `To` et `Subject` configurent le message.
- Le fichier est ouvert en mode binaire.
- `MIMEBase` permet de créer une pièce jointe générique.
- `encode_base64` prépare le fichier pour l'envoi.
- `Content-Disposition` définit le nom de la pièce jointe.
- `msg.attach(piece)` l'ajoute à l'email.