📚 Cours : Automatisation des Tâches avec Python et Bibliothèques Utilitaires

🎯 Objectif du cours :

Apprendre à automatiser des tâches répétitives avec Python en utilisant des bibliothèques puissantes comme `os`, `shutil`, `schedule`, `subprocess`, `datetime`, et d'autres utilitaires pour améliorer la productivité.

---

1. 🔁 Pourquoi automatiser avec Python ?

L'automatisation permet de :

- Gagner du temps sur des tâches répétitives.
- Réduire les erreurs humaines.
- Planifier et exécuter des scripts à des moments précis.
- Interagir automatiquement avec les fichiers, dossiers, serveurs, APIs, navigateurs, etc.

Python est idéal pour cela grâce à sa simplicité, sa large communauté et ses nombreuses bibliothèques.

---

2. 📂 Automatisation de la gestion de fichiers et de dossiers

a. Manipulation de fichiers avec `os` et `shutil`

```
import os
import shutil

# Créer un dossier
os.mkdir("backup")

# Lister les fichiers
fichiers = os.listdir(".")
print(fichiers)

# Copier un fichier
shutil.copy("data.txt", "backup/data_backup.txt")

# Supprimer un fichier
os.remove("ancien.txt")
```

```
# Déplacer un fichier
shutil.move("rapport.txt", "backup/rapport.txt")
```

b. Créer des scripts de sauvegarde automatique

```
def backup_fichier(source, destination):
    shutil.copy(source, destination)
    print(f"Fichier sauvegardé de {source} vers {destination}")
```

---

3. □ Planification des tâches avec `schedule` et `time`

a. Utilisation de `schedule` pour exécuter une tâche régulièrement

```
import schedule
import time

def dire_bonjour():
    print("Bonjour ! Je travaille automatiquement.")

schedule.every(10).seconds.do(dire_bonjour)

while True:
    schedule.run_pending()
    time.sleep(1)
```

b. Planification journalière

```
schedule.every().day.at("09:00").do(dire_bonjour)
```

---

4. ⚙ Automatiser des commandes système avec `subprocess`

a. Exécuter des commandes shell

```
import subprocess

# Afficher le contenu d'un répertoire
result = subprocess.run(["ls", "-l"], capture_output=True, text=True)
print(result.stdout)
```

b. Lancer un autre script Python automatiquement

```
subprocess.run(["python", "mon_script.py"])
```

---

5. 🗓 Travailler avec les dates et heures

a. Utiliser `datetime`

```
from datetime import datetime

# Obtenir la date et heure actuelle
now = datetime.now()
print("Date actuelle :", now)

# Formatter une date
print(now.strftime("%d/%m/%Y %H:%M:%S"))
```

b. Générer des noms de fichiers avec timestamp

```
fichier = f"rapport_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt"
```

---

6. 📊 Automatiser la génération de rapports

a. Créer un fichier log

```
with open("rapport.txt", "w") as f:
    f.write("Rapport généré automatiquement.\n")
    f.write("Statut : OK\n")
```

b. Générer des fichiers CSV automatiquement

```
import csv

with open("utilisateurs.csv", "w", newline="") as fichier:
    writer = csv.writer(fichier)
    writer.writerow(["Nom", "Email"])
    writer.writerow(["Alice", "alice@example.com"])
    writer.writerow(["Bob", "bob@example.com"])
```

---

7. 🌐 Automatiser des tâches sur le web

a. Télécharger une page web

```
import requests

response = requests.get("https://example.com")
print(response.text)
```

b. Automatiser un navigateur avec `selenium` (exemple basique)

```
from selenium import webdriver

driver = webdriver.Chrome()
```

```
driver.get("https://www.google.com")
```

---

8. 🔄 Exemple de script complet d'automatisation

```
import os, shutil, schedule, time
from datetime import datetime

def sauvegarder_fichier():
    nom_fichier = f"sauvegarde_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt"
    shutil.copy("donnees.txt", nom_fichier)
    print(f"Sauvegarde créée : {nom_fichier}")

schedule.every(1).hour.do(sauvegarder_fichier)

while True:
    schedule.run_pending()
    time.sleep(1)
```

📚 Résumé

Tâche automatisée	Bibliothèque utilisée
Gestion de fichiers	`os`, `shutil`
Exécution planifiée	`schedule`, `time`
Commandes système	`subprocess`
Dates et heures	`datetime`
Génération de fichiers	`csv`, `open()`
Web (avancé)	`requests`, `selenium`

✅ Bonnes pratiques :

- Tester vos scripts manuellement avant automatisation.
- Toujours inclure des logs ou messages pour savoir ce qui se passe.
- Ajouter des vérifications d'erreurs (`try/except`).
- Utiliser des chemins absolus si possible.