La gestion des fichiers

Objectifs :

- Lire, écrire et manipuler des fichiers.
- Créer et organiser des dossiers.
- Automatiser des tâches courantes liées aux fichiers.

•Applications :

- Sauvegarde de données.
- Organisation des fichiers dans des systèmes automatisés.
- Traitement de grandes quantités de données textuelles.

Écriture dans un fichier texte

- Syntaxe de base pour écrire dans un fichier.

```
with open("fichier.txt", "w") as file:
    file.write("Bonjour, ceci est une ligne de texte.")
```

Si le fichier n’existe pas , il sera créé

"**w**" : mode écriture
"**r**" : mode lecture
"**a**" : mode ajout

Ecriture sur plusieurs lignes

- Permettez à l'utilisateur de saisir plusieurs lignes
- Quittez le programme lorsqu'il tape q.

```
while True:
    ligne = input("Saisir un texte : ")

    if ligne.lower() == 'q':
        break  # On sort de la boucle while

    with open("nom.txt", "a") as fichier:
        fichier.write(ligne + "\n")  # On ajoute un saut de ligne à chaque ligne écrite

print("Écriture terminée.")
```

Lecture ligne par ligne

- Suppression des Espaces Blancs : **strip()**

```
with open("nom.txt", "r") as file:
    mot_cle = "python"
    for ligne in file:
        if mot_cle in ligne:
            print(ligne.strip())
```

Lecture de fichiers avec readlines()

- lit toutes les lignes d'un fichier et les retourne sous forme de **liste** de string.

```
with open("example.txt", "r") as file:
    lines = file.readlines()

print(lines)
```

- Chaque ligne inclut généralement le caractère de saut de ligne \n
- Idéal pour charger un fichier entier si sa taille est raisonnable
- Combinez avec **strip()** pour supprimer les espaces inutiles ou les sauts de ligne

Création d’un dossier

- Il faut importer le module os.

```
import os
os.mkdir("MonDossier")
```

- Pour éviter les erreurs si le dossier existe déjà

```
try:
    os.mkdir("MonDossier")
    print("Le dossier 'MonDossier' a été créé avec succès.")
except FileExistsError:
    print("Le dossier 'MonDossier' existe déjà.")
except OSError as e:
    print(f"Erreur lors de la création du dossier : {e}")
```

Récupération du chemin actuel

- **os.getcwd()** permet d'obtenir le chemin du répertoire de travail actuel.

```python
import os

# Récupère le répertoire courant
chemin_actuel = os.getcwd()
print(f"Répertoire actuel : {chemin_actuel}")
```

Applications :

•Vérifier où le programme s'exécute.
•Construire des chemins relatifs pour accéder à des fichiers et dossiers.

Construction de chemins

- **os.path.join** permet de construire des chemins indépendants de l'OS

```python
import os

# Combiner des morceaux de chemin
chemin = os.path.join("Documents", "Projets", "test.txt")
print(f"Chemin généré : {chemin}")
```

```
Chemin généré : Documents/Projets/test.txt
```

Pourquoi utiliser os.path.join ?

- Évite les erreurs dues à des séparateurs incorrects (/ ou \).

- Rend le code portable entre différentes plateformes.

Création de dossier et de sous dossier

- Vérifier si un dossier existe déjà à l'emplacement spécifié
- Utiliser **makedirs()** pour créer une arborescence

```
chemin_courant = os.getcwd()
maListe = ["python","javascript","C++"]

for element in maListe:
    chemin_dossier = os.path.join(chemin_courant,"dossier",element)
    if not os.path.exists(chemin_dossier):
        os.makedirs(chemin_dossier)
        print("Création du dossier")
    else:
        print("Dossier existe déja")
```

Vérification avant création de dossier

- Pour éviter les erreurs vérifier si le dossier n’existe pas avant sa création

```
if not os.path.exists("MonDossier"):
    os.mkdir("MonDossier")
    print("Dossier créé.")
else:
    print("Dossier existe déjà")
```

Vérification avant création de fichier

- Pour éviter les erreurs vérifier si le fichier n’existe pas avant sa création

```
if not os.path.exists("mon_fichier.txt"):

    with open("mon_fichier.txt", 'w') as fichier:
        pass
    print("Fichier créé.")
else:
    print("Fichier existe déjà.")
```

Suppression de fichier

- **os.remove()** permet de supprimer un fichier.

```
nom_fichier = "test.txt"
if not os.path.exists(nom_fichier):
    print("Le fichier n'existe pas.")
else:
    os.remove(nom_fichier)
    print("Fichier supprimé.")
```

Suppression du dossier vide

- **os.rmdir()** : Supprime un dossier vide

```
nom_dossier_vide = "DossierVide"

if os.path.exists(nom_dossier_vide):
    # Supprimer le dossier vide
    os.rmdir(nom_dossier_vide)
    print(f"Le dossier vide '{nom_dossier_vide}' a été supprimé.")
else:
    print(f"Le dossier vide '{nom_dossier_vide}' n'existe pas.")
```

Suppression du dossier non vide

- **shutil.rmtree()** : Supprime un dossier, même s'il contient des fichiers.

```python
import os
import shutil
nom_dossier_non_vide = "MonDossier"

if os.path.exists(nom_dossier_non_vide):
    # Supprimer le dossier non vide
    shutil.rmtree(nom_dossier_non_vide)
    print(f"Le dossier non vide '{nom_dossier_non_vide}' a été supprimé.")
else:
    print(f"Le dossier non vide '{nom_dossier_non_vide}' n'existe pas.")
```

Liste le contenu d’un dossier

- **os.listdir()** liste le contenu d’un dossier (fichiers et sous-dossiers).

```python
import os
#  Lister le contenu d'un dossier

contenu = os.listdir("DossierAvecContenu")
# contenu = os.listdir(".")
# contenu = os.listdir("../")
# contenu = os.listdir("/")
print("Contenu du répertoire courant :", contenu)
```

- Les mêmes chemin utilisés en linux

Renommé un fichier ou un dossier

- **os.rename()** permet de renommer un fichier ou un dossier.

```python
import os

# Renommer un fichier
os.rename("ancien_nom.txt", "nouveau_nom.txt")
os.rename("DossierAvecContenu", "newDirectory")
print("Fichier et dossier renommés.")
```

os.path.isfile() et os.path.isdir()

- **os.path.isfile()** : Vérifie si un chemin correspond à un fichier.
- **os.path.isdir()** : Vérifie si un chemin correspond à un dossier.

```python
import os

# Vérifier si un chemin est un fichier ou un dossier
chemin = "newDirectory"

if os.path.isfile(chemin):
    print("C'est un fichier.")
elif os.path.isdir(chemin):
    print("C'est un dossier.")
```

Copier, déplacer ou renommer un fichier

- **shutil.copy()** : Copie un fichier.

- **shutil.move()** : Déplace ou renomme un fichier.

```python
import shutil

# Copier un fichier
shutil.copy("source.txt", "destination.txt")

# Déplacer un fichier
shutil.move("source.txt", "newDirectory/source.txt")
```

Utilisation de glob.glob()

- Trouve des fichiers ou des dossiers selon un modèle (wildcards)

```
import glob

# Trouver tous les fichiers .txt
fichiers = glob.glob("*.txt")
print("Fichiers trouvés :", fichiers)
```

Extraction de l’extension

- **splitext()** permet de diviser un chemin de fichier en deux parties :
  le chemin de base et
  l'extension du fichier.

```python
import os

chemin = "document.txt"
base, extension = os.path.splitext(chemin)
print(f"Base : {base}, Extension : {extension}")

#Résultat:  Base : document, Extension : .txt
```