Fichiers

Fiche de synthèse

1 Pourquoi des fichiers

La mémoire vive est **volatile** : tout ce qu'elle contient disparaît quand le programme s'arrête. Un **fichier**, enregistré sur un disque, conserve les données d'une exécution à l'autre : scores, réglages, relevés de mesures.

2 Ouvrir un fichier

```
with open('eleves.txt', 'r',
          encoding='utf-8') as f:
    contenu = f.read()
# ici, le fichier est fermé
```

`open` reçoit le nom du fichier et le **mode d'ouverture**. Le bloc `with ... as f` range le fichier ouvert dans la variable `f` et le **ferme automatiquement** à sa sortie, même en cas d'erreur. Sans `with`, il faudrait appeler `f.close()`. L'encodage `utf-8` garantit l'affichage correct des lettres accentuées.

Mode	Rôle	Fichier existant	Fichier absent
`'r'`	lecture	lit le contenu	erreur
`'w'`	écriture	**efface** le contenu	crée le fichier
`'a'`	ajout	écrit à la fin	crée le fichier

3 Écrire

```
with open('animaux.txt', 'w',
          encoding='utf-8') as f:
    f.write('Chat\n')
    f.write('Chien\n')

with open('animaux.txt', 'a',
          encoding='utf-8') as f:
    f.writelines(['Girafe\n', 'Tigre\n'])
```

`write` écrit une chaîne, `writelines` une liste de chaînes ; ni l'une ni l'autre n'ajoute de retour à la ligne : on écrit `\n` soi-même. Elles n'acceptent que des chaînes : une note s'écrit avec une chaîne formatée, `f'{nom};{note}\n'`. Le mode `'w'` efface d'abord le fichier ; le mode `'a'` ajoute à la fin.

4 Lire

```
with open('eleves.txt', 'r',
          encoding='utf-8') as f:
    for ligne in f:
        print(ligne.strip())
```

Chaque ligne lue se termine par `\n` ; sans `strip`, `print` ajouterait un second retour à la ligne et l'on verrait une ligne vide entre deux noms. `strip` retire espaces, tabulations et `\n` aux deux bouts.

Lecture	Résultat
`for ligne in f`	une ligne à chaque tour, la plus économe en mémoire
`f.readline()`	la ligne suivante, par exemple pour sauter un en-tête
`f.read()`	tout le fichier en une seule chaîne, pour un petit fichier
`f.readlines()`	la liste des lignes, chacune terminée par `\n`

5 Découper une ligne

Lire un fichier structuré. Chaque ligne de `notes.txt` contient un nom et une note séparés par un point-virgule, comme `Alice Martin;16`.

```
with open('notes.txt', 'r',
          encoding='utf-8') as f:
    for ligne in f:
        parties = ligne.strip().split(';')
        nom = parties[0]
        note = int(parties[1])
        print(f'{nom} a obtenu {note}/20')
```

On retire le `\n`, on découpe sur le séparateur, puis on convertit en nombre ce qui doit l'être : une ligne lue est toujours une chaîne.

À retenir.

- `with open(nom, mode, encoding='utf-8') as f` ouvre un fichier et le ferme tout seul.
- `'r'` lit, `'w'` efface puis écrit, `'a'` ajoute à la fin.
- `write` n'ajoute pas de `\n` ; `strip` le retire à la lecture.
- `split(';')` découpe une ligne ; on convertit les nombres avec `int` ou `float`.