Chapitre 8
Les entrées/sorties

Pour les étudiants en génie civil – M1 RIB

Dr. Hachimi Dahhaoui
Université de Tlemcen

Mercredi 25 Novembre 2025

Diapo 2 – Pourquoi on apprend ça ?

🎯 Objectifs

- ❑ Comprendre la notion d'entrées (input) et de sorties (output)
- ❑ Formater proprement des chaînes de caractères (f-strings, format, %)
- ❑ Lire et écrire des fichiers texte simples
- ❑ Sauvegarder des données structurées avec le module json
- ❑ Relier ces notions à des exemples du génie civil

POURQUOI ????

- ❑ La plupart des programmes doivent lire ou sauvegarder des données.
- ❑ Vie quotidienne : listes de courses, contacts, scores de jeu, notes.
- ❑ Génie civil : résultats d'essais de sol, rapports de chantier, mesures de laboratoire.
- ❑ Python nous offre des outils simples pour automatiser ces tâches.

Diapo 3 – Entrée vs sortie

- Entrée (input) : données qui entrent dans le programme (clavier, fichier, capteur...).
- Sortie (output) : données produites par le programme (écran, fichier, impression...).

Exemples :

- Lire un fichier d'essais géotechniques → entrée.
- Écrire un rapport de résultats dans un fichier → sortie.

Est-ce que c’est claire ?

Diapo 4 – Mini-jeu : vrai ou faux ?

❑1. Lire un fichier .txt est une sortie. → **Faux.**

❑2. Afficher un message avec print() est une sortie. → **Vrai.**

❑ 3. Saisir une valeur avec input() est une entrée. → **Vrai.**

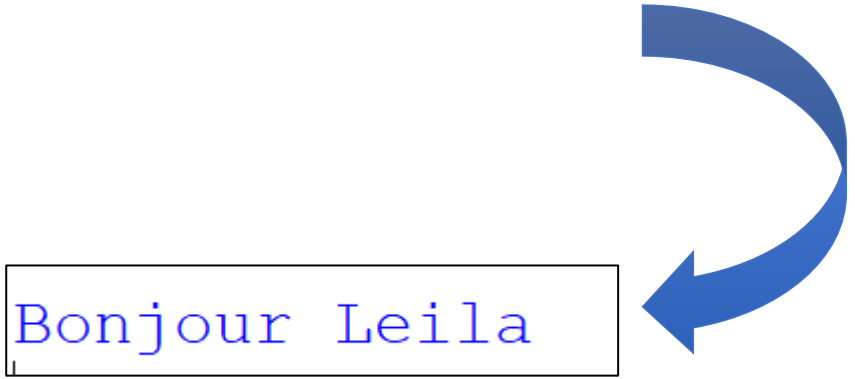
❑4. Écrire dans un fichier est une entrée. → **Faux.**

Diapo 5 – Plan du chapitre

- Formatage des chaînes de caractères (f-strings, format, %)
- Lecture et écriture de fichiers texte
- Sauvegarde de données structurées avec json
- Mini-projet : fiche de chantier sauvegardée en JSON

Diapo 6 – Les f-strings : méthode moderne

```
nom = "Leila"
age = 6
print(f"Ma fille s'appelle {nom} et elle a {age} ans.")
```



Diapo 7 – Principe des f-strings

- On place un f devant les guillemets : f"...".
- Les accolades { } permettent d'insérer des variables dans le texte.
- Python remplace {variable} par sa valeur avant d'afficher la chaîne.
- Avantage : lisible, moderne, recommandé dans le code actuel.

Diapo 8 – Exemples f-strings en génie civil

```
sigma = 120
print(f"Contrainte normale σ = {sigma} kPa")

hauteur = 3.5
print(f"Hauteur du voile : {hauteur} m")
```

Contrainte normale σ = 120 kPa

Hauteur du voile : 3.5 m

Diapo 9 – Formatage numérique avec f-strings

- On peut contrôler le nombre de décimales affichées :
- Syntaxe : {variable:.2f} → nombre avec 2 décimales.
- Utile pour afficher des résultats de calculs avec une précision maîtrisée.

Diapo 10 – Exemples de formatage numérique

Exemple

```
pi = 3.14159
print(f"{pi:.2f}")    # 3.14

densite = 19.245
print(f"Densité sèche = {densite:.1f} kN/m³")
```

3.14

Densité sèche = 19.2 kN/m³

Diapo 11 – corriger la f-string

- On veut afficher : "Le chantier X avance à 35 %."
- Code proposé : print(f"Le chantier {nom} avance à {35} %")
- Questions :
- 1. Quelle variable manque dans cet exemple ?
- 2. Comment écrire une version plus générale avec une variable avancement ?

```
nom = "Chantier X"

avancement = 35

print(f"{nom} avance à {avancement} %")
```

Diapo 12 – Méthode format()

- Ancienne méthode, encore très utilisée.
- On utilise "{}" comme emplacement, puis .format(...).
- Exemple : "Bonjour {}".format(nom)

Exemple

```
nom = "Samad"
nb = 5
print("Bonjour {}, vous avez {} notifications".format(nom, nb))
```

Diapo 13 – Exemple génie civil avec format()

```
phi = 35
print("Angle de frottement φ = {} degrés".format(phi))
```

```
Angle de frottement ϕ = 35 degrés
```

Diapo 14 – Ancienne méthode % (pour culture générale)

- Avant, on utilisait l'opérateur % pour formater les chaînes.
- Exemple : "Résultat = %.2f" % valeur
- Aujourd'hui, on privilégie plutôt les f-strings.

```
resultat = 5.678
print("Résultat = %.2f" % resultat)
```

Diapo 15 – Synthèse : quand utiliser quoi ?

- f-strings : méthode recommandée (simple, lisible).
- format() : encore utile si on ne peut pas utiliser f-strings (vieux code).
- % : à connaître pour lire du code ancien, mais à éviter dans le nouveau code.
- L'idée principale : rendre l'affichage clair pour l'utilisateur (unité, précision...).

Diapo 16 – Mini-exercices formatage (5 minutes)

- ❑ 1. Afficher : "Le chantier X avance à Y %" avec des variables chantier et avancement.
- ❑ 2. Afficher une hauteur de mur avec 1 décimale : 2.3 m.
- ❑ 3. Afficher la masse volumique : 18.7 kN/m³ avec une f-string et .1f.

```
chantier = "X"
avancement = 45
print(f"Le chantier {chantier} avance à {avancement} %")
```

```
h = 2.345
print(f"Hauteur du mur = {h:.1f} m")
```

```
gamma = 18.73
print(f"Masse volumique = {gamma:.1f} kN/m³")
```

Diapo 17 – Lecture et écriture de fichiers

- Fonction de base : open("nom_fichier", "mode").
- Mode "r" : lecture (read).
- Mode "w" : écriture (write, écrase le contenu).
- Mode "a" : ajout à la fin (append).

```
f = open("notes.txt", "r")
contenu = f.read()
print(contenu)
f.close()
```

Diapo 18 – Utiliser with open(...)

```
with open("data.txt", "r") as f:
    contenu = f.read()
    print(contenu)
# Le fichier est fermé automatiquement ici
```

Avantages de with open(...)

- ❑ Le fichier est fermé automatiquement même en cas d'erreur.
- ❑ Le code est plus lisible et plus court.
- ❑ C'est la bonne pratique recommandée en Python.

Diapo 19 – Lire un fichier ligne par ligne

```
with open("densite.txt", "r") as f:
    for ligne in f:
        print(ligne.strip())
```

Diapo 20 – Exemple de fichier densité

- Contenu d'un fichier densite.txt :
- Profondeur;Densité
- 1;18.5
- 2;19.0
- 3;19.2
- On peut ensuite traiter ces valeurs (conversion, statistiques...).

Diapo 21 – Écrire dans un fichier (mode w)

```python
with open("resultats.txt", "w") as f:
    f.write("Contrainte maximale = 250 kPa\n")
    f.write("Profondeur = 2.5 m")
```

✅ Contenu final du fichier `resultats.txt` :

```
java


Contrainte maximale = 250 kPa

Profondeur = 2.5 m
```

Diapo 22 – Ajouter à un fichier (mode a)

```
with open("journal.txt", "a") as f:
    f.write("Nouvelle entrée de chantier\n")
```

Résultat exact dans le fichier :

Une nouvelle ligne sera ajoutée :

```
Nouvelle entrée de chantier
```

Diapo 23 – quel mode choisir ?

- ❑ 1. Je veux recréer un fichier propre avec les résultats du jour → **mode "w".**
- ❑ 2. Je veux compléter un journal sans effacer les anciennes lignes
  → **mode "a".**
- ❑ 3. Je veux juste lire un fichier existant → **mode « r".**

Diapo 24 – Mini-exercice fichiers

- ❑ Créer un fichier texte chantier.txt qui contient :
- ❑ "Chantier : Route RN22"
- ❑ "Avancement : 40 %"
- ❑ Puis écrire un programme qui lit ce fichier et affiche son contenu à l'écran.

Diapo 25 – Mini-exercice fichiers

- ❑ Créer un fichier texte chantier.txt qui contient :
- ❑ "Chantier : Route RN22"
- ❑ "Avancement : 40 %"
- ❑ Puis écrire un programme qui lit ce fichier et affiche son contenu à l'écran.

chantier.txt

```
Chantier : Route RN22
Avancement : 40 %
```

Programme :

```
with open("chantier.txt", "r") as f:
    contenu = f.read()
    print(contenu)
```

Résultat affiché par Python :

```
Chantier : Route RN22
Avancement : 40 %
```

Diapo 26 – Erreurs fréquentes avec les fichiers

- FileNotFoundError : le fichier n'existe pas ou le chemin est mauvais.
- PermissionError : vous n'avez pas le droit d'écrire dans ce dossier.
- Astuce : vérifier le nom du fichier, son emplacement, et les extensions (.txt, .csv...).

Diapo 27 – Partie 3 – Sauvegarde avec JSON

- Idée : sauvegarder des données structurées (dictionnaires, listes).
- JSON est un format texte très utilisé dans les API et les applications web.
- On va voir comment utiliser le module json de Python.

Diapo 28 – C'est quoi JSON ?

- JSON = JavaScript Object Notation.
- Format texte qui ressemble beaucoup aux dictionnaires Python.
- Clés et valeurs : {"nom": "chantier A", "avancement": 40}.
- Facile à lire pour un humain et pour un programme.

Diapo 29 – Exemple simple de données (dictionnaire)

```
essai = {
      "sol": "argile",
      "sigma_max": 180,
      "profondeur": 2.5
}
```

Diapo 30 – Exemple simple de données (dictionnaire)

```
essai = {
        "sol": "argile",
        "sigma_max": 180,
        "profondeur": 2.5
}
```

Si on ajoute : `print(essai)`

```
{'sol': 'argile', 'sigma_max': 180, 'profondeur': 2.5}
```

Diapo 31 – Sauvegarder en JSON avec json.dump

```python
import json

with open("essai.json", "w") as f:
    json.dump(essai, f, indent=4)
```

Ce code enregistre un dictionnaire Python dans un fichier JSON.json.

- dump() transforme le dictionnaire en texte JSON.
- indent=4 permet d'avoir un fichier bien formaté, lisible.
- Le fichier essai.json est créé (ou écrasé) et contient les données structurées.

[1]

Diapo 32 – Sauvegarder en JSON avec json.dump

```python
import json

with open("essai.json", "w") as f:
    json.dump(essai, f, indent=4)
```

Ce code enregistr
- dump() transfo
- indent=4 perm
- Le fichier essa

💾 Contenu final de *essai.json* :

```json
{
    "sol": "argile",
    "sigma_max": 180,
    "profondeur": 2.5
}
```

[1]

Diapo 33 – Lire un fichier JSON avec json.load

```python
import json

with open("essai.json", "r") as f:
    data = json.load(f)

print(data["sigma_max"])
```

🎯 Résultat affiché :

```
180
```

[1]

Diapo 34 – Mini-exercice JSON : retrouver la clé

- À partir du dictionnaire essai :
- {"sol": "argile", "sigma_max": 180, "profondeur": 2.5}
- Comment afficher uniquement :
- • le type de sol ?
- • la contrainte maximale ?
- • la profondeur ?

Mini-projet : fiche chantier en JSON

- Créer un dictionnaire chantier avec :
- • nom, localisation, responsable, avancement.
- Sauvegarder ce dictionnaire dans un fichier chantier.json.
- Écrire ensuite un programme qui lit ce fichier et affiche :
- "Le chantier X situé à Y est à Z % d'avancement."

Dictionnaire chantier : exemple

```
chantier = {
    "nom": "Ouvrage OA12",
    "localisation": "Tlemcen",
    "responsable": "Ing. Samad",
    "avancement": 45
}
```

Affichage de la fiche chantier

```python
import json

with open("chantier.json", "r") as f:
    ch = json.load(f)

print(f"Le chantier {ch['nom']} situé à {ch['localisation']} "
      f"est à {ch['avancement']} % d'avancement.")
```

Synthèse : chaîne complète I/O

- 1. On prépare les données sous forme de dictionnaire (Python).
- 2. On sauvegarde en JSON pour conserver une structure propre.
- 3. On lit ce JSON plus tard pour poursuivre le travail (calculs, rapports...).
- → C'est la base de nombreux logiciels modernes.

QCM rapide (1)

- 1. Quel mode ouvre un fichier en lecture ? (r / w / a)
- 2. Quelle méthode est recommandée aujourd'hui pour formater des chaînes ?
- 3. JSON est plutôt :
- a) un langage de programmation b) un format de données texte c) une base de données

QCM rapide (2)

- 4. Que fait json.dump ?
- a) lit un fichier JSON b) écrit un dictionnaire dans un fichier JSON
- 5. Que signifie l'expression {val:.2f} dans une f-string ?
- 6. Laquelle de ces affirmations est vraie :
- a) "w" ajoute à la fin du fichier b) "a" écrase le fichier c) "r" ne modifie pas le fichier

Idées de TP / mini-projets

- TP 1 : carnet de notes simple (entrée au clavier + sauvegarde dans un fichier).
- TP 2 : tableau de résultats d'essais (texte → JSON).
- TP 3 : petit gestionnaire de chantiers (lecture/écriture JSON).

Conclusion du chapitre

- Vous savez maintenant :
- • formater des résultats avec f-strings et format() ;
- • lire et écrire des fichiers texte ;
- • sauvegarder des données structurées en JSON.
- Ces outils seront utilisés dans les projets suivants (IA, automatisation, etc.).