📓 Cours : Manipulation de Fichiers et Gestion des Erreurs en Python

🎯 Objectif du cours :

Apprendre à lire, écrire et modifier des fichiers en Python, tout en maîtrisant la gestion des erreurs pour éviter les plantages de programme.

---

1. 📂 Manipulation de Fichiers

a. Ouvrir un fichier

```
fichier = open("donnees.txt", "r")  # "r" = lecture
contenu = fichier.read()
print(contenu)
fichier.close()
```

b. Modes d'ouverture

- `"r"` : lecture
- `"w"` : écriture (écrase le fichier existant)
- `"a"` : ajout (append)
- `"x"` : création exclusive
- `"b"` : mode binaire
- `"t"` : mode texte (par défaut)

c. Écrire dans un fichier

```
fichier = open("notes.txt", "w")
fichier.write("Python est génial !\n")
fichier.write("Apprendre à manipuler des fichiers est utile.")
fichier.close()
```

d. Lire ligne par ligne

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

e. Lire un fichier ligne par ligne dans une liste

```
with open("donnees.txt", "r") as f:
    lignes = f.readlines()
    print(lignes)
```

f. Ajouter du contenu sans écraser

```
with open("notes.txt", "a") as f:
    f.write("\nAjout d'une nouvelle ligne.")
```

---

2. ⚠ Gestion des Erreurs (Exceptions)

a. Pourquoi gérer les erreurs ?

- Éviter les crashs
- Gérer les cas inattendus (fichier manquant, division par zéro, etc.)

b. Structure try-except

```
try:
    x = 10 / 0
except ZeroDivisionError:
    print("Erreur : division par zéro.")
```

c. Gestion multiple

```
try:
    fichier = open("inexistant.txt", "r")
    contenu = fichier.read()
except FileNotFoundError:
    print("Fichier non trouvé.")
except IOError:
    print("Erreur d'entrée/sortie.")
```

d. Utilisation de finally

```
try:
    fichier = open("exemple.txt", "r")
    print(fichier.read())
except:
    print("Erreur lors de la lecture.")
finally:
    fichier.close()
```

e. Lever une exception manuellement

```
age = -5
if age < 0:
    raise ValueError("L'âge ne peut pas être négatif.")
```

---

3. ✅ Bonnes pratiques

- Toujours **fermer les fichiers** (ou utiliser `with`)
- Toujours **gérer les exceptions** avec `try...except`
- Ne jamais supposer qu'un fichier existe sans vérification
- Utiliser `os.path.exists()` ou la bibliothèque `pathlib` pour les chemins

📚 Résumé :

- Python permet de lire/écrire facilement des fichiers texte.
- La gestion des exceptions est cruciale pour la robustesse de vos programmes.
- Utilisez `with open(...)` et `try...except` pour écrire du code propre et sûr.