S1J1-2 : Introduction à Python.

Objectif :

Apprendre les concepts de base de Python, mais en les appliquant dans le contexte de l'ingénierie des données. Vous allez comprendre comment le langage est utilisé pour manipuler, nettoyer et transformer des données.

---

1. Les bases fondamentales de Python.

Pour devenir Data Engineer, il est crucial de maîtriser les éléments suivants :

a. Types de données.

Types de base : `int`, `float`, `str`, `bool` .*Exemple :* Comprendre comment gérer différents types de données provenant d'une source externe (par exemple, un fichier CSV ou une API).

```
age = 30  # int
temperature = 22.5  # float
name = "John"  # str
is_active = True  # bool
```

b. Structures de données essentielles.

Listes (`list`) : Pour stocker des séries de données.

```
names = ["Alice", "Bob", "Charlie"]
```

Dictionnaires (`dict`) : Pour des associations clé-valeur (utile pour représenter des colonnes de données).

```
employee = {"name": "Alice", "age": 30, "role": "Engineer"}
```

Sets (`set`) : Pour des collections uniques, comme supprimer des doublons dans des datasets.

```
unique_ids = {101, 102, 103}
```

c. Contrôles de flux.

Conditions (`if`, `else`, `elif`) : Pour appliquer des règles de transformation de données.

```
if age > 18:
    print("Adult")
```

Boucles (`for`, `while`) : Pour parcourir des données en batch.

```
for name in names:
    print(name)
```

---

2. Manipulation des fichiers.

Les Data Engineers manipulent souvent des fichiers, comme des fichiers CSV ou JSON. Apprenez les bases suivantes :

a. Lecture/écriture de fichiers.

Lire un fichier ligne par ligne.

```
with open("data.txt", "r") as file:
    for line in file:
        print(line.strip())
```

Écrire des données dans un fichier.

```
with open("output.txt", "w") as file:
    file.write("Hello, Data Engineering!")
```

b. Manipulation de fichiers CSV.

Utilisez la bibliothèque `csv` pour lire/écrire des fichiers CSV :

```
import csv

with open("data.csv", "r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

---

3. Concepts avancés utiles pour l'ingénierie des données.

Ces concepts sont incontournables pour les tâches fréquentes de transformation des données :

a. Compréhension des listes et dictionnaires.

Utilisez des expressions compactes pour manipuler les données.

```
squared = [x**2 for x in range(10)]
print(squared)
```

b. Gestion des erreurs.

Les données ne sont pas toujours propres : apprenez à anticiper et gérer les erreurs.

```
try:
    value = int("abc")  # Cela lèvera une erreur
except ValueError:
    print("Impossible de convertir en entier.")
```

c. Modules et bibliothèques.

- `os` : Pour manipuler des fichiers et répertoires.
- `sys` : Pour gérer des arguments de scripts.
- `json` : Pour lire et écrire des données au format JSON.

```
import json

data = '{"name": "Alice", "age": 30}'
parsed = json.loads(data)
print(parsed["name"])
```

---

4. Introduction à l'environnement de travail.

Pour l'ingénierie des données, utilisez des environnements robustes et professionnels :

- **Installer et configurer Python** : Assurez-vous d'avoir Python 3.x installé.
- **Utiliser Jupyter Notebook** : Pour tester et expérimenter avec des données rapidement.
- **IDE recommandé** : PyCharm ou VS Code.

---

5. Exercices pour solidifier ces concepts.

Créer une liste de prénoms, filtrer ceux qui commencent par une certaine lettre, et enregistrer les résultats dans un fichier.

Lire un fichier JSON contenant des informations d'utilisateur, extraire uniquement les utilisateurs adultes et les écrire dans un autre fichier JSON.

Exemple d'exercice :

Créer un script Python qui :

- Lit un fichier CSV contenant des données de ventes.
- Calcule la somme des ventes totales.
- Identifie les produits les plus vendus.
- Sauvegarde le rapport dans un fichier texte.