Collège de Maisonneuve
Formation continue

Professeur: Ahmed Imed Eddine Rabah
Mail: arabah@cmaisonneuve.qc.ca
Cours: Concept de Programmation 1

Projet 3 : Gestion d'une Serre Intelligente - SMAG

- **Respectez strictement les instructions mentionnées dans cet énonce.**
- À rendre au plus tard le Samedi 08 Février 2025 à 23h59, Aucun retard accepté.
- Ce projet vaut 40% de la moyenne du cours.
- **Fichier à remettre** :
  - Vous devez remettre sur **Teams** uniquement un fichier **Jupyter Notebook** par le chef de groupe seulement.
  - Nom du fichier : `projet<avec numéro de projet et numéro du groupe>.ipynb`.
  - **Ne soumettez pas** de fichier PDF, Word ou tout autre format.
- Dans le fichier `.ipynb`, on trouve le `Code` et le `Rapport` :
  - Remplissez une cellule **Markdown** au début pour introduire votre projet, incluant :
    - Nom du projet.
    - Description du projet.
    - Votre nom et prénom.
    - Votre groupe.
  - Avant chaque cellule de code, expliquez la partie de ce code la dans une cellule `Markdown`.
  - Commentez chaque section de votre code pour expliquer son rôle (dans les blocs de code).
- **Structure du fichier** :
  - Le fichier doit être organisé, avec des blocs de code et de texte clairement séparés.
  - Assurez-vous que votre code s'exécute correctement dans l'ordre des cellules.
  - Aucun Template n'est donné donc soyez organiser et créative.

Objectif du Projet :

Avec l'évolution des technologies modernes, les serres intelligentes permettent de surveiller et d'optimiser la croissance des plantes grâce à des capteurs et des systèmes automatisés. Ce projet propose de simuler un système de gestion de serre intelligente, où vous devez collecter, analyser et visualiser des données environnementales pour prendre des décisions éclairées.

Vous devez développer une application Python interactive, intégrant des fonctionnalités avancées de gestion, d'analyse et de visualisation des données environnementales, tout en utilisant des outils interactifs comme `ipywidgets` et des notifications via `plyer`.

Fonctionnalités Demandées :

Partie 1 : Structure des Dictionnaires (12%)

Pour organiser les données, les informations sur les capteurs, les paramètres environnementaux et les alertes seront stockées dans des dictionnaires structurés comme suit:

1.) Dictionnaire des paramètres environnementaux :

- Enregistre les données des capteurs.
- Contient :
  - `Date` : Date et heure de l'enregistrement.
  - `Température` : Température mesurée (°C).
  - `Humidité` : Humidité mesurée (%).
  - `Lumière` : Lumière mesurée (lux).
  - `CO2` : Dioxyde de carbone mesuré (ppm).
- Exemple :

```
environment_data = [
    {
        "Date": "2024-12-01 08:00",
        "Température": 25.0,
        "Humidité": 60,
        "Lumière": 300,
        "CO2": 400},
    {
        "Date": "2024-12-01 12:00",
        "Température": 28.0,
        "Humidité": 55,
        "Lumière": 800,
        "CO2": 420}
]
```

2.) Dictionnaire des seuils optimaux :

- Définit les seuils idéaux pour chaque paramètre environnemental.
  - `Température` : Intervalle idéal (min, max).
  - `Humidité` : Intervalle idéal (min, max).
  - `Lumière` : Intervalle idéal (min, max).
  - `CO2` : Intervalle idéal (min, max).
- Exemple :

```
optimal_thresholds = {
    "Température": {"min": 20, "max": 30},
    "Humidité": {"min": 50, "max": 70},
    "Lumière": {"min": 200, "max": 1000},
    "CO2": {"min": 350, "max": 450}
}
```

3.) Dictionnaire des alertes :

- Stocke les alertes générées lorsque les seuils sont dépassés.
- Les informations stockées pour chaque emprunt:
  - `Date` : Date et heure de l'alerte.
  - `Paramètre` : Nom du paramètre concerné.
  - `Valeur` : Valeur mesurée.
  - `Message` : Description de l'alerte.
- Exemple :

```
alerts = [
    {
    "Date": "2024-12-01 12:00",
    "Paramètre": "Température",
    "Valeur": 35.0,
    "Message": "Température trop élevée"
    },
    {
    "Date": "2024-12-01 18:00",
    "Paramètre": "Humidité",
    "Valeur": 45.0,
    "Message": "Humidité trop basse"
    }
]
```

- Quand une nouvelle alerte est ajoutée à ce dictionnaire, une notification s'affiche sur votre machine en utilisant la librairie `plyer`. La notification doit afficher le message de l'alerte ajoutée, par exemple : "Humidité trop basse".

Partie 2 : Requêtes et Analyses 55%

Vous devez implémenter les 9 requêtes avancées suivantes :

1.) (5%) Ajouter les données des capteurs

- Ajouter des données de capteurs manuellement via un widget interactif (utiliser `ipywidgets.Text` et `ipywidgets.IntSlider`).

2.) (5%) Afficher les données les plus récentes.

- Trier les entrées par date.
- Afficher les 5 dernières données environnementales sous forme lisible.

3.) (5%) Calculer les moyennes des paramètres environnementaux sur une période donnée (sélection via ipywidgets.DatePicker).

- L'utilisateur sélectionne une plage de dates via `ipywidgets.DatePicker`.
- Filtrer les données dans cette plage.
- Calculer les moyennes pour chaque paramètre (température, humidité, lumière, CO2).

4.) (5%) Lister toutes les alertes générées avec un filtre sur le paramètre via `ipywidgets.Dropdown`.

- Utiliser un `ipywidgets.Dropdown` pour choisir un paramètre (ex. : Température).
- Filtrer et afficher toutes les alertes associées.

5.) (5%) Analyser les tendances journalières.

- L'utilisateur choisit une date et un paramètre (ex. : Température).
- Afficher les valeurs enregistrées à différents moments de cette journée.

6.) (5%) Trouver le moment avec les conditions les plus optimales.

- Identifier les enregistrements les plus proches des seuils optimaux pour tous les paramètres.
- Calculer un score pour chaque entrée: Score = Somme des écarts absolus entre chaque valeur mesurée et sa plage optimale.
- Afficher l'entrée avec le score le plus faible.

7.) (5%) Proposer des actions correctives (ex. : augmenter la ventilation, réduire l'éclairage).

- Automatiser les suggestions pour améliorer les conditions dans la serre.
- Générer une liste d'actions en fonction des alertes générées.
- Exemple:
  - Température trop élevée: Réduire la température (augmenter la ventilation).
  - Humidité trop basse: Ajouter de l'humidité (augmenter l'irrigation).

8.) (10%) Identifier le paramètre le plus problématique.

- Analyser les alertes pour déterminer quel paramètre (Température, Humidité, Lumière, ou CO2) a généré le plus d'alertes dans une période donnée.
- L'utilisateur sélectionne une période via un widget interactif (`ipywidgets.DatePicker` pour la date de début et de fin).
- Filtrer les alertes pour cette période.
- Compter le nombre d'alertes pour chaque paramètre et identifier celui qui a généré le plus de problèmes.

9.) (10%) (**Documentez vous sur la librairie** `matplotlib`) Diagramme en barres interactif

- Comparer les moyennes journalières des paramètres environnementaux.
- Calculer les moyennes journalières pour chaque paramètre.
- Afficher un graphique en **barres** où chaque barre représente un paramètre (Température, Humidité, etc.).

Partie 3 : Visualisation (08%)

- Visualisation d'un Diagramme en lignes.
  - Affiche l'évolution d'un paramètre (ex. : température) sur une période donnée.
  - Utilisez un `ipywidgets.Dropdown` pour sélectionner le paramètre.
  - L'utilisateur sélectionne un paramètre et une période via `ipywidgets`.
  - Tracer un graphique montrant les valeurs mesurées.
  - Exemple :

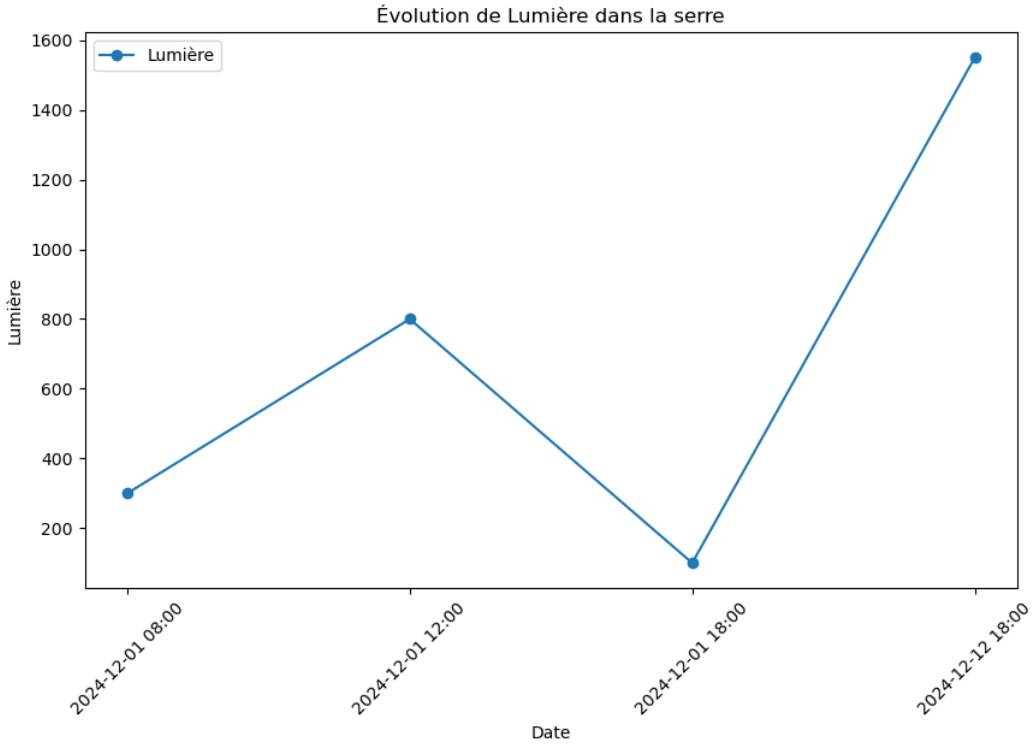

Figura 1: Évolution du paramètre **Lumière** dans la serre du 01-12-2024 au 12-12-2024.

Partie 4 : Menu Interactif (15%)

Menu Interactif (5%)

Créer un menu interactif qui permet à l'utilisateur de choisir les requêtes qu'il souhaite exécuter. Exemple de menu:

```
--- Menu Principal ---
1. Ajouter les données des capteurs.
2. Afficher les données les plus récentes.
3. Calculer les moyennes des paramètres environnementaux sur une période donnée.
4. Lister toutes les alertes générées avec un filtre sur le paramètre
5. Analyser les tendances journalières.
6. Trouver le moment avec les conditions les plus optimales.
7. Proposer des actions correctives
8. Identifier le paramètre le plus problématique.
9. Visualisation: Diagramme en barres interactif.
10. Quitter.
```

Sauvegarde des Données (10%)

À la fin de l'exécution, toutes les données doivent être sauvegardées dans des fichiers CSV:

- (5%) `environment_data.csv` : Date, Température, Humidité, Lumière, CO2.
- (5%) `alerts.csv` : Date, Paramètre, Valeur, Message.

Partie 4.beta : Interface Graphique - Bonus (30%)

Vous pouvez développer une interface graphique pour votre application (**Ce n'est pas obligatoire**) . Par exemple :

- Utiliser les librairies `tkinter`, `PyQt`, ou `Kivy`.
- Fournir des boutons pour chaque requête.
- Afficher les graphiques et résultats dans des fenêtres interactives.

Partie 5 : Présentation du projet (10%)

À la fin du projet, chaque groupe d'étudiants devra réaliser une présentation orale obligatoire devant l'ensemble de la classe. Cette présentation représente une étape clé pour :

- Évaluer votre compréhension du projet.
- Valoriser votre travail collectif.
- Juger votre capacité à communiquer et expliquer des concepts techniques.

Durée

- 15 à 20 minutes pour présenter le projet.
- 5 minutes pour une session de questions-réponses.

Contenu de la Présentation :

1.) Introduction (2-3 minutes) :

- Présenter brièvement le projet et son objectif.
- Expliquer pourquoi ce sujet est pertinent et intéressant.

2.) Fonctionnalités principales (5-7 minutes) :

- Décrire les principales fonctionnalités développées.
- Montrer comment l'utilisateur interagit avec le programme (menu, options, etc.).

3.) Aspect technique (5-7 minutes) :

- Expliquer les bibliothèques utilisées.
- Décrire la structure du code.
- Présenter les défis techniques rencontrés et comment ils ont été résolus.

4.) Démonstration (5 minutes) :

- Exécuter le programme en direct pour montrer son fonctionnement.
- Exemple d'utilisation : effectuer une ou deux requêtes interactives.
- Afficher au moins une visualisation graphique (diagramme en cercle ou graphique en barres).

5.) Session Questions-Réponses (5 minutes) :

- Répondre aux questions du prof ou des autres étudiants.
- Montrer votre capacité à défendre vos choix et expliquer votre raisonnement.

Cette présentation permet non seulement de valider le travail réalisé, mais aussi de renforcer les compétences en communication technique, en gestion de projet, et en esprit critique. Juste amusez vous a la faire - avec un peu de stresse c'est cool. ;)

Rapport Final

Le rapport final doit être rédigé dans un fichier Jupyter Notebook (`.ipynb`), structuré en sections claires et lisibles. Ce format permet de mélanger du texte explicatif (en `Markdown`) et du code Python, tout en incluant les résultats directement dans le document. Cela facilite la démonstration des concepts, du fonctionnement du code, et des visualisations.

Documentation et librairie d'aide

Bibliothèque	**Usage principal**	**Documentation**
csv	Lire et écrire des fichiers CSV pour enregistrer les données des livres, utilisateurs et emprunts.	Lien
json	Manipuler et exporter les données en JSON.	Lien
datetime	Gérer les dates et effectuer des calculs temporels.	Lien
matplotlib	Créer des visualisations, comme des graphiques en cercle et en barres.	Lien
ipywidgets	Créer une interface interactive dans Jupyter Notebook pour ajouter des données, filtrer et sélectionner des paramètres.	Lien
plyer	Afficher des notifications système lorsqu'une alerte est ajoutée.	Lien
tkinter	Créer une interface graphique avec des boutons et des champs interactifs (optionnel).	Lien

Autre aide et recommandations Générales :

- ChatGPT : Utilisez-le intelligemment.
- W3School : Pour toute les fonctionnalités de Python (Dictionnaire / functions / ...etc).

Outils collaboration et travail en groupe :

- **Gestion de temps et de tache**: Trello
- **Coder en temps réel en groupe dans un seul .ipynb** : Google Colab

Message de soutien

- Si vous ressentez une baisse de motivation, rencontrez des difficultés ou faites face à des imprévus, sachez que c'est tout à fait normal dans le cadre d'un projet qui représente 40% de la moyenne du cours. Travaillez en équipe, communiquez avec vos camarades, organisez votre temps efficacement et répartissez les tâches de manière équilibrée selon vos compétences et disponibilités.
- En cas de conflit ou de situation délicate, je reste disponible et à votre écoute. N'hésitez pas à me contacter si vous avez besoin de conseils ou d'assistance.

Bon courage à tous !