Mini-projet carnet d’adressse

Objectifs :

Créer un programme Python permettant :

- d'**ajouter**, **rechercher**, **afficher** des contacts,
- d'**enregistrer** les contacts dans un fichier texte,
- de **relire** les contacts à l'ouverture.

Table des matières

I. Mini-projet : Carnet d’adresses simplifié....1
1. Connaissances mobilisées :....1
2. Fonctionnalités attendues :....1
3. Exemple de fonctionnement....2
4. Conseils pour l'organisation du code :....2
5. Variante pour aller plus loin (optionnel)....2
II. Squelette de code - Carnet d’adresses simplifié....3
1. Explications et suggestions pour les élèves :....4
2. Conseils pour les élèves :....4

I. Mini-projet : Carnet d’adresses simplifié

1. Connaissances mobilisées :

Thème	Application dans le projet
Structures conditionnelles	Menus et choix utilisateur
Boucles	Répéter les actions tant que l'utilisateur veut
Chaînes de caractères	Manipulation des noms, téléphones...
Listes et dictionnaires	Stocker plusieurs contacts
Fonctions	Organiser le programme
Fichiers	Sauvegarde et chargement des contacts

2. Fonctionnalités attendues :

1. **Charger les contacts existants** depuis un fichier `"contacts.txt"` (s'il existe).
2. **Afficher un menu** proposant :
   - Ajouter un nouveau contact
   - Rechercher un contact par nom
   - Afficher tous les contacts
   - Quitter et enregistrer
3. Chaque contact est un **dictionnaire** avec par exemple :

```
{"nom": "Dupont", "prenom": "Alice", "telephone": "0601020304"}
```

4. Tous les contacts sont stockés dans une **liste** de dictionnaires.
5. **À la fermeture**, tous les contacts sont enregistrés dans le fichier `"contacts.txt"` (format simple lisible : une ligne par contact).

3. Exemple de fonctionnement

```
Bienvenue dans votre Carnet d'adresses !

Que voulez-vous faire ?
1 - Ajouter un contact
2 - Rechercher un contact
3 - Afficher tous les contacts
4 - Quitter

Votre choix : 1

Nom : Durand
Prénom : Paul
Téléphone : 0708091011

Contact ajouté avec succès !

Que voulez-vous faire ?
1 - Ajouter un contact
2 - Rechercher un contact
3 - Afficher tous les contacts
4 - Quitter

Votre choix : 4

Contacts enregistrés. À bientôt !
```

4. Conseils pour l'organisation du code :

- Une fonction `charger_contacts()` → lit `"contacts.txt"` et retourne une liste de contacts.
- Une fonction `sauver_contacts(contacts)` → écrit la liste dans `"contacts.txt"`.
- Une fonction `ajouter_contact(contacts)` → demande les infos à l'utilisateur et ajoute à la liste.
- Une fonction `rechercher_contact(contacts)` → recherche dans la liste par le nom.
- Une fonction `afficher_contacts(contacts)` → affiche tous les contacts.

5. Variante pour aller plus loin (optionnel)

- Afficher les résultats triés par nom.
- Pouvoir supprimer un contact.

II. Squelette de code - Carnet d'adresses simplifié

```
# Fonction pour charger les contacts depuis un fichier
def charger_contacts(nom_fichier):
    contacts = []
    try:
        with open(nom_fichier, "r") as fichier:
            for ligne in fichier:
                # Chaque ligne représente un contact sous forme de dictionnaire
                # À adapter selon la structure du fichier
                contact = eval(ligne.strip())  # Convertir la ligne en
dictionnaire
                contacts.append(contact)
    except FileNotFoundError:
        print("Aucun fichier trouvé. Création d'un nouveau carnet d'adresses.")
    return contacts

# Fonction pour sauvegarder les contacts dans un fichier
def sauver_contacts(contacts, nom_fichier):
    with open(nom_fichier, "w") as fichier:
        for contact in contacts:
            fichier.write(str(contact) + "\n")

# Fonction pour ajouter un contact
def ajouter_contact(contacts):
    nom = input("Nom : ")
    prenom = input("Prénom : ")
    telephone = input("Téléphone : ")

    # Créer un dictionnaire pour le nouveau contact
    contact = {"nom": nom, "prenom": prenom, "telephone": telephone}
    contacts.append(contact)
    print("Contact ajouté avec succès !")

# Fonction pour rechercher un contact par nom
def rechercher_contact(contacts):
    nom_recherche = input("Entrez le nom du contact à rechercher : ")
    for contact in contacts:
        if contact["nom"].lower() == nom_recherche.lower():
            print(f"Contact trouvé : {contact}")
            return
    print("Contact non trouvé.")

# Fonction pour afficher tous les contacts
def afficher_contacts(contacts):
    if not contacts:
        print("Aucun contact à afficher.")
    else:
        for contact in contacts:
            print(f"Nom : {contact['nom']}, Prénom : {contact['prenom']},
Téléphone : {contact['telephone']}")

# Fonction principale du menu
def menu():
    contacts = charger_contacts("contacts.txt")  # Charger les contacts
existants
    while True:
        print("\nQue voulez-vous faire ?")
```

```
        print("1 - Ajouter un contact")
        print("2 - Rechercher un contact")
        print("3 - Afficher tous les contacts")
        print("4 - Quitter")

        choix = input("Votre choix : ")

        if choix == "1":
            ajouter_contact(contacts)
        elif choix == "2":
            rechercher_contact(contacts)
        elif choix == "3":
            afficher_contacts(contacts)
        elif choix == "4":
            sauver_contacts(contacts, "contacts.txt")  # Sauvegarder avant de
quitter
            print("Contacts enregistrés. À bientôt !")
            break
        else:
            print("Choix invalide, veuillez réessayer.")

# Lancer le programme
if __name__ == "__main__":
    menu()
```

1. Explications et suggestions pour les élèves :

- **charger_contacts** : Cette fonction lit le fichier `"contacts.txt"` et transforme chaque ligne en dictionnaire pour construire la liste des contacts.
- **sauver_contacts** : Sauvegarde la liste des contacts sous forme de dictionnaires dans un fichier texte.
- **ajouter_contact** : Demande les informations de l'utilisateur (nom, prénom, téléphone) et crée un nouveau contact sous forme de dictionnaire.
- **rechercher_contact** : Permet de rechercher un contact par nom. Si trouvé, il affiche le contact.
- **afficher_contacts** : Affiche tous les contacts présents dans le carnet.
- **menu** : Affiche un menu avec les options pour l'utilisateur, répète les demandes jusqu'à ce que l'utilisateur choisisse de quitter.

2. Conseils pour les élèves :

- Ils devront **compléter** les parties manquantes, notamment la gestion des contacts sous forme de dictionnaires dans le fichier.
- **Tester** chaque fonction indépendamment avant de tout assembler dans la fonction `menu()`.
- **Adapter** la lecture et l'écriture des fichiers selon le format qu'ils choisiront (par exemple, en utilisant `eval()` pour convertir chaque ligne en dictionnaire).