Exercice 1 (5 points)

Cet exercice porte sur le traitement des données en table et les bases de données.
Il est constitué de deux parties indépendantes.

Un étudiant souhaite développer une application permettant de faciliter le covoiturage pour les déplacements du quotidien. Dans cet objectif, il étudie des données extraites de la Base Nationale des Lieux de Covoiturage (BNLC), disponible sur le site data.gouv.fr.

Dans un premier temps (partie A), les données d'une table décrivant des lieux de covoiturage (adresse postale, nombre de places, ...) sont manipulées à l'aide d'un tableau contenant des dictionnaires en langage Python.

Dans un second temps (partie B), une base de données contenant deux tables (les sites de covoiturage et les caractéristiques des communes de France) est exploitée à l'aide du langage SQL.

Partie A : traitement de données en table

Une table est implémentée par un tableau nommé `tab_lieux` contenant des dictionnaires en langage Python. Chaque dictionnaire correspond à un lieu de stationnement pour le covoiturage.

Les clés des dictionnaires, communes à tous les dictionnaires, correspondent aux descripteurs utilisés pour cette table :

`id_lieu` : identifiant du lieu, la donnée est un entier (chaque lieu possède un identifiant unique) ;

`ad_lieu` : adresse du lieu, la donnée est une chaîne de caractères ;

`insee` : code INSEE de la commune où se trouve le lieu, la donnée est une chaîne de caractères ;

`nb_places` : nombre de places du lieu, la donnée est un entier ;

`type` : nature du parking (supermarché, parking municipal, aire de stationnement située en sortie d'autoroute, ...), la donnée est une chaîne de caractères.

On donne en illustration les trois premiers éléments de ce tableau de dictionnaires,

```
tab_lieux = [
{"id_lieu": 1, "ad_lieu": "Place De La Fontaine", "insee":
"1024", "nb_places": 5, "type": "Supermarché"},
{"id_lieu": 2, "ad_lieu": "La Boisse", "insee": "1049",
"nb_places": 100, "type": "Parking municipal"},
{"id_lieu": 3, "ad_lieu": "Château-Gaillard", "insee": "1089",
"nb_places": 15, "type": "Sortie autoroute"},
 ...  ]
```

1. Accès aux informations du tableau :

a) Donner, sans justifier, la valeur à laquelle on accède avec l'instruction `tab_lieux[0]["insee"]`.

b) Écrire l'instruction qui permet d'obtenir la valeur `"La Boisse"`.

2. On propose trois blocs d'instructions pour parcourir le tableau et afficher le nombre de places des lieux de covoiturage.

Parmi ces trois propositions, deux seulement sont correctes. Indiquer, sans justifier, les deux propositions correctes.

Proposition 1

```
1 for i in range(len(tab_lieux)):
2     print(dico["nb_places"])
```

Proposition 2

```
1 for i in range(len(tab_lieux)):
2     print(tab_lieux[i]["nb_places"])
```

Proposition 3

```
1 for dico in tab_lieux:
2     print(dico["nb_places"])
```

3. On dispose de la fonction ci-dessous :

```
1  def fonction_inconnue(table, n, nom_type):
2      """
3      Entrée : un tableau de dictionnaires, un entier, une
4      chaine de caractères
5      Sortie : un entier
6      """
7      k = 0
8      for dico in table:
9          if dico["nb_places"] >= n and dico["type"] == nom_type:
10             k = k + 1
11     return k
```

Décrire, dans le contexte de l'exercice, ce que renvoie cette fonction, lors de l'appel :

```
fonction_inconnue(tab_lieux, 100, "Sortie autoroute")
```

4. La fonction, dont le code est proposé ci-après, permet de renvoyer le code INSEE du lieu de covoiturage possédant le plus grand nombre de places.

Recopier et compléter ce code

```
def insee_max_places(table):
    """
    Entrée : un tableau de dictionnaires
    Sortie : une chaine de caractères (le code INSEE du lieu
    possédant le plus grand nombre de places)
    """
    maxi = ...
    code_insee = ...
    for dico in table:
        if ...
            maxi = ...
            code_insee = ...
    return code_insee
```

5. La fonction dont les spécifications sont données ci-après, prend en paramètres une table de lieux de covoiturage au format tableau contenant des dictionnaires et le nom d'un type de lieu de covoiturage.

Recopier et compléter le code de cette fonction. Dans le code de la fonction, les trois points (. . .) peuvent correspondre à une ou plusieurs lignes de programme.

```
def moyenne_par_type(table, nom_type):
    """
    Entrée : un tableau de dictionnaires et une chaîne de
    caractères (type de lieu)
    Cette fonction renvoie, parmi les lieux de covoiturage
    dont le type est nom_type, la moyenne du nombre de places
    pour le type choisi.
    Sortie : un flottant
    """
    ...
    return moyenne
```

Exemple, avec la table `tab1` ci-après, contenant uniquement trois enregistrements :

```
tab1 = [
{"id_lieu":1, "ad_lieu":"Place Du marché", "insee":"1032",
"nb_places":10, "type":"Supermarché"},
{"id_lieu":2, "ad_lieu":"La Pesse", "insee":"1058",
"nb_places":100, "type":"Parking municipal"},
{"id_lieu":3, "ad_lieu":"Le Pautet", "insee":"1075",
"nb_places":20, "type":"Supermarché"}
]
```

`moyenne_par_type(tab1, "Supermarché")` vaut `15`.

Partie B : base de données

On pourra utiliser les mots du langage SQL suivants :

`SELECT, FROM, WHERE, JOIN, INSERT INTO, VALUES, COUNT.`

Afin de pouvoir gérer un site de covoiturage, on utilise une base de données contenant les relations `LIEU` et `COMMUNE`.

Le schéma relationnel, où les clés primaires sont soulignées et les clés étrangères sont précédées du symbole #, est le suivant :

`LIEU(id_lieu : entier, ad_lieu : texte, #code_insee : texte, nb_places : entier, type : texte)`

`COMMUNE(code_insee : texte, nom : texte, departement : texte, region : texte, latitude : décimal, longitude : décimal)`

On rappelle qu'en langage SQL la fonction d'agrégation `COUNT` permet de compter un nombre d'enregistrements.

Par exemple, pour déterminer le nombre de lieux dans la table `LIEU`, on peut utiliser la requête suivante :

```
1 SELECT COUNT(id_lieu)
2 FROM LIEU ;
```

On donne un extrait des deux premières lignes de ces relations :

LIEU

id_lieu	ad_lieu	code_insee	nb_places	type
1	"Place De La Fontaine"	"01001"	5	"Supermarché"
2	"La Boisse"	"01049"	100	"Parking municipal"

COMMUNE

code_insee	nom	departement	Region	latitude	longitude
"01001"	"L'ABERGEMENT-CLEMENCIAT"	"AIN"	"RHONE-ALPES"	46.1534	4.9261
"01002"	"L'ABERGEMENT-DE-VAREY"	"AIN"	"RHONE-ALPES"	46.0092	5.4280

6. On se place dans la relation `COMMUNE`. Expliquer pourquoi deux communes ne peuvent pas posséder le même code INSEE `code_insee`.

7. Écrire une requête SQL permettant d'obtenir l'identifiant `id_lieu` et le code INSEE `code_insee` des lieux de covoiturage de type `"Sortie autoroute"`.

8. Écrire une requête SQL permettant de compter le nombre de lieux de covoiturage se trouvant dans le département "JURA".

9. La commune de code INSEE "40714" vient d'être intégrée à la commune voisine "40146". En conséquence, il faut mettre à jour la base de données.

a) La requête suivante a été saisie. Une erreur a été renvoyée par le logiciel. Expliquer pourquoi.

```
1 DELETE FROM COMMUNE
2 WHERE code_insee = "40714" ;
```

b) Les informations liées à la commune de code INSEE "40714" devront désormais être rattachées à la commune de code INSEE "40146".

Écrire une requête permettant de mettre à jour la table `LIEU`.