Exercice 3 (8 points)

Cet exercice porte sur les types de données construits (listes et dictionnaires), sur les arbres binaires de recherche et sur les bases de données

Un ballon-sonde (utilisé en météorologie et en astronautique) est un ballon à gaz libre utilisé pour faire des mesures locales dans l'atmosphère. Chaque mesure réalisée est horodatée et géolocalisée par GPS.

À la fin de leur vol, les radiosondes redescendent vers la terre, se posent « n'importe où » et sont « abandonnées » à leur sort. C'est pourquoi vous pouvez en trouver. Diane Delstar est une radio amatrice appartenant à un club de passionnés. Le club récupère les ballons équipés de leur radiosonde et répertorie toutes les infos de leur atterrissage sur la terre ferme dans une base de données consultable sur un site web (source : https://sondehub.org). Le programme, écrit par Diane, lui permet de récupérer les données relevées par les sondes. La variable `enregistrement` fait référence à un dictionnaire qui contient un ensemble de données pour une sonde.

Exemple :

```
enregistrement={
    'num_serie': 623, 'altitude': 1150.0,
    'datetime': '2024-06-27T23:36:01.000000Z',
    'latitude': 38.38825, 'longitude': 27.09004
}
```

1. Écrire la ligne de code qui, dans la console, permet d'afficher la valeur `38.38825` de la variable `enregistrement`.

Les valeurs dont la clé est `'datetime'` sont des chaînes de caractères.

Exemple : `'2024-06-27T23:36:01.000000Z'` est une chaîne de caractères qui comprend la date au format année-mois-jour(`'2024-06-27'`), suivie de l'indicatif (`'T'`), puis de l'heure (`'23:36:01.000000'`) et enfin d'un indicatif de fuseau horaire (`'Z'`).

Diane écrit la fonction `nettoyage_datetime` qui prend en paramètre une chaîne de caractères correspondant à la clé `'datetime'` au format ISO 8601.

```
1 def nettoyage_datetime(chaine):
2     date=''
3     horaire=''
4     for i in range(10):
5         date=date+chaine[i]
6     for i in range(8):
7         horaire=horaire+chaine[i+11]
8     return date,horaire
```

2. Écrire et expliquer ce que renvoie l'appel de la fonction ci-dessous:

```
nettoyage_datetime ('2024-06-27T23:36:01.000000Z')
```

Tous les enregistrements de données de plusieurs radiosondes sont stockés dans la variable `frames` de type `list` dont un extrait est donné ci-dessous :

```
 1 frames=[
 2     {'num_serie': 623, 'altitude': 620.1,
 3      'datetime': '2024-07-01T01:58:55.000Z',
 4      'latitude': 43.20223, 'longitude': -72.05708},
 5     {'num_serie': 500, 'altitude': 6375.75,
 6      'datetime': '2024-07-01T23:35:32.000Z',
 7      'latitude': -20.8759, 'longitude': 55.58805},
 8     {'num_serie': 623, 'altitude': 622.6,
 9      'datetime': '2024-07-01T01:59:01.000Z',
10      'latitude': 43.20224, 'longitude': -72.05711},
11     {'num_serie': 700, 'altitude': 60000.0,
12      'datetime': '2024-07-01T11:51:23.000000Z',
13      'latitude': 40.87873, 'longitude': 29.09587}
14 ]
```

3. Donner les valeurs renvoyées par les appels `len(frames)` et `len(frames[1])`.

Une radiosonde peut monter au maximum à une altitude de 35 000 mètres et, bien sûr, ne peut atteindre une altitude de valeur négative. La fonction `detecter_anomalie` prend en paramètre un élément d'une liste analogue à `frames` et renvoie un booléen indiquant si la clé 'altitude' est associée à une valeur négative ou à une valeur strictement supérieure à 35 000 mètres.

Exemple :

```
>>> detecter_anomalie({'num_serie': 700, 'altitude': 60000.0,
'datetime': '2024-07-01T11:51:23.000000Z', 'latitude':
40.87873, 'longitude': 29.09587})
True
```

4. Écrire, en langage Python, la fonction `detecter_anomalie`.

Diane a besoin d'obtenir la liste des numéros de séries des radiosondes. Elle écrit donc la fonction `liste_num_serie`.

Une sonde ne doit figurer qu'une seule fois dans la liste représentée par son numéro de série. Exemple d'appel de la fonction `liste_num_serie` avec comme paramètre la variable `frames` ci-dessus :

```
>>> liste_num_serie(frames)
[623, 500, 700]
```

On rappelle que l'opérateur `in` permet de réaliser un test d'appartenance.

Exemple :

```
>>> fruits=['Banane','Pomme','Poire']
>>> 'Banane' in fruits
True
>>> 'Orange' in fruits
False
```

5. Écrire, en langage Python, la fonction `liste_num_serie`.

En exploitant les enregistrements de données d'une même sonde, Diane décide d'en déterminer la distance totale parcourue. La fonction `distance_haversine` :

- prend en paramètre deux tuples de coordonnées (latitude, longitude) ;
- renvoie la distance, en kilomètres, entre les deux points.

Exemple : Les points `point1` et `point2` ont les coordonnées géographiques :

Point 1 : latitude1 = 46.815, longitude1 = 6.943

Point 2 : latitude2 = 47.049 et longitude2 = 7.52828

```
>>> point1 = (46.815,6.943)
>>> point2 = (47.049,7.52828)
>>> distance_haversine(point1, point2)
51.5
```

La fonction `distance_totale` prend en paramètre une liste de tuples de coordonnées (formées d'une latitude et d'une longitude) constituant le déplacement de la sonde. Elle additionne les distances entre les points successifs et renvoie la distance totale parcourue. Exemples :

```
>>> deplacement = [(46.8125, 6.9433), (46.91498, 7.23039),
(47.00661, 7.48054)]
>>> distance_totale(deplacement)
46.1
>>> deplacement = [(46.8125, 6.9433), (46.91498, 7.23039),
(47.00661, 7.48054), (47.2512,7.5201)]
>>> distance_totale(deplacement)
73.5
```

6. Compléter la fonction `distance_totale` :

```
1 def distance_totale(dep):
2     total = 0
3     for i in range(...):
4         ...
5     return total
```

Lorsque les sondes atterrissent à la fin de leur voyage, elles sont récupérées par les membres du club. Les dernières positions des sondes sont ensuite enregistrées pour être affichées sur le site Web du club. Un arbre binaire de recherche sera exploité par Diane pour gérer la récupération des sondes :

- Chaque sonde est associée à un arbre binaire de recherche (ou plus précisément à sa racine) ;
- les arbres binaires de recherche sont des instances de la classe `Sonde` ;
- les instances de la classe `Sonde` possèdent les attributs `num_serie`, `latitude`, `longitude` et `date` qui décrivent la sonde ainsi que des attributs `gauche` et `droite` qui font références aux deux sous-arbres reliés à la racine ;
- l'arbre binaire de recherche vide est une instance de `Sonde` dont tous les attributs valent `None` ;
- la méthode `est_vide` renvoie un booléen qui permet de savoir si l'arbre représenté est vide ;
- si le sous-arbre gauche n'est pas vide, alors son attribut `num_serie` est strictement inférieur à l'attribut `num_serie` de la racine ;
- si le sous-arbre droit n'est pas vide, alors son attribut `num_serie` est strictement supérieur à l'attribut `num_serie` de la racine.

Les premières lignes de code de la classe `Sonde` sont les suivantes :

```
 1 class Sonde:
 2     def __init__(self, num_serie, latitude, longitude, date,
 3                        gauche, droit):
 4         self.num_serie = num_serie
 5         self.latitude = latitude
 6         self.longitude = longitude
 7         self.date = date
 8         self.gauche = gauche
 9         self.droit = droit
10
11     def est_vide(self):
12         return self.num_serie is None
13
```

7. Écrire la ligne de code qui permet de créer une instance `sonde623` de la classe `Sonde` à l'aide des paramètres suivants:

num : 623; lati : 38.38825; long : 27.09004; date : ‘2024-06-27’

Ci-dessous, un extrait de la représentation graphique de l’arbre binaire de recherche. Les numéros de série (`num_serie`) de chaque sonde y sont indiqués.

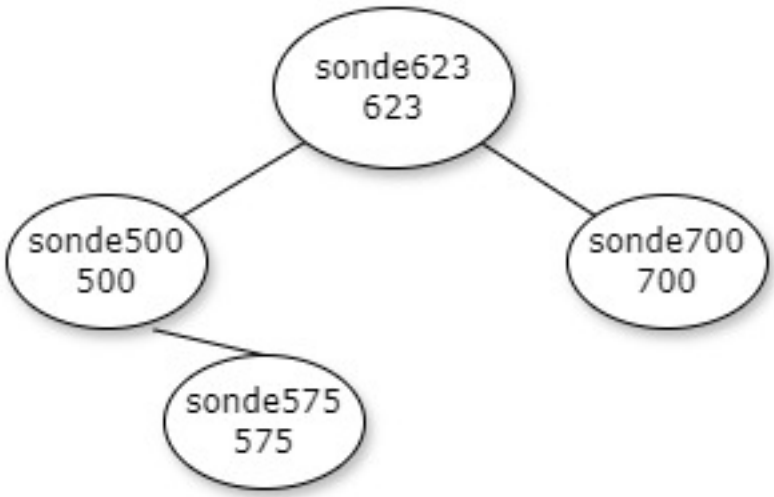

Figure 2. Arbre binaire de recherche

8. Recopier et compléter la représentation graphique, ci-dessus, en y ajoutant successivement et dans cet ordre les nœuds suivants :
   - sonde portant le nom `sonde900` et le numéro de série 900 ;
   - sonde portant le nom `sonde650` et le numéro de série 650 ;
   - sonde portant le nom `sonde300` et le numéro de série 300.
9. L’arbre peut être parcouru selon l’ordre de parcours *préfixe*, *infixe*, *suffixe* ou *en largeur d’abord*. Indiquer celui qui permet de lire les numéros des sondes dans l’ordre croissant des numéros de série.
10. Compléter la méthode `rechercher` (lignes 1, 3, 4 et 7), ci-dessous, qui prend en paramètre un numéro de série (`numero`) et qui renvoie l’instance de la classe `Sonde` correspondant au numéro de série ou `None` si la sonde n’a pas été référencée dans l’arbre.

```
 1     def rechercher(..., numero):
 2         if self.est_vide():
 3             return ...
 4         if numero == ...:
 5             return self
 6         elif numero < self.num_serie:
 7             return self.....rechercher(numero)
 8         else:
 9             return self.droit.rechercher(numero)
10
```

11. Expliquer pourquoi la méthode `rechercher` est une méthode récursive.

Finalement, Diane, après avoir traité toutes les données issues des enregistrements des sondes, décide pour son site web d'exploiter une base de données SQL. Son objectif est de référencer toutes les radiosondes récupérées par les membres du club. La base de données simplifiée a pour nom : `InventaireSondesRetrouvees`

On pourra utiliser les clauses du langage SQL pour :

- construire des requêtes d'interrogation à l'aide de `SELECT`, `FROM`, `WHERE` (avec les opérateurs logiques `AND`, `OR`), `JOIN ... ON` ;
- construire des requêtes d'insertion et de mise à jour à l'aide de `UPDATE`, `INSERT`, `DELETE` ;
- affiner les recherches à l'aide de `DISTINCT`, `ORDER BY`.

`InventaireSondesRetrouvees` est composée de 3 relations dont des extraits sont donnés ci-dessous:

abonne			
id_abonne	nom	prenom	email
32	'Détoile'	'Diane'	'D.Detoile@nsi.fr'
15	'Petit'	'Claire'	'P.Claire@nsi.fr'
24	'Girard'	'Antoine'	'A.Girard@nsi.fr'
16	'Lemoine'	'Martin'	'M.Lemoine@huit.fr'

sonde			
num_serie	modele	constructeur	date_lancement
623	'RS41'	'Vaisala'	'2024-07-10'
500	'M10'	'Météomodem'	'2024-07-05'
900	'M20'	'Météomodem'	'2024-07-17'
480	'RS41'	'Vaisala'	'2024-06-20'
810	'WxR'	'Weathex'	'2024-06-05'

info_recuperation					
ref	num_serie	id_abonne	date_recup	latitude	longitude

info_recuperation					
10	900	15	‘2024-08-05’	47.999	2.549
11	500	24	‘2024-07-06’	47.159	3.151
12	623	15	‘2024-07-18’	47.257	11.974
13	810	32	‘2024-06-10’	44.374	22.448

12. Citer, en justifiant votre réponse, un attribut pouvant servir de clé primaire dans la relation `abonne`.

13. Citer les attributs qui sont des clés étrangères dans la relation `infos_recuperation`.

14. Écrire l’affichage que provoque l’exécution de la requête SQL ci-dessous sur les extraits de la base de données `InventaireSondesRetrouvees`

```
SELECT nom,prenom FROM Abonnes WHERE id_abonne>20
```

La sonde de numéro de série 480 a été retrouvée par Antoine Girard le 2024-07-10 à la latitude 47.230 et la longitude 12.244.

15. Écrire la requête SQL qui permet d’ajouter les données de récupération de la sonde 480 dans la relation `InfosRecuperation`. L’attribut `ref` doit prendre la valeur 14.

On souhaite afficher plusieurs renseignements sur les radiosondes récupérées.

16. Écrire la requête SQL qui pour chaque sonde affiche le numéro de série, le nom de l’abonné, et la date de récupération.