Centres étrangers – 2025 – sujet1 - Correction

Exercice 3 (8 points)

Question 1

```
enregistrement['latitude']
```

💬 On accède à la latitude par sa clé dans le dictionnaire.

Question 2

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

💬 La fonction renvoie le tuple `('2024-06-27', '23:36:01')`, en extrayant les 10 premiers caractères pour la date, et les 8 suivants après le 'T' pour l'heure.

Question 3

```
len(frames) → 4
len(frames[1]) → 5
```

💬 La liste `frames` contient 4 dictionnaires, et chacun a 5 clés (num_serie, altitude, datetime, latitude, longitude).

Question 4

```
def detecter_anomalie(enregistrement):
    return enregistrement['altitude'] < 0 or enregistrement['altitude'] > 35000
```

💬 La fonction vérifie si l'altitude est anormale: soit négative, soit supérieure à 35 000 mètres.

Question 5

```
def liste_num_serie(frames):
    liste = []
    for elt in frames:
        if elt['num_serie'] not in liste:
            liste.append(elt['num_serie'])
    return liste
```

💬 On construit une liste sans doublons à partir des numéros de série des sondes.

Question 6

```
1 def distance_totale(dep):
2     total = 0
3     for i in range(len(dep) - 1):
4         total += distance_haversine(dep[i], dep[i + 1])
5     return total
```

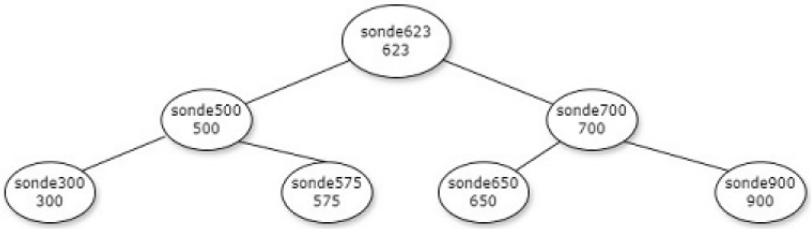
💬 La fonction additionne les distances entre chaque paire successive de points.

Question 7

```
sonde623 = Sonde(623, 38.38825, 27.09004, '2024-06-27',None, None)
```

💬 On instancie un objet `Sonde` avec les données fournies.

Question 8



Question 9

`Parcours infixe (in-order)`

💬 Le parcours infixe permet de lire les éléments d'un arbre binaire de recherche dans l'ordre croissant : 300->500->575->623->650->700->900

Question 10

```
def rechercher(self, numero):
    if self.est_vide():
        return None
    if numero ==self.num_serie:
        return self
    elif numero < self.num_serie:
        return self.gauche.rechercher(numero)
    else:
        return self.droit.rechercher(numero)
```

💬 On compare le numéro à celui de la racine, et on cherche récursivement dans le bon sous-arbre.

Question 11

La méthode `rechercher` est récursive car elle s'appelle elle-même sur les sous-arbres gauche ou droit.

> 💬 Elle divise le problème en sous-problèmes similaires (arbre plus petit).

Question 12

L'attribut `id_abonne` peut servir de clé primaire car il identifie de façon unique chaque abonné.

> 💬 Chaque abonné possède un identifiant unique.

Question 13

`num_serie` et `id_abonne` sont des clés étrangères dans la table `info_recuperation`.

> 💬 Elles font référence respectivement aux tables `sonde` et `abonne`.

Question 14

Résultat :

```
Détoile Diane
Girard Antoine
```

> 💬 Ce sont les abonnés dont l'identifiant est strictement supérieur à 20.

Question 15

```
INSERT INTO InfosRecuperation VALUES (14, 480, 24, '2024-07-10', 47.230, 12.244);
```

> 💬 On insère une ligne complète dans la table avec les données indiquées.

Question 16

```
SELECT sonde.num_serie, infosRecuperation.date_recup
FROM sonde
JOIN info_recuperation  ON sonde.num_serie = infosRecuperation.num_serie
JOIN abonne ON infosRecuperation.id_abonne = abonne.id_abonne;
```

> 💬 La requête combine les trois tables avec une double jointure via les clés primaires/étrangères pour obtenir les informations souhaitées. On pouvait faire les jointure dans un autre ordre.