Exercice 3 (8 points)

Cet exercice porte sur la programmation objet en langage Python, les graphes et les bases de données.

Partie A

Nous avons représenté un parc d'attractions par un graphe. Les sommets de ce graphe sont des attractions. Chaque attraction a une durée (en minutes). Les arêtes de ce graphe représentent la durée (en minutes) pour aller d'une attraction à une autre. Dans ce parc d'attractions, toutes les attractions ont des noms uniques.

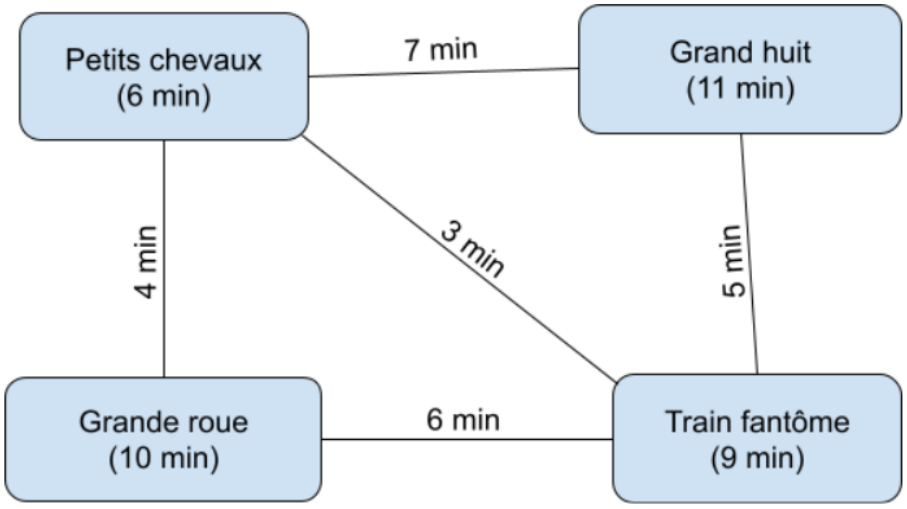

Figure 1. Parc d'attractions

Les attractions sont représentées par des objets de la classe `Attraction` dont le code est donné ci-dessous.

```
1 class Attraction:
2     def __init__(self, nom, duree):
3         self.nom = nom
4         self.duree = duree
5         self.voisines = []
```

Le graphe précédent peut être représenté, d'une façon incomplète, en langage Python ainsi :

```
1 a1 = Attraction("Grand huit", 11)
2 a2 = Attraction("Petits chevaux", 6)
3 a3 = Attraction("Train fantôme", 9)
4 a4 = Attraction("Grande roue", 10)
5 a1.voisines = [(a2,7), (a3,5)]
6 a2.voisines = [(a1,7), (a3,3), (a4,4)]
7 a3.voisines = [(a1,5), (a2,3), (a4,6)]
8 a4.voisines = ...
```

Par mesure de sécurité, les gérants du parc d'attractions ont ralenti la vitesse de rotation de la grande roue. Sa durée est maintenant de 12 minutes.

1. En considérant la modélisation du parc d'attractions ci-dessus, écrire une ligne de code permettant de faire cette modification.

2. Donner et expliquer la valeur de l'expression `a2.voisines[2][1]`.

3. Expliquer la ligne 7 de ce code.

4. Recopier et compléter la ligne 8 de ce code.

5. Expliquer pourquoi cette modélisation du parc d'attractions est réalisée avec un graphe non orienté.

Pour faciliter la gestion du parc d'attractions, ses dirigeants proposent aux usagers des *balades* dans le parc. Une *balade* est un chemin du graphe représentant le parc d'attractions. Les usagers choisissant une balade doivent faire les attractions dans l'ordre de parcours du chemin. La durée d'une balade est la durée totale pour parcourir la balade, c'est-à-dire la somme des durées de ses sommets et de ses arêtes.

En langage Python, on modélise une balade par un tableau de sommets du graphe. Par exemple, le tableau `[a1, a2, a3, a1, a3]` est une balade du graphe précédent.

6. Calculer la durée en minutes de la balade représentée par le tableau `[a1, a2, a3]` et expliquer le calcul effectué en une phrase.

7. Expliquer pourquoi le tableau `[a2, a1, a4, a3]` n'est pas une balade du parc d'attractions.

On considère qu'il est possible de comparer des objets de la classe `Attraction` entre eux à l'aide de l'opérateur ==.

8. Écrire une fonction `sont_voisines` qui prend comme arguments deux attractions de la classe `Attraction` et qui renvoie `True` si ces deux attractions sont voisines et `False` sinon.

9. Écrire une fonction `est_balade` qui prend comme argument un tableau de sommets de type `Attraction` et qui renvoie `True` si ce tableau est une balade et `False` sinon.

Les gérants du parc d'attractions souhaitent automatiser la création de balades, de telle sorte que désormais chaque attraction apparaisse au maximum une fois dans la balade. Pour cela, ils proposent de faire un parcours de graphe à partir d'une des attractions du parc, avec un tableau pouvant représenter une balade en paramètre. Pendant le parcours du graphe, si une attraction est atteignable depuis la dernière attraction placée dans la balade, alors elle est ajoutée à la balade.

Le code suivant est proposé :

```
1 def parcours(attr, deja_vues, balade, nb):
2     if not attr.nom in deja_vues:
3         deja_vues[attr.nom] = True
4         if nb == 0 or sont_voisines(attr, balade[nb-1]):
5             balade[nb] = attr
6             nb = nb + 1
7         for voisine in attr.voisines:
8             nb = parcours(voisine[0], deja_vues, balade, nb)
9     return nb
```

10. Donner le type de parcours effectué par la fonction `parcours` ci-dessus.

Chaque attraction apparaît au maximum une fois dans une balade. Ainsi, un tableau représentant la balade peut être initialisé à `[None, None, None, None]` si le parc d'attractions n'a que 4 attractions. Si à l'issue du parcours, les attractions n'ont pas été toutes utilisées, il sera possible de créer une copie partielle du tableau contenant uniquement les éléments différents de `None`.

11. Déterminer ce que contient le tableau `balade` après l'exécution du code ci-dessous, en utilisant les variables `a1`, `a2`, `a3` et `a4` :

```
>>> balade = [ None for _ in range(4) ]
>>> parcours(a4, {}, balade, 0)
```

12. Déterminer maintenant ce que contient le tableau `tableau` après l'exécution du code ci-dessous :

```
>>> a2.voisines = [(a1,7), (a3,3)]
>>> a4.voisines = [(a3,6)]
>>> tableau = [ None for _ in range(4) ]
>>> parcours(a3, {}, tableau, 0)
```

13. Déduire des appels à la fonction `parcours` le nom de la structure de données utilisée pour la variable `deja_vues` et expliquer en une phrase son rôle.

Partie B

Dans cette partie de l'exercice, 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` .

Les visiteurs qui sont d'accord reçoivent un bracelet magnétique à l'entrée du parc d'attractions. Ce bracelet permet de les identifier et de les prendre en photos à différents points clés des attractions. Ces photos leur sont ensuite proposées à la vente. Le système est calibré pour ne pas prendre de photos des personnes ne le souhaitant pas. Les données personnelles associées sont stockées en France et les utilisateurs disposent, conformément à la loi, d'un droit de consultation, de retrait et de rectification.

Pour gérer ces photos et leur vente, le parc d'attractions utilise une base de données. La Figure 2 présente une représentation des trois relations de cette base dont les clés primaires sont les attributs soulignés, appelés `id` dans chaque relation et dont les clés étrangères sont précédées d'un caractère #. Pour chaque attribut est indiqué le nom de l'attribut, et son type après le symbole `:` le type `int` représente des entiers, le type `text` des chaînes de caractères et le type `float` des nombres flottants.

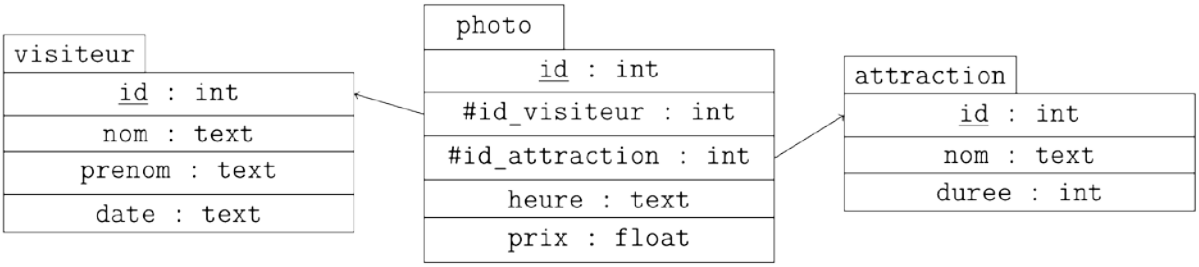

Figure 2. Représentation des relations de la base de données utilisée.

L'attribut `date` de la relation `visiteur` est une chaîne de caractères au format `'année-mois-jour'`, l'année étant écrite sur 4 chiffres, le mois sur 2 chiffres et le jour sur 2 chiffres. Par exemple, le 1er février 2025 sera représenté par la chaîne de caractères `'2025-02-01'`. L'attribut `heure` de la relation `photo` est une chaîne de caractères au format `'heures:minutes'`, en utilisant 2 chiffres pour les heures et 2 chiffres pour les minutes. Par exemple, l'heure 5 heures 49 minutes sera représentée par `'05:49'`.

14. Expliquer ce qu'est une clé primaire, puis ce qu'est une clé étrangère.

15. Écrire une requête en langage SQL qui permet d'obtenir les noms et prénoms des visiteurs présents le 11 janvier 2025 sans doublons.

En langage SQL, les opérateurs de comparaison classiques peuvent être utilisés pour comparer des chaînes de caractères entre elles. Par exemple, la condition '2025-01-01' > '2024-01-01' serait évaluée à vrai.

La fonction d'agrégation `SUM` permet de renvoyer la somme des valeurs d'un attribut. Par exemple, le code ci-dessous permet de déterminer le prix total des photos de la relation `photo` :

```
SELECT SUM(prix)
FROM photo;
```

Un visiteur, Alan TURING, est venu plusieurs fois dans le parc d'attractions en 2024. À chaque visite, il a acheté toutes les photos proposées.

16. Écrire une requête en langage SQL qui permet d'obtenir la somme totale de ce qu'Alan TURING a payé pour des photos au parc d'attractions en 2024.

Suite à un problème technique, les gérants ont utilisé la requête suivante :

```
SELECT visiteur.nom, prenom
FROM visiteur JOIN photo
  ON visiteur.id = photo.id_visiteur
JOIN attraction
  ON attraction.id = photo.id_attraction
WHERE attraction.nom = 'Grande roue'
  AND heure = '12:34'
  AND date = '2024-07-26';
```

17. Expliquer ce qu'ils voulaient savoir.

Les gérants du parc d'attractions décident d'étoffer leur offre d'achat de photos en proposant pour un cliché plusieurs formats et supports (A5, A6, poster, porte-clé, …).

18. Proposer des modifications de la base de données précédente pour qu'elle puisse prendre en charge cette nouvelle offre.