TD: implémentation d'un graphe:

Exercice 1: implémentation d'un graphe (liste des successeurs): On s'interesse au programme suivant:

```
graphe = {
    "A": ["B","E"],
    "B": ["A","C","E"],
    "C": ["B"],
    "D": [],
    "E": ["A","B"],
    }

def deg(g,s):
    if g.get(s) is not None:
        return len(g[s])
```

graphe1.py

1. Quel type est utilisé pour implémenter le graphe dans ce programme?
2. Représenter le graphe.
3. Expliquer le fonctionnement de deg() et tester cette fonction.

Exercice 2: implémentation d'un graphe (matrice): On s'interesse au programme suivant:

```
G = [
     [0,1,0,0,1]
     [1,0,1,0,1]
     [0,1,0,0,0]
     [0,0,0,0,0]
     [1,1,0,0,0]
     ]
```

graphe2.py

1. Quel type est utilisé pour implémenter le graphe dans ce programme?
2. Représenter le graphe.

Exercice 3: implémentation d'un graphe orienté (classes): On s'interesse au programme suivant:

```python
class Arete:
    def __init__(self,src,dest):
        self.src = src
        self.dest = dest

    def get_source(self):
        return self.src

    def get_destination(self):
        return self.dest

    def __str__(self):
        return str(self.src) + "->" + str(self.dest)


class Graphe_Or:
    def __init__(self):
        self.sommets = []
        self.aretes = {}

    def ajoute_sommet(self,s):
        if s not in self.sommets:
            self.sommets.append(s)  #on ajoute dans la liste
            self.aretes[s] = []     #on ajoute dans le dictionnaire

    def ajoute_arete(self,a):
        src = a.get_source()        #on utilise la classe Arete
        dest = a.get_destination()
        if src in self.sommets and dest in self.sommets:
            if not dest in self.aretes[src]:
                self.aretes[src].append(dest)   #on ajoute la destination liee a la source via l'arete

    def get_sommets(self):
        return self.sommets

    def __str__(self):
        res = ""
        for k in self.aretes:
            for d in self.aretes[k]:
                res = res + str(k) + "->" + str(d) + "\n"
        return res[:-1]
```

graphe3.py

1. Expliquer avec précision les méthodes et attributs utilisés dans les classes Arete et Graphe_Or.
2. Implémenter le graphe suivant à l'aide des classes de ce programme.

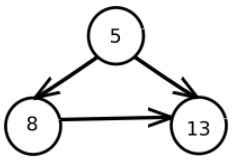

Figure 1: Graphe orienté

Exercice 4: implémentation d'un graphe non orienté (classes): On ajoute cette classe au programme de l'exercice 3:

```
class Graphe(Graphe_Or):
    def ajoute_arete(self, a):
        Graphe_Or.ajoute_arete(self, a)
        retour = Arete(a.get_destination(),a.get_source())
        Graphe_Or.ajoute_arete(self, retour)
```

graphe4.py

1. Expliquer brièvement le fonctionnement de cette nouvelle classe.
2. Implémenter le graphe suivant à l'aide des classes de ce programme.

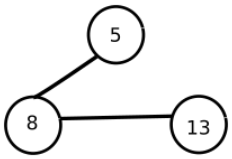

Figure 2: Graphe non orienté

Exercice 5: matrices d'adjacence: Tracer les graphes associés aux matrices d'adjacence suivante:

1. $M_1 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$
2. $M_2 = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$
3. $M_3 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}$

Exercice 6: matrices d'adjacence: Ecrire la matrice d'adjacence du graphe suivant:

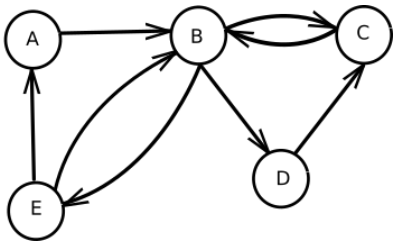

Figure 3: Graphe orienté

Exercice 7: Soit le graphe suivant:

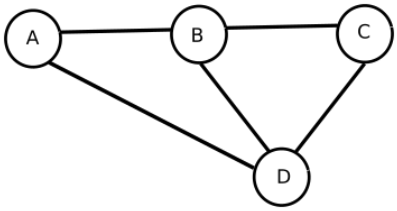

Figure 4: Graphe non orienté

1. Écrire un dictionnare Python où les clés sont les sommets pour représenter ce graphe.
2. Écrire une fonction sommets() qui prend en paramètres un sommet s et un graphe g, sous la forme d'un dictionnaire et qui renvoie la liste des sommets liés par une arête au sommet s.