DS : graphes (30 mn – Mr Gras)
Memento Python autorisé – calculatrice interdite

Notation : le barème est indiqué pour chaque question.

Exercice 1 :

1.1 (2 pt) Dessiner un graphe non orienté à 4 sommets et 5 arêtes.
1.2 (2 pt) Dessiner un graphe orienté à 5 sommets et 7 arêtes

Exercice 2 : On donne un graphe non orienté en figure 1.

2.1 (1 pt) Quel est l'ordre de ce graphe.
2.2 (1 pt) Quel est le degré du sommet **C**.
2.2 (1 pt) Quel est le degré du sommet **D**.
1.3 (3 pts) Un graphe d'ordre n > 1 est dit complet si tous ses sommets sont deux à deux adjacents. Sur votre copie, recopier en noir le graphe figure 1 et le compléter avec un nombre minimal d'arêtes en rouge pour qu'il soit complet.

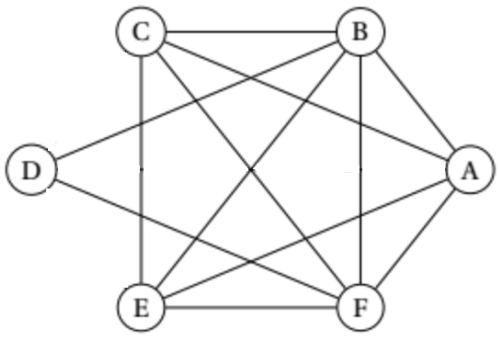
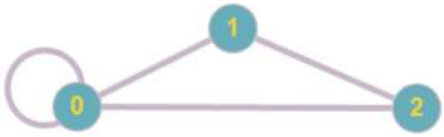
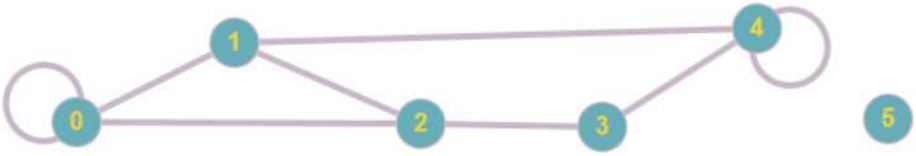


Figure 1 : graphe d'étude

Exercice 3 : Codage d'un graphe : on désire développer des fonctions de création de graphes non orientés. On travaille sur des graphes codés par liste d'adjacence. On rafraichît rapidement la mémoire des étudiants en donnant les exemples ci-dessous montrant les correspondances entre liste d'adjacence et graphe.

G0 = [ ] Graphe vide (sans sommet ni arête)	G1 = [[0,0] , [1,0]] 	G2 = [[0,0,1,2] ,[1,0,2] , [2,0,1]] 
G3 = [[0,0,1,2] , [1,0,2,4] , [2,0,1,3] , [3,2,4] , [4,1,3,4] , [5]] 		

On impose les règles suivantes :
1- chaque sommet est repéré par un entier naturel,
2- pour tout entier naturel k, un graphe de k sommets aura ses sommets repérés de 0 à k-1,
3- dans la liste d'adjacence, les voisins d'un sommet avec boucle éventuelle seront ordonnés par ordre croissant.

Les graphes G0, G1, G2 et G3 donnés en exemple respectent ces règles.

3.1 (3 pts) Proposer un script Python de la fonction « plus_ordre_graphe » permettent d'augmenter l'ordre d'un graphe de +1.

Par exemple, on devra avoir :

```
>>> G1 = [ [0,0] , [1,0] ]
>>> G1 = plus_ordre_graphe(G1)
>>> print(G1)
[[0, 0], [1, 0], [2]]
>>>
```

3.2 On propose ci-après, en page 3, le script Python de la fonction « plus_arete_graphe ».
On tape l'instruction G4 = plus_arete_graphe(G3). On rappelle que le graphe G3 est définit en début d'exercice 3.
Donner G4 (sous forme de sa liste d'adjacence) si :

3.2.1 (2 pts) S1 = 5 et S2 = 5.
3.2.2 (2 pts) S1 = 5 et S2 = 4.
3.2.3 (2 pts) S1 = 4 et S2 = 3.
3.2.4 (2 pts) S1 = 4 et S2 = 4.
3.2.5 (2 pts) S1 = 6 et S2 = 4.

```python

# définition de la fonction plus_arete_graphe (G)

def plus_arete_graphe (G : list) -> list :

    """Ajoute une arête au graphe G"""

    S1 = int(input("Saisir la 1 ère extrémité de l'arête (nombre entier) : "))
    S2 = int(input("Saisir la 2 nde extrémité de l'arête (nombre entier) : "))

    if S1>=len(G) or S2>=len(G) :
        print(" Graphe inchangé, saisie sommet incorrecte !")

    else :
        n11,n12,n22,n21 = 0,0,0,0

        if S1==S2  :
            for i in G[S1] :
                if S1 == i :
                    n11+=1

            if n11==1 :
                G[S1].append(S1)
                T=G[S1][1:len(G[S1])]
                T.sort()
                G[S1]=[G[S1][0]]+T

        else :
            for i in G[S1] :
                if S2 == i :
                    n12+=1
                if S1 == i :
                    n11+=1

            for i in G[S2] :
                if S1 == i :
                    n21+=1
                if S2 == i :
                    n22+=1

            if n12==0 and n21==0 :
                G[S1].append(S2)
                T=G[S1][1:len(G[S1])]
                T.sort()
                G[S1]=[G[S1][0]]+T

                G[S2].append(S1)
                T=G[S2][1:len(G[S2])]
                T.sort()
                G[S2]=[G[S2][0]]+T

            elif n11==2 and n22==1 :
                G[S2].append(S1)
                T=G[S2][1:len(G[S2])]
                T.sort()
                G[S2]=[G[S2][0]]+T

            elif n11==1 and n22==2 :
                G[S1].append(S2)
                T=G[S1][1:len(G[S1])]
                T.sort()
                G[S1]=[G[S1][0]]+T

    return G
```