II.5 Vendredi 10 avril ♥

On considère le graphe `G={0: [1,2,3], 1 : [0,2,4], 2 : [], 3 : [1,2], 4 : [2,3]}`

1–❑ Donner la matrice d'adjacence `M` de `G`.

Solution: Le graphe est orienté, la matrice n'est pas symétrique.

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

2–❑ Écrire une fonction `degre_mat(M:list,s:int)->int` qui prend en argument un graphe représenté par sa matrice d'adjacence `M` et un sommet `s`, et renvoie le degré du sommet `s`.

Solution: Il faut compter le nombre d'arcs qui partent de s.

```
def degre_mat(M,s):
    deg=0
    for j in range(len(M[s])):
        deg=deg+M[s][j] # ajoute 1 s'il y a un arc, 0 sinon
    return deg
```

3–❑ Écrire une fonction `voisin_mat(M:list,s:int)->list` qui prend en entrée une matrice d'adjacence et un sommet `s`, et renvoie la liste des voisins du sommet `s`.

Solution:

```
def voisin_mat(M,s):
    voi=[] # liste des voisins
    for j in range(len(M[s])):
        if s!=j and M[s][j]!=0: # il y a un arc entre s et j
            voi.append(j) # j est voisin de s
    return voi
```

4–❑ Écrire une fonction `Isole(G)` qui permet de renvoyer la liste des sommets isolés du graphe `G`.

Solution: Un graphe est isolé s'il n'y a aucun arc qui n'en part, à part éventuellement bouclé sur lui-même

```
def Isole(G):
    seuls=[] # liste des sommets isolés
    for i in range(len(M)):
        if voisin_mat(M,i)!=[]: # aucun voisin à i
            seuls.append(i)
    return seuls
```