# Exercices de cours

**Exercice 1 :** Ecrire une fonction **Python**, nommée `graph_orient`, qui :

- prend en argument la matrice d'adjacence `M` d'un graphe
- renvoie le booléen `True` si ce graphe est orienté et `False` sinon

On importera la bibliothèque `numpy` sous l'alias `np`.

**Exercice 2 :** Tracer les graphes pour chacune des matrices d'adjacence suivantes. On numérotera les sommets à partir de 1.

1. $M = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
2. $M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$
3. $M = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$
4. $M = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}$

**Exercice 3 :** Compléter la fonction **Python** suivante qui

- prend en argument la liste `L` des listes d'adjacence d'un graphe
- renvoie la matrice d'adjacence de ce graphe

```
1  import numpy as np
2  def ad_matrix_from_list(L):
3      nbS = __________
4      M = np.zeros([nbS,nbS])
5      for i in range(nbS):
6          nbV = _______________
7          for j in range(nbV):
8              M[i,L[i][j]] = _____
9      return M
```

**Exercice 4 :** Ecrire une fonction **Python**, nommée `List_deg`, qui :

- prend en argument la liste `L` des listes d'adjacence d'un graphe
- renvoie la liste des degrés des sommets de ce graphe

**Exercice 5 :** Ecrire une fonction **Python**, nommée `Graph_connexe`, qui

- prend en argument la matrice d'adjacence `M` d'un graphe
- renvoie le booléen `True` si ce graphe est connexe et `False` sinon

On importera la bibliothèque `numpy` sous l'alias `np` et la bibliothèque `numpy.linalg` sous l'alias `al`. On rappelle que la commande `al.matrix_power(M,k)` calcule la puissance $k^e$ de la matrice `M`.