Chapitre 8 : Listes et graphes

Dans tout ce chapitre, on utilisera : `import numpy as np`

1. Les listes

Une liste est un ensemble ordonné, qui peut contenir des éléments de toute nature (nombres, texte, …)

Définir une liste :
_ par extension : on donne la liste des éléments, entre crochets, séparés par des virgules :
Ex : `L = [1,2,3,4,5]` `L=['Lundi','Mardi','Mercredi']` `L = []` (liste vide)
_ par compréhension :
Ex : `L = [2*i+1 for i in range(5)]`

Accès aux éléments d'une liste
_ `L[i]` : élément en position i (attention le premier est en position 0)
_ `L[i:j]` : éléments entre les positions i et j – 1
_ `L[i:]` : éléments à partir de la position i
_ `L[:j]` : éléments de la position 0 à la position j-1
_ `L[-k]` : élément en position k à partir de la fin
_ `L[-k:]` : k derniers éléments

Longueur d'une liste : `len(L)`

Appartenance :
_ `x in L` : `True` si `x` appartient à `L`, `False` sinon
_ `L.count(x)` : nombre de fois où `x` apparait dans `L`

Opérations sur les listes :
_ Ajout d'un élément x à la fin de L : **`L.append(x)`**
_ suppression d'un élément : **`del L[k]`** (attention, les indices des termes suivants sont décalés)
_ **`L1 + L2`** : opération de **concaténation**, et non d'addition !
Si L1 = [1,2,3] et L2=[2,3,4] L1 + L2=[1,2,3,2,3,4]

Parcourir tous les éléments d'une liste, deux possibilités :
_ **`for i in range(len(L)) :`**
_ **`for x in L :`**

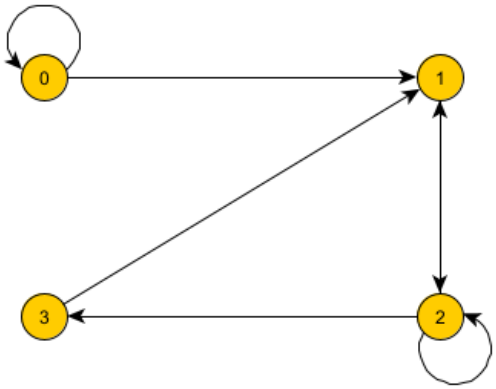
Exemple : Soit L une liste qui contient des entiers. Ecrire un script qui ajoute 1 à chaque élément de la liste

2. Python et graphes

Définir un graphe en Python :

On peut implémenter un graphe à l'aide de listes d'adjacence (rassemblées par exemple dans une liste) ou par sa matrice d'adjacence.

Ex : Le graphe :



a pour liste d'adjacence : L=

et a pour matrice d'adjacence : M=

Remarque :

Pour définir un graphe pondéré en Python, on peut le représenter par sa matrice des poids.

Exemple :

Matrice des poids :

N=