

Algorithmique Appliquée

BTS SIO SISR

Résolution de problèmes classiques



CHAMBRE DE COMMERCE
ET D'INDUSTRIE

1^{er} ACCÉLÉRATEUR DES ENTREPRISES



Loïc Yvonnet



Plan

- Listes chaînées
- Queue et FIFO
- Pile et LIFO
- Comparaison entre FIFO et LIFO
- Rappels sur la théorie des ensembles
- Rappels sur le calcul matriciel

Listes chaînées

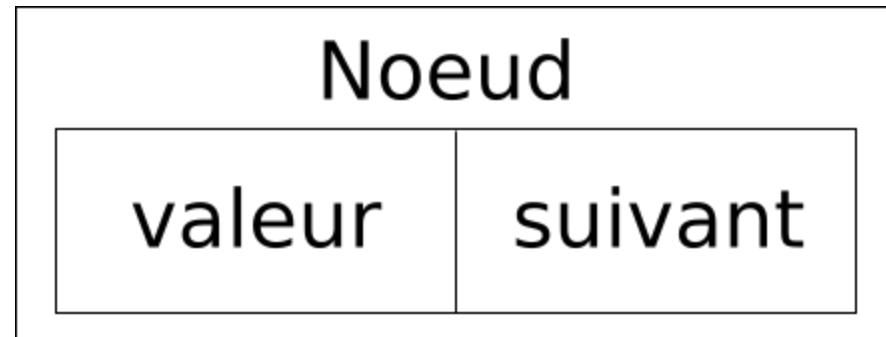
Introduction

- En Python, une `list` permet de rassembler un nombre variable d'éléments.
- Nous allons voir une manière d'implémenter ce type de liste.

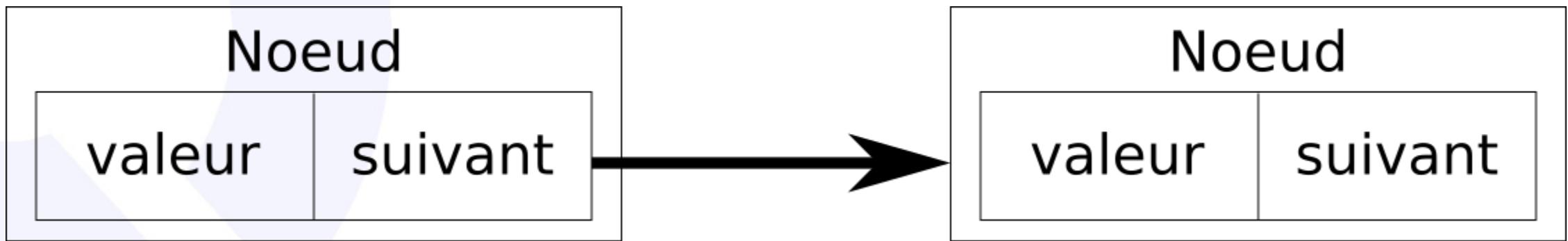
Notion de liste chaînée

- Une liste chaînée est une **structure de données récursive**.
- Une liste chaînée est composée de **noeuds**.
- Un noeud comporte 2 variables :
 - Une valeur,
 - Le noeud suivant.

Noeud d'une liste chaînée



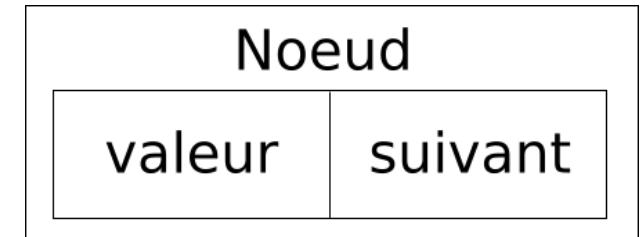
2 noeuds



Structure de données

```
from dataclasses import dataclass

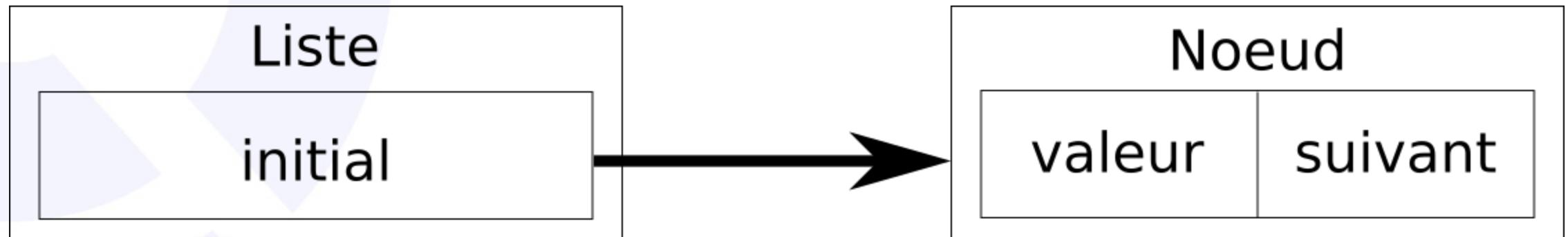
@dataclass
class Noeud:
    """Noeud de la liste chaine.
    La valeur peut etre n'importe quel objet
    Python valide.
    Le suivant doit avoir pour type Noeud
    ou None.
    """
    valeur = None
    suivant = None
```



Noeud de départ

- Comment identifier le **noeud de départ** de la liste ?
- On souhaite que chaque noeud ait la **même représentation**.
- On introduit un nouveau type, **Liste**, qui référence le noeud de départ.
- Une **Liste** n'a pas de valeur.

Noeud de départ identifié par la liste



Structure de données

```
from dataclasses import dataclass

@dataclass
class Liste:
    """Liste chaînée.

    Il s'agit simplement d'un point d'entrée
    vers le 1er noeud de la liste chaînée,
    nommé initial.

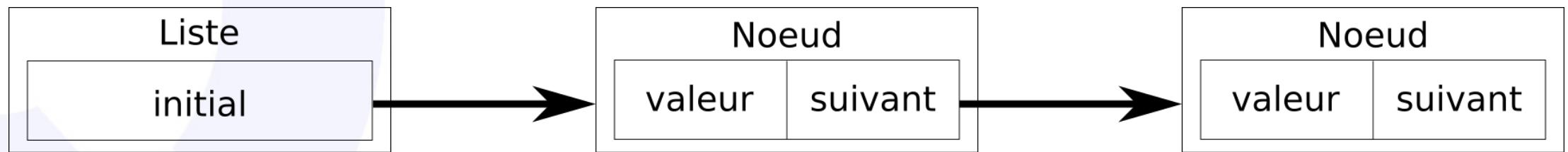
    La variable initial doit avoir pour type
    Noeud ou None.

    """
    initial = None
```

Liste

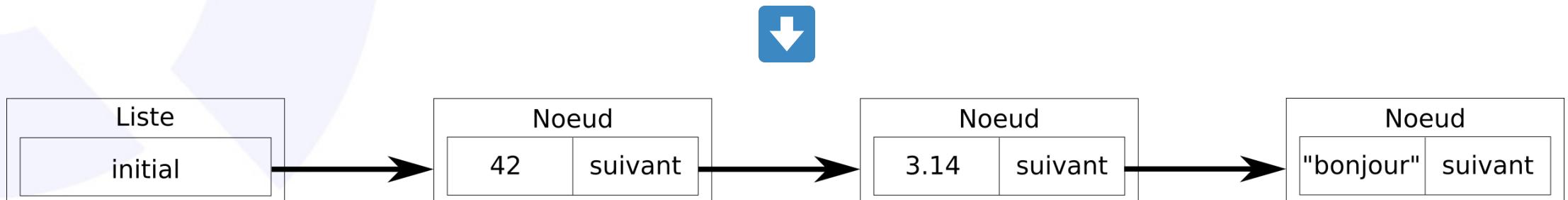
initial

Liste chaînée comportant 2 valeurs



Exemple avec 3 valeurs

```
liste_chaine = creer_liste_chaine(42, 3.14, "bonjour")
```



Dernier noeud

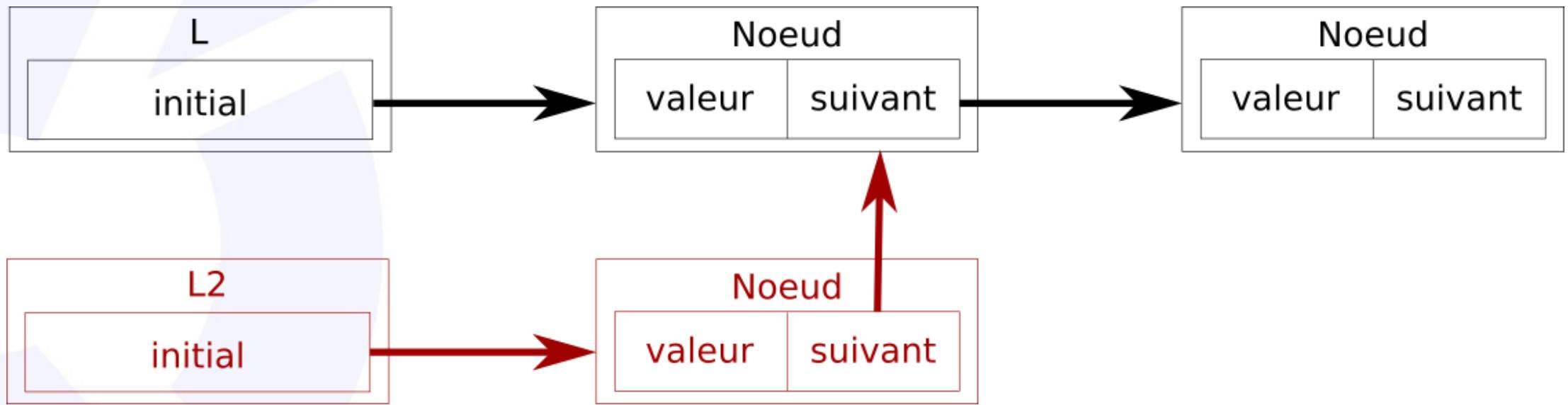
Le suivant du dernier noeud est None .

Parcours d'une liste chaînée

```
def parcours(liste_chainee, f):
    """Appelle f sur chaque valeur de la liste chaînée.

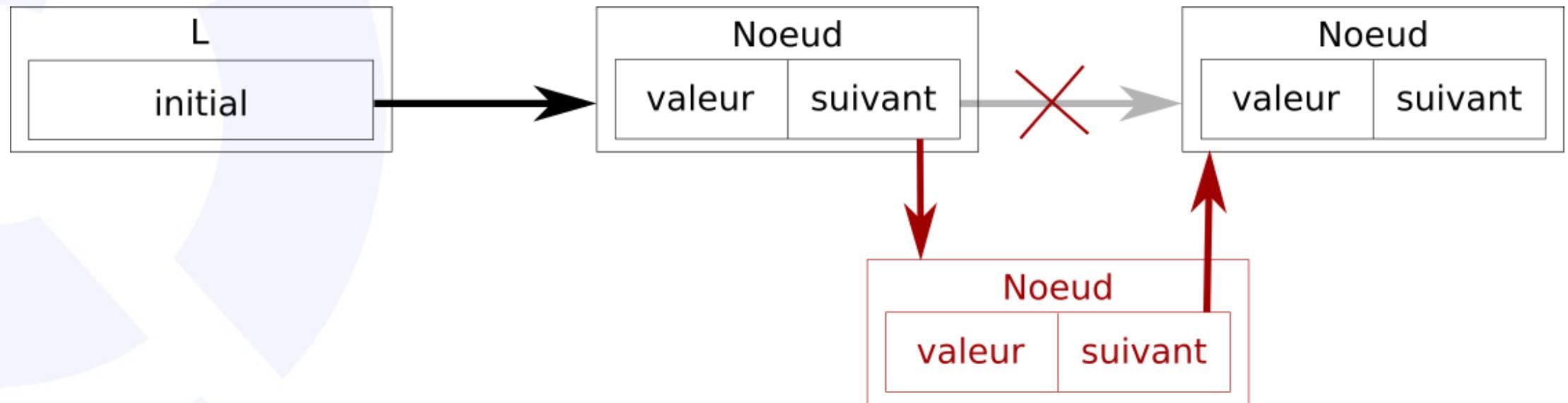
    liste_chainee - liste chaînée de type Liste.
    f - fonction prenant un argument.
    """
    noeud = liste_chainee.initial
    while noeud != None:
        f(noeud.valeur)          # appelle la fonction f
        noeud = noeud.suivant # passage au noeud suivant
```

Insertion au début



L'insertion au début est **non-destructive**.
Il est possible de construire des listes chaînées **immutables**.

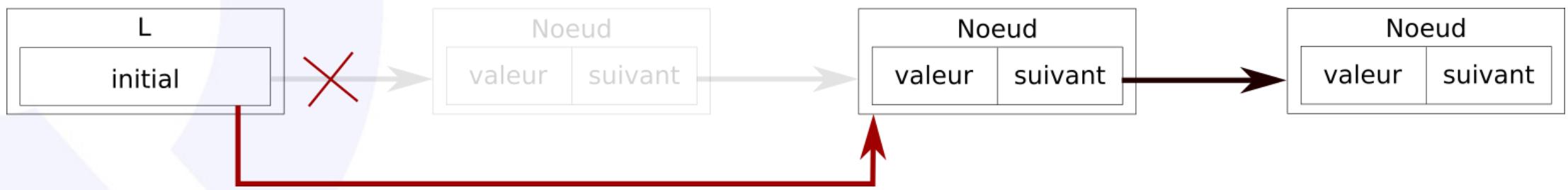
Insertion au milieu



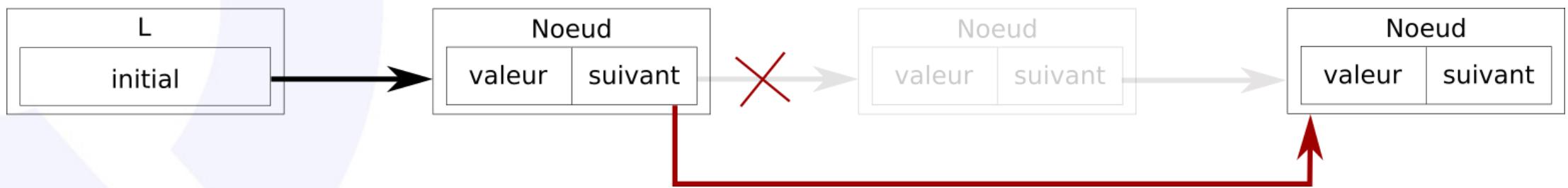
Insertion à la fin



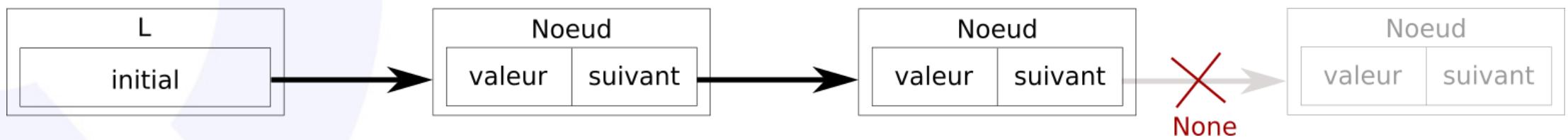
Suppression au début



Suppression au milieu



Suppression à la fin



Liste doubllement chaînée

```
from dataclasses import dataclass

@dataclass
class NoeudBidirectionnel:
    valeur = None
    suivant = None
    precedent = None    # Nouveau

@dataclass
class ListeBidirectionnelle:
    initial = None
    final = None        # Nouveau
```



Evaluation des listes chaînées

- **Avantages :**
 - Insertion rapide au début.
 - Insertion rapide à la fin pour une liste doublement chaînée.
- **Inconvénients :**
 - Pas d'indexation : il faut parcourir potentiellement tous les éléments pour en retrouver un.
 - Mémoire éparse : chaque noeud a sa propre adresse en mémoire.

Comparaison avec une list

- La Liste chaînée est un exercice intéressant pour comprendre comment une list peut être implémentée.
- La Liste chaînée introduite ici et dans le prochain TP a un but purement pédagogique.
- Dans du code industriel de production, utilisez une list .

TP : Manipulation d'une liste chaînée

TP : Manipulation d'une liste chaînée

[**Lien** vers le sujet de TP.](#)

Queue et FIFO

First-In, First-Out 

Notion de file d'attente

- Une **queue**, également nommée **file d'attente**, est une collection.
- Cette collection comporte 2 opérations principales :
 - Empiler un élément.
 - Dépiler le 1er élément empilé.
- Premier entré, premier sorti.

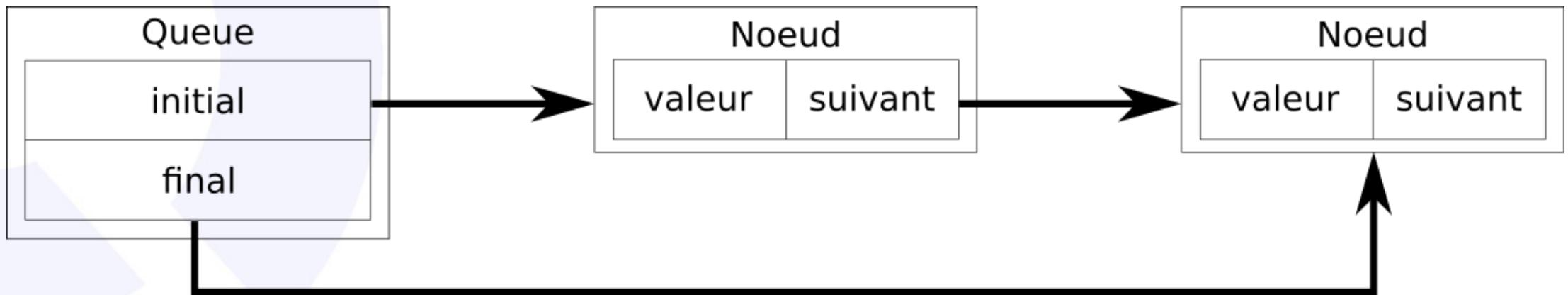
Métaphore



Principe



Exemple avec 2 éléments

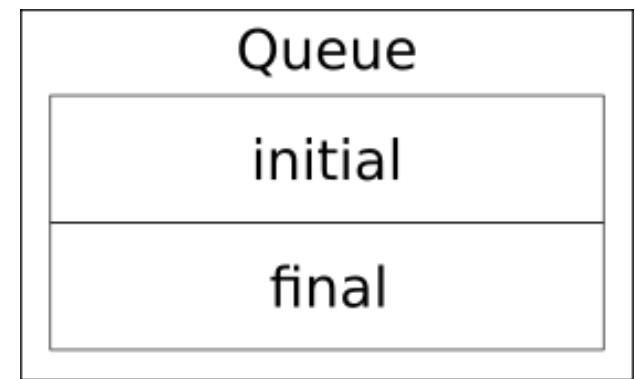


Structure de données

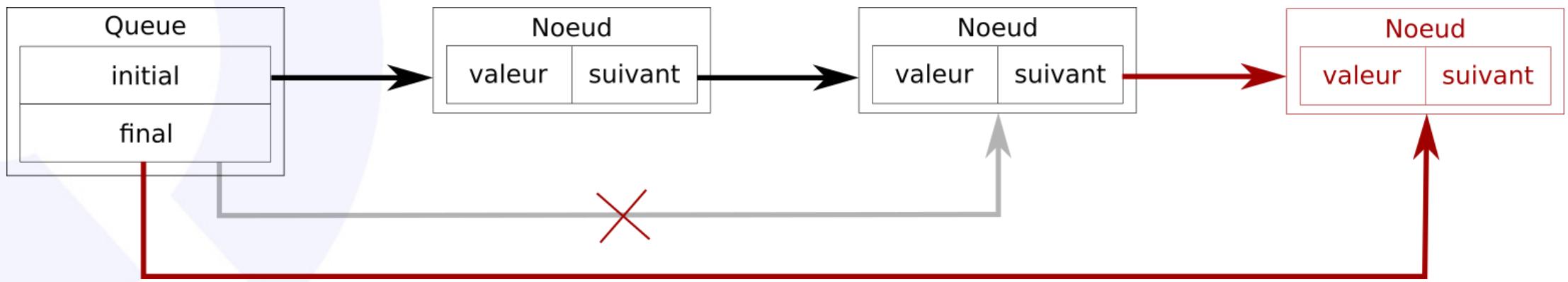
```
from dataclasses import dataclass

@dataclass
class Queue:
    """Queue utilisant une liste chaînée.

    initial - doit avoir pour type
              Noeud ou None.
    final - doit avoir pour type
            Noeud ou None.
    """
    initial = None
    final = None
```



Empile



Dépile



Utilisation d'une list

```
def empile(queue, element):
    """Empile l'élément dans la queue.

    queue - la queue à modifier.
    element - élément à empiler dans la queue.
    """
    queue.append(element)

def depile(queue):
    """Dépile le 1er élément de la queue.

    queue - la queue à modifier.
    Retourne le 1er élément de la queue.
    """
    return queue.pop(0)
```



Exemple

```
queue = [6, 3, 7]
empile(queue, 10)
print(queue)      # [6, 3, 7, 10]

valeur = depile(queue)
print(valeur)      # 6
print(queue)      # [3, 7, 10]
```

Utilisation de deque

```
from collections import deque

def empile(queue, element):
    """Empile l'élément dans la queue.

    queue - la queue à modifier.
    element - élément à empiler dans la queue.
    """
    queue.append(element)

def depile(queue):
    """Dépile le 1er élément de la queue.

    queue - la queue à modifier.
    Retourne le 1er élément de la queue.
    """
    return queue.popleft()
```

Exemple

```
queue = deque([6, 3, 7])
empile(queue, 10)
print(queue)          # deque([6, 3, 7, 10])

valeur = depile(queue)
print(valeur)          # 6
print(queue)          # deque([3, 7, 10])
```

Pile et LIFO

Stack & Last-In, First-Out 

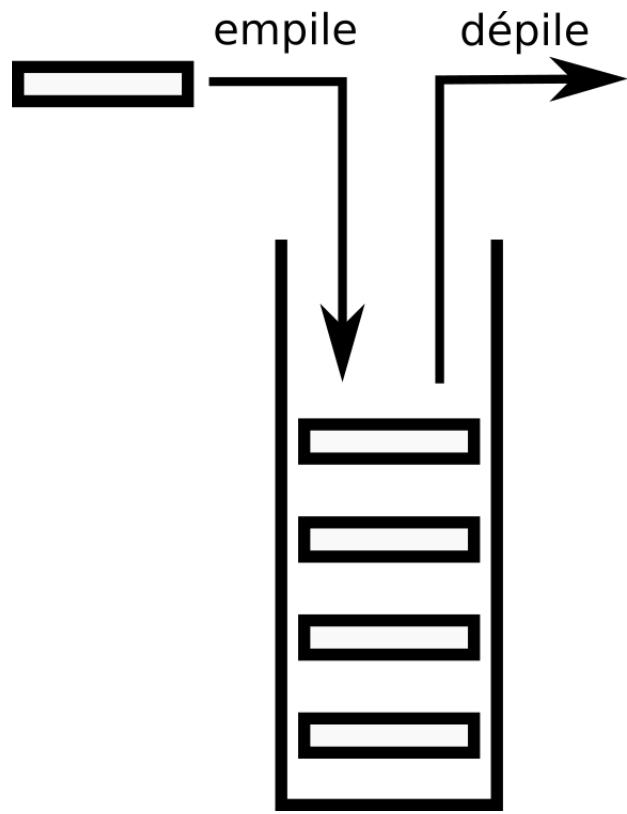
Notion de pile

- Une **pile** est une collection.
- Cette collection comporte 2 opérations principales :
 - Empiler un élément.
 - Dépiler le **dernier** élément empilé.
- Dernier entré, premier sorti.

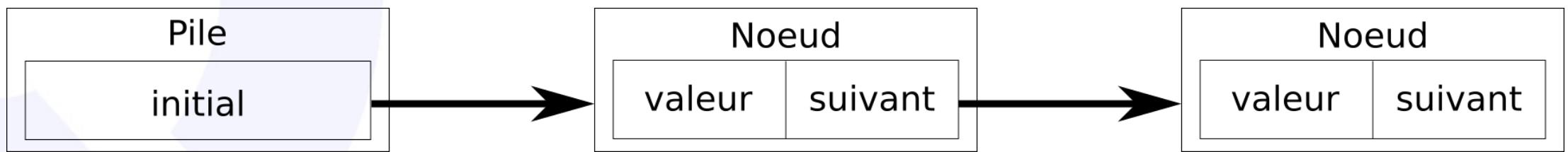
Métaphore



Principe



Exemple avec 2 éléments



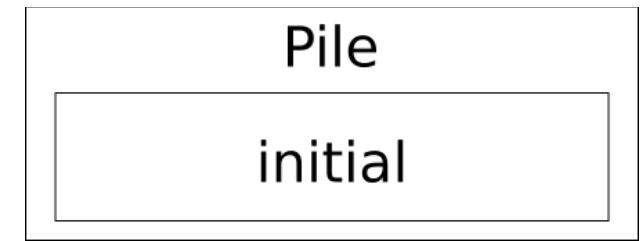
Structure de données

```
from dataclasses import dataclass

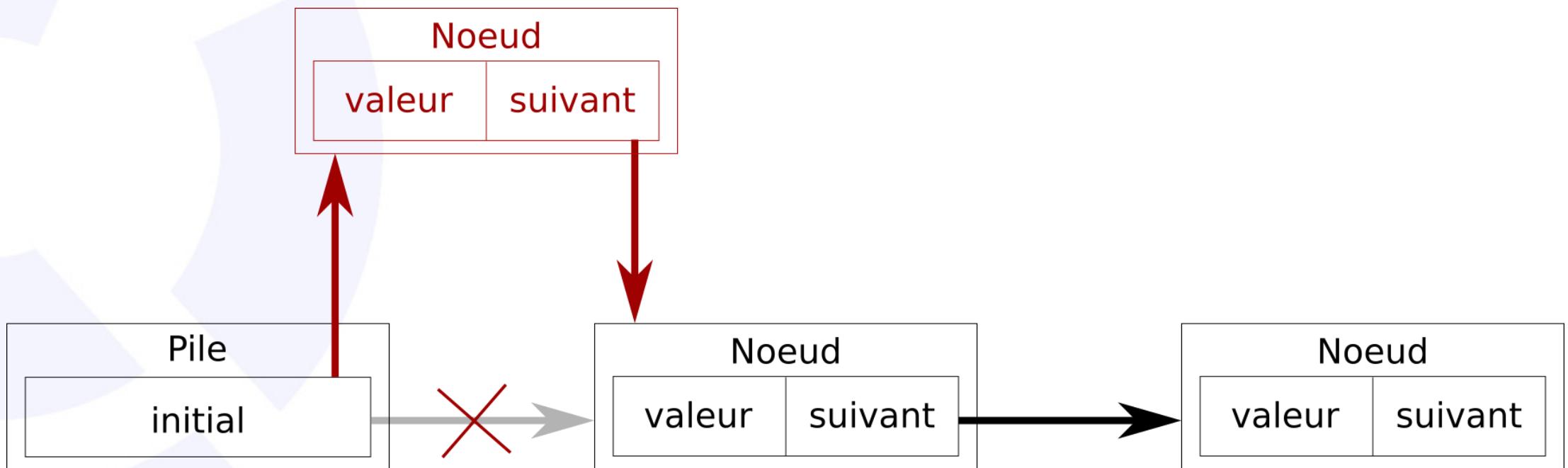
@dataclass
class Pile:
    """Pile utilisant une liste chaînée.

    initial - doit avoir pour type
              Noeud ou None.

    """
    initial = None
```



Empile



Dépile



Utilisation d'une list

```
def empile(pile, element):
    """Empile l'élément dans la pile.

    pile - la pile à modifier.
    element - élément à empiler dans la pile.
    """
    pile.insert(0, element)

def depile(pile):
    """Dépile le 1er élément de la pile.

    pile - la pile à modifier.
    Retourne le 1er élément de la pile.
    """
    return pile.pop(0)
```



Exemple

```
pile = [6, 3, 7]
empile(pile, 10)
print(pile)          # [10, 6, 3, 7]

valeur = depile(pile)
print(valeur)        # 10
print(pile)          # [6, 3, 7]
```

Utilisation de deque

```
from collections import deque

def empile(pile, element):
    """Empile l'élément dans la pile.

    pile - la pile à modifier.
    element - élément à empiler dans la pile.
    """
    pile.appendleft(element)

def depile(pile):
    """Dépile le 1er élément de la pile.

    pile - la pile à modifier.
    Retourne le 1er élément de la pile.
    """
    return pile.popleft()
```



Exemple

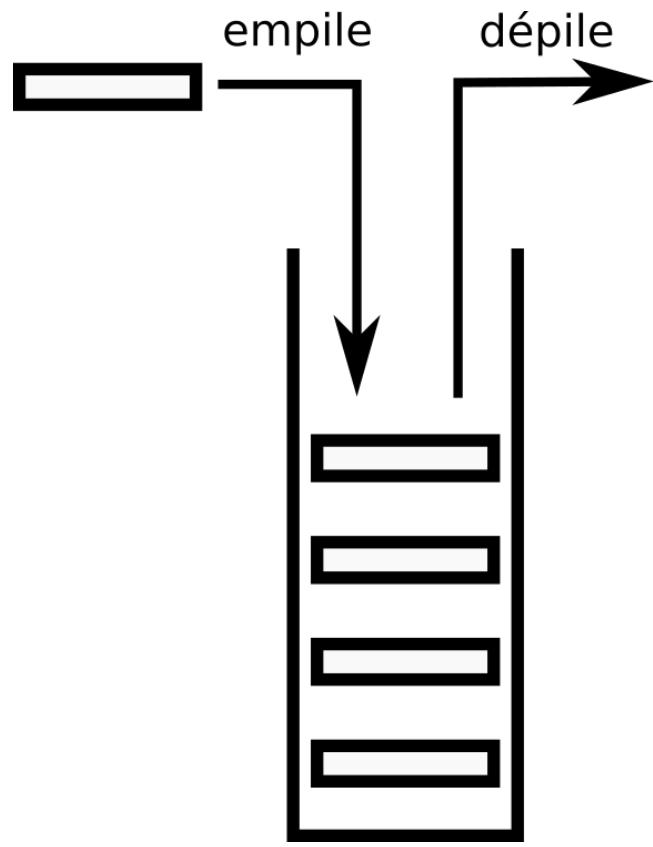
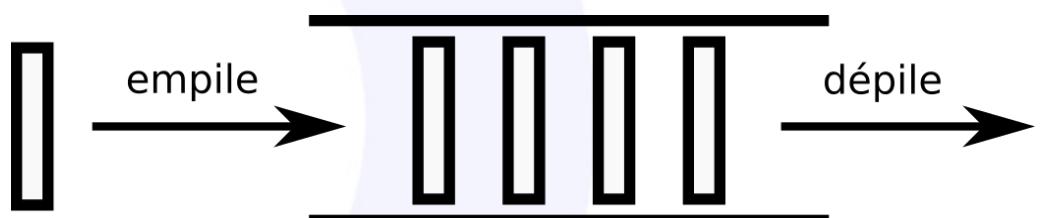
```
pile = deque([6, 3, 7])
empile(pile, 10)
print(pile)          # deque([10, 6, 3, 7])

valeur = depile(pile)
print(valeur)        # 10
print(pile)          # deque([6, 3, 7])
```

Comparaison entre FIFO et LIFO

FIFO LIFO

| | Anglais | Français | Collection |
|------|------------------------|---------------------------------|------------|
| FIFO | First In, First Out | Premier Entré, Premier Sorti | Queue |
| LIFO | Last In, First Out | Dernier Entré, Premier Sorti | Pile |



FIFO dans la réalité

Les queues de messages

- Ordonnanceur de tâches (systèmes d'exploitation).
- Traitements asynchrones dans un système.
- Version itérative d'algorithmes récursifs.

LIFO dans la réalité

- Pile d'appels de fonctions.
- Interpréteur.
- Version itérative d'algorithmes récursifs.

TP : Queues de messages simples

TP : Queues de messages simples

[Lien vers le sujet de DM.](#)

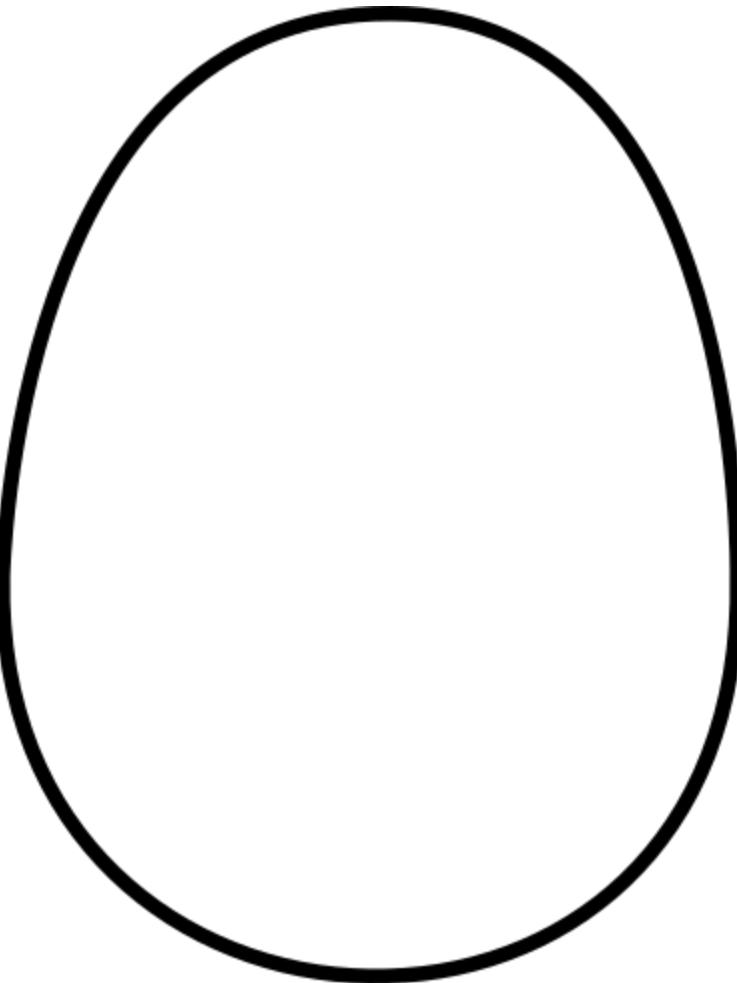
Rappels sur la théorie des ensembles

Union, intersection, différence

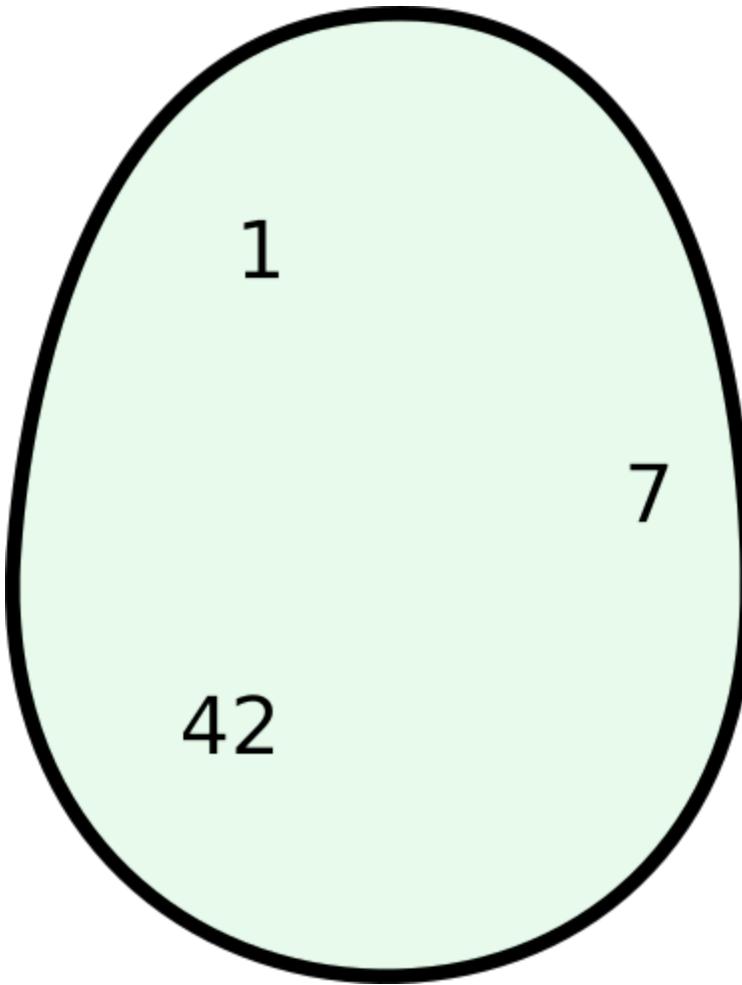
Intérêt

- Nous avons vu le type `set` dans le cours précédent.
- Le DM n°3 vous demande d'implémenter les opérations classiques sur les ensembles.
- Nous revenons rapidement sur ces définitions pour préparer le DM.

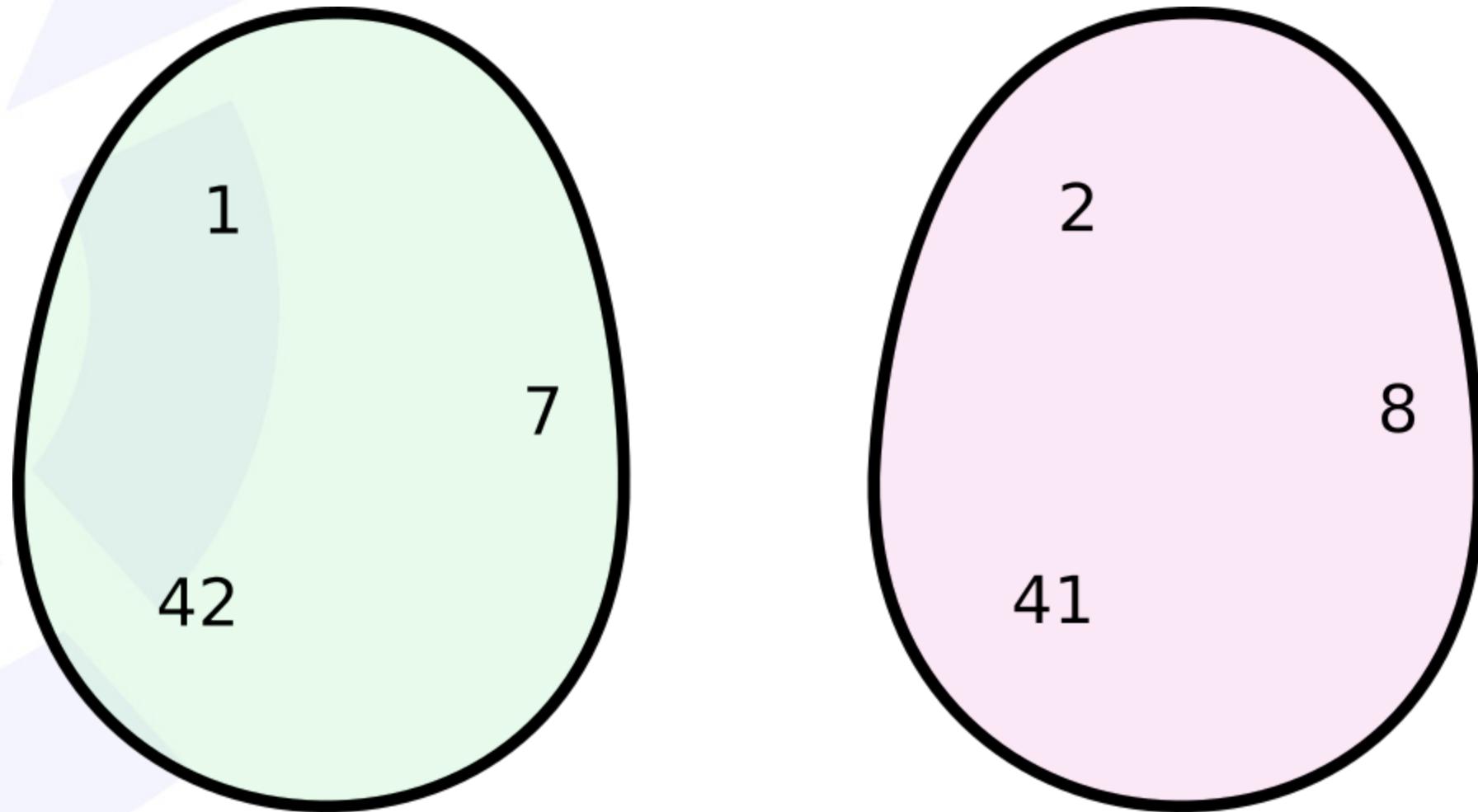
Ensemble vide



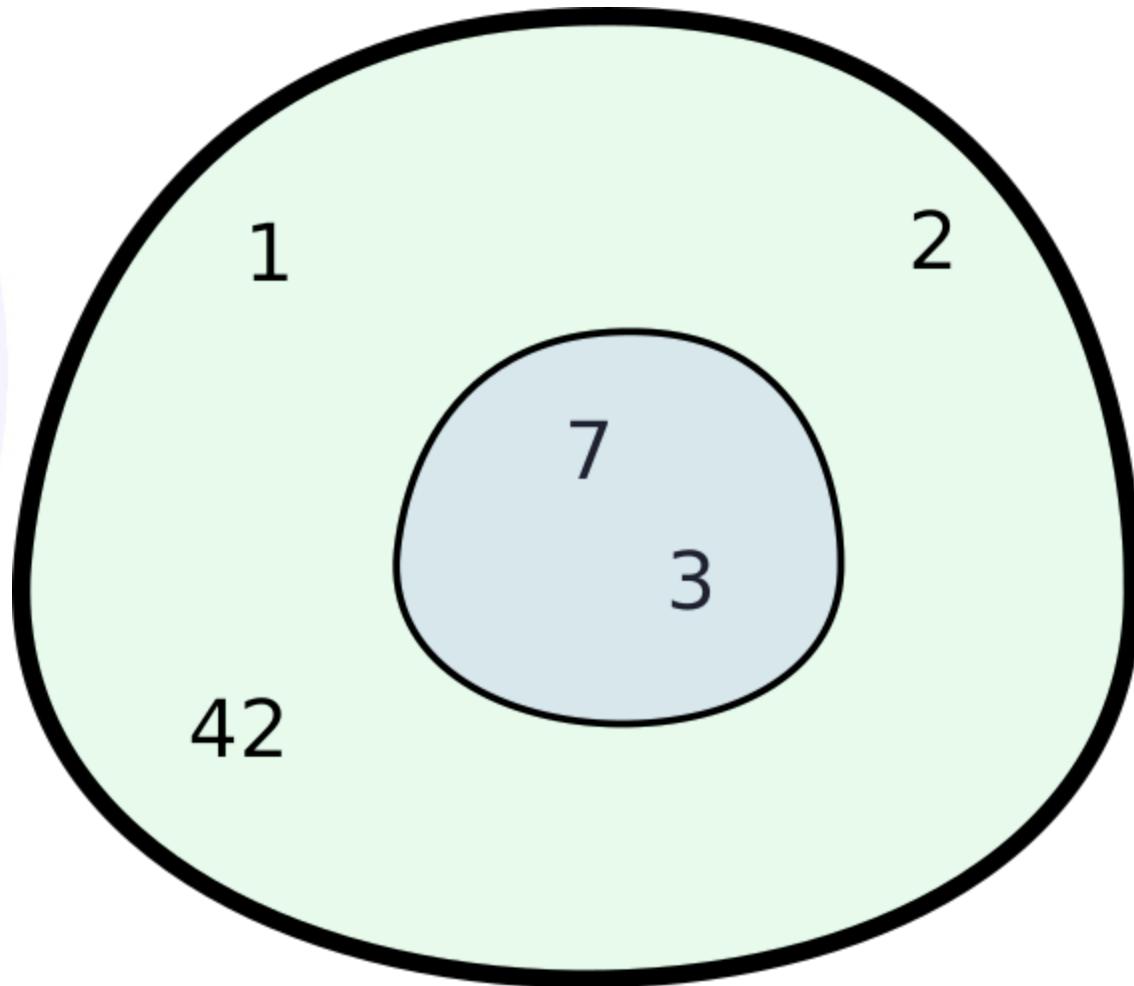
Ensemble avec quelques éléments



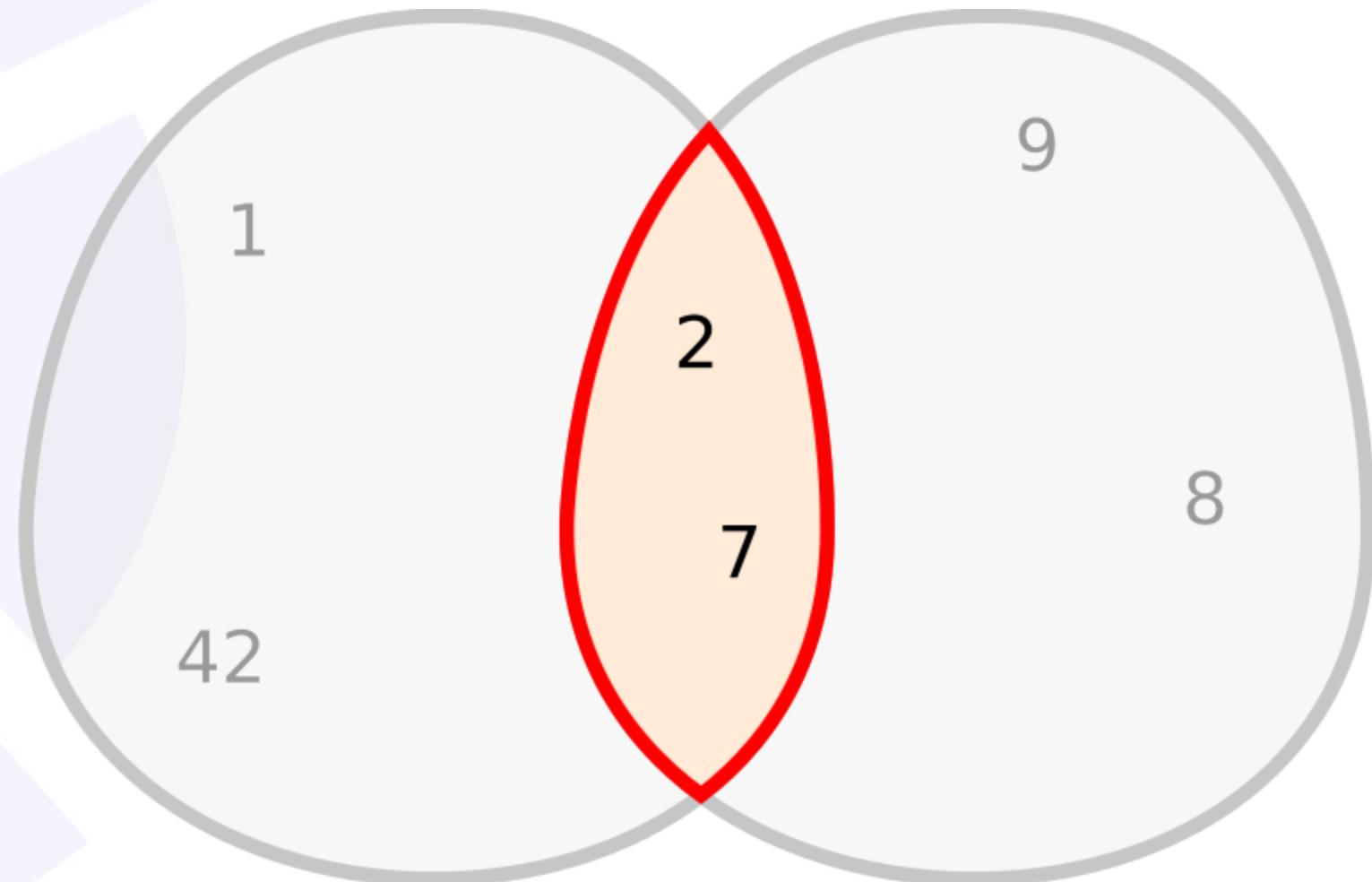
Ensembles disjoints



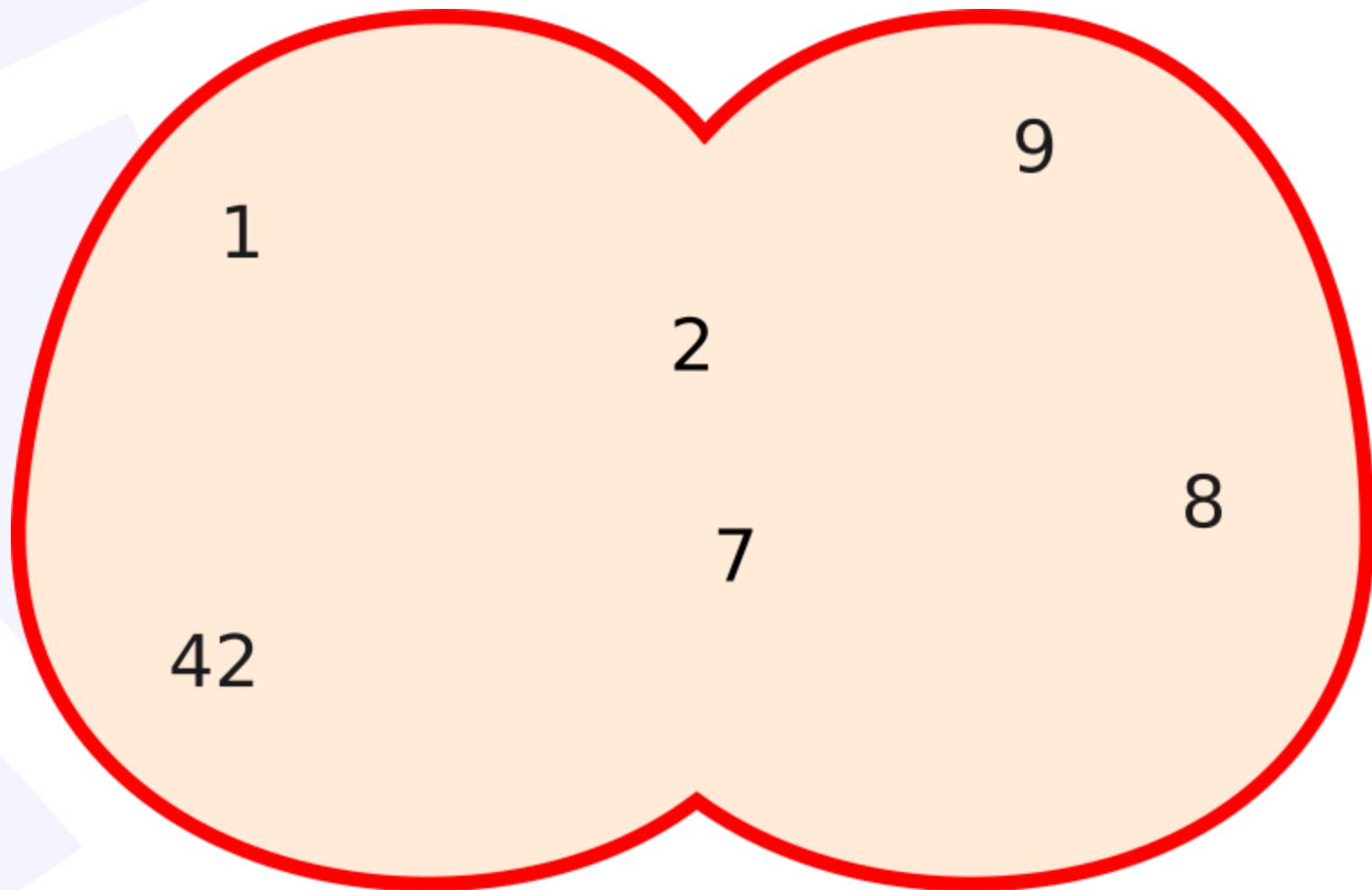
Sous-ensemble



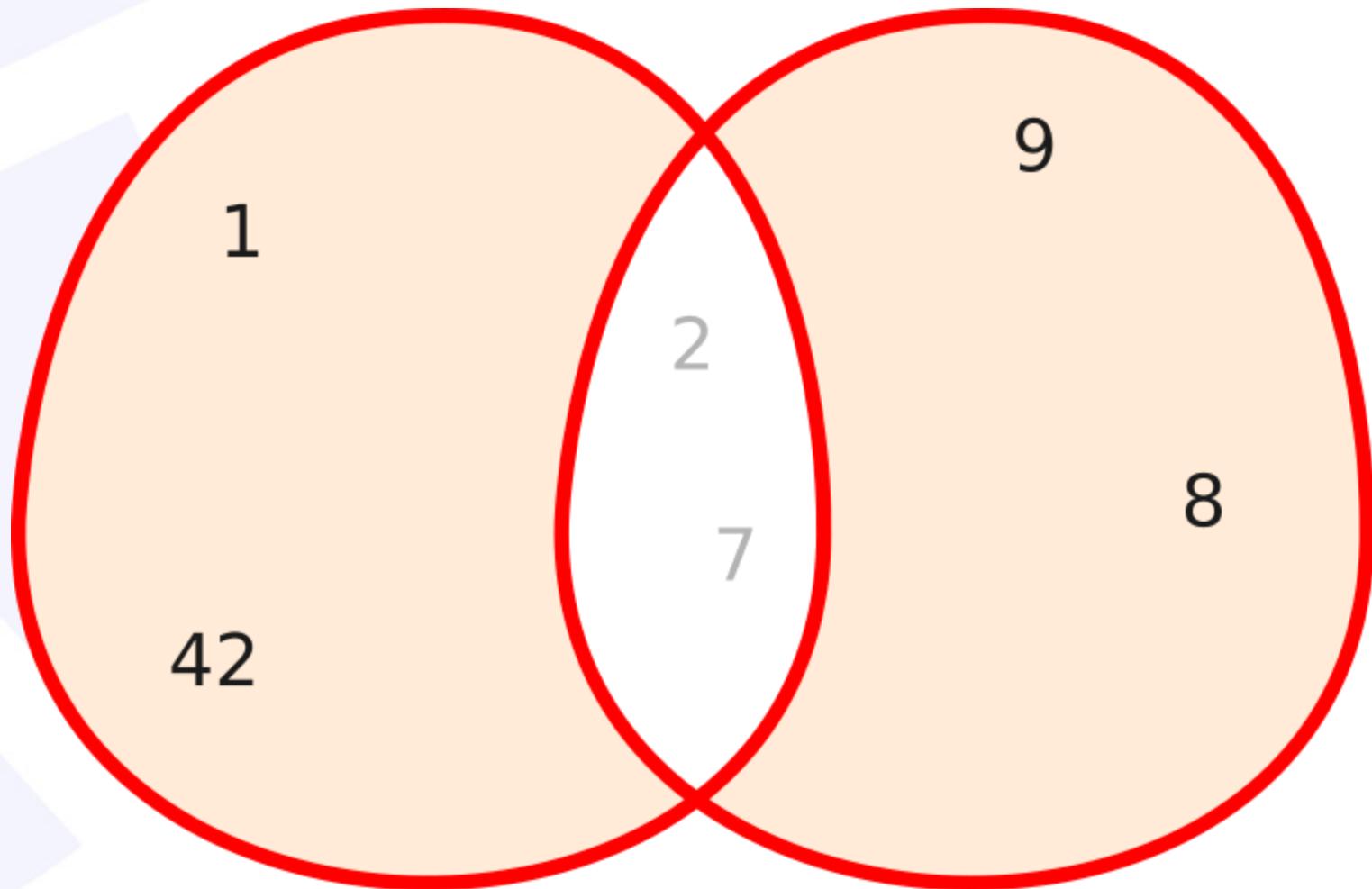
Intersection



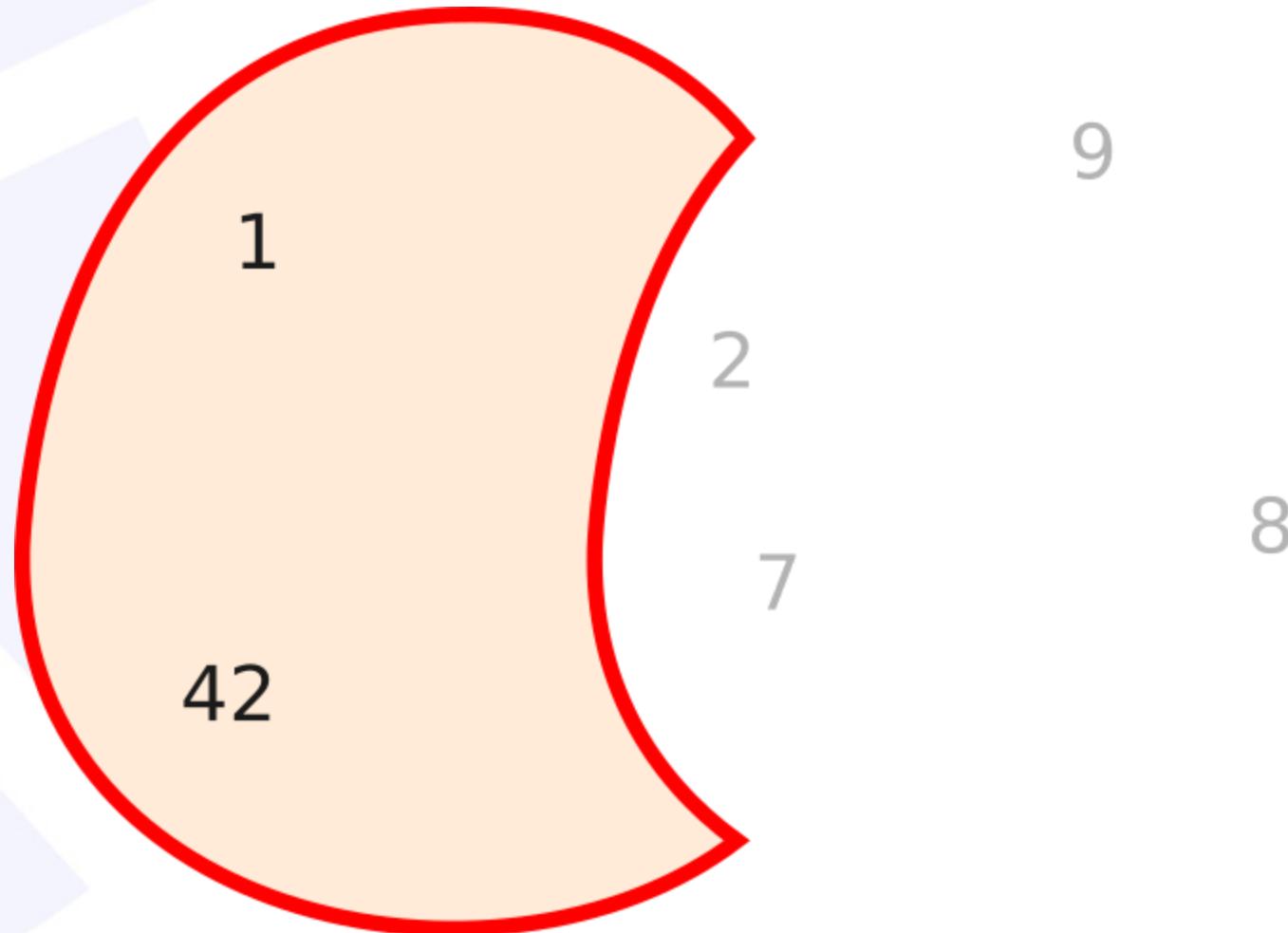
Union



Exclusion



Différence



Rappels sur le calcul matriciel

Résolution de systèmes d'équations linéaires

Intérêt

- En prévision du DM n°3 et de l'examen.
- Pour faire des jeux vidéos.
- Pour la conception assistée par ordinateur.

Déterminant d'une matrice

2x2

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - cb$$

Expansion de Laplace

Déterminant d'une matrice 3x3

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} e & f \\ h & i \end{vmatrix}$$

Il s'agit d'une définition récursive.

Expansion de Laplace

Déterminant d'une matrice NxN

$$|M| = \begin{vmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,N} \\ m_{2,1} & m_{2,2} & \dots & m_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N,1} & m_{N,2} & \dots & m_{N,N} \end{vmatrix} = \sum_{j=1}^N m_{1,j} (-1)^{1+j} |M_{1,j}|$$

Les $M_{1,j}$ sont les **mineurs des matrices** de la première ligne de M .

Mineur d'une matrice $M_{i,j}$

$$M_{i,j} = \begin{pmatrix} m_{1,1} & \dots & m_{1,j-1} & m_{1,j+1} & \dots & m_{1,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ m_{i-1,1} & \dots & m_{i-1,j-1} & m_{i-1,j+1} & \dots & m_{i-1,n} \\ m_{i+1,1} & \dots & m_{i+1,j-1} & m_{i+1,j+1} & \dots & m_{i+1,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ \vdots & & \vdots & \vdots & & \vdots \\ m_{n,1} & \dots & m_{n,j-1} & m_{n,j+1} & \dots & m_{n,n} \end{pmatrix}$$

Le mineur d'une matrice M aux indices (i, j) est noté
 $M_{i,j}$.

Système d'équations linéaires

Sous forme matricielle

$$\begin{cases} a_1x + b_1y + c_1z = d_1 \\ a_2x + b_2y + c_2z = d_2 \\ a_3x + b_3y + c_3z = d_3 \end{cases} \iff \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

Résolution du système

$$\begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}^{-1} \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}^{-1} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}^{-1} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$$

Problème : L'inversion matricielle n'est pas triviale.

Règle de Cramer

$$x = \frac{\begin{vmatrix} d_1 & b_1 & c_1 \\ d_2 & b_2 & c_2 \\ d_3 & b_3 & c_3 \end{vmatrix}}{\det(M)}$$

$$y = \frac{\begin{vmatrix} a_1 & d_1 & c_1 \\ a_2 & d_2 & c_2 \\ a_3 & d_3 & c_3 \end{vmatrix}}{\det(M)}$$

$$z = \frac{\begin{vmatrix} a_1 & b_1 & d_1 \\ a_2 & b_2 & d_2 \\ a_3 & b_3 & d_3 \end{vmatrix}}{\det(M)}$$

Matrice Identité

L'**élément neutre** de la multiplication matricielle est noté I et comporte des 0 partout sauf sur sa diagonale, composée de 1.

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

$$I \cdot M = M \cdot I = M$$

Matrice triangulaire

Soit la partie supérieure de la matrice, soit la partie inférieure de la matrice, n'est composée que de 0.

$$M_1 = \begin{pmatrix} 1 & 4 & 5 \\ 0 & 2 & 6 \\ 0 & 0 & 3 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 2 & 0 \\ 5 & 6 & 3 \end{pmatrix}$$

M_1 est triangulaire supérieure.

M_2 est triangulaire inférieure.

Matrice diagonale

Tous les éléments de la matrice sont nuls, sauf ceux sur sa diagonale.

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

Puissance d'une matrice diagonale

$$M = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}, \quad M^N = \begin{pmatrix} a^N & 0 & 0 \\ 0 & b^N & 0 \\ 0 & 0 & c^N \end{pmatrix}$$

Donc :

$$M^{-1} = \begin{pmatrix} a^{-1} & 0 & 0 \\ 0 & b^{-1} & 0 \\ 0 & 0 & c^{-1} \end{pmatrix}$$

Diagonalisation : principe (1/2)

- Si une matrice M est diagonalisable alors on peut l'écrire $M = PDP^{-1}$ où D est sa matrice diagonale, et P la matrice de passage.
- Pour diagonaliser, on calcule les **valeurs propres** λ telles que $|M - \lambda I| = 0$.
- Pour chaque valeur propre, on calcule le **vecteur propre** associé tel que $MX = \lambda X$.

Diagonalisation : principe (2/2)

- La matrice diagonale équivalent D s'obtient en mettant les valeurs propres sur la diagonale.
- La matrice de passage P est obtenue en mettant les vecteurs propres obtenus en colonnes.

Matrice augmentée

Si $M = \begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}$, et $D = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix}$

Alors $[M|D] = \begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{pmatrix}$

Elimination de Gauss Jordan (1/3)

On sait que les opérations suivantes sont possibles pour résoudre le système sans changer la solution :

- Interchanger 2 lignes.
- Multiplier une ligne par un scalaire non-nul.
- Ajouter un multiple d'une ligne à une autre.

Elimination de Gauss Jordan (2/3)

- On parcourt chaque colonne col de M :
 - Le pivot est l'élément sur la diagonale dans cette colonne : $Pivot = M[col][col]$.
 - On parcourt chaque ligne lig de M sauf celle où se trouve le pivot :
 - On calcule le coefficient multiplicateur $Coeff = \frac{M[lig][col]}{Pivot}$.
 - On soustrait $Coeff$ fois la ligne col à la ligne lig .

Elimination de Gauss Jordan (3/3)

La solution est la dernière colonne dont chaque élément est divisé par le pivot sur la même ligne.

Exemple

On considère, dans \mathbb{R}^3 , le système suivant :

$$\begin{cases} x - 0.5y - z = 2 \\ 2x - y + z = 1 \\ x - y - 2z = 3 \end{cases}$$

Sous forme matricielle

$$\begin{cases} x - 0.5y - z = 2 \\ 2x - y + z = 1 \\ x - y - 2z = 3 \end{cases} \iff \begin{pmatrix} 1 & -0.5 & -1 \\ 2 & -1 & 1 \\ 1 & -1 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$$

Matrice augmentée

$$\begin{pmatrix} 1 & -0.5 & -1 & 2 \\ 2 & -1 & 1 & 1 \\ 1 & -1 & -2 & 3 \end{pmatrix}$$

Sous forme de tableau

| | col1 | col2 | col3 | col4 |
|------|------|------|------|------|
| lig1 | 1 | -0.5 | -1 | 2 |
| lig2 | 2 | -1 | 1 | 1 |
| lig3 | 1 | -1 | -2 | 3 |

1er pivot

| | col1 | col2 | col3 | col4 |
|------|------|------|------|------|
| lig1 | 1 | -0.5 | -1 | 2 |
| lig2 | 2 | -1 | 1 | 1 |
| lig3 | 1 | -1 | -2 | 3 |

| | col1 | col2 | col3 | col4 | Opérations |
|------|------|------|------|------|------------------|
| lig1 | 1 | -0.5 | -1 | 2 | |
| lig2 | 0 | 0 | 3 | -3 | lig2 -= 2 * lig1 |
| lig3 | 0 | -0.5 | -1 | 1 | lig3 -= 1 * lig1 |

Echange lig2 et lig3

| | col1 | col2 | col3 | col4 |
|------|------|------|------|------|
| lig1 | 1 | -0.5 | -1 | 2 |
| lig3 | 0 | -0.5 | -1 | 1 |
| lig2 | 0 | 0 | 3 | -3 |

Un pivot ne peut pas être nul.

2e pivot

| | col1 | col2 | col3 | col4 |
|------|------|-------------|------|------|
| lig1 | 1 | -0.5 | -1 | 2 |
| lig3 | 0 | -0.5 | -1 | 1 |
| lig2 | 0 | 0 | 3 | -3 |

| | col1 | col2 | col3 | col4 | Opérations |
|------|------|-------------|------|------|------------------|
| lig1 | 1 | 0 | 0 | 1 | lig1 -= lig3 |
| lig3 | 0 | -0.5 | -1 | 1 | |
| lig2 | 0 | 0 | 3 | -3 | lig2 -= 0 * lig3 |

3e pivot

| | col1 | col2 | col3 | col4 |
|------|------|------|------|------|
| lig1 | 1 | 0 | 0 | 1 |
| lig3 | 0 | -0.5 | -1 | 1 |
| lig2 | 0 | 0 | 3 | -3 |

| | col1 | col2 | col3 | col4 | Opérations |
|------|------|------|------|------|--------------------|
| lig1 | 1 | 0 | 0 | 1 | lig1 -= 0 * lig2 |
| lig3 | 0 | -0.5 | 0 | 0 | lig3 += 1/3 * lig2 |
| lig2 | 0 | 0 | 3 | -3 | |

Dernière étape

Division de la dernière colonne par les pivôts

$$\text{col4} = \begin{pmatrix} 1 \\ 0 \\ -3 \end{pmatrix}, \quad \text{Pivôts} = \begin{pmatrix} 1 \\ -0.5 \\ 3 \end{pmatrix}, \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

Devoir à la Maison 03

DM : Retours sur les fonctions et le débogage

[**Lien vers le sujet de DM.**](#)