PÔLE CHATEAUBRIAND JOLIOT-CURIE

# Fiche TP17 - Matrices et images

Informatique - Structure de données

**Objectifs :** Savoir définir et recopier une matrice ; représenter une image à l'aide d'une matrice.

## A Représentation des matrices à deux dimensions en Python

Les deux objets usuels pour représenter les matrices sont les listes de listes et les tableaux numpy utilisés en physique et en SI. Nous nous interessons ici aux listes de listes.

**Définition :** avec les types de base de Python, chaque ligne d'une **matrice** peut être représentée par une liste. La matrice est alors définie par une liste de lignes, soit par une **liste de listes**.

Exemple :

```
1  L0 = [2., 2, 3, 5]
2  L1 = [1, 3, 5, 0]
3  L2 = [0, 0, 0, 1]
4  A = [L0, L1, L2]
```

Pour une définition littérale (rare et pour de petites matrices), on preférera une définition sans variables intermédiaires :
`A = [[2., 2, -3], [-2, -1, 1], [4, 1, 1]]`
ou, pour plus de lisibilité :

```
1  A = [[2., 2, 3, 5],
2       [1, 3, 5, 0],
3       [0, 0, 0, 1]]
```

On obtient les **éléments** et les **dimensions** ainsi :

- **ligne** $i$ sous la forme d'une liste : `A[i]`
- **élément** $A_{i,j}$ : `A[i][j]`
- **nombre de lignes**, le nombre d'éléments de `A` : `len(A)`
- **nombre de colonnes**, la longueur d'une ligne : `len(A[0])`

On remarque qu'il n'y a pas d'expression simple retournant une colonne.

## B Images numériques

Les images peuvent être codées sous forme vectorielle ou sous forme matricielle.

Sous forme vectorielle, l'image est un ensemble de fonctions mathématiques représentant des formes simples (traits, carrés, éllipses...), des formes paramétriques (courbes de bézier...) et des informations sur les positions relatives de ces formes et la façon dont elles sont coloriées. Ce format assure la continuité de la géométrie : il est possible d'agrandir l'image presque indéfiniement sans perte de qualité. Inkscape, Adobe Illustrator ou ... powerpoint, sont quelques exemples de logiciels gérant ce type d'images. Des formats usuels sont SVG, pdf. Les polices de caractères et les logos sont généralement définis sous forme vectorielle.

Sous forme **matricielle**, une image est une grille de points, une matrice de points. A chaque point, appelé **pixel**, est associé une information de couleur. Les images générées par les appareils photo numériques, les caméras numériques, les scanners sont de ce type. Il existe de nombreuse manière de coder la couleur de chaque pixel.

Pour une image en niveaux de gris, c'est la luminosité qui est codée, généralement par un entier compris entre 0 et 255 (ou 0 et 65535).

La **synthèse additive** est utilisée par les moniteurs et les projecteurs. La couleur est obtenue par addition des trois lumières de base qui sont le **rouge**, le **vert** et le **bleu**, dites **RVB** (RGB en anglais). Les couleurs secondaires sont plus claires que les couleurs primaires et le blanc est obtenu par superposition des 3 couleurs

primaires. Remarque : la synthèse soustractive est utilisée par les imprimantes, l'imprimerie et la peinture. Les couleurs primaires sont le cyan, le magenta, le jaune et le noir, dite CMJN. Le mélange conduit à des couleurs plus sombres par absorpsion de la lumière ambiante.

Le codage RVB code la couleur avec **3 entiers** usuellement compris chacun entre 0 et 255 (codage sur un octet par couleur, soit sur 8 bits). Remarque : le nombre de niveaux par couleur de base est donc faible pouvant provoquer l'apparition d'aplats et de bandes dans les dégradées monochromes (ciel bleu, soleils couchant, images en niveaux de gris). Un codage sur 10 bits permettant 1024 niveaux différents se démocratise actuellement pour limiter ce phénomène.

Adobe Photoshop, Gimp, Paint sont des logiciels de création et traitement d'images matricielles. Les formats usuels sont png, gif, jpeg. Ils comprennent tous des algorithmes de compression pour réduire la taille du fichier de données.

**Définition :** les images numériques se représentent par des **matrices** d'entiers (images en niveaux de gris) ou de triplets d'entiers (images RVB...).

Un codage de la luminance par un flottant compris entre 0 et 1 est aussi possible.

Exemple pour une image en niveau de gris. 0 pour un pixel noir, 255 pour un pixel blanc.

Matrice d'entiers :

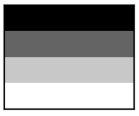
```
img = [[50, 0, 255, 0, 255, 0],
       [100, 0, 0, 0, 0, 0],
       [150, 0, 0, 0, 0, 0],
       [200, 0, 0, 0, 0, 0]]
```

Image obtenue :

Exemple pour une image RVB avec un dégradé de gris (lorsque les 3 valeurs d'un triplet sont identiques, la couleur est grise). `[0, 0, 0]` pour un pixel noir, `[255, 255, 255]` pour un pixel blanc.

```
img = [[[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]],
       [[100,100,100], [100,100,100], [100,100,100], [100,100,100], [100,100,100]],
       [[200,200,200], [200,200,200], [200,200,200], [200,200,200], [200,200,200]],
       [[255,255,255], [255,255,255], [255,255,255], [255,255,255], [255,255,255]]]
```

Image obtenue :



Exemple pour une image RVB avec deux diagonales de couleur (cyan et magenta, mais moins visible avec une impression noir et blanc...).

```
img = [[[255,255,255], [255,255,255], [255,255,255], [255,255,255], [255,255,255]],
       [[255,255,255], [200,255,255], [255,255,255], [255,200,255], [255,255,255]],
       [[255,255,255], [255,255,255], [150,150,255], [255,255,255], [255,255,255]],
       [[255,255,255], [255,100,255], [255,255,255], [100,255,255], [255,255,255]],
       [[255,0,255], [255,255,255], [255,255,255], [255,255,255], [0,255,255]]]
```

Image obtenue :

Une image comprend facilement des millions de pixels : 1,9 millions pour un écran d'ordinateur classique, 12 à 48 millions pour une photo de téléphone.

## C Création et copie de matrices de pixels

**Création d'une matrice :** pour créer des matrices génériques (remplies de 0, d'une même valeur...), on utilise une création ligne par ligne, faite élément par élément ou par compréhension.

**Création d'une ligne :** une ligne ne comprenant que des entiers ou flottants peut être créée par répétition, sinon il faut la créer élément par élément ou par compréhension.

Par la suite, les dimensions de la matrice, `n` et `p`, entiers, sont supposées définies.

### C.1 Création d'une matrice de 0: on peut utiliser la répétition pour créer une ligne.

Création élément par élément

```
M = []
for i in range(n):
    M.append([0] * p) # ajout d'une ligne de
p valeur nulles
```

Création par compréhension

```
M = [[0] * p for i in rang(n)]
```

### C.2 Création d'une image RVB blanche: un pixel blanc a pour couleur le triplet (255, 255, 255). Il ne faut pas utiliser la répétition pour définir une ligne, sinon tous les pixel d'une ligne changeront de couleur simultanément, cf. aspect mutable des listes présenté plus loin. Le plus simple est de créer la **matrice élément par élément**.

Création élément par élément d'une ligne

```
M = []
for i in range(n):
    ligne = []
    for j in range(p):
        ligne.append([255, 255, 255])
    M.append(ligne)
```

Création par compréhension d'une ligne

```
M = []
for i in range(n):
    ligne = [[255, 255, 255] for j in range(p)]
    M.append(ligne)
```

Remarque : on peut aussi créer directement la matrice par compréhension (à n'utiliser que si les codes précédents sont maitrisés) : `M = [[[255, 255, 255] for j in range(p)] for i in range(n)]`

### C.3 Recopie d'une image: une image RVB étant une liste de listes de listes, il faut s'assurer que chacune des listes est dupliquée. L'instruction `A = M[:]` n'est pas du tout suffisante.

**Dupliquer une image :** pour une image en niveau de **gris**, il faut **dupliquer les lignes** ligne par ligne ; pour une image en **RVB** il faut **dupliquer les pixels et les lignes**, pixel par pixel puis ligne par ligne.

Niveau de gris : on duplique les lignes

```
A = []
for i in range(n): # pour chaque ligne
    A.append(M[i][:])
```

RVB : on duplique les lignes et les pixels

```
A = []
for i in range(n): # pour chaque ligne
    ligne = []
    for j in range(p): # pour chaque pixel
        ligne.append(M[i][j][:])
    A.append(ligne)
```

Les données sont dupliquées par le parcours d'une liste du premier au dernier élément : `M[i][:]` et `M[i][j][:]`. C'est une source d'erreur facile, car si l'on remplace ces deux instructions par `M[i]` et`M[i][j]`, le code ne présente pas d'erreur de syntaxe, crée une matrice `A` contenant les bonnes données, mais celles-ci ne sont pas dupliquées. La modification de `A` entraine alors celle de `M`.

Ainsi : `ligne = M[i]` ne duplique pas les données, alors que `ligne = M[i][:]` créé une nouvelle liste en utilisant les éléments de `M[i]` d'indices allant du premier au dernier élément : les données sont dupliquées.

Remarque : le code peut aussi être écrit par compréhension : `A = [[M[i][:] for i in range(n)]` et `A = [[[M[i][j][:] for j in range(p)] for i in range(n)]`. Mais comme précédemment, il faut d'abord bien comprendre et savoir utiliser les boucles.

## D Aspect mutable des listes

Les listes sont mutables, c'est à dire, modifiables : on peut changer un élément, en ajouter un (avec `.append()`) ou en supprimer un (avec `.pop()` par exemple).

```
>>> L = [1, 2, 3]
>>> L[0] = 0 # changement du premier élément. Il valait 1, il vaut 0
>>> L
[0, 2, 3]
```

Comme tous les objets, une liste peut avoir plusieurs noms (références). Mais comme est mutable, elle peut être modifiée en utilisant l'un ou l'autre des noms.

```
>>> M = L # M et L, deux noms d'une même liste
>>> L[1] = 0 # modification par L
>>> M
[0, 0, 3]
>>> M[2] = 0 # modification par M
>>> L
[0, 0, 0]
```

C'est un aspect pratique qui permet a une fonction de modifier une liste qui lui est transmise en paramètre et donc de réaliser un tri en place, par exemple, ou de lui ajouter des éléments avec `.append`.

Lors de la création d'une liste de listes, le même phénomène peut apparaitre, car une liste contient les références aux objets, pas les objets eux-même. Elle peut donc contenir plusieurs références à une même liste.

```
>>> L = [1, 2, 3]
>>> M = [L, L] # les deux lignes sont identiques car elles sont le même objet
>>> M[0][0] = 0 # modification de L et donc des 2 lignes
```

La liste `L` contient alors `[0, 2, 3]` et la matrice `M` : `[[0, 2, 3], [0, 2, 3]]`.

La répétition d'une liste ne fait que dupliquer les références aux éléments et pose le même problème, lorsque l'élément répété est mutable.

```
>>> M = [[0, 0]] * 3
>>> M
[[0, 0], [0, 0], [0, 0]]
>>> M[0][0] = 1
```

`M` contient alors : `[[1, 0], [1, 0], [1, 0]]`.

On peut considérer que l'instruction `M = [[0, 0]] * 3` est équivalente à définir l'élément répété `L = [0, 0]`, puis a définir la matrice ainsi : `M = [L, L, L]`. A éviter absolument.

**Conséquence 1 :** la répétition ne sera utilisée que si les éléments répétés sont non mutables (entiers, flottants, booléens, n-uplets).

**Conséquence 2 :** pour créer des matrices génériques (remplies de 0, d'une même valeur...) ou pour dupliquer une matrice, il faut utiliser une création élément par élément, comme indiqué précédemment.