

## Séance 8 : ENSEMBLES, DICTIONNAIRES ET MATRICES

L1 – Université Côte d’Azur

Dans ce TP, vous allez manipuler des images aux formats PBM, PGM et PPM. Ces formats permettent très facilement de lire et modifier une (petite) image avec un éditeur de texte. Pour commencer, récupérez l’archive `tp8.zip` à l’adresse : <https://upinfo.univ-cotedazur.fr/~obaldellon/L1/py/tp8/tp8.zip>, décompressez-la dans le répertoire qui contient tous vos TP Python, puis allez dans le répertoire `tp8`. Vous devez y trouver les fichiers

```
exemple.pbm
space-invader-pieuvre.pbm    pnm.py
space-invader-soucoupe.pbm  chat.py
```

### Le format PBM

Un fichier PBM (portable bitmap) est un fichier qui contient une image noir et blanc pixelisée.

```
# fichier exemple.pbm
P1
# indique le format pbm
5 # indique la largeur de l'image
3 # indique la hauteur de l'image
1 1 1 1 1 # une premiere ligne noire
0 0 0 0 0 # une deuxieme ligne blanche
1 1 1     # debut troisieme ligne
1 1       # fin troisieme ligne
```

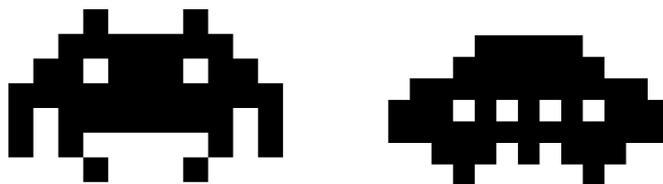
En plus d’une extension `.pbm`, un fichier PBM doit commencer par `P1`. Ensuite, le fichier contient seulement des nombres et des commentaires. Comme en Python, tout ce qui suit un `#` sur une ligne est un commentaire. Les nombres sont séparés par des espaces ou des passages à la ligne. Les deux premiers nombres sont la largeur et la hauteur de l’image. Les nombres suivants codent les couleurs de chaque pixel, ligne par ligne et de gauche à droite, et prennent la valeur 0 pour un pixel blanc et 1 pour un pixel noir.

### Exercice 1 – Lire et modifier un fichier PBM (\*)

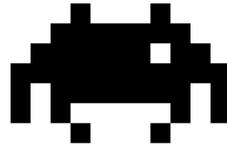
1. Créez un fichier `tp8.py` dans le répertoire contenant tous les fichiers extraits de l’archive. Affichez les deux images (`space-invader-pieuvre.pbm` et `space-invader-soucoupe.pbm`) en utilisant la fonction `voir_fichier` du module `pnm.py`. Vous pouvez vous inspirer de l’exemple ci-dessous.

```
1 from pnm import * # À vous de modifier la suite
2 voir_fichier('space-invader-pieuvre.pbm', quadrillage=True)
3 voir_fichier('space-invader-soucoupe.pbm')
```

Vous devriez voir, successivement, les images suivantes :



- À l’aide de Thonny (Menu Fichier, Ouvrir, et « tous les fichiers » ou « all files » en bas à droite), ouvrez le fichier : `space-invader-pieuvre.pbm`. Quelles sont les dimensions de l’image ? Si vous remplacez le 9 par un 8, comment est modifiée l’image ? Et le 13 par un 12 ? Comment faut-il modifier le fichier pour obtenir l’image ci-dessous ?



- Ajoutez au fichier `tp8.py` le code ci-dessous et exécutez-le. À quoi correspond le contenu de la matrice `M` ?

```
1 M = fichier_vers_matrice('space-invader-pieuvre.pbm')
2 print(M)
```

- Complétez les `...` dans le code ci-dessous afin de créer un fichier `space-invader-pieuvre2.pbm` qui contiendra une pieuvre qui ferme son œil gauche comme vu plus haut.

```
1 M = fichier_vers_matrice('space-invader-pieuvre.pbm')
2 M[ ... ][ ... ] = ....
3 matrice_vers_fichier(M, 'space-invader-pieuvre2.pbm')
4 voir_fichier('space-invader-pieuvre2.pbm')
```

Attention, il y a un bord blanc dans l’image `space-invader-pieuvre.pbm`.

**Exercice 2** – Afficher une image PBM en mode texte (\*-\*\*)

On rappelle que, dans ce TP, vous pouvez définir et utiliser les deux fonctions vues en cours :

- `dimensions(M)` qui renvoie un couple  $(n,m)$  correspondant au nombre de lignes et de colonnes.
- `matrice_vide(n,m)` qui renvoie une matrice de dimensions  $n \times m$  et ne contenant que des `None`.

Dans le fichier `tp8.py`, écrivez une fonction `affiche_matrice_booléens(M,plein,vide)` qui prend en argument une matrice `M`, un caractère `plein` et un caractère `vide` et qui affiche la matrice sous forme rectangulaire en remplaçant `True` par `plein` et `False` par `vide`. Par exemple, si `M` est la matrice obtenue en lisant `space-invader-pieuvre.pbm`, `affiche_matrice(M, '#', '.')` affichera le texte ci-dessous.

```
.....
...#...#...
..#####.
.##.###.##.
.#####.
.#.#####.#.
.#.#....#.#.
...#...#...
.....
```

**Exercice 3** – Une armée tentaculaire (\*\*)

On veut maintenant construire l’image suivante



On va pour cela utiliser un accumulateur pour construire chaque ligne puis faire leur somme par concaténation. Cependant, l’opération de concaténation de matrices n’est pas prédéfinie en Python, et il va falloir commencer par la définir.

1. Écrivez une fonction `ajoute_horizontal(M1,M2)` qui prend en arguments deux matrices  $M_1, M_2$  de dimensions respectives  $l_1$  colonnes  $\times$   $h$  lignes et  $l_2$  colonnes  $\times$   $h$  lignes et qui renvoie la matrice dont les dimensions sont  $l_1 + l_2$  colonnes  $\times$   $h$  lignes obtenue en concaténant  $M_2$  à droite de  $M_1$ . Testez votre fonction en exécutant le code ci-dessous puis en ouvrant le fichier `space-invader-3-pieuvres.pbm`.

```

1 from pnm import *
2
3 def ajoute_horizontal(M1,M2) :
4     # votre code ici
5
6 M = fichier_vers_matrice('space-invader-pieuvre.pbm')
7 M2 = ajoute_horizontal(M , M)
8 M3 = ajoute_horizontal(M2 , M)
9 matrice_vers_fichier(M3 , 'space-invader-3-pieuvres.pbm')
```

2. Écrivez une fonction `répète_horizontal(M,n)` qui prend en argument une matrice  $M$  et un entier  $n > 0$  et qui renvoie la matrice  $M$  répétée  $n$  fois horizontalement. En utilisant cette fonction, créez un fichier que vous nommerez `space-invader-20-pieuvres.pbm` contenant une rangée de 20 pieuvres.
3. Écrivez une fonction `ajoute_vertical(M1,M2)` qui prend en arguments une matrice  $M_1$  de dimension  $l_1 \times h_1$  et une matrice  $M_2$  de dimension  $l_2 \times h_2$  et qui renvoie une matrice  $M$  de dimension  $\max(l_1, l_2) \times (h_1 + h_2)$  obtenue en concaténant  $M_2$  en dessous de  $M_1$  en les alignant sur la gauche.

*Optionnel : au lieu d'aligner à gauche, on pourra centrer les matrices l'une sous l'autre.*

Testez votre fonction avec le code suivant.

```

1 M1 = fichier_vers_matrice('space-invader-pieuvre.pbm')
2 M2 = fichier_vers_matrice('space-invader-soucoupe.pbm')
3 M3 = ajoute_vertical(M1 , M2)
4 matrice_vers_fichier(M3, 'space-invader-vertical.pbm')
```

4. Créez le fichier `space-invader-army.pbm` qui contient l’image représentée plus haut.

**Exercice 4** – Un peu de couleurs dans ce monde de brutes (★)

Pour l’instant, nous avons travaillé uniquement avec le format PBM ou les pixels sont noirs ou blancs. Il existe deux autres formats de la même famille, le format PGM en nuance de gris et le format PPM en couleur. C’est ce dernier que nous allons utiliser dans cette exercice. L’ensemble de ces trois formats est parfois désigné sous le nom de PNM.

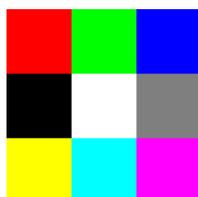
Une matrice correspondant au format PPM est une matrice contenant des triplets  $(r, g, b)$  où  $r, g$  et  $b$  sont des entiers entre 0 et 255 compris.

Dans un nouveau fichier, créez une matrice  $M$  de taille  $3 \times 3$ , dont la première ligne contient du rouge, du vert et du bleu, la seconde du noir du blanc et du gris et la dernière du jaune, du cyan et du magenta. Nous avons vu dans le cours 4 comment coder des couleurs via des triplets.

```

1 from pnm import *
2
3 M=[...] # À compléter
4 voir_matrice(M)
```

La fonction `voir_matrice` permet d’afficher l’image correspondant à la matrice. Vous devriez donc obtenir l’image ci-dessous.



**Exercice 5** – Une photo de chat (★)

Nous arrivons à la fin de l’année et pour vous remercier de votre assiduité, nous vous offrons une photo de chat. Cette photo est définie dans une matrice `chat_mignon` dans le fichier `chat.py` (ne cherchez pas à ouvrir ce fichier, il est très volumineux).

```
1 from pnm import *
2 from chat import chat_mignon
3
4 voir_matrice(chat_mignon)
```

Malheureusement, la matrice est corrompue, chaque valeur ayant été multipliée par 2. Par exemple une couleur (100, 255, 3) est codée dans la matrice `chat_mignon` par (200, 510, 6).

Créez une nouvelle matrice en corrigeant les valeurs des couleurs et affichez l’image correspondante.