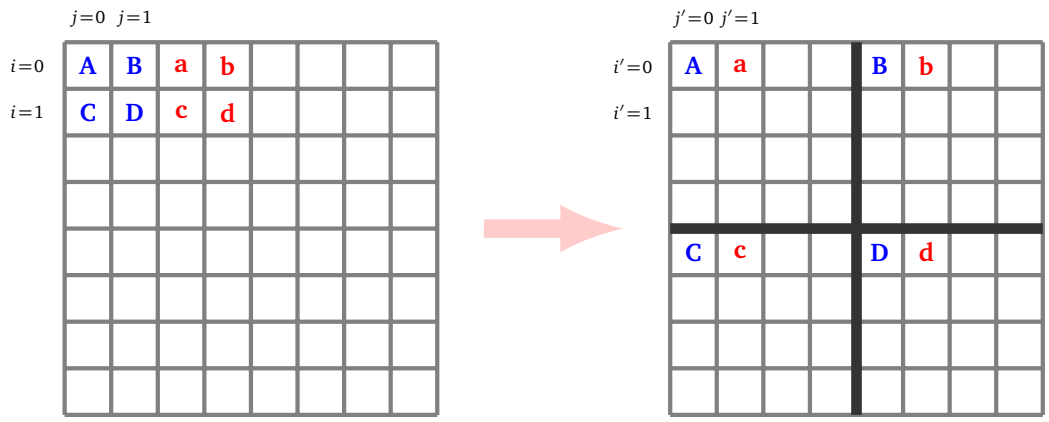
Images dynamiques

Nous allons déformer des images. En répétant ces déformations, les images deviennent brouillées. Mais par miracle au bout d'un certain nombre de répétitions l'image de départ réapparaît !

Vidéo ■ Images dynamiques - partie 1
Vidéo ■ Images dynamiques - partie 2

Cours 1 (Transformation du photomaton)**.**
On part d'un tableau $n \times n$, avec n pair, chaque élément du tableau représente un pixel. Les lignes sont indexées de $i = 0$ à $i = n-1$, les colonnes de $j = 0$ à $j = n-1$. À partir de cette image on calcule une nouvelle image en déplaçant chaque pixel selon une transformation, appelée ***transformation du photomaton***.
On découpe l'image de départ selon des petits carrés de taille 2×2. Chaque petit carré est donc composé de quatre pixels. On envoie chacun de ces pixels à quatre endroits différents de la nouvelle image : le pixel en haut à gauche reste dans une zone en haut à gauche, le pixel en haut à droite du petit carré, s'envoie dans une zone en haut à droite de la nouvelle image,...



Par exemple le pixel en position $(1,1)$ (symbolisé par la lettre **D**) est envoyé en position $(4,4)$.

Explicitons ce principe par des formules. Pour chaque couple (i,j), on calcule son image (i',j') par la transformation du photomaton selon les formules suivantes :

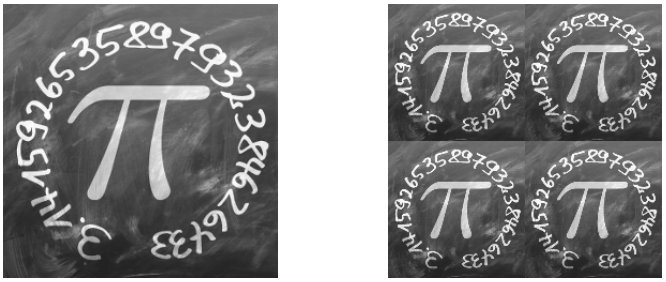
- Si i et j sont pairs : $(i',j') = (i//2, j//2)$.
- Si i est pair et j est impair : $(i',j') = (i//2, (n+j)//2)$.
- Si i est impair et j est pair : $(i',j') = ((n+i)//2, j//2)$.
- Si i et j sont impairs : $(i',j') = ((n+i)//2, (n+j)//2)$.

Voici un exemple d'un tableau 4×4 avant (à gauche) et après (à droite) la transformation du photomaton.

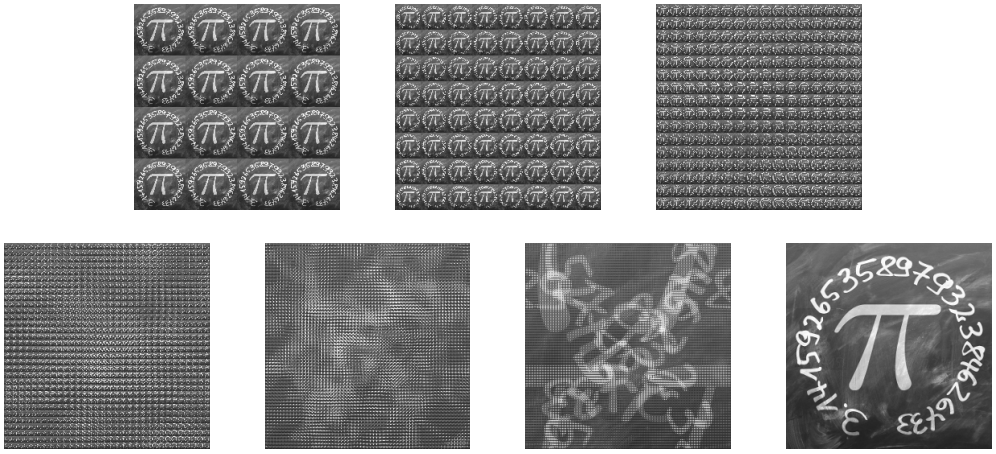
1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	3	2	4
9	11	10	12
5	7	6	8
13	15	14	16

Voici une image 256×256 et sa première transformation :



Voici ce qui se passe si on répète plusieurs fois la transformation du photomaton :



L'image devient de plus en plus brouillée, mais au bout d'un certain nombre de répétitions de la transformation, on retombe sur l'image de départ !

Activité 1 (Transformation du photomaton)**.**

Objectifs : programmer la transformation du photomaton qui décompose une image en sous-images. Lorsque l'on itère cette transformation l'image se déstructure petit à petit, puis d'un coup se reforme !

1. Programme une fonction `transformation(i,j,n)` qui met en œuvre la formule de la transformation du photomaton et renvoie les coordonnées (i', j') de l'image du pixel (i, j).

   Par exemple `transformation(1,1,8)` renvoie `(4,4)`.

2. Programme une fonction `photomaton(tableau)` qui renvoie le tableau calculé après transformation.

   Par exemple le tableau de gauche est transformé en le tableau de droite.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	3	2	4
9	11	10	12
5	7	6	8
13	15	14	16

Indications. Tu peux initialiser un nouveau tableau par la commande :

```
nouv_tableau = [[0 for j in range(n)] for i in range(n)]
```

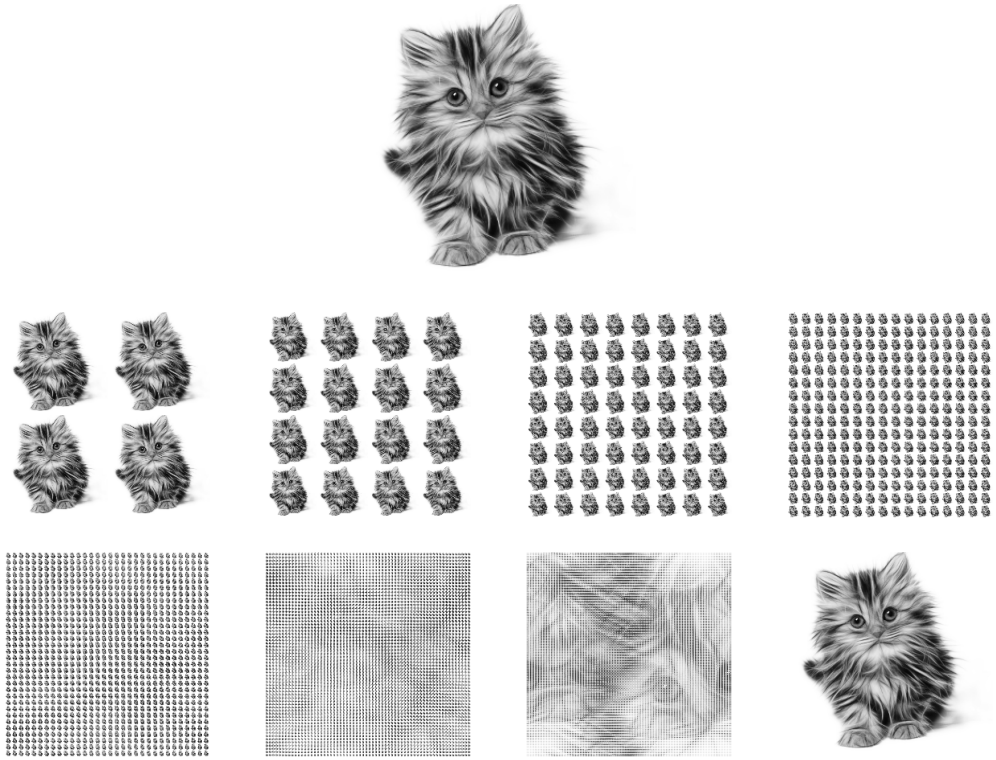
Puis le remplir par des commandes du type :

```
nouv_tableau[ii][jj] = tableau[i][j]
```

3. Programme une fonction `photomaton_iterer(tableau,k)` qui renvoie le tableau calculé après k itérations de la transformation du photomaton.
4. *À finir après avoir fait l'activité 2.*

   Programme une fonction `photomaton_images(nom_image,kmax)` qui calcule les images correspondant à la transformation du photomaton, pour toutes les itérations allant de $k = 1$ à $k = k_{\max}$.
5. Expérimente pour différentes valeurs de la taille n, afin de voir au bout de combien d'itérations on retrouve l'image de départ.

Voici l'image de départ de taille 256×256 et les images obtenues par itérations de la transformation du photomaton pour $k = 1$ jusqu'à $k = 8$. Au bout de 8 itérations on retrouve l'image initiale !



Activité 2 (Conversion tableau/image)**.**

Objectifs : passer d'un tableau à un fichier d'image et inversement. Le format pour afficher les images est le format « pgm » qui a été manipulé dans la fiche « Fichiers ».

1. **Tableau vers image.**

   Programme une fonction `tableau_vers_image(tableau,nom_image)` qui écrit un fichier image au format « pgm » à partir d'un tableau de niveaux de gris.

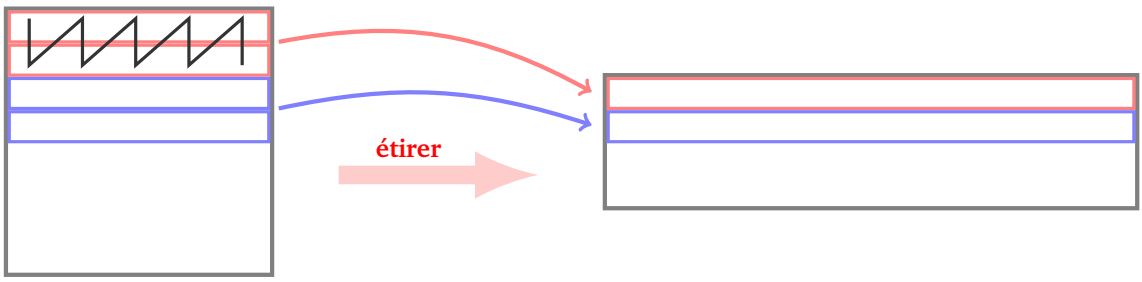
```
P2
5 5
255
128 192 128 192 128
224   0 228   0 224
228 228 228 228 228
224  64  64  64 224
192 192 192 192 192
```

Par exemple avec `tableau = [ [128,192,128,192,128], [224,...] ]`, la commande `tableau_vers_image(tableau,"test")` écrit un fichier `test.pgm` (à gauche) qui s'afficherait comme l'image à droite.

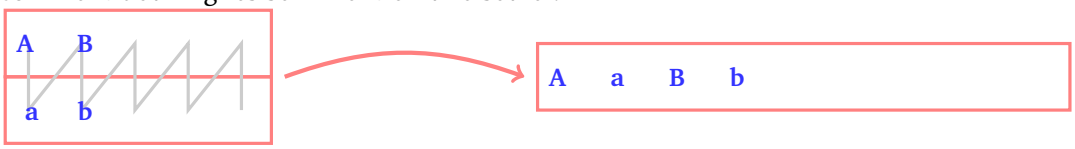
2. **Image vers tableau.**
   Programme une fonction `image_vers_tableau(nom_image)` qui à partir d'un fichier image au format « pgm », renvoie un tableau des niveaux de gris.

Cours 2 (Transformation du boulanger)**.**
On part d'un tableau $n \times n$, avec n pair dont chaque élément représente un pixel. On va appliquer deux transformations élémentaires à chaque fois :

- **Étirer.** Le principe est le suivant : les deux premières lignes (chacune de longueur n) produisent une seule ligne de longueur $2n$ en mixant les valeurs de chaque ligne en alternant un élément du haut, un élément du bas.



Voici comment deux lignes se mixent en une seule :



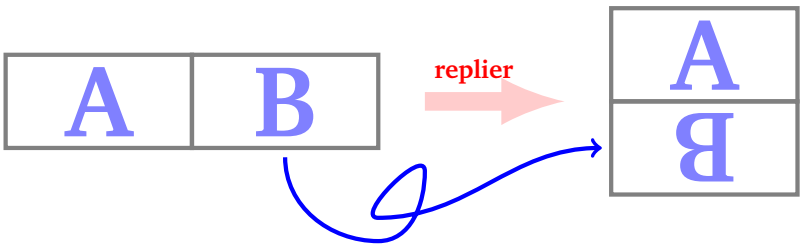
Formules. Un élément en position (i, j) du tableau d'arrivée, correspond à un élément $(2i, j//2)$ (si j est pair) ou bien $(2i+1, j//2)$ (si j est impair) avec ici $0 \leqslant i < \frac{n}{2}$ et $0 \leqslant j < 2n$.

Exemple. Voici un tableau 4×4 à gauche, et le tableau étiré 2×8 à droite. Les lignes 0 et 1 à gauche donnent la ligne 0 à droite. Les lignes 2 et 3 à gauche donne la ligne 1 à droite.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	5	2	6	3	7	4	8
9	13	10	14	11	15	12	16

- **Replier.** Le principe est le suivant : la partie droite d'un tableau étiré est retournée, puis ajoutée sous la partie gauche. Partant d'un tableau $\frac{n}{2} \times 2n$ on obtient un tableau $n \times n$.



Formules. Pour $0 \leqslant i < \frac{n}{2}$ et $0 \leqslant j < n$ les éléments en position (i, j) du tableau sont conservés. Pour $\frac{n}{2} \leqslant i < n$ et $0 \leqslant j < n$ un élément du tableau d'arrivée (i, j), correspond à un élément $\left(\frac{n}{2} - i - 1, 2n - 1 - j\right)$ du tableau de départ.

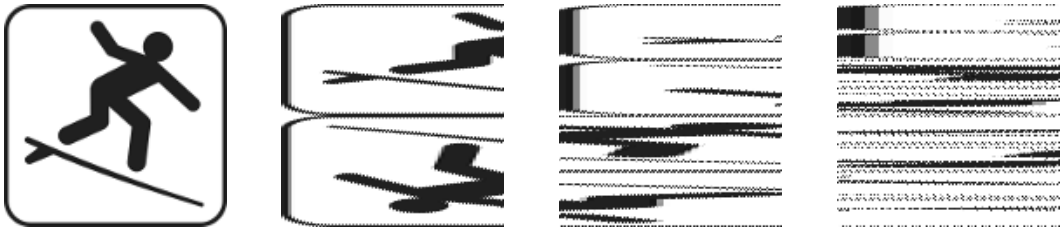
Exemple. À partir du tableau étiré 2×8 à gauche, on obtient un tableau replié 4×4 à droite.

1	5	2	6	3	7	4	8
9	13	10	14	11	15	12	16

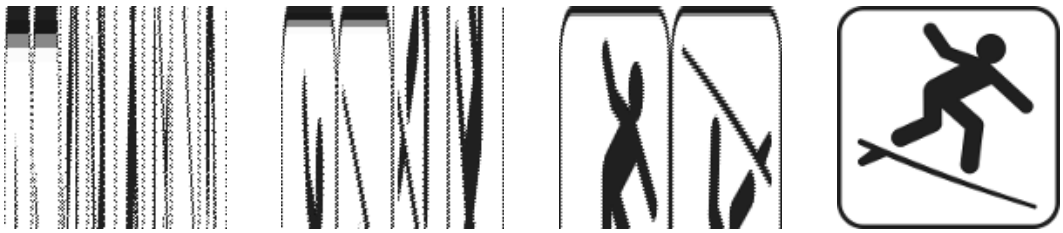
1	5	2	6
9	13	10	14
16	12	15	11
8	4	7	3

La ***transformation du boulanger*** est la succession d'un étirement et d'un repliement. Partant d'un tableau $n \times n$ on obtient encore un tableau $n \times n$.

Voyons un exemple de l'action de plusieurs transformations du boulanger. À gauche l'image initiale de taille 128×128, puis le résultat de $k = 1, 2, 3$ itérations.



Voici les images pour $k = 12, 13, 14, 15$ itérations :



Activité 3 (Transformation du boulanger)**.**

Objectifs : programmer une nouvelle transformation qui étire puis replie une image. Encore une fois l'image se déforme de plus en plus mais, au bout d'un certain nombre d'itérations, on retrouve l'image de départ.

1. Programme une fonction `boulanger_etirer(tableau)` qui renvoie un nouveau tableau obtenu en « étirant » le tableau donné en entrée.
2. Programme une fonction `boulanger_replier(tableau)` qui renvoie un tableau obtenu en « repliant » le tableau donné en entrée.
3. Programme une fonction `boulanger_iterer(tableau,k)` qui renvoie le tableau calculé après k itérations de la transformation du boulanger.

Par exemple, voici un tableau 4×4 à gauche, son image par la transformation ($k = 1$) et son image après une seconde transformation ($k = 2$).

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

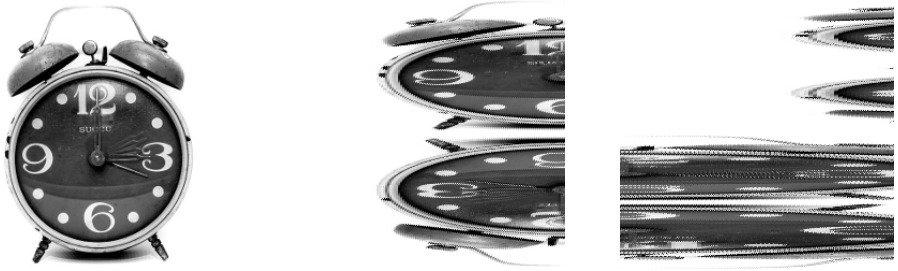
1	5	2	6
9	13	10	14
16	12	15	11
8	4	7	3

1	9	5	13
16	8	12	4
3	11	7	15
14	6	10	2

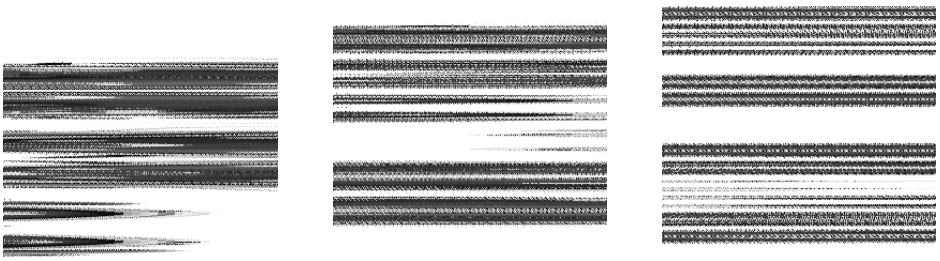
4. Programme une fonction `boulanger_images(nom_image,kmax)` qui calcule les images correspondant à la transformation du boulanger, avec des itérations allant de $k = 1$ à $k = k_{\max}$.
5. Expérimente pour différentes valeurs de la taille n, afin de voir au bout de combien d'itérations on retrouve l'image de départ.

   Attention ! Il faut parfois itérer beaucoup avant de retrouver l'image de départ. Par exemple avec $n = 4$, on retrouve l'image de départ au bout de $k = 5$ itérations, avec $n = 256$ c'est $k = 17$. Conjecture une valeur de retour dans le cas où n est une puissance de 2. Par contre pour $n = 10$, il faut $k = 56\,920$ itérations !

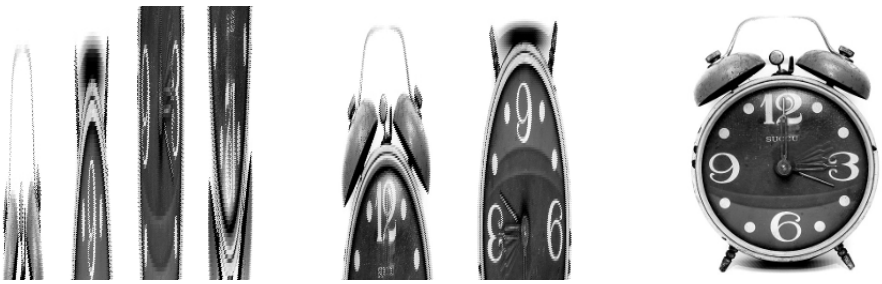
Voici un exemple avec une image de taille 256×256, d'abord l'image initiale, puis une première itération ($k = 1$) et une deuxième itération ($k = 2$).



$k = 3, 4, 5$:



$k = 15, 16, 17$:



Pour $k = 17$ on retrouve l'image de départ !

Cette fiche est basée sur l'article « Images brouillées, images retrouvées » par Jean-Paul Delahaye et Philippe

Mathieu (Pour la Science, 1997).