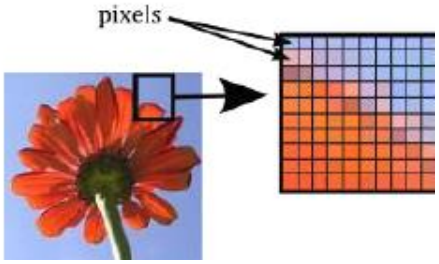
Représentation d'une image

Une image numérique est composée de pixels, arrangés en un rectangle.



Le pixel en haut à gauche est numéroté (0,0), celui en bas à droite $(H-1, L-1)$ si l'image est de hauteur H et de largeur L. La convention d'ordre des indices est la même que pour les matrices.
Dans notre cas, chaque pixel est décrit par la donnée de trois intensités de couleurs R, G et B qui sont des entiers entre 0 et 255. Ainsi chaque pixel est codé par une liste de longueur 3 [R, G, B] composée d'octets.
Nos objets images en python seront donc des listes de lignes de pixels c'est à dire des listes de listes de pixels ou encore des listes de listes de listes...
Pour charger une image depuis un fichier, nous disposons de la fonction `lit_image(chemin)` qui prend une chaîne de caractère représentant le chemin du fichier à charger comme argument.

```
image = lit_image('le chemin vers le fichier')
image[0][0] # retourne le premier pixel, une liste de longueur 3
image[0][0][1] # retourne la composante verte
```

Manipulations simple

Niveaux de gris

Le principe des images en noir et blanc est d'imposer aux trois couleurs d'avoir la même intensité. On choisit d'utiliser la moyenne de R, G, B. Remplir la fonction *noir_et_blanc* et la tester. On fera attention au type de donné utilisé par numpy : il s'agit de **uint8**, c'est à dire d'octets ou encore d'entiers non signés encodés sur 8 bits. Ainsi une somme ou un produit entre deux tels nombres peut très bien donner un résultat non spécifié : par exemple 145 + 212 n'est pas calculable dans ce type de données.
Remplir ensuite la fonction `noir_et_blanc2`.

Eliminer des couleurs

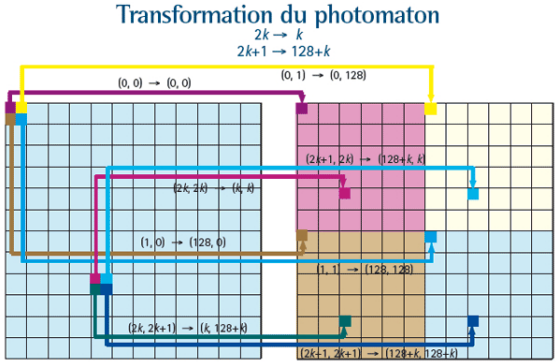
Jouons un peu avec les couleurs. On décide d'éliminer une, deux, ou trois couleurs dans une image. Pour se faire, il faut mettre son intensité à 0 sur tous les pixels. A vous de jouer !

Negatif

Pour l'expérimentation, afficher l'image `255-image` (taper ceci directement, si votre image s'appelle image, bien entendu, la magie de numpy fera le reste)

Photomaton

Dans ce paragraphe, on considère une image de taille 256×256. Le but est de transformer l'image de la manière suivante :



Notre image (lena.png) est de taille 512×512. Que deviennent les formules de transformation ? Implémenter cette transformation et l'effectuer 9 fois sur notre image.

Réduction de la palette, algorithme de Floyd

Le but de la transformation est cette fois de coder notre image avec beaucoup moins d'information. On décide de retenir 8 couleurs, c'est à dire que les intensité R, G, B ne peuvent valoir que 0 ou 255. La suite de ce paragraphe décrit la transformation à effectuer sur une couleur spécifique, c'est à dire qu'elle sera à répéter sur chaque couleur.
Notons $C = (c_{i,j})$ la matrice de l'image de départ (pour une seule couleur), et $s \in [\![0, 255]\!]$ un seuil qui nous servira à raffiner le résultat.
Pour chaque ligne à partir de 0 juqu'à $H-1$ et pour chaque colonne de 1 jusqu'à $L-1$ on définit $\widetilde{c_{i,j}} = 0$ si $c_{i,j} < s$ et $\widetilde{c_{i,j}} = 255$ sinon. Le problème de cette méthode est que l'on perd beaucoup d'information.

Notons $e = c_{i,j} - \widetilde{c_{i,j}}$ l'erreur commise.On redistribue cette erreur de la manière suivante :

$+0$	$\widetilde{c_{i,j}}$	$+\frac{5}{16}e$
$+\frac{3}{16}e$	$+\frac{5}{16}e$	$+\frac{1}{16}e$

Ceci suppose un parcours du tableau C dans le sens de la lecture : on modifie les cases non encore visitées.
Pour les lignes et colonnes du bord, on peut choisir de les effacer, ou d'ignorer les reports d'erreurs impossibles. Implémenter ceci et admirer le résultat. Utiliser plusieurs valeurs de seuil pour comparer