TP 8. Traitement d'images en niveau de gris.

Avant toute chose, créer un répertoire TP08 dans votre répertoire "Informatique", "TPsup" dans votre espace personnel.
Sur le site `https://ptsilamartin.github.io/info/TP.html`, télécharger les 2 images `allium.png`, `ciel.png` ainsi que le script `TP08.py` que vous copiez dans votre répertoire TP08. Ouvrir le script `TP08.py` dans lequel vous travaillerez tout au long de ce TP.

1 Créer des images en niveau de gris

Dans ce TP nous utilisons sur des **images matricielles** (formées de points, les pixels).
Le codage d'une image se fait alors sous la forme d'une matrice. Chaque pixel de l'image correspond à un élément de la matrice.
Ci-contre une image ne contenant que du noir (codé par un 0) et du blanc (codé par un 1).

1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	0	0	1	1	1
1	1	0	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	0	1
1	0	1	0	1	1	0	1	0	1
1	0	1	1	1	1	1	1	0	1
1	0	1	0	1	1	0	1	0	1
1	0	1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0	1	1
1	1	1	0	0	0	0	1	1	1
1	1	1	1	1	1	1	1	1	1

Les images dites en "noir et blanc" sont en fait des images en dégradé de gris (ou niveaux de gris). Chaque élément de la matrice correspondra comme précédemment à un pixel, mais ces éléments seront des flottants (compris entre 0. et 1.) donnant l'intensité de gris voulue. On utilise ici le 0 pour le noir et le 1 pour le blanc. Une valeur de 0.5 sera un gris moyen, de 0.2 un gris foncé et de 0.8 un gris clair.

Question 1. Compléter les commentaires de la fonction `image_noire` fournie dans le script `TP08.py`. Appeler ensuite cette fonction pour créer une image noire de largeur 6 pixels et de hauteur 3 pixels.

Question 2. Afficher la matrice obtenue dans le script. Afficher ensuite l'image obtenue dans une figure `matplotlib` comme indiqué ci-dessous.

```
import matplotlib.pyplot as plt #importation de la bibliothèque matplotlib

plt.figure("NOM␣de␣la␣figure") #crée une nouvelle figure
plt.imshow(im,cmap='gray',vmin=0, vmax=1) #affiche l'image codée par la matrice im / en
    niveau de gris, les pixels étant codés par des valeurs allant du 0 au 1
```

Question 3. Ecrire les fonctions `rayure_hor` et `rayure_ver` qui créent des images rayées horizontalement et verticalement. Tester ces fonctions sur des images de hauteur 9 et de largeur 14. Afficher les résultats obtenus.

Question 3. BIS (OPTIONNELLE : si vous êtes en avance uniquement) Ecrire la fonction `echiquier` qui crée une image avec un motif d'échiquier.Tester et afficher l'image obtenue.

2 Eclaircir une image

Dans cette partie on souhaite éclaircir l'ensemble d'une image.

Question 4. Ecrire la fonction `eclaicir` qui prend pour argument la matrice-image `im` et le flottant `k` et qui modifie chaque pixel de l'image pour éclaircir d'une valeur `k` chaque pixel. (plus la valeur de `k` sera élevée plus l'éclaircissement sera conséquent).
Tester ensuite votre fonction sur l'image rayée horizontalement que vous avez obtenue précédemment. Les bandes noires doivent devenir grises.
Afficher 3 figures : l'image rayée initiale, l'image rayée éclaicie et l'image rayée initiale après application de la fonction `eclaicir`.

Si on ne prend pas de précautions particulières, le fait d'apliquer la fonction `eclaicir` *modifie l'image source. Ceci peut être génant si ce n'est pas volontaire. Il faut bien veiller à créer une* ***copie indépendante*** *de la matrice-image source pour éviter ce problème.*

Copie indépendante

Question 5. Réécrire la fonction `eclaicir2` en ajoutant les commandes permettant de réaliser une copie indépendante de la matrice-image source avant de la modifier. Tester en affichant les 3 figures de la question précédente.

Lecture d'une image contenue dans un fichier .png

On va maintenant appliquer cette fonction sur une autre image, contenue dans le fichier **allium.png**. Pour lire un fichier et obtenir la matrice-image correspondante on utilise l'instruction :

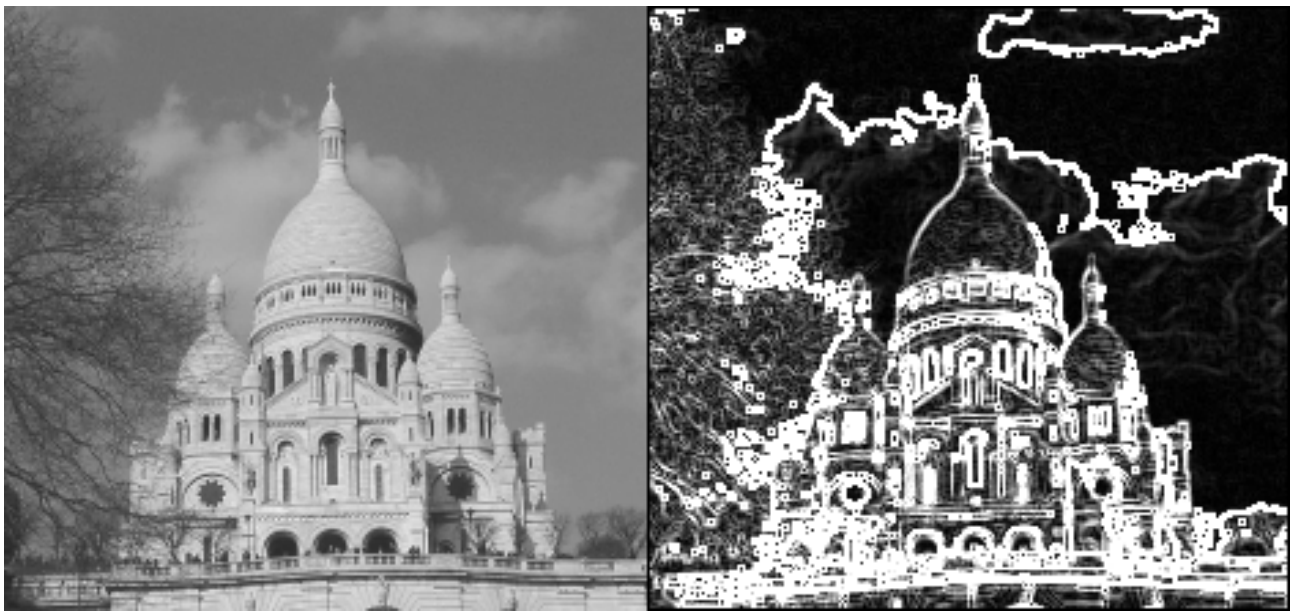
```
import matplotlib.image as img #import d'une autre bibliothèque de matplotlib
imF=img.imread("nom_du_fichier.png") #lecture du fichier
```

Attention : Il faut que le fichier lu se situe dans le même répertoire que le script. Veillez donc à ce que vos fichiers `allium.png` et `TP08.py` soient dans le même répertoire. Il faudra ensuite exécuter le script en effectuant un clic droit sur l'onglet de votre script puis "éxecuter en tant que script".

Question 6. Lire le fichier **allium.png** et stocker la matrice image obtenue dans la variable **imF**. Afficher cette image. Eclaircir l'image d'une valeur de 0.2 par exemple. Observer l'image obtenue. Vérifier que la matrice-image source **imF** n'est pas modifée.

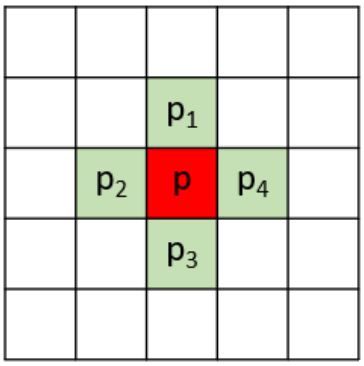
3 Détecter les contours

La reconnaissance de formes dans une image est une composante importante de l'analyse d'images. Elle se décompose en plusieurs étapes qui consistent à extraire les contours des objets dans l'image afin de les reconnaitre ou d'en détecter le mouvement. La première de ces étapes est la mise en évidence des contours des objets dans l'image. C'est cette étape que nous allons aborder très succinctement.



Un contour définit la limite d'un objet dans une image. Cette limite est caractérisée par un changement dans l'image : un changement de couleur ou de contraste. Ce changement se traduit dans la valeur des pixels qui sont localisés de part et d'autre de la limite. Nous sommes donc à la recherche d'un moyen de détecter et de localiser un changement.

Considérons un pixel **p** dans une image. Ce pixel est-il semblable, de même teinte, que ses voisins ? Si non, quelle est la différence de teinte entre lui et ses voisins ? Est-elle grande, ce qui signifierait qu'il est situé à la limite d'un objet ?
Le principe est pour chaque pixel de l'image, de récupérer la valeur de chaque pixel avoisinant ce pixel, puis de mesurer la différence, la "distance" euclidienne, entre ces valeurs en utilisant une fonction de norme standard :



$$distance = \sqrt{(p_1 - p_3)^2 + (p_2 - p_4)^2} \quad (1)$$

où p_1, p_2, p_3 et p_4 sont les valeurs des 4 pixels voisins.

Après avoir calculé la distance entre les valeurs des voisins du pixel courant, on compare cette distance à une valeur seuil :

- si la distance est plus petite ou égale que le seuil : la variation est faible : ce pixel n'est pas considéré sur un contour : on le laisse en noir ;
- si la distance est plus grande strictement que le seuil : la variation est grande : ce pixel est sur le contour : on le trace en blanc.

Question 7. Lire l'image `ciel.png`. Afficher cette image dans une nouvelle figure en utilisant le module `matplotlib.pyplot`.

Question 8. Écrire une fonction `contour`, qui prend comme argument une image en niveaux de gris `im` ainsi qu'un flottant `seuil`, et qui renvoie une image noire sauf les pixels situés sur le contour qui seront en blanc.

Question 9. Tester cette fonction sur l'image `ciel.png`, avec un seuil de 0.2.