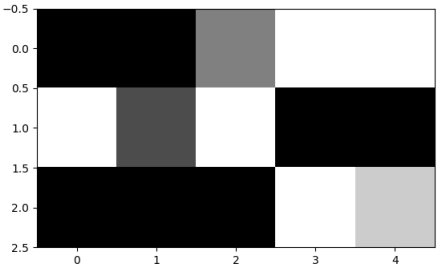
Introduction au traitement d'images — TP18

Les matrices sont un moyen de coder des images en Python. Chaque coefficient de la matrice correspond à un pixel et indique la couleur du pixel. Dans ce TP on se limite à des images en noir et blanc, on utilise pour cela des coefficients compris entre 0 et 1 avec la convention : 0 = noir, 1 = blanc (et toute valeur comprise entre 0 et 1 correspond à une nuance de gris). D'autres façons de coder les images existent cependant [1].
Voici un exemple de correspondance entre matrice et image :

$$\begin{pmatrix} 0 & 0 & 0,5 & 1 & 1 \\ 1 & 0,3 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0,8 \end{pmatrix}$$ correspond à



Pour afficher l'image correspondant à une matrice `matrice` on utilisera les commandes suivantes faisant appel à la bibliothèque `matplotlib.pyplot` :

```
plt.imshow(matrice,cmap='gray') # cmap pour color map
plt.show()
```

Exercice 1 — Construction d'une image à la main

Q1 Écrire une fonction `nuancier` prenant en arguments deux entiers n et p et renvoyant la matrice M de taille $n \times p$ dont les coefficients sont données par :

$$\forall i \in [\![0, n-1]\!],\ \forall j \in [\![0, p-1]\!],\ M_{i,j} = \frac{i+j}{n+p-2}.$$

Q2 Afficher à l'écran l'image correspondant à la matrice `nuancier(10,5)` puis à la matrice `nuancier(100,50)`. Est-ce bien ce à quoi vous vous attendiez ?

Exercice 2 — Préparation pour le traitement

Une image test vous est fournie sur cahier de prépa, il s'agit du fichier `zebre.png`, téléchargez-le et enregistrez-le dans un dossier de votre espace personnel (là où vous enregistrez habituellement vos TPs).

Pour pouvoir utiliser ce fichier dans Python, il faut l'importer en précisant son emplacement, c'est-à-dire la suite de dossiers à ouvrir pour accéder à ce fichier. On utilise pour cela la bibliothèque `matplotlib.image` :

```
import matplotlib.image as mpimg # Pour ouvrir et manipuler des images
```

1. On peut par exemple manipuler des images dont les pixels peuvent prendre des valeurs entières entre 0 et 255 (0 pour noir, 255 pour blanc). On peut aussi manipuler des images en couleurs grâce au code "RGB" (rouge, vert, bleu), on utilise alors 3 images en parallèle (codant les niveau de rouge, vert, et bleu) qu'on associe pour recomposer la couleur de chaque pixel.

La commande `imread` permet alors d'importer l'image dans Python sous forme d'un tableau de réels. Cette commande prend en argument une chaîne de caractères, correspondant à l'emplacement du fichier à ouvrir, et renvoie le tableau des valeurs des pixels de l'image. Par exemple la commande

```
1 mon_image_RGB = mpimg.imread(r"\\SRV-DC\ccaillaud$\Documents\zebre.png")
```

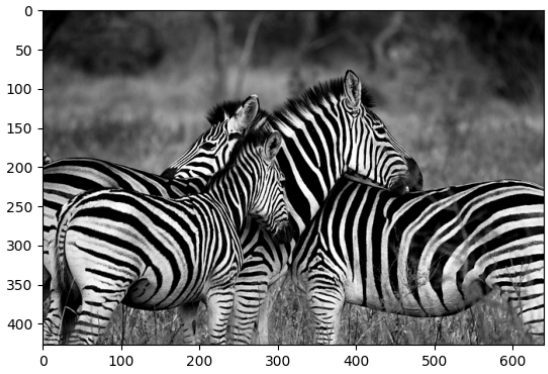
permet d'ouvrir le fichier `zebre.png` si l'utilisateur `ccaillaud` l'a rangé dans son dossier `Documents`. Pour récupérer l'emplacement de votre fichier, utilisez l'explorateur de fichier et faites "clic droit" sur le fichier en question. Copiez alors son emplacement pour créer votre chaîne de caractères. Attention, il faut également faire précéder cette chaîne d'un `r`.

La fonction `imread` crée naturellement un tableau permettant de stocker une image en couleur, comportant donc plus de dimensions que nécessaire pour notre TP. On utilise donc la commande `mon_image = mon_image_RGB[:,:,0]` pour récupérer simplement un tableau de taille "$n \times p$" comme manipulé dans l'exercice précédent.

Pour afficher une image, on utilise la commande `imshow`, avec l'argument `cmap='gray'` (pour préciser qu'il s'agit d'une image en niveaux de gris) puis `show`.

```
1 mon_image = mon_image_RGB[:,:,0] # pour avoir un simple tableau n x p
2 plt.imshow(mon_image, cmap='gray') # dessiner l'image
3 plt.show() # afficher l'image
```

Vous devez obtenir le résultat suivant :

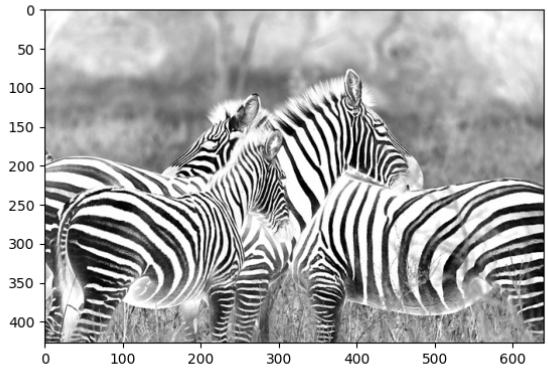


Q1 Souhaitez-vous vraiment continuer à appeler cette image `mon_image` pour tout le reste du TP ? Choisissez un nom parlant et simple.

Q2 Quelle est la taille de cette image ? Combien compte-t-elle de pixels en tout ?

Exercice 3 Négatif

On souhaite inverser les couleurs de l'image afin d'obtenir son négatif.

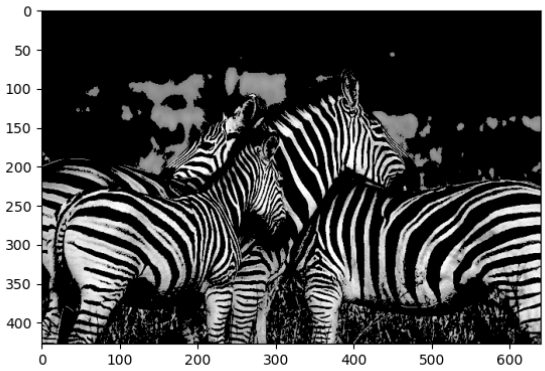


Q1 Écrire une fonction `pixel_negatif` prenant en argument la valeur d'un pixel correspondant à un niveau de gris et renvoyant la valeur correspondant à sa couleur négative.

Q2 Écrire et tester une fonction `negatif` prenant en argument une image quelconque et renvoyant son négatif. Pour nos zèbres, on obtient l'image ci-contre.

Exercice 4 Seuillage

Pour simplifier le traitement d'une image en niveaux de gris, on peut décider d'assimiler ses pixels les plus sombres à des pixels noirs. On réalise alors un seuillage de l'image : les pixels d'intensité inférieure à un certain seuil donné voient leur valeur mise à 0.



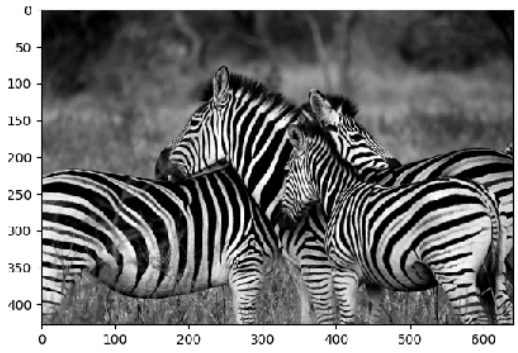
Q1 Écrire une fonction `pixel_seuil` prenant en argument une valeur `seuil`, et la valeur `x` correspondant à un pixel, et renvoyant la valeur de ce pixel dans l'image seuillée.

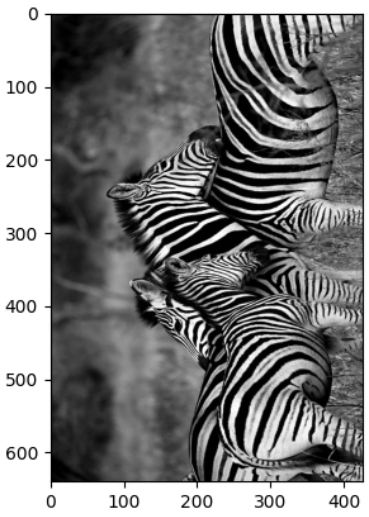
Q2 Écrire et tester une fonction `seuillage` prenant en argument une image, une valeur `seuil` et renvoyant l'image seuillée. Pour la valeur `seuil = 0.5`, vous devez obtenir l'image ci-contre (←).

Exercice 5 Miroir et rotation

Q1 Écrire une fonction `miroir` permettant de retourner une image horizontalement. Vous devez obtenir le résultat de gauche ci-dessous.

Q2 Écrire une fonction `rotation90gauche` renvoyant une image ayant subit une rotation de 90° dans le sens anti-horaire (et sans effet miroir !). Vous devez obtenir le résultat de droite ci-dessous.





Exercice 6 Un peu de dessin

Dans ce dernier exercice, on propose de dessiner des figures géométriques dans des images. On produira des images carrées de taille $N \times N$ qui représenteront des figures comprises dans le carré $[-2,2]\times[-2,2]$ de $\mathbb{R}^2$.

Les figures seront dessinées sur une variable image globale, et la taille N sera aussi une variable globale, c'est-à-dire qu'on écrira en début de script :

```
N = 1000
dessin = np.zeros((N,N))
```

Afin d'utiliser les équations algébriques usuelles des figures géométriques telles que le disque, il faut être en mesure de transformer les indices d'un tableau en abscisses et ordonnées. Par exemple, le pixel d'indices $[N/2, N/2]$ au centre de l'image correspond à l'origine du repère $(0, 0)$. On admet qu'un indice `k` $\in [\![0, N-1]\!]$ correspond à une abscisse ou une ordonnée `abs_ord(k)` où `abs_ord` est la fonction suivante, que l'on recopiera :

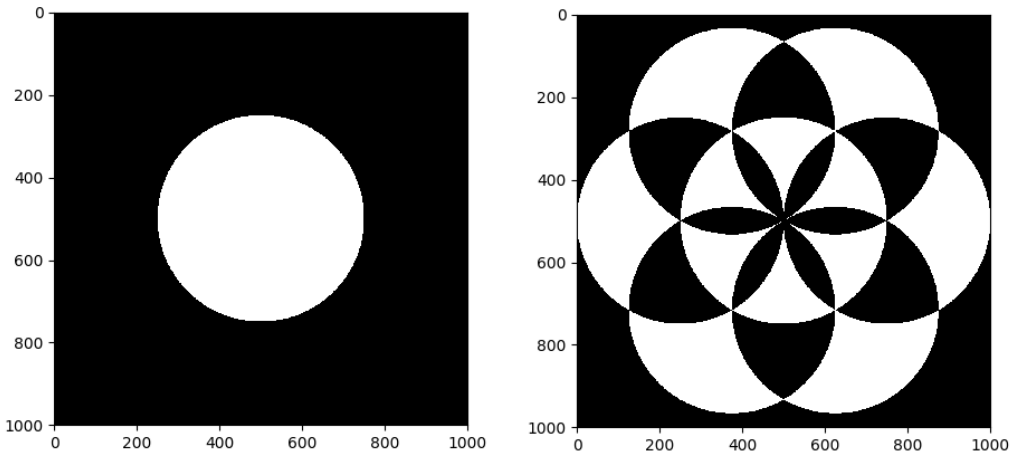
```
1 def abs_ord(k):
2     return (k-N/2)*(4/N)
```

On rappelle que le disque de centre $A(x_A, y_A)$ et de rayon r est l'ensemble suivant :

$$D(A, r) = \{(x, y) \in \mathbb{R}^2 \ : \ (x - x_A)^2 + (y - y_A)^2 \leq r^2\}$$

Q1 Écrire une fonction `disque` permettant de dessiner un disque de centre (x_A, y_A) et de rayon r. Par exemple, avec $x_A = y_A = 0$ et $r = 1$, on doit obtenir l'image ci-dessous à gauche.

Q2 Reproduire le dessin de rosace ci-dessous à droite.



Pour rendre votre TP :

- *Inscrivez votre nom en haut du script et enregistrez votre fichier,*
- *dans Spyder, cliquez sur "Tools" puis "Preferences",*
- *dans le cadre "syntax highlighting theme" choisir "Minimal" dans le menu déroulant,*
- *cliquez sur "OK". Spyder demande si vous souhaitez redémarrer, cliquez sur "Non",*
- *dans Spyder, cliquez sur "File" puis "Print",*
- *sélectionnez "Microsoft Print to PDF" puis enregistrez ce pdf avec pour nom : `TP16-NOM-Prenom`*
- *déposez votre fichier sur cahier de prépa, onglet "Informatique : transfert de document" OU enregistrez-le dans Ce PC > Ma classe > Restitution de devoir*