
Cheat sheet numpy

```
import numpy as np

# Transforme une liste L en matrice numpy
np.array(L)

# Crée une matrice ligne de n valeurs uniformément réparties entre a et b (inclus)
np.linspace(a, b, n)

# Crée la matrice nulle de taille n x m
np.zeros((n, m))

# Crée la matrice remplie de 1 de taille n x m
np.ones((n, m))

# Crée la matrice identité de taille n
np.eye(n)

# Crée la matrice diagonale dont les termes diagonaux sont les éléments de la liste L
np.diag(L)

# Calcule de la transposée de la matrice M
np.transpose(M)
M.T

# Multiplication matricielle de M par P
np.dot(M, P)
M @ P

# Somme de tous les éléments de M
np.sum(M)
M.sum()

# Le plus grand des éléments de M
np.max(M)
M.max()

# Le plus petit des éléments de M
np.min(M)
M.min()

# Renvoie dans un couple le format de la matrice M
np.shape(M)

# Renvoie le nombre d'éléments de M
np.size(M)

# Accéder à l'élément (i, j) de M
M[i, j]

# Accéder à la i-ième ligne de M
M[i, :]

# Accéder à la j-ième colonne de M
M[:, j]
```

Images en noir et blanc

Qu'est-ce qu'une image

Une image en noir et blanc en informatique est tout simplement une grille de pixels gris. Un pixel gris contient une unique valeur entre 0 (noir) et 255 (blanc) (voir figure 10.1). Par exemple, un pixel de valeur 50 sera gris foncé, et un pixel de valeur 200 sera gris très clair.

(a) une image grise

(b) zoom sur le carré rouge

FIGURE 10.1 – Une image est une grille de pixels

Une image est donc naturellement représentée par une matrice M d'entiers entre 0 et 255. Nous allons utiliser les matrices de `numpy` pour représenter et manipuler les images. En python, le coin en haut à gauche de l'image correspond au coefficient `M[0, 0]` de la matrice, comme en mathématiques. Mais attention, comme en mathématiques, le premier indice correspond à l'axe des y et le second à l'axe des x. On essaye de toujours avoir figure 10.2 en tête. Moyen mnémotechnique : les pixels sont organisés comme lorsqu'on `print(M)`.

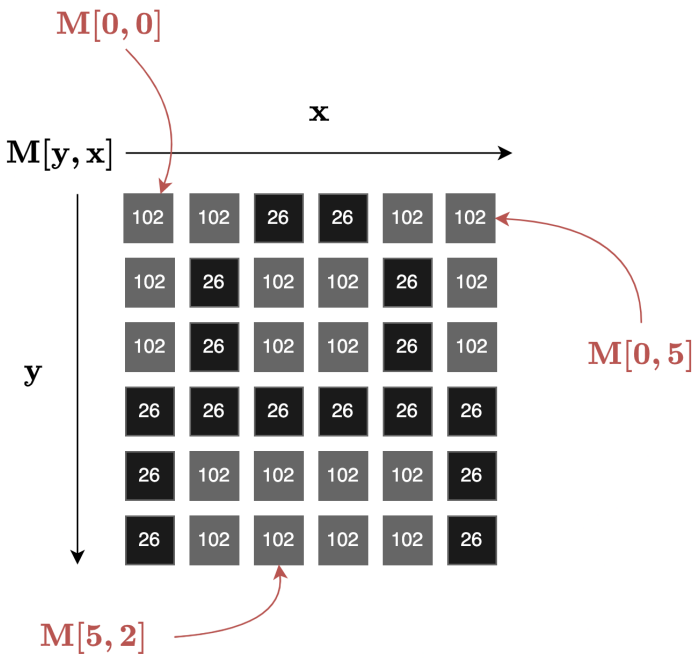

FIGURE 10.2 – Convention image

Importer, affichage, manipuler, sauver

Modules utiles On utilise le module `PIL` pour ouvrir, sauvegarder et afficher des images. On utile `numpy` pour les manipuler.

Écrire dans la console de `pyzo`

```
pip install Pillow
```

Puis on écrit en début de fichier

```
from PIL import Image
import numpy as np
```

Importer une image Téléchargez l'image à l'adresse https://cutt.ly/T4j2eT6 et enregistrez là dans le même dossier que votre fichier python de travail, au nom `poivrons.png`. Vous pouvez aussi utiliser vos propres images si vous le souhaitez. On peut l'importer dans python avec

```
img = Image.open('poivrons.png')
```

Afficher une image On une utilise la méthode `show()` :

```
img.show()
```

Obtenir la valeur des pixels sous la forme d'un tableau `numpy` Il faut convertir l'image en tableau numpy *et le copier*, sinon on ne peut pas modifier les pixels.

```
img_grey = img.convert('L') # conversion en image grise
arr = np.asarray(img_grey).copy()
```

Repasser d'un tableau `numpy` à une `Image`

```
img = Image.fromarray(arr)
```

sauvegarder une `Image`

```
img.save('poivrons_gris.png', format='PNG')
```

1. Vérifiez que vous arrivez à exécuter les étapes ci-dessus pour finalement produire une version grise de l'image `poivrons.png`.

Manipulation basique d'image

2. Créez les images figure 10.3

(a) Image mirroir (b) Image en négatif (c) Moitié gauche

FIGURE 10.3 – Manipulations basiques d'images

Jeu sur la luminosité

Le but de cette question est d'éclaircir une image. Pour éclaircir une image, il faut globalement augmenter la valeur des pixels. Mais attention, on ne peut pas juste ajouter une constante car la valeur des pixels ne doit pas dépasser 255. À la place, il faut trouver une fonction

$$eclair : \begin{cases} [0,255] & \to [0,255] \\ pixel & \mapsto \quad ? \end{cases}$$

telle que $eclair(x) \geqslant x$.

3. a) Donner une fonction qui ressemble à celle figure 10.4

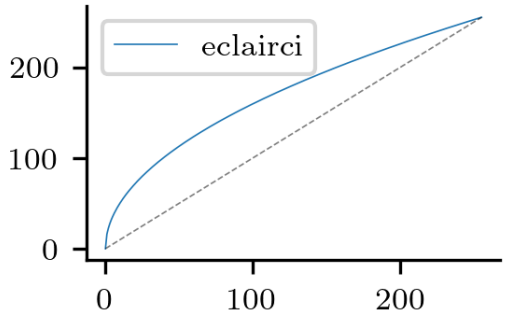

FIGURE 10.4 – Fonction d'éclaircissement

b) Appliquer la fonction à chaque pixel de l'image et vérifier qu'on obtient une image éclaircie, similaire à 10.5a. Attention, par défaut lorsqu'on lit une `Image` dans un tableau `numpy`, les valeurs sont de type `np.uint8`, c'est à dire des entiers de $[\![0, 255]\!]$ et les calculs se font alors dans $\mathbb{Z}/256\mathbb{Z}$. Pour sortir de ce mode de calcul, il faut faire une conversion explicite vers `np.float64` en début de calcul, puis reconvertir en `np.uint8` à la fin des calculs.

```
arr = arr.astype(np.float64)
... # opérations sur arr
arr = arr.astype(np.uint8)
```

(a) Image éclaircie (b) Image assombrie

FIGURE 10.5 – Changement de luminosité

c) Trouver une transformation pour obtenir une image cette fois-ci assombrie, comme 10.5b.

Jeu sur les contrastes

Une image est très contrastée lorsque les zones sombres sont très sombres et les zones claires très claires, c'est-à-dire que l'ensemble des valeurs des pixels a une grande variance. À l'inverse, une image est peu contrastée si elle est globalement grise, c'est-à-dire que l'ensemble des valeurs des pixels a une faible variance.

4. a) Trouver deux fonctions dont le graphe ressemble à ceux figure 10.6

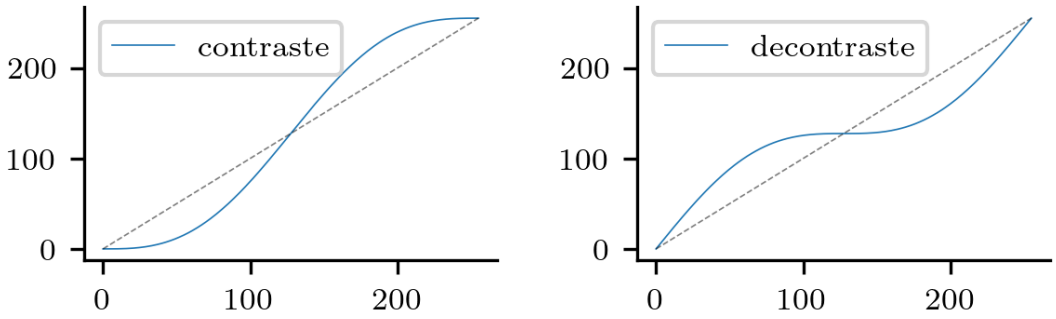

FIGURE 10.6 – Fonctions de contrastes

b) Recréer les images figure 10.7

(a) Image contrastée

(b) Image peu contrastée

FIGURE 10.7 – Changement de contrastes