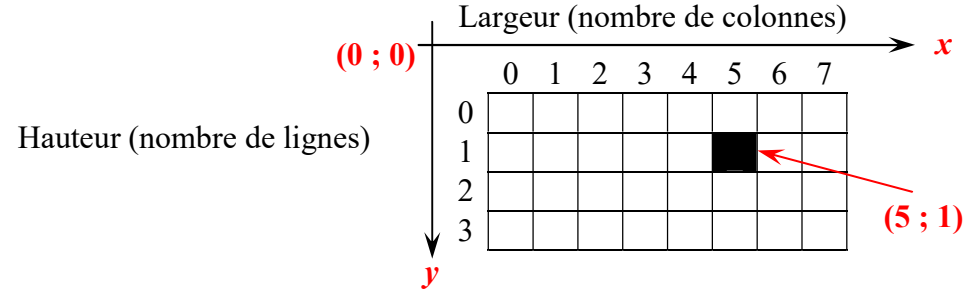
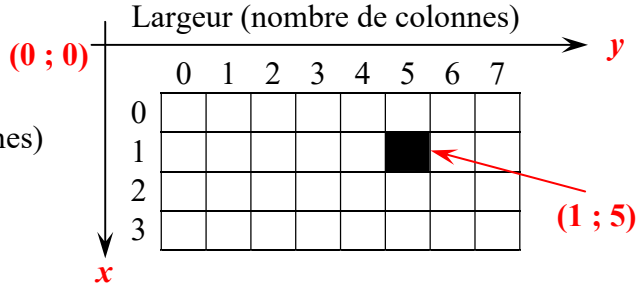
Stéphane KELLER – Lycée agricole Louis Pasteur

TRAITEMENT DES IMAGES

I. Module PIL : quelques commandes utiles.

imgpil.convert("L")	Permet de convertir une image RGB en mode niveaux de gris.
imgpil.convert("RGB")	Permet de passer du mode niveaux de gris au mode RGB.
imgpil.copy()	Crée en mémoire une copie de l'image d'origine. Utiliser la syntaxe imgpil2 = imgpil.copy()
imgpil.getpixel((x, y))	Donne la couleur du pixel (x, y). Pour une image en niveaux de gris ou en 256 couleurs, la couleur correspond à un entier compris entre 0 et 255 Pour une image couleur, c'est un triplet de valeurs numériques (R, V, B) comprises entre 0 et 255.
imgpil.mode	Renvoie le mode de l'image. Le mode d'une image définit le type et la profondeur d'un pixel dans l'image. La version actuelle prend en charge les modes standard suivants : ➢ 1 (pixels 1 bit, noir et blanc, stockés avec un pixel par octet) ➢ L (pixels 8 bits, noir et blanc) ➢ P (pixels de 8 bits, mappés sur un autre mode à l'aide d'une palette de couleurs) ➢ RVB (pixels 3x8 bits, couleur vraie) ➢ RGBA (pixels 4x8 bits, couleur vraie avec masque de transparence) ➢ CMJN (pixels 4x8 bits, séparation des couleurs) ➢ YCbCr (pixels 3x8 bits, format vidéo couleur) ➢ I (pixels entiers signés 32 bits) ➢ F (pixels à virgule flottante 32 bits)
imgpil.open("nom_fichier.jpg")	Ouvre le fichier dans le répertoire de travail
imgpil.putpixel((x, y), color)	Met le pixel (x, y) à la couleur indiquée. Pour une image en couleur, la couleur est un triplet de valeurs numériques (R, V, B)
imgpil.save("nom_fichier.jpg")	Sauvegarde le fichier dans le répertoire de travail
imgpil.show()	Affiche l'image. on peut ajouter d'autres paramètres : ➢ cmap='gray' pour que l'affichage soit en niveaux de gris interpolation='nearest' : par défaut imshow() fait un dégradé de couleur entre les points du tableau A, ce qui ne correspond pas nécessairement au résultat escompté, surtout pour des tableaux de petite dimension ; pour palier cela il faut ajouter l'option interpolation='nearest')
imgpil.size	renvoie le tuple (nombre de colonnes, nombre de lignes) correspondant à la définiton de l'image.
imgpil.split()	La méthode Image.split() est utilisée pour diviser l'image en bandes individuelles. Cette méthode renvoie un tuple de bandes individuelles à partir d'une image. La division d'une image "RGB" crée trois nouvelles images contenant chacune une copie d'une des bandes originales (rouge, vert, bleu).
Image.new(mode, (largeur,hauteur), color = couleur)	Permet de créer image. On choisira comme mode : ➢ "1" pour avoir une image en noir et blanc (1 bit par pixel) ➢ "L" pour une image en nuances de gris. ➢ "RGB" pour une image en couleurs (rouge, vert, bleu). "couleur" est la couleur de fond : soit une valeur numérique, soit un triplet de valeurs numériques (R, V, B).

PIL.Image	matplotlib.pyplot
Coordonnées	
	
Importation du ou des modules	
from PIL import Image import numpy as np	import matplotlib.image as mpimg import numpy as np import matplotlib.pyplot as plt
Lecture de l'image et transformation en tableau numpy	

imgpil = Image.open("fichier.png") #imgpil est de type PngImageFile
A = np.array(imgpil) #A est de type numpy.ndarray
	A = mpimg.imread("fichier.png ") #A est de type numpy.ndarray
	if A.dtype == np.float32: #si le résultat n'est pas un tableau d'entiers
	A = (A * 255).astype(np.uint8)

Affichage de l'image	
imgpil.show()	plt.imshow(A) #affiche les données sous forme d'image plt.show() #affiche l'image correspond au tableau numpy A
Conversion d'une image PIL en tableau numpy	
imgpil = Image.fromarray(A) #transformation du tableau numpy en image PIL	A = np.array(imgpil) #transformation de l'image PIL en tableau numpy
Création d'une image RGB vierge	
imgpil = Image.new("RGB", (largeur, hauteur), (r, v, b))	A = np.zeros((hauteur, largeur, 3), dtype=np.uint8)
Propriétés d'une image	
imgpil.size #retourne le tuple(largeur, hauteur)	A.size #retourne le produit hauteur × largeur A.shape #retourne le tuple(hauteur, largeur, [nombre de paramètres : # 3 pour (R, G, B) ou 4 pour (R, G, B, transparence)])
Lecture d'un pixel : imgpil.getpixel((x, y)) = A[y, x]	
imgpil.getpixel((x, y))	A[y, x]
Écriture d'un pixel : imgpil.putpixel((x, y)) = A[y, x]	
imgpil.putpixel((x, y), (valeur_r, valeur_g, valeur_b, [valeur_transparence]))	A[y, x] = (valeur_r, valeur_g, valeur_b, [valeur_transparence])
Copie d'une image	
Imgpil2 = imgpil.copy()	B = np.copy(A) #copie du tableau numpy dans un autre tableau
Sauvegarde d'une image	

imgpil = Image.fromarray(A) #Transformation du tableau en image PIL
imgpil.save("resultat.jpg")
	mpimg.imsave("resultat.png", A)

II. Module numpy et matplotlib.

2.1 Importation.

```
import numpy as np
```

Ici l'objectif sera toujours de transformer une image en tableau numpy, pour pouvoir ensuite la manipuler. Le sous module Image du module matplotlib, matplotlib.image ne permet que de charger des images au format **png**. Mais vous avez directement un tableau numpy (pas besoin de transformation).

2.2 Lecture de l'image et transformation en tableau numpy.

2.2.1 Avec le module PIL.

```
from PIL import Image
import numpy as np

imgpil = Image.open("Joconde_384.png")
# anciennement np.asarray
A = np.array(imgpil) # Transformation de l'image en tableau numpy
```

2.2.2 Avec le module matplotlib.image.

```
import matplotlib.image as mpimg
import numpy as np

A = mpimg.imread("Joconde_384.png")
```

L'objet img est un tableau numpy. C'est parfois un tableau de float (entre 0 et 1) ou parfois un tableau d'octets. On peut être amenés à faire une transformation :

```
if A.dtype == np.float32: #si le résultat n'est pas un tableau d'entiers
    A = (A * 255).astype(np.uint8)
```

2.3 Affichage de l'image.

2.3.1 Avec le module PIL.

On rajoute l'instruction :

```
imgpil.show() #affichage de l'image avec le module PIL
```

2.3.2 Avec le module matplotlib.image.

On rajoute les instructions :

```
import matplotlib.pyplot as plt
plt.imshow(A)
plt.show() #affichage de l'image avec le module matplotlib.pyplot
```

L'affichage est normalement plus rapide qu'avec le module PIL.

2.4 Création d'une image vierge.

2.4.1 Avec le module PIL.

```
from PIL import Image
```

```
imgpil = Image.new("RGB", (100, 200), (0, 0, 0))
>>> imgpil.size
(100, 200)
```

2.4.2 Avec le module matplotlib.image.

```
import numpy as np

A = np.zeros((100, 200, 3), dtype=np.uint8)
>>> A.shape
(100, 200, 3)
>>> A.size
60000
```

2.5 Manipulation d'une image.

2.5.1 Avec le module PIL.

Modification d'un pixel :

```
imgpil.putpixel((x, y), ( valeur_r, valeur_g, valeur_b, [valeur_transparence]))
```

Exemple de manipulation d'une image avec la conception d'un filtre jaune :

```
def filtre_jaune(imgpil2):
    imgpil3 = imgpil2.copy() # On fait une copie de l'original
    nb_colonnes = imgpil3.size[0]
    nb_lignes = imgpil3.size[1]
    for i in range(nb_colonnes):
        for j in range(nb_lignes):
            r, g, b = imgpil3.getpixel((i, j))
            imgpil3.putpixel((i, j), (r, g, 0))
    return imgpil3

imgpil4 = filtre_jaune(imgpil)
imgpil4.show()
```

2.5.2 Avec le module matplotlib.image.

Modification d'un pixel :

```
A[y, x] = ( valeur_r, valeur_g, valeur_b, [valeur_transparence])
```

Dans le cas où une image possède 4 plans, il est possible de copier cette image dans une autre image en ne copiant pas le quatrième plan :

```
B = A[:, :, :3].copy()
```

Exemple de manipulation d'une image avec la conception d'un filtre bleu :

```
def filtre_bleu(C):
    I = np.copy(C) # On fait une copie de l'original
    nb_colonnes = I.shape[0]
    nb_lignes = I.shape[1]
    for i in range(nb_colonnes):
        for j in range(nb_lignes):
```

```
            r, g, b = I[i, j]
            I[i, j] = (0, 0, b)
    return I

A2 = filtre_bleu(A)
plt.imshow(A2)
plt.show()
```

2.6 Sauvegarde d'une image.

Pour enregistrer des images, on utilise une méthode similaire au chargement. Matplotlib permet ainsi de sauvegarder directement un tableau numpy au format PNG uniquement. PIL permettra de sauvegarder dans n'importe quel format, pourvu qu'on ait transformé le tableau numpy en Image PIL.

2.6.1 Avec le module PIL.

Avec PIL, il faut s'abord transformer le tableau numpy en image PIL.

```
from PIL import Image
imgpil = Image.fromarray(img) #transformation du tableau numpy en image PIL
imgpil.save("resultat.jpg")
```

2.6.2 Avec le module matplotlib.image.

```
import matplotlib.image as mpimg
mpimg.imsave("resultat.png", A)
```

III. Affichage d'une image à partir d'un tableau numpy.

```
import matplotlib.pyplot as plt
import numpy as np
from math import cos, sin
from random import randint

#création d'un tableau de 128 colonnes * 256 lignes zéros
A = np.zeros((128, 256, 3), dtype=np.uint8)
nb_ligne, nb_colonne, nb_plans = A.shape #récupération des dimensions de l'image et du nombre de
plans
for ligne in range(nb_ligne) :
    for colonne in range(nb_colonne):
        A[ligne][colonne] = [randint(0,255), randint(0,255), randint(0,255)]

plt.imshow(A)
plt.title("Exemple de création d'une image") #affichage du titre
plt.axis ("off") #cache les axes
plt.show() #affichage de l'image
```

Stéphane KELLER – Lycée agricole Louis Pasteur
Voici quelques exemples de traitement d'image, basés sur une image de référence en traitement d'image.

```
import matplotlib.pyplot as plt
import numpy as np
import os

os.chdir('E:/Cours/3 - Informatique/BCPST/Cours/Traitement images')
#os.chdir('P:/Mes documents')
#affichage du répertoire courant
print('Le répertoire courant est : ', os.getcwd())

#image initiale
image1 = plt.imread('Image caillou PGM.pgm') #ouverture de l'image
nblig , nbcol = image1.shape #récupération des dimensions de l'image
print('Nombre de lignes : ', nblig)
print('Nombre de colonnes : ', nbcol)

#image en négatif
image2 = np.copy(image1) #copie de l'image initiale
for lig in range(nblig) :
    for col in range(nbcol) :
        image2[lig][col] = 255 - image1[lig][col]

#seuil pixels de l'image en gris
#image en noir : 0 , image en blanc : 255
image3 = np.copy(image1) #copie de l'image initiale
for lig in range(nblig) :
    for col in range(nbcol) :
        if image3[lig][col] >= 150 :
            image3[lig][col] = 0
        else :
            image3[lig][col] = 255

#atténuation de l'image
image4 = np.copy(image1) #copie de l'image initiale
for lig in range(nblig) :
    for col in range(nbcol) :
        image4[lig][col] = int(image4[lig][col] / 50)

#portion de l'image (zoom)
xmin, xmax, ymin, ymax = 100, 200, 200, 300
image5 = np.zeros((xmax - xmin, ymax - ymin))
for lig in range(xmax - xmin) :
    for col in range(ymax -ymin) :
        image5[lig][col] = image1[xmin + lig][ymin + col]

#rotation +90° de l'image
image6 = np.zeros((nbcol, nblig))
for lig in range(nbcol) :
    for col in range(nblig) :
        image6[lig][col] = image1[-319 + col][ 479 - lig]

#ou plus simplement
```

```
#image6 = np.rot90(image1, 1)

#symétrie horizontale
image7 = np.copy(image1) #copie de l'image initiale
for lig in range(int(nblig / 2)) :
    for col in range(nbcol) :
        image7[lig][col], image7[nblig-lig-1][col] = image7[nblig - lig - 1][col], image7[lig][col]

#symétrie verticale
image8 = np.copy(image1) #copie de l'image initiale
for lig in range(nblig) :
    for col in range(int(nbcol / 2)) :
        image8[lig][col], image8[lig][nbcol - col - 1] = image8[l][nbcol - col - 1], image8[lig][col]

#affichage de l'image initiale en niveaux de gris
plt.subplot(3, 3, 1)
plt.axis('off') #suppression des axes
plt.title('Image initiale') #affichage du titre
plt.imshow(image1, cmap = 'gray', interpolation = 'nearest')

#affichage en négatif
plt.subplot(3, 3, 2)
plt.axis('off') #suppression des axes
plt.title('Image en négatif') #affichage du titre
plt.imshow(image2, cmap = 'gray', interpolation = 'nearest')

#affichage avec un seuil
plt.subplot(3, 3, 3)
plt.axis('off') #suppression des axes
plt.title('Image avec seuil') #affichage du titre
plt.imshow(image3, cmap = 'gray', interpolation = 'nearest')

#affichage avec atténuation
plt.subplot(3, 3, 4)
plt.axis('off') #suppression des axes
plt.title('Image avec atténuation') #affichage du titre
plt.imshow(image4, cmap = 'gray', interpolation = 'nearest')

#portion d'image
plt.subplot(3, 3, 5)
plt.axis('off') #suppression des axes
plt.title('Portion d\'image') #affichage du titre
plt.imshow(image5, cmap = 'gray', interpolation = 'nearest')

#rotation de l'image
plt.subplot(3, 3, 6)
plt.axis('off') #suppression des axes
plt.title('Rotation image') #affichage du titre
plt.imshow(image6, cmap = 'gray', interpolation = 'nearest')

#symétrie horizontale
plt.subplot(3, 3, 7)
plt.axis('off') #suppression des axes
```

```
plt.title('Symétrie horizontale') #affichage du titre
plt.imshow(image7, cmap = 'gray', interpolation = 'nearest')

#symétrie verticale
plt.subplot(3, 3, 8)
plt.axis('off') #suppression des axes
plt.title('Symétrie verticale') #affichage du titre
plt.imshow(image8, cmap = 'gray', interpolation = 'nearest')
```