Séance 12 :

Traitement des images

1. Exercice 1 : Chargement d'une image

1. ***Créer l'objet associé à l'image*** `image_couleurs.png` ***, regarder sa taille, son format, et son mode***
   Comme d'habitude, avant toute chose, je crée un dossier `ressources` dans lequel je place les ressources (ici les images) de la séance.
   Ensuite j'importe les librairies nécessaires :

```
import numpy as np
import matplotlib.pyplot as plt
import PIL.Image as im
```

   Puis je réponds à la question :

```
dirname = os.path.dirname(__file__)
filename = os.path.join(dirname, "ressources/originale.jpg")

img = im.open("ressources/original.jpg")

imSize = img.size
imMode = img.mode
imFormat = img.format

print("size:\t{}\nmode:\t{}\nformat:\t{}".format(imSize, imMode, imFormat))
```

   Et j'obtiens :

```
size:   (800, 864)
mode:   RGB
format: JPEG
```

2. ***A partir de cet objet faire afficher l'image***

```
# Open with the default image viewer of the PC
img.show()

# Open into a matplotlib graph
plt.imshow(img)
plt.show()
```

3. ***Récupérer les pixels de l'image sous forme d'un tableau***

```
pixelArray = np.array(img)
```

2. Exercice 2 : Conversion d'une image couleur en niveau de gris

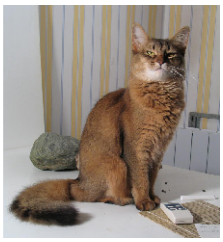
1. ***Convertir l'image en noveau de gris avec la fonction*** `convert` ***et faire afficher le résultat.***

   ❏ Dans le dossier `ressources`, je crée un fichier `helpers.py` dans lequel je crée la fonction `plotImage` qui me servira aussi pour les exerices suivants.
   Elle permet de rajouter plusieurs images sur le même `plot`. Elle ne lance le rendu de toutes les images que si le paramètre `show` est passé à `True`.

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

# On definit les variables globales permettant de gerer les subplot
numberOfPlots = 1
size = (1, 1)


# On cree une fonction qui affiche l'image dans une figure pyplot
def plotImage(image, s=None, show=False):
    global numberOfPlots
    global size
    if s:
        size = s
    plt.subplot(size[0], size[1], numberOfPlots)
    plt.axis('off')
    mode = 'L'
    if type(image) == np.ndarray:
        if len(image) == 3:
            mode = 'RGB'
    else:
        if image.mode == 'RGB':
            mode = 'RGB'
    if mode == 'L':
        plt.imshow(image, cmap='gray')
    else:
        plt.imshow(image)

    numberOfPlots += 1
    if show:
        plt.show()
```

❏ Ensuite, je convertis l'image et j'affiche le avant-après.

```python
# On cree l'image en noir et blanc
greyImg = img.convert('L')

# On affiche les images
plotImage(greyImg, (1, 2))
plotImage(img, show=True)
```

2. ***Ecrire une fonction qui retroune un tableau de niveau de gris d'une image.***

```python
def geryScale(image):
    # On definit les valeurs de luminance
    R, G, B, = 0.299, 0.587, 0.114

    # On recupere les couleurs de l'image separemment
    r, g, b, = image.split()

    # On les ajoute dans la nouvelle image
    return r * np.array(R) + g * np.array(G) + b * np.array(B)
```

3. ***Créer une image depuis le tableau précédent.***

```python
def arrayToImage(array):
    s = array.shape
    if len(s) == 3:
        mode = 'RGB'
        array = array.reshape(s[0] * s[1], 3)
        array = [(r, g, b) for r, g, b in array]
    else:
        mode = 'L'
        array = array.flat
    newIm = im.new(mode, (s[1], s[0]))
    data = list(array)
    newIm.putdata(data)
    return newIm
```

Il suffit maintenant d'éxecuter `greyScaleImg = arrayToImage(geryScale(img))` pour avoir le tableau demandé.

4. ***Enregistrer cette image.***

```python
greyScaleImg.save('ressources/image_grise.jpg')
```

3. Exercice 3 : Transformation d'une image

1. ***Réaliser et faire afficher une symétrie par rapport à l'axe horizontal.***
   Je définis la symétrie verticale :

```python
def hSymetry(greyScaleArray):
    reversedArray = np.empty_like(greyScaleArray)
    l = len(reversedArray)-1
    for i in range(l, -1, -1):
        reversedArray[l-i] = greyScaleArray[i]
    return reversedArray
```

   Et j'affiche le tout :

```python
plotImage(geryScale(img), (2, 1))
plotImage(vSymetry(geryScale(img)), show=True)
```

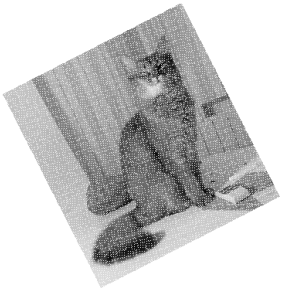
2. ***Réaliser et faire afficher une symétrie par rapport à l'axe vertical.***
   Pareil :

```python
def vSymetry(greyScaleArray):
    return (np.transpose((hSymetry(np.transpose(greyScaleArray)))))
```

```python
plotImage(geryScale(img), (1, 2))
plotImage(vSymetry(geryScale(img)), show=True)
```

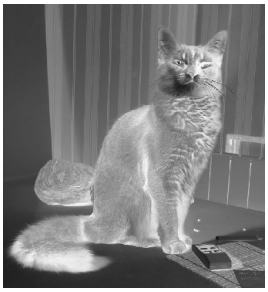
3. ***Réaliser et faire afficher une rotation de 20%.***
Je définis : `def cos(x): return np.cos(x)` et `def sin(x): return np.sin(x)` pour un peu plus de clareté. Puis je définis `rotation` :

```
def rotation(greyScaleArray, percentage):
    height, width = greyScaleArray.shape
    alpha = (percentage / 100) * (np.pi / 2)
    c, s = cos(alpha), sin(alpha)
    w = int(width * c + height * s)
    h = int(height * c + width * s)
    newArray = np.full((h, w), 255)
    rMatrix = np.array(np.array(((c, -s), (s, c))))
    for y in range(height):
        for x in range(width):
            try:
                nY, nX = np.dot(rMatrix, np.array((y, x)))
                nY = nY + width * s
                newArray[int(nY), int(nX)] = greyScaleArray[y][x]
            except IndexError:
                pass
    return (newArray)
```



4. ***Faire afficher l'image en négatif.***

```
def negative(greyScaleArray):
    return -greyScaleArray
```



5. ***Faire afficher l'image en négatif.***

```
def seuil(greyScaleArray, seuil):
    return np.where(greyScaleArray < seuil, 0, 255)
```