VI/ Interpréter un tableau comme une image

1. Structure

Pour Python, une image est représentée par un tableau numpy dont chaque coefficient est un **pixel**.

Chaque pixel est une liste de la forme **[R,G,B]** où R représente la quantité de rouge, G la quantité de vert et B la quantité de bleu. R, G et B prennent des valeurs entre 0 (absence totale de la couleur) et 255 (présence totale de la couleur).

```
pn = [0  , 0  , 0]          #pixel noir
pb = [255  , 255  , 255]    #pixel blanc
pg = [125  , 125  , 125]    #pixel gris
```

Par ailleurs, pour être interprétés correctement par Python, R, G et B doivent être de type **np.uint8** ; cela sera spécifié au moment de la création des images.

2. Créer une image

- Exemple 1

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

pn = [0,0,0]
pb = [255,255,255]
pg = [125,125,125]

def exemple1():
    ligne1 = [pn,pn,pn,pn,pn]
    ligne2 = [pn,pg,pg,pg,pn]
    ligne3 = [pn,pg,pb,pg,pn]
    image = np.array([ligne1,ligne2,ligne3,ligne2,ligne1], dtype = np.uint8)
    plt.imshow(image)
    plt.show()
```

Affichage à l'exécution de la fonction :

A retenir :

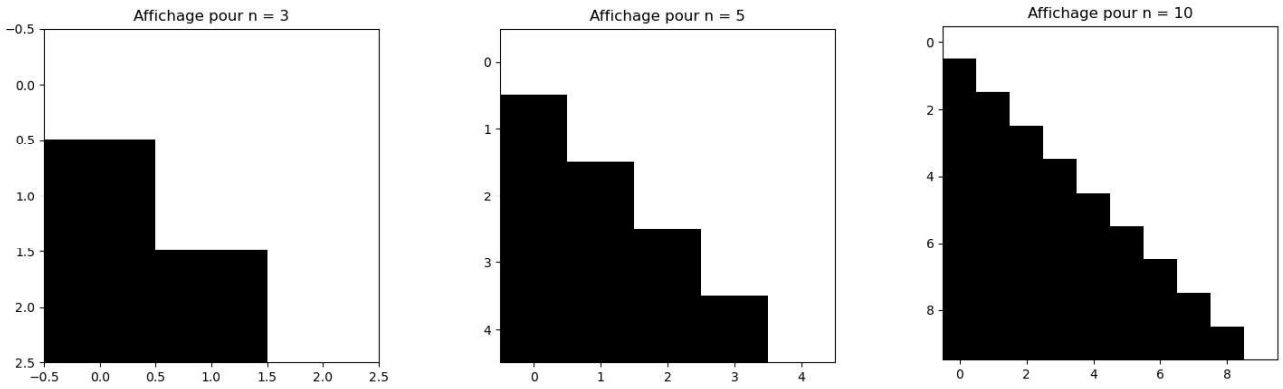
→ Place du **dtype = np.uint8** au moment de la création de l'image

→ Commande **plt.imshow** pour l'affichage d'une image

- Exemple 2

  Dans la fonction ci-dessous, **n** est un entier naturel non nul. Compléter les pointillées de sorte que la fonction permette l'affichage des images proposées.

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

def exemple2(n):

    resultat = np.zeros((n,n,3),dtype=np.uint8)

    for i in range(n):

        for j in range(n):

            if ...

                ...

            else:

                ...

    plt.imshow(resultat)

    plt.title("Affichage pour n = "+str(n))

    plt.show()
```



A retenir :

→ Utilisation de la commande **np.zeros** pour créer une image de la bonne taille

→ Pour modifier une image, il *faut* la parcourir sur les indices

- Exemple 3

  Dans la fonction ci-dessous, **p1** et **p2** sont deux listes dont on suppose qu'elles sont de même longueur. Compléter les pointillées de sorte que la fonction renvoie **True** si les listes sont identiques et **False** sinon.

```
def comparaison(p1,p2):

    n = len(p1)

    for i in range(n):

        if ...

            return ...

    return ...
```

  Dans la fonction ci-dessous, **image** est un tableau numpy représentant une image. Compléter les pointillés de sorte que la fonction remplace tous les pixels noirs de **image** par des pixels bleus puis affiche le résultat obtenu.

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

def exemple3(image):

    n = ...

    p = ...

    for i in range(n):

        for j in range(p):

            if ...

                ...

    plt.imshow(image)

    plt.show()
```

A retenir :

→ On ne peut pas comparer directement des listes ; il *faut* utiliser une fonction annexe pour le faire

3. Manipuler une image préexistante

On peut importer une image enregistrée, la modifier, puis enregistrer la version modifiée de cette image grâce aux commande suivantes :

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

Image_enregistree = plt.imread("Chemin d'acces a l'image/Nom de l'image.jpg")

def fonction(im):
    resultat = np.copy(im)
    ...
    ...
    ...
    plt.imsave("Chemin d'acces a l'image/Nom de la nouvelle image.jpg" , resultat)
```