TP 11 : Traitement d'images avec Numpy

1. Décomposition d'une image en ses composantes

Question 1. Pour obtenir la composante rouge, il faut mettre les composantes vertes et bleues de chaque pixel à 0 (elles sont indexées par 1 et 2) :

```
def composante_rouge(im):
    t=np.array(im)
    h,l,r=t.shape
    for i in range(h):
        for j in range(l):
            for k in [1,2]:
                t[i][j][k]=0
    return Image.fromarray(t)
```

Pour obtenir les composantes vertes et bleues, il suffit de changer la liste `[1, 2]` en `[0, 2]` et `[0, 1]`.

2. Négatif d'une image

Question 2. Première version :

```
def negatif(im):
    t=np.array(im)
    h,l,r=t.shape
    for i in range(h):
        for j in range(l):
            for k in range(2):
                t[i][j][k]=255-t[i][j][k]
    return Image.fromarray(t)
```

Deuxième version : on peut éviter d'écrire les trois boucles en utilisant les propriétés des tableaux Numpy :

```
def negatif(im):
    t=np.array(im)
    h,l,r=t.shape
    t=255-t
    return Image.fromarray(t)
```

3. Mise en place d'un cadre autour d'une image

Question 3. Par exemple :

```
def cadre_noir(im,ep):
    t=np.array(im)
    h,l,r=t.shape
    for i in list(range(ep))+list(range(h-ep,h)):
        for j in range(l):
            for k in range(3):
                t[i][j][k]=0
    for i in range(h):
        for j in list(range(ep))+list(range(l-ep,l)):
            for k in range(3):
                t[i][j][k]=0
    return Image.fromarray(t)
```

4. Rajout d'un cadre autour de l'image

Question 4. Ici, il faut créer un nouveau tableau plus gros, dans lequel on va recopier le tableau initial. Comme les zéros sont associés à la couleur noire, il n'y a rien à faire pour le cadre : il est déja là !

```
def rajout_cadre(im,ep):
    t=np.array(im)
    h,l,r=t.shape
    t2=np.zeros((h+2*ep,l+2*ep,r),dtype="uint8")
    for i in range(h):
        for j in range(l):
            for k in range(3):
                t2[i+ep][j+ep][k]=t[i][j][k]
    return Image.fromarray(t2)
```

5. Conversion d'une image couleur en image en niveau de gris

Question 5. Attention à calculer une seule fois la moyenne pondérée des composantes :

```
def niveau_gris(im):
    t=np.array(im)
    h,l,r=t.shape
    for i in range(h):
        for j in range(l):
            r,v,b=t[i][j]
            a=0.299*r+0.587*v+0.114*b
            for k in range(3):
                t[i][j][k]=a
    return Image.fromarray(t)
```

6. Image au format « B »

Question 6. Ici on crée un tableau ayant seulement deux dimensions :

```
def niveau_gris2(im):
    t=np.array(im)
    h,l,r=t.shape
    t2=np.zeros((h,l),dtype="uint8")
    for i in range(h):
        for j in range(l):
            r,v,b=t[i][j]
            a=0.299*r+0.587*v+0.114*b
            t2[i][j]=a
    return Image.fromarray(t2)
```

L'image renvoyée par cette fonction pèse moins lourd que l'image en noire et blanc au format RGB : 150 ko contre 255 ko. Remarque : PNG est un format *compressé sans perte*, l'image initiale `lena.png` pèse 466 ko, il y a donc déja économie en passant en noir et blanc au format RGB.

7. Histogramme d'une image

Question 7.

```
def histogramme(im):
    t=np.array(im)
    T=[0]*256
    h,l=t.shape
    for i in range(h):
        for j in range(l):
            T[t[i][j]]+=1
    return T
```

La fonction suivante trace et affiche l'histogramme d'une image au format « B » :

```
def trace_histo(im):
    T=histogramme(im)
    plt.plot(list(range(256)),T)
    plt.show()
```

8. Modifier la luminosité d'une image

Question 8.

```
def change_luminosite(im,d):
    t=np.array(im)
    h,l=t.shape
    for i in range(h):
        for j in range(l):
            if t[i][j]+d>255:
                t[i][j]=255
            elif t[i][j]+d<0:
                t[i][j]=0
            else:
                t[i][j]+=d
    return Image.fromarray(t)
```

9. Augmentation du contraste par dilatation de l'histogramme

Question 9. Par exemple :

```
def augmente_contraste(im,s):
    t=np.array(im)
    h,l=t.shape
    histo=histogramme(im)
    imin=min([i for i in range(256) if histo[i]>=s])
    imax=max([i for i in range(256) if histo[i]>=s])
    def g(x):
        if x<0:
            return 0
        if x>255:
            return 255
        return int(round(x))
    def f(x):
        return g((x-imin)*256/(imax-imin))
    for i in range(h):
        for j in range(l):
            t[i][j]=f(t[i][j])
    return Image.fromarray(t)
```

10. Ensemble de Mandelbrot

```
def bornee(c):
    """ effectue 100 itérations au max, si module > 2 on s'arrete """
    z=0
    for i in range(100):
        z=z**2+c
        if abs(z)>2:
            return False
    return True

def mandelbrot(N):
    """ N: nombre de pixels """
    """ [0,N-1]*[0,N-1] -> [-2,2] * [-2,2] """
    T=np.zeros((N,N), dtype=np.uint8)+255 #tout blanc
    def c(i,j):
        """ le complexe associé au pixel (i,j) """
        return complex(4/(N-1)*j-2,-4/(N-1)*i+2)
    for i in range(N):
        for j in range(N):
            if bornee(c(i,j)):
                T[i,j]=0
    return Image.fromarray(T)
```