TD informatique du chapitre 5 : Traitement d'images 1

Pour ce TD, on utilisera l'image de Lena disponible à l'emplacement suivant :

P :\par_professeur_B_\BERGEZ \lena.png

On commencera donc par la commande :

```
lena=Image.open("P :\par_professeur\_B_\BERGEZ \lena.png")
```

Et on vérifiera que la commande `lena.show()` affiche bien l'image de Lena.

Exercice 1 : Symétrie et rotation

1. Ecrire une fonction `SymetrieVerticale` prenant en argument une image `im` et renvoyant l'image obtenue par symétrie verticale de `im`.
   Afficher `SymetrieVerticale(lena)`.
2. Ecrire une fonction `SymetrieHorizontale` prenant en argument une image `im` et renvoyant l'image obtenue par symétrie horizontale de `im`.
   Afficher `SymetrieHorizontale(lena)`.

Exercice 2 : Négatif

Une image négative est une image dont les couleurs ont été inversées par rapport à l'original. La couleur négative d'une couleur (R,V,B) est $(255-R,255-V,255-B)$ ainsi le blanc devient noir, le noir devient blanc, ...
Ecrire une fonction `negatif` prenant en argument une image `im` et renvoyant l'image négative de l'image.
Afficher `negatif(lena)`.

Exercice 3 : Réduction/agrandissement

On considère une image de taille $h \times l$ et un facteur $f > 0$. On souhaite obtenir une image de taille $\lfloor h.f \rfloor \times \lfloor l.f \rfloor$, ainsi, si $f > 1$, on va agrandir l'image et si $f < 1$, on va rétrécir l'image.
Ecrire une fonction `ReductionAgrandissement` prenant en argument une image `im` et un facteur `f` et renvoyant l'image redimensionnée.

Exercice 4 : Cadre

1. Ecrire une fonction `cadre_noir` prenant en argument une image `im` et un entier `ep` et renvoyant une image dont les `ep` pixels situés sur chaque bord ont été remplacés par un cadre noir.
   Afficher `cadre_noir(lena,20)`.
2. Ecrire une fonction `rajout_cadre` prenant en argument une image `im` et un entier `ep` et renvoyant une image à laquelle on a rajouté `ep` pixels noirs sur chaque bord.
   Afficher `rajout_cadre(lena,20)`.

Exercice 5 : Luminosité

En augmentant la valeur des composantes de chaque couleur, on augmente la luminosité de l'image. On va donc modifier chaque composante en ajoutant une constante `lum` de la façon suivante : si la valeur de la composante est `t`, elle devient :

$$\begin{array}{c} \texttt{t+lum si } 0 \leq \texttt{ t+lum} \leq 255, \\ 255 \text{ si } \texttt{t+lum} > 255, \\ 0 \text{ si } \texttt{t+lum} < 0. \end{array}$$

Ainsi, si `lum`> 0 l'image obtenue sera plus claire et si `lum`< 0 l'image obtenue sera plus foncée.
Ecrire une fonction `Luminosite` prenant en argument une image `im` et une constante `lum` et renvoyant une image dont la luminosité est modifiée.
Afficher `Luminosite(lena,-50)` et `Luminosite(lena,50)`.

Exercice 6 : Effet popart

1. On souhaite, comme dans l'exemple du cours, appliquer un filtre de différentes couleurs à une image. On utilisera ici des filtres (R,V,B) avec R,V,B valant 0 ou 255. Le principe du filtre est de remplacer dans l'image toute les composantes correspondant à une valeur nulle du filtre par 0.

   Ecrire une fonction `filtre` prenant en argument un tableau `t` représentant une image et une liste `p` de 3 entiers valant 0 ou 255 et renvoyant un tableau qui correspond à l'image obtenue par le filtre `p`.
2. On souhaite créer une image contenant 4 images filtrés avec les filtres : $[255,255,0]$ (jaune), $[255,0,0]$ (rouge), $[255,0,255]$ (magenta) et $[0,255,0]$ (vert). Les 4 images doivent être placés sous la forme 2×2.

   On pourra utiliser la fonction `concatenate` de `numpy` afin de concatener des tableaux :

- concatenate((t1,. . .,tn),axis=0) permet d'avoir une concatenation des tableaux t1,. . .,tn par colonnes (les tableaux sont les uns en dessous des autres),
- concatenate((t1,. . .,tn),axis=1) permet d'avoir une concatenation des tableaux t1,. . .,tn par lignes (les tableaux sont les uns à côté des autres).

Ecrire une fonction popart prenant en argument une image im et renvoyant l'image popart décrite ci-dessus.
Afficher popart(lena).