1 La bibliothèque Pillow

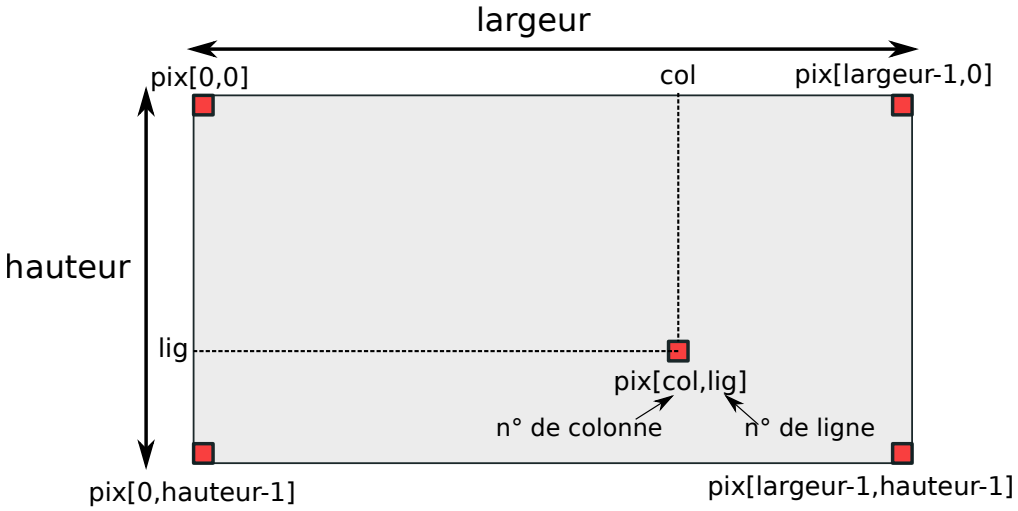
`Pillow` est une bibliothèque de traitement d'images, qui succède au projet PIL (*Python Imaging Library*). Elle est conçue de manière à offrir un accès rapide aux données contenues dans une image, et offre un support pour différents formats de fichiers tels que PPM, PNG, JPEG, GIF, TIFF et BMP.

À partir d'une image de base, nous allons construire des fonctions Python qui modifieront l'aspect de cette image : filtre de couleur, niveaux de gris, couleurs inversées,...
Le script Python de base contiendra les lignes suivantes :

Code Python :

```
# Importation de PIL
from PIL import Image
# Définition d'une fonction de modification
def modification(img) :
    largeur, hauteur = img.size # création des variables largeur et hauteur qui reçoivent les dimensions de l'image
    img2 = Image.new(img.mode, img.size) # création d'une image "vide" de même format et même dimension que l'image en paramètre de la fonction
    pix = img.load() # chargement de l'image comme un tableau de pixels de dimensions largeur x hauteur
    pix2 = img2.load() # même chose pour l'image "vide"
    ...          # instruction spécifiques à la fonction
    return img2 # renvoi du tableau
# programme principal
img = Image.open("mon_image.jpg") # ouverture de l'image souhaitée
imgnew=modification(img) # appel de la fonction de modification
imgnew.show() # affichage de la version modifiée de l'image
imgnew.save("resultat.png") # sauvegarde de l'image modifiée \end{lstlisting}
```

Avec les notations du script, chaque pixel sera repéré par un numéro de colonne et un numéro de ligne :



Repérage d'un pixel dans le tableau
obtenu par la méthode `load()`

Dans la partie suivante, nous allons parcourir chaque colonne et chaque ligne du tableau `pix` pour modifier les pixels un par un. Ces deux listes de coordonnées seront parcourues à l'aide de deux boucles imbriquées `for ... in range(...)`

2 Programmation Python

2.1 Préparation des fichiers

1. Créez un répertoire `python_images` dans votre espace personnel du réseau.
2. Copiez les fichiers images `ara_macao.jpg` et `arles.png` présents dans le dossier partagé de la classe.
3. Lancez une distribution `Python` et ouvrez le fichier `traitement_images.py`.
4. Enregistrez ce fichier sous un autre nom dans votre dossier `python_images`.

Pour chaque traitement ci-après, vous enregistrerez un nouveau fichier sous la forme `nom_de_la_modification.py`.

2.2 Filtre de couleur

Le principe du filtre de couleur est de ne conserver qu'une seule des 3 composantes RGB d'un pixel. Compléter le script suivant pour qu'il filtre l'image `"ara_macao.jpg"` en rouge puis en vert ou en bleu selon votre choix :

Code Python :

```
from PIL import Image
def filtre_rouge(img):
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix = img.load()
    pix2 = img2.load()
    for col in range(largeur):
        for lig in range(hauteur):
            (r, v, b) = pix[col, lig] # on extrait les composantes RGB du pixel
            pix2[col, lig] = ..... # on ne garde que la composante rouge
    return img2
img = Image.open("ara_macao.jpg")
imgnew = filtre_rouge(img)
imgnew.show()
imgnew.save("ara_rouge.png")
```

`filtre_rouge.py`

Code Python :

```
from PIL import Image
def filtre_vert(img):
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix = img.load()
    pix2 = img2.load()
    for col in range(largeur):
        for lig in range(hauteur):
            (r, v, b) = pix[col, lig] # on extrait les composantes RGB du pixel
            pix2[col, lig] = ..... # on ne garde que la composante verte
    return img2
img = Image.open("ara_macao.jpg")
imgnew = filtre_vert(img)
imgnew.show()
imgnew.save("ara_vert.png")
```

`filtre_vert.py`

2.3 Inversion des couleurs

Le principe ici est de prendre le complément à 255 de chaque composante RGB : si la composante rouge d'un pixel est de 90, alors il faudra la mettre à $255 - 90 = 165$.

Code Python :

```
from PIL import Image
def inversion(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig]
            pix2[col, lig] = .....        # complément à 255 des composantes
    return img2
img = Image.open("ara_macao.jpg")
imgnew = inversion(img)
imgnew.show()
imgnew.save("ara_inversion.png")
```

2.4 Niveaux de gris

Dans une image numérique, le niveau de gris représente la luminosité d'un pixel, lorsque les valeurs de ses composantes de couleur sont identiques.
Pour convertir une image infographique couleur en niveau de gris il donc faut remplacer, pour chaque pixel les trois valeurs représentant les niveaux de rouge, de vert et de bleu, par une seule valeur représentant la luminosité.
Par exemple, on peut remplacer chacune des composantes par la moyenne de celles-ci : par exemple si un pixel a pour composantes (189,217,34), on calcule $\frac{189+217+34}{3} \approx 146{,}67$. Cette moyenne n'est pas un nombre entier donc il est plus simple d'effectuer la division entière par 3 (commande `//` dans Python) qui renverra le quotient de la division euclidienne de la somme par 3 : `(189+217+34)//3`.

Code Python :

```
from PIL import Image
def niveaux_gris(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig] #composantes RGB du pixel
            m = .....      # moyenne des composantes
            pix2[col, lig] = .....       # affectation de la moyenne m à toutes
    les composantes
    return img2
img = Image.open("ara_macao.jpg")
imgnew = niveaux_gris(img)
imgnew.show()
imgnew.save("ara_gris.png")
```

Dans sa norme 709, la CIE (Commission Internationale de l'Éclairage) propose de remplacer les composantes par la valeur suivante :

$$m = 0.2126 \times r + 0.7152 \times v + 0.0722 \times b$$

Reprenez le script précédent et modifiez la définition de `m` selon le standard. Y-a-t-il une différence ?

3 Mise en noir et blanc (binarisation)

Le principe de binarisation d'une image couleur consiste à classer chaque pixel selon une valeur de seuil :

- calculer la moyenne `m` de ses composantes RGB ;
- si cette moyenne est supérieure au seuil , alors on considère que c'est un pixel "clair" et on lui affecte la valeur `blanc (255,255,255)`
- si cette moyenne est inférieure ou égale au seuil, alors on considère que c'est un pixel "sombre" et lui affecte la valeur `noir (0,0,0)`

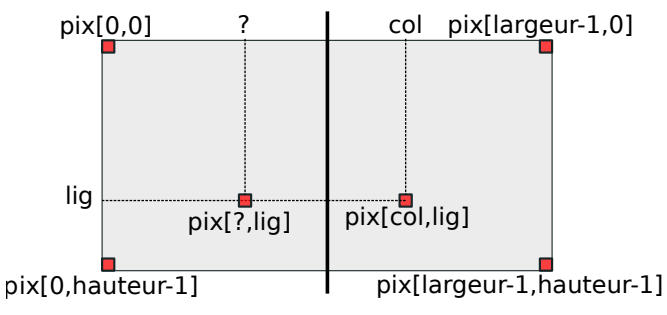
Compléter l'exemple ci-dessous en prenant comme valeur de seuil le nombre 127, valeur centrale de l'échelle des niveaux allant de 0 à 255.

Code Python :

```
from PIL import Image
def binarisation(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig] #composantes RGB du pixel
            m = ..... # moyenne des composantes
            if ..... :
                pix2[col, lig] = .....
            else :
                pix2[col, lig] = .....
    return img2
img = Image.open("ara_macao.jpg")
imgnew = binarisation(img)
imgnew.show()
imgnew.save("ara_binarisation.png")
```

4 Symétrie axiale d'axe vertical (miroir)

Pour obtenir une image symétrisée, il suffit d'affecter à un pixel `pix[col,lig]` du tableau `pix`, les composantes du pixel situé sur la même ligne mais dont la colonne est "inversée".



Code Python :

```
from PIL import Image
def symetrie(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig] # composantes RGB du pixel de coordonnées [col, lig]
            pix2[..... , lig] = (r, v, b) # affectation de ces composantes au pixel symétrique
    return img2
img = Image.open("ara_macao.jpg")
imgnew = symetrie(img)
imgnew.show()
imgnew.save("ara_symetrie.png")
```

Pour aller plus loin : flou d'une image par moyenne des pixels voisins (carré 5×5) détection de contours.

5 Scripts corrigés

5.1 Filtres rouges et verts

Code Python :

```
from PIL import Image
def filtre_rouge(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix = img.load()
    pix2 = img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig] #composantes RGB du pixel
            pix2[col, lig] = (r,0,0)
    return img2
img = Image.open("ara_macao.jpg")
imgnew = filtre_rouge(img)
imgnew.show()
imgnew.save("ara_rouge.png")
```

Code Python :

```
from PIL import Image
def filtre_vert (img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix = img.load()
    pix2 = img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig] # on extrait les composantes RGB du pixel
            pix2[col, lig] = (0, r, 0) # on ne garde que la composante verte
    return img2
img = Image.open("ara_macao.jpg")
imgnew = filtre_vert(img)
imgnew.show()
imgnew.save("ara_vert.png")
```

5.2 Inversion des couleurs

Code Python :

```
from PIL import Image
def inversion(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix = img.load()
    pix2 = img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig]
            pix2[col, lig] = (255 - r, 255 - v, 255 - b)
    return img2
img = Image.open("ara_macao.jpg")
imgnew = inversion(img)
imgnew.show()
imgnew.save("ara_inversion.png")
```

5.3 Niveaux de gris

Code Python :

```
from PIL import Image
def niveaux_gris(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig] #composantes RGB du pixel
            m = (r+v+b)//3 # moyenne des composantes
            pix2[col, lig] = (m, m, m) # affectation de la moyenne m à toutes les composantes
    return img2
img = Image.open("ara_macao.jpg")
imgnew = niveaux_gris(img)
imgnew.show()
imgnew.save("ara_gris.png")
```

5.4 Binarisation

Code Python :

```
from PIL import Image
def binarisation(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig] #composantes RGB du pixel
            m = (r + v + b)//3 # moyenne des composantes
            if m > 127 :
                pix2[col, lig] = (255, 255, 255)
            else :
                pix2[col, lig] = (0, 0, 0)
    return img2
img = Image.open("ara_macao.jpg")
imgnew = binarisation(img)
imgnew.show()
imgnew.save("ara_binarisation.png")
```

5.5 Miroir

Code Python :

```
from PIL import Image
def symetrie(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig] # composantes RGB du pixel de coordonnées [col, lig]
            pix2[largeur - col -1, lig] = (r, v, b) # affectation de ces composantes au pixel symétrique
    return img2
img = Image.open("ara_macao.jpg")
imgnew = symetrie(img)
imgnew.show()
imgnew.save("ara_symetrie.png")
```

5.6 Détection de contours

Code Python :

```python
from PIL import Image
from math import sqrt
def distance(pix1,pix2):
    """ distance euclidienne entre deux triplets"""
    return sum((pix1[i]-pix2[i])**2 for i in range(len(pix1)))

def contour(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    seuil = 30
    for lig in range(hauteur):
        for col in range(largeur):
            if col==0 or lig==0 or col==largeur-1 or lig==hauteur-1:
                pix[col,lig] = (255,255,255)
            else:
                dist=sqrt((distance(pix[col+1,lig],pix[col-1,lig])+distance(pix[col,lig-1],pix[col,lig+1]))//2)
                if dist< seuil:
                    pix2[col-1,lig-1]=(255,255,255)
                else:
                    pix2[col-1,lig-1]=(0,0,0)
    return img2
img = Image.open("ara_macao.jpg")
imgnew = contour(img)
imgnew.show()
imgnew.save("ara_contour.png")
```

5.7 Flou

Code Python :

```python
from PIL import Image
def moyenne_composantes(img, i, j, l):
    """ Renvoie la moyenne (entière) des composantes
        du carré de côté 2 * l + 1, centré en i, j
        avec une gestion correcte des bords
    """
    pix = img.load()
    c = [0] * 3
    n = 0
    for ii in range(max(0, i - l), min(i + l + 1, img.size[0])):
        for jj in range(max(0, j - l), min(j + l + 1, img.size[1])):
            n = n+1
            for k in range(3):
                c[k]= c[k]+pix[ii,jj][k]
    return c[0] // n, c[1] // n, c[2] // n

def flou(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            pix2[col,lig] = moyenne_composantes(img,col,lig,2)
    return img2
img = Image.open("arles.png")
imgnew = flou(img)
imgnew.show()
imgnew.save("arles_flou.png")
```

5.8 Sépia

Code Python :

```python
from PIL import Image
def sepia(img) :
    largeur, hauteur = img.size
    img2 = Image.new(img.mode, img.size)
    pix, pix2 = img.load(), img2.load()
    for col in range(largeur) :
        for lig in range(hauteur) :
            (r, v, b) = pix[col, lig] #composantes RGB du pixel
            m=(r+v+b)//3
            r,v,b= m,m,m # passage préalable en niveaux de gris
            if r <= 62 :
                r = int(1.1*r)
                b = int(0.9*b)
            elif r <= 191 :
                r = int(1.15*r)
                b = int(0.85*b)
            else :
                r = min(255, int(1.08*r))
                b = int(0.92*b)
            pix2[col, lig] = (r, m, b)
    return img2
img = Image.open("ara_macao.jpg")
imgnew = sepia(img)
imgnew.show()
imgnew.save("ara_sepia.png")
```