Traitement des données multimédia - TP1

FILALI Hicham

In [1]:
```
# Importing libraries
import cv2
import matplotlib.pyplot as plt
from collections import Counter
from google.colab.patches import cv2_imshow
```

1) Pour charger une image en coleur on utilise la fonction imread avec le parametre IMREAD_COLOR

In [2]:
```
img = cv2.imread("03.jpg", cv2.IMREAD_COLOR)
```

Pour l'affichage de l'image on utilise la fonction imshow

In [3]:
```
cv2_imshow(img)
```

2) Affichage de quelques propriétés de l'image:

In [4]:
```
dimensions = img.shape

height = img.shape[0]
width = img.shape[1]
channels = img.shape[2]
size = height * width
type = img.dtype

print("Image Properties:")
print(" Image width    :" , width , "\n" + " Image height   :" , height , "\n" , "Image size     :", size , "\n" , "Im
age channels :", channels , "\n" , "Image type     :", type)
```

```
Image Properties:
 Image width    : 200
 Image height   : 200
 Image size     : 40000
 Image channels : 3
 Image type     : uint8
```

3) histogramme de couleur de l’image en RGB

> a- methode manuelle:

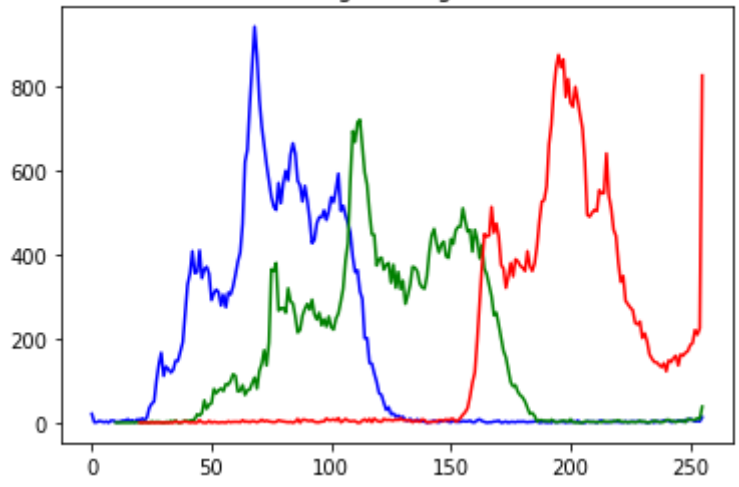
Tout d'abord, j'ai créé la fonction flatten qui transforme une liste de listes en une liste à 1 dimension, car l'image est représentée par une matrice (une liste de listes).

In [5]:
```
def flatten(l):
    return [item for sublist in l for item in sublist]
```

Ensuite, j'ai créé la fonction histogramme qui prend 2 paramètres, le premier est l'image pour laquelle nous voulons faire des histogrammes de couleur, le second est le titre du graphique.

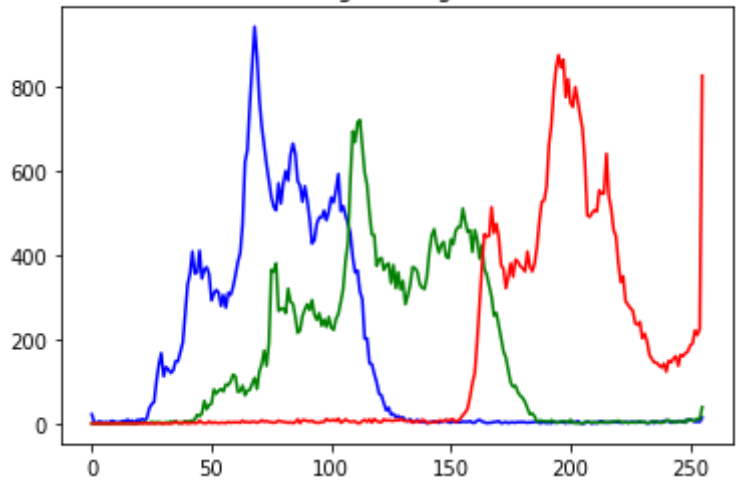
In [6]:
```
def Histogramme(img, title):
    blue = flatten(cv2.split(img)[0]) # extracting the blue component of the image
    green = flatten(cv2.split(img)[1]) # extracting the green component of the image
    red = flatten(cv2.split(img)[2]) # extracting the red component of the image

    blue_occ = Counter(blue)
    lists = sorted(blue_occ.items()) # sorted by key, return a list of tuples
    x, y = zip(*lists) # unpack a list of pairs into two tuples
    plt.plot(x, y, "-b")

    green_occ = Counter(green)
    lists = sorted(green_occ.items()) # sorted by key, return a list of tuples
    x, y = zip(*lists) # unpack a list of pairs into two tuples
    plt.plot(x, y, "-g")

    red_occ = Counter(red)
    lists = sorted(red_occ.items()) # sorted by key, return a list of tuples
    x, y = zip(*lists) # unpack a list of pairs into two tuples
    plt.plot(x, y, "-r")
    plt.title(title)
    plt.show()
```

In [7]:
```
Histogramme(img,"Image Histogram")
```



b- CalcHist:

In [8]:
```
colors = ('b','g','r')
# compute and plot the image histograms
for i,color in enumerate(colors):
    hist = cv2.calcHist([img],[i],None,[256],[0,256])
    plt.plot(hist,color = color)
plt.title('Image Histogram')
plt.show()
```



c- Pour comparer les distances de similarité entre les images par histogrammes avec la fonction compareHist, nous avons besoin de 2 images. Nous n'avons reçu qu'une seule image dans ce TP donc j'ai passé cette question.

4) Pour calculer la moyenne de chaque composante couleur j'ai créé la fonction suivante:

In [9]:
```
def CompMoy(img):
    blue = flatten(cv2.split(img)[0]) # extracting the blue component of the image
    green = flatten(cv2.split(img)[1]) # extracting the green component of the image
    red = flatten(cv2.split(img)[2]) # extracting the red component of the image

    moy_blue = sum(blue)/len(blue)
    moy_green = sum(green)/len(green)
    moy_red = sum(red)/len(red)

    return moy_blue,moy_green,moy_red
```

In [10]:
```
b,g,r=CompMoy(img)

print("La moyenne de la composante blue est: ",b)
print("La moyenne de la composante verte est: ",g)
print("La moyenne de la composante rouge est: ",r)
```

```
La moyenne de la composante blue est:  77.99985
La moyenne de la composante verte est:  122.996275
La moyenne de la composante rouge est:  200.0049
```

5) Conversion d'une image en image aux niveaux de gris

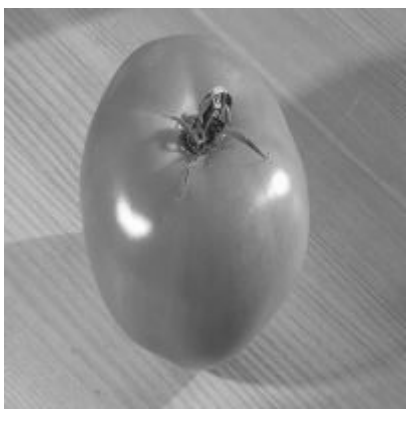
a- En utilisant la formule : gray=(R+G+B)/3

In [11]:
```
def ConvGray(img):
    for x in range(img.shape[1]):
        for y in range(img.shape[0]):
                img[x][y]= (int(img[x][y][0])+int(img[x][y][1])+int(img[x][y][2]))/3
```

In [12]:
```
ConvGray(img)
cv2_imshow(img)
```

b- En utilisant la fonction prédéfinie cvtColor

In [13]:
```
img = cv2.imread("03.jpg", cv2.IMREAD_COLOR)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2_imshow(gray)
```



c- Comparaison des résultats

On remarque que la deuxième méthode donne une image plus lumineuse que la méthode de la moyenne

6) Pour calculer la moyenne des intensités de l’image aux niveaux de gris j'ai créé la fonction suivante:

In [14]:
```
def GrayMoy(img):
    return sum(flatten(img))/len(flatten(img))
```

In [15]:
```
graymoy=GrayMoy(gray)
print("La moyenne des intensités de l’image aux niveaux de gris est : ",graymoy)
```

```
La moyenne des intensités de l’image aux niveaux de gris est :  140.89385
```

7) Sauvegarder l’image convertie dans un fichier

In [16]:
```
cv2.imwrite('gray.png', gray)
```

Out[16]:
```
True
```