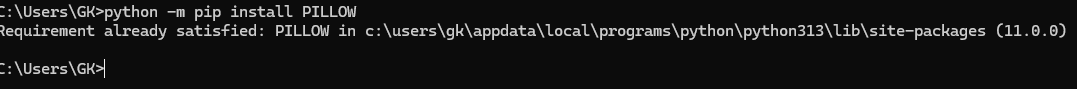
***Compte Rendu Du Travail Pratique 1***

**Dans cette partie, nous allons manipuler des images en utilisant les modules Image et ImageOps de la bibliothèque PIL, avec une explication et une implémentation en Python.**

**Install Pillow par cmd.exe :**

****

**Activité n°1 manipuler une image :**

Dans cette partie on va effectuer des opérations basiques sur image Lenna.png, afficher le format, taille, mode, les dimensions, nombres pixel dans image ensuite la résolution taille écran :

**Afficher image :**

**from PIL import Image, ImageOps**

**import matplotlib.pyplot as plt**

**img = Image.open("../images/lenna.png")**

**img.show()**

****

**Afficher format, size et mode sont des données (attributs) de la classe Image.**

**print(img.format, img.size, img.mode) print("Fichier :",img.filename)**

**Résultat :**

**PNG (220, 220) RGB**

**Fichier : ../images/lenna.png**

**Affichage des dimensions de l'image et nombre de pixels de l'image.**

**print("L=",img.width, "x H =",img.height)**

**print("Nombre de pixels =", (img.width\*img.height) , "pixels")**

**Résultat :**

**L= 220 x H = 220**

**Nombre de pixels = 48400 pixels**

**Modifier la taille d'une image par Resize() est une fonction (méthode) de la classe Image et sauvegarder nouvelle image im2.**

**size = (228,192)**

**im2 = img.resize(size)**

**path = "../images/lena\_228x192.png"**

**im2.save(path)**

**Afficher la nouvelle image et tous les attributs précédant da im2.**

**print(im2.format, im2.size, im2.mode)**

**print("Nombre de pixels =", "L =", im2.width, "x H =", im2.height, "=", (im2.width\*im2.height) , "pixels")**

**im2.show()**

**None (228, 192) RGB**

**Nombre de pixels = L = 228 x H = 192 = 43776 pixels**

****

**Activité n°1 : code python Complete.**

from PIL import Image, ImageOps

import matplotlib.pyplot as plt

img = Image.open("../images/lenna.png")

img.show()

# format, size et mode sont des données (attributs) de la classe Image

print(img.format, img.size, img.mode)

print("Fichier :",img.filename)

# Affichage des dimensions de l'image

print("L=",img.width, "x H =",img.height)

# Calcul et Affichage du nombre de pixels de l'image

print("Nombre de pixels =", (img.width\*img.height) , "pixels")

# resize() est une fonction (méthode) de la classe Image qui permet de modifier la taille d'une image et d'en obtenir une nouvelle

# im2 est un objet de la classe Image

size = (228,192)

im2 = img.resize(size)

# save() est une fonction (méthode) de la classe Image qui sauvegarder l'image dans un fichier (ici, l'extension précisera le format de l'image)

path = "../images/lena\_228x192.png"

im2.save(path)

# Afficher toutes les attributs precedant da im2

print(im2.format, im2.size, im2.mode)

print("Nombre de pixels =", "L =", im2.width, "x H =", im2.height, "=", (im2.width\*im2.height) , "pixels")

im2.show()

# size est un tuple (les tuples sont des séquences qu'on ne pourra plus modifier)

size = (144,96)

#= les parenthèses ne sont pas obligatoires

width, height = size

im3 = img.resize(size)

im3.save("../images/lena\_144x96.png")

print(im3.format, im3.size, im3.mode)

print("Nombre de pixels =", "L =", im3.width, "x H =", im3.height, "=", (im3.width\*im3.height) , "pixels")

im3.show()

**Activité n°2 : Histogrammes D’images**

**Importez les librairies Python, comme suit.**

**from PIL import Image**

**Ensuite** **d’ouvrir l’image avec la fonction open () du module Image.**

**img = Image.open(r"../images/Lenna.png")**

**img.show()**

****

**Convertir l’image en niveau de gris (avec la conversion couleur -> niveau de gris) par méthode Convert () de Classe Image**

**img\_niveaux\_gris = img.convert('L')**

**img\_niveaux\_gris.show()**

****

**Importez librairie matplotlib.pyplot pour visualiser Histogramme.**

**import matplotlib.pyplot as plt**

**Calculer Histogram D ’image (matrice de taille 256).**

**hist = img\_niveaux\_gris.histogram()**

**Ensuite on va afficher histogramme.**

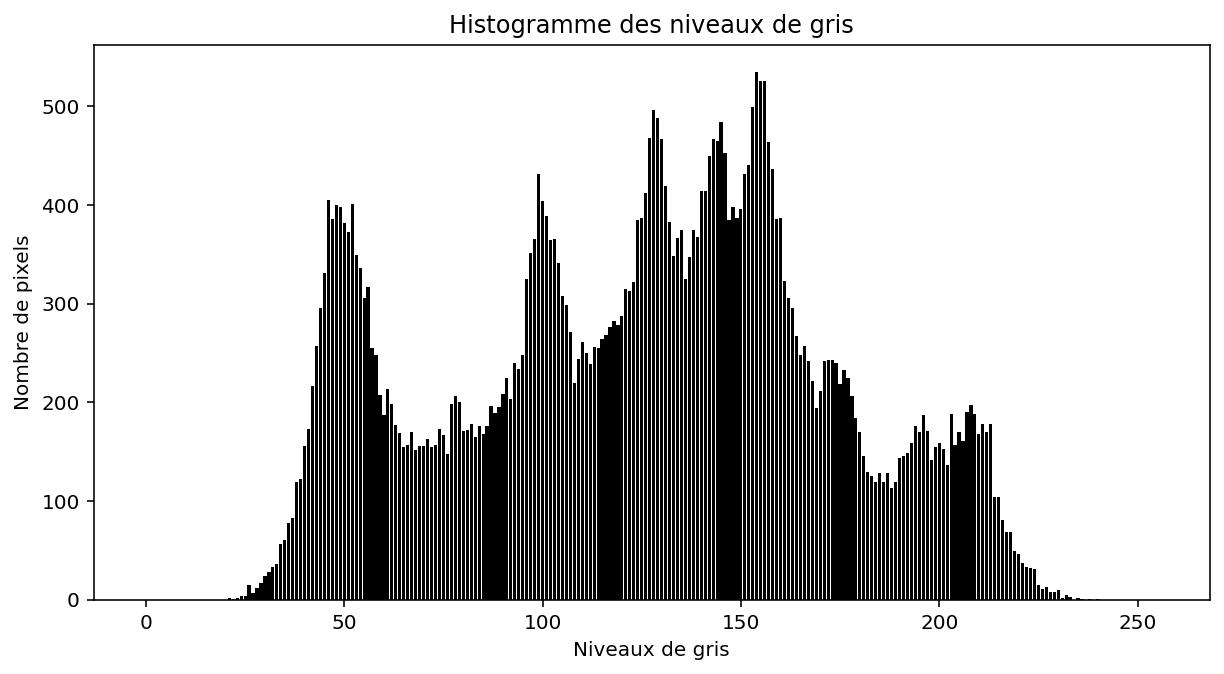
**plt.figure(figsize=(10, 5))**

**plt.bar(range(256), hist, color='black')**

**plt.xlabel("Niveaux de gris")**

**plt.ylabel("Nombre de pixels")**

**plt.title("Histogramme des niveaux de gris")**



**On peut aussi séparer les couches et visualiser image de chaque couche.**

**r, g, b = img.split()**

**r.show()**

**g.show()**

**b.show()**

**R G B**

**  **

**Ensuite on va Calculer Histogramme De chaque couche RGB (sortie 3 matrice de Taille 256).**

**hist\_r = r.histogram()**

**hist\_g = g.histogram()**

**hist\_b = b.histogram()**

**Ensuite on va afficher histogram de chaque couche par transparence alpha=0.3.**

**plt.figure(figsize=(10, 5))**

**plt.bar(range(256),hist\_r, color='red', alpha=0.3, label='R')**

**plt.bar(range(256),hist\_g, color='green', alpha=0.3, label='G')**

**plt.bar(range(256),hist\_b, color='blue', alpha=0.3, label='B')**

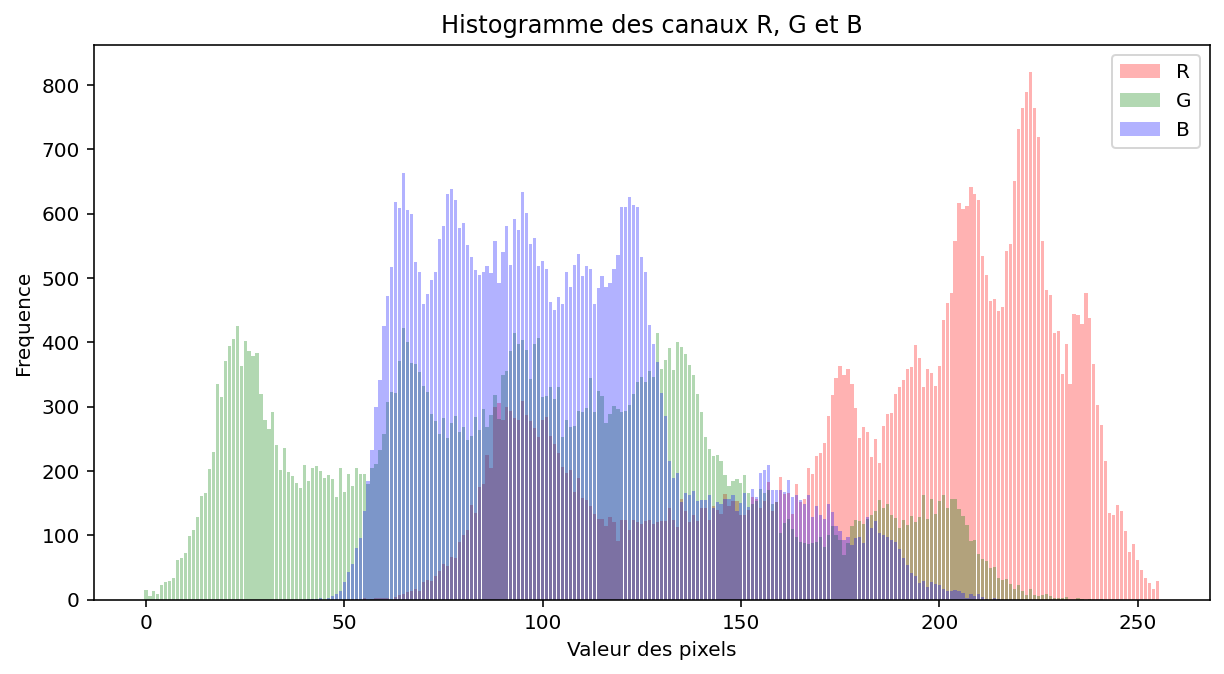
**plt.xlabel("Valeur des pixels")**

**plt.ylabel("Frequence")**

**plt.title("Histogramme des canaux R, G et B")**

**plt.legend()**

**plt.show()**



**Code python complète d’activité 2:**

from PIL import Image

import matplotlib.pyplot as plt

# lire une image

img = Image.open("../images/Lenna.png")

img.show()

# convertir image aux niveaux de gris

img\_niveaux\_gris = img.convert('L')

img\_niveaux\_gris.show()

# Calculer l'histogramme

hist = img\_niveaux\_gris.histogram()

# Afficher Histogramme de image en niveaux de gris

plt.figure(figsize=(10, 5))

plt.bar(range(256), hist, color='black')

plt.xlabel("Niveaux de gris")

plt.ylabel("Nombre de pixels")

plt.title("Histogramme des niveaux de gris")

# séparer les trois couches

r, g, b = img.split()

r.show()

g.show()

b.show()

# Calculer les histogrammes

hist\_r = r.histogram()

hist\_g = g.histogram()

hist\_b = b.histogram()

# Afficher les histogrammes sur le même graphique

plt.figure(figsize=(10, 5))

plt.bar(range(256),hist\_r, color='red', alpha=0.3, label='R')

plt.bar(range(256),hist\_g, color='green', alpha=0.3, label='G')

plt.bar(range(256),hist\_b, color='blue', alpha=0.3, label='B')

plt.xlabel("Valeur des pixels")

plt.ylabel("Frequence")

plt.title("Histogramme des canaux R, G et B")

plt.legend()

plt.show()

**Activité n°3 : Traitement D’images :**

Nous **allons utiliser le module Image et ImageOps de la bibliothèque PIL. ImageOps nous offre plusieurs méthodes permettant d’effectuer des opérations sur les images de manières simple.**

**from PIL import Image, ImageOps**

**Afficher et ouvrir image :**

**img = Image.open("../images/simba\_surexp.png")**

**img.show()**

****

**Convertir image en niveaux de Gris.**

**img\_niveaux\_gris = ImageOps.grayscale(img)**

**img\_niveaux\_gris.show()**

****

**Application autocontrast sur images en utilisant module ImageOps.**

**img1\_contraste = ImageOps.autocontrast(img\_niveaux\_gris);**

**img1\_contraste.show()**

****

**Afficher histogramme avant ajustement contraste.**

**hist = img\_niveaux\_gris.histogram()**

**plt.figure(figsize=(10,5))**

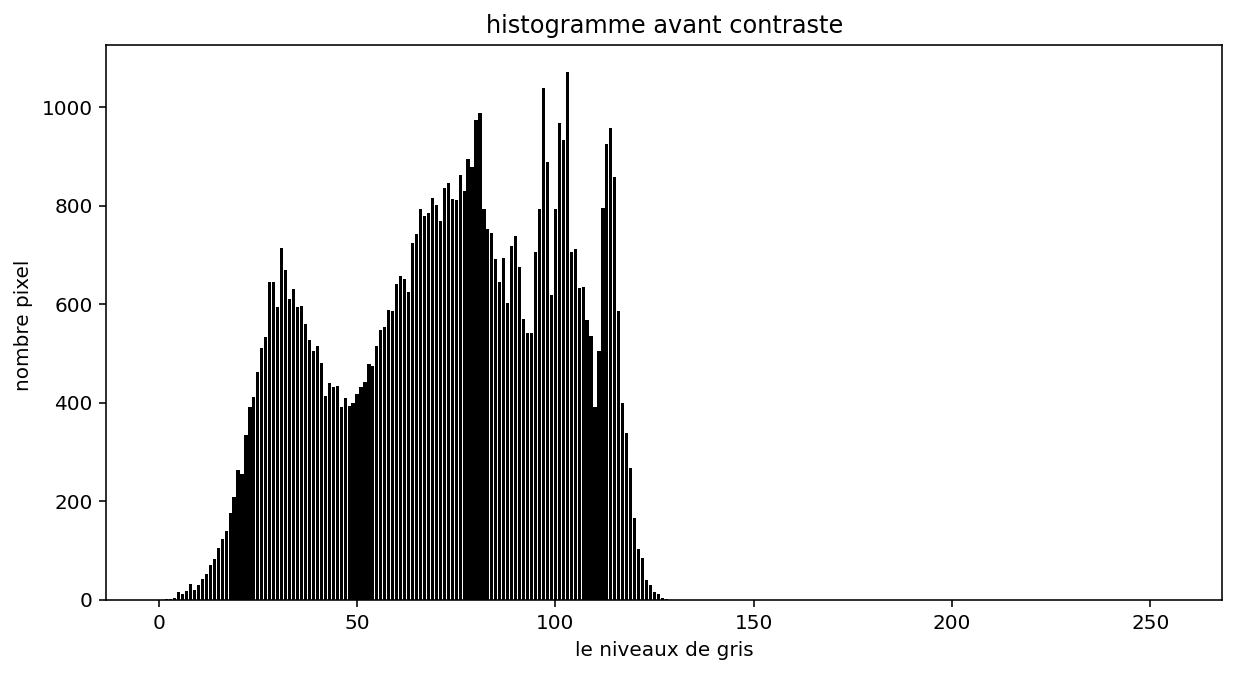
**plt.bar(range(256),hist,color='black')**

**plt.xlabel("le niveaux de gris")**

**plt.ylabel("nombre pixel ")**

**plt.title("histogramme avant contraste")**

**plt.show()**



**Afficher histogramme après ajustement contraste.**

**hist1 = img1\_contraste.histogram()**

**plt.figure(figsize=(10,5))**

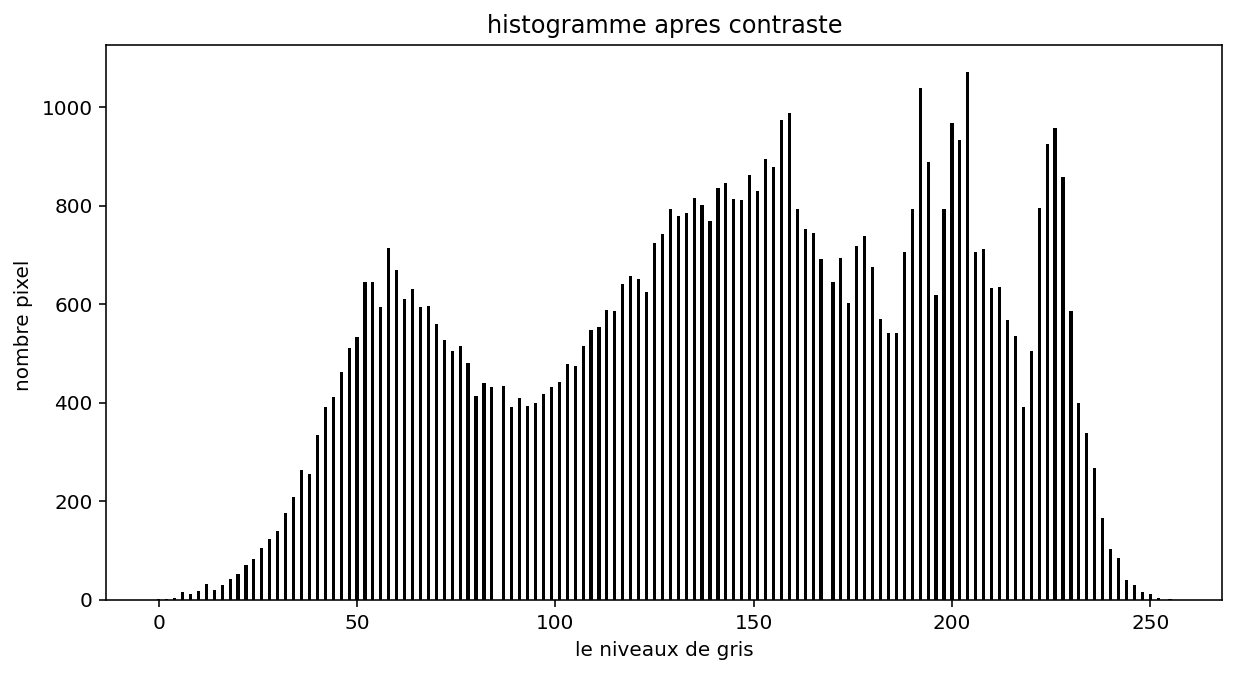
**plt.bar(range(256),hist1,color='black')**

**plt.xlabel("le niveaux de gris")**

**plt.ylabel("nombre pixel ")**

**plt.title("histogramme apres contraste")**

**plt.show()**



**Application égalisation de histogramme en utilisant la méthode equalize() de module ImageOps.**

**img\_egaliser = ImageOps.equalize(img\_niveaux\_gris)**

**hist\_eg = img\_egaliser.histogram()**

**plt.figure(figsize=(10,5))**

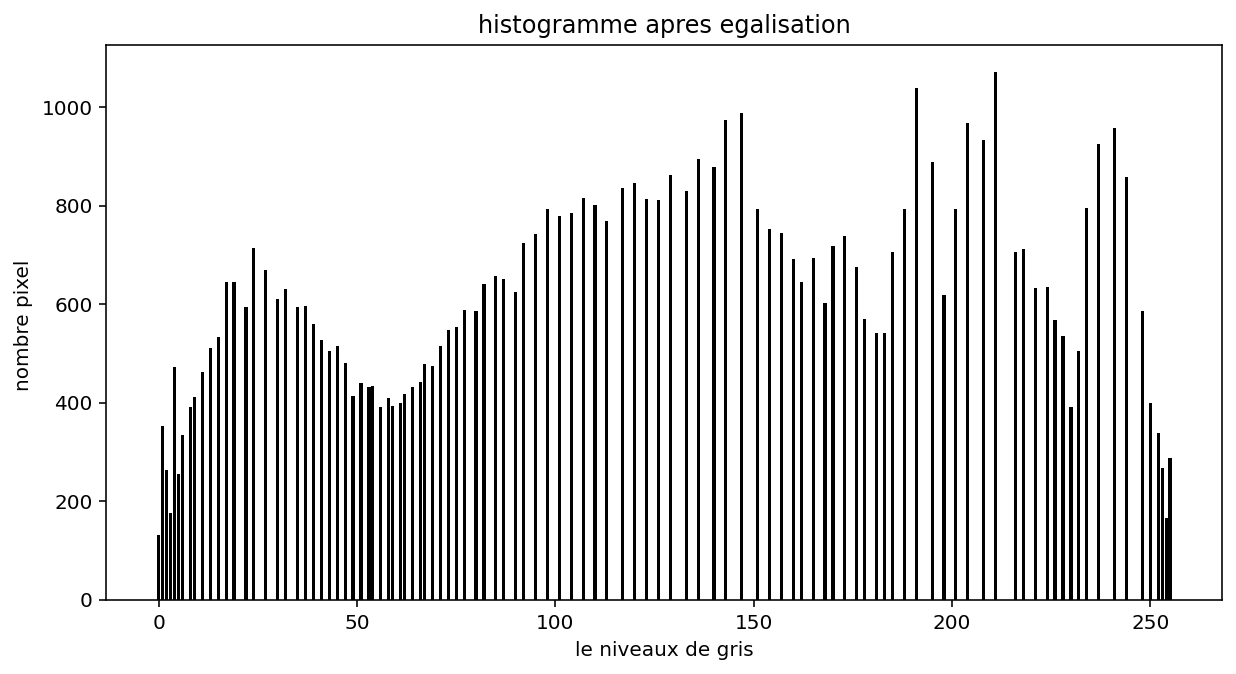
**plt.bar(range(256),hist\_eg,color='black')**

**plt.xlabel("le niveaux de gris")**

**plt.ylabel("nombre pixel ")**

**plt.title("histogramme apres egalisation")**

**plt.show()**



**Explorer les methodes invert de module ImageOps.**

**img\_invert = ImageOps.invert(img\_niveaux\_gris)**

**img\_invert.show()**

****

**Afficher Histogramme apres Exploration invert.**

**hist\_invert = img\_invert.histogram()**

**plt.figure(figsize=(20,5))**

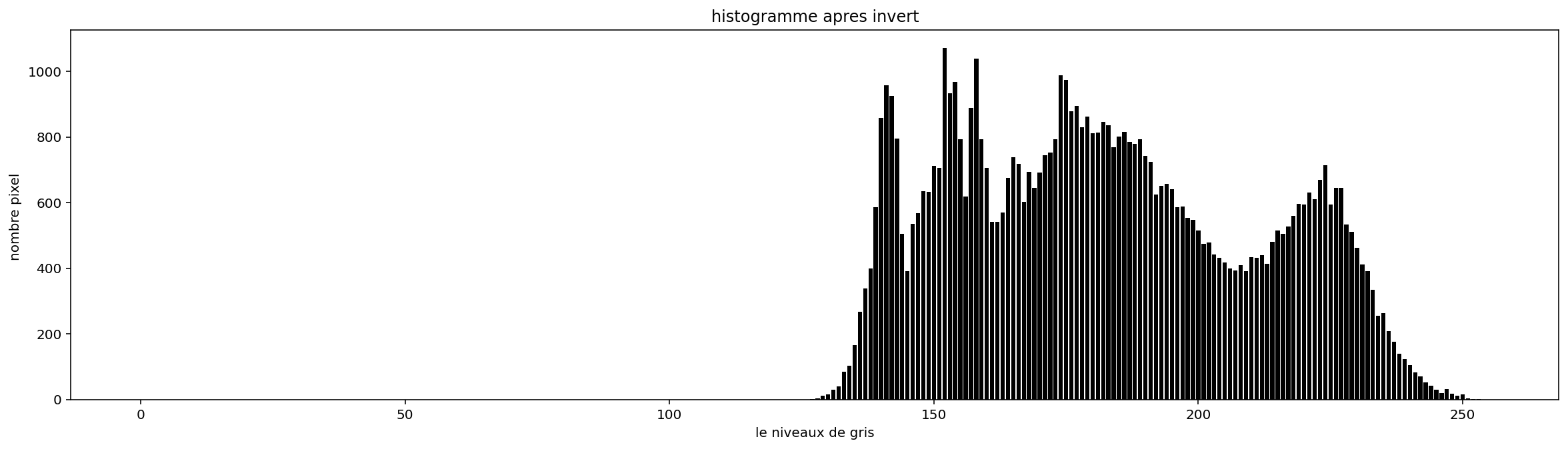
**plt.bar(range(256), hist\_invert, color='black')**

**plt.xlabel("le niveaux de gris")**

**plt.ylabel("nombre pixel ")**

**plt.title("histogramme apres invert")**

**plt.show()**



**Explorer les méthodes mirror de module ImageOps.**

**img\_mirror = ImageOps.mirror(img\_niveaux\_gris)**

**img\_mirror.show()**

****

**Afficher Histogram après Exploration mirror.**

**hist\_mi = img\_mirror.histogram()**

**plt.figure(figsize=(20,5))**

**plt.bar(range(256), hist\_mi, color='black')**

**plt.xlabel("le niveaux de gris")**

**plt.ylabel("nombre pixel ")**

**plt.title("histogramme apres mirror")**

**plt.show()**



**Code Python Complete de cette activité trois :**

from PIL import Image, ImageOps

import matplotlib.pyplot as plt

img = Image.open("../images/simba\_surexp.png")

img.show()

img\_niveaux\_gris = ImageOps.grayscale(img)

img\_niveaux\_gris.show()

# application de autocontrast sur les deux images

img1\_contraste = ImageOps.autocontrast(img\_niveaux\_gris);

img1\_contraste.show()

# afficher histogramme avant et apres operation contraste

hist = img\_niveaux\_gris.histogram()

plt.figure(figsize=(10,5))

plt.bar(range(256),hist,color='black')

plt.xlabel("le niveaux de gris")

plt.ylabel("nombre pixel ")

plt.title("histogramme avant contraste")

plt.show()

# afficher histogramme apres operation contraste

hist1 = img1\_contraste.histogram()

plt.figure(figsize=(10,5))

plt.bar(range(256),hist1,color='black')

plt.xlabel("le niveaux de gris")

plt.ylabel("nombre pixel ")

plt.title("histogramme apres contraste")

plt.show()

# egalisation de image

img\_egaliser = ImageOps.equalize(img\_niveaux\_gris)

hist\_eg = img\_egaliser.histogram()

plt.figure(figsize=(10,5))

plt.bar(range(256),hist\_eg,color='black')

plt.xlabel("le niveaux de gris")

plt.ylabel("nombre pixel ")

plt.title("histogramme apres egalisation")

plt.show()

# Explorer les methodes invert de module ImageOps

img\_invert = ImageOps.invert(img\_niveaux\_gris)

img\_invert.show()

# Afficher Histogramme apres Exploration invert.

hist\_mihist\_invert = img\_invert.histogram()

plt.figure(figsize=(20,5))

plt.bar(range(256), hist\_invert, color='black')

plt.xlabel("le niveaux de gris")

plt.ylabel("nombre pixel ")

plt.title("histogramme apres invert")

plt.show()

# Explorer les methodes mirror de module ImageOps

img\_mirror = ImageOps.mirror(img\_niveaux\_gris)

img\_mirror.show()

# Afficher Histogramme apres Exploration mirror.

hist\_mi = img\_mirror.histogram()

plt.figure(figsize=(20,5))

plt.bar(range(256), hist\_mi, color='black')

plt.xlabel("le niveaux de gris")

plt.ylabel("nombre pixel ")

plt.title("histogramme apres mirror")

plt.show()

**On peut aussi appliquer la transformation par autre méthode Flip ImageOps qui permet d’inverser image.**