Sup Galilée ingénieurs

L'ÉCOLE D'INGÉNIEURS DE L'UNIVERSITÉ SORBONNE PARIS NORD

Compte Rendu de TP 01

Traitement d'image

Nom: BOUARICHE

Prénom: Iheb

Année : 2023/2024

Spécialité : Instrumentation an2

Nom de l'étudiant: Iheb BOUARICHE /// Module: Traitement d'image /// Instrumentation an2

Partie A

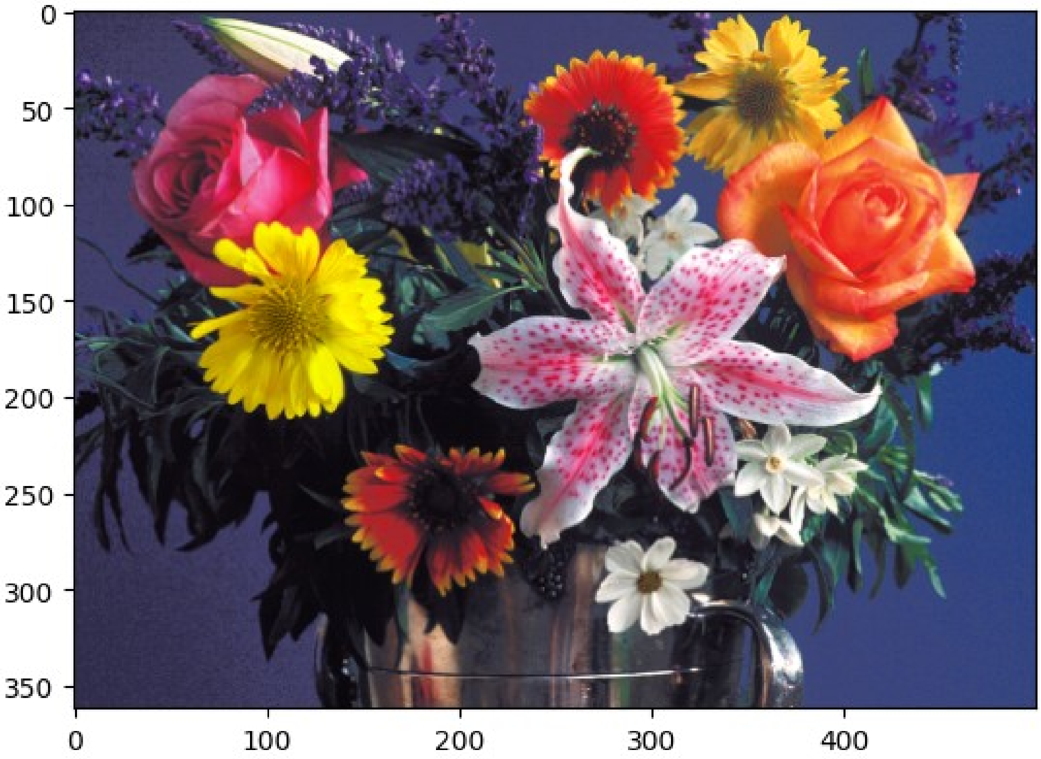
```
from skimage import io
import cv2
import matplotlib.pyplot as plt
```

Exercice 01

1)

```
image1 = io.imread("flowers.tif",False)
image1
```

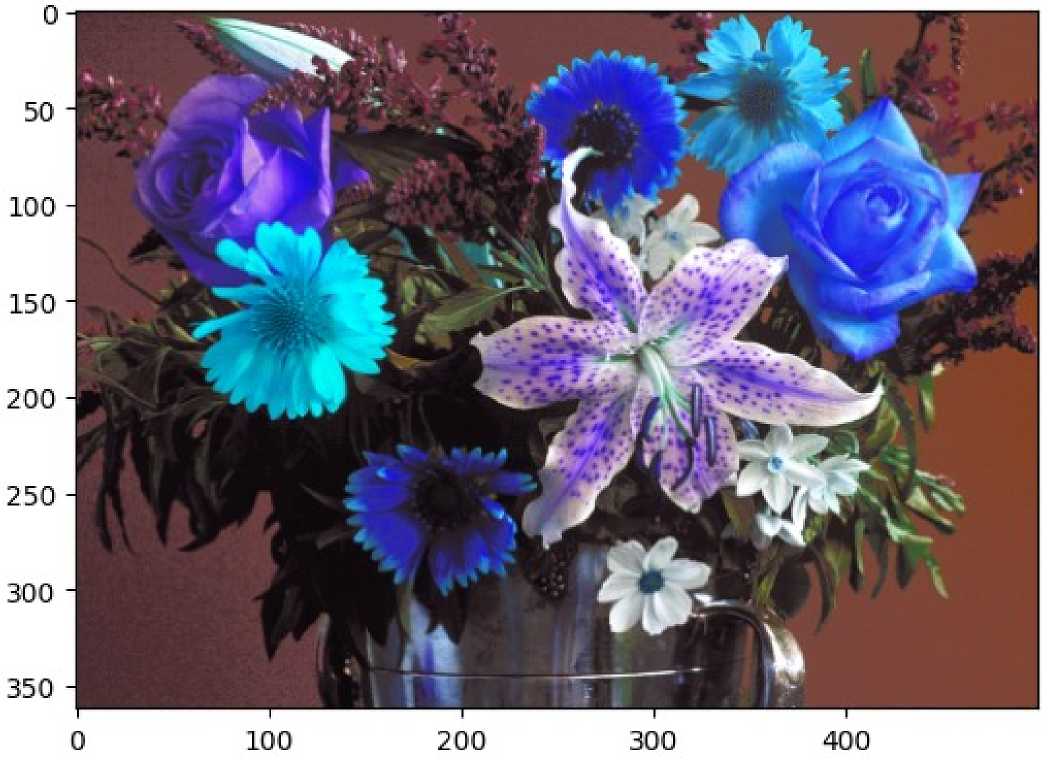
```
array([[[ 48,  45,  76],
        [ 49,  46,  76],
        [ 52,  49,  84],
        ...,
        [ 58,  72, 125],
        [ 53,  69, 129],
        [ 53,  70, 128]],

       [[ 48,  45,  77],
        [ 54,  50,  83],
        [ 49,  45,  78],
        ...,
        [ 54,  70, 129],
        [ 53,  68, 129],
        [ 47,  66, 133]],

       [[ 51,  48,  79],
        [ 51,  49,  80],
        [ 53,  50,  82],
        ...,
        [ 51,  69, 130],
        [ 48,  67, 131],
        [ 48,  67, 132]],

       ...,

       [[ 49,  45,  74],
        [ 50,  46,  78],
        [ 49,  45,  73],
        ...,
        [ 57,  67, 128],
        [ 54,  68, 129],
        [ 59,  70, 124]],

       [[ 48,  44,  74],
        [ 49,  46,  77],
        [ 48,  44,  74],
        ...,
        [ 53,  65, 131],
        [ 47,  64, 133],
        [ 56,  67, 126]],

       [[ 48,  44,  72],
        [ 49,  45,  79],
        [ 49,  45,  75],
        ...,
        [ 58,  68, 128],
        [ 55,  69, 127],
        [ 55,  70, 127]]], dtype=uint8)

image2 = io.imread("flowers.tif",True)
image2

array([[0.18773569, 0.19137451, 0.20455294, ..., 0.28567176,
0.27421961,
        0.27674235],
       [0.18801843, 0.20874235, 0.18913451, ..., 0.27785843,
0.27141412,
        0.26193412],
       [0.19950039, 0.20258863, 0.20762627, ..., 0.27283569,
0.26500745,
        0.2652902 ],
       ...,
       [0.18800353, 0.19277333, 0.18772078, ..., 0.27165922,
0.27224745,
        0.28061137],
       [0.18436471, 0.19165725, 0.18436471, ..., 0.26356314,
0.25632314,
        0.27026039],
       [0.18379922, 0.18941725, 0.18828627, ..., 0.27529804,
0.27532078,
        0.27812627]])

plt.imshow(image1)
plt.show()
```



```
from google.colab.patches import cv2_imshow
image3 = cv2.imread("flowers.tif", cv2.IMREAD_UNCHANGED)
print(image3)
cv2_imshow(image3)
```

```
[[[ 76  45  48]
  [ 76  46  49]
  [ 84  49  52]
  ...
  [125  72  58]
  [129  69  53]
  [128  70  53]]

 [[ 77  45  48]
  [ 83  50  54]
  [ 78  45  49]
  ...
  [129  70  54]
  [129  68  53]
  [133  66  47]]

 [[ 79  48  51]
  [ 80  49  51]
  [ 82  50  53]
  ...
  [130  69  51]
  [131  67  48]
  [132  67  48]]

 ...

 [[ 74  45  49]
  [ 78  46  50]
  [ 73  45  49]
  ...
  [128  67  57]
  [129  68  54]
  [124  70  59]]

 [[ 74  44  48]
  [ 77  46  49]
  [ 74  44  48]
  ...
  [131  65  53]
  [133  64  47]
  [126  67  56]]

 [[ 72  44  48]
  [ 79  45  49]
  [ 75  45  49]
  ...
  [128  68  58]
  [127  69  55]
  [127  70  55]]]
```

```
plt.imshow(image3)
plt.show()
```



```
image1.dtype
```

```
dtype('uint8')
```

```
image2.dtype
```

```
dtype('float64')
```

```
image1.shape
```

```
(362, 500, 3)
```

```
image2.shape
```

```
(362, 500)
```

2)

```
print("la valeur du coin droit haut =",image1[361,0,:])
print("la valeur du coin droit bas =",image1[361,499,:])
print("la valeur du coin gauche haut =",image1[0,0,:])
print("la valeur du coin gauche bas =",image1[0,499,:])
```

```
la valeur du coin droit haut = [48 44 72]
la valeur du coin droit bas = [ 55  70 127]
la valeur du coin gauche haut = [48 45 76]
la valeur du coin gauche bas = [ 53  70 128]
```

```python
print("la valeur du coin droit haut =",image3[361,0,:])
print("la valeur du coin droit bas =",image3[361,499,:])
print("la valeur du coin gauche haut =",image3[0,0,:])
print("la valeur du coin gauche bas =",image3[0,499,:])
```

```
la valeur du coin droit haut = [72 44 48]
la valeur du coin droit bas = [127  70  55]
la valeur du coin gauche haut = [76 45 48]
la valeur du coin gauche bas = [128  70  53]
```

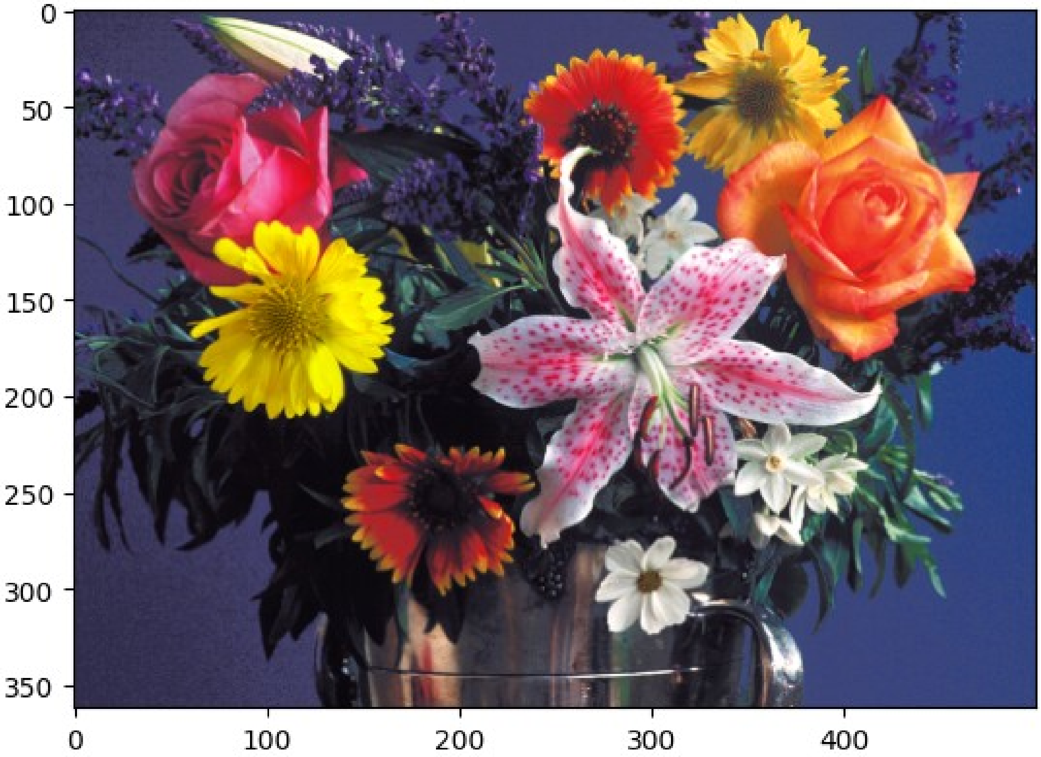
3) La fonction cv2 change l'ordre de position entre la couche de matrice du bleu et du rouge.

Exercice 02

```python
splited_image = cv2.split(image1)

print("la premiere couche: \n",splited_image[0])
print("\n")
print("la deuxieme couche: \n",splited_image[1])
print("\n")
print("la troisieme couche: \n",splited_image[2])
```

```
la premiere couche: 
 [[48 49 52 ... 58 53 53]
 [48 54 49 ... 54 53 47]
 [51 51 53 ... 51 48 48]
 ...
 [49 50 49 ... 57 54 59]
 [48 49 48 ... 53 47 56]
 [48 49 49 ... 58 55 55]]


la deuxieme couche: 
 [[45 46 49 ... 72 69 70]
 [45 50 45 ... 70 68 66]
 [48 49 50 ... 69 67 67]
 ...
 [45 46 45 ... 67 68 70]
 [44 46 44 ... 65 64 67]
 [44 45 45 ... 68 69 70]]


la troisieme couche: 
 [[ 76  76  84 ... 125 129 128]
 [ 77  83  78 ... 129 129 133]
 [ 79  80  82 ... 130 131 132]
 ...
 [ 74  78  73 ... 128 129 124]
 [ 74  77  74 ... 131 133 126]
 [ 72  79  75 ... 128 127 127]]
```

```python
import numpy as np
image_data = np.array(splited_image)
reshaped_image = np.transpose(image_data, (1, 2, 0))
plt.imshow(reshaped_image)
plt.show()
```



```python
#L (luminance), A (composante verte-magenta), et B (composante bleue-
jaune)

# Charger l'image
image = cv2.imread('flowers.tif')

# Convertir l'image en espace de couleur LAB
lab_image = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)

# Diviser l'image en canaux LAB
LAB = cv2.split(lab_image)

LAB
```

```
(array([[52, 53, 57, ..., 81, 79, 80],
        [52, 58, 52, ..., 80, 78, 76],
        [55, 56, 57, ..., 79, 77, 77],
        ...,
        [52, 53, 52, ..., 78, 78, 80],
        [50, 53, 50, ..., 76, 75, 77],
        [50, 53, 52, ..., 79, 79, 80]], dtype=uint8),
 array([[137, 137, 139, ..., 139, 141, 140],
        [138, 138, 139, ..., 140, 142, 143],
        [138, 137, 139, ..., 141, 142, 142],
        ...,
        [137, 139, 137, ..., 143, 142, 140],
        [139, 138, 139, ..., 144, 144, 142],
        [137, 139, 138, ..., 142, 141, 140]], dtype=uint8),
 array([[110, 110, 107, ...,  96,  92,  93],
        [109, 109, 108, ...,  93,  92,  88],
        [109, 109, 109, ...,  91,  90,  89],
        ...,
        [111, 109, 111, ...,  92,  92,  96],
        [110, 109, 110, ...,  89,  87,  93],
        [111, 108, 110, ...,  93,  93,  94]], dtype=uint8))
```

```python
#Teinte, Saturation, Luminance

# Charger l'image
image = cv2.imread('flowers.tif')

# Convertir l'image en espace de couleur HSV
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Diviser l'image en canaux HSV
HSV = cv2.split(hsv_image)

HSV
```

```
(array([[123, 123, 123, ..., 114, 114, 113],
        [123, 124, 124, ..., 114, 114, 113],
        [123, 122, 123, ..., 113, 113, 113],
        ...,
        [124, 124, 124, ..., 116, 114, 115],
        [124, 123, 124, ..., 115, 114, 115],
        [124, 124, 124, ..., 116, 114, 114]], dtype=uint8),
 array([[104, 101, 106, ..., 137, 150, 149],
        [106, 101, 108, ..., 148, 150, 165],
        [100,  99, 100, ..., 155, 162, 162],
        ...,
        [100, 105,  98, ..., 141, 148, 134],
        [103, 103, 103, ..., 152, 165, 142],
        [ 99, 110, 102, ..., 139, 145, 145]], dtype=uint8),
 array([[ 76,  76,  84, ..., 125, 129, 128],
        [ 77,  83,  78, ..., 129, 129, 133],
        [ 79,  80,  82, ..., 130, 131, 132],
        ...,
        [ 74,  78,  73, ..., 128, 129, 124],
        [ 74,  77,  74, ..., 131, 133, 126],
        [ 72,  79,  75, ..., 128, 127, 127]], dtype=uint8))
```

```
print(LAB[0].shape)
print(LAB[1].shape)
print(LAB[2].shape)
```

```
(362, 500)
(362, 500)
(362, 500)
```

```
print(HSV[0].shape)
print(HSV[1].shape)
print(HSV[2].shape)
```

```
(362, 500)
(362, 500)
(362, 500)
```

Partie B

```
from matplotlib import pyplot as plt
import numpy as np
import cv2
from skimage import io, color
```

```
noir16 = np.zeros([16,16])
```

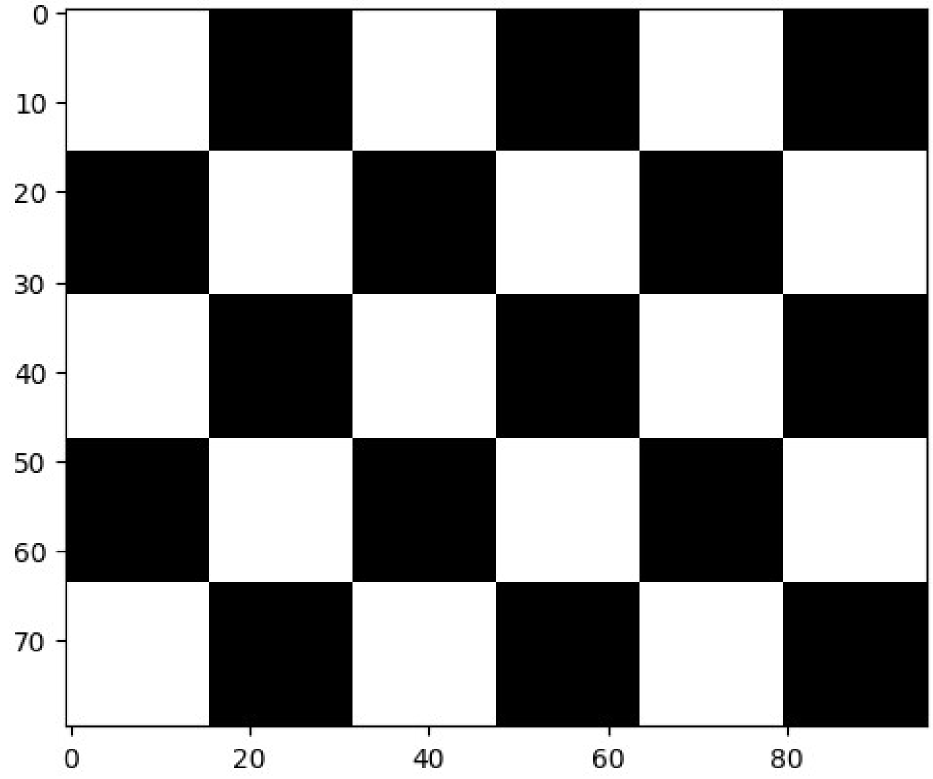
```
blanc16 = np.ones([16,16])
```

```
noir16
```

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
0.]])
```

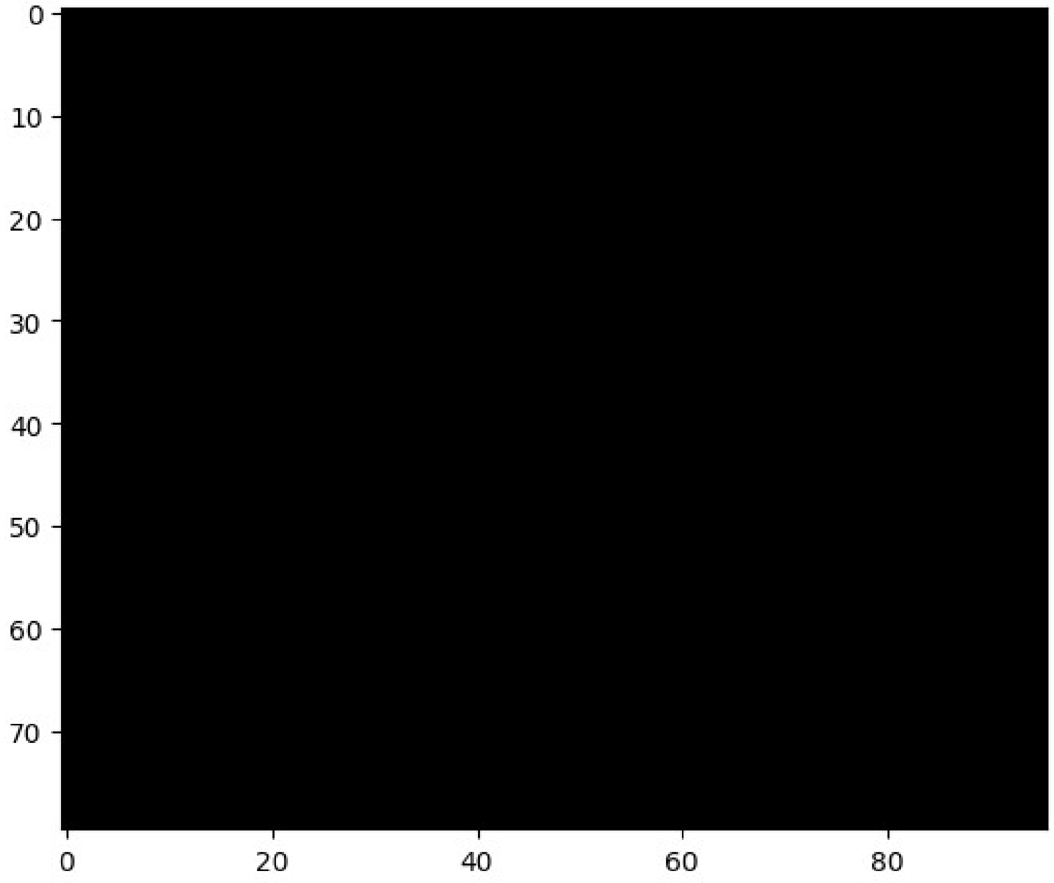
```
NBimage = np.hstack((blanc16, noir16))
#print(np.shape(NBimage))
for j in range(2):
  NBimage = np.hstack((NBimage, np.hstack((blanc16, noir16))))
  #print(np.shape(NBimage))
  #end

Line1 = NBimage
Line2 = np.hstack((Line1[:,16:],blanc16))


NBimage = np.vstack((NBimage, Line2))
NBimage = np.vstack((NBimage, Line1))
NBimage = np.vstack((NBimage, Line2))
NBimage = np.vstack((NBimage, Line1))

plt.imshow(NBimage,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7d3b5451dc00>
```



```
NBimage.nbytes

61440

NBimage.shape

(80, 96)

range(np.shape(NBimage)[0])

range(0, 80)

N = np.matrix(NBimage)
for i in range(np.shape(NBimage)[0]):
  for j in range(np.shape(NBimage)[1]):
    if N[i,j] == 1:
       N [i,j] = 0

N

matrix([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]])
```

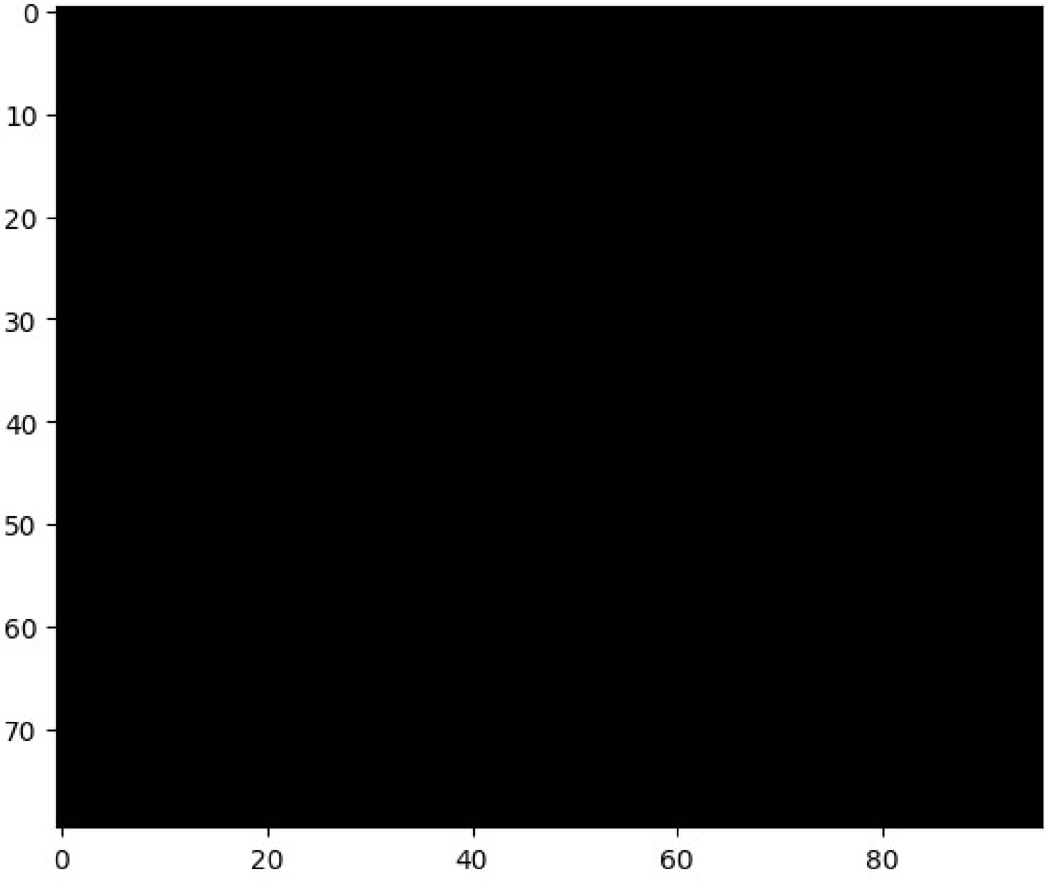
```
N = np.array(N)

io.imshow(N,cmap="gray")

<matplotlib.image.AxesImage at 0x7d3b51b64490>
```



```
B = np.matrix(NBimage)
for i in range(np.shape(NBimage)[0]):
  for j in range(np.shape(NBimage)[1]):
    if B[i,j] == 0:
       B[i,j] = 1

B = B.astype(np.uint8)

B

matrix([[1, 1, 1, ..., 1, 1, 1],
        [1, 1, 1, ..., 1, 1, 1],
        [1, 1, 1, ..., 1, 1, 1],
        ...,
        [1, 1, 1, ..., 1, 1, 1],
        [1, 1, 1, ..., 1, 1, 1],
        [1, 1, 1, ..., 1, 1, 1]], dtype=uint8)
```

```
io.imshow(B)
```

```
<matplotlib.image.AxesImage at 0x7d3b54596f50>
```

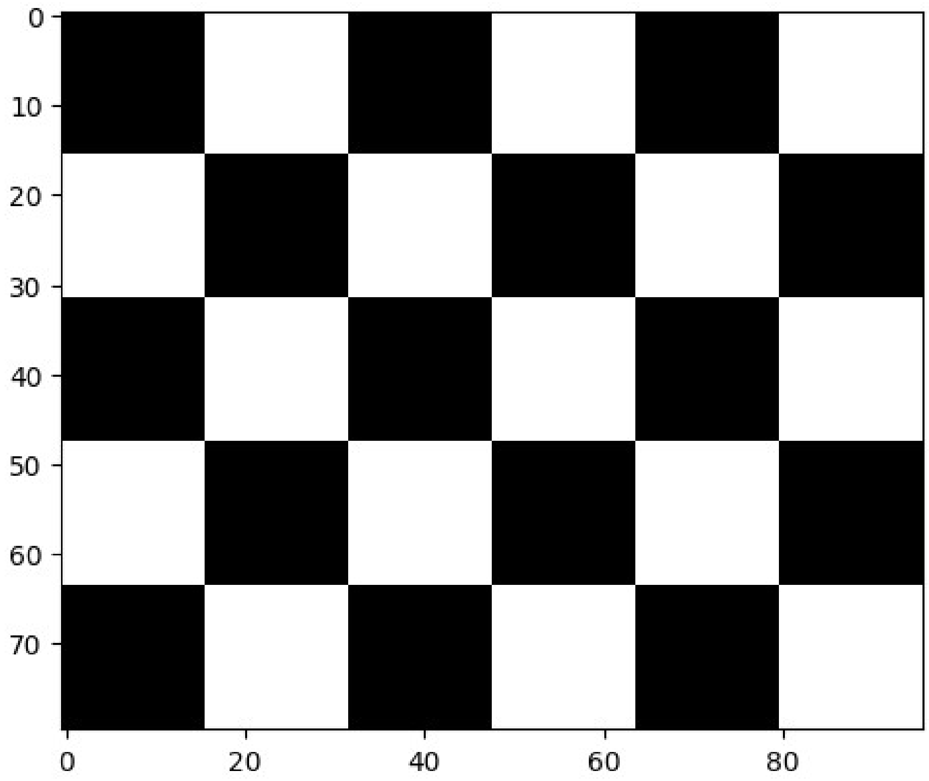


```
Inv = np.matrix(NBimage)
for i in range(np.shape(NBimage)[0]):
  for j in range(np.shape(NBimage)[1]):
    if Inv[i,j] == 0:
       Inv[i,j] = 1
    else:
       Inv[i,j] = 0

np.array(Inv)
```

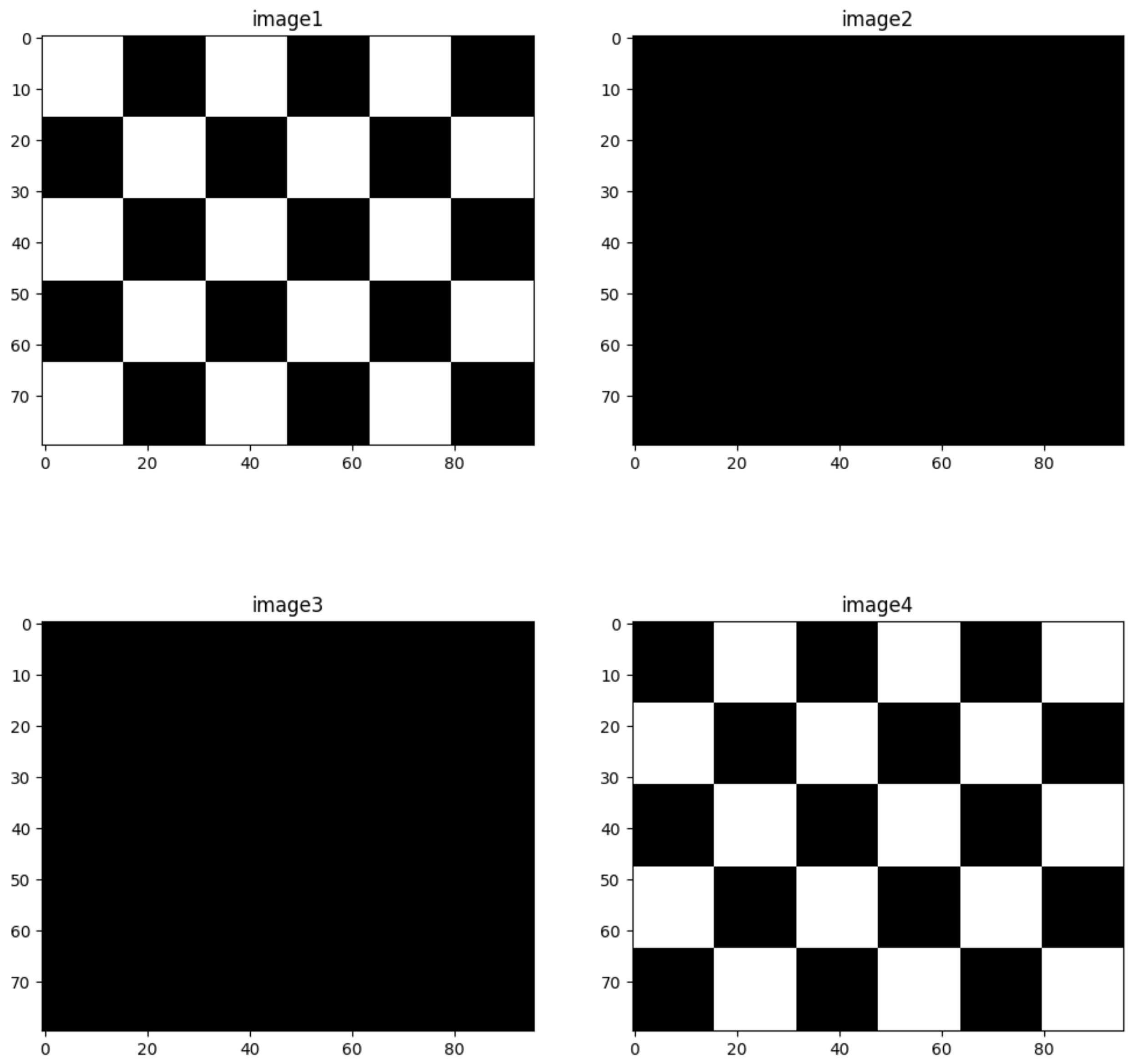
```
array([[0., 0., 0., ..., 1., 1., 1.],
       [0., 0., 0., ..., 1., 1., 1.],
       [0., 0., 0., ..., 1., 1., 1.],
       ...,
       [0., 0., 0., ..., 1., 1., 1.],
       [0., 0., 0., ..., 1., 1., 1.],
       [0., 0., 0., ..., 1., 1., 1.]])
```

```
plt.imshow(Inv,cmap = "gray")
```

```
<matplotlib.image.AxesImage at 0x7d3b5451cac0>
```



```
img1 = NBimage
img2 = N
img3 = B
img4 = Inv

fig = plt.figure(figsize=(12,12))
ax1 = fig.add_subplot(2,2,1)
ax1.imshow(img1, cmap="gray")
ax1.title.set_text("image1")
ax2 = fig.add_subplot(2,2,2)
ax2.imshow(img2,cmap="gray")
ax2.title.set_text('image2')
ax2 = fig.add_subplot(2,2,3)
ax2.imshow(img3,cmap="gray")
ax2.title.set_text('image3')
ax2 = fig.add_subplot(2,2,4)
ax2.imshow(img4,cmap="gray")
ax2.title.set_text('image4')
```



```
plt.savefig('saved_image.tif')

<Figure size 640x480 with 0 Axes>
```

J'ai un probleme aux affichage d'une image blanche. Mais on peut voire que l'image blanche est une matrice de taille 80x96 et elle contient seulements des uns.

exercice 02

```
image4 = io.imread("saved_image.tif",False)

image4.shape

(480, 640, 4)
```

```
cv2.imwrite("flowers2.tif", image4)
```

```
True
```

Exercice 03

```
zeros = np.zeros([25,25])
ones = np.ones([25,25])
Blanc = np.array([ones,ones,ones])
Noir = np.array([zeros,zeros,zeros])
Rouge = np.array([ones,zeros,zeros])
Jaune = np.array([ones,ones,zeros])
Vert = np.array([zeros,ones,zeros])
Cyan = np.array([zeros,zeros,ones])
Bleu = np.array([zeros,zeros,ones])
Magenta = np.array([ones,zeros,ones])

Rouge = np.transpose(Rouge, (1, 2, 0))
Blanc = np.transpose(Blanc, (1, 2, 0))
Noir = np.transpose(Noir, (1, 2, 0))
Jaune = np.transpose(Jaune, (1, 2, 0))
Vert = np.transpose(Vert, (1, 2, 0))
Cyan = np.transpose(Cyan, (1, 2, 0))
Bleu = np.transpose(Bleu, (1, 2, 0))
Magenta = np.transpose(Magenta, (1, 2, 0))
```

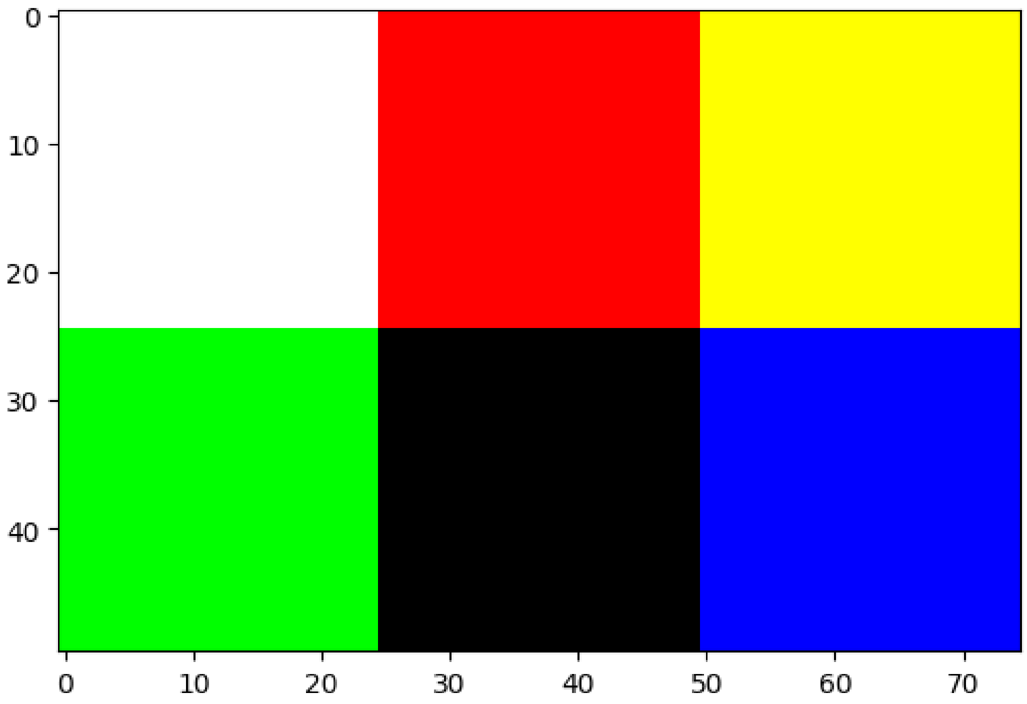
```
Rouge
```

```
array([[[1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.],
        ...,
        [1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.]],

       [[1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.],
        ...,
        [1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.]],

       [[1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.],
        ...,
        [1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.]],

       ...,

       [[1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.],
        ...,
        [1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.]],

       [[1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.],
        ...,
        [1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.]],

       [[1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.],
        ...,
        [1., 0., 0.],
        [1., 0., 0.],
        [1., 0., 0.]]]])
```

```
image6couleurs1 = np.hstack((np.hstack((Blanc*255, Rouge*255)),
Jaune*255))
image6couleurs2 = np.hstack((np.hstack((Vert*255, Noir*255)),
Bleu*255))
image6couleurs = np.vstack((image6couleurs1,image6couleurs2))

plt.imshow(image6couleurs)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for
imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

```
<matplotlib.image.AxesImage at 0x7d3b513a68f0>
```



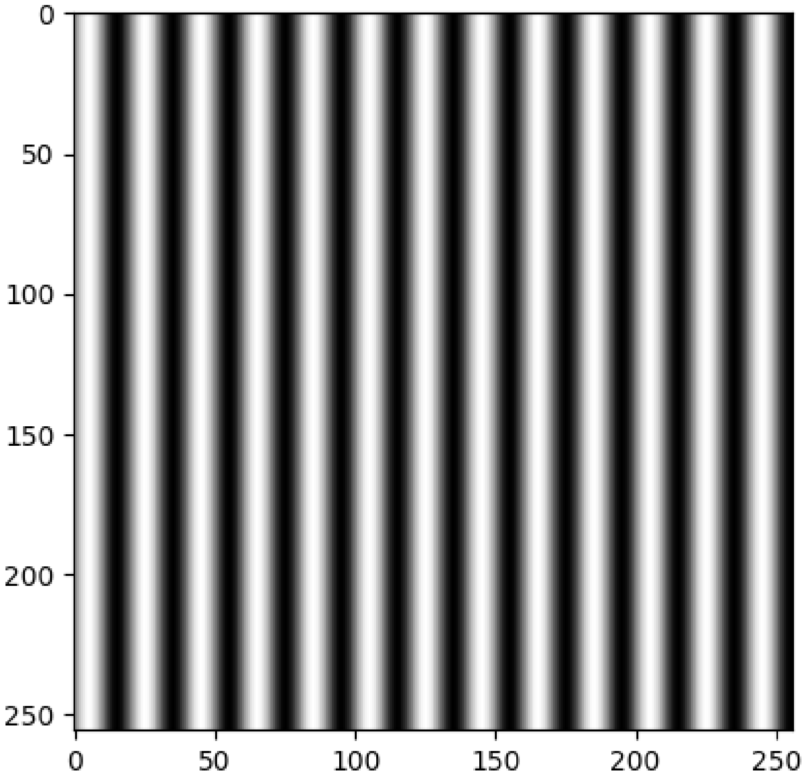
```
image6couleurs.nbytes # Cette fonction fait le calcule exacte de la taille memoire de cette matrice.
```

```
90000
```

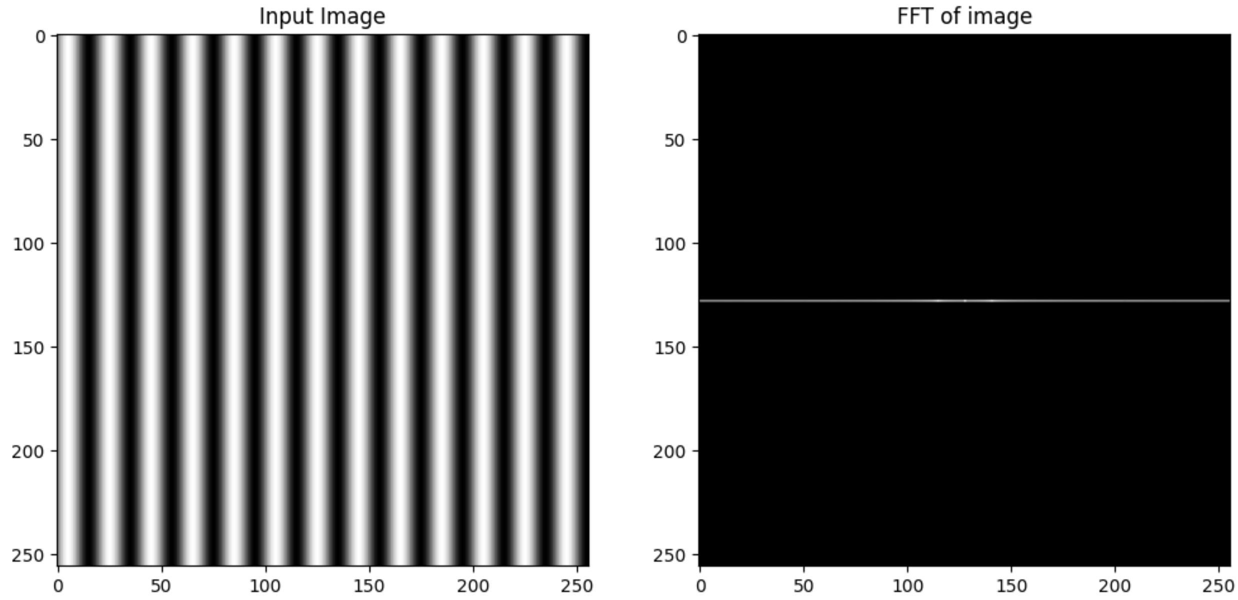
Exercice 04

```
d=20
x = np.arange(256)
y = np.sin(2*np.pi*x/d)
y += max(y)
img = np.array([[y[j]*127 for j in range(256)] for i in range(256)], dtype = np.uint8)
plt.imshow(img,cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7d3b5124c490>
```



```
dft = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)
magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0]+1,
dft_shift[:,:,1]+1))

fig = plt.figure(figsize=(12,12))
ax1 = fig.add_subplot(2,2,1)
ax1.imshow(img,cmap="gray")
ax1 .title.set_text('Input Image')

ax2 = fig.add_subplot(2,2,2)
ax2.imshow(magnitude_spectrum, cmap = "gray")
ax2.title.set_text('FFT of image')
plt.show()
```



```
#La transformé de fourier
dft
```

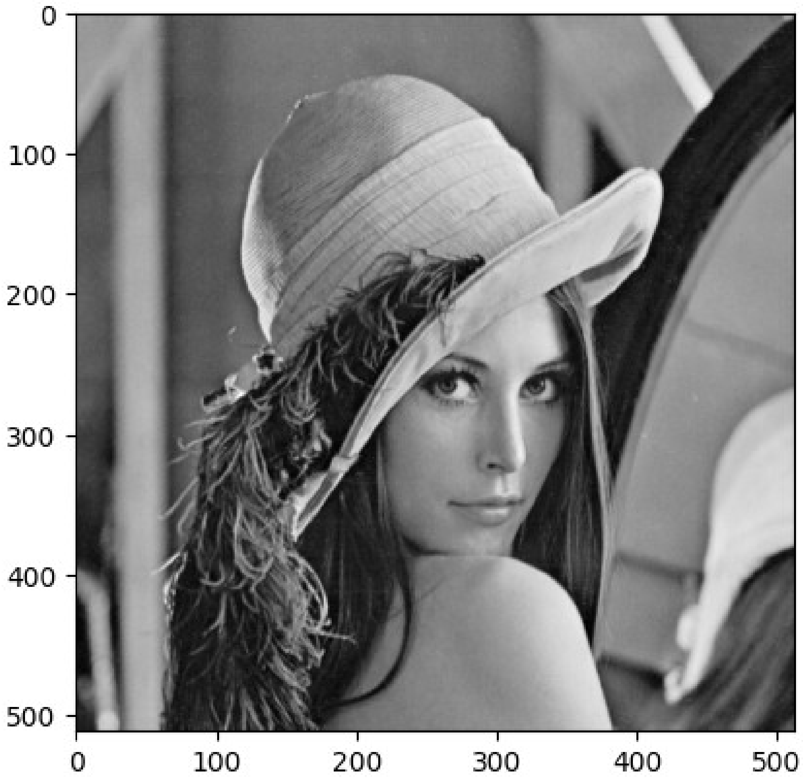
```
array([[[ 8.3791360e+06,  0.0000000e+00],
        [ 8.7241109e+04,  7.2273555e+03],
        [ 8.8619328e+04,  1.5709379e+04],
        ...,
        [ 9.0611984e+04, -2.4964957e+04],
        [ 8.8619328e+04, -1.5709379e+04],
        [ 8.7241109e+04, -7.2273555e+03]],

       [[ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        ...,
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00]],

       [[ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        ...,
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00]],

       ...,

       [[ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        ...,
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00]],

       [[ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        ...,
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00]],

       [[ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        [ 0.0000000e+00,  0.0000000e+00],
        ...,
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00],
        [ 0.0000000e+00, -0.0000000e+00]]], dtype=float32)
```

On va avoir des amplitude hautes autour de la frequence des pixels de l'image

Partie C

Exercice 01

```
Lena = io.imread("lenna512.bmp",True)
plt.imshow(Lena,cmap="gray")
plt.show()
```



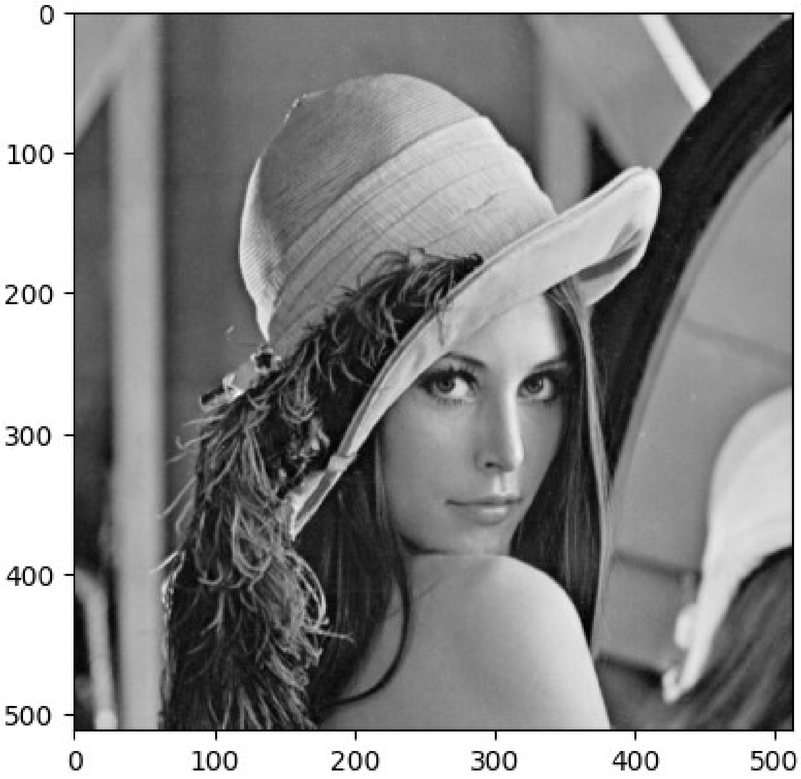
```
print("la dimension de l'image est: ", Lena.shape)
```

```
la dimension de l'image est:  (512, 512)
```

```
from skimage.measure import block_reduce
import numpy as np
factor = 1
downscaled_image = block_reduce(Lena, factor, np.mean)
```

```
plt.imshow(downscaled_image, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7d3b5109a860>
```

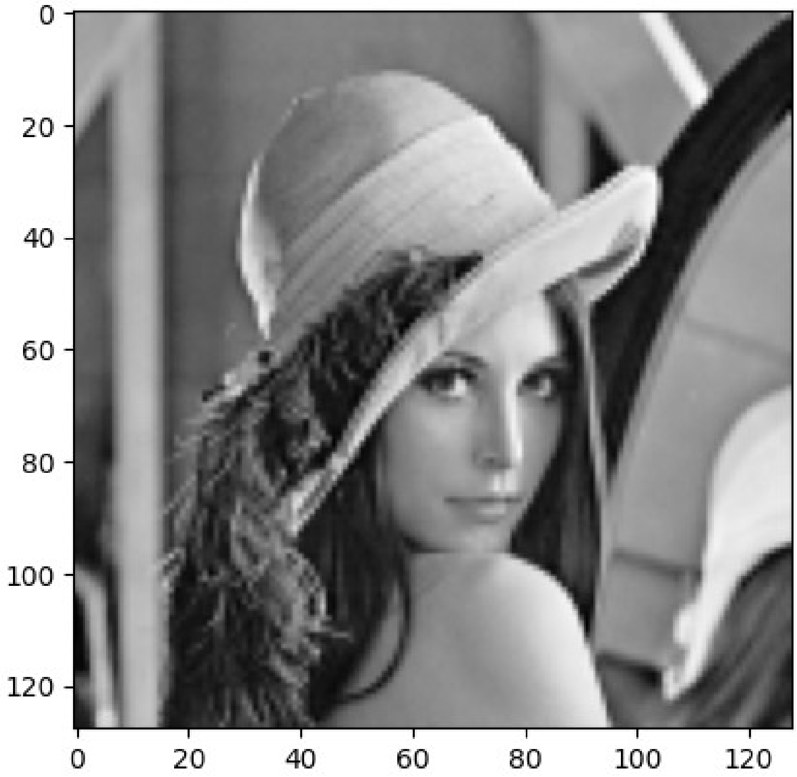


```
print("la taille de l'image apres l'échantionnage
est:" ,downscaled_image.shape)
```

```
la taille de l'image apres l'échantionnage est: (512, 512)
```

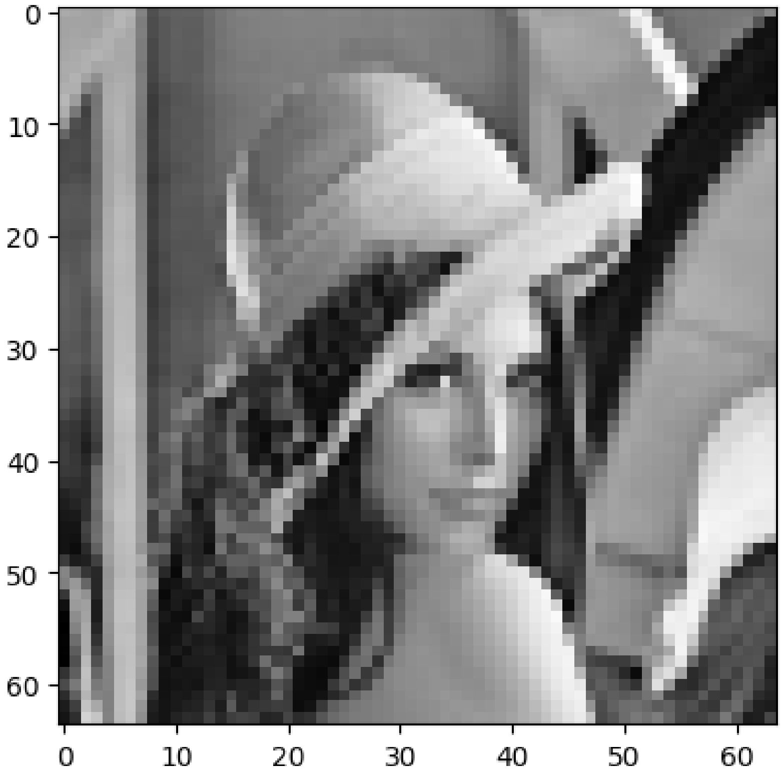
```
factor = 4
downscaled_image = block_reduce(Lena, factor, np.mean)
plt.imshow(downscaled_image, cmap = 'gray')
print('la dimesion est: ',downscaled_image.shape)
```

```
la dimesion est:  (128, 128)
```



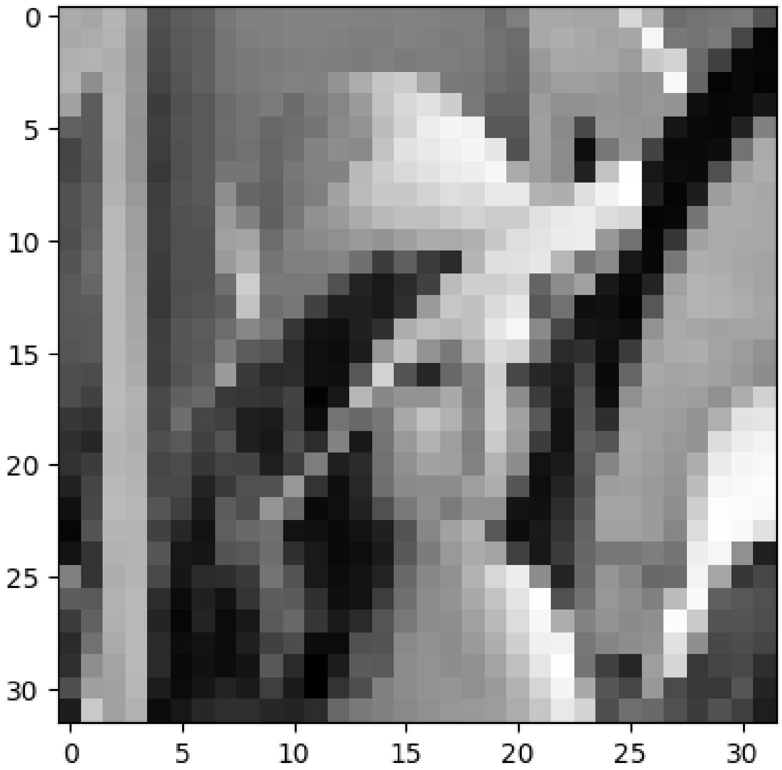
```
factor = 8
downscaled_image = block_reduce(Lena, factor, np.mean)
plt.imshow(downscaled_image, cmap = 'gray')
print('la dimesion est: ',downscaled_image.shape)
```

```
la dimesion est:  (64, 64)
```



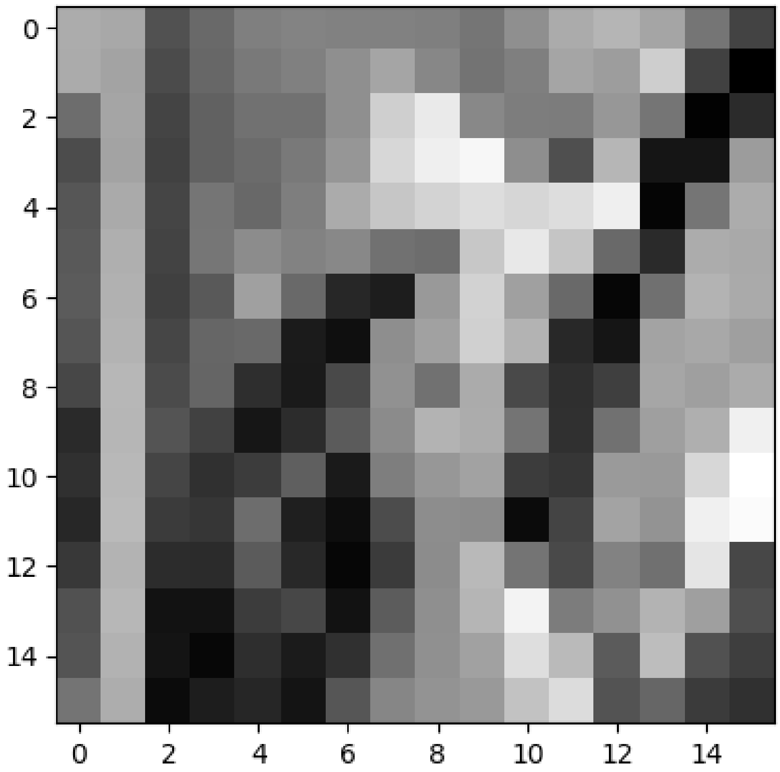
```
factor = 16
downscaled_image = block_reduce(Lena, factor, np.mean)
plt.imshow(downscaled_image, cmap = 'gray')
print('la dimesion est: ',downscaled_image.shape)
```

```
la dimesion est:  (32, 32)
```

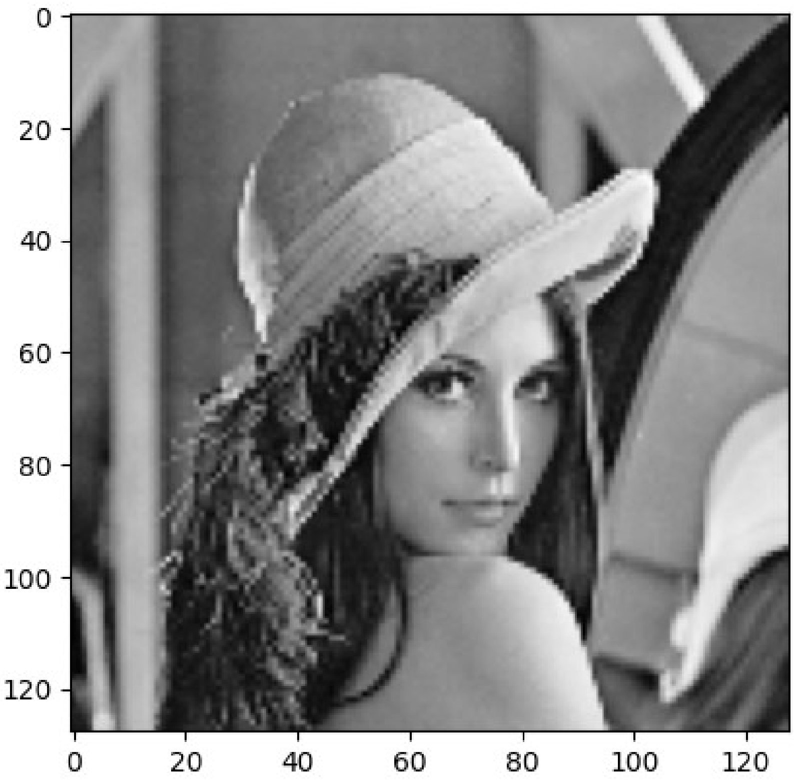


```
factor = 32
downscaled_image = block_reduce(Lena, factor, np.mean)
plt.imshow(downscaled_image, cmap = 'gray')
print('la dimesion est: ',downscaled_image.shape)
```

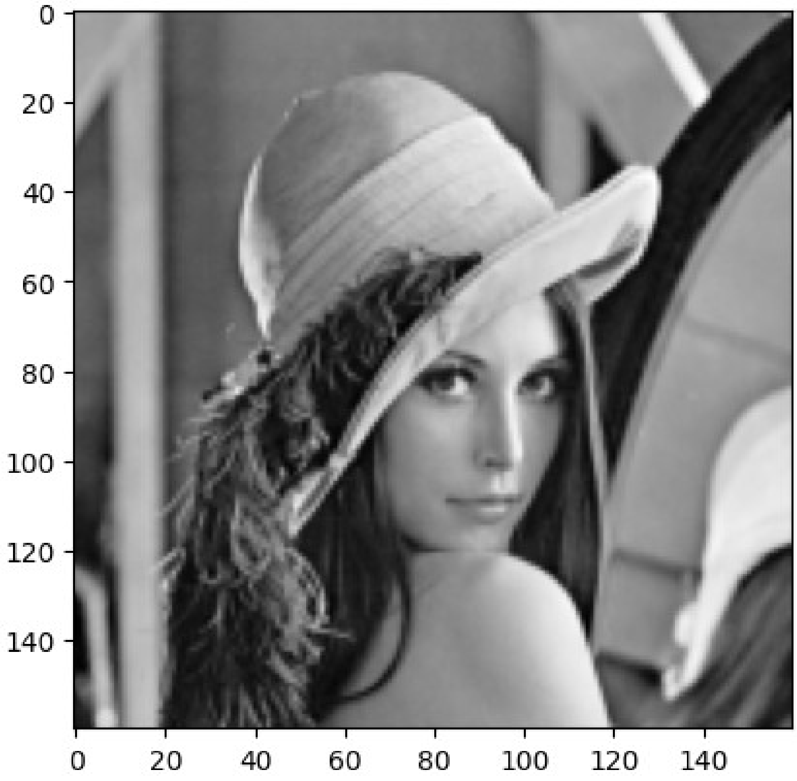
```
la dimesion est:  (16, 16)
```



```
from skimage.transform import rescale
img_rescaled = rescale(Lena, 1.0/4.0, anti_aliasing = False)
plt.imshow(img_rescaled, cmap="gray")

<matplotlib.image.AxesImage at 0x7d3b51301c60>
```



```
from skimage.transform import resize
img_resize = resize(Lena, (160,160),anti_aliasing = True)
plt.imshow(img_resize, cmap="gray")

<matplotlib.image.AxesImage at 0x7d3b5133b3d0>
```



```
Lena.itemsize * 8
```

```
8
```

```
np.max(Lena)
```

```
245
```

On remarque que la valeur maximal est 245 ce qui implique que l'image elle est codé sur 8 bits

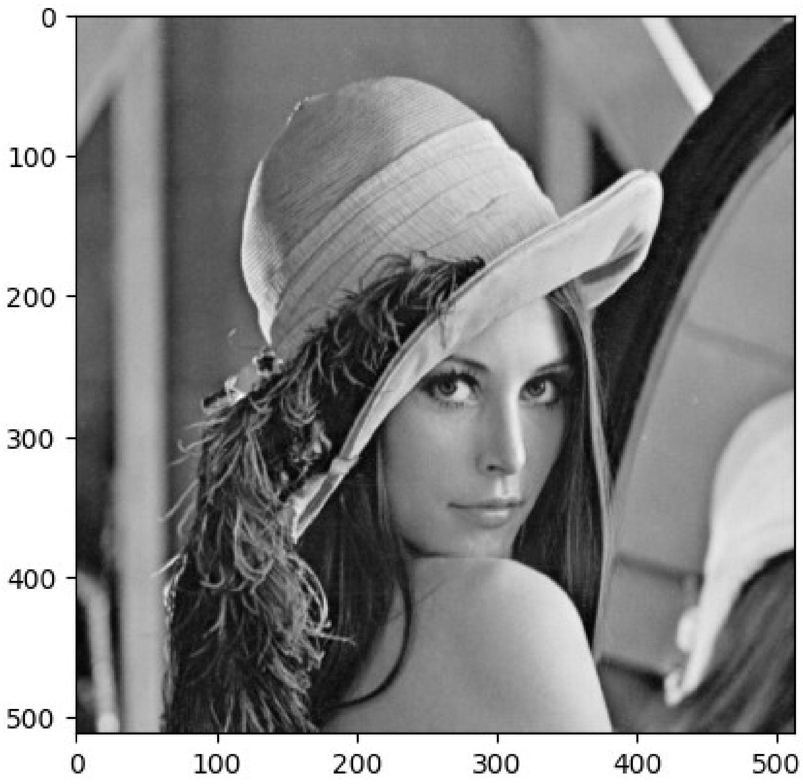
```
Lena128 = Lena / 2
Lena128 = np.round(Lena128, decimals=0)
```

```
np.max(Lena128)
```

```
122.0
```

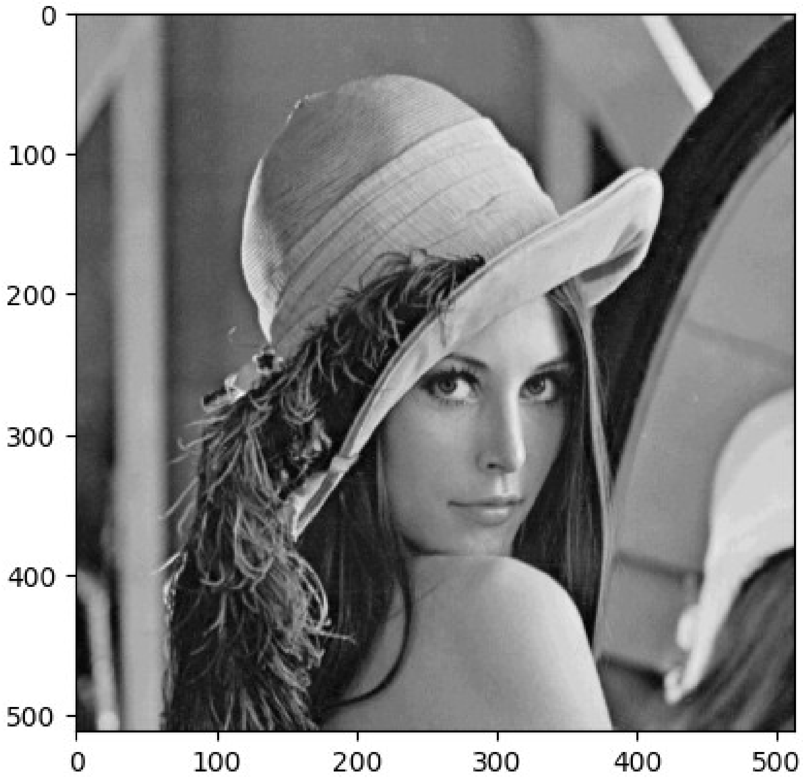
```
Lena64 = Lena / 4
Lena64 = np.round(Lena128, decimals=0)*4
plt.imshow(Lena64, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7d3b51339d50>
```



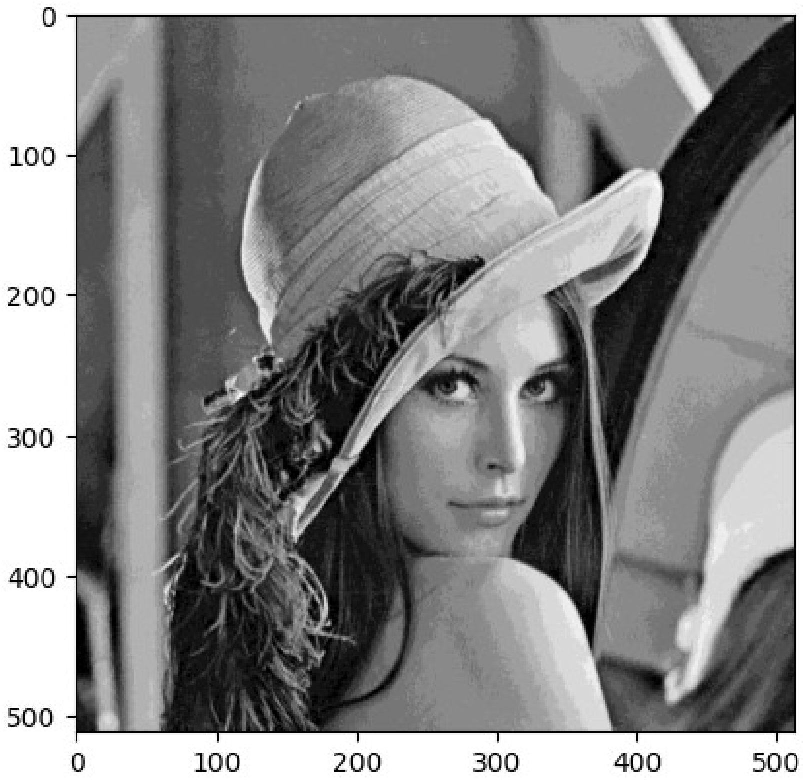
```
Lena32 = Lena / 8
Lena32 = np.round(Lena32, decimals=0)*8
plt.imshow(Lena32, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7d3b51c29a80>
```

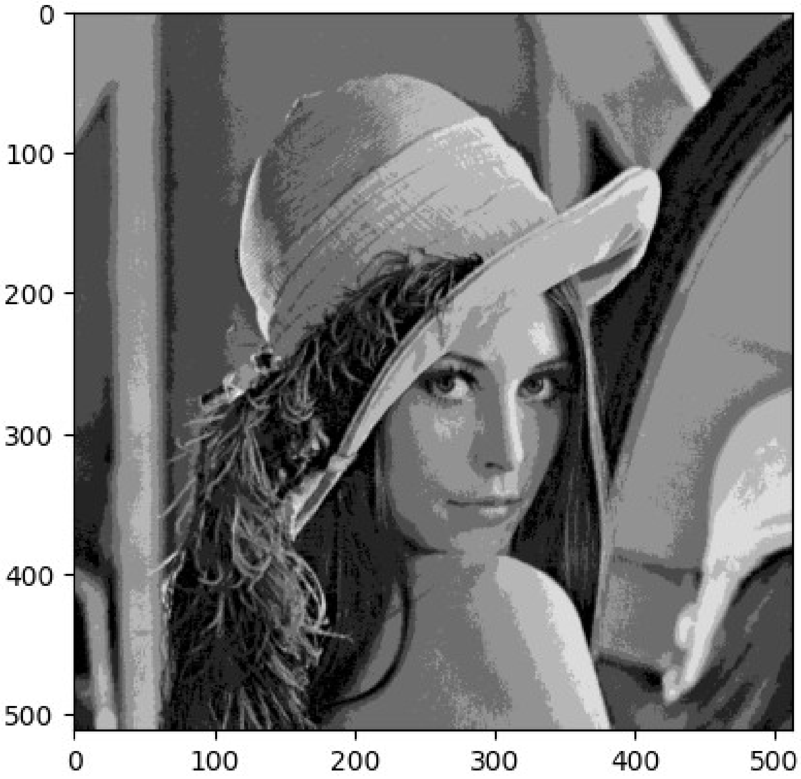


```
Lena16 = Lena / 16
Lena16 = np.round(Lena16, decimals=0)*16
plt.imshow(Lena16, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7d3b514da260>
```

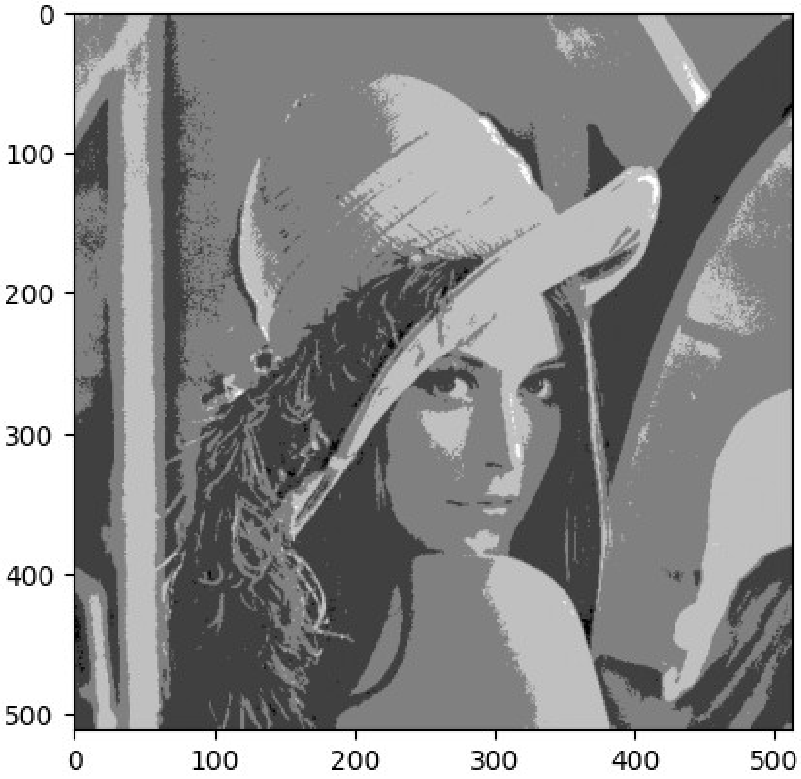


```
Lena8 = Lena / 32
Lena8 = np.round(Lena8, decimals=0)*32
plt.imshow(Lena8, cmap="gray")

<matplotlib.image.AxesImage at 0x7d3b52773970>
```



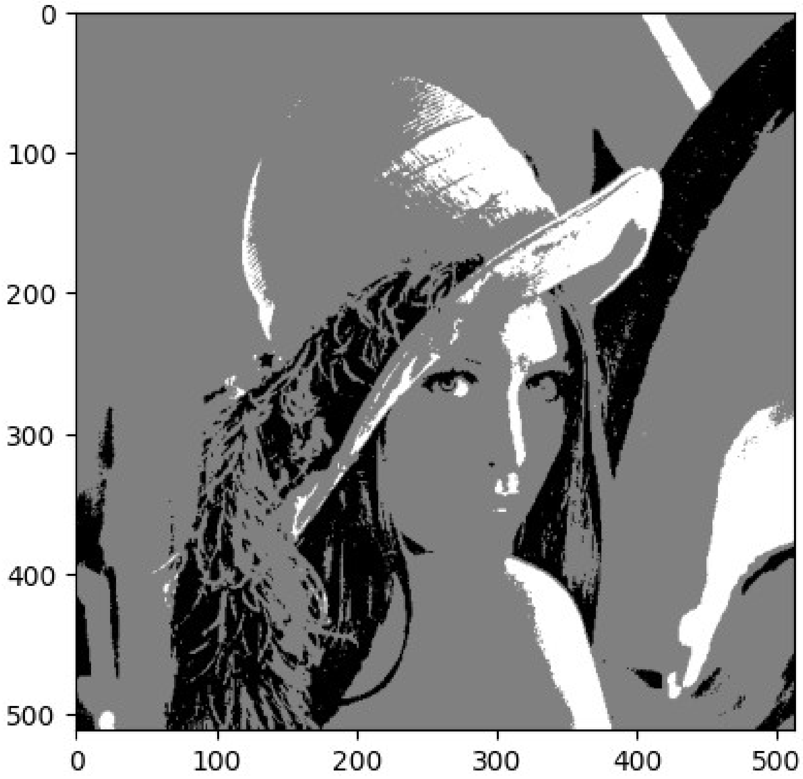
```
Lena4 = Lena / 64
Lena4 = np.round(Lena4, decimals=0)*64
plt.imshow(Lena4, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7d3b51c412d0>
```



```
Lena2 = Lena / 128
Lena2 = np.round(Lena2, decimals=0)*128
plt.imshow(Lena2, cmap="gray")
```

```
<matplotlib.image.AxesImage at 0x7d3b51199c30>
```



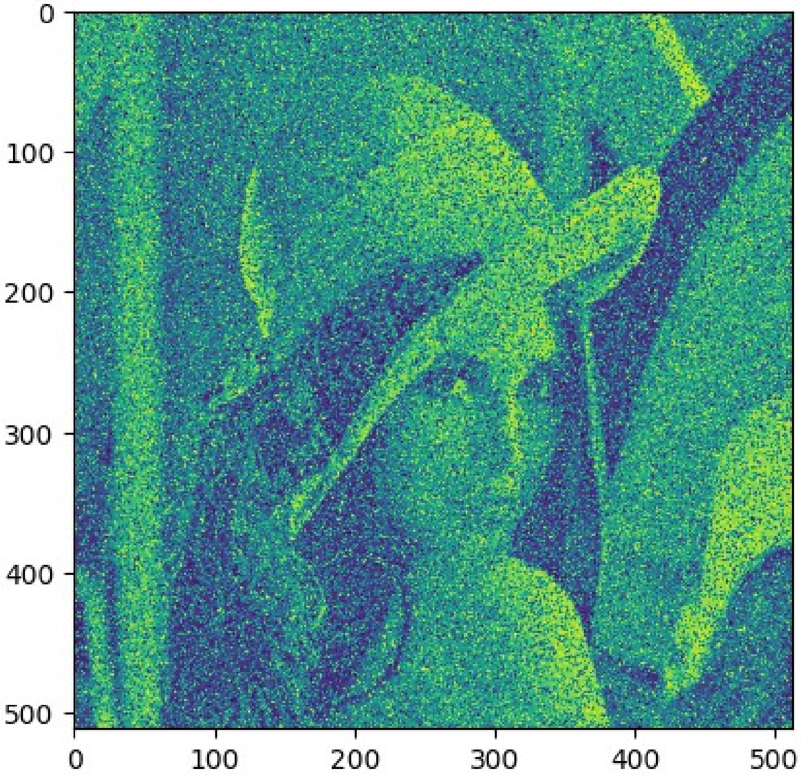
```
#L'erreur de quantification
e1 = np.square(np.subtract(Lena,Lena128)).mean()
print('e1 = ',e1)

e2 = np.square(np.subtract(Lena,Lena64)).mean()
print('e2 = ',e2)

e3 = np.square(np.subtract(Lena,Lena32)).mean()
print('e3 = ',e3)

e4 = np.square(np.subtract(Lena,Lena16)).mean()
print('e4 = ',e4)

e5 = np.square(np.subtract(Lena,Lena8)).mean()
print('e5 = ',e5)

#l'erreur quadratique
```

```
e1 =  4419.652454376221
e2 =  17679.058197021484
e3 =  5.511199951171875
e4 =  20.810455322265625
e5 =  88.55068969726562
```

Partie D

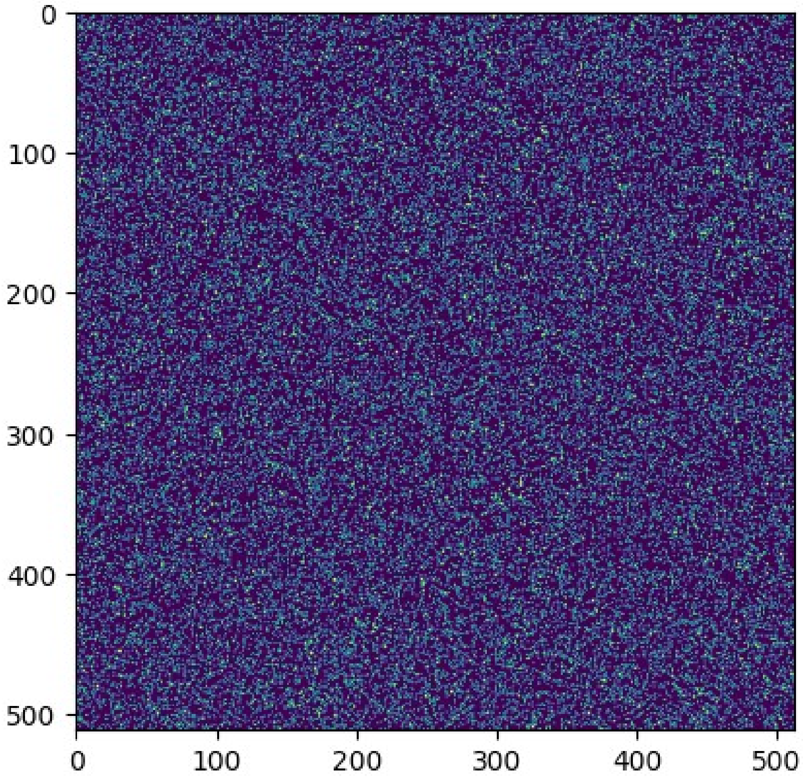
```
import cv2
import numpy as np
from skimage.util import random_noise
from matplotlib import pyplot as plt
from skimage.filters import gaussian
from skimage import io, img_as_float
from skimage.metrics import peak_signal_noise_ratio

noisyimage=random_noise(Lena,mode='s&p',amount=0.3)
plt.imshow(noisyimage)
```

```
<matplotlib.image.AxesImage at 0x7d3b50ed2f20>
```

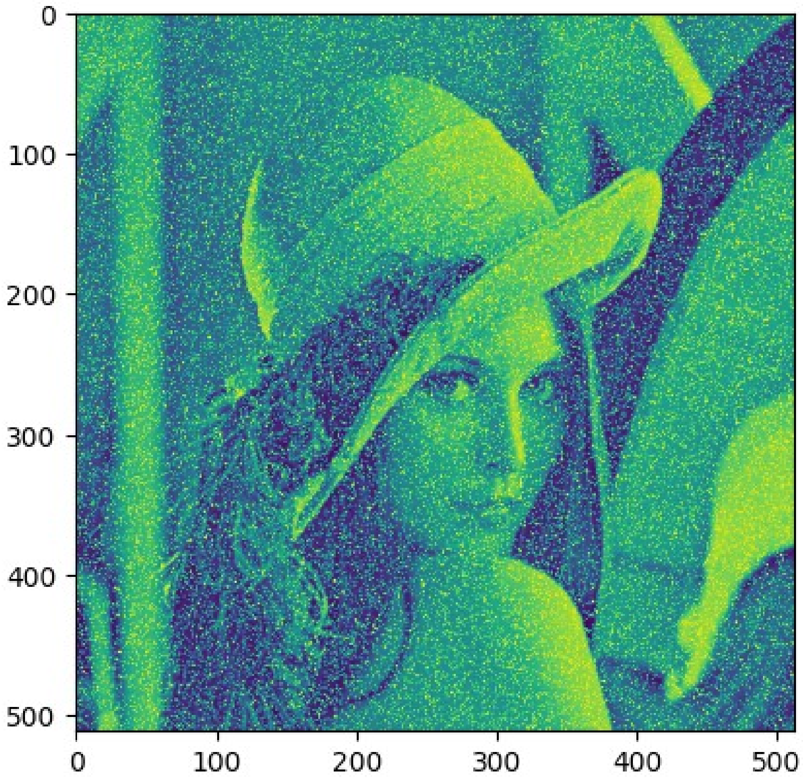


```
gauss=np.random.normal(0,1,Lena.size)
gauss=gauss.reshape(Lena.shape[0],Lena.shape[1]).astype('uint8')
plt.imshow(gauss)
```

```
<matplotlib.image.AxesImage at 0x7d3b50f48670>
```

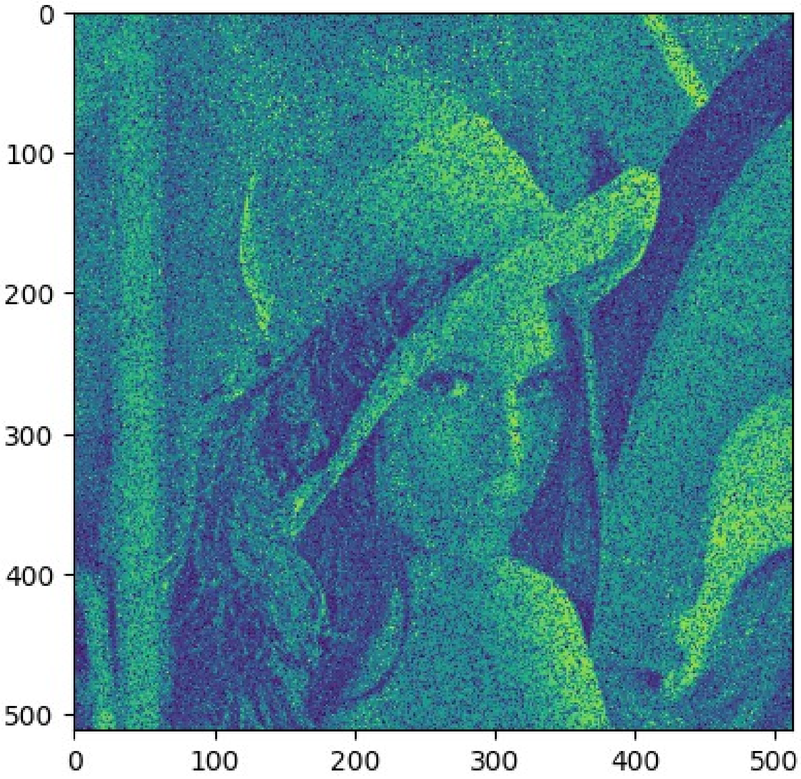


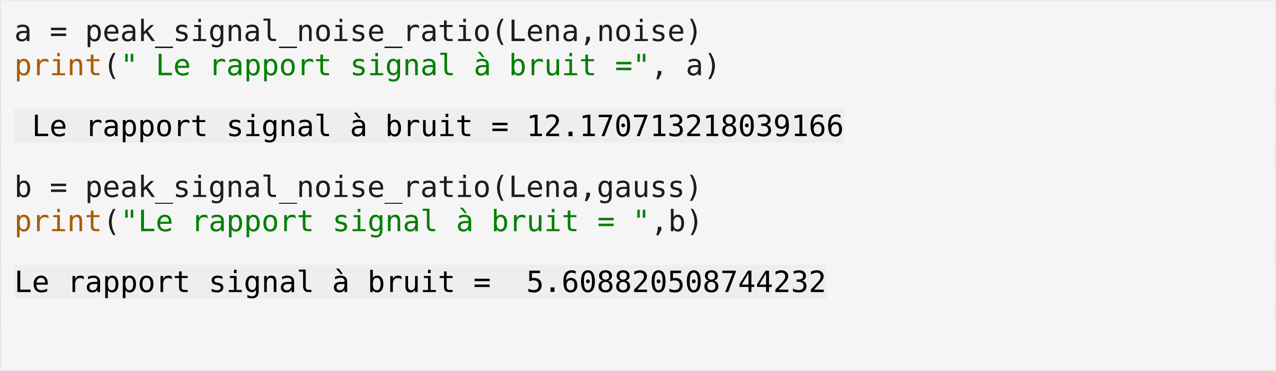
```
image_gauss=cv2.add(Lena,gauss)
plt.imshow(image_gauss)

<matplotlib.image.AxesImage at 0x7d3b50fa3550>
```



```
gauss=np.random.normal(0,1,Lena.size)
gauss=gauss.reshape(Lena.shape[0],Lena.shape[1]).astype('uint8')
noise=Lena+Lena*gauss
plt.imshow(noise)
```

```
<matplotlib.image.AxesImage at 0x7d3b50e1a800>
```





```
a = peak_signal_noise_ratio(Lena,noise)
print(" Le rapport signal à bruit =", a)

 Le rapport signal à bruit = 12.170713218039166

b = peak_signal_noise_ratio(Lena,gauss)
print("Le rapport signal à bruit = ",b)

Le rapport signal à bruit =  5.608820508744232
```