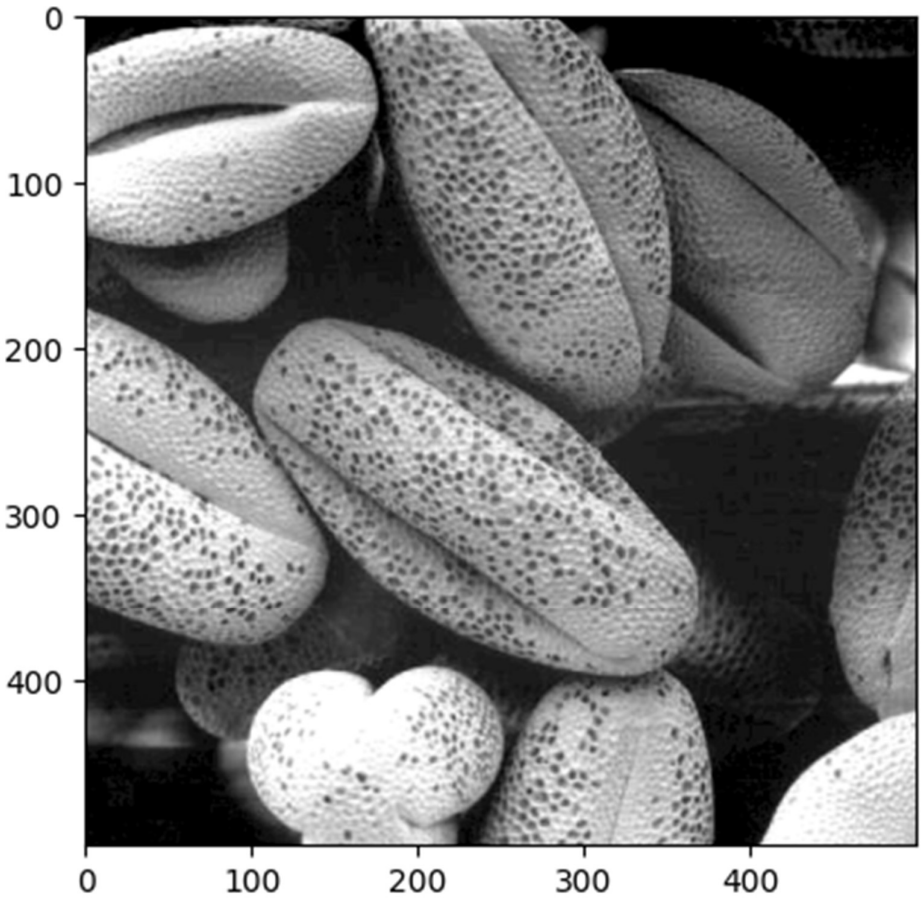
Code :

```
#Loads and display image
import cv2
import numpy as np
from matplotlib import pyplot as plt
image_path = r"../dip_Images/"

sample_image = cv2.imread(image_path+"Fig0316(3)(third_from_top).tif",0)
window_name = "sample"
plt.imshow(sample_image,cmap='gray')
```

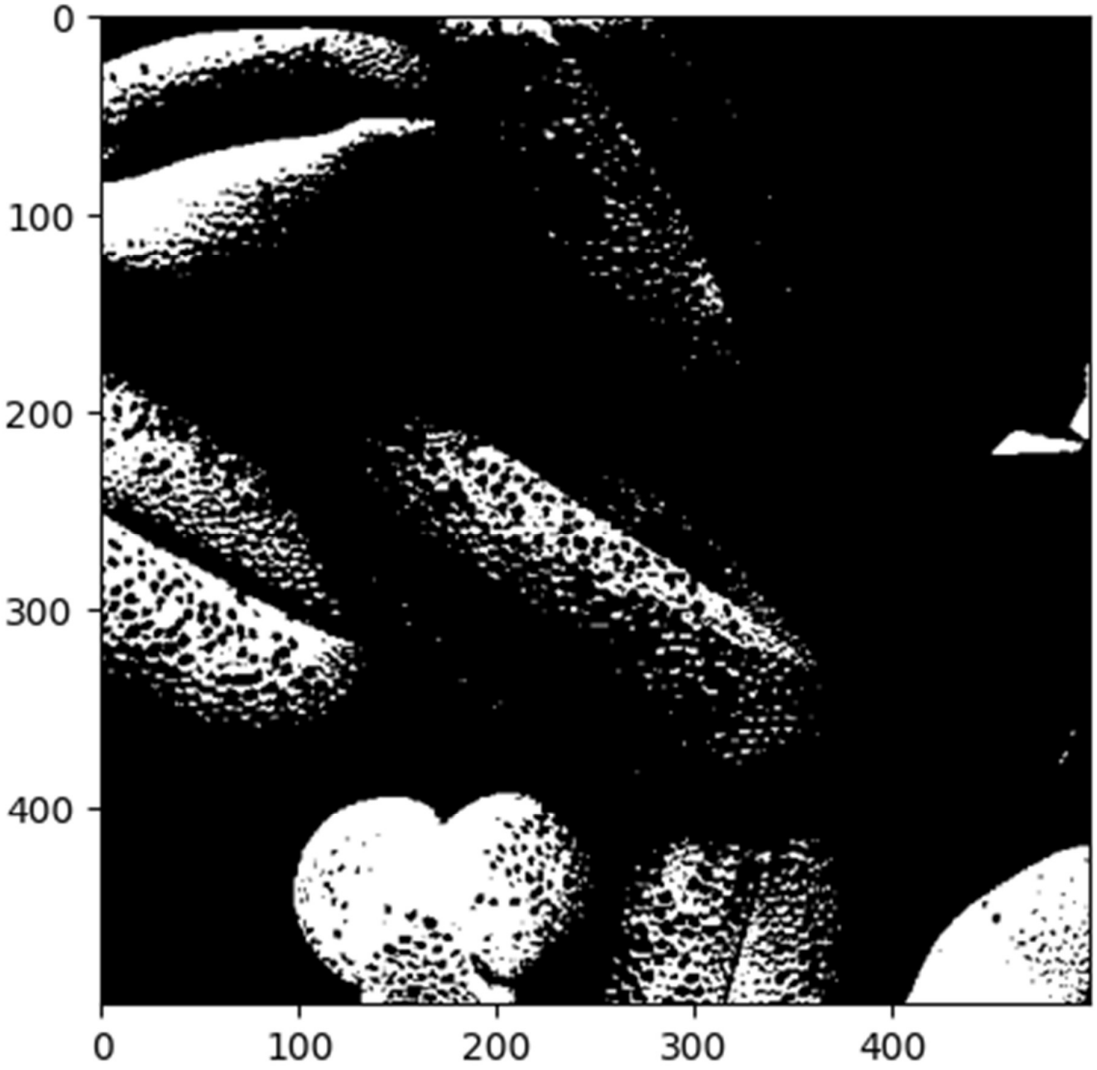
Output:



Code:

```
def gray_level_slice(image,lower,upper,value,background=False):
  """
  Perform gray-level slicing on an image.

    Parameters:
    - image: 2D array representing the image.
    - lower_intensity: Lower bound of intensity range (inclusive).
    - upper_intensity: Upper bound of intensity range (inclusive).
    - replacement_value: Value to replace pixels within the intensity range.

    Returns:
    - Processed image with specified intensity replacements.
  """
  new = image.copy()
  assert upper in range(0,256)
  assert lower in range(0,256)
  (row,col) = new.shape
  for i in range(row):
    for j in range(col):
      if background:
        if lower <= new[i][j] <= upper:
          new[i][j] = value
      else:
        if lower <= new[i][j] <= upper:
          new[i][j] = value
        else:
          new[i][j] = 0
  return new

new = gray_level_slice(sample_image,200,255,255,False)
plt.imshow(new,cmap='gray')
```

Output:

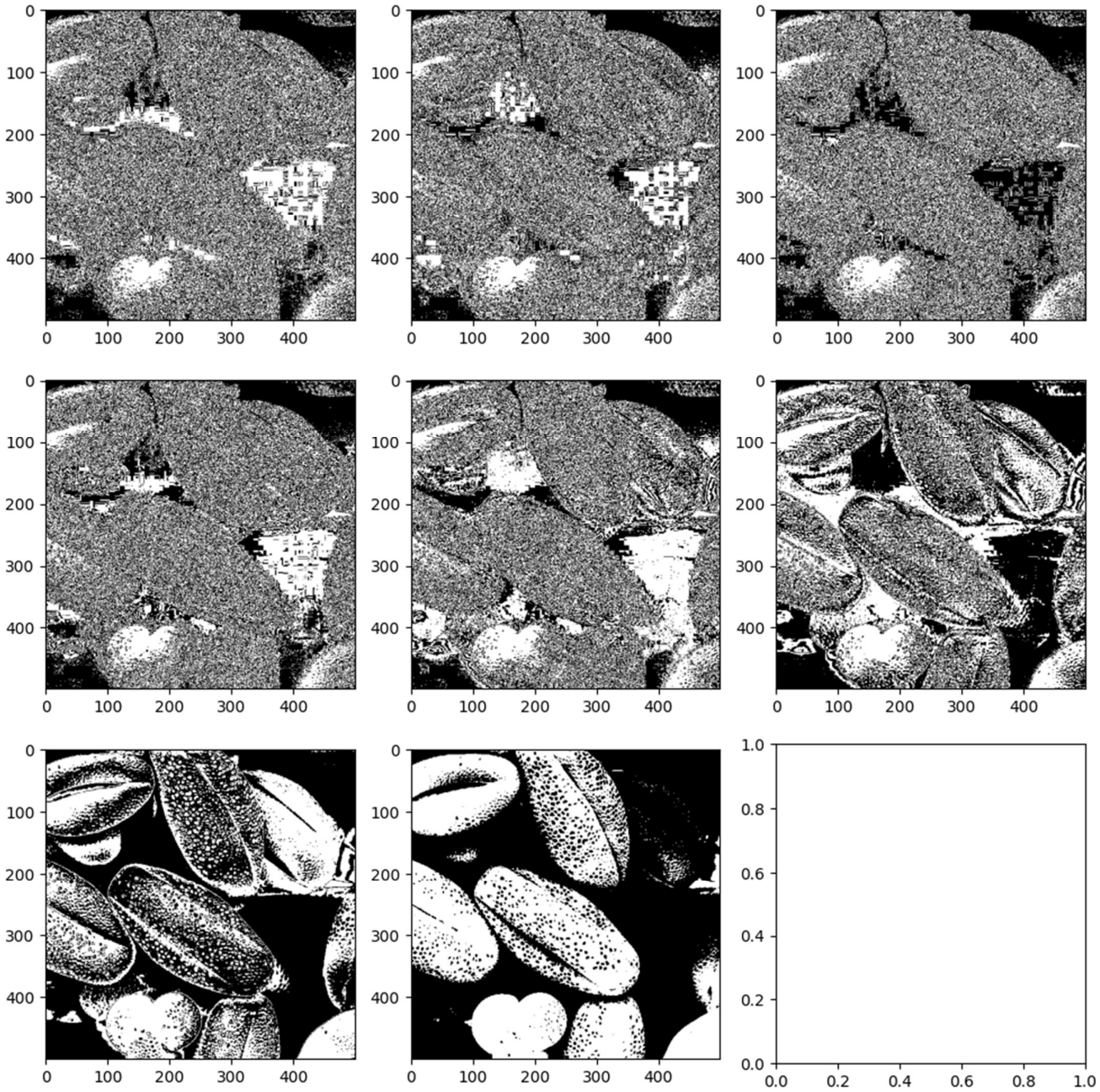


Code:

```
#Bit Plane Slicing
def bit_slice(image):
  """
     Perform bit-plane slicing on an 8-bit grayscale image.

     Parameters:
     - image: 2D array representing the 8-bit grayscale image.

     Returns:
     - A list containing bit-plane sliced images.
       Each element in the list represents a different bit-plane,
       where higher index corresponds to higher bit position (7 to 0).
       Pixel values are set to 255 for bit value 1, and 0 for bit value 0.
  """

  planes = []
  (row,col) = image.shape
  for plane in range(8):
    new = image.copy()
    for i in range(row):
      for j in range(col):
        pixel = image[i][j]
        pixel = pixel>>int(plane)
        pixel = pixel&1
        if pixel == 1:
          new[i][j] = 255
        else:
          new[i][j] = 0
    planes.append(new)
  return planes

#Bit Plane slicing
planes = bit_slice(sample_image)
rows = 3
cols = 3
fig, axes = plt.subplots(rows, cols, figsize=(10, 10))

for i in range(rows):
    for j in range(cols):
        index = i * cols + j
        if index >= len(planes):
```

```
        break
    axes[i, j].imshow(planes[index], cmap='gray')


plt.tight_layout()
plt.show()
```

Output:

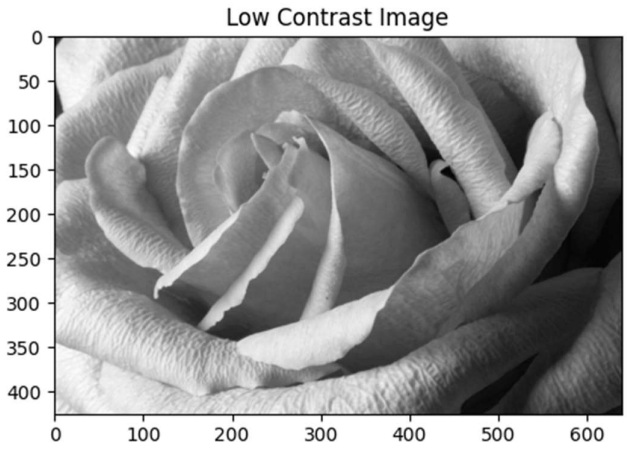


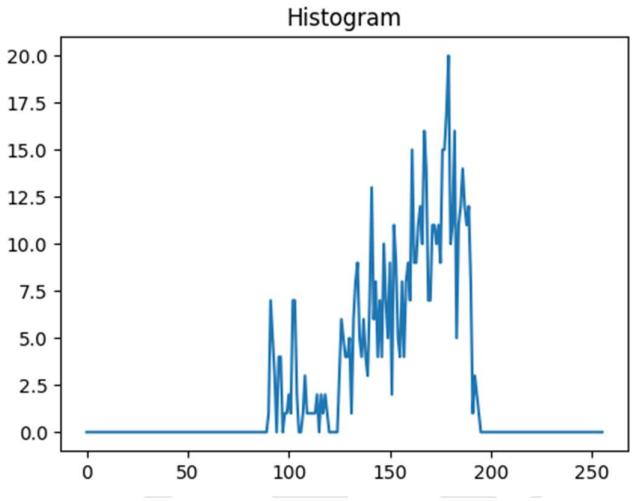
Code:

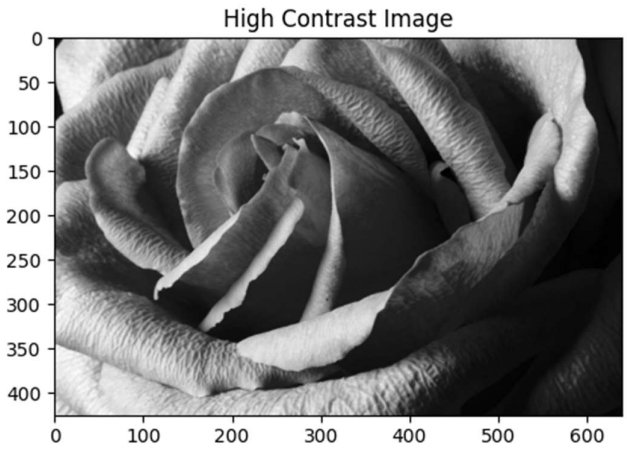
```
#Contrast stretching
def contrast_stretch_pixel(r1,r2,s1,s2,r):
    """
    Perform contrast stretching on a single pixel value.

    Parameters:
    - r1 (int): Lower limit of the input pixel intensity range.
    - r2 (int): Upper limit of the input pixel intensity range.
    - s1 (int): Lower limit of the output pixel intensity range.
    - s2 (int): Upper limit of the output pixel intensity range.
    - r (int): Input pixel intensity value to be transformed.

    Returns:
    - s (int): Transformed pixel intensity value after contrast stretching.
    """
    if 0 <= r <= r1:
        s = s1/r1 * r
    elif r1 < r <= r2:
        s = (s2-s1)*(r-r1)/(r2-r1) + s1
    else:
        s = (255-s2)*(r-r2)/(255-r2) + s2
    return s

def contrast_stretching(image,r1 = 96,s1 = 64,r2 = 160,s2 = 192):
    """
    Apply contrast stretching to an entire image.

    Parameters:
    - image (numpy.ndarray): Input grayscale image to be processed.
    - r1 (int): Lower limit of the input pixel intensity range (default: 96).
    - s1 (int): Lower limit of the output pixel intensity range (default: 64).
    - r2 (int): Upper limit of the input pixel intensity range (default: 160).
    - s2 (int): Upper limit of the output pixel intensity range (default: 192).

    Returns:
    - new_image (numpy.ndarray): Processed image after contrast stretching.
    """
    new = image.copy()
    (row,col) = image.shape
    for i in range(row):
        for j in range(col):
```

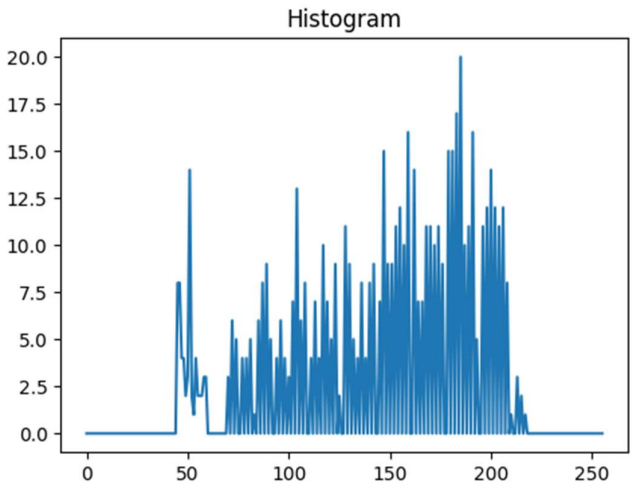
```
            new[i][j] = contrast_stretch_pixel(r1,r2,s1,s2,image[i][j])
    return new

low_contrast = cv2.imread("low-contrast-ex05.jpg",0)
old_hist = cv2.calcHist(low_contrast,channels=[0],mask=None,histSize=[256],ranges =
[0,256])
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

ax1.imshow(low_contrast, cmap='gray')
ax1.set_title('Low Contrast Image')

ax2.plot(old_hist)
ax2.set_title('Histogram')
plt.show()



new = contrast_stretching(image=low_contrast,r1=120,r2=200,s1=60,s2=230)
new_hist = cv2.calcHist(new,channels=[0],mask=None,histSize=[256],ranges = [0,256])
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))

ax1.imshow(new, cmap='gray')
ax1.set_title('High Contrast Image')

ax2.plot(new_hist)
ax2.set_title('Histogram')
plt.show()
```

Output:


Low Contrast Image
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600


Histogram
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
50
100
150
200
250


High Contrast Image
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600


Histogram
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
50
100
150
200
250