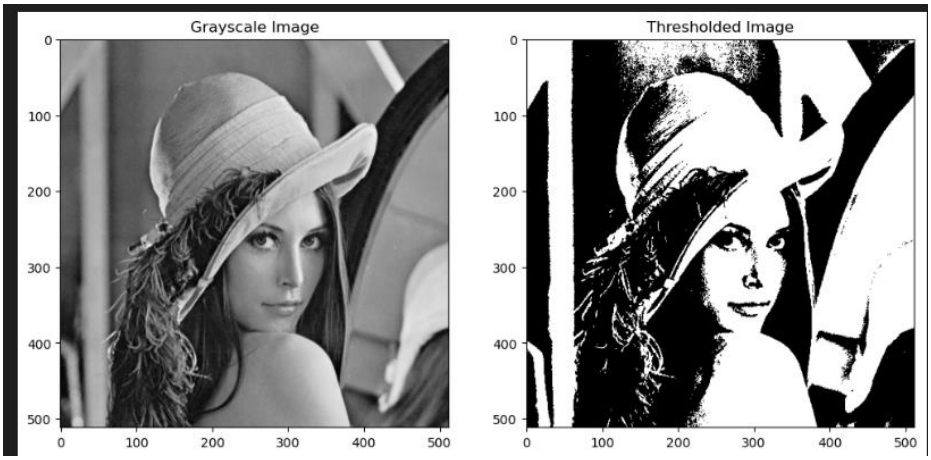
Assignment #1 Python Platform & Basic Image Processing

P68131509 林均有

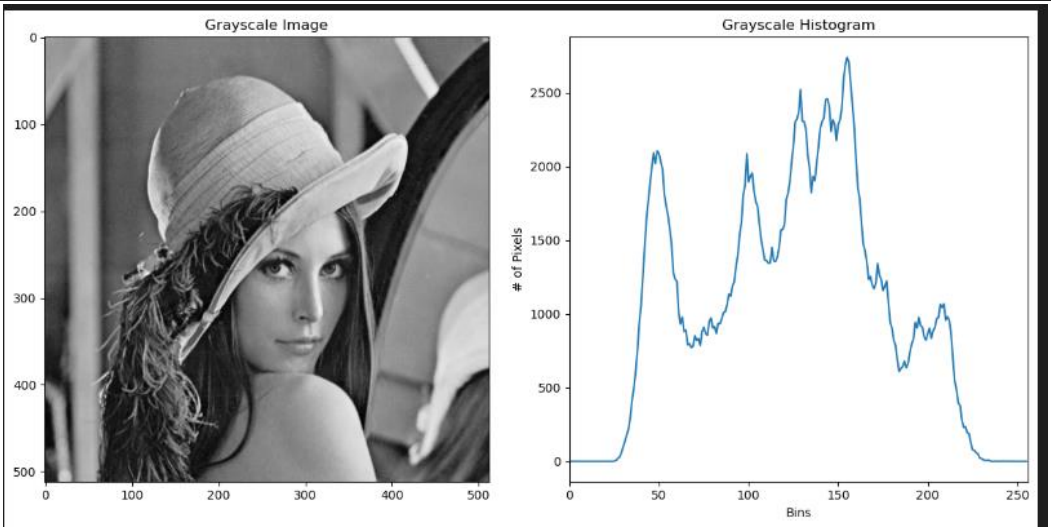
1. **Setup Python coding environment (Google Colab or your Laptop/PC).**
   I chose VSCode as my code editor, Anaconda to set up my Python coding environment, and Jupyter Notebook for interactive computing, which offers interactive visualizations and real-time feedback.
2. **Explain and report the codes in Part A and the codes before Part A**
   Before Part A:
   Imports: Bring in tools for image processing (cv2), displaying images (matplotlib), and managing files (os).
   os.chdir(): Sets the working folder to where my images are, making it easier to work with them in the script.
   PartA:
   `cv2.imread("Lenna.png")`: Reads the image file "Lenna.png" into the img variable. `plt.figure(figsize=(4,4))` set the figure size in inches.
   `plt.imshow(img)`: Displays the image. Since OpenCV uses BGR. `plt.title("Lenna orig")`: Sets the title of the plot. `plt.show()`: Displays the plot.
   `cv2.cvtColor(img, cv2.COLOR_BGR2RGB)`: Converts the image from BGR to RGB format so it displays correctly. `cv2.COLOR_BGR2RGB` is a color conversion code which can convert an image from BGR color space to RGB color space.
   `print(type(img))` and `print(img.shape)` shows the data type of current image and the dimensions of an image.
   `plt.subplots(1, 3, figsize=(12, 4))`: Creates a figure and a grid of subplots with 1 row and 3 columns. `img[:, :, 0]`: Extracts the Red channel from the image array. In OpenCV, color images are represented in BGR format. `cmap='gray'`: Displays the image in grayscale. This is used because each channel is a single grayscale image representing the intensity of that color channel. `ax_arr[0].set_title("Red-channel image")`: Sets the title for the first subplot, indicating that it shows the Red channel. `vmin=0 and vmax=255` specify the minimum and maximum values for the color map scale. This ensures that the entire range of pixel intensities (0 to 255) is used.
3. **Implement the functions of image thresholding and image histogram by using OpenCV in Part C with the aids of Germini or ChatGPT.**

```
img_gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)  # Convert to
grayscale for thresholding
_, img_thresh = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)
plt.figure(figsize=(12, 6))
```

```
plt.subplot(1, 2, 1)
plt.imshow(img_gray, cmap='gray')
plt.title("Grayscale Image")
plt.subplot(1, 2, 2)
plt.imshow(img_thresh, cmap='gray')
plt.title("Thresholded Image")
plt.show()
```




```
# image histogram
hist=cv2.calcHist([img_gray], [0], None, [256], [0, 256])
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(img_gray, cmap='gray')
plt.title("Grayscale Image")
plt.subplot(1, 2, 2)
plt.plot(hist)
plt.title("Grayscale Histogram")
plt.xlabel("Bins")
plt.ylabel("# of Pixels")
plt.xlim([0, 256])
plt.tight_layout()
plt.show()
```



Assignment #1. Source code_P68131509_林均有 

OpenCV (Open Source Computer Vision Library) The OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications. Officially launched in 1999. OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. [https://opencv.org/about/]

Free OpenCV course: https://opencv.org/university/free-opencv-course/?utm_source=opcv&utm_medium=menu&utm_campaign=obc

More OpenCV functions: https://github.com/BhanuPrakashNani/Image_Processing

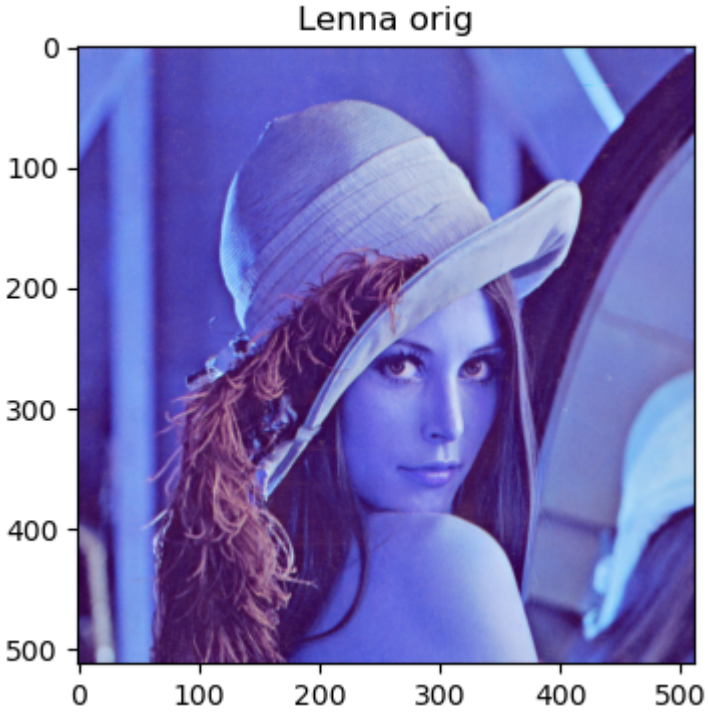
In [3]:
```
import cv2  # opencv for python package
import matplotlib.pyplot as plt
import os
```

In [4]:
```
#change the directory
os.chdir('C:/Users/user/ncku/Assignment 1 Python Platform and Basic Image Proces
```

PartA. Read and Plot an Image

1. Read and plot an image with OpenCV.

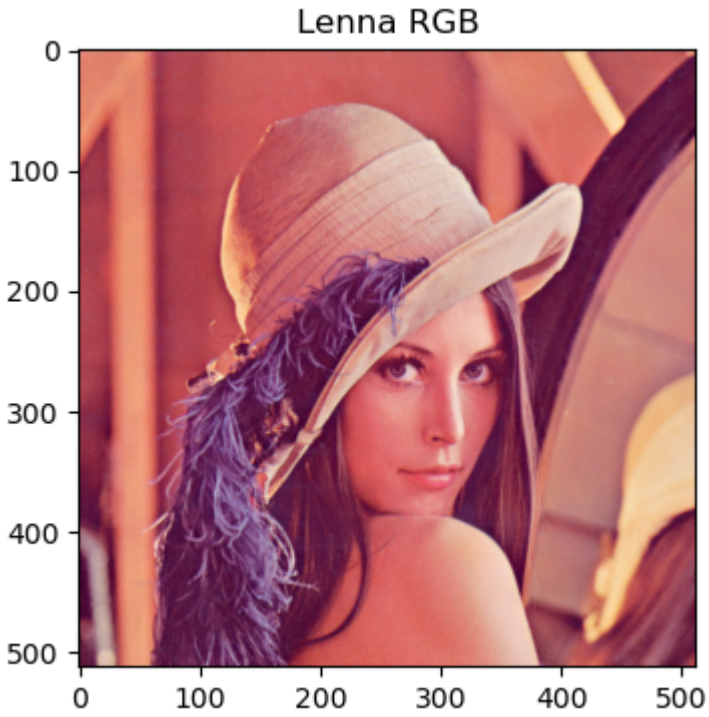
In [5]:
```
img=cv2.imread("Lenna.png")
plt.figure(figsize=(4,4))
plt.imshow(img)
plt.title("Lenna orig")
plt.show()
```



We got a weird image colors... This is because OpenCV uses image reading convention of BGR and matplotlib uses RGB.

The fix is easy:

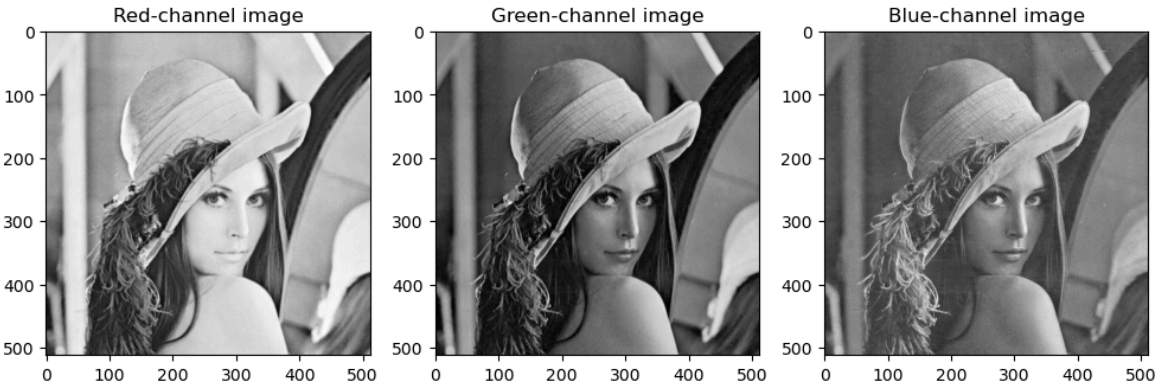
In [6]:
```
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.figure(figsize=(4,4))
plt.imshow(img)
plt.title("Lenna RGB")
plt.show()

# some image info:
print(type(img))
print(img.shape)
```




```
<class 'numpy.ndarray'>
(512, 512, 3)
```

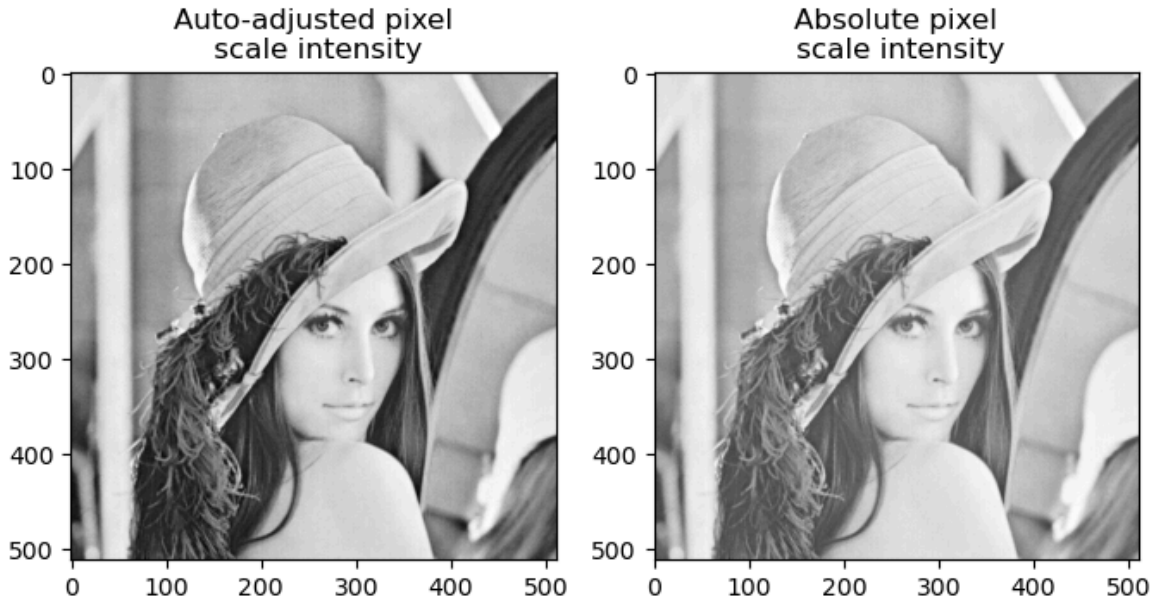
2. Show the channels of a color image. A color image contains R, G, B channels. Learn to show each channel.

In [7]:
```
# show the R, G, and B channels
fig,ax_arr=plt.subplots(1,3,figsize=(12,4))
ax_arr[0].imshow(img[:,:,0],cmap='gray')
ax_arr[0].set_title("Red-channel image")
ax_arr[1].imshow(img[:,:,1],cmap='gray')
ax_arr[1].set_title("Green-channel image")
ax_arr[2].imshow(img[:,:,2],cmap='gray')
ax_arr[2].set_title("Blue-channel image")
plt.show()
```



In [8]:
```
# gray color-mapping
fig,ax_arr=plt.subplots(1,2,figsize=(8,4))

#ax_arr[0].imshow(img[:,:,0], cmap="gray")
ax_arr[0].imshow(img[:,:,0], cmap="gray")
ax_arr[0].set_title("Auto-adjusted pixel\n scale intensity")

ax_arr[1].imshow(img[:,:,0], cmap="gray", vmin=0, vmax=255)
```

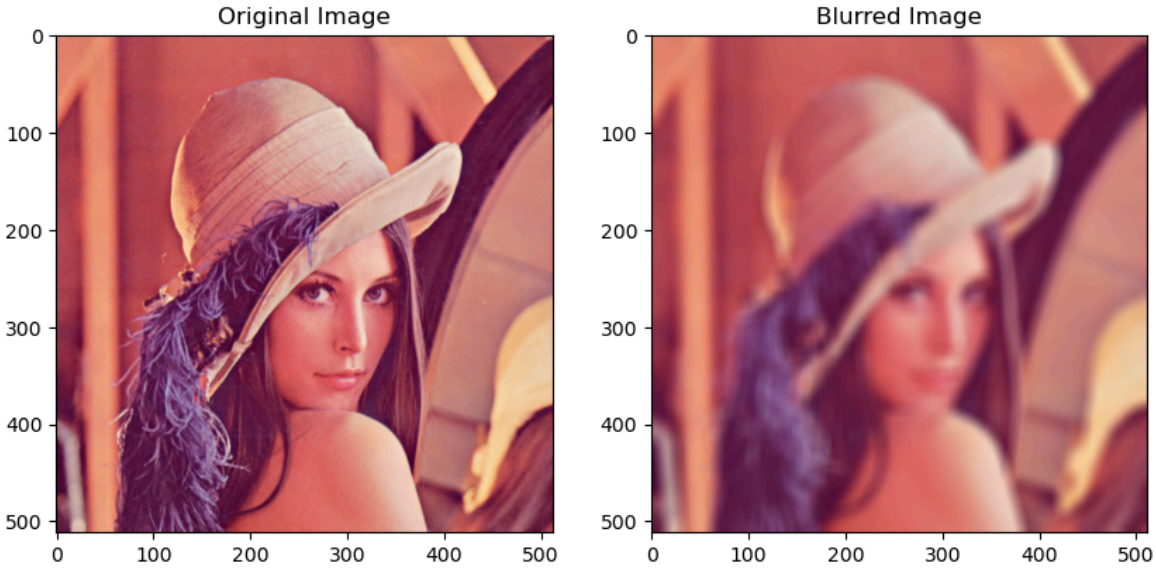
```
ax_arr[1].set_title("Absolute pixel\n scale intensity")
plt.show()
```



Part B. Advanced Image Processing Functions

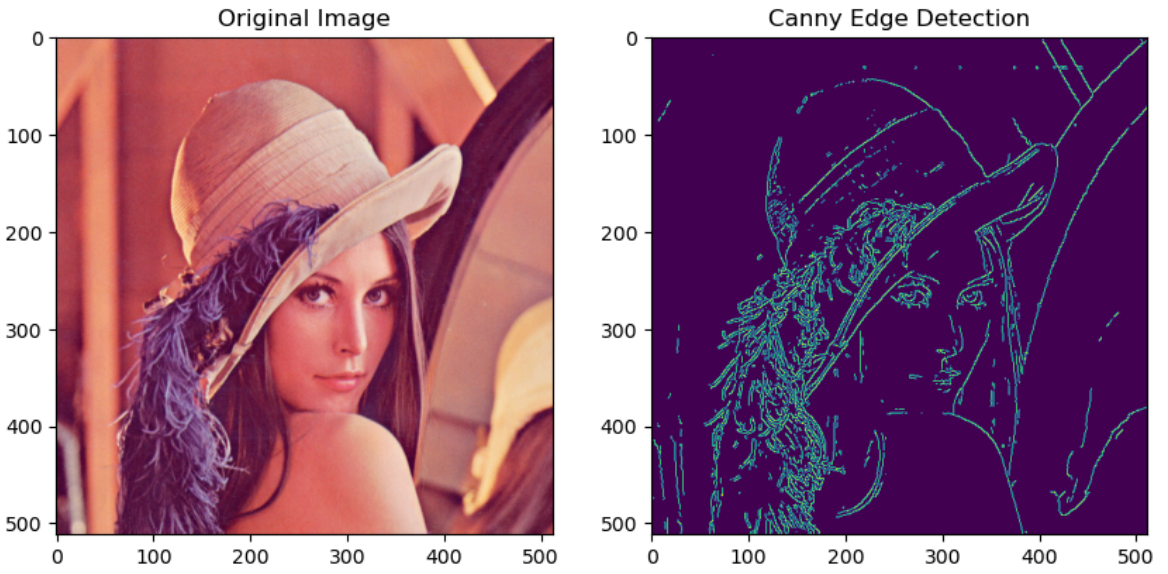
1. **Image blurring**. Apply an average filtering to the image.
2. **Edge detection.**. Apply Canny edge decteion to the image.
3. **Circle detection**. Apply Hough transform to the image.

In [9]:
```
# image blurring
img_blurred=cv2.blur(img,(15,15))  # use a 15x15 average kernel
fig,ax_arr=plt.subplots(1, 2, figsize=(10,10))
ax_arr[0].imshow(img)
ax_arr[0].set_title("Original Image")
ax_arr[1].imshow(img_blurred)
ax_arr[1].set_title("Blurred Image")
plt.show()
```

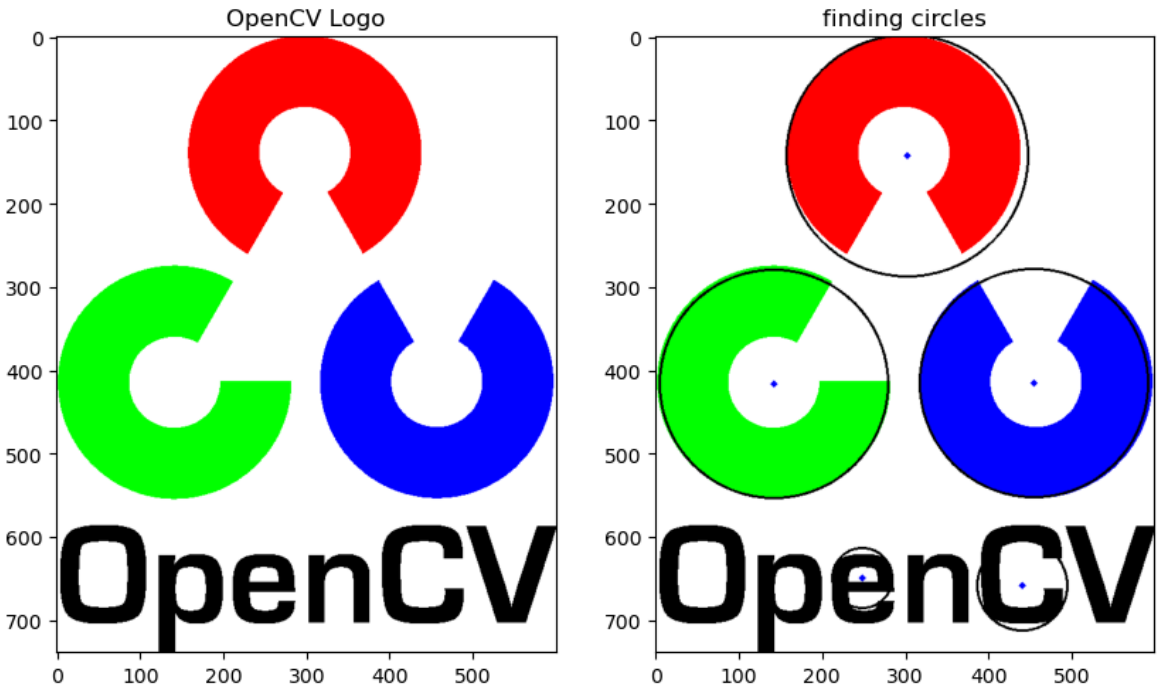


In [10]:
```
# edge detection
img_canny=cv2.Canny(img,180,200)  # end args are the lower & upper TH of hystere
```

```
fig,ax_arr=plt.subplots(1, 2, figsize=(10,10))
ax_arr[0].imshow(img)
ax_arr[0].set_title("Original Image")
ax_arr[1].imshow(img_canny)
ax_arr[1].set_title("Canny Edge Detection")
plt.show()
```



In [11]:
```
# detect circles
img_logo=cv2.imread("Opencv_logo.png")
img_logo=cv2.cvtColor(img_logo, cv2.COLOR_BGR2RGB)

fig,ax_arr=plt.subplots(1, 2, figsize=(10,10))
ax_arr[0].imshow(img_logo)
ax_arr[0].set_title("OpenCV Logo")

img_gray=cv2.cvtColor(img_logo,cv2.COLOR_RGB2GRAY)
circles=cv2.HoughCircles(img_gray, cv2.HOUGH_GRADIENT, 0.1, 50, param1=50, param
for x, y, r in circles[0, :]:
    # draw the outer circle
    cv2.circle(img_logo, (int(x), int(y)), int(r), (0, 0, 0), 2)
    # draw the center of the circle
    cv2.circle(img_logo, (int(x), int(y)), 2, (0, 0, 255), 3)

ax_arr[1].imshow(img_logo)
ax_arr[1].set_title("finding circles")
plt.show()
```

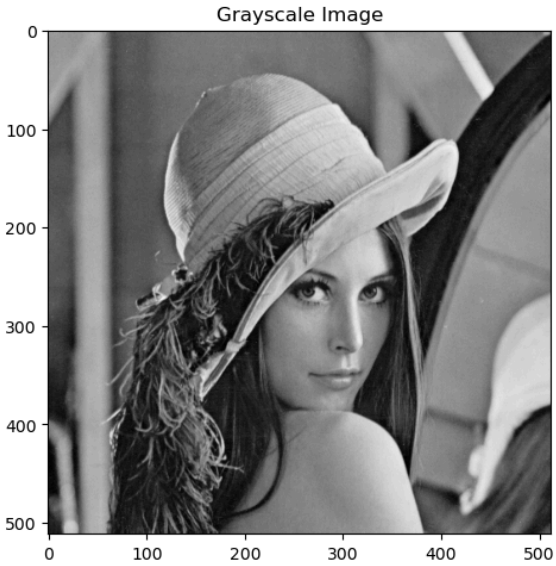


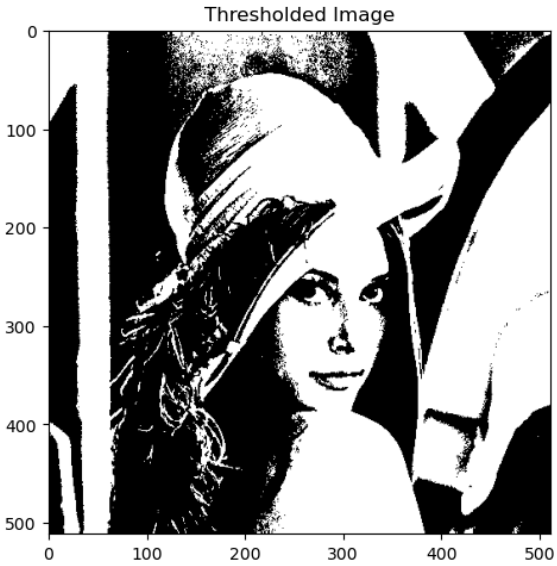
Part C. Basic Image Processing Implementation

Implement the following image processing functions using CV2 with the aids of Germini.

- Image thresholding
- Image histogram

In [12]:
```
# Image Thresholding
img_gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)  # Convert to grayscale for thr
_, img_thresh = cv2.threshold(img_gray, 127, 255, cv2.THRESH_BINARY)

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(img_gray, cmap='gray')
plt.title("Grayscale Image")

plt.subplot(1, 2, 2)
plt.imshow(img_thresh, cmap='gray')
plt.title("Thresholded Image")
plt.show()
```





In [13]:
```
# image histogram
hist=cv2.calcHist([img_gray], [0], None, [256], [0, 256])
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.imshow(img_gray, cmap='gray')
plt.title("Grayscale Image")

plt.subplot(1, 2, 2)
plt.plot(hist)
plt.title("Grayscale Histogram")
plt.xlabel("Bins")
plt.ylabel("# of Pixels")
plt.xlim([0, 256])
plt.tight_layout()
plt.show()
```