1) Load image in python, read and write images

In [1]:
```
!pip install opencv-python
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: opencv-python in c:\users\ranjeet gupta\appdata\ro
aming\python\python312\site-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in c:\programdata\anaconda3\lib\site
-packages (from opencv-python) (1.26.4)
```

In [ ]:
```
import cv2
# Load an image
image = cv2.imread(r"C:\Users\Ranjeet Gupta\Downloads\test_image.jpg")

resized_image = cv2.resize(image, (1200, 900))

# Display the image in a window
cv2.imshow("Resized Image", resized_image)

# Convert the image to grayscale
gray_image = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)

# Display the grayscale image
cv2.imshow("Grayscale Image", gray_image)

cv2.waitKey(0)
cv2.destroyAllWindows()

# Save the resized image
cv2.imwrite("resized_image.jpg", resized_image)
# Save the grayscale image
cv2.imwrite("grayscale_image.jpg", gray_image)
```

2) Access pixel values and modify them

In [4]:
```
import cv2
img = cv2.imread(r"C:\Users\Ranjeet Gupta\Downloads\test_image.jpg")

# Access a pixel at position (x=100, y=50)
pixel = img[50, 100]

# Pixel will return an array with BGR values
print(f"Pixel at (100, 50) position: {pixel}")

# Access the blue, green, and red components separately
blue = img[50, 100, 0]
green = img[50, 100, 1]
red = img[50, 100, 2]

print(f"Blue: {blue}, Green: {green}, Red: {red}")

# Modify the pixel at position (x=100, y=50) to white (255, 255, 255)
img[50, 100] = [255, 255, 255]
```

```
# Save the modified image
cv2.imwrite("modified_image.jpg", cv2.resize(img, (800, 600)))
```

```
Pixel at (100, 50) position: [238 154  82]
Blue: 238, Green: 154, Red: 82
```

Out[4]: True

3) Access image properties

In [8]:
```
# Get image properties
# Image shape (height, width, number of channels)
height, width, channels = img.shape
print(f"Width: {width}, Height: {height}, Channels: {channels}")

# Total number of pixels (height * width * channels)
total_pixels = img.size
print("Total Pixels: ", total_pixels)

# Image data type
img_dtype = img.dtype
print("Image Data Type: ", img_dtype)

# Image dimensions (2 for grayscale, 3 for color)
dimensions = img.ndim
print("Image Dimensions: ", dimensions)
```

```
Width: 6000, Height: 4000, Channels: 3
Total Pixels:  72000000
Image Data Type:  uint8
Image Dimensions:  3
```

4) Setting Region of Image (ROI)

In [11]:
```
# Define ROI
x1, y1 = 5000, 1000
x2, y2 = 8000, 6000

# Extract the ROI
roi = img[y1:y2, x1:x2]

# Display the ROI
cv2.imshow("ROI", roi)
cv2.waitKey(0)
cv2.destroyAllWindows()

# save the ROI
cv2.imwrite("roi.jpg", roi)
```

Out[11]: True

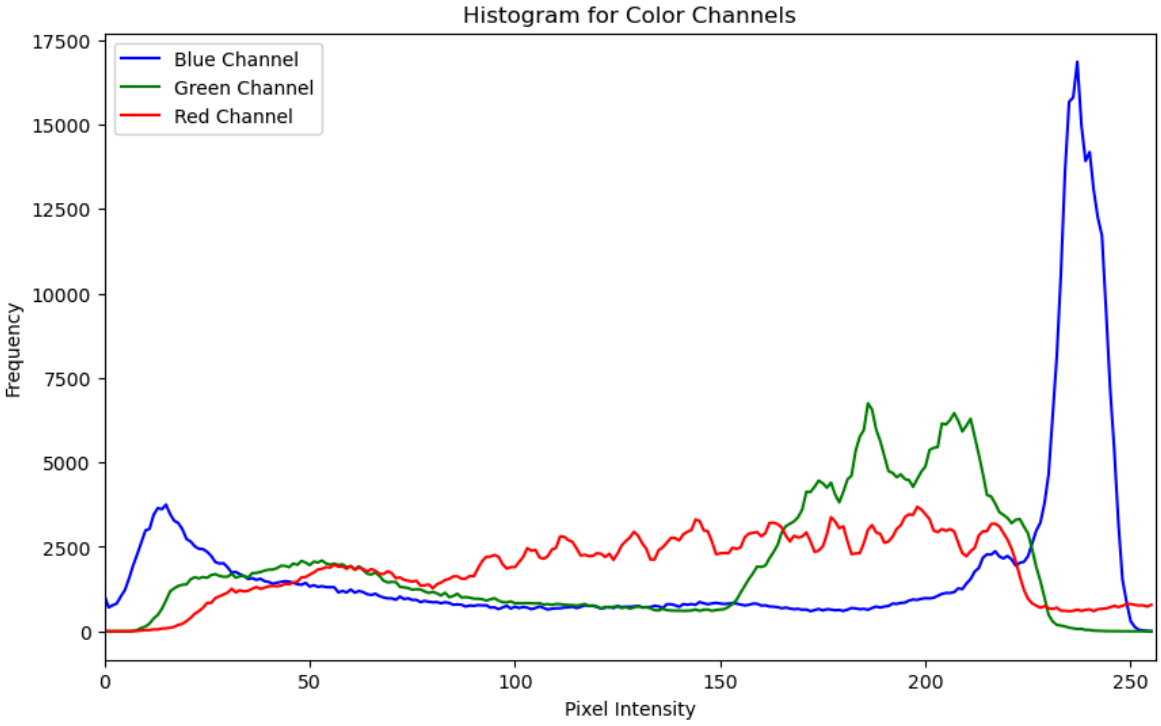
5) Splitting and Merging images

In [1]:
```python
import cv2
img = cv2.imread(r"C:\Users\Ranjeet Gupta\Downloads\test_image.jpg")
r_img = cv2.resize(img, (800, 600))

# Split the image into its color channels (BGR)
blue_channel, green_channel, red_channel = cv2.split(r_img)

# Display each channel
cv2.imshow("Blue Channel", blue_channel)
cv2.imshow("Green Channel", green_channel)
cv2.imshow("Red Channel", red_channel)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Merge color channels back into one image
merged_image = cv2.merge([blue_channel, green_channel, red_channel])

# Display the merged image
cv2.imshow("Merged Image", merged_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

6) Computing the histogram of an image and plot the histogram of each color channel

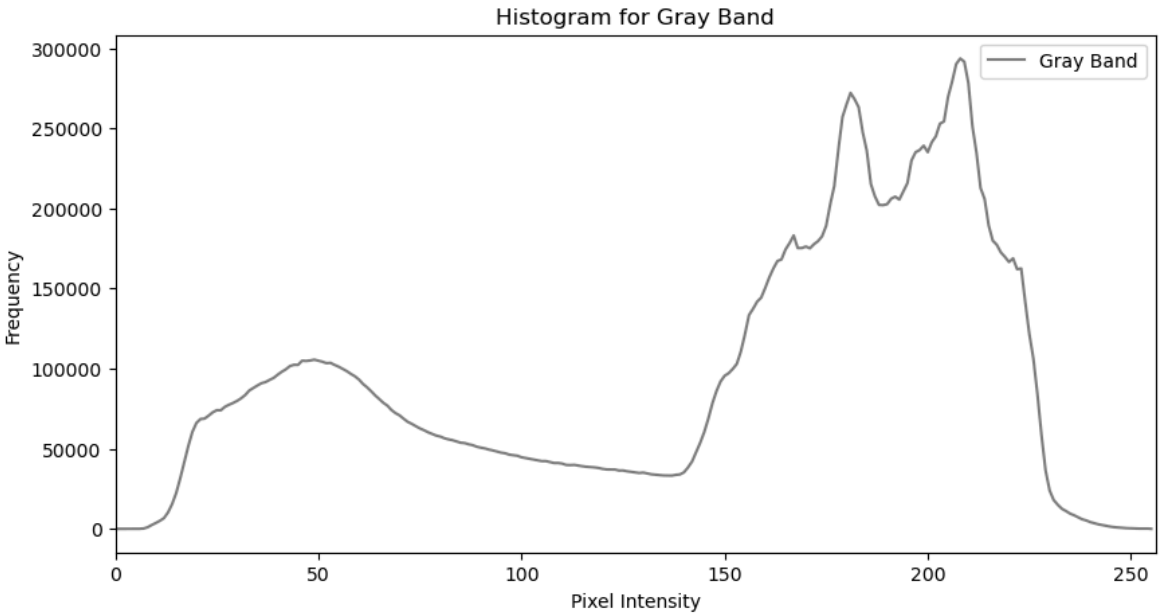
In [4]:
```python
import matplotlib.pyplot as plt

# Compute the histogram for each channel
hist_blue = cv2.calcHist([blue_channel], [0], None, [256], [0, 256])
hist_green = cv2.calcHist([green_channel], [0], None, [256], [0, 256])
hist_red = cv2.calcHist([red_channel], [0], None, [256], [0, 256])

# Plot the histograms
plt.figure(figsize=(10, 6))

# Plot blue channel histogram
plt.plot(hist_blue, color='blue', label='Blue Channel')
plt.xlim([0, 256])

# Plot green channel histogram
plt.plot(hist_green, color='green', label='Green Channel')
plt.xlim([0, 256])

# Plot red channel histogram
plt.plot(hist_red, color='red', label='Red Channel')
plt.xlim([0, 256])

# Add labels and title
plt.title('Histogram for Color Channels')
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.legend()
```

```
# Show the plot
plt.show()
```



7) Convert the image to grayscale and plot its histogram.

In [9]:
```
import cv2
img = cv2.imread(r"C:\Users\Ranjeet Gupta\Downloads\test_image.jpg")

# Convert the image to grayscale
grayscale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

hist_gray = cv2.calcHist([grayscale], [0], None, [256], [0,256]) #compute histog

import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))
plt.plot(hist_gray, color='gray', label='Gray Band')
plt.xlim([0, 256])

plt.title('Histogram for Gray Band')
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.legend()

plt.show()
```



8) Zoom in and Zoom out operation

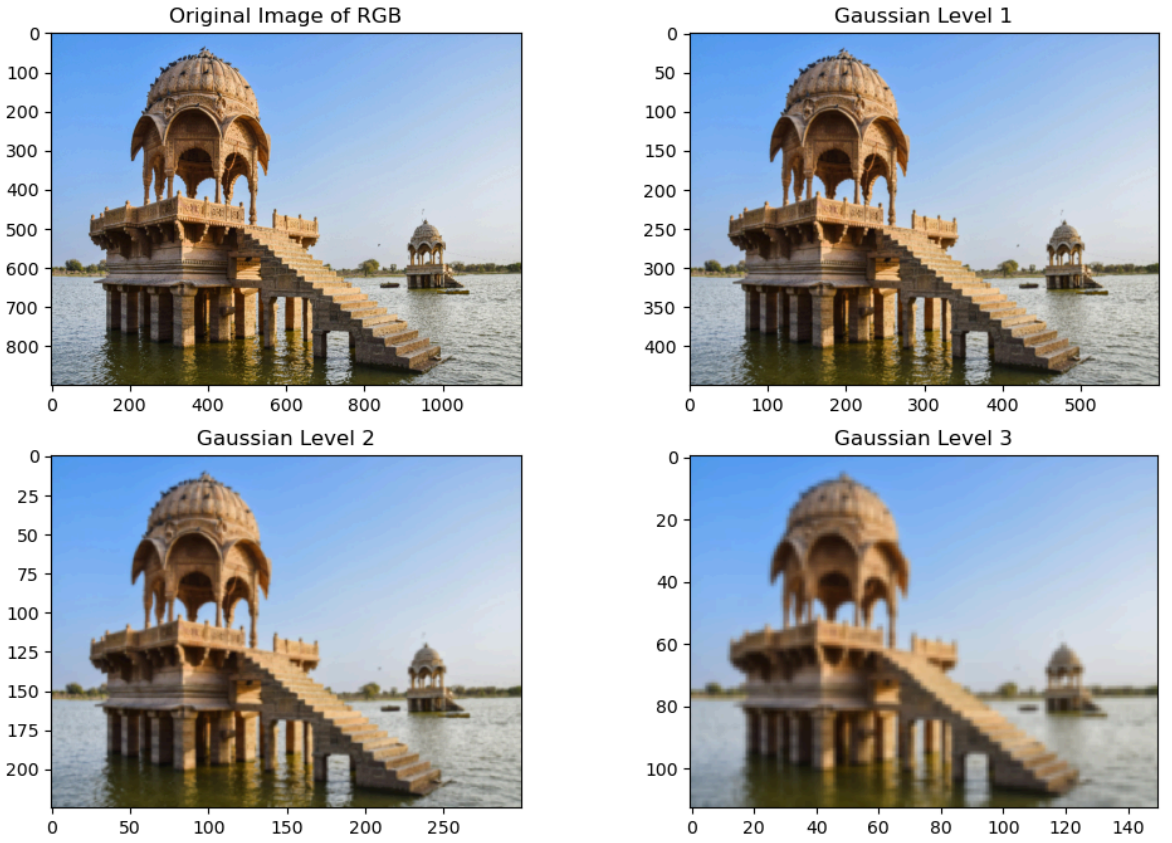
In [11]:
```
# Zoom In (Scale up the image by 2x)
zoom_in = cv2.resize(img, None, fx=2, fy=2, interpolation=cv2.INTER_LINEAR)
# dsize: The desired output size of the image (width, height). If None is provid

# Zoom Out (Scale down the image by 0.5x)
zoom_out = cv2.resize(img, None, fx=0.5, fy=0.5, interpolation=cv2.INTER_LINEAR)

# Display the results
cv2.imshow("Original Image", img)
cv2.imshow("Zoom In", zoom_in)
cv2.imshow("Zoom Out", zoom_out)

cv2.waitKey(0)
cv2.destroyAllWindows()

# Save the zoomed images
cv2.imwrite("zoom_in.jpg", zoom_in)
cv2.imwrite("zoom_out.jpg", zoom_out)
```

Out[11]: True

9) Building pyramids

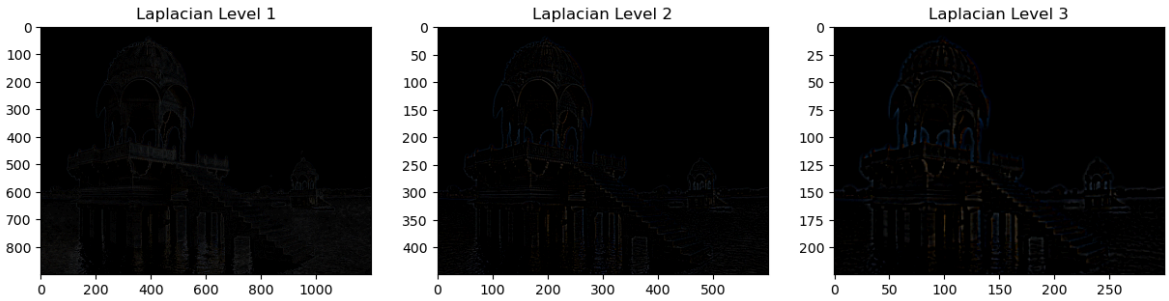
In [19]:
```
#Building Gaussian pyramid: Reduces the image resolution by repeatedly applying

resized_img = cv2.resize(img, (1200, 900))
img_rgb = cv2.cvtColor(resized_img, cv2.COLOR_BGR2RGB)

# Build Gaussian pyramid: Downsample multiple times
G1 = cv2.pyrDown(img_rgb)  # First layer downsampled
G2 = cv2.pyrDown(G1)       # Second layer downsampled
G3 = cv2.pyrDown(G2)       # Third layer downsampled

# Display the pyramid
```

```
cv2.imshow("Original Image of RGB", img_rgb)
cv2.imshow("Gaussian Level 1", G1)
cv2.imshow("Gaussian Level 2", G2)
cv2.imshow("Gaussian Level 3", G3)

cv2.waitKey(0)
cv2.destroyAllWindows()

import matplotlib.pyplot as plt
# Display Gaussian Pyramid Levels
plt.figure(figsize=(12, 8))

plt.subplot(2, 2, 1)
plt.imshow(img_rgb)
plt.title("Original Image of RGB")

plt.subplot(2, 2, 2)
plt.imshow(G1)
plt.title("Gaussian Level 1")

plt.subplot(2, 2, 3)
plt.imshow(G2)
plt.title("Gaussian Level 2")

plt.subplot(2, 2, 4)
plt.imshow(G3)
plt.title("Gaussian Level 3")

plt.show()
```



In [28]:
```
# Build Laplacian Pyramid: Calculate the difference between Gaussian levels
L1 = cv2.subtract(img_rgb, cv2.pyrUp(G1, dstsize=(img_rgb.shape[1], img_rgb.shap
L2 = cv2.subtract(G1, cv2.pyrUp(G2, dstsize=(G1.shape[1], G1.shape[0])))     # L
L3 = cv2.subtract(G2, cv2.pyrUp(G3, dstsize=(G2.shape[1], G2.shape[0])))     # L
```

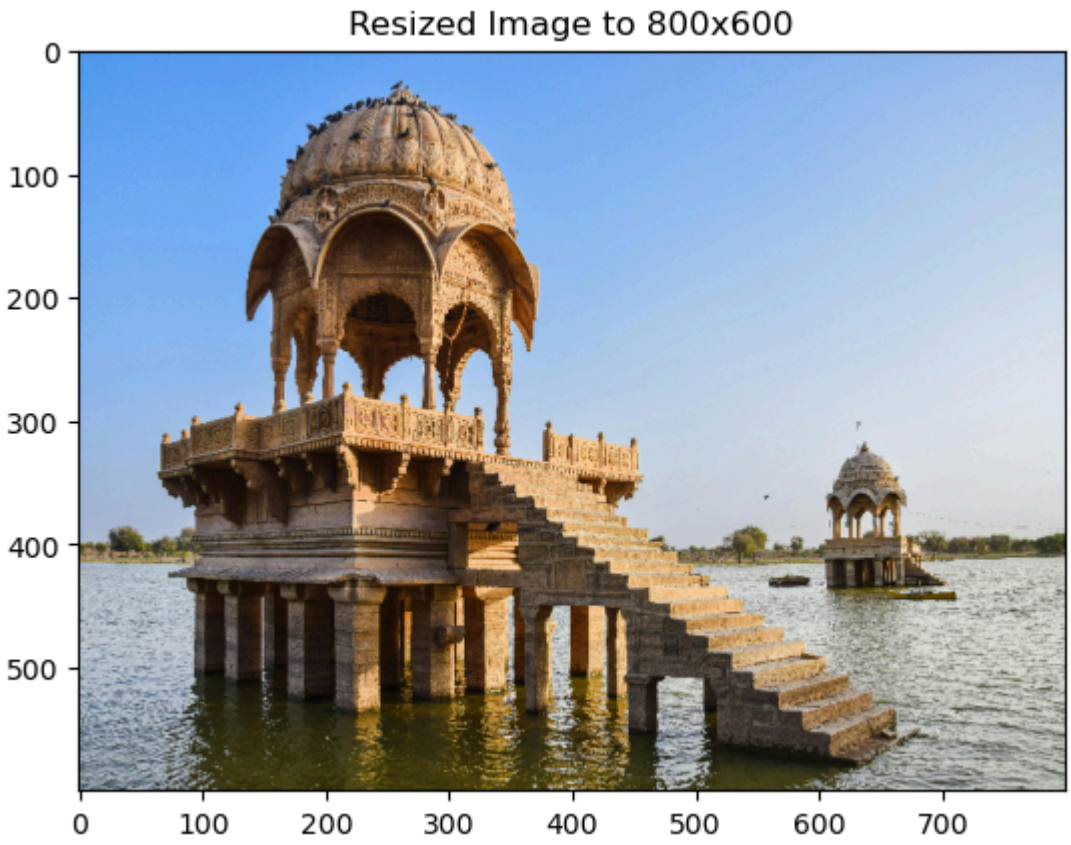
```
# Display Laplacian Pyramid Levels
plt.figure(figsize=(15, 8))

plt.subplot(1, 3, 1)
plt.imshow(L1)
plt.title("Laplacian Level 1")

plt.subplot(1, 3, 2)
plt.imshow(L2)
plt.title("Laplacian Level 2")

plt.subplot(1, 3, 3)
plt.imshow(L3)
plt.title("Laplacian Level 3")

plt.show(),
```



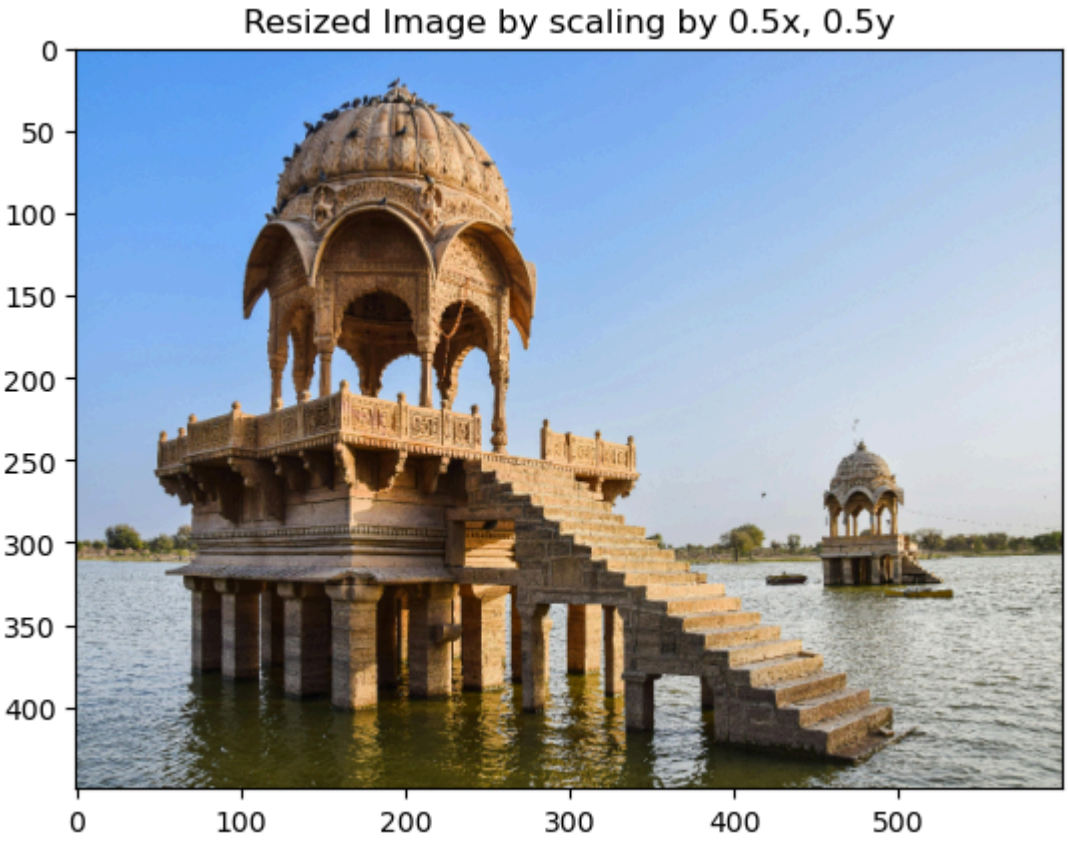
Out[28]: (None,)

10) Resize image

In [35]:
```
# Resize by specifying dimensions
new_width = 800
new_height = 600
resize_img = cv2.resize(img, (new_width, new_height))
resize_rgb = cv2.cvtColor(resize_img, cv2.COLOR_BGR2RGB)

plt.imshow(resize_rgb)
plt.title(f"Resized Image to {new_width}x{new_height}")
plt.show()
```



In [34]:
```python
# Resize by scaling factors
scale_x = 0.5
scale_y = 0.5
resize_img = cv2.resize(img_rgb, None, fx= scale_x, fy= scale_y, interpolation=

plt.imshow(resize_img)
plt.title(f"Resized Image by scaling by {scale_x}x, {scale_y}y")
plt.show()
```


Resized Image by scaling by 0.5x, 0.5y
0
50
100
150
200
250
300
350
400
0
100
200
300
400
500

In [ ]: