Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Practical based on Image Processing with Numpy	
Experiment No: 11	**Date:**	**Enrollment No:92301733025**

Aim: Practical based on Image Processing with Numpy

IDE:

NumPy for Image Processing

NumPy is a robust tool for image processing in Python.

Importing Libraries

The required libraries: PIL, NumPy, and Matplotlib. PIL is used for opening images. NumPy allows for efficient array operations and image processing. Matplotlib is used for visualizing images

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
```

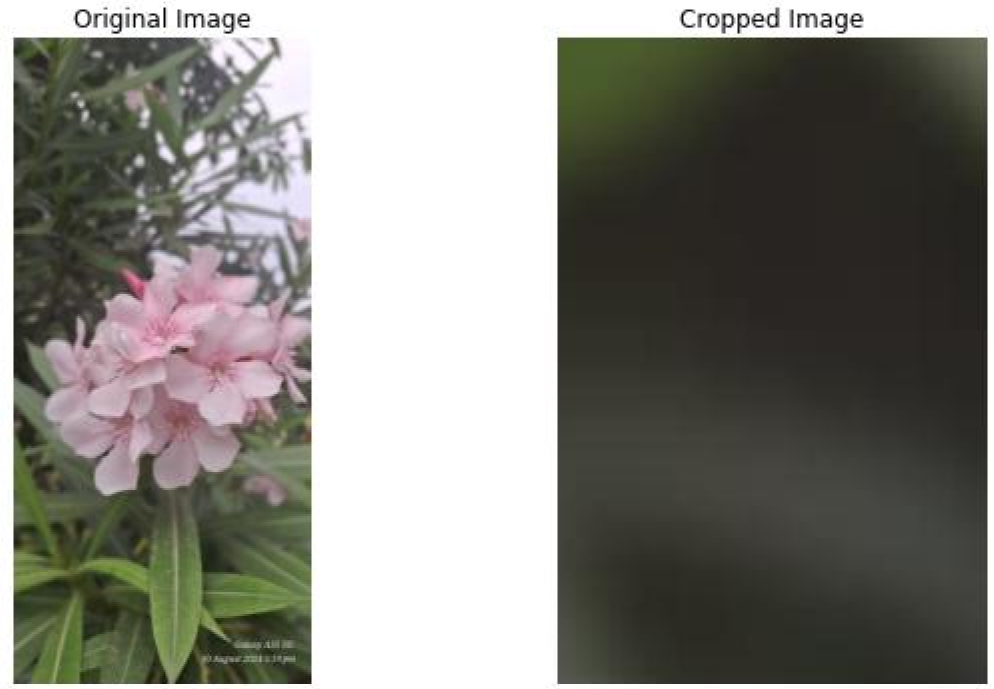
Crop Image

We define coordinates to mark the area we want to crop from the image. The new image contains only the selected part and discards the rest.

Example:

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
img_array = np.array(img)
print(img_array)
y1, x1 = 100, 100  # Top-left corner of ROI
y2, x2 = 250, 200  # Bottom-right corner of ROI
cropped_img = img_array[y1:y2, x1:x2]
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img_array)
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(cropped_img)
plt.title('Cropped Image')
plt.axis('off')
plt.tight_layout()
plt.show()
```

Output



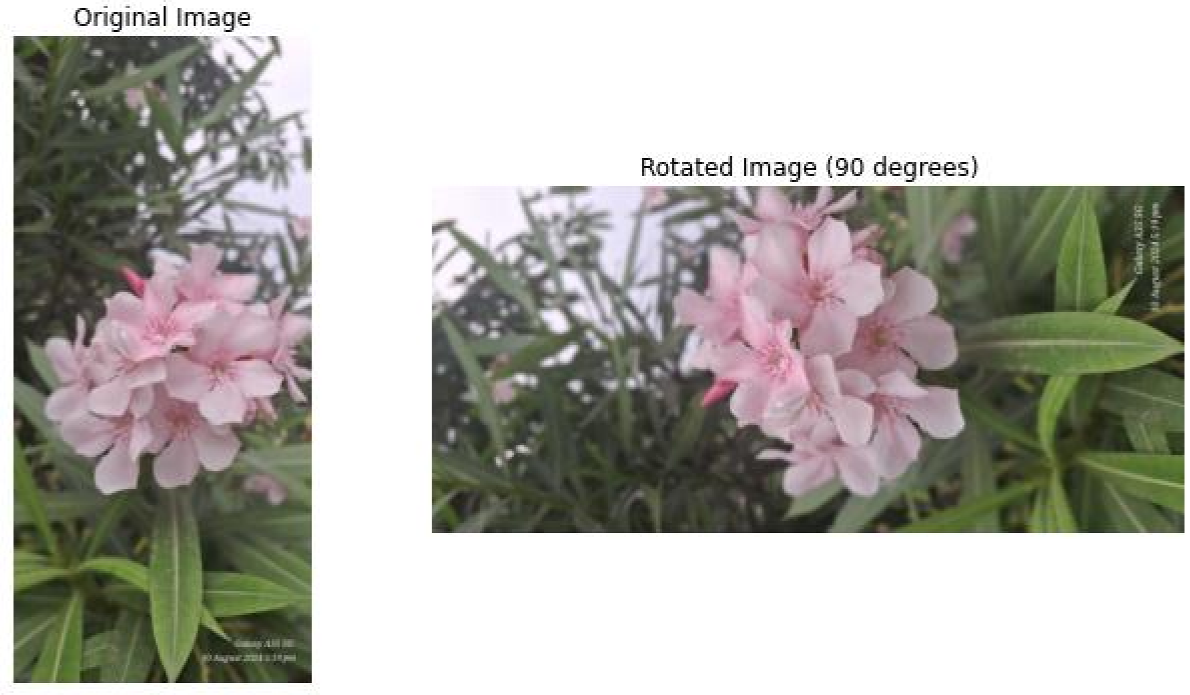
Rotate Image

We rotate the image array 90 degrees counterclockwise using NumPy's 'rot90' function.

Example:

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
img_array = np.array(img)
rotated_img = np.rot90(img_array)
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img_array)
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(rotated_img )
plt.title('Rotated Image (90 degrees)')
plt.axis('off')
plt.tight_layout()
plt.show()
```

Output



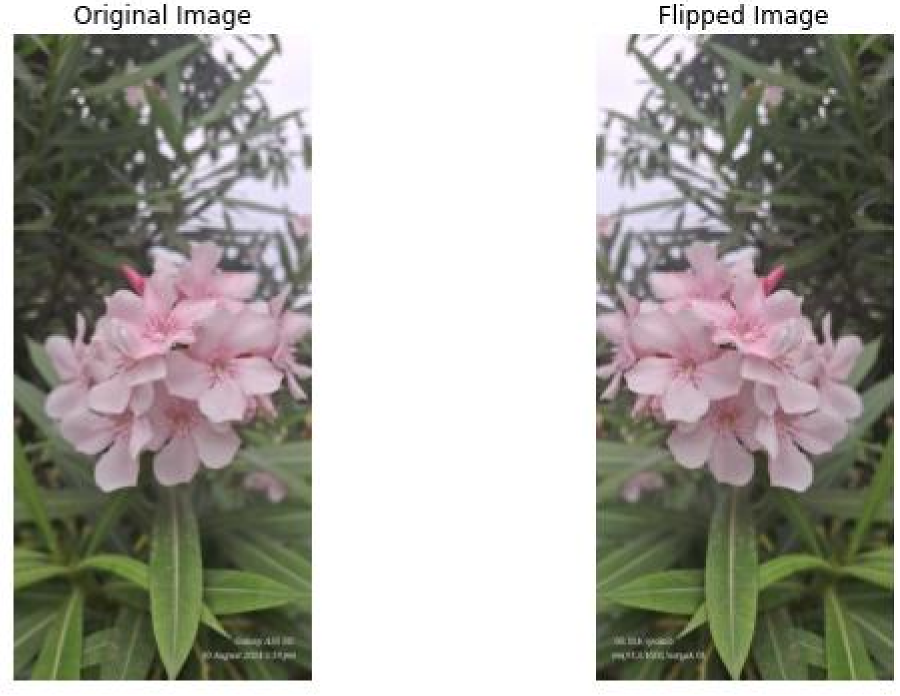
Flip Image

We use NumPy's 'fliplr' function to flip the image array horizontally.

Example:

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
img_array = np.array(img)
flipped_img = np.fliplr(img_array)
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img_array)
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(flipped_img )
plt.title('Flipped Image')
plt.axis('off')
plt.tight_layout()
plt.show()
```

Output



Negative of an Image

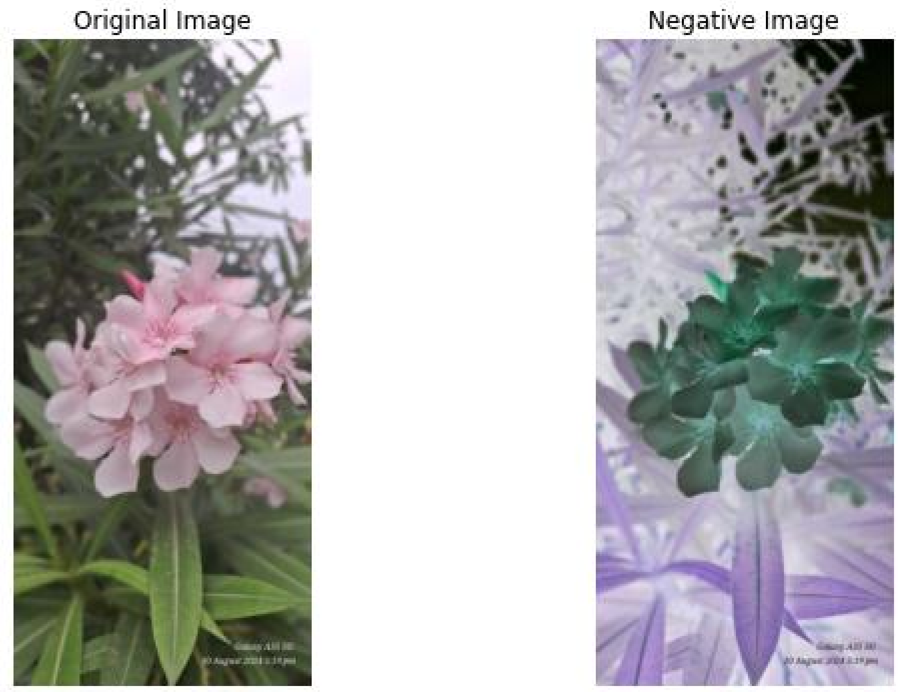
The negative of an image is made by reversing its pixel values. In grayscale images, each pixel's value is subtracted from the maximum (255 for 8-bit images). In color images, this is done separately for each color channel.

Example:

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
```

```
img_array = np.array(img)
is_grayscale = len(img_array.shape) < 3
# Function to create negative of an image
def create_negative(image):
    if is_grayscale:
        # For grayscale images
        negative_image = 255 - image
    else:
        # For color images (RGB)
        negative_image = 255 - image
    return negative_image
# Create negative of the image
negative_img = create_negative(img_array)
# Display the original and negative images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img_array)
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(negative_img)
plt.title('Negative Image')
plt.axis('off')
plt.tight_layout()
plt.show()
```

Output



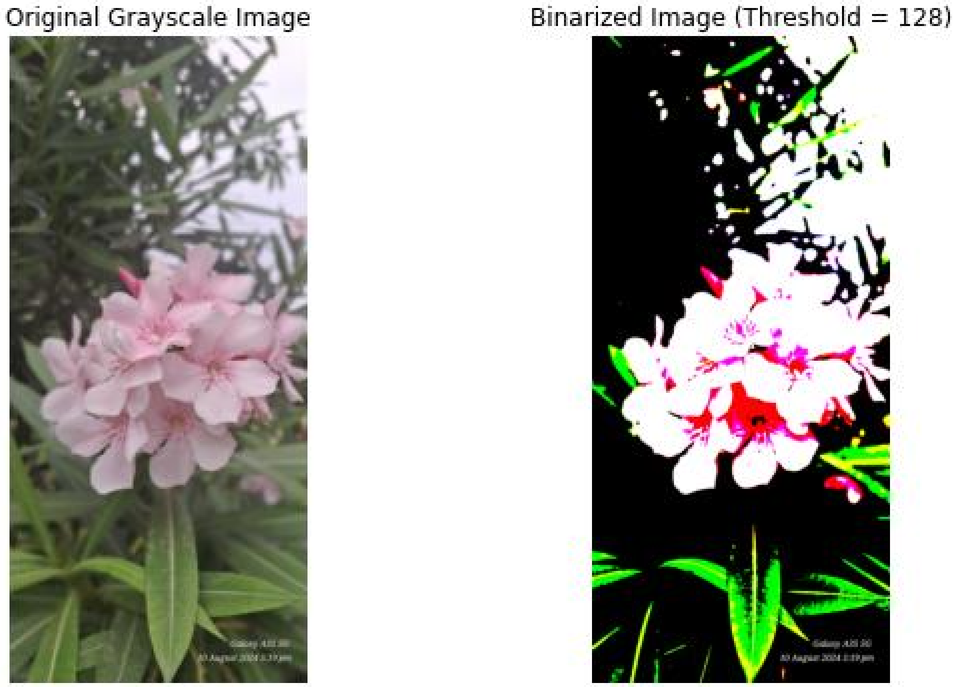
Binarize Image

Binarizing an image converts it to black and white. Each pixel is marked black or white based on a threshold value. Pixels that are less than the threshold become 0 (black) and above those above it become 255 (white).

Example

```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
img_array = np.array(img)
# Binarize the image using a threshold
threshold = 128
binary_img = np.where(img_array < threshold, 0, 255).astype(np.uint8)
# Display the original and binarized images
plt.figure(figsize= (10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img_array, cmap='gray')
plt.title('Original Grayscale Image')
plt.axis('off')
plt.subplot(1, 2, 2)
plt.imshow(binary_img, cmap='gray')
plt.title('Binarized Image (Threshold = 128)')
plt.axis('off')
plt.tight_layout()
plt.show()
```

Output



Color Space Conversion

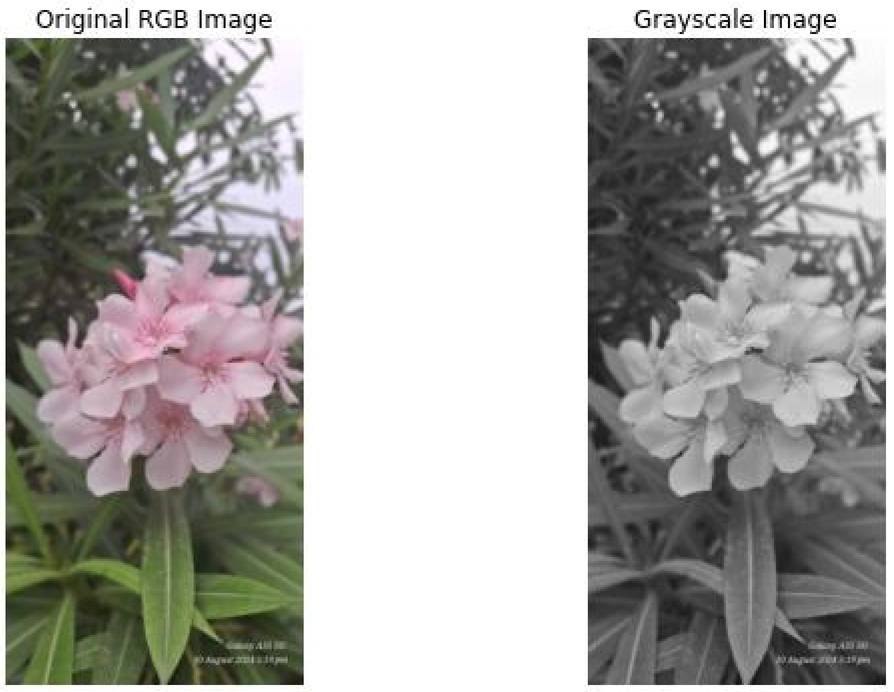
Color space conversion changes an image from one color model to another. This is done by changing the array of pixel values. We use a weighted sum of the RGB channels to convert a color image to a grayscale.

Example

```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
img_array = np.array(img)
# Grayscale conversion formula: Y = 0.299*R + 0.587*G + 0.114*B
gray_img = np.dot (img_array[..., :3], [0.299, 0.587, 0.114])
# Display the original RGB image
```

```
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img_array)
plt.title('Original RGB Image')
plt.axis('off')
# Display the converted grayscale image
plt.subplot(1, 2, 2)
plt.imshow(gray_img, cmap='gray')
plt.title('Grayscale Image')
plt.axis('off')
plt.tight_layout()
plt.show()
```

Output



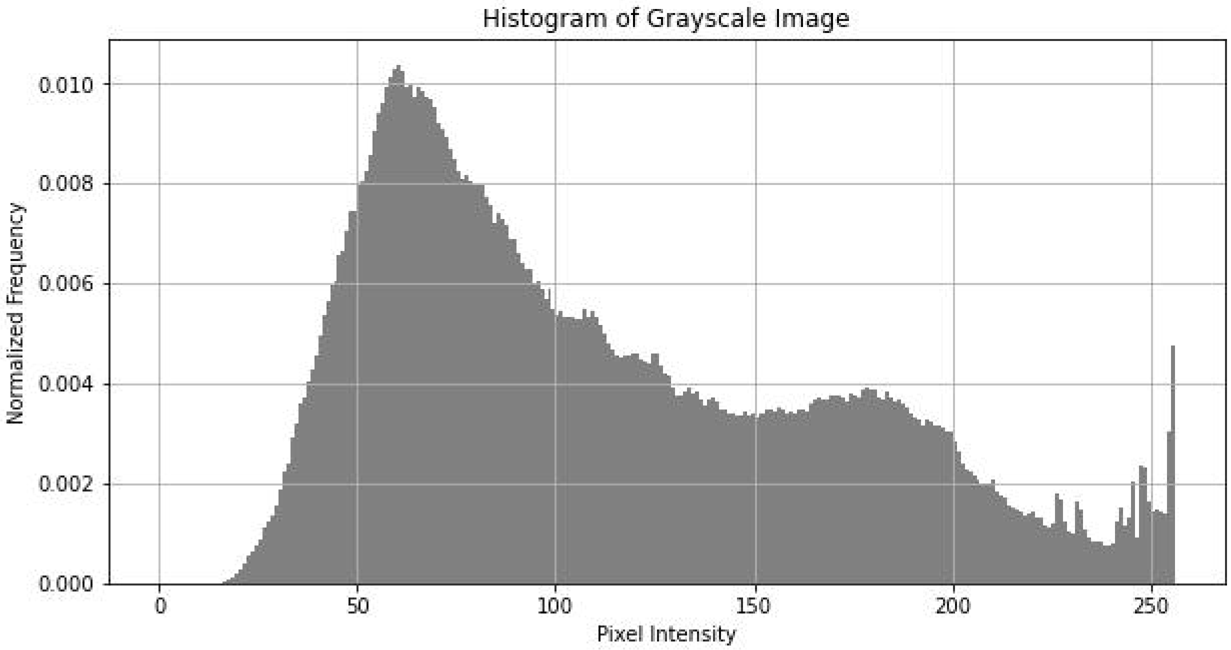
Pixel Intensity Histogram

The histogram shows the distribution of pixel values in an image. The image is flattened into a one-dimensional array to compute the histogram.

Example:

```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
img_array = np.array(img)
# Compute the histogram of the image
hist, bins = np.histogram(img_array.flatten(), bins=256, range= (0, 256))
# Plot the histogram
plt.figure(figsize=(10, 5))
plt.hist(img_array.flatten(), bins=256, range= (0, 256), density=True, color='gray')
plt.xlabel('Pixel Intensity')
plt.ylabel('Normalized Frequency')
plt.title('Histogram of Grayscale Image')
plt.grid(True)
plt.show()
```

Output



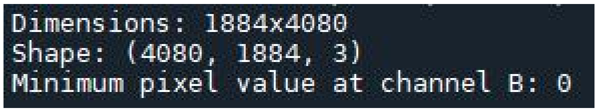
Post Lab Exercise:

a. Write a Python program to display details of an image (dimension of an image, shape of an image, min pixel value at channel B).

Code :

```
import numpy as np
from PIL import Image
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
img_array = np.array(img)
height, width, channels = img_array.shape
print(f"Dimensions: {width}x{height}")
print(f"Shape: {img_array.shape}"
minchannel_b = img_array[:, :, 2].min()
print(f"Minimum pixel value at channel B: {minchannel_b}")
```

Output :



```
Dimensions: 1884x4080
Shape: (4080, 1884, 3)
Minimum pixel value at channel B: 0
```

b. Write a Python program to padding black spaces

Code :

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
img_array = np.array(img)
padding_size = 50
padded_img_array = np.pad(img_array, ((padding_size, padding_size), (padding_size, padding_size), (0,
0)), mode='constant')
padded_img = Image.fromarray(padded_img_array)
padded_img.show()
```

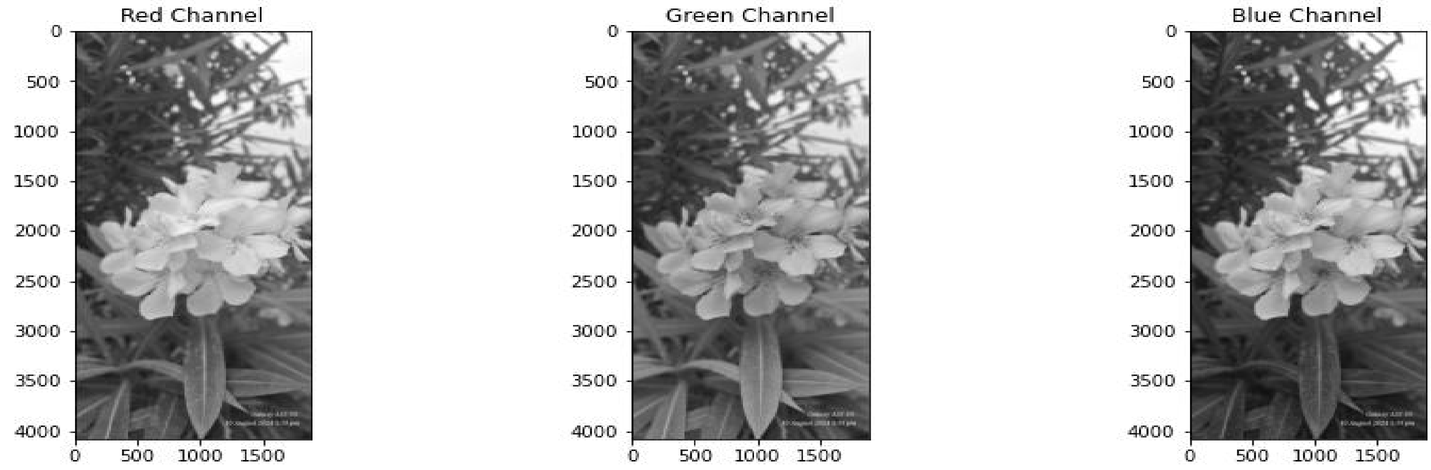
Output :

c. Write a Python program to visualize RGB channels

Code :

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
img = Image.open(r'C:\Users\student\Downloads\image.jpeg')
img_array = np.array(img)
r, g, b = img_array[:, :, 0], img_array[:, :, 1], img_array[:, :, 2]
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(r, cmap='gray')
axs[0].set_title('Red Channel')
axs[1].imshow(g, cmap='gray')
axs[1].set_title('Green Channel')
axs[2].imshow(b, cmap='gray')
axs[2].set_title('Blue Channel')
plt.show()
```

Output :



Red Channel
Green Channel
Blue Channel
0
500
1000
1500
2000
2500
3000
3500
4000
0
500
1000
1500