**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\Mitesh\OneDrive\Desktop\images.jpg')

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)

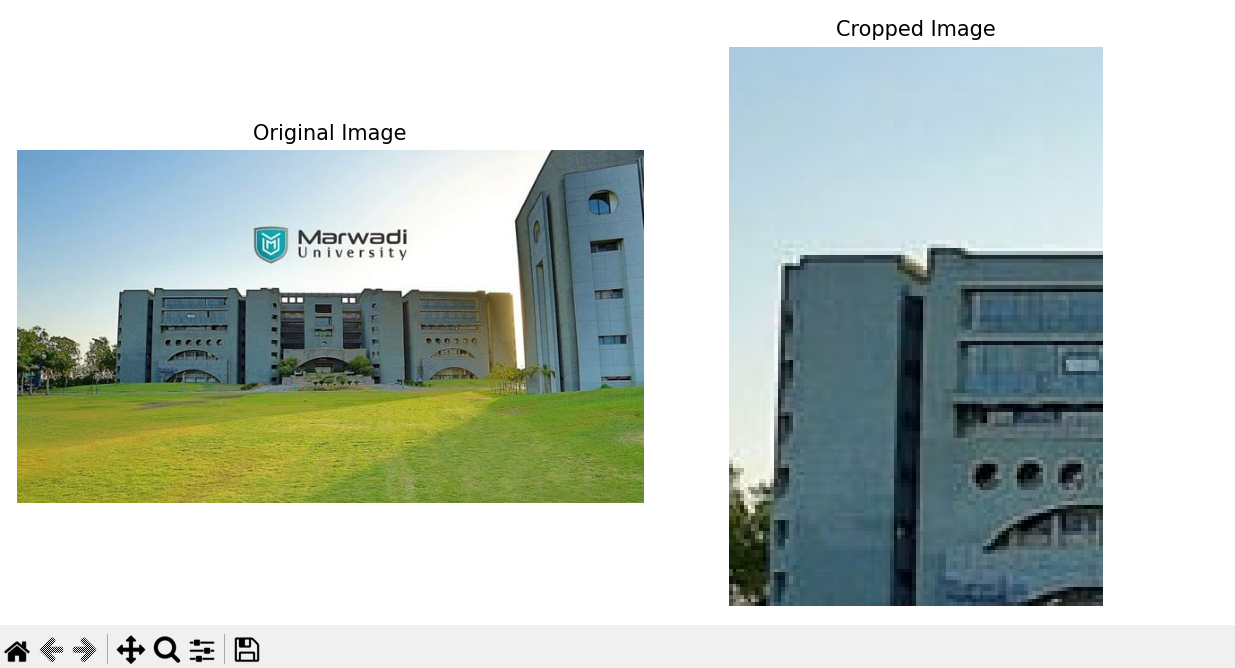
plt.title('Cropped Image')

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\Mitesh\OneDrive\Desktop\images.jpg')

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 )

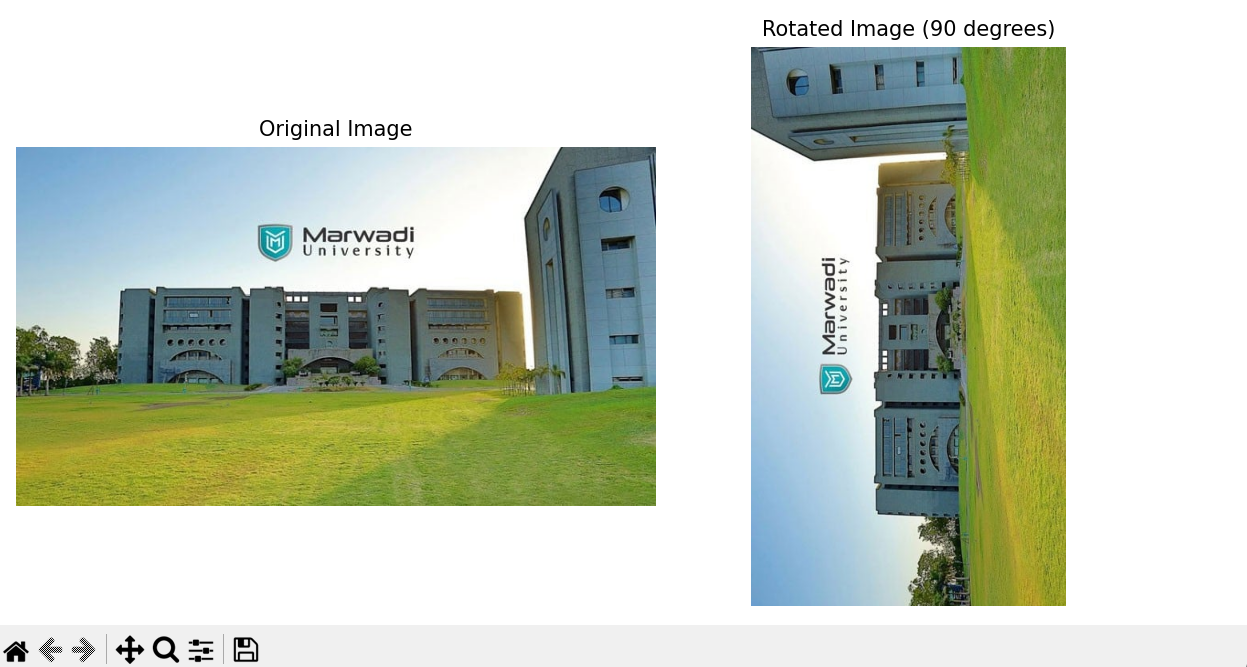
plt.title('Rotated Image (90 degrees)')

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\Mitesh\OneDrive\Desktop\images.jpg')

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 )

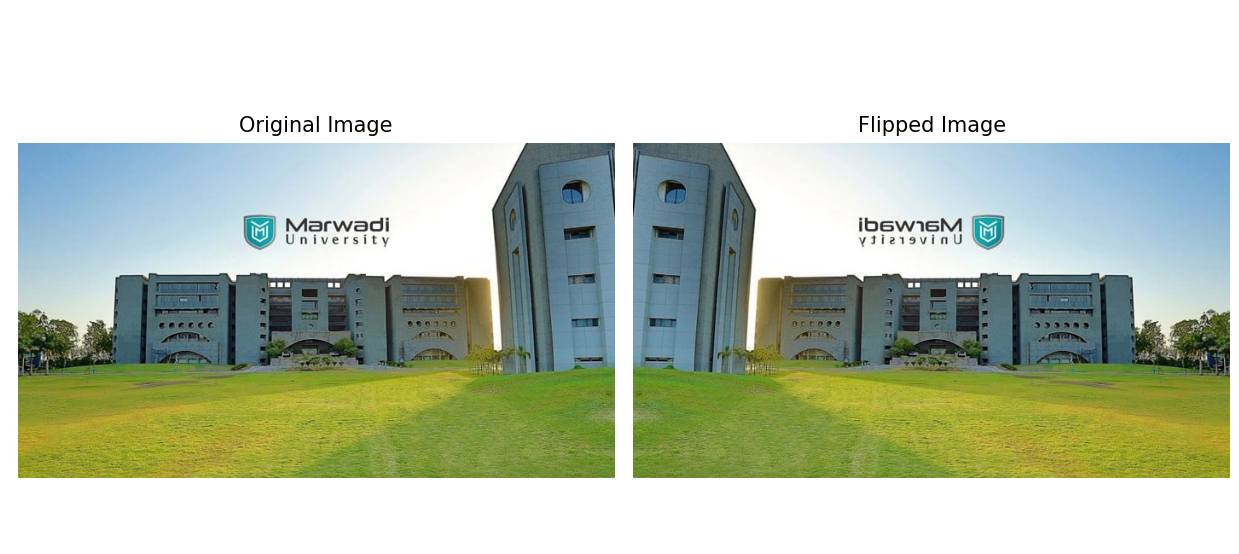
plt.title('Flipped Image')

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\Mitesh\OneDrive\Desktop\images.jpg')

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)

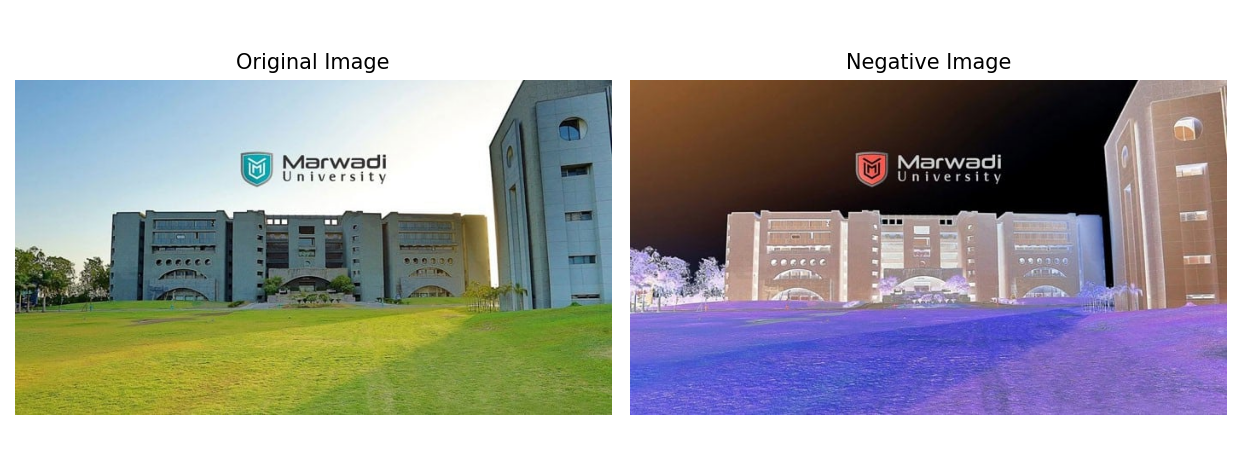
plt.title('Negative Image')

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\Mitesh\OneDrive\Desktop\images.jpg')

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\Mitesh\OneDrive\Desktop\images.jpg')

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')

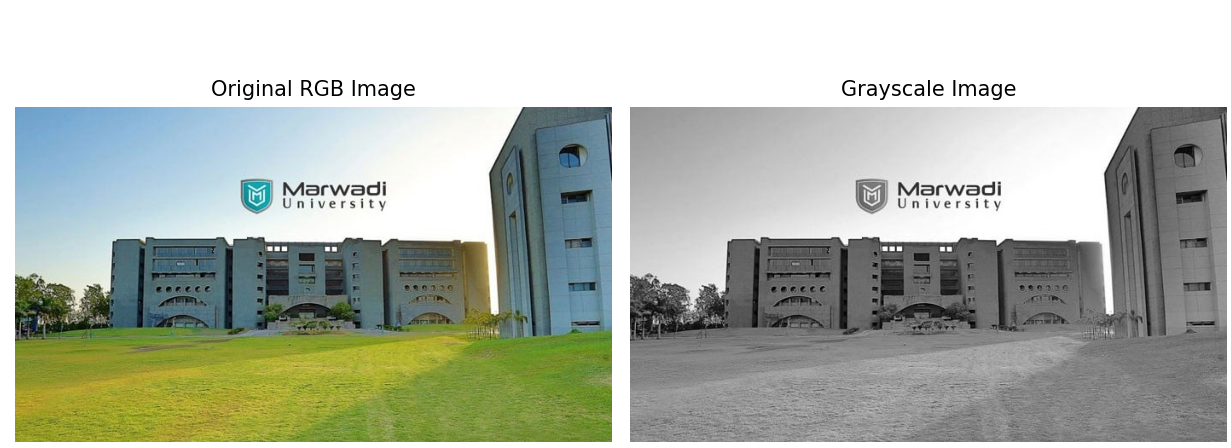
plt.title('Grayscale Image')

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\Mitesh\OneDrive\Desktop\images.jpg')

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')

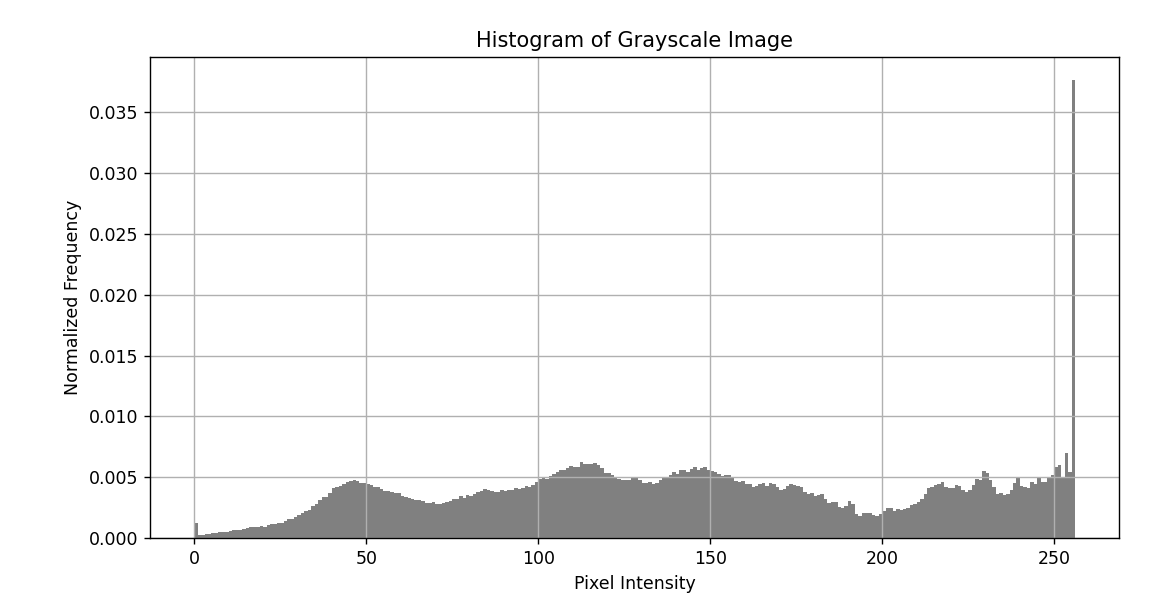
plt.ylabel('Normalized Frequency')

plt.title('Histogram of Grayscale Image')

plt.grid(True)

plt.show()

Output

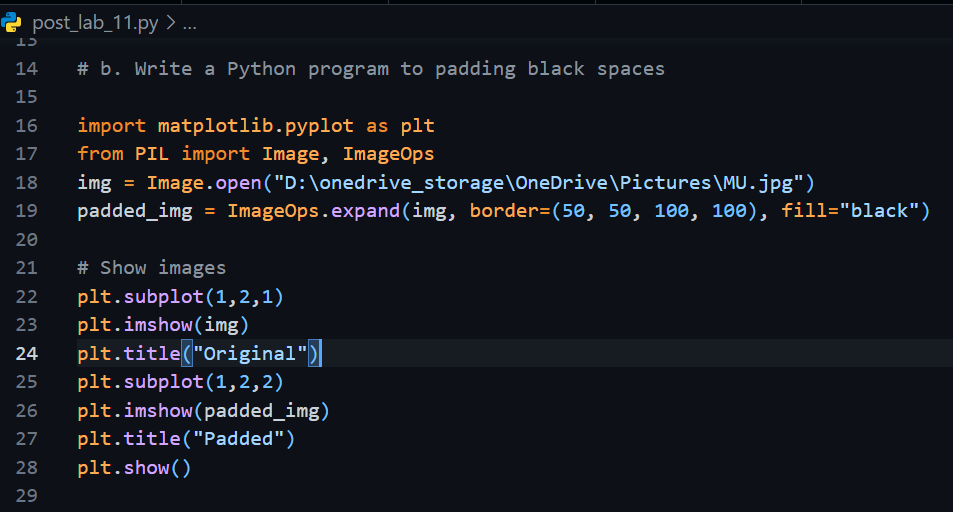


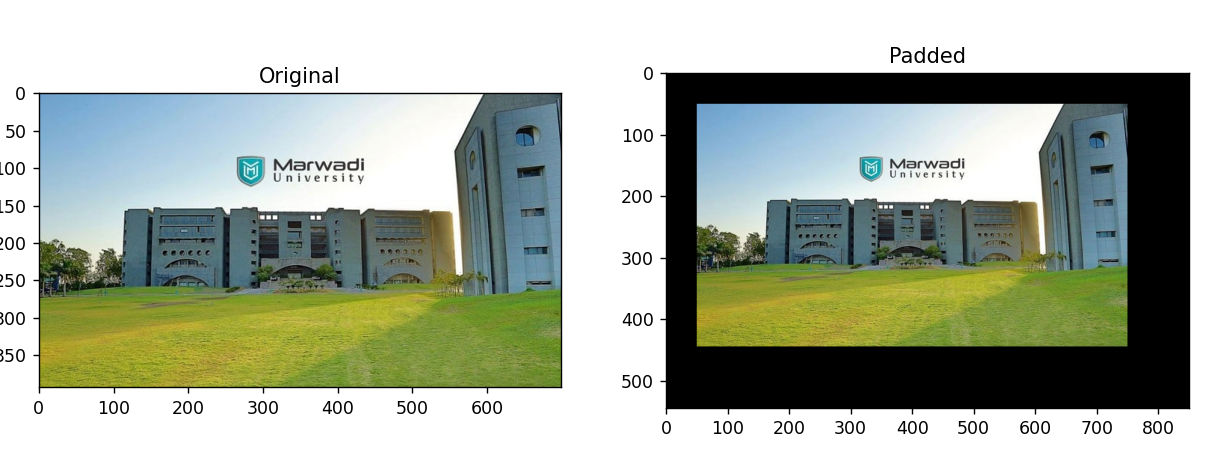
**Post Lab Exercise:**

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

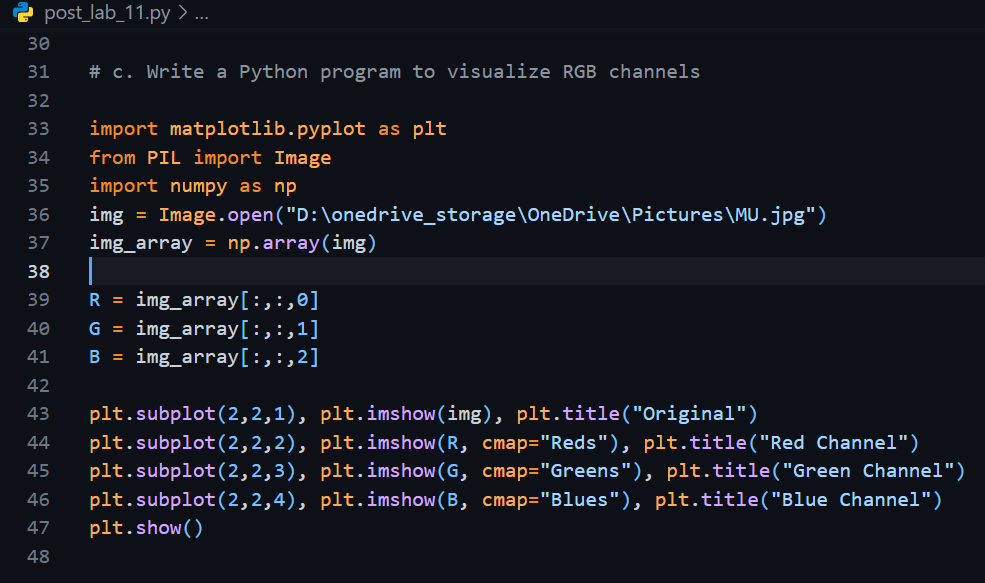


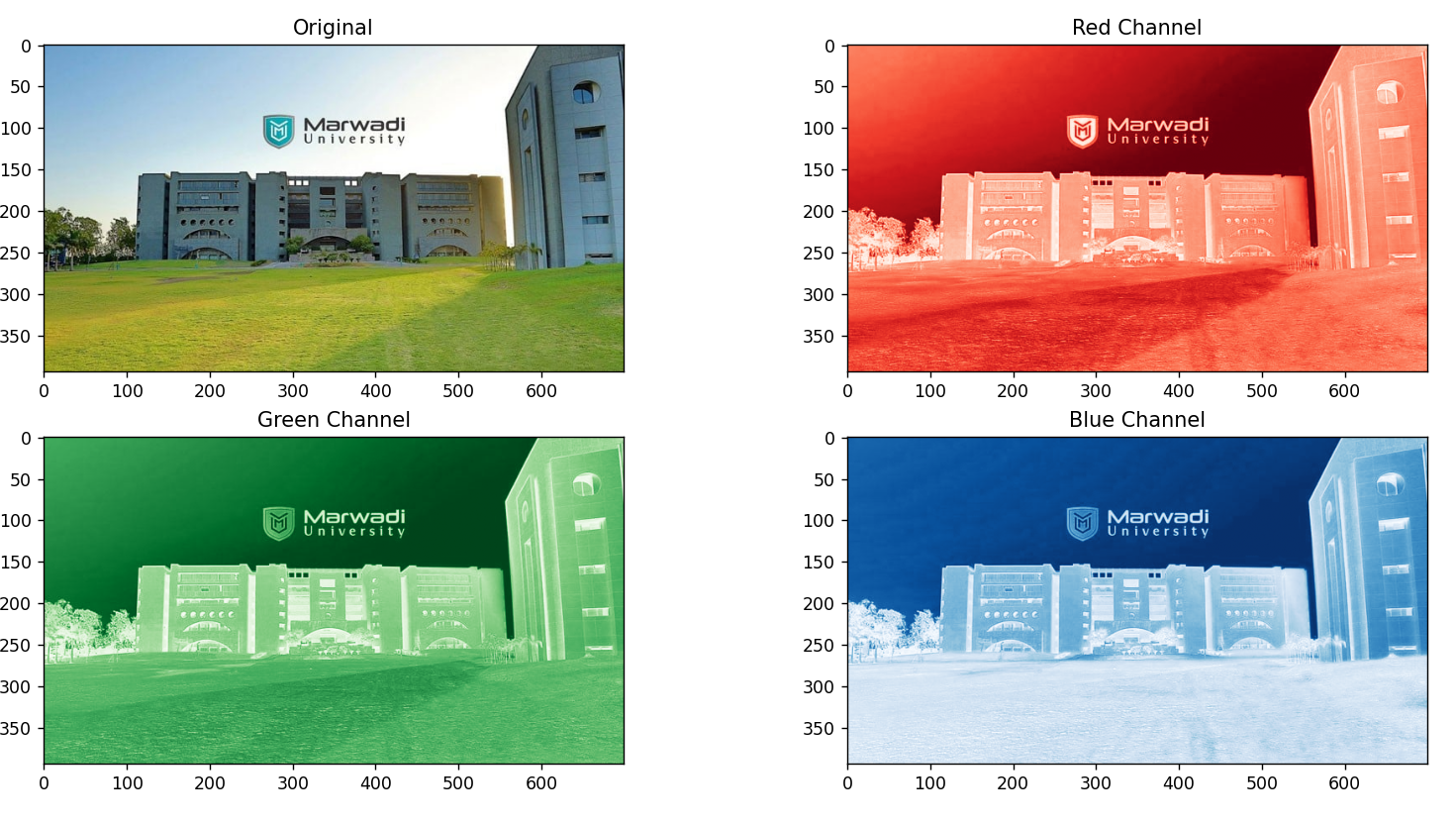
1. Write a Python program to padding black spaces





1. Write a Python program to visualize RGB channels





More Practice

Reference : <https://www.analyticsvidhya.com/blog/2021/05/image-processing-using-numpy-with-practical-implementation-and-code/>

**Github link :**