	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:92400133189**

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
```

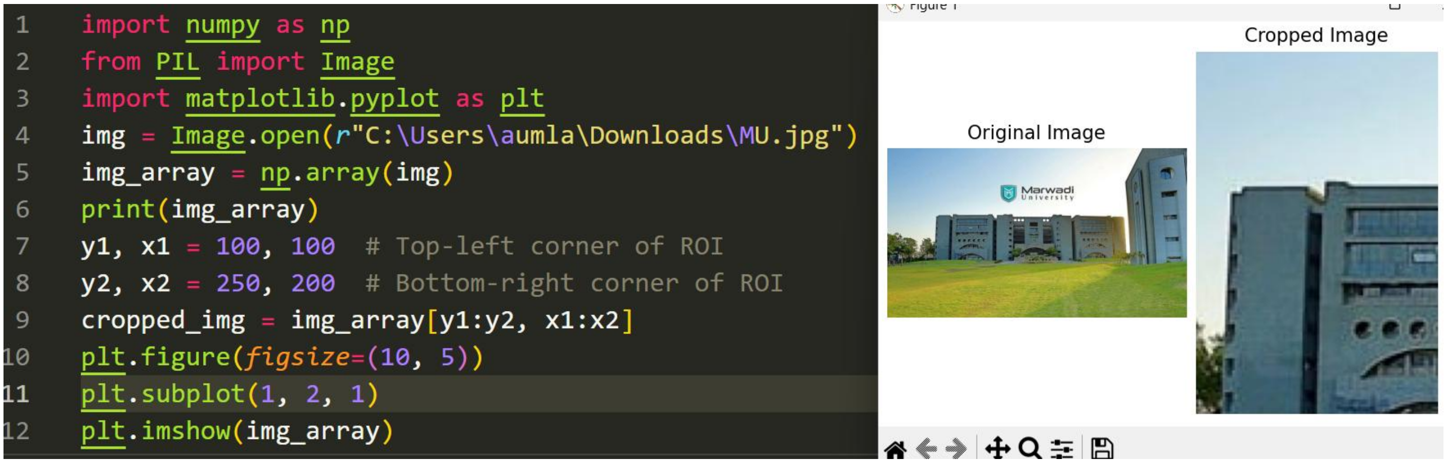
Crop Image

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



```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open(r"C:\Users\aumla\Downloads\MU.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)
```

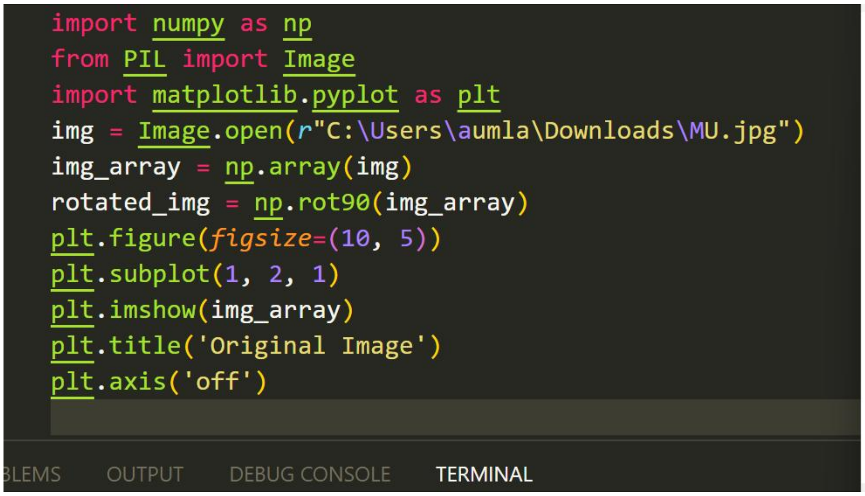
Rotate Image

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

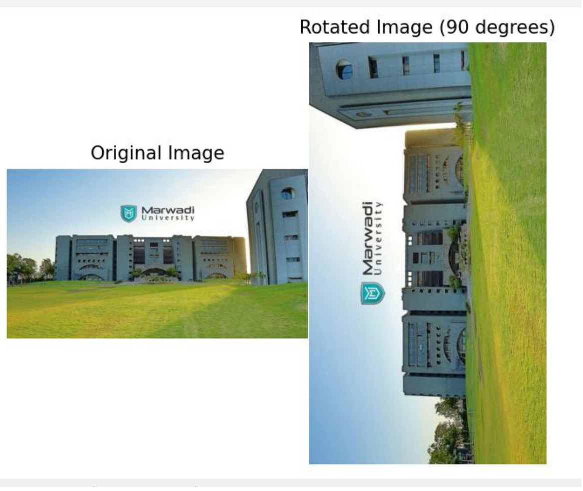
Example:

```
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



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



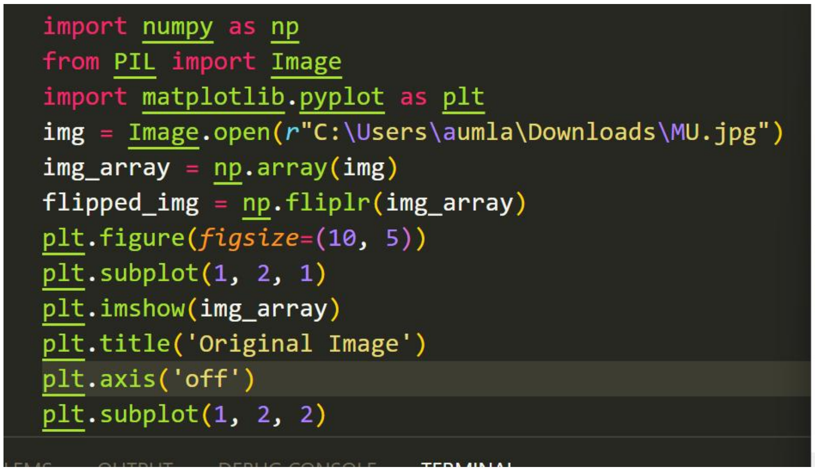
Flip Image

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

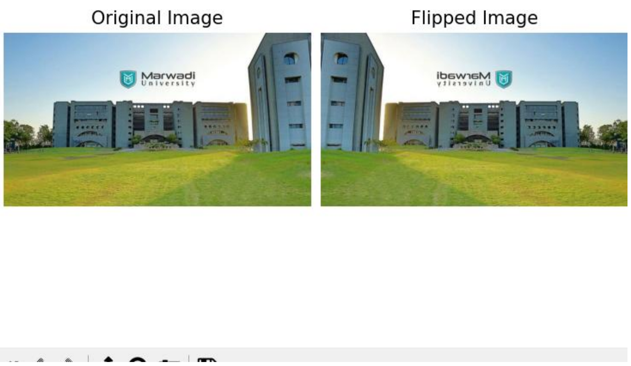
Example:

```
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



```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open(r"C:\Users\aumla\Downloads\MU.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)
```



Negative of an Image

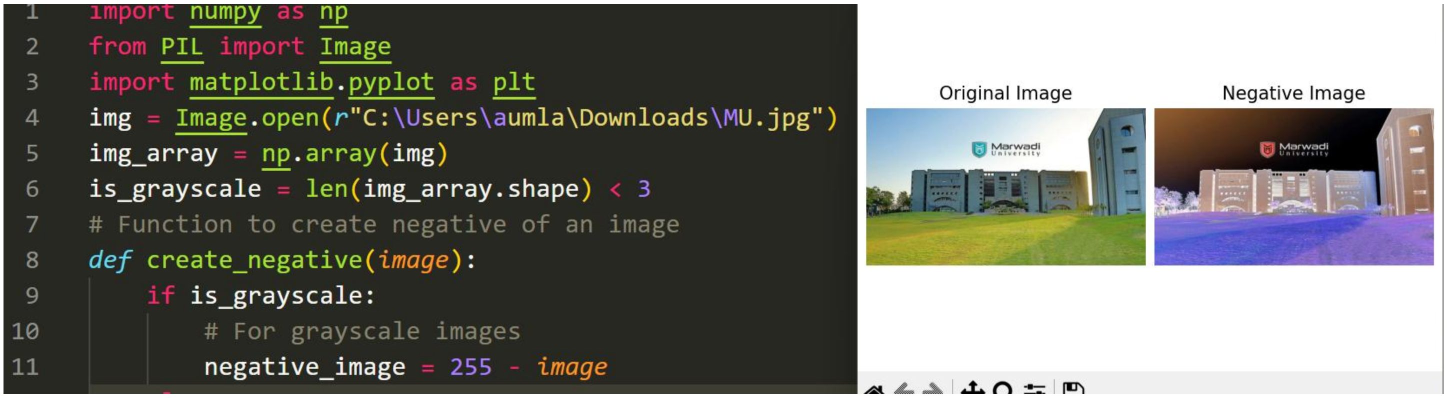
The negative of an image is made by reversing its pixel values. In grayscale images, each pixel's value is subtracted from the maximum (255 for 8-bit images). In color images, this is done separately for each color channel.

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



```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open(r"C:\Users\aumla\Downloads\MU.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
```

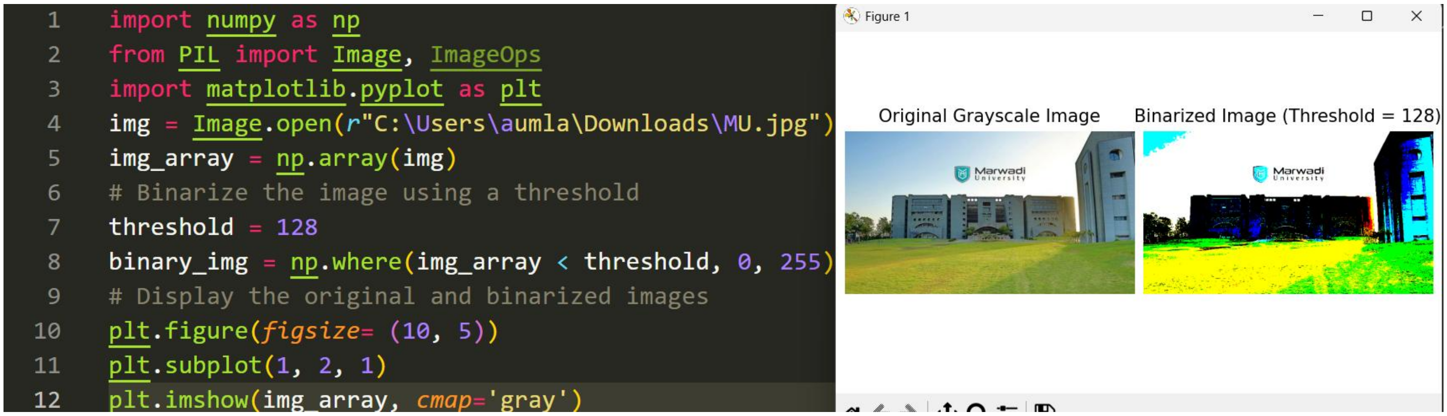
Binarize Image

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



```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
img = Image.open(r"C:\Users\aumla\Downloads\MU.jpg")
img_array = np.array(img)
# Binarize the image using a threshold
threshold = 128
binary_img = np.where(img_array < threshold, 0, 255)
# Display the original and binarized images
plt.figure(figsize= (10, 5))
plt.subplot(1, 2, 1)
plt.imshow(img_array, cmap='gray')
```

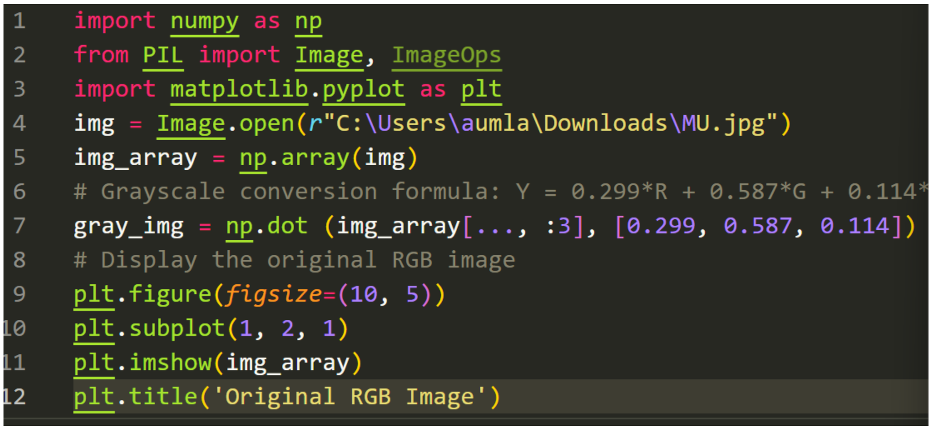
Color Space Conversion

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.

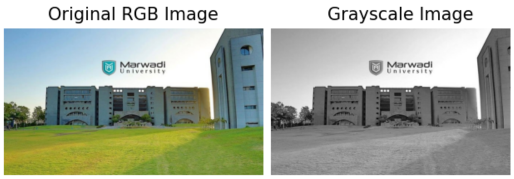
Example

```
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



```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
img = Image.open(r"C:\Users\aumla\Downloads\MU.jpg")
img_array = np.array(img)
# Grayscale conversion formula: Y = 0.299*R + 0.587*G + 0.114*
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')
```

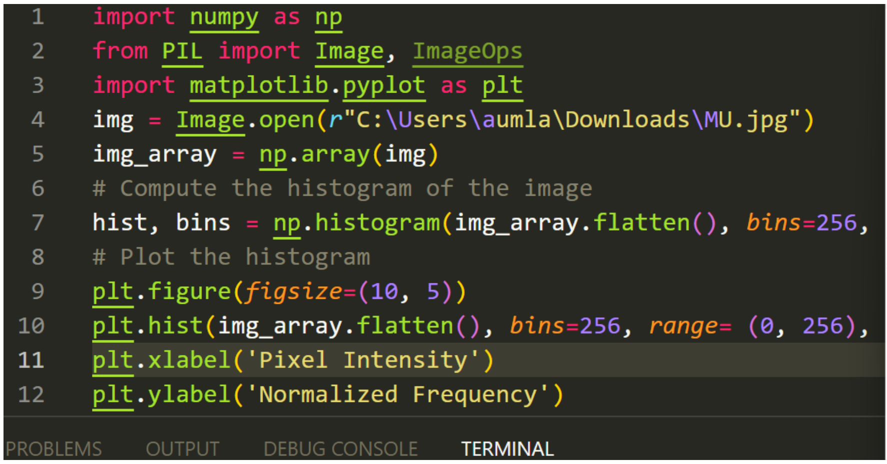


Pixel Intensity Histogram
The histogram shows the distribution of pixel values in an image. The image is flattened into a one-dimensional array to compute the histogram.

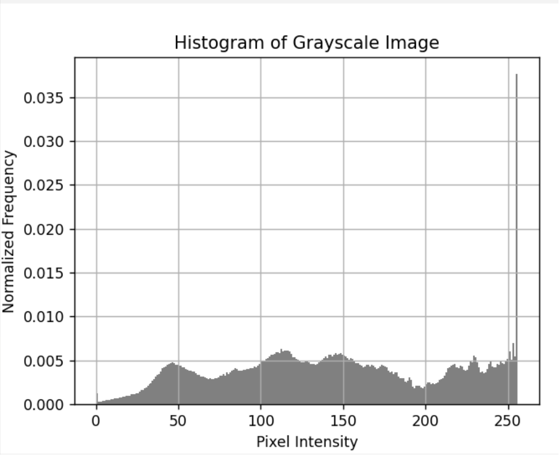
Example:

```
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

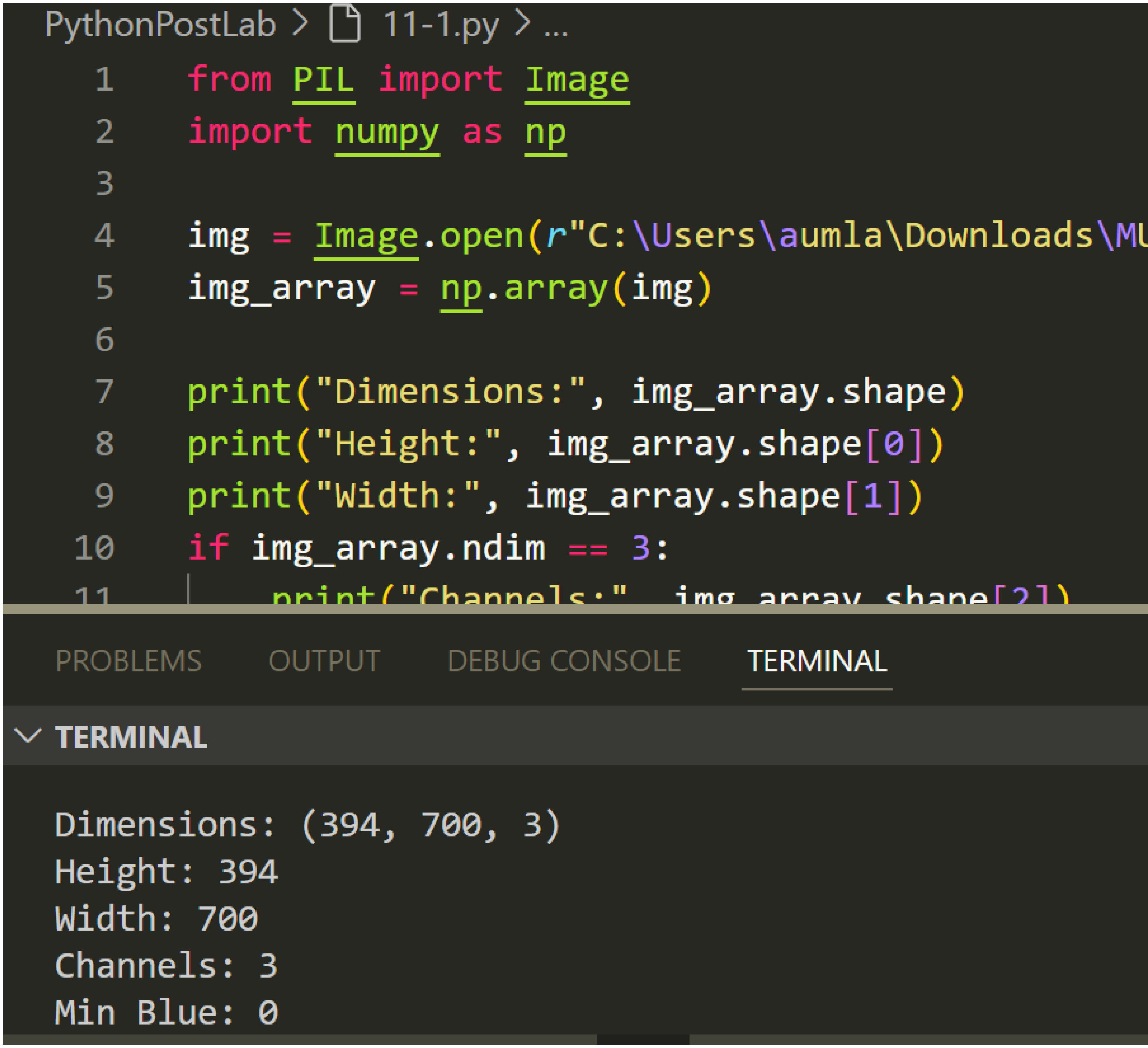


```
import numpy as np
from PIL import Image, ImageOps
import matplotlib.pyplot as plt
img = Image.open(r"C:\Users\aumla\Downloads\MU.jpg")
img_array = np.array(img)
# Compute the histogram of the image
hist, bins = np.histogram(img_array.flatten(), bins=256,
# Plot the histogram
plt.figure(figsize=(10, 5))
plt.hist(img_array.flatten(), bins=256, range= (0, 256),
plt.xlabel('Pixel Intensity')
plt.ylabel('Normalized Frequency')
```



Post Lab Exercise:

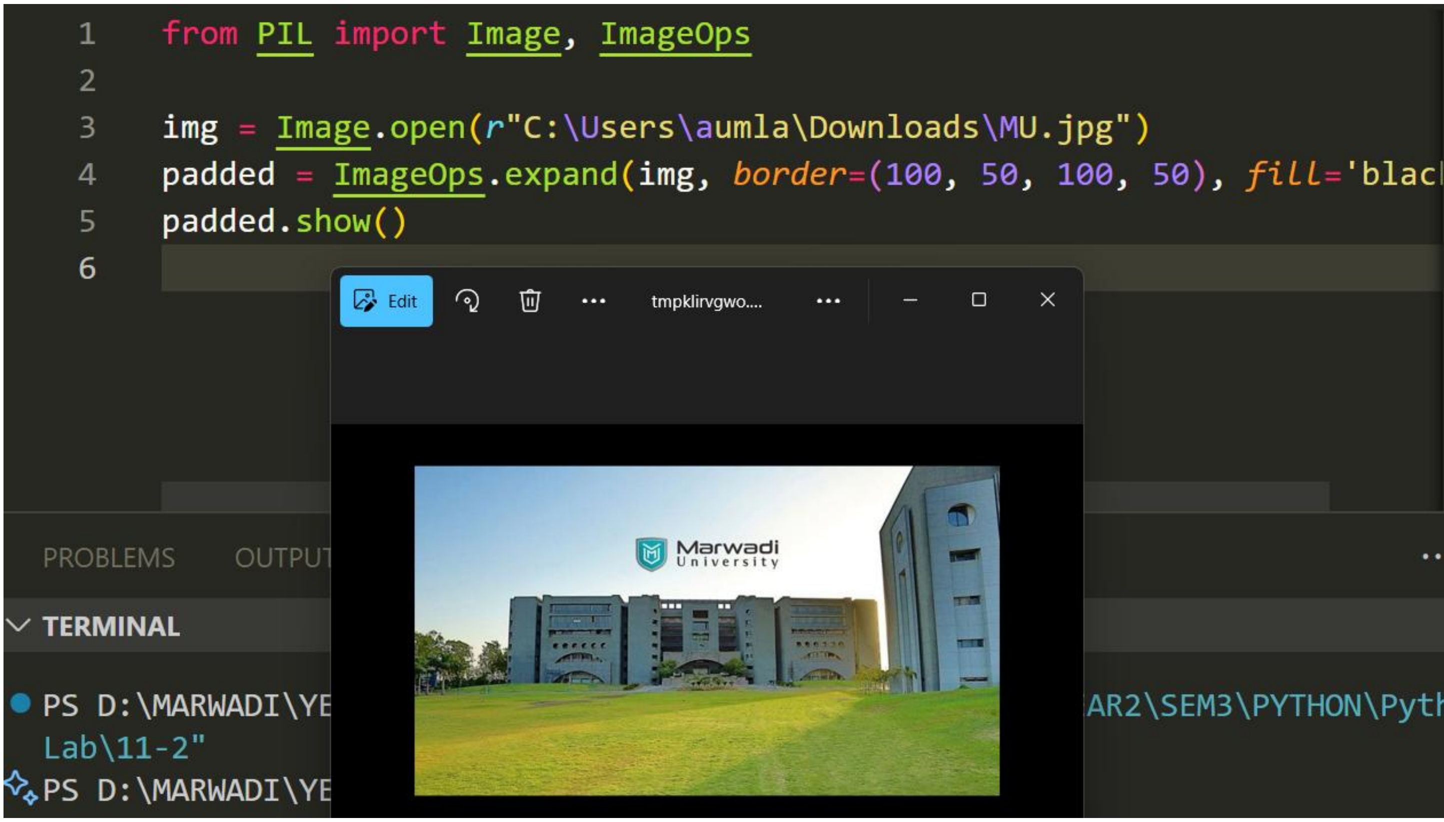
a. Write a Python program to display details of an image (dimension of an image, shape of an image, min



```python
from PIL import Image
import numpy as np

img = Image.open(r"C:\Users\aumla\Downloads\MU
img_array = np.array(img)

print("Dimensions:", img_array.shape)
print("Height:", img_array.shape[0])
print("Width:", img_array.shape[1])
if img_array.ndim == 3:
    print("Channels:", img_array.shape[2])
```

```
Dimensions: (394, 700, 3)
Height: 394
Width: 700
Channels: 3
Min Blue: 0
```

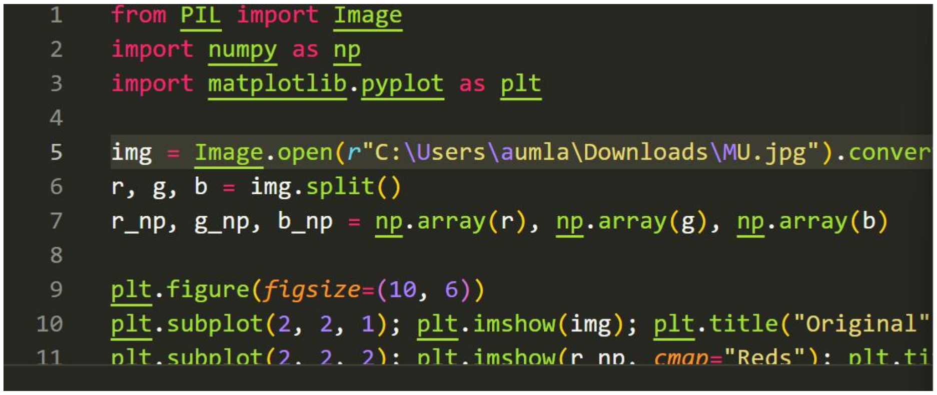
b. pixel value at channel B).



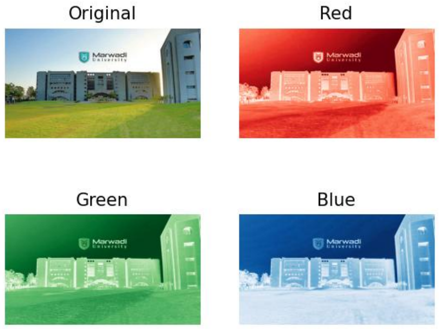
```
from PIL import Image, ImageOps

img = Image.open(r"C:\Users\aumla\Downloads\MU.jpg")
padded = ImageOps.expand(img, border=(100, 50, 100, 50), fill='blac
padded.show()
```

```
PS D:\MARWADI\YE                          AR2\SEM3\PYTHON\Pyth
Lab\11-2"
PS D:\MARWADI\YE
```

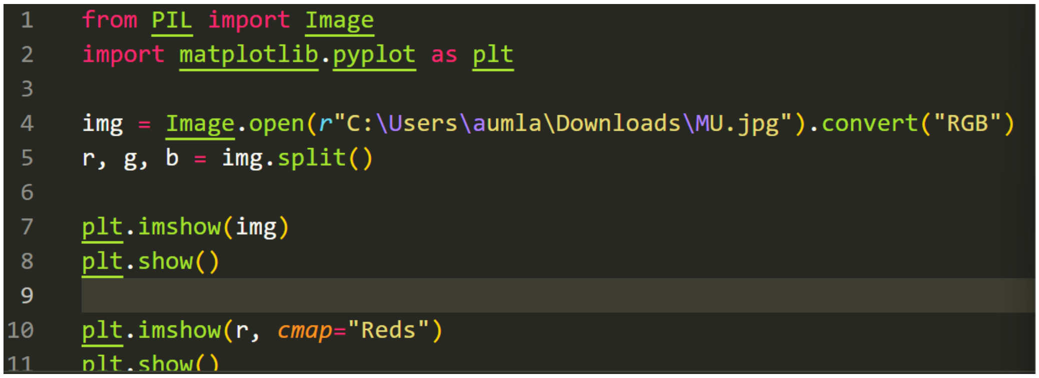
c. Write a Python program to padding black spaces



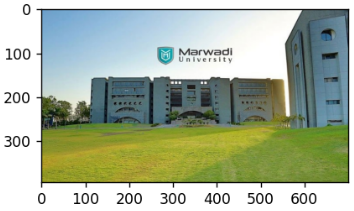
```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

img = Image.open(r"C:\Users\aumla\Downloads\MU.jpg").conver
r, g, b = img.split()
r_np, g_np, b_np = np.array(r), np.array(g), np.array(b)

plt.figure(figsize=(10, 6))
plt.subplot(2, 2, 1); plt.imshow(img); plt.title("Original"
plt.subplot(2, 2, 2); plt.imshow(r_np, cmap="Reds"); plt.ti
```



d. Write a Python program to visualize RGB channels



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

img = Image.open(r"C:\Users\aumla\Downloads\MU.jpg").convert("RGB")
r, g, b = img.split()

plt.imshow(img)
plt.show()


plt.imshow(r, cmap="Reds")
plt.show()
```



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

Github : PythonPostLab/11 at main · Om-Lathigara/PythonPostLab