BAHRIA UNIVERSITY, ISLAMABAD

Department of Computer Science

CEN 444

Digital Image Processing

Lab 5

Student Name: Kacho Hannan Haider

Enrolment No: 01-134212-213

Title: Basic Image Processing Operations.

Objectives: To introduce you to the basic functions in the Opencv. To be able to read images from the disk display them, write them back to the disk and perform conversions between different image classes.

Tools Used: Jupyter Nootebook

Instructions for Submission:

- **Create a main repository** for the entire course on GitHub.
- **Within this main repository**, create separate folders for each week's content.
- **Upload your Jupyter notebooks** into the appropriate folder for each week.
- For the final submission, simply **upload a PDF file** that contains the GitHub link to your repository.

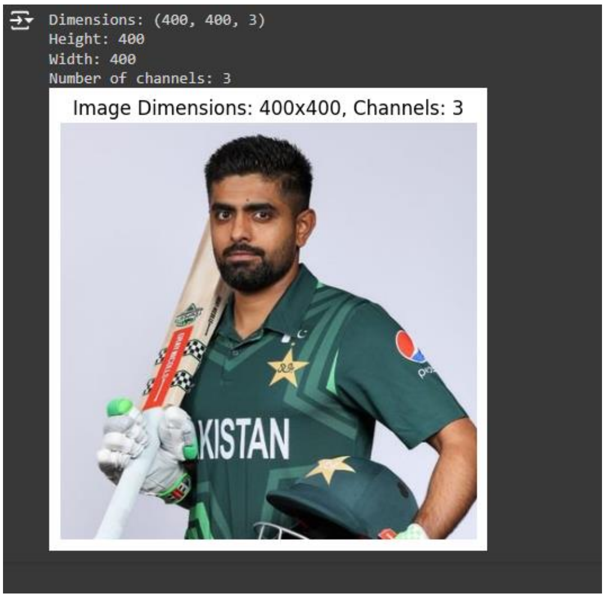
Task 1:

a) Load any image, find its dimensions and number of channels and display it.

CODE:

```
import cv2
import matplotlib.pyplot as plt

image = cv2.imread(r'/content/babar1.jpg')

if image is None:
    print("Error: Image not found or unable to load.")
else:
    dimensions = image.shape
    height, width, channels = dimensions

    print(f'Dimensions: {dimensions}')
    print(f'Height: {height}')
    print(f'Width: {width}')
    print(f'Number of channels: {channels}')
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title(f'Image Dimensions: {width}x{height}, Channels: {channels}')
    plt.axis('off')
    plt.show()
```

OUTPUT:

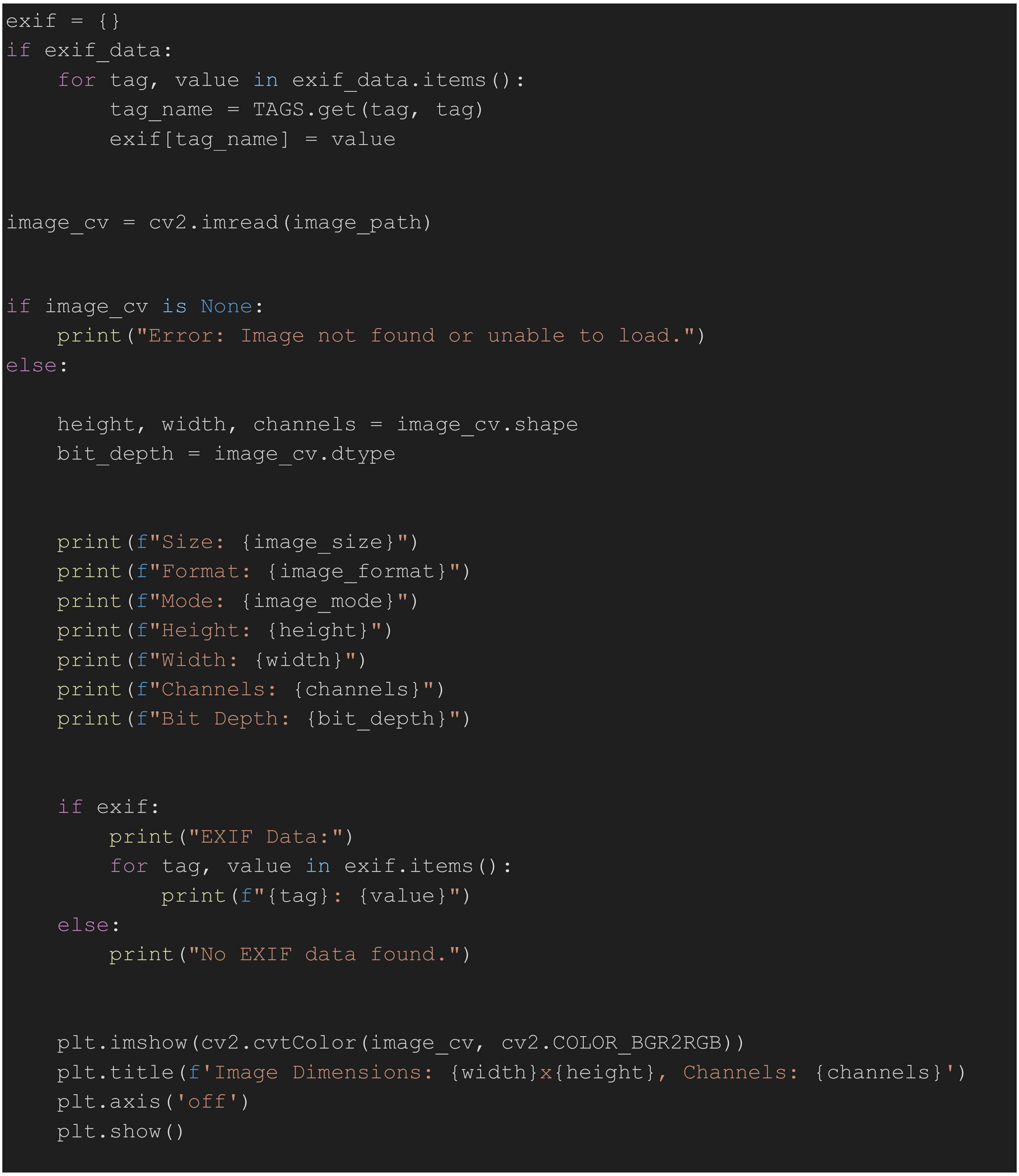


```
Dimensions: (400, 400, 3)
Height: 400
Width: 400
Number of channels: 3
```

b) Find the size, date, coding method, bit depth, height and width.

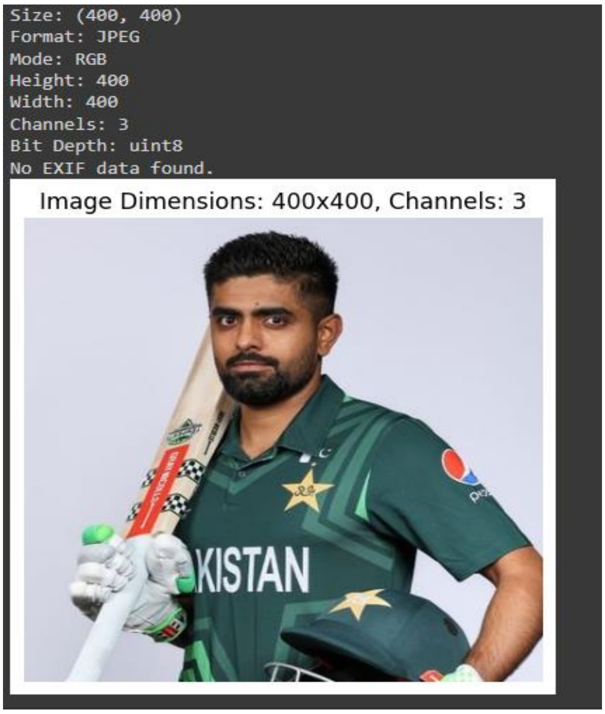
CODE:

```
from PIL import Image
from PIL.ExifTags import TAGS
import cv2
import matplotlib.pyplot as plt


image_path = '/content/babar1.jpg'
image_pil = Image.open(image_path)


image_size = image_pil.size
image_format = image_pil.format
image_mode = image_pil.mode


exif_data = image_pil._getexif()
```



```python
exif = {}
if exif_data:
    for tag, value in exif_data.items():
        tag_name = TAGS.get(tag, tag)
        exif[tag_name] = value


image_cv = cv2.imread(image_path)


if image_cv is None:
    print("Error: Image not found or unable to load.")
else:

    height, width, channels = image_cv.shape
    bit_depth = image_cv.dtype


    print(f"Size: {image_size}")
    print(f"Format: {image_format}")
    print(f"Mode: {image_mode}")
    print(f"Height: {height}")
    print(f"Width: {width}")
    print(f"Channels: {channels}")
    print(f"Bit Depth: {bit_depth}")


    if exif:
        print("EXIF Data:")
        for tag, value in exif.items():
            print(f"{tag}: {value}")
    else:
        print("No EXIF data found.")


    plt.imshow(cv2.cvtColor(image_cv, cv2.COLOR_BGR2RGB))
    plt.title(f'Image Dimensions: {width}x{height}, Channels: {channels}')
    plt.axis('off')
    plt.show()
```

OUTPUT:

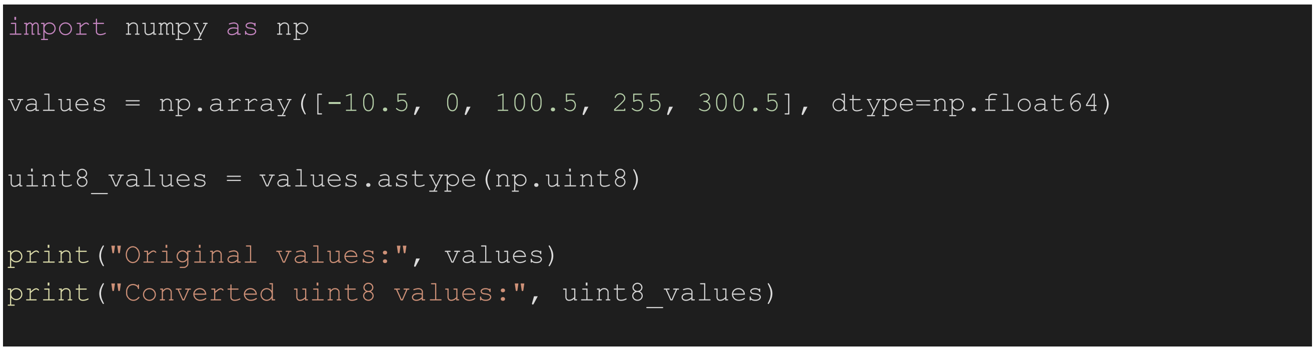


```
Size: (400, 400)
Format: JPEG
Mode: RGB
Height: 400
Width: 400
Channels: 3
Bit Depth: uint8
No EXIF data found.
```

Image Dimensions: 400x400, Channels: 3

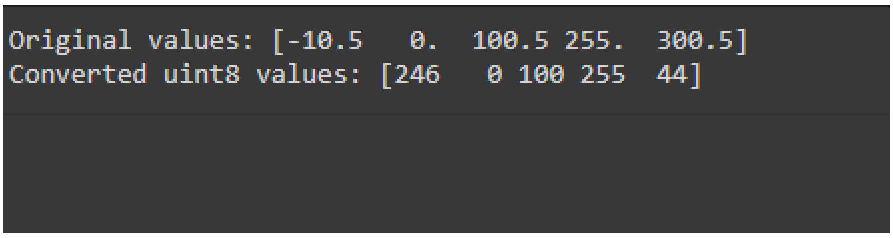
c) What happens if you convert a double having values outside the range [0 255] to an uint8?

CODE:



```
import numpy as np

values = np.array([-10.5, 0, 100.5, 255, 300.5], dtype=np.float64)

uint8_values = values.astype(np.uint8)

print("Original values:", values)
print("Converted uint8 values:", uint8_values)
```

OUTPUT:

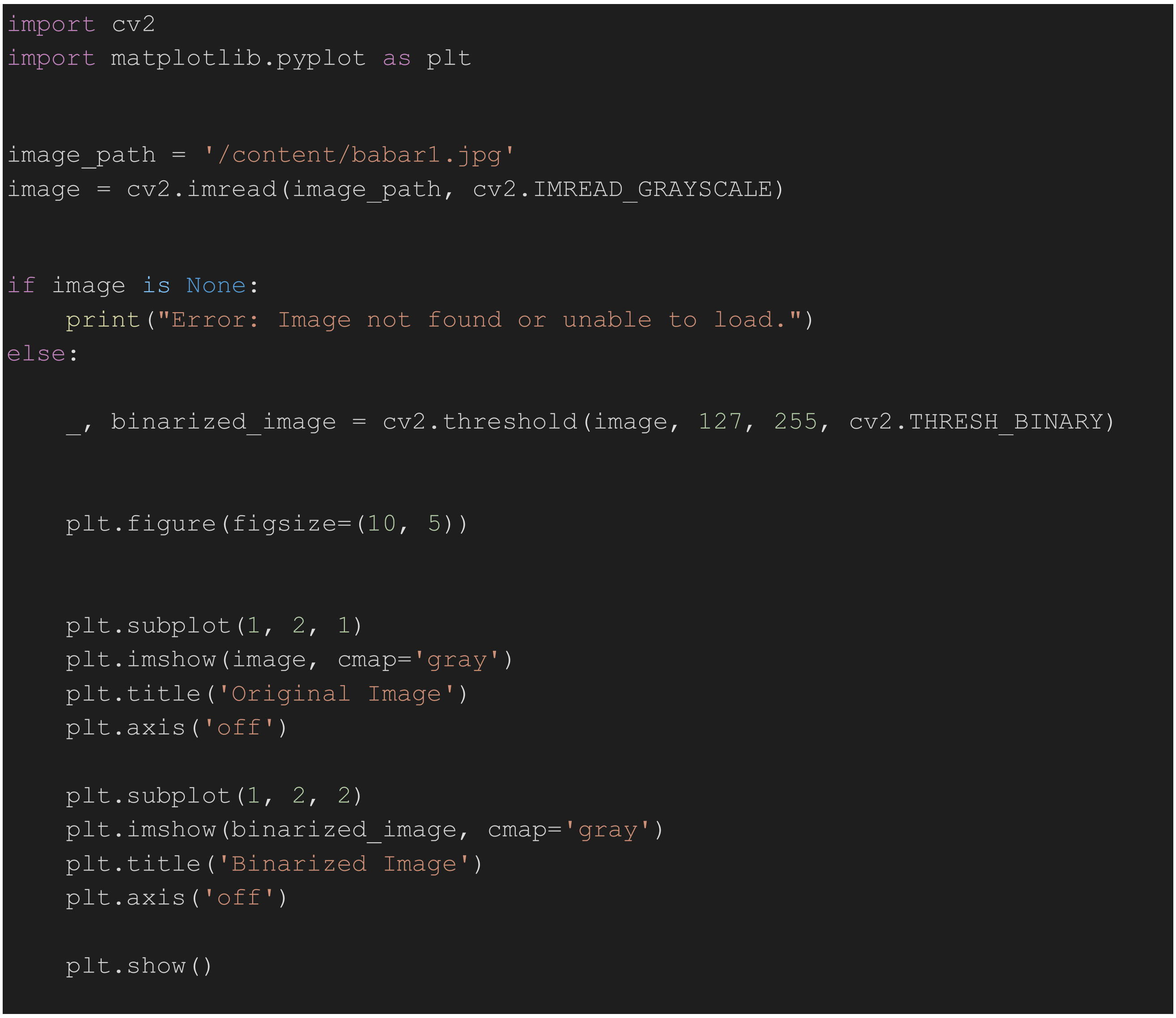


```
Original values: [-10.5   0.  100.5 255.  300.5]
Converted uint8 values: [246   0 100 255  44]
```

Task 2:

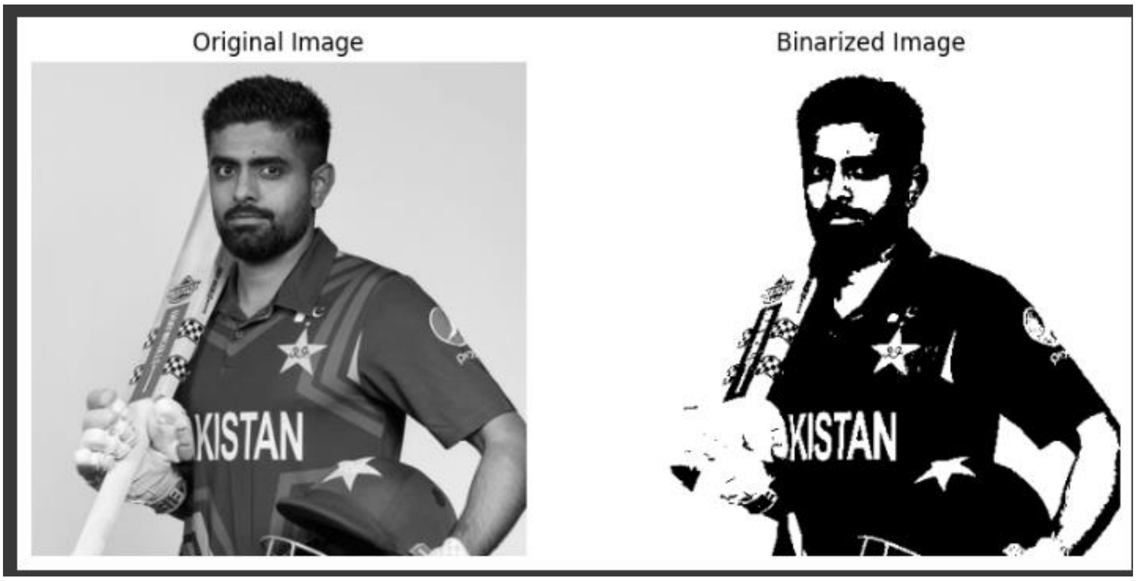
Load any image, binarize it, using the function 'cv2.threshold' with a threshold of 127 and display both original and binarized images.

CODE:



```python
import cv2
import matplotlib.pyplot as plt


image_path = '/content/babar1.jpg'
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


if image is None:
    print("Error: Image not found or unable to load.")
else:

    _, binarized_image = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)


    plt.figure(figsize=(10, 5))


    plt.subplot(1, 2, 1)
    plt.imshow(image, cmap='gray')
    plt.title('Original Image')
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(binarized_image, cmap='gray')
    plt.title('Binarized Image')
    plt.axis('off')

    plt.show()
```

OUTPUT:



Task 3:

The function rgb2gray() can be used to convert three channel-colored images into single channel gray scale images.

Write program which reads any image in the. Find the size of the image and the number of channels in it. Use if else. If it is a three-channel image, then:

- display that it is a three-channel image
- then convert it to grayscale.
- Now binarize the gray image.
- Finally, display it.

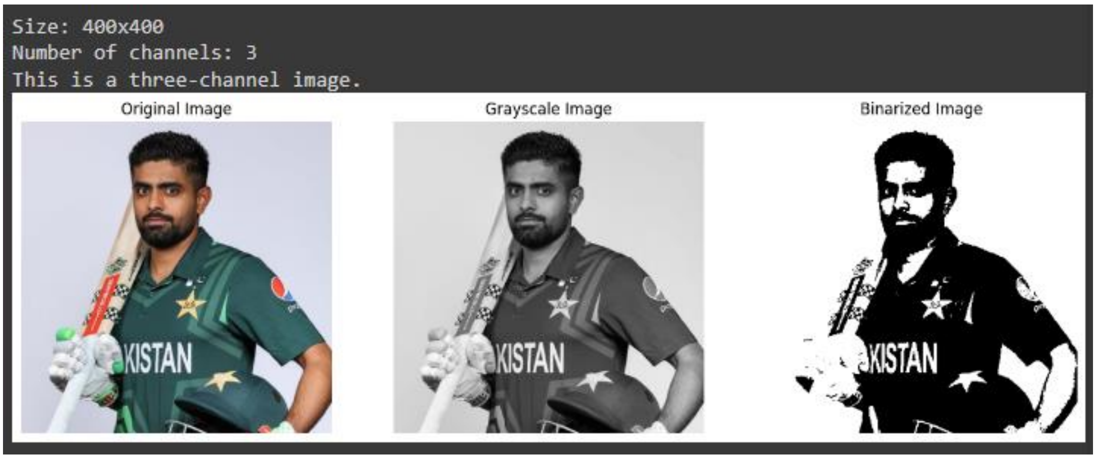
If it is a one-channel image, then:

- display that it is a one-channel image
- Just display image.

CODE:

```
import cv2
import matplotlib.pyplot as plt


def rgb2gray(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)


image_path = '/content/babar1.jpg'
image = cv2.imread(image_path)


if image is None:
    print("Error: Image not found or unable to load.")
else:

    dimensions = image.shape
    height, width = dimensions[:2]
    channels = dimensions[2] if len(dimensions) == 3 else 1

    print(f'Size: {width}x{height}')
    print(f'Number of channels: {channels}')

    if channels == 3:
        print("This is a three-channel image.")


        gray_image = rgb2gray(image)
        _, binarized_image = cv2.threshold(gray_image, 127, 255, 
cv2.THRESH_BINARY)


        plt.figure(figsize=(15, 5))


        plt.subplot(1, 3, 1)
        plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
        plt.title('Original Image')
        plt.axis('off')


        plt.subplot(1, 3, 2)
        plt.imshow(gray_image, cmap='gray')
        plt.title('Grayscale Image')
        plt.axis('off')


        plt.subplot(1, 3, 3)
        plt.imshow(binarized_image, cmap='gray')
        plt.title('Binarized Image')
        plt.axis('off')

        plt.show()

    elif channels == 1:
        print("This is a one-channel image.")


        plt.imshow(image, cmap='gray')
        plt.title('Grayscale Image')
        plt.axis('off')
        plt.show()
```

OUTPUT:

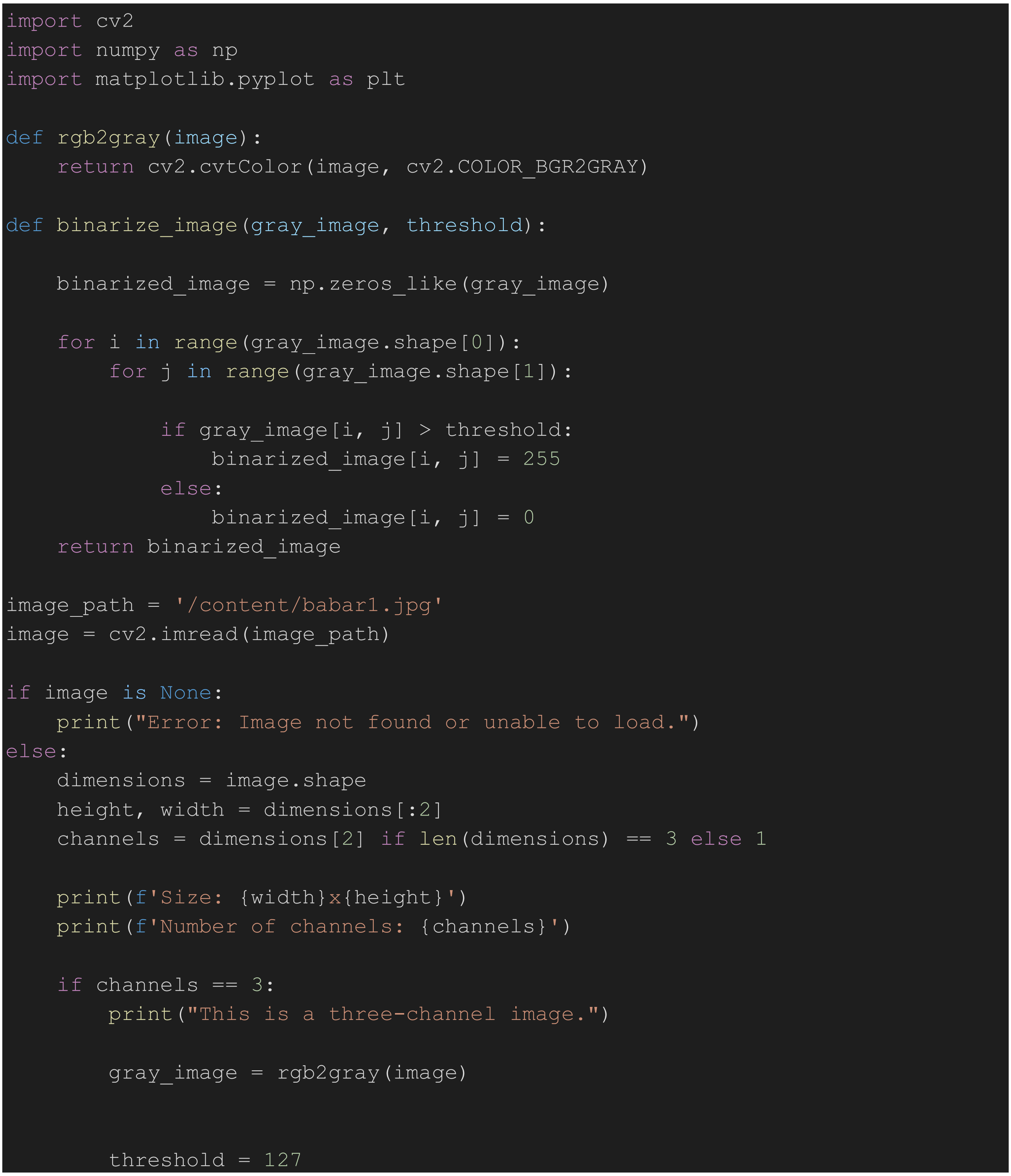


```
Size: 400x400
Number of channels: 3
This is a three-channel image.
```

Task 4:

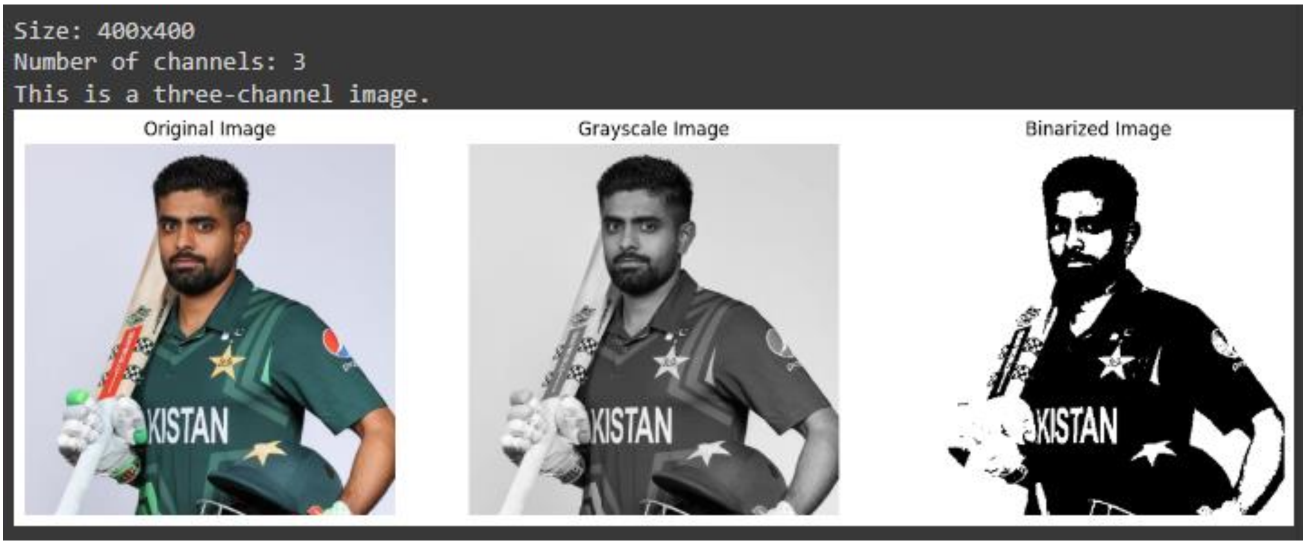
Write which reads the image. Binarize the image using your own implementation rather than the functions. Use two loops (nested), compare each value with a given threshold and generate the output image.

CODE:



```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def rgb2gray(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

def binarize_image(gray_image, threshold):

    binarized_image = np.zeros_like(gray_image)

    for i in range(gray_image.shape[0]):
        for j in range(gray_image.shape[1]):

            if gray_image[i, j] > threshold:
                binarized_image[i, j] = 255
            else:
                binarized_image[i, j] = 0
    return binarized_image

image_path = '/content/babar1.jpg'
image = cv2.imread(image_path)

if image is None:
    print("Error: Image not found or unable to load.")
else:
    dimensions = image.shape
    height, width = dimensions[:2]
    channels = dimensions[2] if len(dimensions) == 3 else 1

    print(f'Size: {width}x{height}')
    print(f'Number of channels: {channels}')

    if channels == 3:
        print("This is a three-channel image.")

        gray_image = rgb2gray(image)


        threshold = 127
        binarized_image = binarize_image(gray_image, threshold)

        plt.figure(figsize=(15, 5))

        plt.subplot(1, 3, 1)
        plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
        plt.title('Original Image')
        plt.axis('off')

        plt.subplot(1, 3, 2)
        plt.imshow(gray_image, cmap='gray')
        plt.title('Grayscale Image')
        plt.axis('off')

        plt.subplot(1, 3, 3)
        plt.imshow(binarized_image, cmap='gray')
        plt.title('Binarized Image')
        plt.axis('off')

        plt.show()

    elif channels == 1:
        print("This is a one-channel image.")

        plt.imshow(image, cmap='gray')
        plt.title('Grayscale Image')
        plt.axis('off')
        plt.show()
```

OUTPUT:

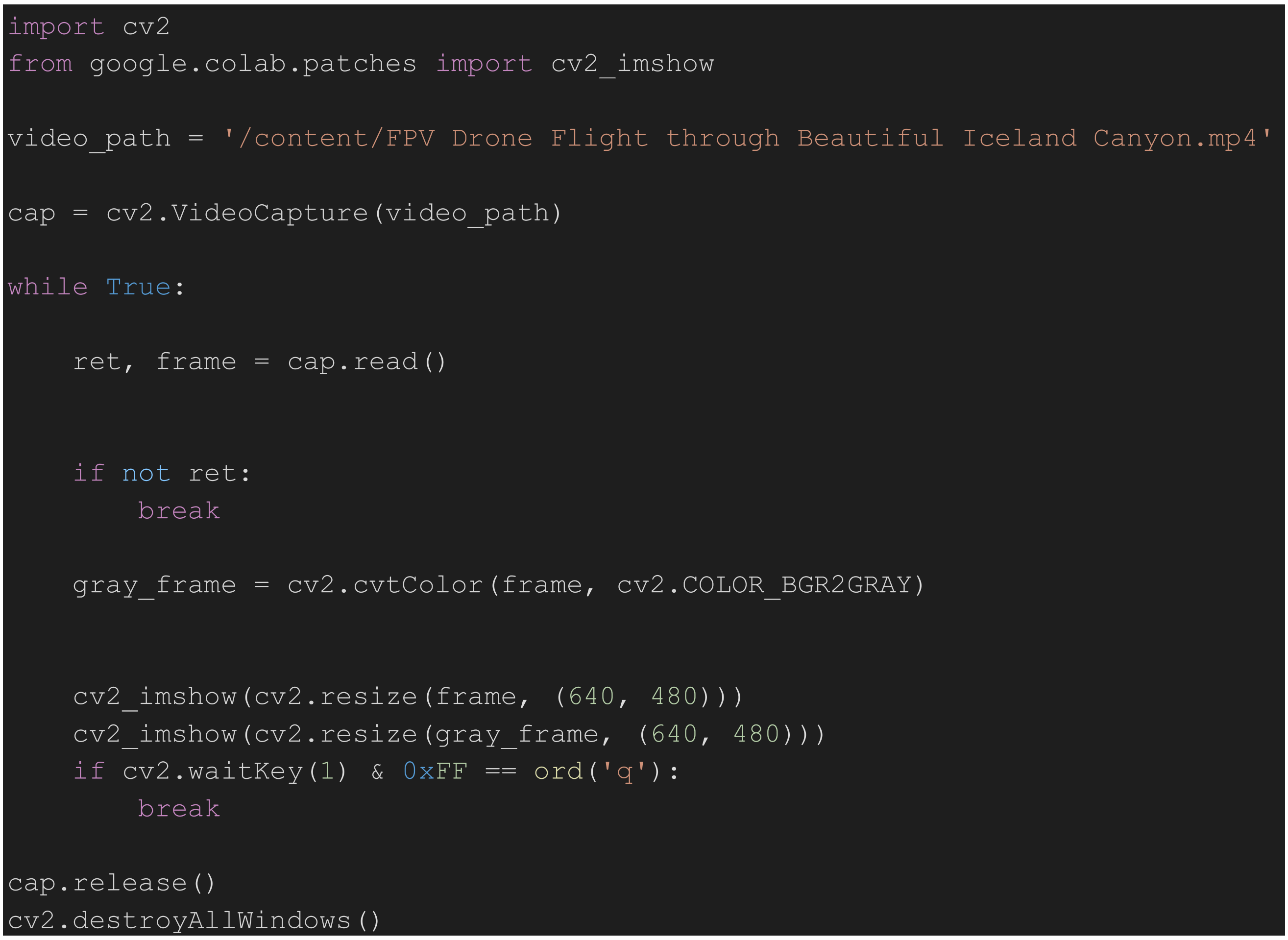


```
Size: 400x400
Number of channels: 3
This is a three-channel image.
```

Task 5:

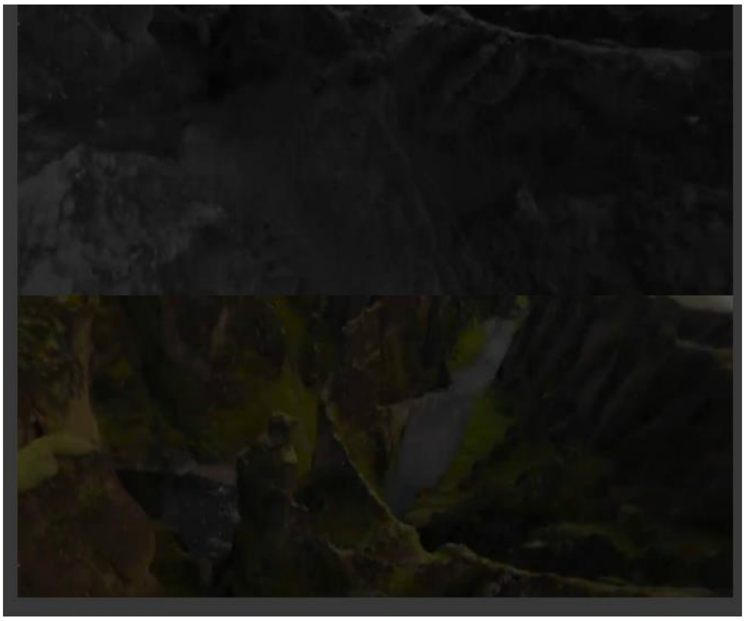
Open webcam and perform gray scaling on the video frames, you can use opencv functions for gray scale.

CODE:



```
import cv2
from google.colab.patches import cv2_imshow

video_path = '/content/FPV Drone Flight through Beautiful Iceland Canyon.mp4'

cap = cv2.VideoCapture(video_path)

while True:

    ret, frame = cap.read()


    if not ret:
        break

    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    cv2_imshow(cv2.resize(frame, (640, 480)))
    cv2_imshow(cv2.resize(gray_frame, (640, 480)))
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

OUTPUT:



Submission Date: **Signature:**