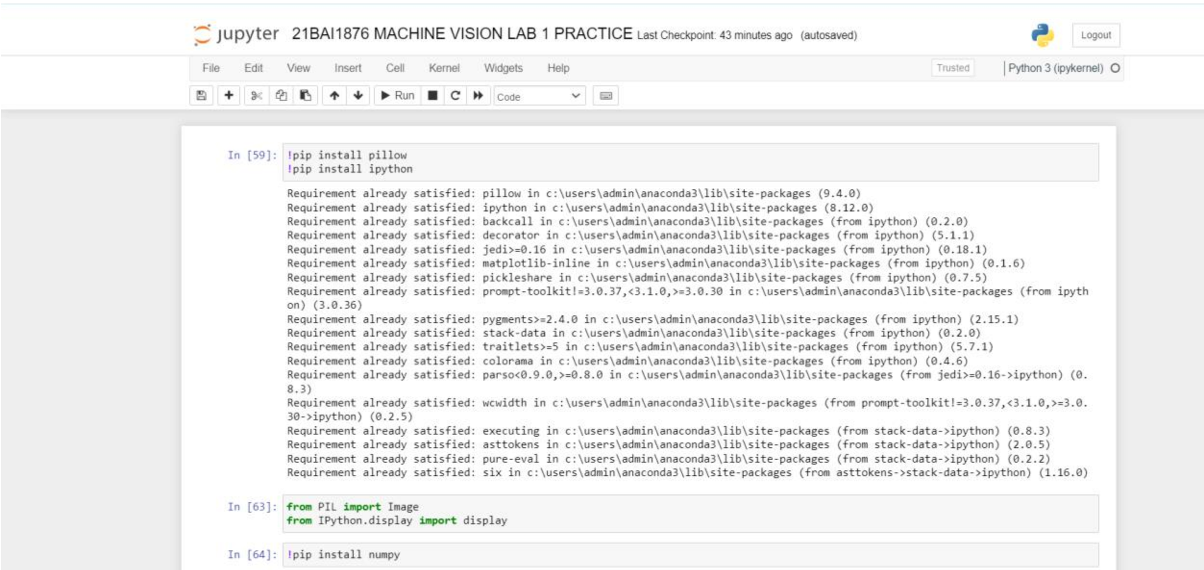
MACHINE VISION

LAB 1

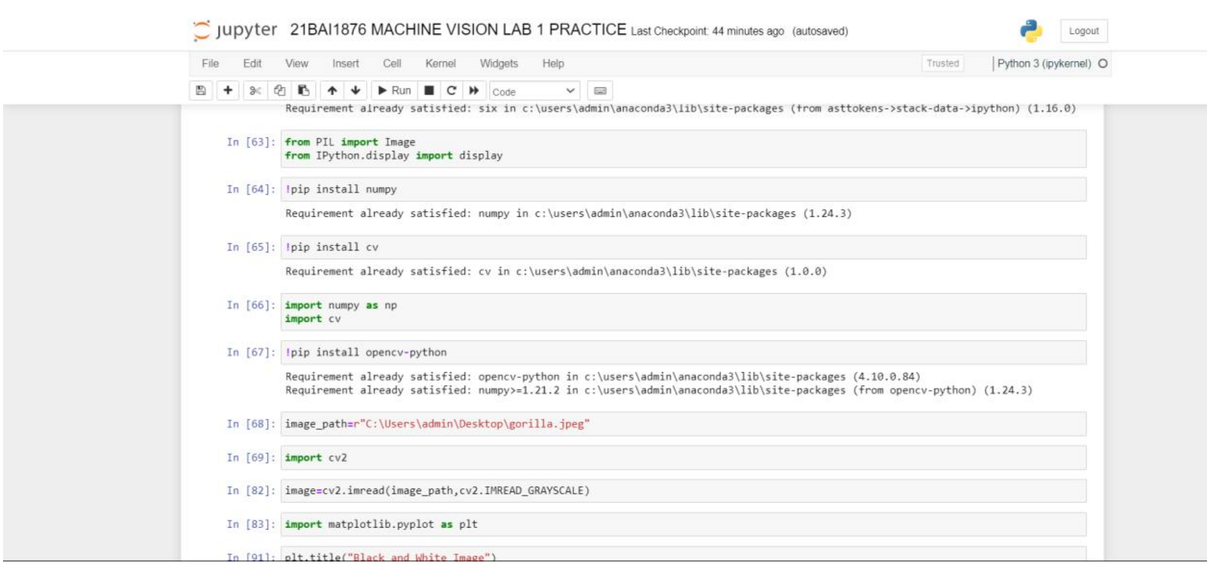
NAME : OM SUBRATO DEY

REGISTER NUMBER: 21BAI1876

1. READ A BLACK AND WHITE IMAGE AND DISPLAY THE BLACK AND WHITE PIXELS OF THE IMAGE READ RESPECTIVELY. ALSO DISPLAY THE IMAGE IN BLACK AND WHITE CHANNELS SEPERATELY.



```
jupyter 21BAI1876 MACHINE VISION LAB 1 PRACTICE Last Checkpoint: 43 minutes ago (autosaved)

In [59]: !pip install pillow
         !pip install ipython

Requirement already satisfied: pillow in c:\users\admin\anaconda3\lib\site-packages (9.4.0)
Requirement already satisfied: ipython in c:\users\admin\anaconda3\lib\site-packages (8.12.0)
Requirement already satisfied: backcall in c:\users\admin\anaconda3\lib\site-packages (from ipython) (0.2.0)
Requirement already satisfied: decorator in c:\users\admin\anaconda3\lib\site-packages (from ipython) (5.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\admin\anaconda3\lib\site-packages (from ipython) (0.18.1)
Requirement already satisfied: matplotlib-inline in c:\users\admin\anaconda3\lib\site-packages (from ipython) (0.1.6)
Requirement already satisfied: pickleshare in c:\users\admin\anaconda3\lib\site-packages (from ipython) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30 in c:\users\admin\anaconda3\lib\site-packages (from ipyth
on) (3.0.36)
Requirement already satisfied: pygments>=2.4.0 in c:\users\admin\anaconda3\lib\site-packages (from ipython) (2.15.1)
Requirement already satisfied: stack-data in c:\users\admin\anaconda3\lib\site-packages (from ipython) (0.2.0)
Requirement already satisfied: traitlets>=5 in c:\users\admin\anaconda3\lib\site-packages (from ipython) (5.7.1)
Requirement already satisfied: colorama in c:\users\admin\anaconda3\lib\site-packages (from ipython) (0.4.6)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\admin\anaconda3\lib\site-packages (from jedi>=0.16->ipython) (0.
8.3)
Requirement already satisfied: wcwidth in c:\users\admin\anaconda3\lib\site-packages (from prompt-toolkit!=3.0.37,<3.1.0,>=3.0.
30->ipython) (0.2.5)
Requirement already satisfied: executing in c:\users\admin\anaconda3\lib\site-packages (from stack-data->ipython) (0.8.3)
Requirement already satisfied: asttokens in c:\users\admin\anaconda3\lib\site-packages (from stack-data->ipython) (2.0.5)
Requirement already satisfied: pure-eval in c:\users\admin\anaconda3\lib\site-packages (from stack-data->ipython) (0.2.2)
Requirement already satisfied: six in c:\users\admin\anaconda3\lib\site-packages (from asttokens->stack-data->ipython) (1.16.0)

In [63]: from PIL import Image
         from IPython.display import display

In [64]: !pip install numpy
```



```
jupyter 21BAI1876 MACHINE VISION LAB 1 PRACTICE Last Checkpoint: 44 minutes ago (autosaved)

Requirement already satisfied: six in c:\users\admin\anaconda3\lib\site-packages (from asttokens->stack-data->ipython) (1.16.0)

In [63]: from PIL import Image
         from IPython.display import display

In [64]: !pip install numpy

Requirement already satisfied: numpy in c:\users\admin\anaconda3\lib\site-packages (1.24.3)

In [65]: !pip install cv

Requirement already satisfied: cv in c:\users\admin\anaconda3\lib\site-packages (1.0.0)

In [66]: import numpy as np
         import cv

In [67]: !pip install opencv-python

Requirement already satisfied: opencv-python in c:\users\admin\anaconda3\lib\site-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in c:\users\admin\anaconda3\lib\site-packages (from opencv-python) (1.24.3)

In [68]: image_path=r"C:\Users\admin\Desktop\gorilla.jpeg"

In [69]: import cv2

In [82]: image=cv2.imread(image_path,cv2.IMREAD_GRAYSCALE)

In [83]: import matplotlib.pyplot as plt

In [91]: plt.title("Black and White Image")
```

jupyter 21BAI1876 MACHINE VISION LAB 1 PRACTICE Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

In [83]:
```
import matplotlib.pyplot as plt
```

In [91]:
```
plt.title("Black and White Image")
plt.axis('off')
plt.show()
```

Black and White Image

In [85]:
```
black_pixels=np.argwhere(image==0)
white_pixels=np.argwhere(image==255)
```

In [86]:
```
print(f"Black pixels (total {black_pixels.shape[0]})")
print(black_pixels)
```

jupyter 21BAI1876 MACHINE VISION LAB 1 PRACTICE Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

In [85]:
```
black_pixels=np.argwhere(image==0)
white_pixels=np.argwhere(image==255)
```

In [86]:
```
print(f"Black pixels (total {black_pixels.shape[0]})")
print(black_pixels)
```


```
Black pixels (total 559)
[[  6  24]
 [ 20   6]
 [ 26  42]
 ...
 [220 219]
 [221  34]
 [223 206]]
```

In [87]:
```
print(f"\n white pixels (total {white_pixels.shape[0]})")
print(white_pixels)
```

```
 [ 94 202]
 [ 95 173]
 [ 95 202]
 [ 96 174]
 [ 96 201]
 [ 97 174]
 [ 97 201]
 [ 98 175]
 [ 98 202]
 [ 99 175]
 [ 99 202]
```

jupyter 21BAI1876 MACHINE VISION LAB 1 PRACTICE Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted | Python 3 (ipykernel)

```
 [102 202]
 [103 173]
 [103 202]
```

In [77]:
```
ig=Image.open('gorilla.jpeg')
```

In [78]:
```
ig
```

Out[78]:

In [92]:
```
# Threshold the image to separate black and white pixels
_, black_channel = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY_INV)
_, white_channel = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

# Display the original image
plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')
```

jupyter 21BAI1876 MACHINE VISION LAB 1 PRACTICE Last Checkpoint: an hour ago (autosaved)

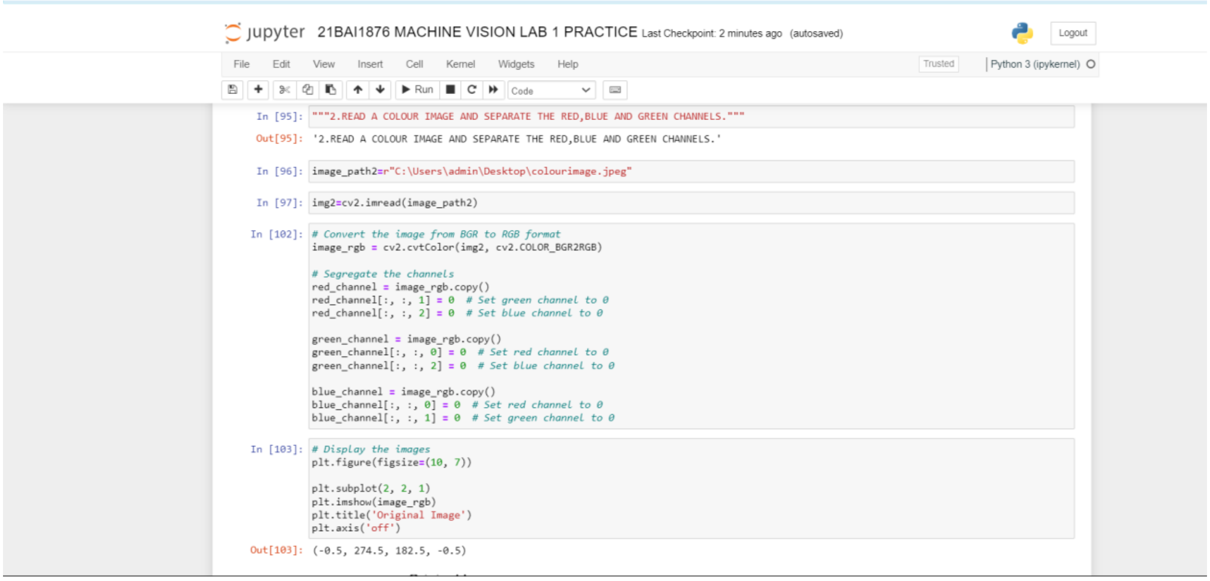
Logout

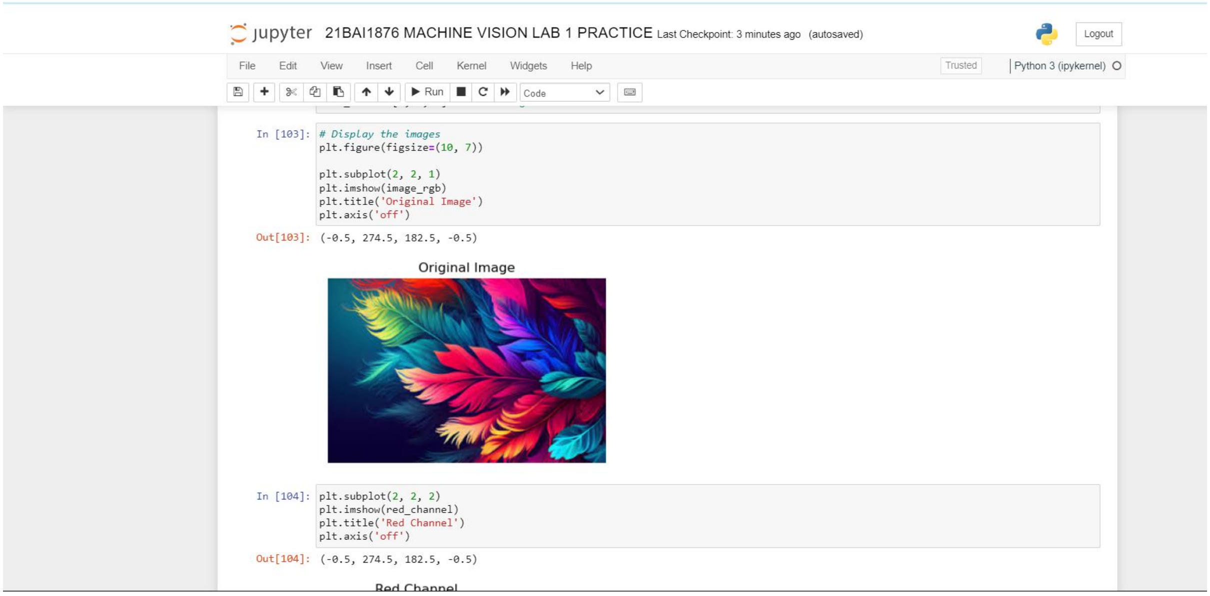
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

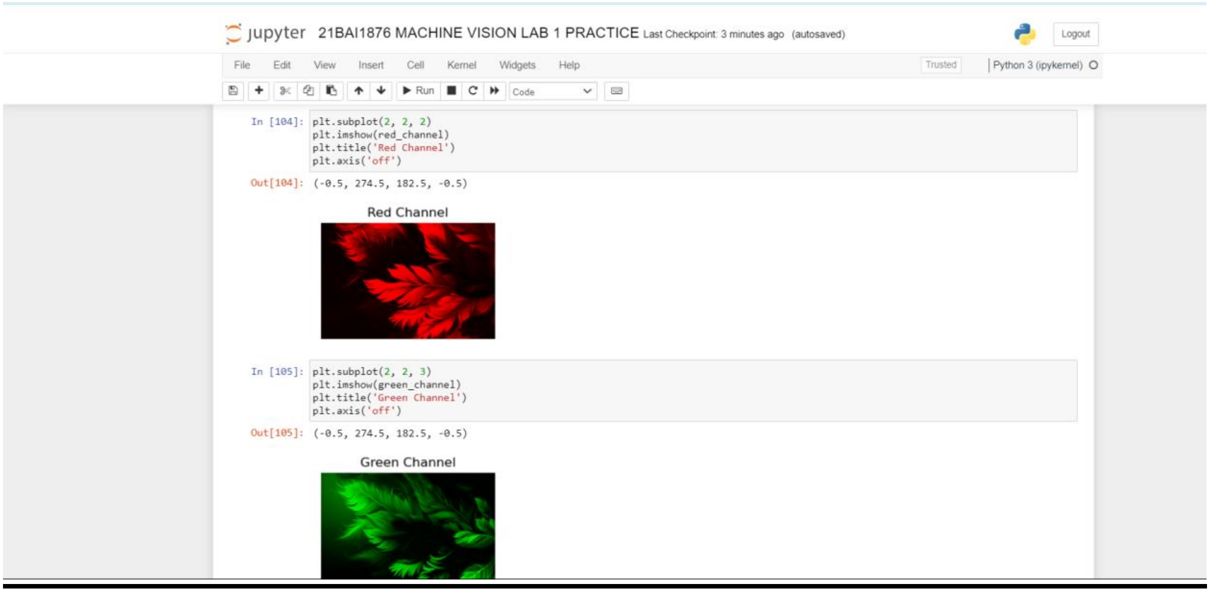
In [92]:

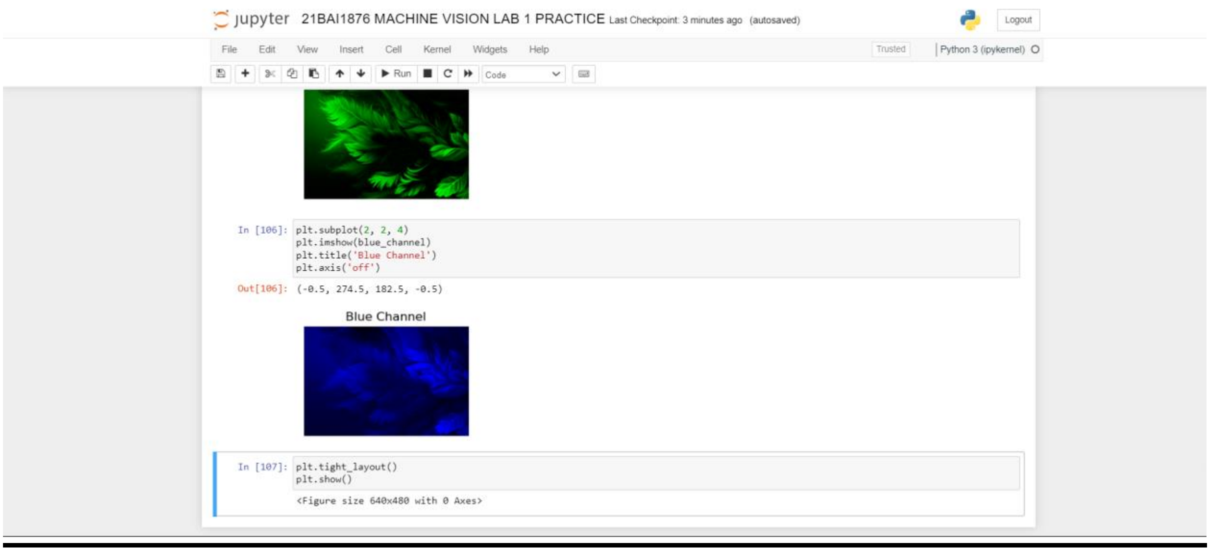
```
# Threshold the image to separate black and white pixels
_, black_channel = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY_INV)
_, white_channel = cv2.threshold(image, 127, 255, cv2.THRESH_BINARY)

# Display the original image
plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')

# Display the black channel
plt.subplot(1, 3, 2)
plt.imshow(black_channel, cmap='gray')
plt.title('Black Channel')
plt.axis('off')

# Display the white channel
plt.subplot(1, 3, 3)
plt.imshow(white_channel, cmap='gray')
plt.title('White Channel')
plt.axis('off')

plt.show()
```

2. READ A COLOUR IMAGE AND SEPARATE THE RED,BLUE AND GREEN CHANNELS.






jupyter 21BAI1876 MACHINE VISION LAB 1 PRACTICE Last Checkpoint: 3 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [104]: plt.subplot(2, 2, 2)
plt.imshow(red_channel)
plt.title('Red Channel')
plt.axis('off')
Out[104]: (-0.5, 274.5, 182.5, -0.5)
Red Channel
In [105]: plt.subplot(2, 2, 3)
plt.imshow(green_channel)
plt.title('Green Channel')
plt.axis('off')
Out[105]: (-0.5, 274.5, 182.5, -0.5)
Green Channel


jupyter 21BAI1876 MACHINE VISION LAB 1 PRACTICE Last Checkpoint: 3 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [106]: plt.subplot(2, 2, 4)
plt.imshow(blue_channel)
plt.title('Blue Channel')
plt.axis('off')
Out[106]: (-0.5, 274.5, 182.5, -0.5)
Blue Channel
In [107]: plt.tight_layout()
plt.show()
<Figure size 640x480 with 0 Axes>

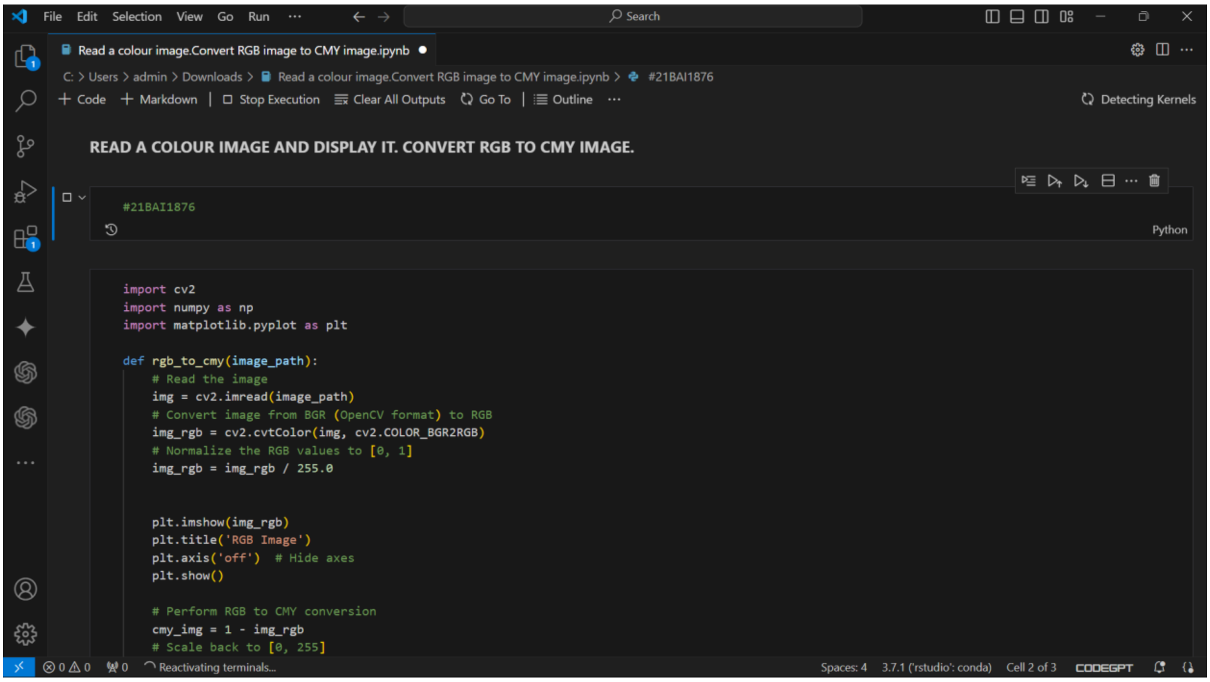
3. CONVERT A RGB IMAGE TO CMY IMAGE WHICH WILL BE THE NEGATIVE OF THE ORIGINAL IMAGE

CODE:

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

def rgb_to_cmy(image_path):
    # Read the image
    img = cv2.imread(image_path)
    # Convert image from BGR (OpenCV format) to RGB
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    # Normalize the RGB values to [0, 1]
    img_rgb = img_rgb / 255.0



    plt.imshow(img_rgb)
    plt.title('RGB Image')
    plt.axis('off')  # Hide axes
    plt.show()

    # Perform RGB to CMY conversion
    cmy_img = 1 - img_rgb
    # Scale back to [0, 255]
    cmy_img = (cmy_img * 255).astype(np.uint8)
    return cmy_img
```

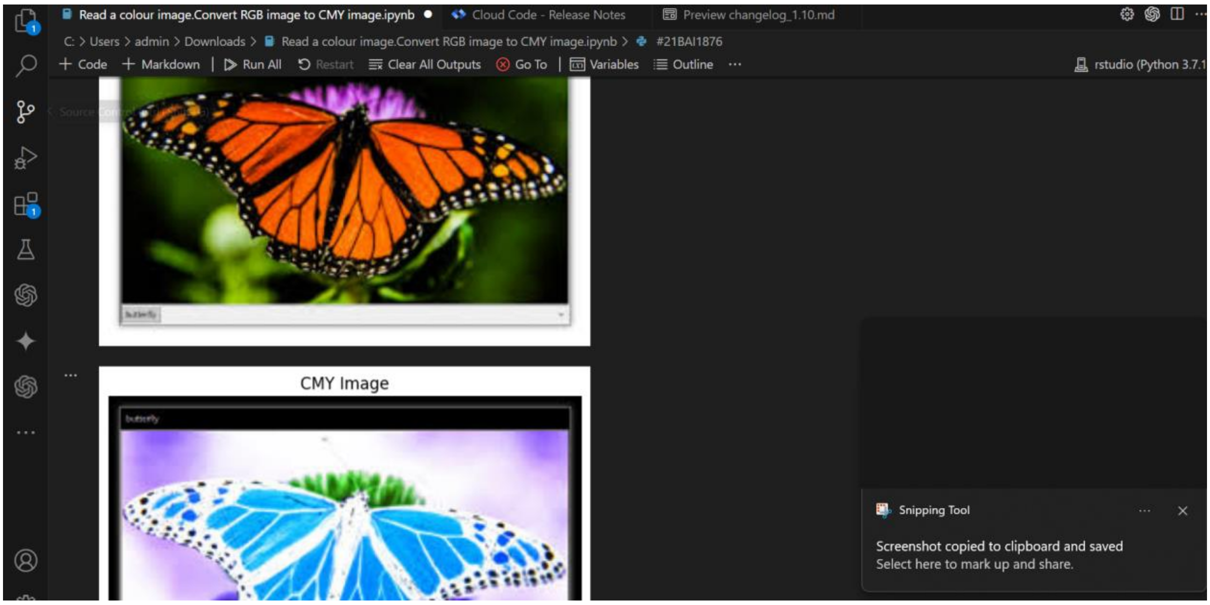
```python
# Example usage
image_path = '/content/moth.jpeg'
cmy_image = rgb_to_cmy(image_path)

# Display the CMY image using Matplotlib
plt.imshow(cmy_image)
plt.title('CMY Image')
plt.axis('off')  # Hide axes
plt.show()

# Save the CMY image
cv2.imwrite('/content/moth_cmy.jpg', cv2.cvtColor(cmy_image,
cv2.COLOR_RGB2BGR))
```

OUTPUT:



```python
        # Normalize the RGB values to [0, 1]
        img_rgb = img_rgb / 255.0


        plt.imshow(img_rgb)
        plt.title('RGB Image')
        plt.axis('off')  # Hide axes
        plt.show()

        # Perform RGB to CMY conversion
        cmy_img = 1 - img_rgb
        # Scale back to [0, 255]
        cmy_img = (cmy_img * 255).astype(np.uint8)
        return cmy_img

    # Example usage
    image_path = '/content/moth.jpeg'
    cmy_image = rgb_to_cmy(image_path)

    # Display the CMY image using Matplotlib
    plt.imshow(cmy_image)
    plt.title('CMY Image')
    plt.axis('off')  # Hide axes
    plt.show()

    # Save the CMY image
    cv2.imwrite('/content/moth_cmy.jpg', cv2.cvtColor(cmy_image, cv2.COLOR_RGB2BGR))
```

[13] Python



4. And 5. CONVERT A RGB IMAGE TO HSI IMAGE AND THEN BACK TO RGB

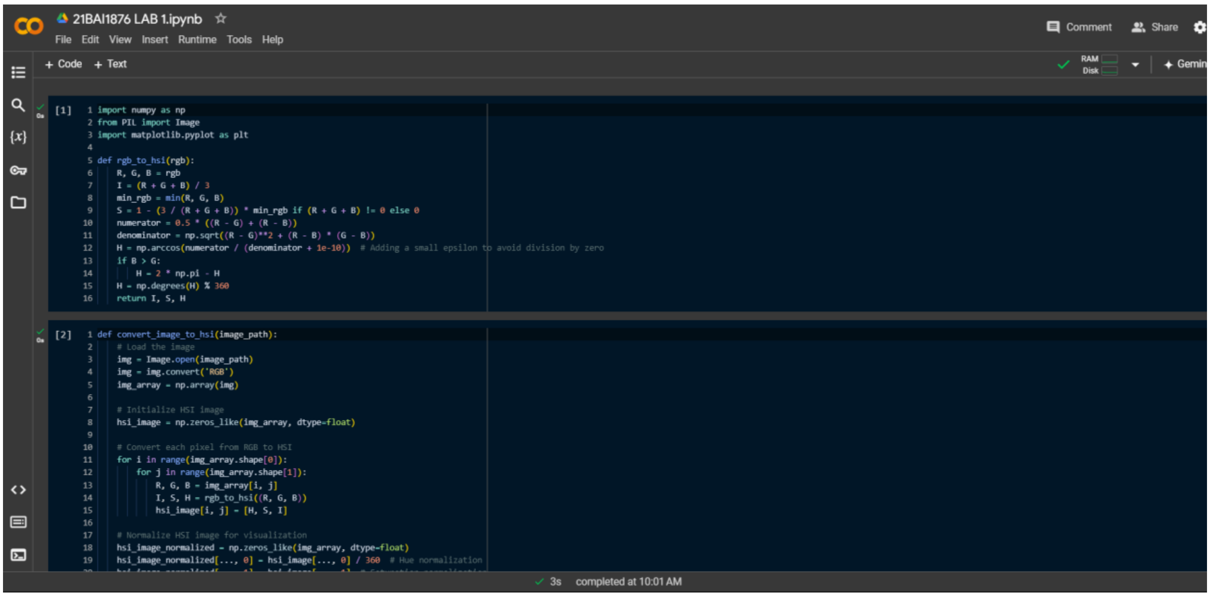
CODE:

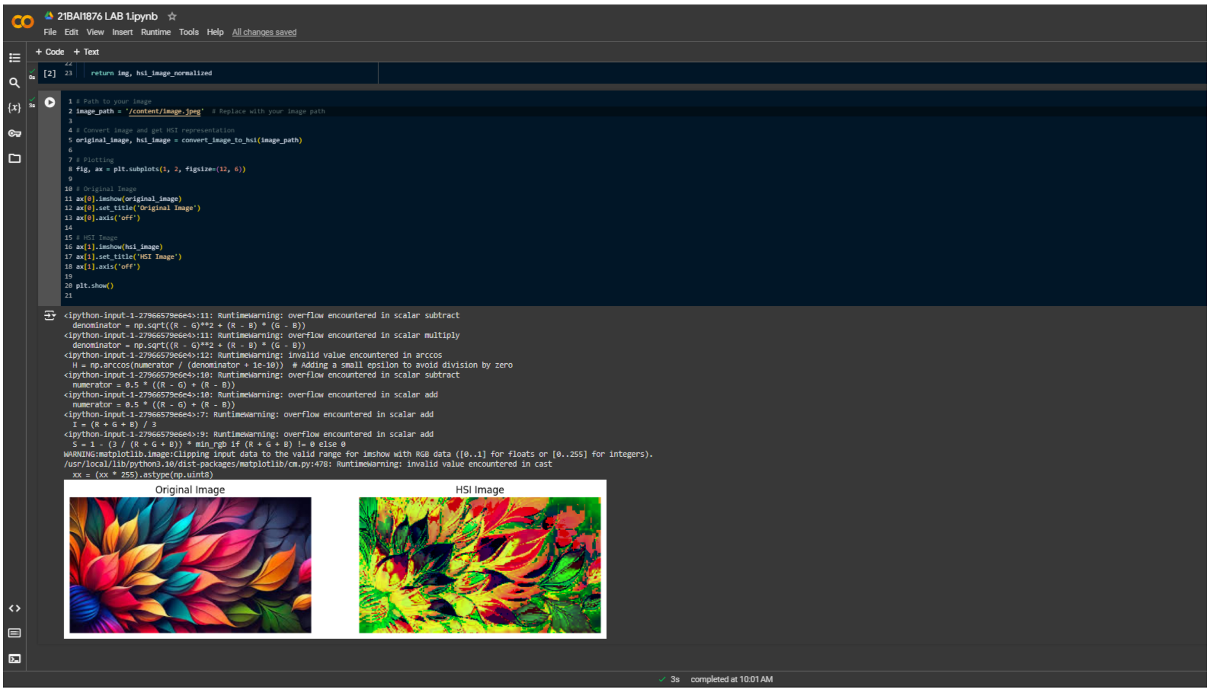
```
import numpy as np

from PIL import Image

import matplotlib.pyplot as plt


def rgb_to_hsi(rgb):

    R, G, B = rgb

    I = (R + G + B) / 3

    min_rgb = min(R, G, B)

    S = 1 - (3 / (R + G + B)) * min_rgb if (R + G + B) != 0 else
0

    numerator = 0.5 * ((R - G) + (R - B))

    denominator = np.sqrt((R - G)**2 + (R - B) * (G - B))

    H = np.arccos(numerator / (denominator + 1e-10))  # Adding a
small epsilon to avoid division by zero

    if B > G:

        H = 2 * np.pi - H

    H = np.degrees(H) % 360

    return I, S, H


def convert_image_to_hsi(image_path):

    # Load the image

    img = Image.open(image_path)

    img = img.convert('RGB')

    img_array = np.array(img)


    # Initialize HSI image
```

```
    hsi_image = np.zeros_like(img_array, dtype=float)

    # Convert each pixel from RGB to HSI
    for i in range(img_array.shape[0]):
        for j in range(img_array.shape[1]):
            R, G, B = img_array[i, j]
            I, S, H = rgb_to_hsi((R, G, B))
            hsi_image[i, j] = [H, S, I]

    # Normalize HSI image for visualization
    hsi_image_normalized = np.zeros_like(img_array, dtype=float)
    hsi_image_normalized[..., 0] = hsi_image[..., 0] / 360  # 
Hue normalization
    hsi_image_normalized[..., 1] = hsi_image[..., 1]  # 
Saturation normalization
    hsi_image_normalized[..., 2] = hsi_image[..., 2] / 255  # 
Intensity normalization
    return img, hsi_image_normalized

# Path to your image
image_path = '/content/moth.jpeg'  # Replace with your image 
path

# Convert image and get HSI representation
original_image, hsi_image = convert_image_to_hsi(image_path)

# Plotting
fig, ax = plt.subplots(1, 2, figsize=(12, 6))

# Original Image
ax[0].imshow(original_image)
ax[0].set_title('Original Image')
```

```
ax[0].axis('off')
# HSI Image
ax[1].imshow(hsi_image)
ax[1].set_title('HSI Image')
ax[1].axis('off')
plt.show()
```

OUTPUT:





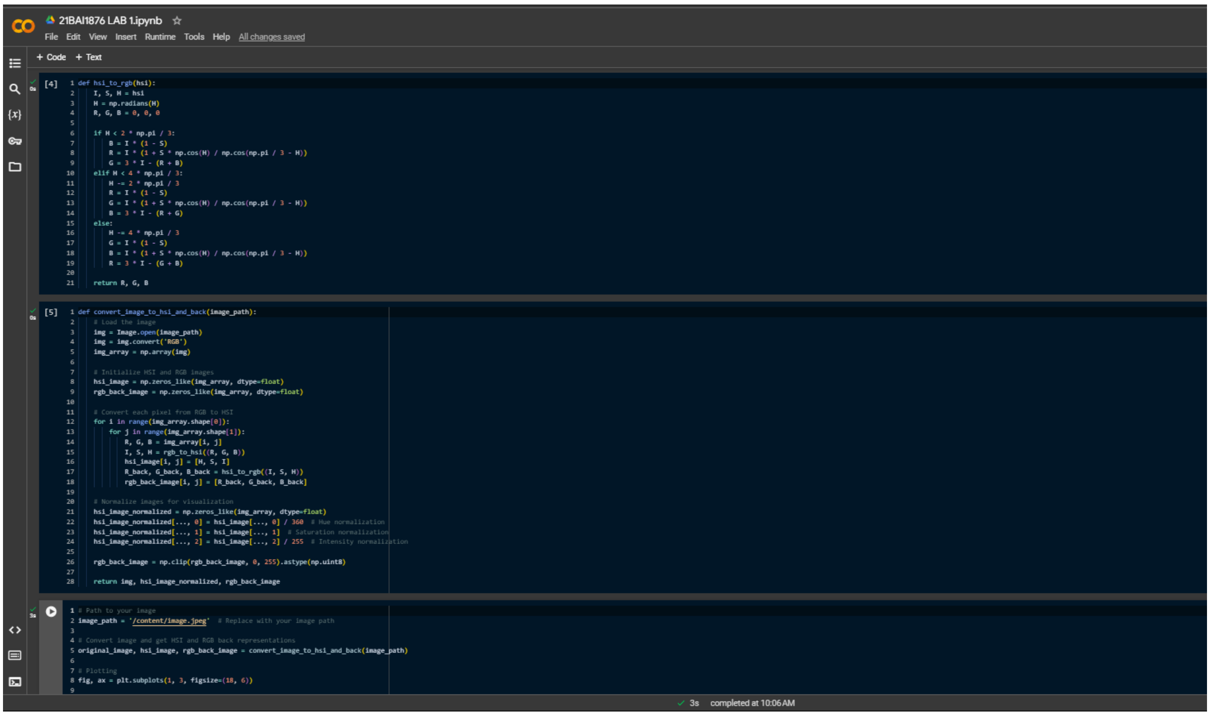
CODE FOR HIS TO RGB:

```
def hsi_to_rgb(hsi):
    I, S, H = hsi
    H = np.radians(H)
    R, G, B = 0, 0, 0

    if H < 2 * np.pi / 3:
        B = I * (1 - S)
        R = I * (1 + S * np.cos(H) / np.cos(np.pi / 3 - H))
        G = 3 * I - (R + B)
    elif H < 4 * np.pi / 3:
        H -= 2 * np.pi / 3
        R = I * (1 - S)
        G = I * (1 + S * np.cos(H) / np.cos(np.pi / 3 - H))
        B = 3 * I - (R + G)
    else:
        H -= 4 * np.pi / 3
        G = I * (1 - S)
        B = I * (1 + S * np.cos(H) / np.cos(np.pi / 3 - H))
        R = 3 * I - (G + B)

    return R, G, B

def convert_image_to_hsi_and_back(image_path):
    # Load the image
    img = Image.open(image_path)
    img = img.convert('RGB')
    img_array = np.array(img)
```

```
    # Initialize HSI and RGB images
    hsi_image = np.zeros_like(img_array, dtype=float)
    rgb_back_image = np.zeros_like(img_array, dtype=float)

    # Convert each pixel from RGB to HSI
    for i in range(img_array.shape[0]):
        for j in range(img_array.shape[1]):
            R, G, B = img_array[i, j]
            I, S, H = rgb_to_hsi((R, G, B))
            hsi_image[i, j] = [H, S, I]
            R_back, G_back, B_back = hsi_to_rgb((I, S, H))
            rgb_back_image[i, j] = [R_back, G_back, B_back]

    # Normalize images for visualization
    hsi_image_normalized = np.zeros_like(img_array, dtype=float)
    hsi_image_normalized[..., 0] = hsi_image[..., 0] / 360  #
Hue normalization
    hsi_image_normalized[..., 1] = hsi_image[..., 1]  #
Saturation normalization
    hsi_image_normalized[..., 2] = hsi_image[..., 2] / 255  #
Intensity normalization


    rgb_back_image = np.clip(rgb_back_image, 0,
255).astype(np.uint8)
    return img, hsi_image_normalized, rgb_back_image


# Path to your image
image_path = '/content/moth.jpeg'  # Replace with your image
path
```

```
# Convert image and get HSI and RGB back representations
original_image, hsi_image, rgb_back_image =
convert_image_to_hsi_and_back(image_path)


# Plotting
fig, ax = plt.subplots(1, 3, figsize=(18, 6))


# Original Image
ax[0].imshow(original_image)
ax[0].set_title('Original Image')
ax[0].axis('off')


# HSI Image
ax[1].imshow(hsi_image)
ax[1].set_title('HSI Image')
ax[1].axis('off')


# RGB Back Image
ax[2].imshow(rgb_back_image)
ax[2].set_title('RGB Back Image')
ax[2].axis('off')


plt.show()
```

OUTPUT:



21BAI1876 LAB 1.ipynb

File Edit View Insert Runtime Tools Help All changes saved

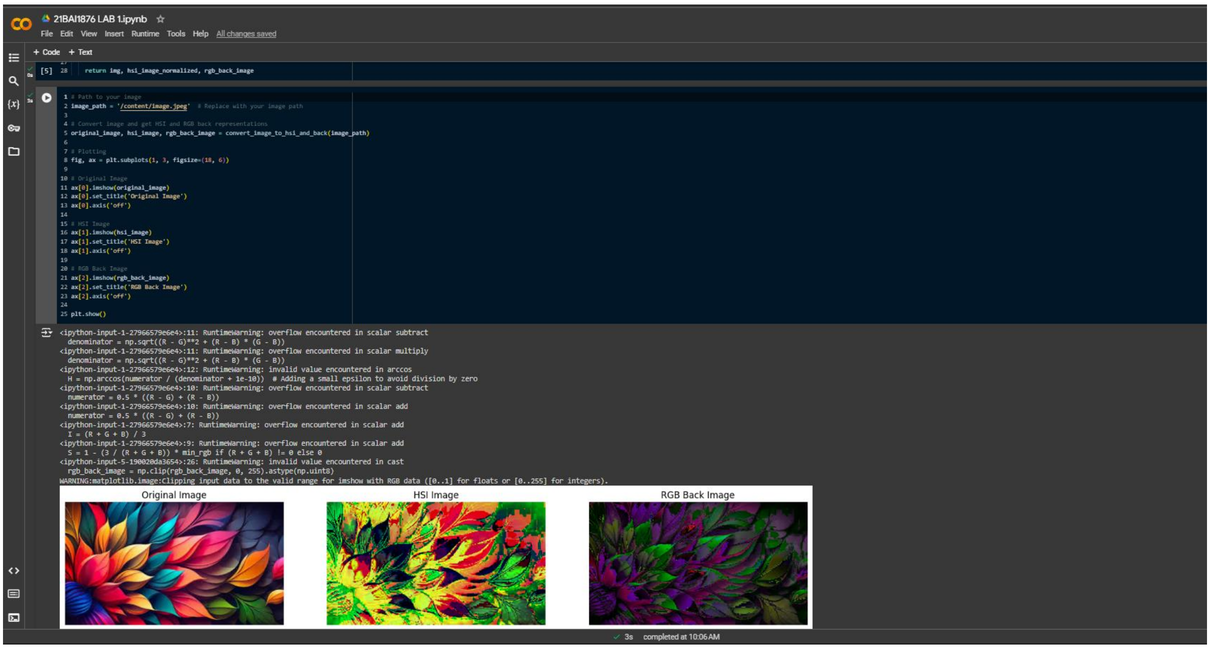
+ Code + Text

[4]
```python
def hsi_to_rgb(hsi):
    I, S, H = hsi
    H = np.radians(H)
    R, G, B = 0, 0, 0

    if H < 2 * np.pi / 3:
        B = I * (1 - S)
        R = I * (1 + S * np.cos(H) / np.cos(np.pi / 3 - H))
        G = 3 * I - (R + B)
    elif H < 4 * np.pi / 3:
        H -= 2 * np.pi / 3
        R = I * (1 - S)
        G = I * (1 + S * np.cos(H) / np.cos(np.pi / 3 - H))
        B = 3 * I - (R + G)
    else:
        H -= 4 * np.pi / 3
        G = I * (1 - S)
        B = I * (1 + S * np.cos(H) / np.cos(np.pi / 3 - H))
        R = 3 * I - (G + B)

    return R, G, B
```

[5]
```python
def convert_image_to_hsi_and_back(image_path):
    # Load the image
    img = Image.open(image_path)
    img = img.convert('RGB')
    img_array = np.array(img)

    # Initialize HSI and RGB images
    hsi_image = np.zeros_like(img_array, dtype=float)
    rgb_back_image = np.zeros_like(img_array, dtype=float)

    # Convert each pixel from RGB to HSI
    for i in range(img_array.shape[0]):
        for j in range(img_array.shape[1]):
            R, G, B = img_array[i, j]
            I, S, H = rgb_to_hsi((R, G, B))
            hsi_image[i, j] = [H, S, I]
            R_back, G_back, B_back = hsi_to_rgb((I, S, H))
            rgb_back_image[i, j] = [R_back, G_back, B_back]

    # Normalize images for visualization
    hsi_image_normalized = np.zeros_like(img_array, dtype=float)
    hsi_image_normalized[..., 0] = hsi_image[..., 0] / 360  # Hue normalization
    hsi_image_normalized[..., 1] = hsi_image[..., 1]  # Saturation normalization
    hsi_image_normalized[..., 2] = hsi_image[..., 2] / 255  # Intensity normalization

    rgb_back_image = np.clip(rgb_back_image, 0, 255).astype(np.uint8)

    return img, hsi_image_normalized, rgb_back_image
```

```python
# Path to your image
image_path = '/content/image.jpeg'  # Replace with your image path

# Convert image and get HSI and RGB back representations
original_image, hsi_image, rgb_back_image = convert_image_to_hsi_and_back(image_path)

# Plotting
fig, ax = plt.subplots(1, 3, figsize=(18, 6))

```

3s completed at 10:06 AM



21BAI1876 LAB 1.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[5]
```python
    return img, hsi_image_normalized, rgb_back_image
```

```python
# Path to your image
image_path = '/content/image.jpeg'  # Replace with your image path

# Convert image and get HSI and RGB back representations
original_image, hsi_image, rgb_back_image = convert_image_to_hsi_and_back(image_path)

# Plotting
fig, ax = plt.subplots(1, 3, figsize=(18, 6))

# Original Image
ax[0].imshow(original_image)
ax[0].set_title('Original Image')
ax[0].axis('off')

# HSI Image
ax[1].imshow(hsi_image)
ax[1].set_title('HSI Image')
ax[1].axis('off')

# RGB Back Image
ax[2].imshow(rgb_back_image)
ax[2].set_title('RGB Back Image')
ax[2].axis('off')

plt.show()
```

```
<ipython-input-1-27966579e6e4>:11: RuntimeWarning: overflow encountered in scalar subtract
  denominator = np.sqrt((R - G)**2 + (R - B) * (G - B))
<ipython-input-1-27966579e6e4>:11: RuntimeWarning: overflow encountered in scalar multiply
  denominator = np.sqrt((R - G)**2 + (R - B) * (G - B))
<ipython-input-1-27966579e6e4>:12: RuntimeWarning: invalid value encountered in arccos
  H = np.arccos(numerator / (denominator + 1e-10))  # Adding a small epsilon to avoid division by zero
<ipython-input-1-27966579e6e4>:10: RuntimeWarning: overflow encountered in scalar subtract
  numerator = 0.5 * ((R - G) + (R - B))
<ipython-input-1-27966579e6e4>:10: RuntimeWarning: overflow encountered in scalar add
  numerator = 0.5 * ((R - G) + (R - B))
<ipython-input-1-27966579e6e4>:7: RuntimeWarning: overflow encountered in scalar add
  I = (R + G + B) / 3
<ipython-input-1-27966579e6e4>:9: RuntimeWarning: overflow encountered in scalar add
  S = 1 - (3 / (R + G + B)) * min_rgb if (R + G + B) != 0 else 0
<ipython-input-5-190020da3654>:26: RuntimeWarning: invalid value encountered in cast
  rgb_back_image = np.clip(rgb_back_image, 0, 255).astype(np.uint8)
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```


3s completed at 10:06 AM