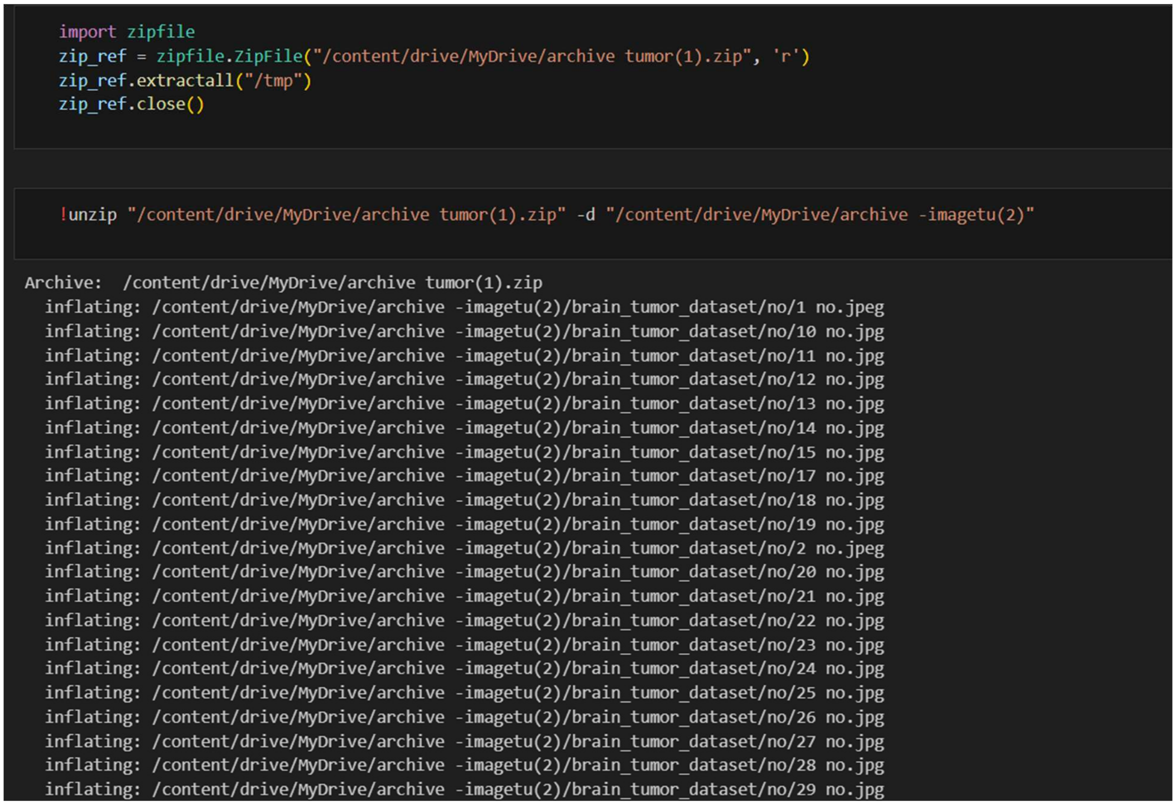
Hello everyone! My name is Rittika Jana and it's my first time using GitHub and tomorrow is my last date of submission of my project work. I have to show some of my works and provide a URL. Through it's not a super great work but it can help many beginners to learn. I will further know more about this and upload the ipynb file (which I am not being able to do due to lack of information and size issue). Thank you…….



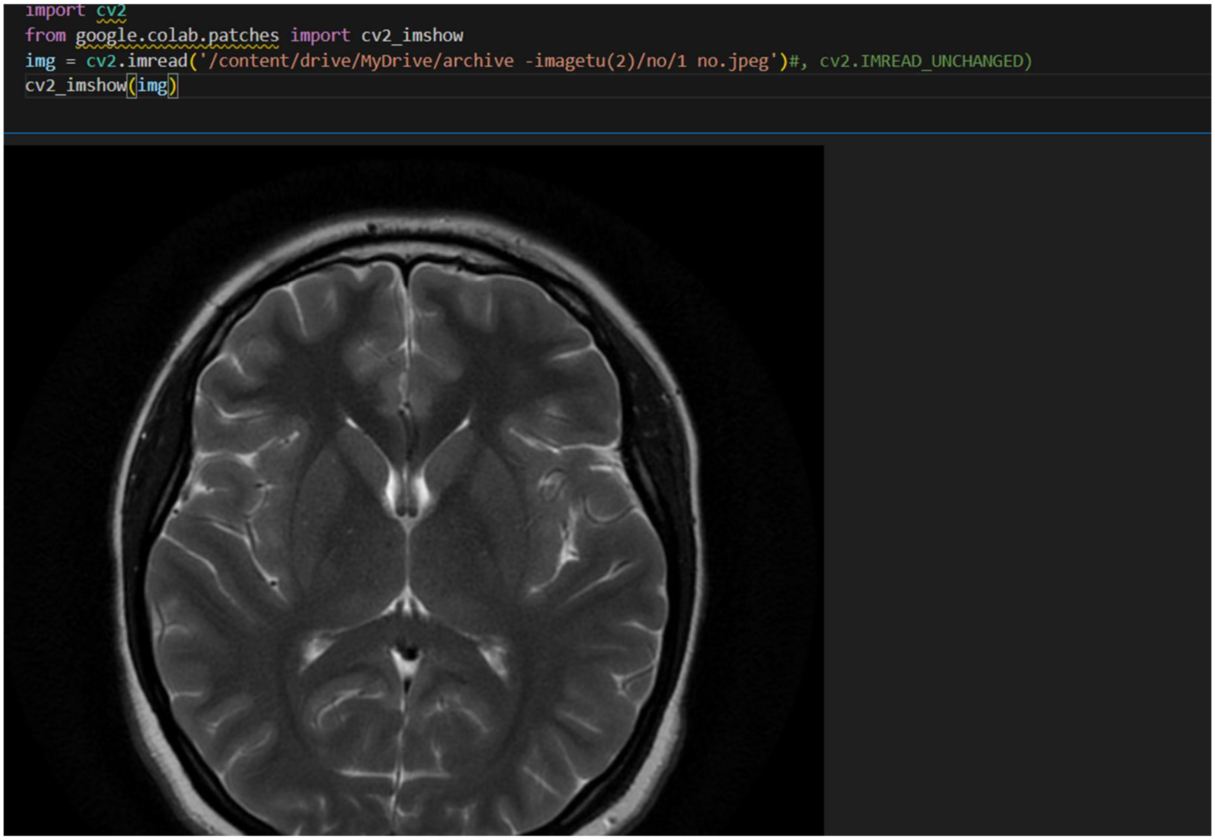
```
import zipfile
zip_ref = zipfile.ZipFile("/content/drive/MyDrive/archive tumor(1).zip", 'r')
zip_ref.extractall("/tmp")
zip_ref.close()
```

```
!unzip "/content/drive/MyDrive/archive tumor(1).zip" -d "/content/drive/MyDrive/archive -imagetu(2)"
```

```
Archive:  /content/drive/MyDrive/archive tumor(1).zip
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/1 no.jpeg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/10 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/11 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/12 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/13 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/14 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/15 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/17 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/18 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/19 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/2 no.jpeg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/20 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/21 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/22 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/23 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/24 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/25 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/26 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/27 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/28 no.jpg
  inflating: /content/drive/MyDrive/archive -imagetu(2)/brain_tumor_dataset/no/29 no.jpg
```

Next step:

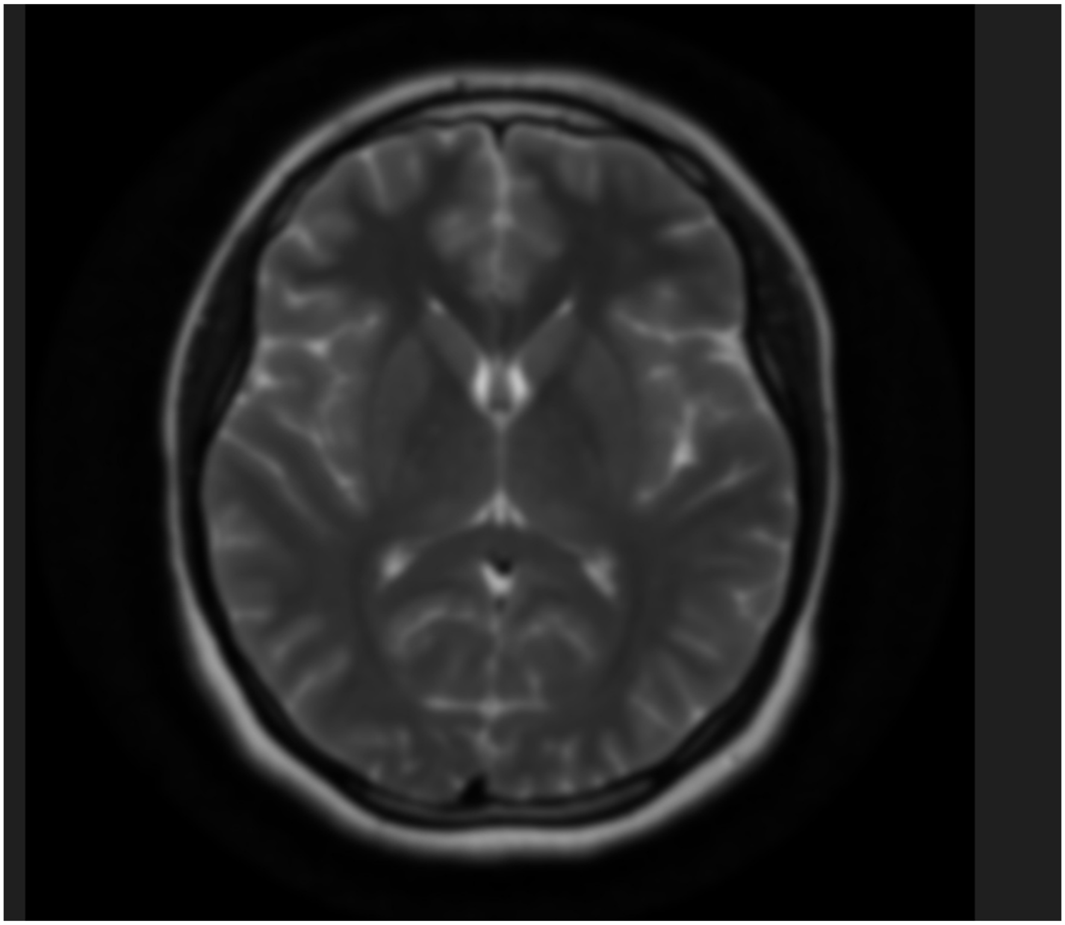
```
import cv2

import os


def load_images_from_folder(flowers):

  images = []

  for filename in os.listdir(flowers):

    img = cv2.imread(os.path.join("/content/drive/MyDrive/archive -imagetu(2)/no","/content/drive/MyDrive/archive -imagetu(2)/no/1 no.jpeg"))

    if img is not None:

      images.append(img)

  return images
```



```
import cv2
from google.colab.patches import cv2_imshow
img = cv2.imread('/content/drive/MyDrive/archive -imagetu(2)/no/1 no.jpeg')#, cv2.IMREAD_UNCHANGED)
cv2_imshow(img)
```

Next step:

```
import cv2

import numpy as np


image = cv2.imread("/content/drive/MyDrive/archive -imagetu(2)/no/1 no.jpeg")

blurred_image = cv2.GaussianBlur(image, (17, 17), 0)


# Save or display the blurred image

cv2.imwrite("blurred_image.jpg", blurred_image)

cv2_imshow(blurred_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```



Next:

```
from PIL import Image, ImageFilter
import os


# Path to the folder containing images
folder_path = '/content/drive/MyDrive/archive -imagetu(2)/no'
output_folder = '/content/drive/MyDrive/archive -imagetu(2)/blurred_images(NO)'


if not os.path.exists(output_folder):
    os.makedirs(output_folder)
```

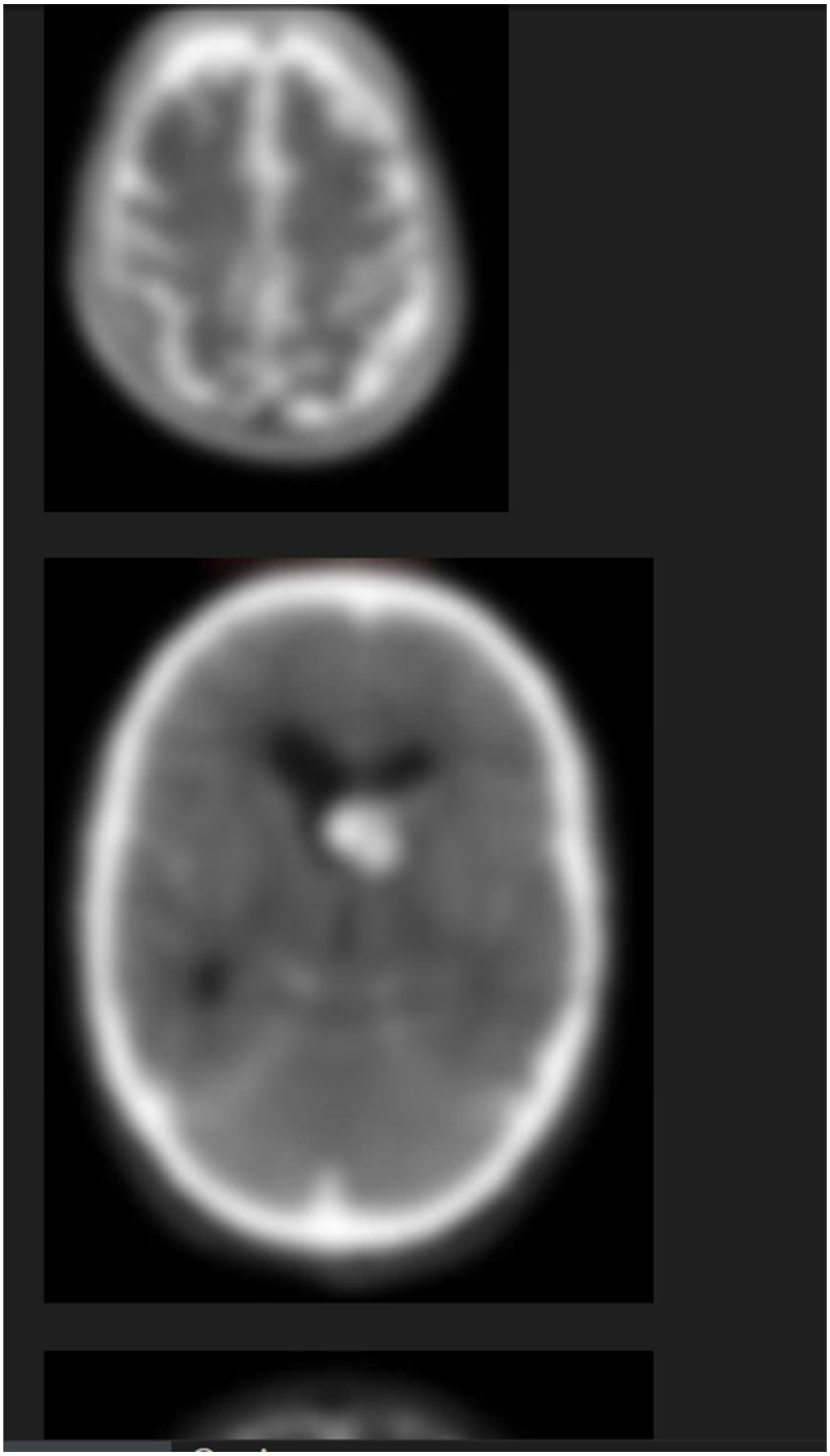
```
# Loop through all files in the folder
for filename in os.listdir(folder_path):
    # Check if the file is an image (you can extend this condition to check for other formats)
    if filename.endswith(('.png', '.jpg', '.jpeg','.JPG')):
        # Full path to the image file
        img_path = os.path.join(folder_path, filename)

        # Open the image
        with Image.open(img_path) as img:
            # Apply Gaussian blur
            blurred_img = img.filter(ImageFilter.GaussianBlur(radius=5))  # Adjust radius for desired blur strength

            # Save the blurred image (you can overwrite or save to a new folder)
            blurred_img.save(os.path.join(output_folder, 'blurred_' + filename))

        print(f"Processed {filename}")
```

Output:

```
Processed N26.JPG
Processed N1.JPG
Processed N22.JPG
Processed 48 no.jpeg
Processed no 91.jpeg
Processed N19.JPG
Processed N2.JPG
Processed no 7.jpeg
Processed no 5.jpeg
Processed 1 no.jpeg
Processed 2 no.jpeg
Processed N20.JPG
```

Next:

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

fol_name = '/content/drive/MyDrive/archive -
imagetu(2)/blurred_images(NO)'

# Iterate over files in the folder
for filename in os.listdir(fol_name):
  # Construct the full path to the image
  img_path = os.path.join(fol_name, filename)

  # Read the image
  img = cv2.imread(img_path)

  # Display the image using cv2_imshow
  if img is not None:  # Check if image was loaded successfully
    cv2_imshow(img)
  else:
    print(f"Could not load image: {img_path}")
```



Next:

```
from PIL import Image, ImageFilter
```

```
import os


# Path to the folder containing images
folder_path = '/content/drive/MyDrive/archive -imagetu(2)/yes'
output_folder = '/content/drive/MyDrive/archive -
imagetu(2)/blurred_images(YES)'


if not os.path.exists(output_folder):
    os.makedirs(output_folder)


# Loop through all files in the folder
for filename in os.listdir(folder_path):
    # Check if the file is an image (you can extend this condition to check for
other formats)
    if filename.endswith(('.png', '.jpg', '.jpeg','.JPG')):
        # Full path to the image file
        img_path = os.path.join(folder_path, filename)


        # Open the image
        with Image.open(img_path) as img:
            # Apply Gaussian blur
            blurred_img = img.filter(ImageFilter.GaussianBlur(radius=5))  # Adjust
radius for desired blur strength


            # Save the blurred image (you can overwrite or save to a new folder)
            blurred_img.save(os.path.join(output_folder, 'blurred_' + filename))
        print(f"Processed {filename}")
```

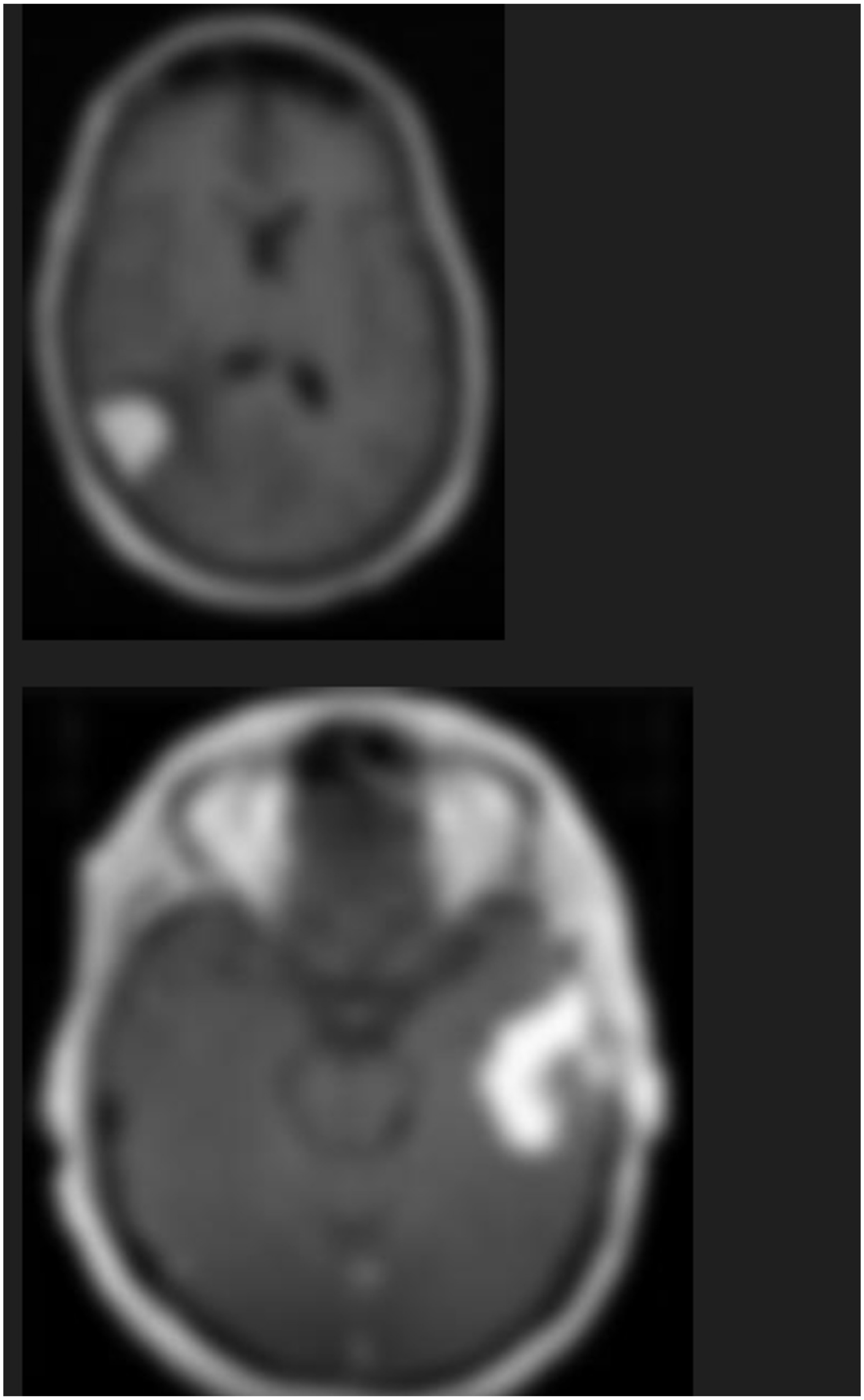
Output:

```
Processed Y165.JPG
Processed Y155.JPG
Processed Y109.JPG
Processed Y184.JPG
Processed Y160.JPG
Processed Y193.JPG
Processed Y117.JPG
Processed Y146.JPG
Processed Y120.JPG
Processed Y166.JPG
Processed Y115.JPG
Processed Y111.JPG
Processed Y161.JPG
Processed Y159.JPG
Processed Y164.JPG
Processed Y192.JPG
Processed Y100.JPG
Processed Y195.JPG
Processed Y167.JPG
Processed Y158.JPG
Processed Y182.JPG
Processed Y170.JPG
Processed Y163.JPG
Processed Y113.JPG
Processed Y19.JPG
...
Processed Y255.JPG
Processed Y75.JPG
Processed Y71.JPG
Processed Y47.JPG
```

Next:

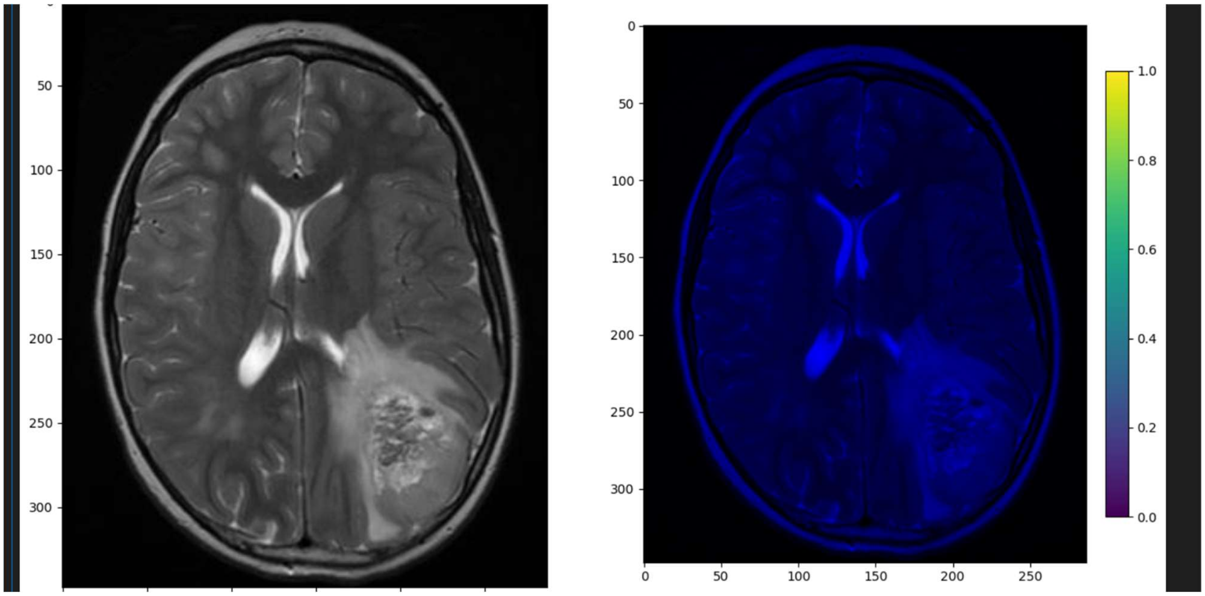
```
import cv2
import os
from google.colab.patches import cv2_imshow

fol_name = '/content/drive/MyDrive/archive - imagetu(2)/blurred_images(YES)'

# Iterate over files in the folder
for filename in os.listdir(fol_name):
  # Construct the full path to the image
  img_path = os.path.join(fol_name, filename)

  # Read the image
  img = cv2.imread(img_path)

  # Display the image using cv2_imshow
  if img is not None:  # Check if image was loaded successfully
    cv2_imshow(img)
```

```
else:
  print(f"Could not load image: {img_path}")
```



Next:

```
# Importing Necessary Libraries
from skimage import data, io
from skimage.color import rgb2hsv
import matplotlib.pyplot as plt

# Setting the plot size to 15,15
plt.figure(figsize=(15, 15))

# Load your image using io.imread
coffee = io.imread('/content/drive/MyDrive/archive - imagetu(2)/yes/Y100.JPG')
plt.subplot(1, 2, 1)

# Displaying the sample image
plt.imshow(coffee)

# Converting RGB Image to HSV Image
hsv_coffee = rgb2hsv(coffee) # Pass the image to rgb2hsv
plt.subplot(1, 2, 2)

# Displaying the sample image - HSV Format
hsv_coffee_colorbar = plt.imshow(hsv_coffee)

# Adjusting colorbar to fit the size of the image
plt.colorbar(hsv_coffee_colorbar, fraction=0.046, pad=0.04)
```



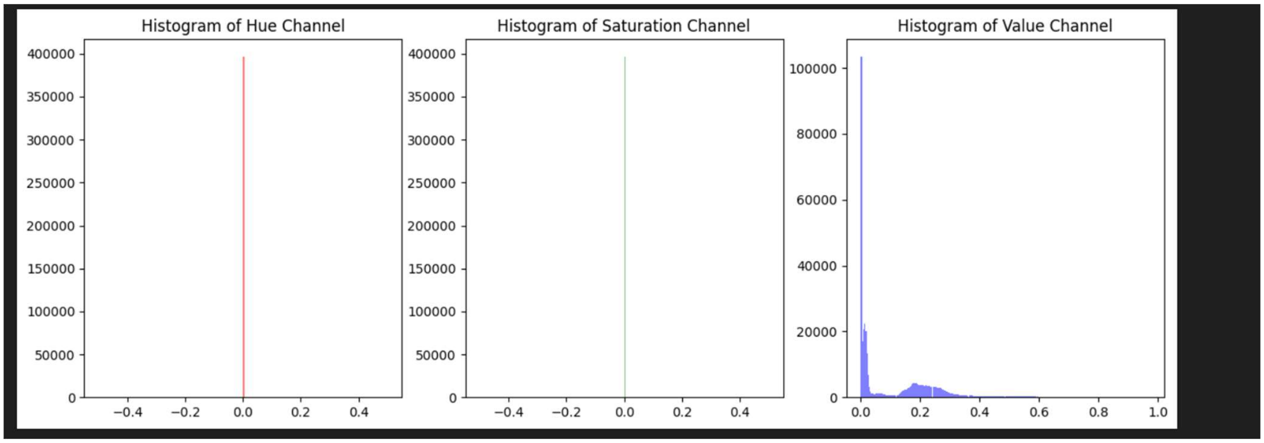
Then we will do it for The whole folder so the code will be

```
# Function to process and display images in a folder
def process_images_in_folder(folder_path):
    # Get a list of image files in the folder
    image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.jpeg', '.png', '.JPG'))]

    # Loop over each image in the folder
    for image_file in image_files:
        # Construct full image path
        image_path = os.path.join(folder_path, image_file)

        # Load the image
        image = io.imread(image_path)
        # Set up the plot
        plt.figure(figsize=(15, 15))

        # Plot original image
        plt.subplot(1, 2, 1)
        plt.imshow(image)
        plt.title(f"Original Image: {image_file}")

        # Convert image to HSV
        hsv_image = rgb2hsv(image)

        # Plot HSV image
        plt.subplot(1, 2, 2)
        hsv_image_display = plt.imshow(hsv_image)
        plt.title(f"HSV Image: {image_file}")

        # Add colorbar
        plt.colorbar(hsv_image_display, fraction=0.046, pad=0.04)

        # Show the plot for each image
        plt.show()

# Path to your folder with images
folder_path = '/content/drive/MyDrive/archive - imagetu(2)/blurred_images(YES)'

# Call the function to process images
```

```
process_images_in_folder(folder_path)  #we will do the same for both yes and no folder and we can all apply gaussian blur to this
```

Next:

```
from skimage import io
import matplotlib.pyplot as plt
import numpy as np # Import numpy for array manipulation

# Load the image
image = io.imread('/content/drive/MyDrive/archive -imagetu(2)/no/1 no.jpeg')

# Check if the image is grayscale (2D) and convert to RGB if necessary
if len(image.shape) == 2:  # Grayscale image (height, width)
    image = np.stack([image] * 3, axis=-1)  # Convert grayscale to RGB (height, width, 3)

# Convert image to HSV (or RGB)
hsv_image = rgb2hsv(image)

# Compute the color histograms for each channel (Hue, Saturation, and Value)
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.hist(hsv_image[..., 0].ravel(), bins=256, color='r', alpha=0.5, label="Hue")
plt.title("Histogram of Hue Channel")
plt.subplot(1, 3, 2)
plt.hist(hsv_image[..., 1].ravel(), bins=256, color='g', alpha=0.5, label="Saturation")
plt.title("Histogram of Saturation Channel")

plt.subplot(1, 3, 3)

plt.hist(hsv_image[..., 2].ravel(), bins=256, color='b', alpha=0.5, label="Value")

plt.title("Histogram of Value Channel")

plt.show()
```



```
#I have taken the dataset from Kaggle
```