Vegetable Recognition using Computer Vision and Image Processing Techniques

Higher Diploma in Software Engineering22.2F

Digital Image Processing Module Project

KAHDSE22.2F-020 L.H.M Fernando

School of Computing and Engineering

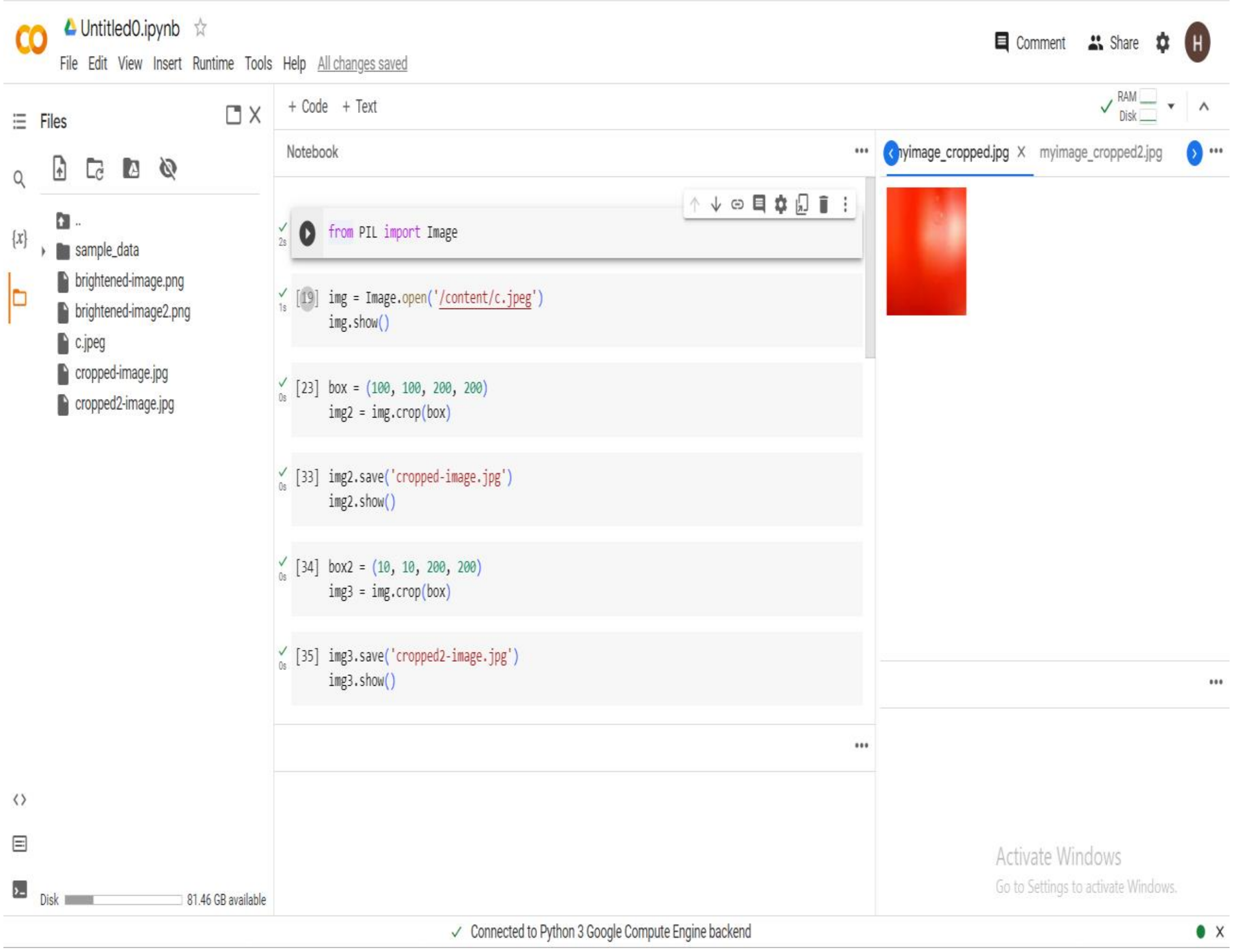
National Institute of Business Management

Image Processing and Data Augmentation

Image 1 tomato

cropping, and adjusting brightness.

reason-: increase the quality of the image



```
from PIL import Image
```

```
[19] img = Image.open('/content/c.jpeg')
     img.show()
```

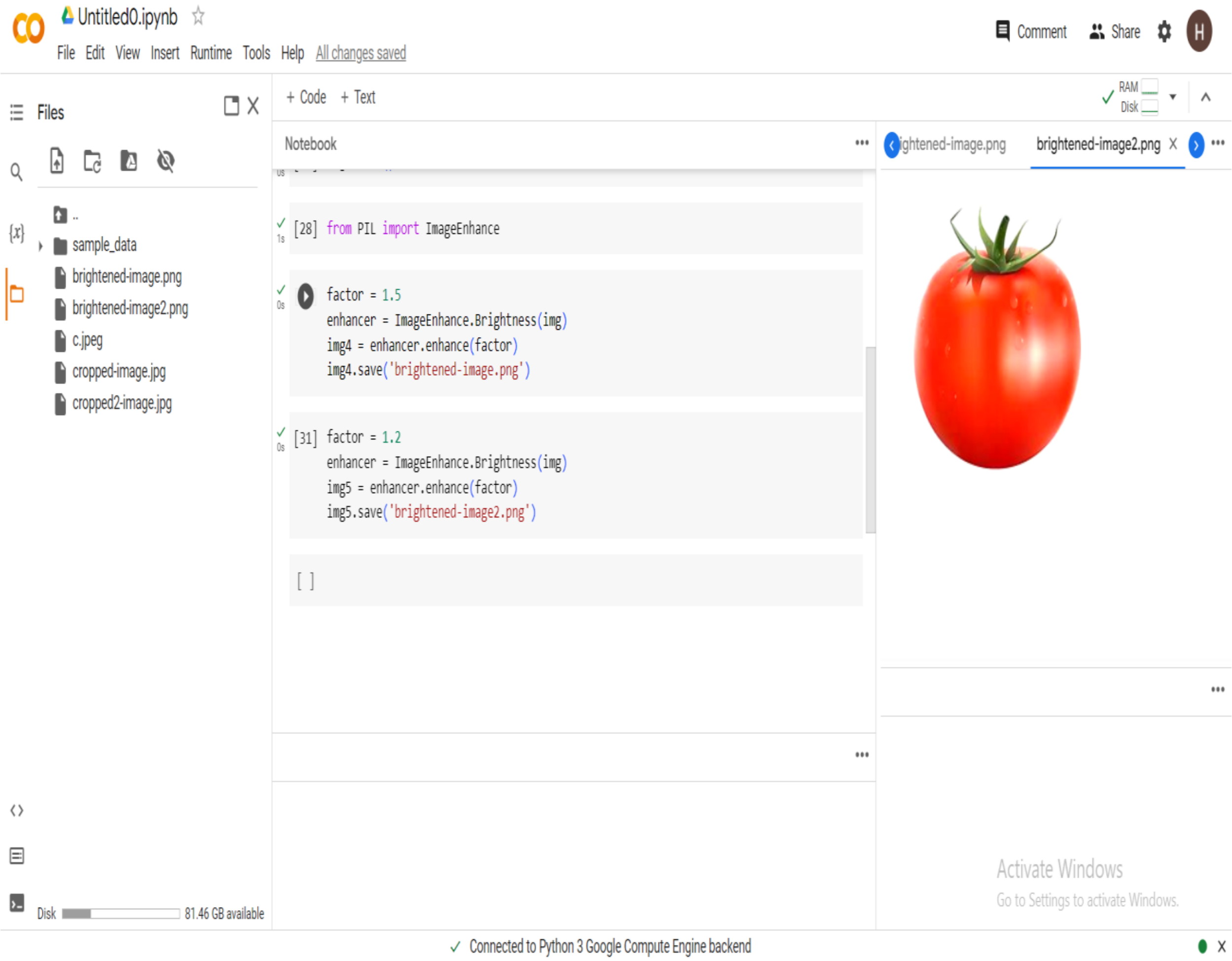
```
[23] box = (100, 100, 200, 200)
     img2 = img.crop(box)
```

```
[33] img2.save('cropped-image.jpg')
     img2.show()
```

```
[34] box2 = (10, 10, 200, 200)
     img3 = img.crop(box)
```

```
[35] img3.save('cropped2-image.jpg')
     img3.show()
```

✓ Connected to Python 3 Google Compute Engine backend



```
[28] from PIL import ImageEnhance
```

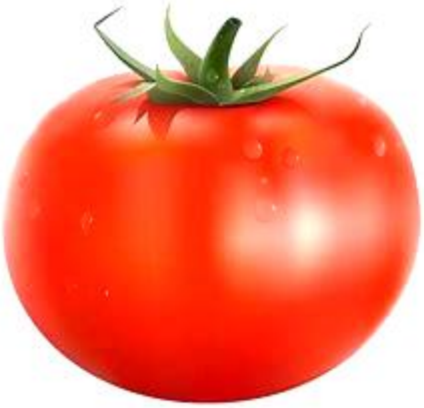
```
factor = 1.5
enhancer = ImageEnhance.Brightness(img)
img4 = enhancer.enhance(factor)
img4.save('brightened-image.png')
```

```
[31] factor = 1.2
enhancer = ImageEnhance.Brightness(img)
img5 = enhancer.enhance(factor)
img5.save('brightened-image2.png')
```

```
[ ]
```

✓ Connected to Python 3 Google Compute Engine backend

Input

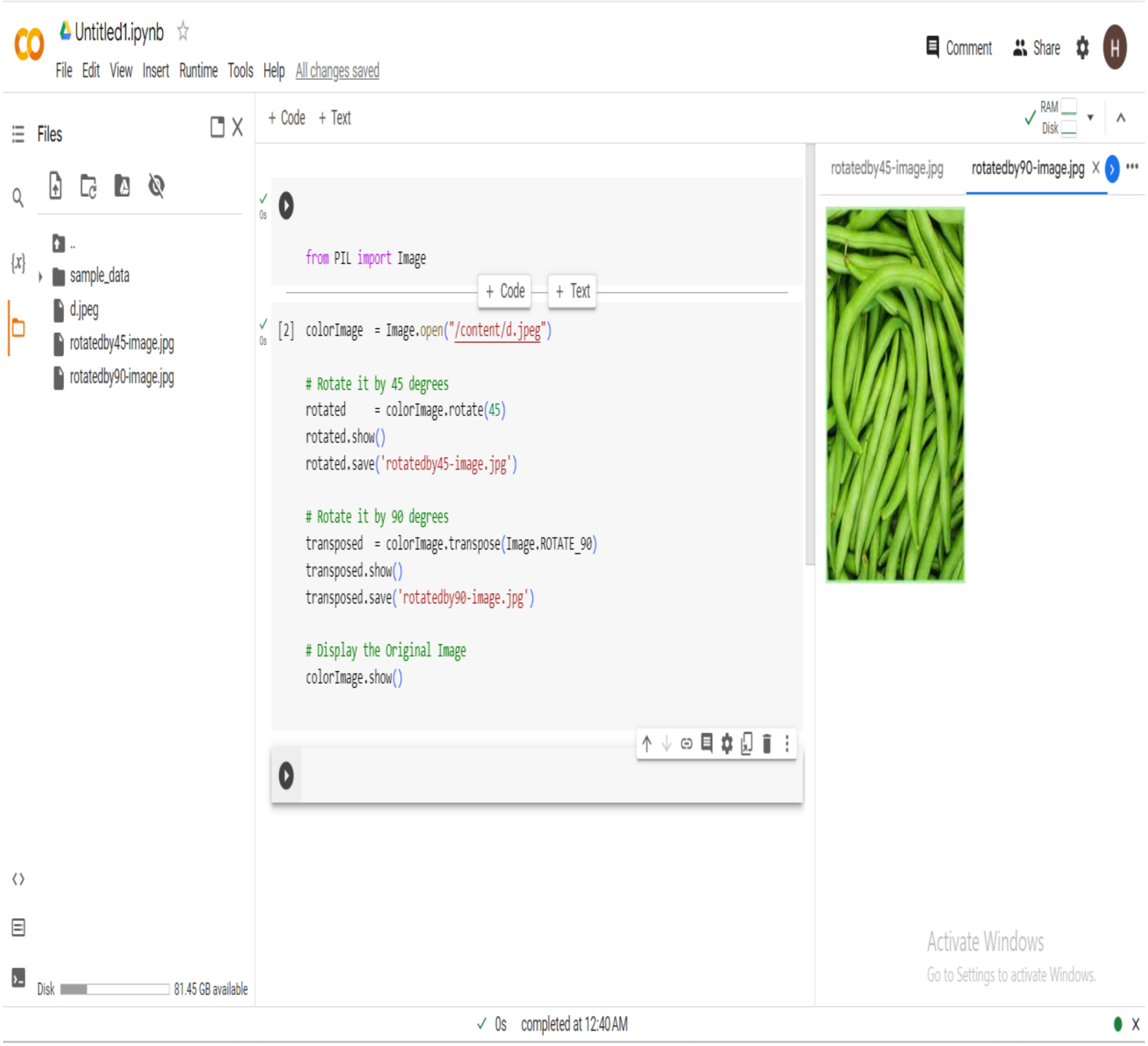


output

Image 2 beans

rotating, adjusting brightness and contrast

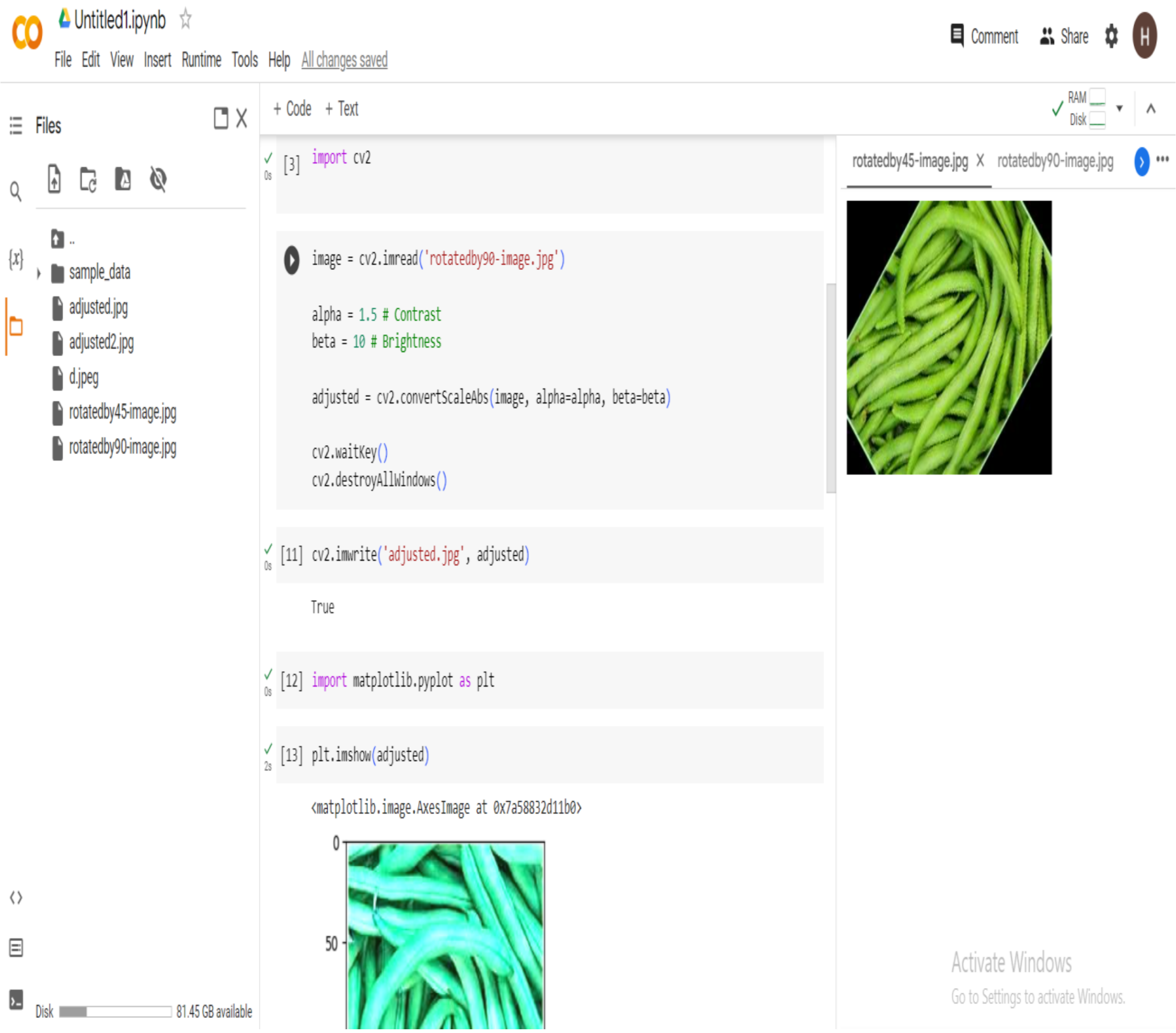
reason-: increase the quality of the image



```
from PIL import Image
```

```
[2] colorImage  = Image.open("/content/d.jpeg")

    # Rotate it by 45 degrees
    rotated     = colorImage.rotate(45)
    rotated.show()
    rotated.save('rotatedby45-image.jpg')

    # Rotate it by 90 degrees
    transposed  = colorImage.transpose(Image.ROTATE_90)
    transposed.show()
    transposed.save('rotatedby90-image.jpg')

    # Display the Original Image
    colorImage.show()
```

✓ 0s completed at 12:40 AM



```
[3] import cv2
```

```
image = cv2.imread('rotatedby90-image.jpg')

alpha = 1.5 # Contrast
beta = 10 # Brightness

adjusted = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)

cv2.waitKey()
cv2.destroyAllWindows()
```

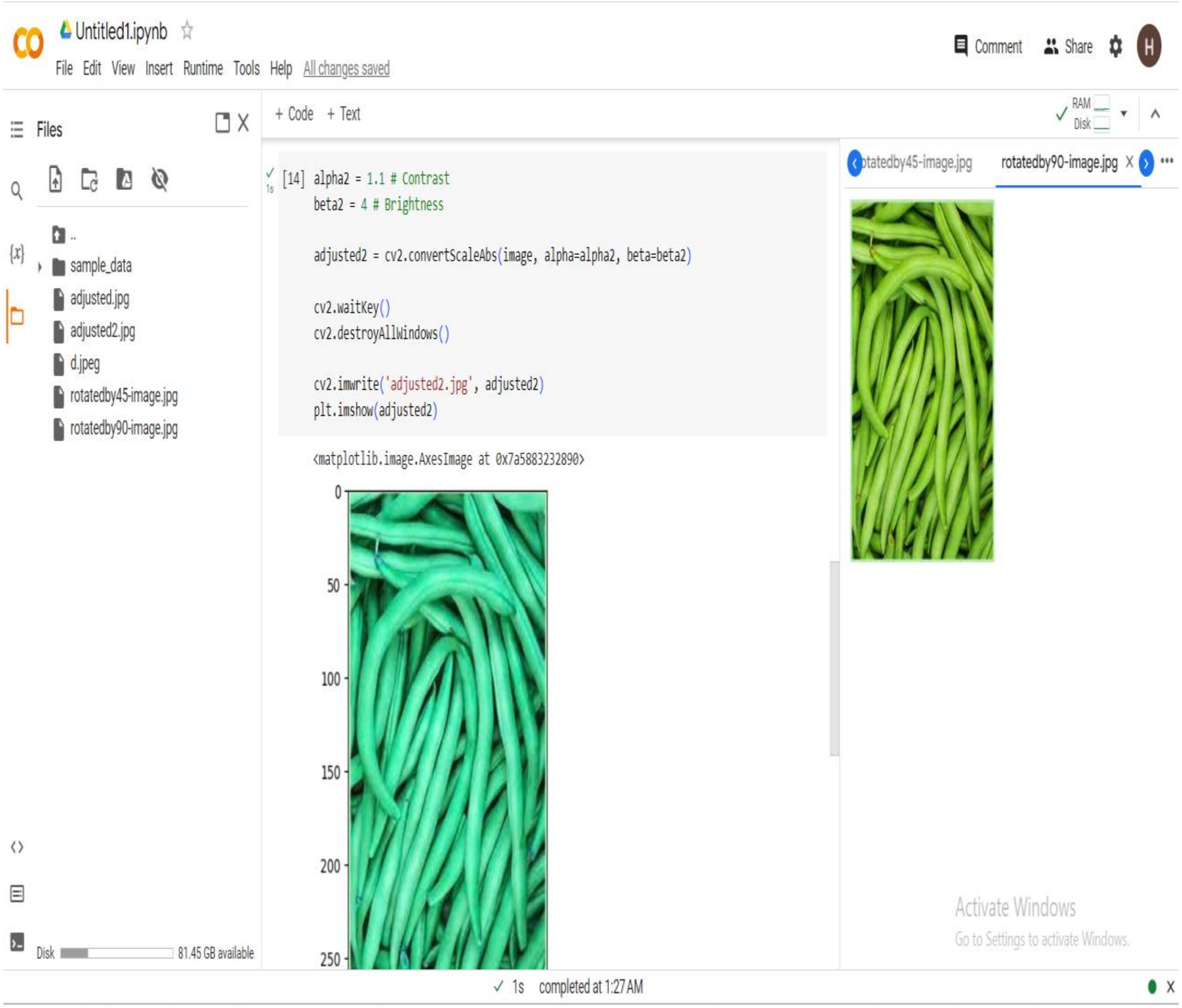
```
[11] cv2.imwrite('adjusted.jpg', adjusted)
```

```
True
```

```
[12] import matplotlib.pyplot as plt
```

```
[13] plt.imshow(adjusted)
```

```
<matplotlib.image.AxesImage at 0x7a58832d11b0>
```



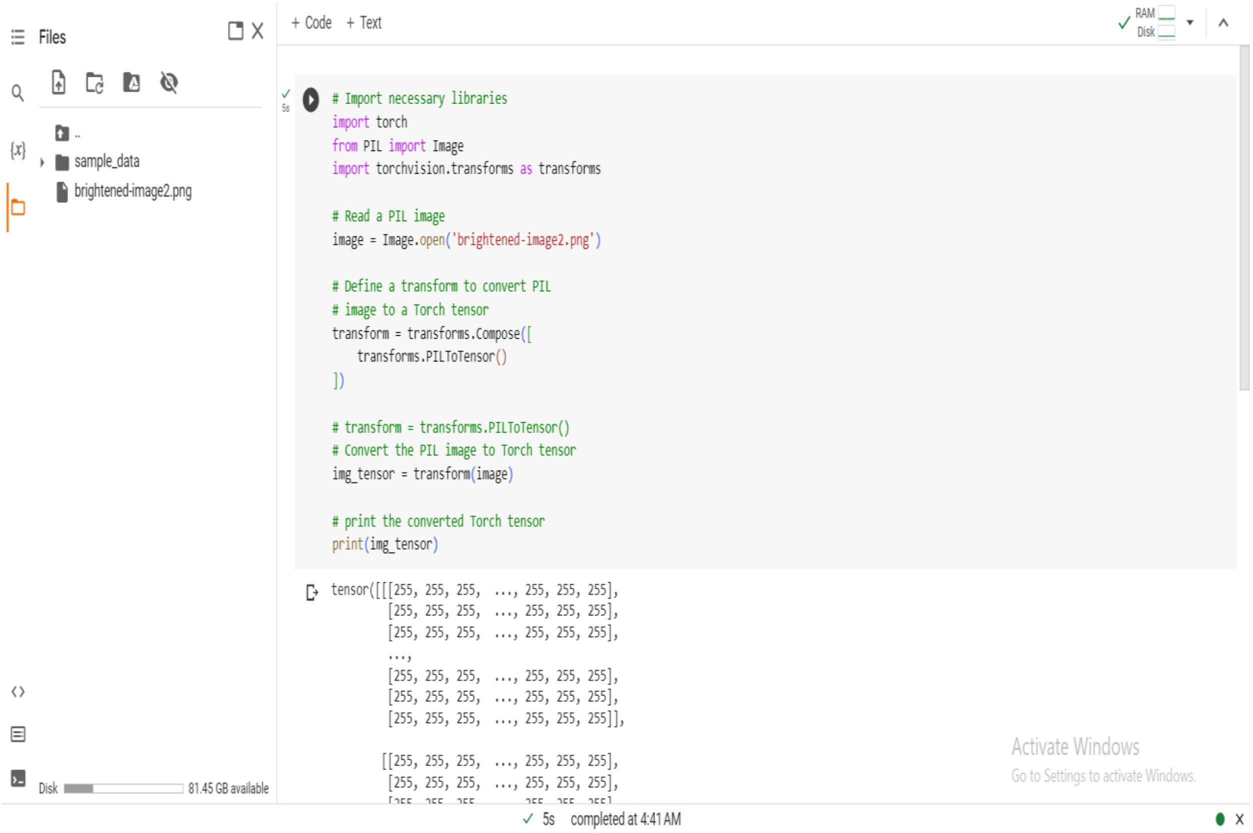
```
[14] alpha2 = 1.1 # Contrast
     beta2 = 4 # Brightness

     adjusted2 = cv2.convertScaleAbs(image, alpha=alpha2, beta=beta2)

     cv2.waitKey()
     cv2.destroyAllWindows()

     cv2.imwrite('adjusted2.jpg', adjusted2)
     plt.imshow(adjusted2)
```

```
<matplotlib.image.AxesImage at 0x7a5883232890>
```

Input

output

Data Preprocessing



```
# Import necessary libraries
import torch
from PIL import Image
import torchvision.transforms as transforms

# Read a PIL image
image = Image.open('brightened-image2.png')

# Define a transform to convert PIL
# image to a Torch tensor
transform = transforms.Compose([
    transforms.PILToTensor()
])

# transform = transforms.PILToTensor()
# Convert the PIL image to Torch tensor
img_tensor = transform(image)

# print the converted Torch tensor
print(img_tensor)
```

```
tensor([[[255, 255, 255,  ..., 255, 255, 255],
         [255, 255, 255,  ..., 255, 255, 255],
         [255, 255, 255,  ..., 255, 255, 255],
         ...,
         [255, 255, 255,  ..., 255, 255, 255],
         [255, 255, 255,  ..., 255, 255, 255],
         [255, 255, 255,  ..., 255, 255, 255]],

        [[255, 255, 255,  ..., 255, 255, 255],
         [255, 255, 255,  ..., 255, 255, 255],
```

Application Development

```
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt
import numpy as np, cv2 as cv
```

```
train_path = '/kaggle/input/vegetable-image-dataset/Vegetable Images/train'

val_path = '/kaggle/input/vegetable-image-dataset/Vegetable Images/validation'
```

```
train_ds = tf.keras.utils.image_dataset_from_directory(train_path)

val_ds = tf.keras.utils.image_dataset_from_directory(val_path)
```

```
vege_names = train_ds.class_names
```

```
vege_names
```

```
model = Sequential()
```

```
model.add(layers.Rescaling(1./255, input_shape=(256, 256, 3)))
model.add(layers.Conv2D(16, 3, padding='same', activation='relu'))
model.add(layers.MaxPooling2D())

model.add(layers.Conv2D(32, 3, padding='same', activation='relu'))
model.add(layers.MaxPooling2D())

model.add(layers.Conv2D(16, 3, padding='same', activation='relu'))
model.add(layers.MaxPooling2D())

model.add(layers.Flatten())

model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(len(vege_names)))

model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])



model.summary()



hist = model.fit(train_ds, validation_data=val_ds, epochs=10)



acc = hist.history['accuracy']
val_acc = hist.history['val_accuracy']
```

```
loss = hist.history['loss']
val_loss = hist.history['val_loss']

epochs_range = range(10)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='upper left')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper left')
plt.title('Training and Validation Loss')
plt.show()



test_path = '/kaggle/input/vegetable-image-dataset/Vegetable Images/test/Bitter_Gourd/1208.jpg'

img = cv.imread(test_path)
rgb = cv.cvtColor(img, cv.COLOR_BGR2RGB)
resized = cv.resize(rgb, (256, 256))
img_array = np.expand_dims(resized, 0)

predictions = model.predict(img_array)
```

```
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(vege_names[np.argmax(score)], 100 * np.max(score))
)

plt.imshow(resized)
plt.show()
```