



Shri Vile Parle Kelavani Mandal's

DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING

(Autonomous College Affiliated to the University of Mumbai)

NAAC Accredited with "A" Grade (CGPA : 3.18)



Department of Computer Science and Engineering (Data Science)

Image Processing and Computer Vision I (DJ19DSL603) Lab

1: Arithmetic and Logical Operations on Image

Name: Yash Thakar

SAP ID.: 60009210205

Aim: To Perform Basic Image Processing Operations in Python.

Problem Statement: Develop a Python program utilizing the OpenCV library to manipulate images from the MNIST digits dataset. The program should address the following tasks:

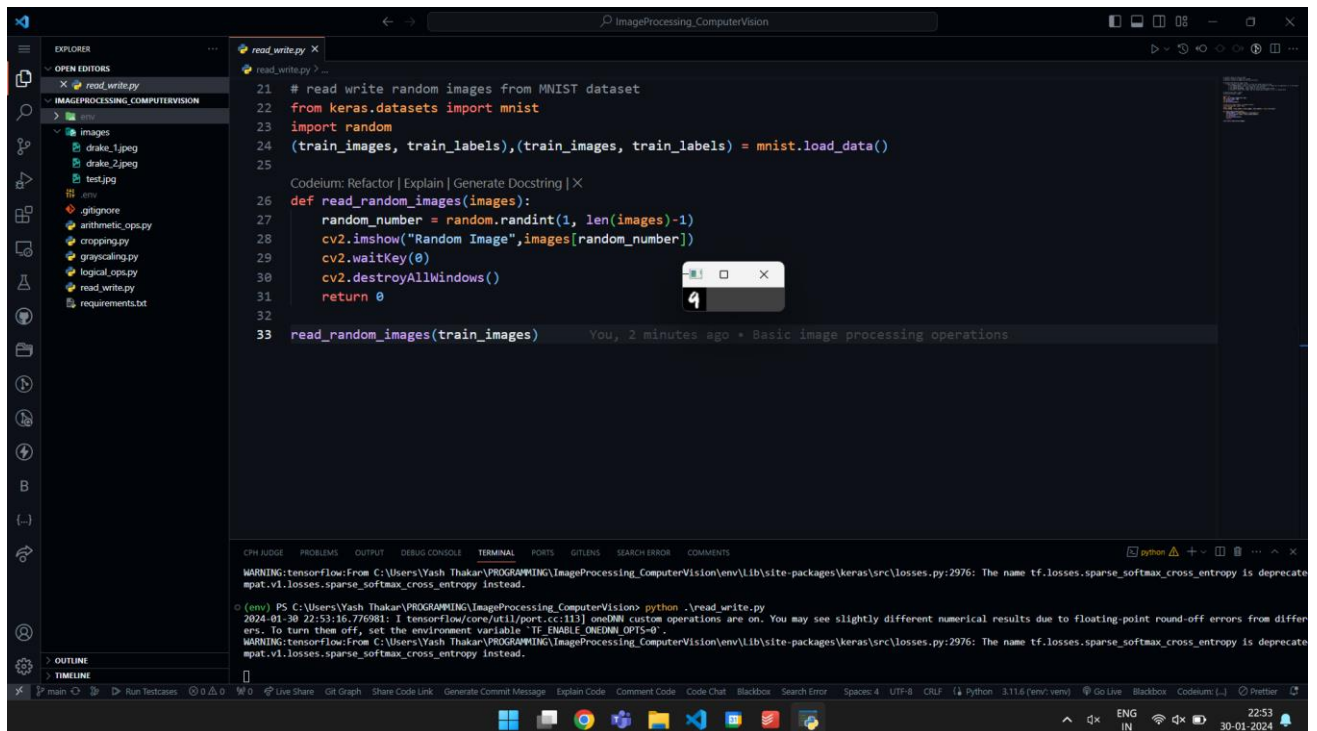
1. Read random image(s) from the MNIST digits dataset given the digit value.
2. Display the before & after image(s) used in every task below.
3. Convert the image to grayscale.
4. Implement image cropping functionality.
5. Perform arithmetic operations on the images - Addition & Subtraction.
6. Implement logical operations on the image(s) - AND, OR, NOT & XOR.

The solution to the operations performed must be produced by scratch coding without the use of built in OpenCV methods.

Dataset Link: [MNIST Digits Kaggle](#)

Results:

Read random image(s) from the MNIST digits dataset given the digit value.



```
21 # read write random images from MNIST dataset
22 from keras.datasets import mnist
23 import random
24 (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
25
26 Codeium: Refactor | Explain | Generate Docstring | X
27 def read_random_images(images):
28     random_number = random.randint(1, len(images)-1)
29     cv2.imshow("Random Image", images[random_number])
30     cv2.waitKey(0)
31     cv2.destroyAllWindows()
32     return 0
33 read_random_images(train_images)
```

WARNING:tensorflow:From C:\Users\Yash Thakar\PROGRAMMING\ImageProcessing_ComputerVision\env\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use keras.losses.sparse_softmax_cross_entropy instead.

(env) PS C:\Users\Yash Thakar\PROGRAMMING\ImageProcessing_ComputerVision> python .\read_write.py

2024-01-30 22:53:16.776981: I tensorflow/core/util/port.cc:113] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different builds. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.

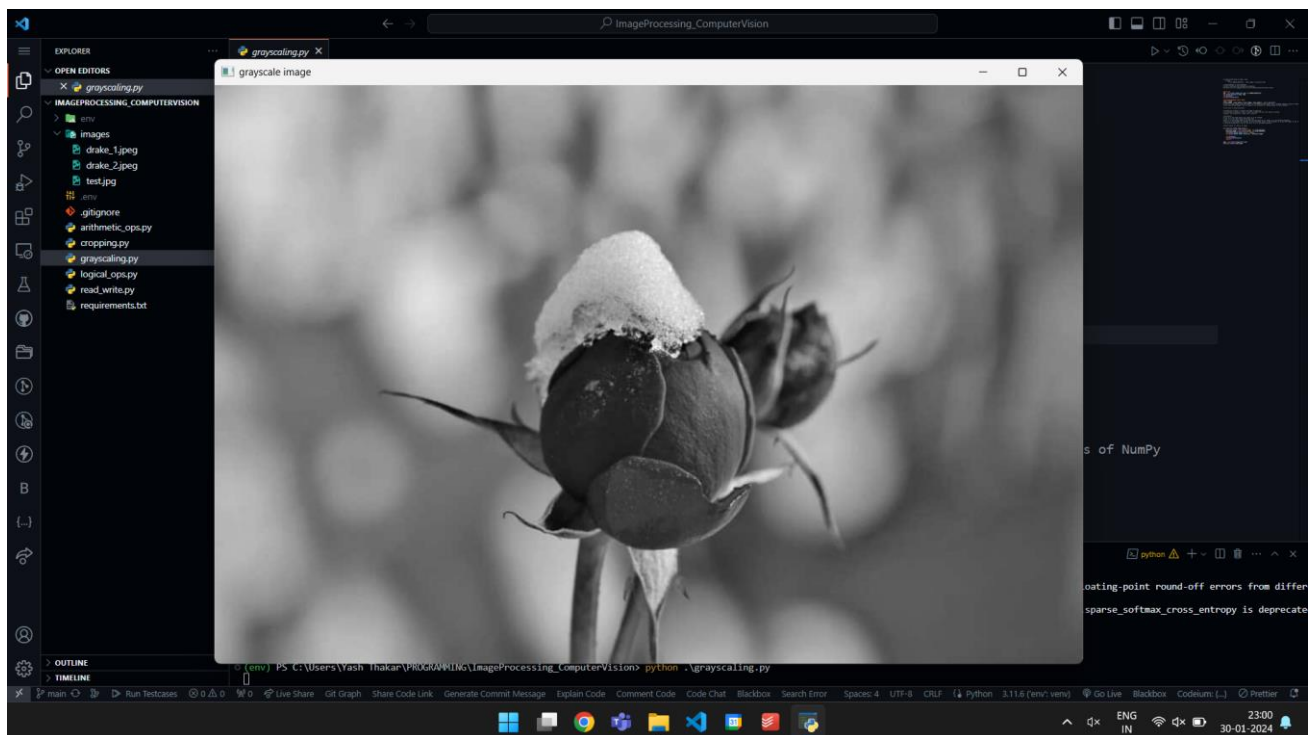
WARNING:tensorflow:From C:\Users\Yash Thakar\PROGRAMMING\ImageProcessing_ComputerVision\env\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use keras.losses.sparse_softmax_cross_entropy instead.

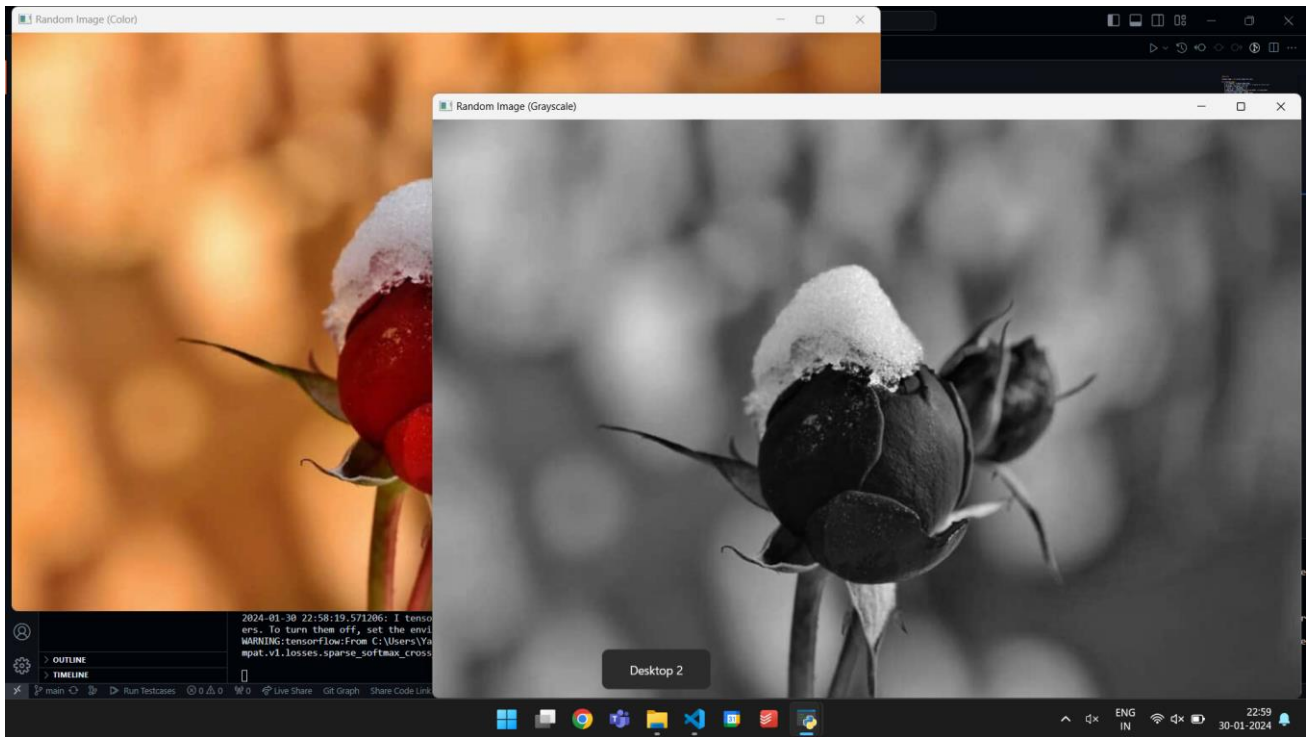
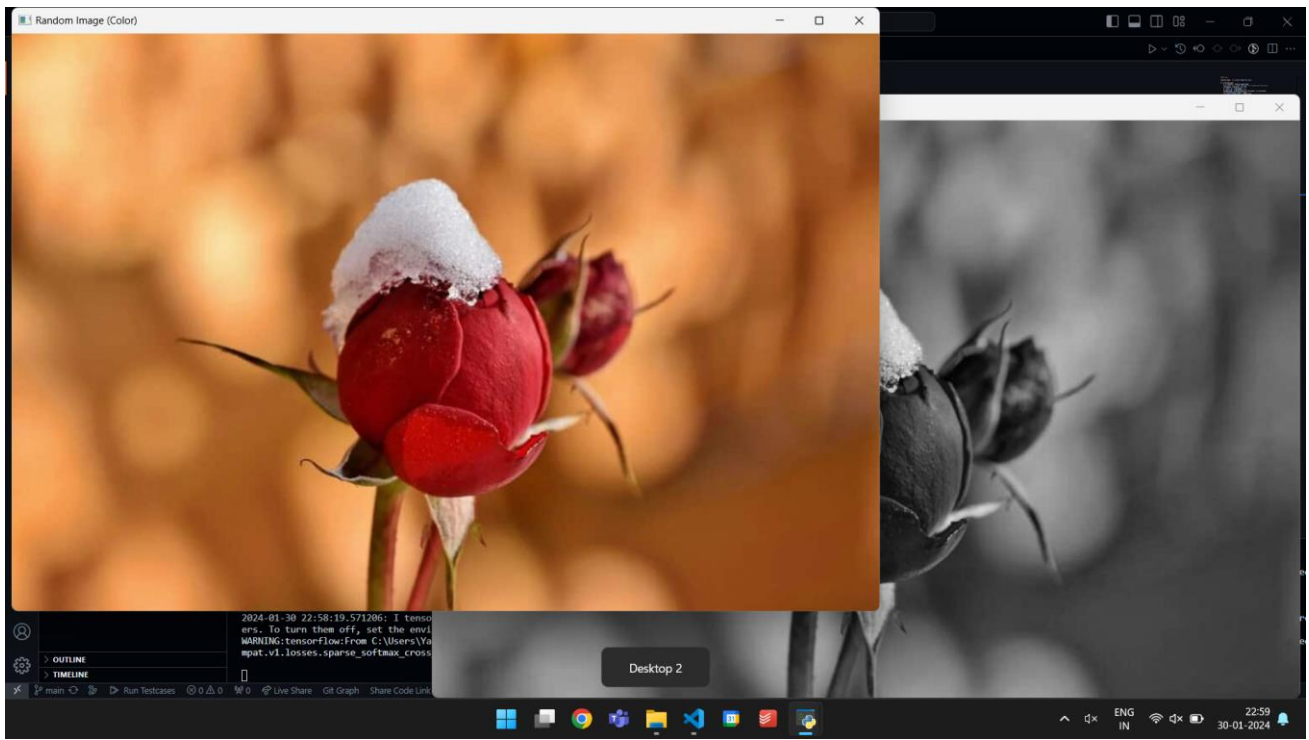
4

Convert the image to grayscale.

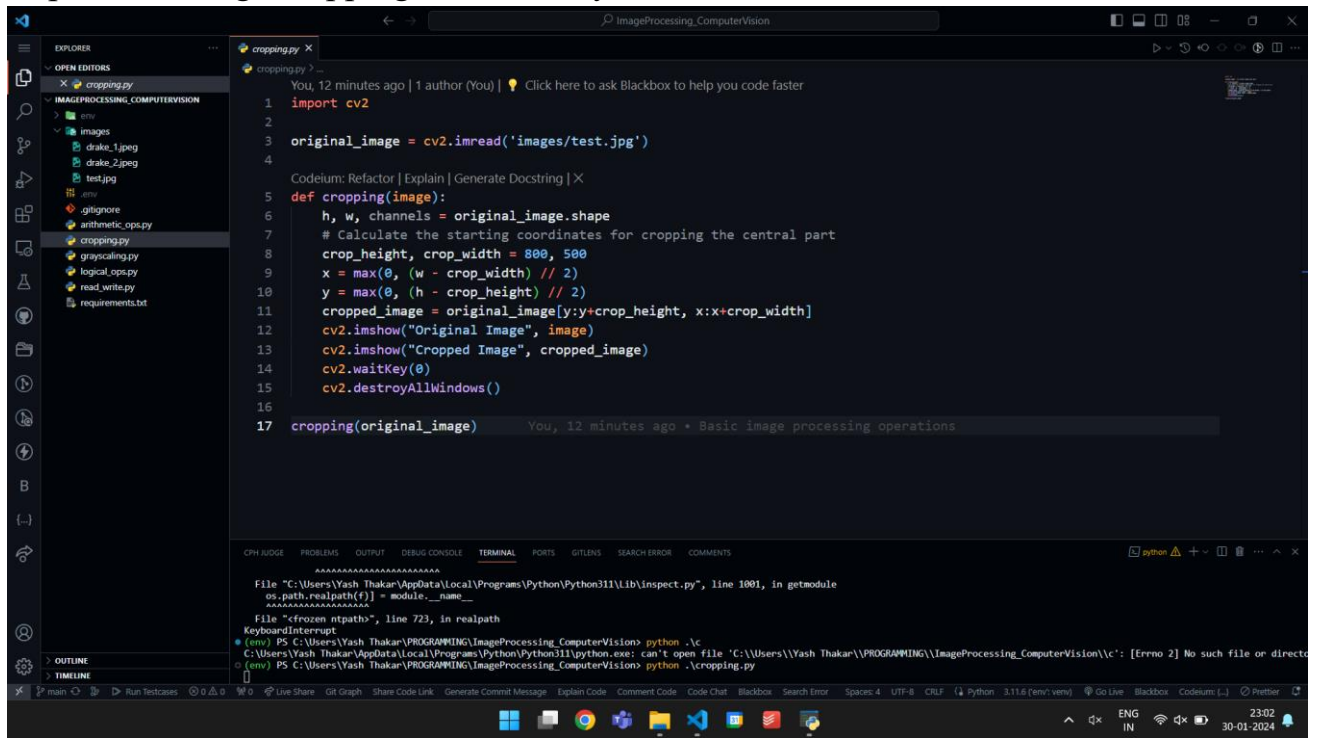
```
You, 6 minutes ago | 1 author (You) | Click here to ask Blackbox to help you code faster
1 # imread(folder/path_to_image, flag)
2 # flag:
3 # cv2.IMREAD_GRAYSCALE : Loads image in grayscale mode
4
5 # COLOR_BGR2GRAY vs COLOR_RGB2GRAY
6 # https://stackoverflow.com/questions/62855718/why-would-cv2-color-rgb2gray-and-cv2-color-bgr2gray-give-different-results
7
8
9 import cv2
10 img = cv2.imread('images/test.jpg',cv2.IMREAD_GRAYSCALE)
11 cv2.imshow('grayscale image',img)
12 cv2.waitKey(0)
13 cv2.destroyAllWindows()
14
15 from keras.datasets import mnist
16 import random
17 (train_images, train_labels),(train_images, train_labels) = mnist.load_data()
```

```
Codeium: Refactor | Explain | Generate Docstring | X
34 def grayscale_random_images(images):
35     # grayscale_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
36     grayscale_image = cv2.cvtColor(images, cv2.COLOR_RGB2GRAY)
37     cv2.imshow("Random Image (Color)", images)
38     cv2.imshow("Random Image (Grayscale)", grayscale_image)
39
40     cv2.waitKey(0)
41     cv2.destroyAllWindows()
42     return 0
43
44 img2 = cv2.imread('images/test.jpg')
45 grayscale_random_images(img2)
```





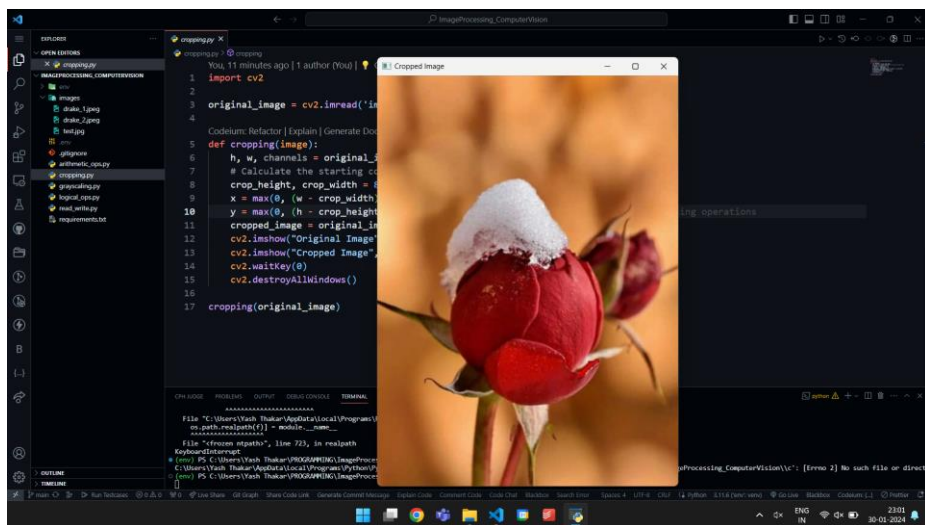
Implement image cropping functionality.



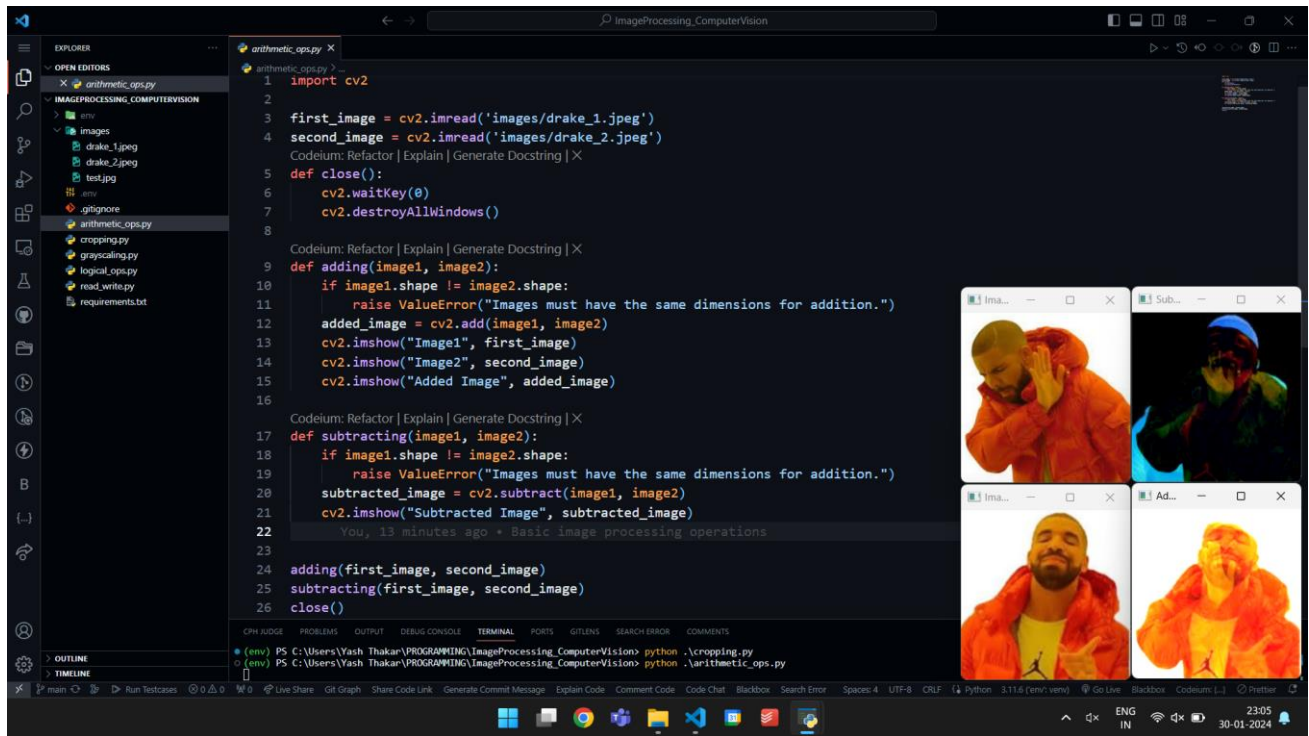
The screenshot shows a VS Code editor with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The file explorer shows a project named 'ImageProcessing_ComputerVision' with a subdirectory 'images' containing 'drake_1.jpeg', 'drake_2.jpeg', and 'test.jpg'. The code editor shows the file 'cropping.py' with the following Python code:

```
1 import cv2
2
3 original_image = cv2.imread('images/test.jpg')
4
5 def cropping(image):
6     h, w, channels = original_image.shape
7     # Calculate the starting coordinates for cropping the central part
8     crop_height, crop_width = 800, 500
9     x = max(0, (w - crop_width) // 2)
10    y = max(0, (h - crop_height) // 2)
11    cropped_image = original_image[y:y+crop_height, x:x+crop_width]
12    cv2.imshow("Original Image", image)
13    cv2.imshow("Cropped Image", cropped_image)
14    cv2.waitKey(0)
15    cv2.destroyAllWindows()
16
17 cropping(original_image)
```

The terminal at the bottom shows the output of the program, indicating that the image was successfully cropped and displayed.



Perform arithmetic operations on the images - Addition & Subtraction.



Implement logical operations on the image(s) - AND, OR, NOT & XOR.

```
you, 12 seconds ago | 1 author (you) | Click here to ask Blackbox to help you code faster
1  import cv2
2
Codeium: Refactor | Explain | Generate Docstring | X
3  def logical_and(image1, image2):
4      img1 = cv2.imread(image1)
5      img2 = cv2.imread(image2)
6
7      img2 = cv2.resize(img2, (img1.shape[1], img1.shape[0]))
8      result = cv2.bitwise_and(img1, img2)
9
10     return result
11
Codeium: Refactor | Explain | Generate Docstring | X
12 def logical_or(image1, image2):
13     img1 = cv2.imread(image1)
14     img2 = cv2.imread(image2)
15
16     img2 = cv2.resize(img2, (img1.shape[1], img1.shape[0]))
17     result = cv2.bitwise_or(img1, img2)
18
19     return result
20
```

```
Codeium: Refactor | Explain | Generate Docstring | X
21 def logical_not(image):
22     img = cv2.imread(image)
23
24     result = cv2.bitwise_not(img)
25
26     return result
27
Codeium: Refactor | Explain | Generate Docstring | X
28 def logical_xor(image1, image2):
29     img1 = cv2.imread(image1)
30     img2 = cv2.imread(image2)
31     img2 = cv2.resize(img2, (img1.shape[1], img1.shape[0]))
32     result = cv2.bitwise_xor(img1, img2)
33
34     return result
35
36
37 image1_path = 'images\drake_1.jpeg'
38 image2_path = 'images\drake_2.jpeg'
```

```

40 result_and = logical_and(image1_path, image2_path)
41 cv2.imshow('AND', result_and)
42
43 result_or = logical_or(image1_path, image2_path)
44 cv2.imshow('OR', result_or)
45
46 result_not = logical_not(image1_path)
47 cv2.imshow('NOT', result_not)
48
49 result_xor = logical_xor(image1_path, image2_path)
50 cv2.imshow('XOR', result_xor)
51
52 cv2.waitKey(0)
53 cv2.destroyAllWindows()
54

```

