NAME- ANIRUDH SHUKLA

QUESTION:

Write a program to perform basic image processing functions with openCV and show the result in GUI such as flask, Tkinter, Matlab etc. Basic image processing functions require:

1) Browse to upload an image.

2) Convert the image in RGB, Grayscale, Binary (thresholding) etc by user choice/button.

3) Perform brightness and contrast improvement

4) Perform various image annotation such as to add line, rectangle, circle or text at desired positions.

CODE:

```
import cv2

import numpy as np

from tkinter import *

from tkinter import filedialog

from tkinter import simpledialog, messagebox

from PIL import Image, ImageTk


class ImageProcessor:

    def __init__(self, root):

        self.root = root

        self.root.title("Image Processor")

        self.image_label = Label(root)

        self.image_label.pack()

        self.btn_load = Button(root, text="Load Image", command=self.load_image)

        self.btn_load.pack()

        self.btn_rgb = Button(root, text="RGB", command=self.convert_to_rgb)

        self.btn_rgb.pack()

        self.btn_gray = Button(root, text="Grayscale", command=self.convert_to_gray)

        self.btn_gray.pack()
```

```
        self.btn_binary = Button(root, text="Binary", command=self.convert_to_binary)
        self.btn_binary.pack()
        self.btn_brightness = Button(root, text="Brightness", command=self.adjust_brightness)
        self.btn_brightness.pack()
        self.btn_contrast = Button(root, text="Contrast", command=self.adjust_contrast)
        self.btn_contrast.pack()
        self.btn_annotationline = Button(root, text="Add Annotation Line",
command=self.add_annotation_line)
        self.btn_annotationline.pack()
        self.btn_annotationrect = Button(root, text="Add Annotation Rectangle",
command=self.add_annotation_rectangle)
        self.btn_annotationrect.pack()
        self.btn_annotationcirc = Button(root, text="Add Annotation Circle",
command=self.add_annotation_circle)
        self.btn_annotationcirc.pack()
        self.btn_annotationputtext = Button(root, text="Add a Text",
command=self.add_annotation_puttext)
        self.btn_annotationputtext.pack()
        self.image = None

    def load_image(self):
        path = filedialog.askopenfilename()
        if path:
            self.image = cv2.imread(path)
            self.display_image()

    def display_image(self):
        image_rgb = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)
        image_pil = Image.fromarray(image_rgb)
        image_tk = ImageTk.PhotoImage(image_pil)
        self.image_label.configure(image=image_tk)
        self.image_label.image = image_tk
```

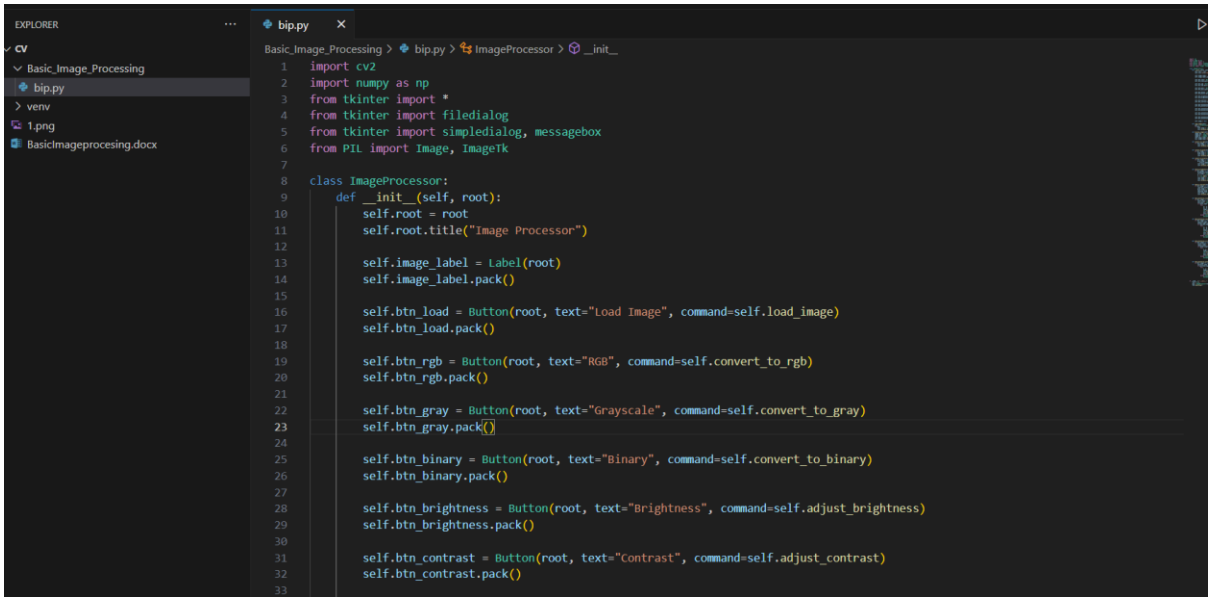
```
def convert_to_rgb(self):
    if self.image is not None:
        self.image = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)
        self.display_image()

def convert_to_gray(self):
    if self.image is not None:
        gray_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
        self.image = cv2.cvtColor(gray_image, cv2.COLOR_GRAY2BGR)
        self.display_image()

def convert_to_binary(self):
    if self.image is not None:
        gray_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
        _, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
        self.image = cv2.cvtColor(binary_image, cv2.COLOR_GRAY2BGR)
        self.display_image()

def adjust_brightness(self):
    if self.image is not None:
        brightness_value = 50  # Example: adjust brightness by adding 50
        hsv_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2HSV)
        h, s, v = cv2.split(hsv_image)
        v = cv2.add(v, brightness_value)
        self.image = cv2.merge((h, s, v))
        self.image = cv2.cvtColor(self.image, cv2.COLOR_HSV2BGR)
        self.display_image()

def adjust_contrast(self):
```

```
        if self.image is not None:
            contrast_factor = 1.5  # Example: adjust contrast by multiplying by 1.5
            gray_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
            mean_intensity = np.mean(gray_image)
            adjusted_image = cv2.convertScaleAbs(self.image, alpha=contrast_factor, beta=-mean_intensity*(contrast_factor-1))
            self.image = cv2.cvtColor(adjusted_image, cv2.COLOR_BGR2GRAY)
            self.display_image()

    def add_annotation_line(self):
        if self.image is not None:
            start_input = simpledialog.askstring("Line Start Point", "Enter start point (x1,y1) for the line (comma-separated):")
            end_input = simpledialog.askstring("Line End Point", "Enter end point (x2,y2) for the line (comma-separated):")
            if start_input and end_input:
                try:
                    x1, y1 = map(int, start_input.split(','))
                    x2, y2 = map(int, end_input.split(','))
                    image_copy = self.image.copy()
                    cv2.line(image_copy, (x1, y1), (x2, y2), (0, 0, 0), 2)
                    self.image = image_copy
                    self.display_image()
                except ValueError:
                    messagebox.showerror("Error", "Invalid input. Please enter coordinates in the format 'x,y'.")

    def add_annotation_rectangle(self):
        if self.image is not None:
            top_left_input = simpledialog.askstring("Rectangle Top-Left", "Enter top-left corner (x1,y1) for the rectangle (comma-separated):")
```

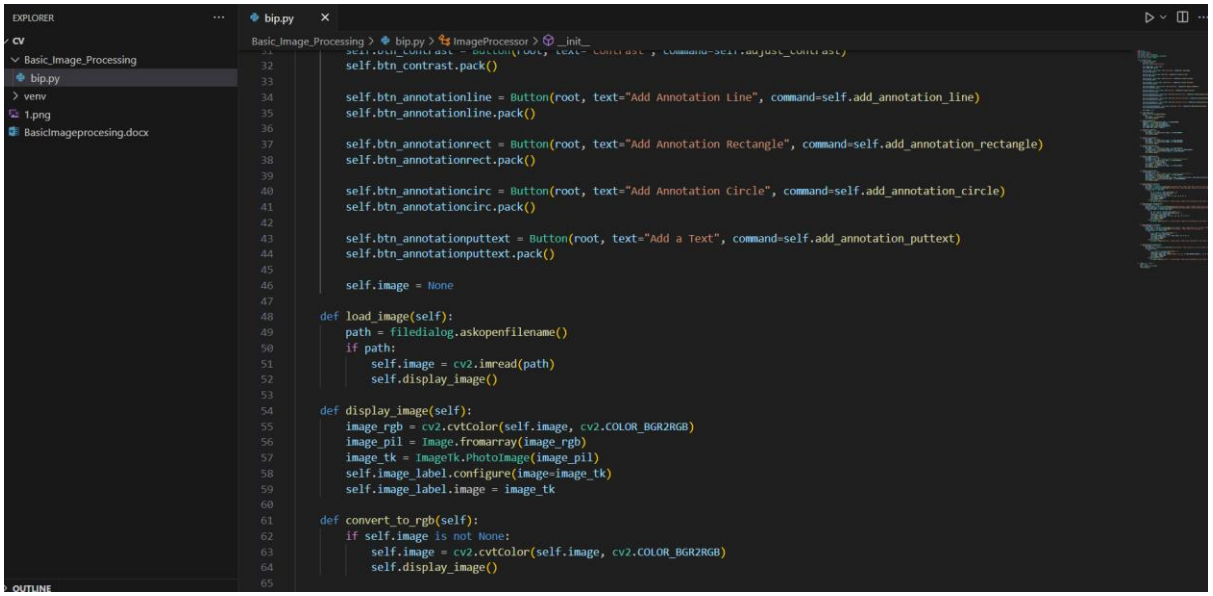
```
        bottom_right_input = simpledialog.askstring("Rectangle Bottom-Right", "Enter bottom-right corner (x2,y2) for the rectangle (comma-separated):")
        if top_left_input and bottom_right_input:
            try:
                x1, y1 = map(int, top_left_input.split(','))
                x2, y2 = map(int, bottom_right_input.split(','))
                image_copy = self.image.copy()
                cv2.rectangle(image_copy, (x1, y1), (x2, y2), (0, 0, 0), 2)
                self.image = image_copy
                self.display_image()
            except ValueError:
                messagebox.showerror("Error", "Invalid input. Please enter coordinates in the format 'x,y'.")

    def add_annotation_circle(self):
        if self.image is not None:
            center_input = simpledialog.askstring("Circle Center", "Enter center point (x,y) for the circle (comma-separated):")
            radius_input = simpledialog.askinteger("Circle Radius", "Enter radius for the circle:")
            if center_input and radius_input:
                try:
                    x, y = map(int, center_input.split(','))
                    image_copy = self.image.copy()
                    cv2.circle(image_copy, (x, y), radius_input, (0, 0, 0), 2)
                    self.image = image_copy
                    self.display_image()
                except ValueError:
                    messagebox.showerror("Error", "Invalid input. Please enter coordinates in the format 'x,y'.")

    def add_annotation_puttext(self):
        if self.image is not None:
```

```
        position_input = simpledialog.askstring("Text Position", "Enter position (x,y) for the text (comma-separated):")
        if position_input:
            try:
                x, y = map(int, position_input.split(','))
                image_copy = self.image.copy()
                cv2.putText(image_copy, "ANIRUDH SHUKLA", (x, y), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)
                self.image = image_copy
                self.display_image()
            except ValueError:
                messagebox.showerror("Error", "Invalid input. Please enter coordinates in the format 'x,y'.")


if __name__ == "__main__":
    root = Tk()
    app = ImageProcessor(root)
    root.mainloop()
```

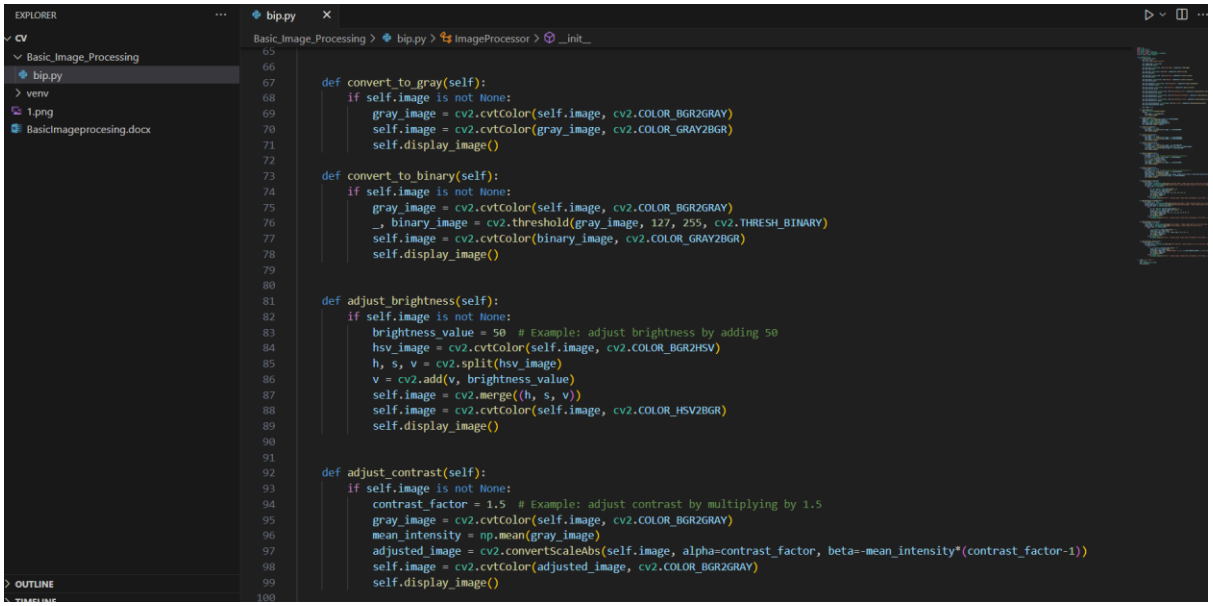
CODE SCREENSHOT:



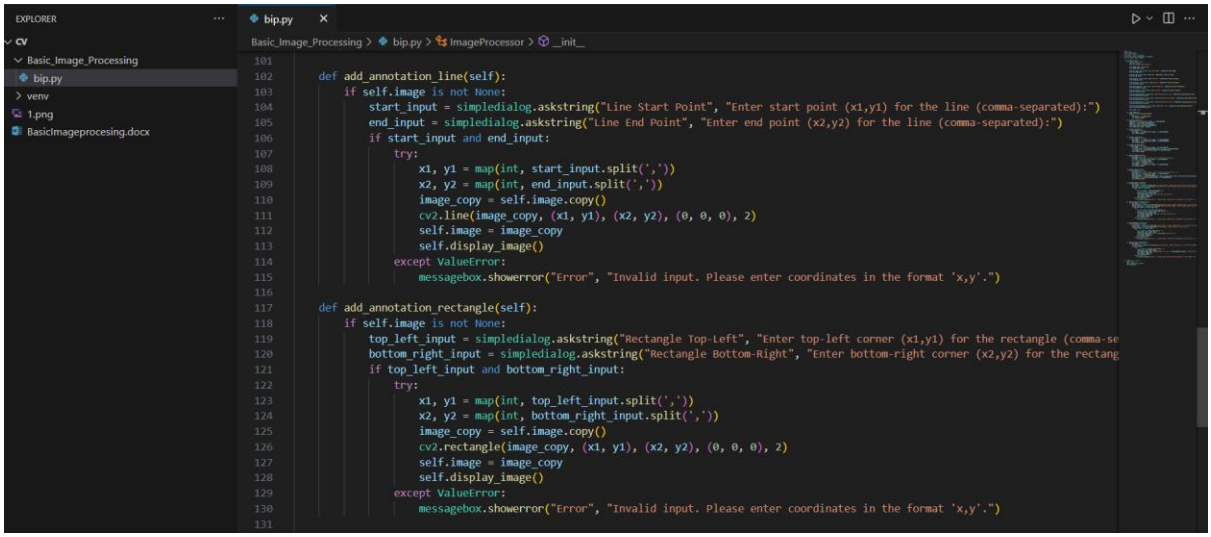
```
import cv2
import numpy as np
from tkinter import *
from tkinter import filedialog
from tkinter import simpledialog, messagebox
from PIL import Image, ImageTk

class ImageProcessor:
    def __init__(self, root):
        self.root = root
        self.root.title("Image Processor")

        self.image_label = Label(root)
        self.image_label.pack()

        self.btn_load = Button(root, text="Load Image", command=self.load_image)
        self.btn_load.pack()

        self.btn_rgb = Button(root, text="RGB", command=self.convert_to_rgb)
        self.btn_rgb.pack()

        self.btn_gray = Button(root, text="Grayscale", command=self.convert_to_gray)
        self.btn_gray.pack()

        self.btn_binary = Button(root, text="Binary", command=self.convert_to_binary)
        self.btn_binary.pack()

        self.btn_brightness = Button(root, text="Brightness", command=self.adjust_brightness)
        self.btn_brightness.pack()

        self.btn_contrast = Button(root, text="Contrast", command=self.adjust_contrast)
        self.btn_contrast.pack()
```

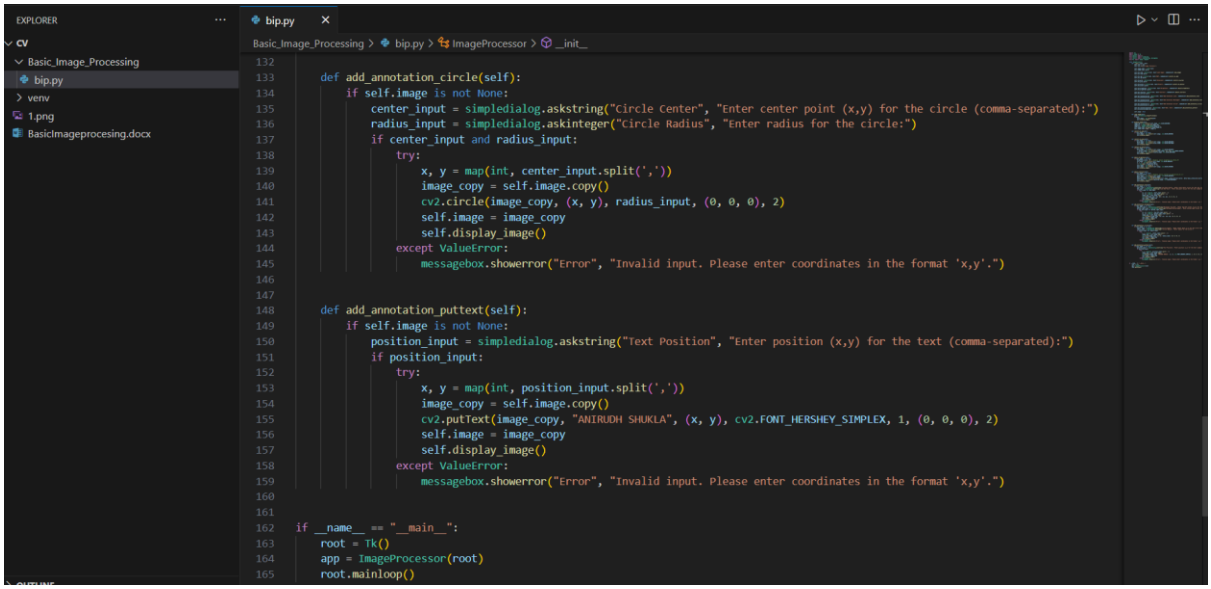


```
        self.btn_contrast.pack()

        self.btn_annotationline = Button(root, text="Add Annotation Line", command=self.add_annotation_line)
        self.btn_annotationline.pack()

        self.btn_annotationrect = Button(root, text="Add Annotation Rectangle", command=self.add_annotation_rectangle)
        self.btn_annotationrect.pack()

        self.btn_annotationcirc = Button(root, text="Add Annotation Circle", command=self.add_annotation_circle)
        self.btn_annotationcirc.pack()

        self.btn_annotationputtext = Button(root, text="Add a Text", command=self.add_annotation_puttext)
        self.btn_annotationputtext.pack()

        self.image = None

    def load_image(self):
        path = filedialog.askopenfilename()
        if path:
            self.image = cv2.imread(path)
            self.display_image()

    def display_image(self):
        image_rgb = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)
        image_pil = Image.fromarray(image_rgb)
        image_tk = ImageTk.PhotoImage(image_pil)
        self.image_label.configure(image=image_tk)
        self.image_label.image = image_tk

    def convert_to_rgb(self):
        if self.image is not None:
            self.image = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)
            self.display_image()
```



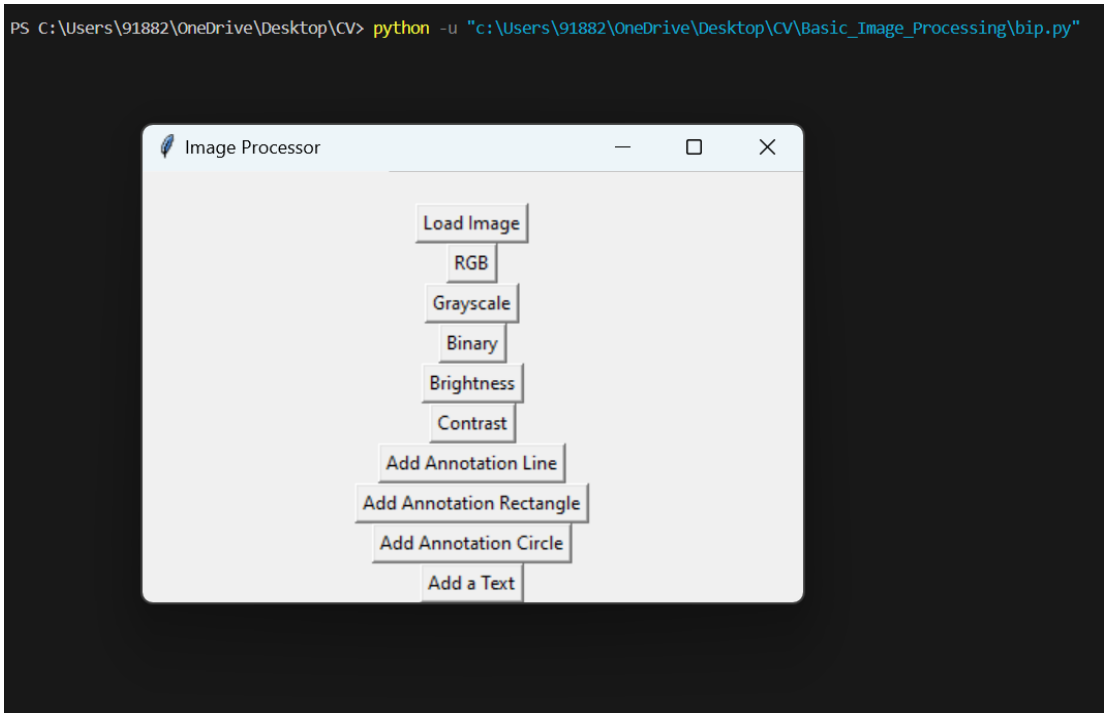
```


    def convert_to_gray(self):
        if self.image is not None:
            gray_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
            self.image = cv2.cvtColor(gray_image, cv2.COLOR_GRAY2BGR)
            self.display_image()

    def convert_to_binary(self):
        if self.image is not None:
            gray_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
            _, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
            self.image = cv2.cvtColor(binary_image, cv2.COLOR_GRAY2BGR)
            self.display_image()


    def adjust_brightness(self):
        if self.image is not None:
            brightness_value = 50  # Example: adjust brightness by adding 50
            hsv_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2HSV)
            h, s, v = cv2.split(hsv_image)
            v = cv2.add(v, brightness_value)
            self.image = cv2.merge((h, s, v))
            self.image = cv2.cvtColor(self.image, cv2.COLOR_HSV2BGR)
            self.display_image()


    def adjust_contrast(self):
        if self.image is not None:
            contrast_factor = 1.5  # Example: adjust contrast by multiplying by 1.5
            gray_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
            mean_intensity = np.mean(gray_image)
            adjusted_image = cv2.convertScaleAbs(self.image, alpha=contrast_factor, beta=-mean_intensity*(contrast_factor-1))
            self.image = cv2.cvtColor(adjusted_image, cv2.COLOR_BGR2GRAY)
            self.display_image()
```



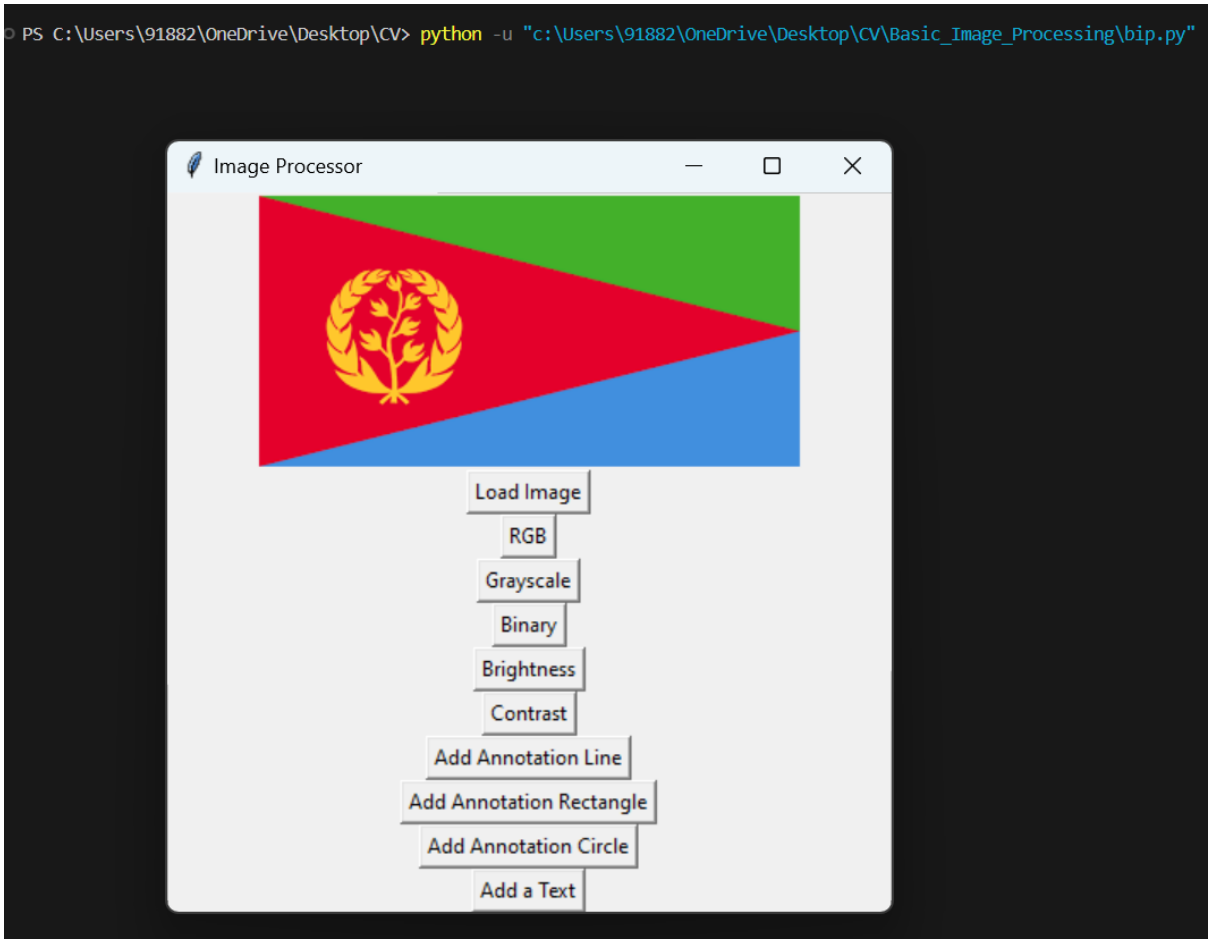


OUTPUT SCREENSHOT:

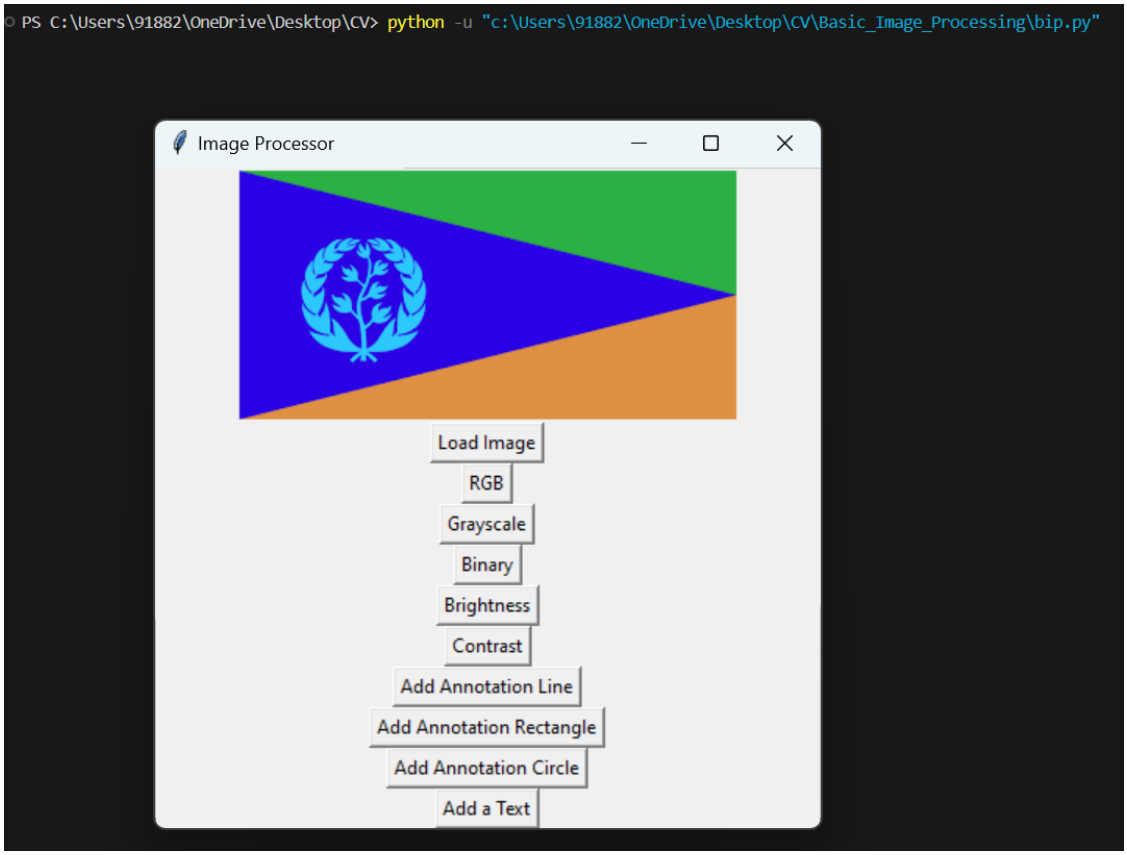
GUI:



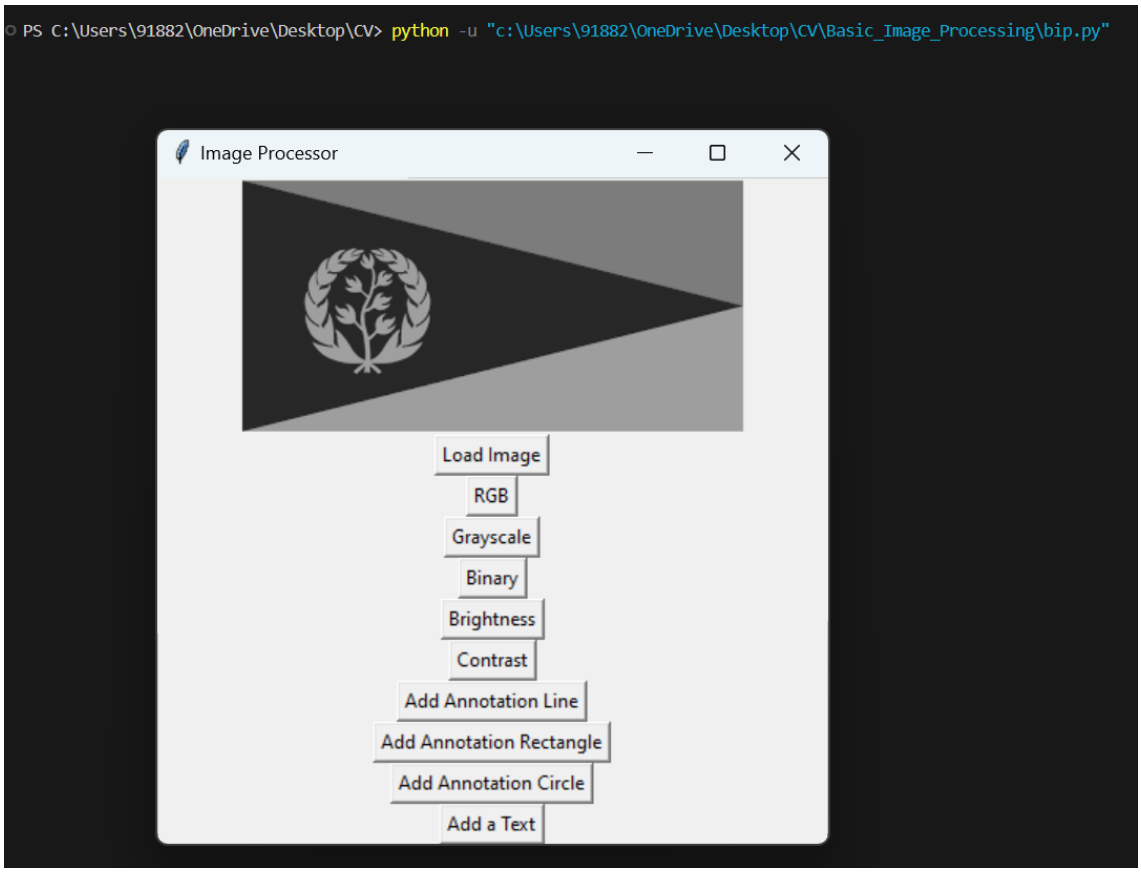
FIRST OF ALL, LOAD THE IMAGE.



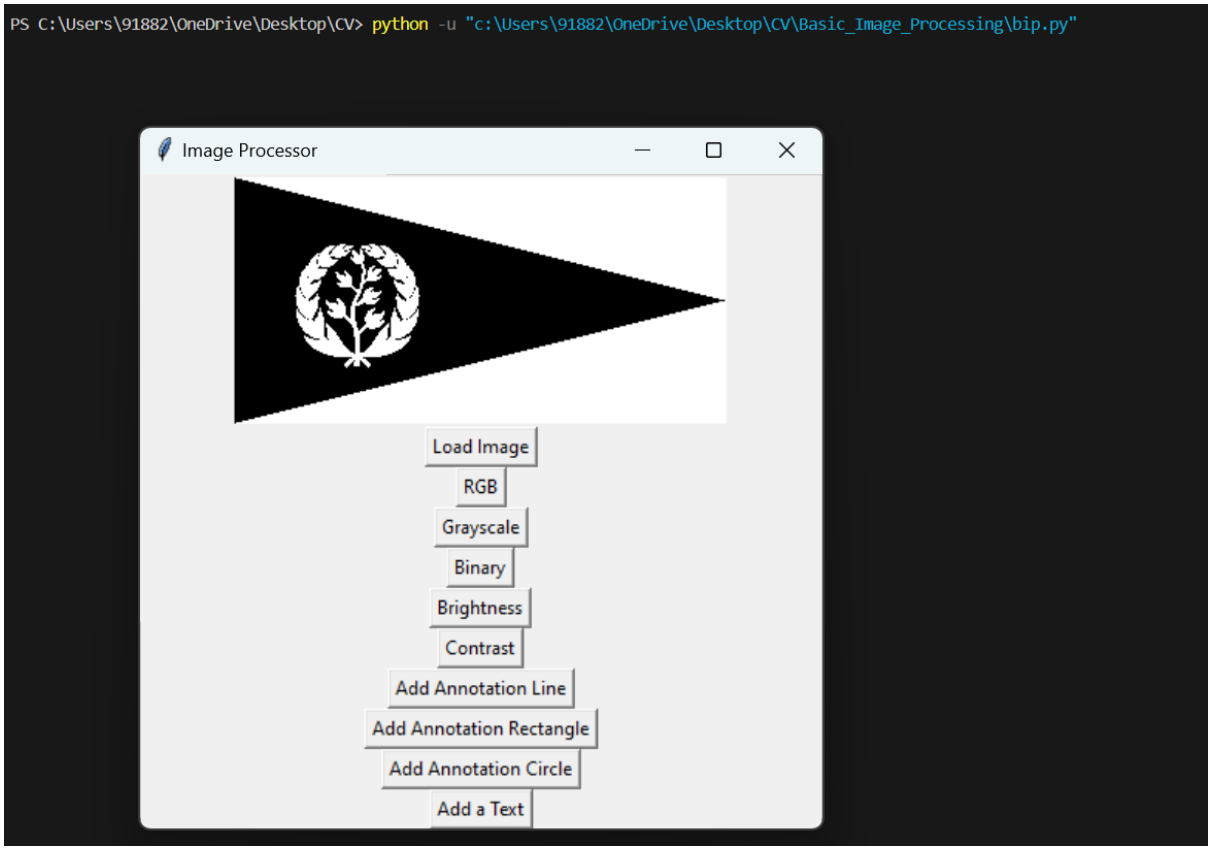
CONVERT INTO RGB IMAGE BY CLICKING ON THE BUTTON OF RGB.



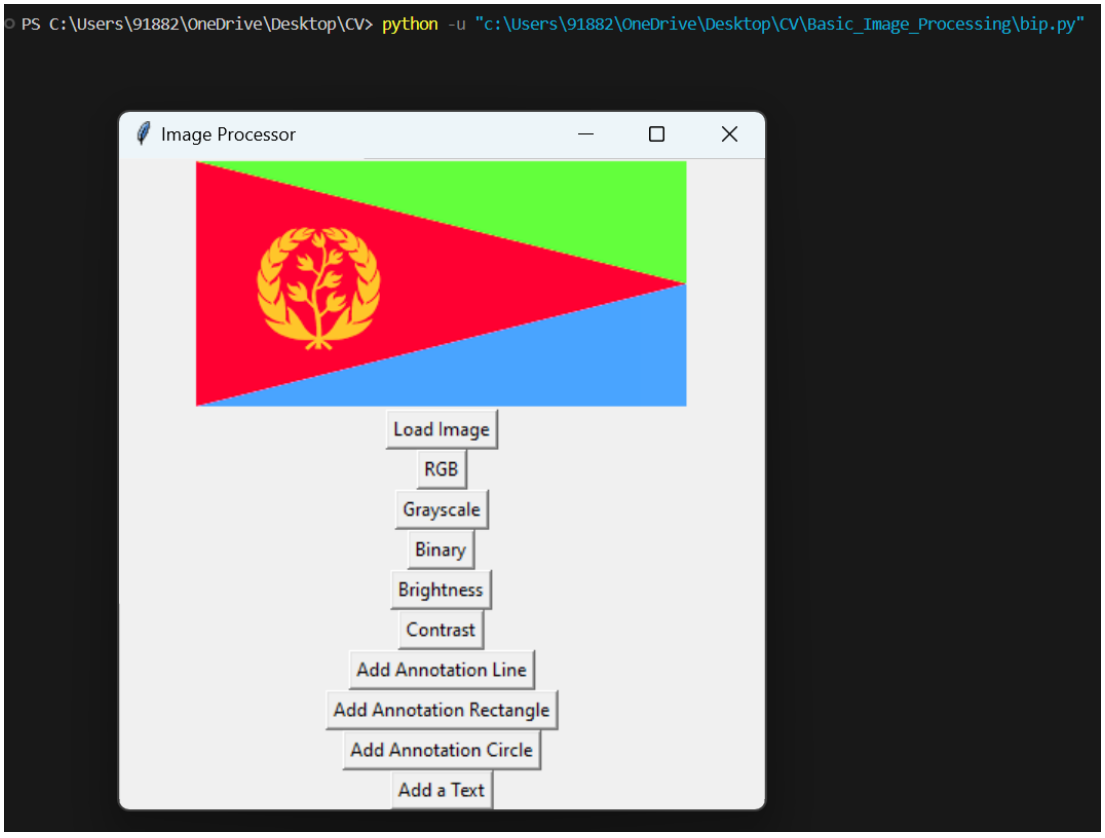
CONVERT INTO GRAYSCALE BY CLICKING ON THE GRAYSCALE BUTTON.



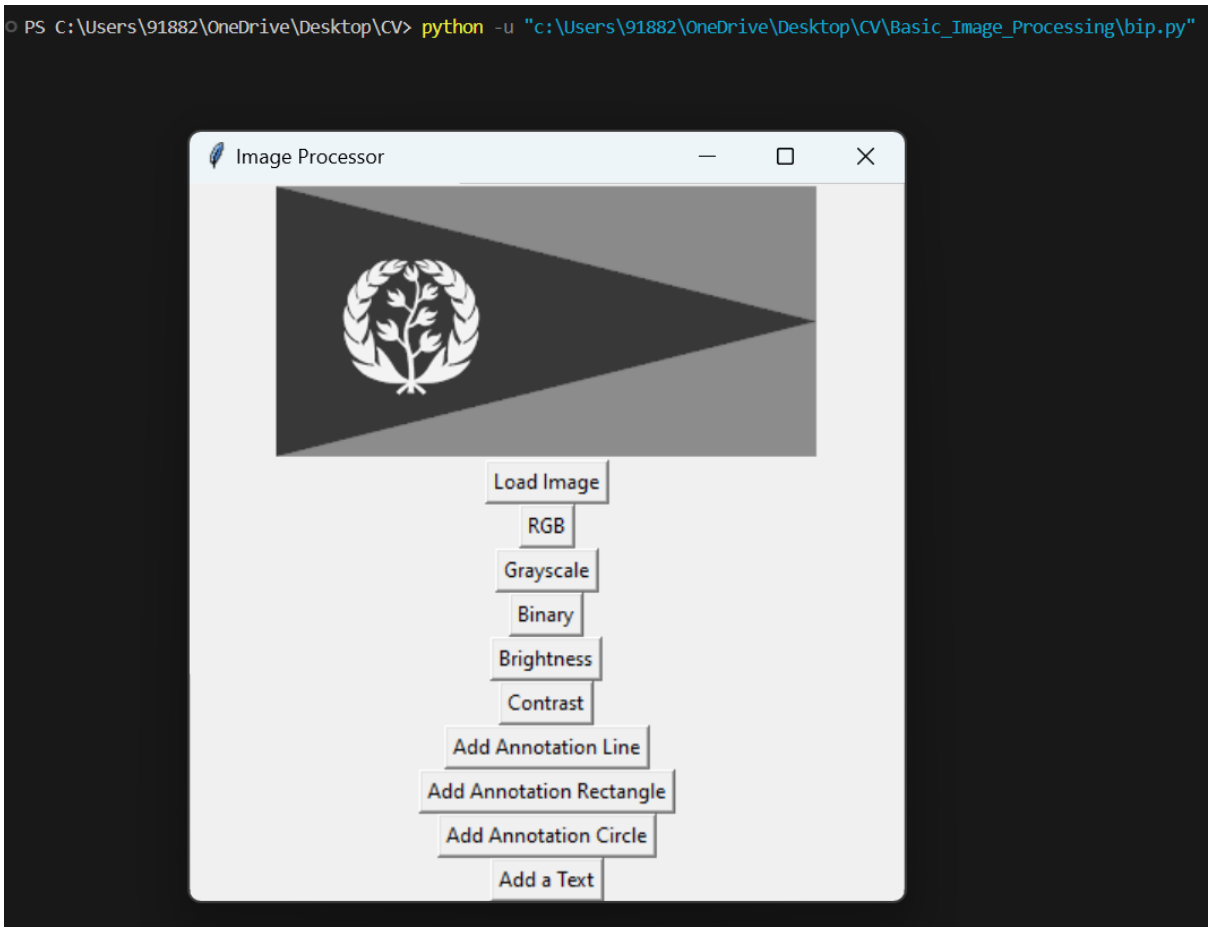
CONVERT INTO BINARY BY CLICKING ON THE BINARY BUTTON.



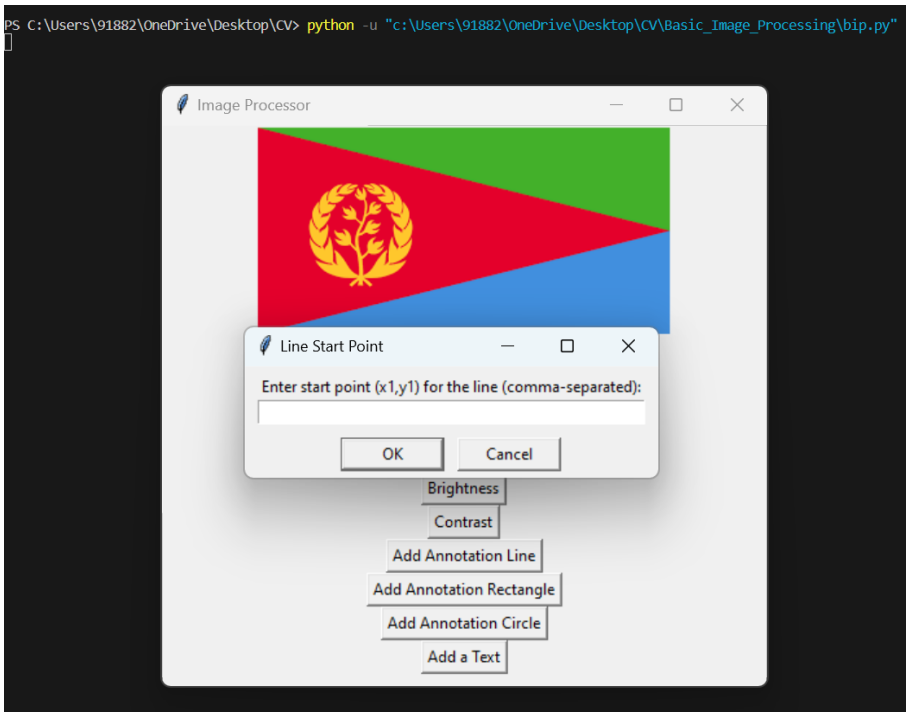
WE CAN ADJUST BRIGHTNESS BY CLICKING REPEATEDLYON BRIGHTNESS BUTTON.



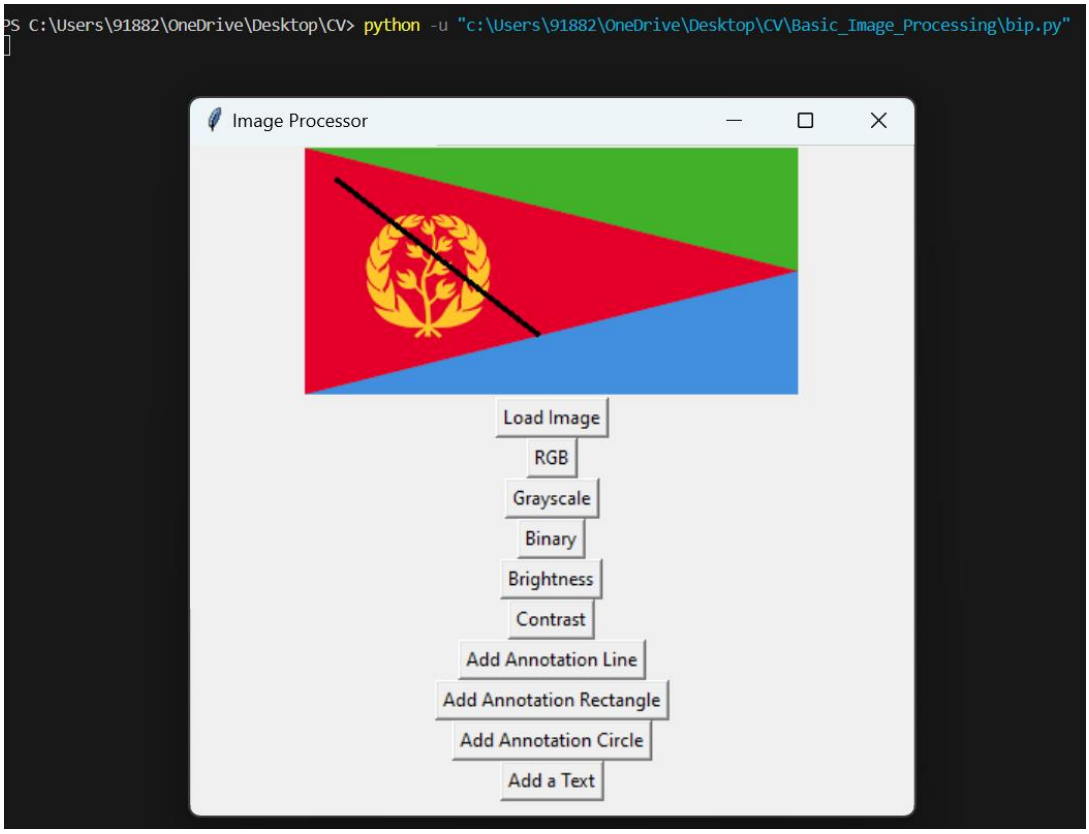
WE CAN ADJUST CONTRAST BY CLICKING REPEATEDLYON CONTRAST BUTTON



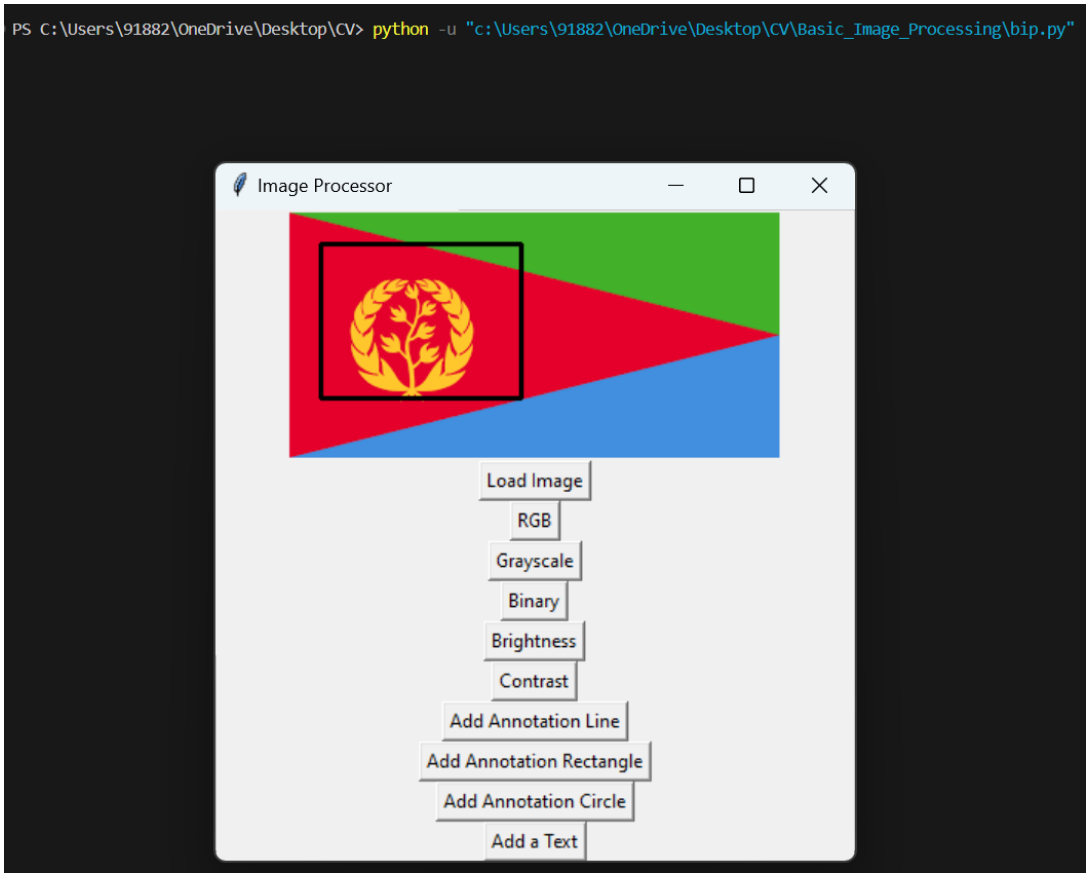
NOW,WE CAN ADD LINE ,RECTANGLE AND CIRCLE AT DESIRED POSITION BY PROVIDING CORDINATES AS:



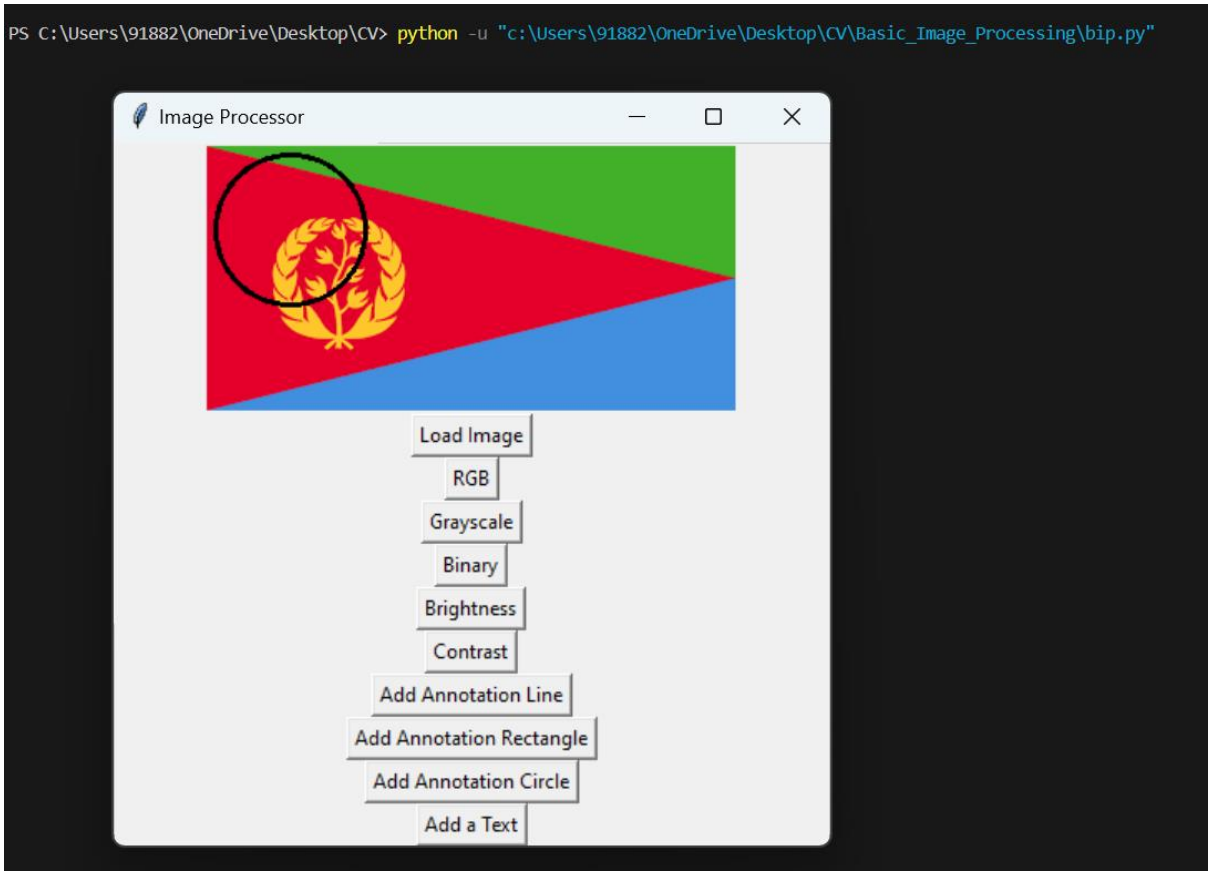
LINE:



RECTANGLE:



CIRCLE:



WE CAN ALSO PUT A TEXT ON DESIRED POSITION BY PROVIDING COORDINATES: