

## 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 = simpdialog.askstring("Line Start Point", "Enter start point (x1,y1) for
the line (comma-separated):")

        end_input = simpdialog.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 = simpdialog.askstring("Rectangle Top-Left", "Enter top-left corner
(x1,y1) for the rectangle (comma-separated):")

```

```
bottom_right_input = simpdialog.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 = simpdialog.askstring("Circle Center", "Enter center point (x,y) for  
the circle (comma-separated):")
```

```
        radius_input = simpdialog.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:**

```
EXPLORER
CV
  Basic_Image_Processing
    bip.py
  .venv
  1.png
  BasicImageProcessing.docx

bip.py
Basic_Image_Processing > bip.py > ImageProcessor > __init__
1  import cv2
2  import numpy as np
3  from tkinter import *
4  from tkinter import filedialog
5  from tkinter import simpledialog, messagebox
6  from PIL import Image, ImageTk
7
8  class ImageProcessor:
9      def __init__(self, root):
10         self.root = root
11         self.root.title("Image Processor")
12
13         self.image_label = Label(root)
14         self.image_label.pack()
15
16         self.btn_load = Button(root, text="Load Image", command=self.load_image)
17         self.btn_load.pack()
18
19         self.btn_rgb = Button(root, text="RGB", command=self.convert_to_rgb)
20         self.btn_rgb.pack()
21
22         self.btn_gray = Button(root, text="Grayscale", command=self.convert_to_gray)
23         self.btn_gray.pack()
24
25         self.btn_binary = Button(root, text="Binary", command=self.convert_to_binary)
26         self.btn_binary.pack()
27
28         self.btn_brightness = Button(root, text="Brightness", command=self.adjust_brightness)
29         self.btn_brightness.pack()
30
31         self.btn_contrast = Button(root, text="Contrast", command=self.adjust_contrast)
32         self.btn_contrast.pack()
33
```

```
EXPLORER
CV
  Basic_Image_Processing
    bip.py
  .venv
  1.png
  BasicImageProcessing.docx

bip.py
Basic_Image_Processing > bip.py > ImageProcessor > __init__
31 self.btn_contrast = Button(root, text="Contrast", command=self.adjust_contrast)
32 self.btn_contrast.pack()
33
34 self.btn_annotationline = Button(root, text="Add Annotation Line", command=self.add_annotation_line)
35 self.btn_annotationline.pack()
36
37 self.btn_annotationrect = Button(root, text="Add Annotation Rectangle", command=self.add_annotation_rectangle)
38 self.btn_annotationrect.pack()
39
40 self.btn_annotationcirc = Button(root, text="Add Annotation Circle", command=self.add_annotation_circle)
41 self.btn_annotationcirc.pack()
42
43 self.btn_annotationputtext = Button(root, text="Add a Text", command=self.add_annotation_puttext)
44 self.btn_annotationputtext.pack()
45
46 self.image = None
47
48 def load_image(self):
49     path = filedialog.askopenfilename()
50     if path:
51         self.image = cv2.imread(path)
52         self.display_image()
53
54 def display_image(self):
55     image_rgb = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)
56     image_pil = Image.fromarray(image_rgb)
57     imageTk = ImageTk.PhotoImage(image_pil)
58     self.image_label.configure(image=imageTk)
59     self.image_label.image = imageTk
60
61 def convert_to_rgb(self):
62     if self.image is not None:
63         self.image = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)
64         self.display_image()
65
```

```
EXPLORER
CV
  Basic_Image_Processing
    bip.py
  .venv
  1.png
  BasicImageProcessing.docx

bip.py
Basic_Image_Processing > bip.py > ImageProcessor > __init__
65
66
67 def convert_to_gray(self):
68     if self.image is not None:
69         gray_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
70         self.image = cv2.cvtColor(gray_image, cv2.COLOR_GRAY2BGR)
71         self.display_image()
72
73 def convert_to_binary(self):
74     if self.image is not None:
75         gray_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
76         binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
77         self.image = cv2.cvtColor(binary_image, cv2.COLOR_GRAY2BGR)
78         self.display_image()
79
80
81 def adjust_brightness(self):
82     if self.image is not None:
83         brightness_value = 50 # Example: adjust brightness by adding 50
84         hsv_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2HSV)
85         h, s, v = cv2.split(hsv_image)
86         v = cv2.add(v, brightness_value)
87         self.image = cv2.merge((h, s, v))
88         self.image = cv2.cvtColor(self.image, cv2.COLOR_HSV2BGR)
89         self.display_image()
90
91
92 def adjust_contrast(self):
93     if self.image is not None:
94         contrast_factor = 1.5 # Example: adjust contrast by multiplying by 1.5
95         gray_image = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
96         mean_intensity = np.mean(gray_image)
97         adjusted_image = cv2.convertScaleAbs(self.image, alpha=contrast_factor, beta=-mean_intensity*(contrast_factor-1))
98         self.image = cv2.cvtColor(adjusted_image, cv2.COLOR_BGR2GRAY)
99         self.display_image()
100
```

```
EXPLORER    ...    bip.py  X
CV
  Basic_Image_Processing
    bip.py
    venv
    1.png
    BasicImageprocessing.docx

Basic_Image_Processing > bip.py > ImageProcessor > __init__
101
102 def add_annotation_line(self):
103     if self.image is not None:
104         start_input = simpledialog.askstring("Line Start Point", "Enter start point (x1,y1) for the line (comma-separated):")
105         end_input = simpledialog.askstring("Line End Point", "Enter end point (x2,y2) for the line (comma-separated):")
106         if start_input and end_input:
107             try:
108                 x1, y1 = map(int, start_input.split(','))
109                 x2, y2 = map(int, end_input.split(','))
110                 image_copy = self.image.copy()
111                 cv2.line(image_copy, (x1, y1), (x2, y2), (0, 0, 0), 2)
112                 self.image = image_copy
113                 self.display_image()
114             except ValueError:
115                 messagebox.showerror("Error", "Invalid input. Please enter coordinates in the format 'x,y'.")
116
117 def add_annotation_rectangle(self):
118     if self.image is not None:
119         top_left_input = simpledialog.askstring("Rectangle Top-Left", "Enter top-left corner (x1,y1) for the rectangle (comma-separated):")
120         bottom_right_input = simpledialog.askstring("Rectangle Bottom-Right", "Enter bottom-right corner (x2,y2) for the rectangle (comma-separated):")
121         if top_left_input and bottom_right_input:
122             try:
123                 x1, y1 = map(int, top_left_input.split(','))
124                 x2, y2 = map(int, bottom_right_input.split(','))
125                 image_copy = self.image.copy()
126                 cv2.rectangle(image_copy, (x1, y1), (x2, y2), (0, 0, 0), 2)
127                 self.image = image_copy
128                 self.display_image()
129             except ValueError:
130                 messagebox.showerror("Error", "Invalid input. Please enter coordinates in the format 'x,y'.")
131
```

```
EXPLORER    ...    bip.py  X
CV
  Basic_Image_Processing
    bip.py
    venv
    1.png
    BasicImageprocessing.docx

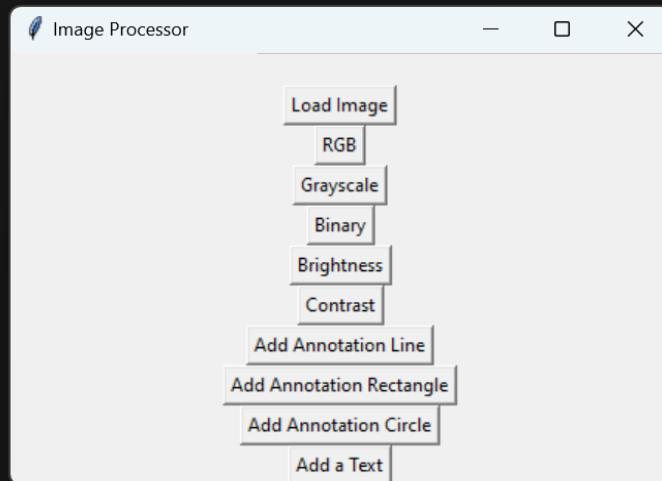
Basic_Image_Processing > bip.py > ImageProcessor > __init__
132
133 def add_annotation_circle(self):
134     if self.image is not None:
135         center_input = simpledialog.askstring("Circle Center", "Enter center point (x,y) for the circle (comma-separated):")
136         radius_input = simpledialog.askinteger("Circle Radius", "Enter radius for the circle:")
137         if center_input and radius_input:
138             try:
139                 x, y = map(int, center_input.split(','))
140                 image_copy = self.image.copy()
141                 cv2.circle(image_copy, (x, y), radius_input, (0, 0, 0), 2)
142                 self.image = image_copy
143                 self.display_image()
144             except ValueError:
145                 messagebox.showerror("Error", "Invalid input. Please enter coordinates in the format 'x,y'.")
146
147 def add_annotation_puttext(self):
148     if self.image is not None:
149         position_input = simpledialog.askstring("Text Position", "Enter position (x,y) for the text (comma-separated):")
150         if position_input:
151             try:
152                 x, y = map(int, position_input.split(','))
153                 image_copy = self.image.copy()
154                 cv2.putText(image_copy, "ANIRUDH SHUKLA", (x, y), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)
155                 self.image = image_copy
156                 self.display_image()
157             except ValueError:
158                 messagebox.showerror("Error", "Invalid input. Please enter coordinates in the format 'x,y'.")
159
160
161 if __name__ == "__main__":
162     root = Tk()
163     app = ImageProcessor(root)
164     root.mainloop()
165
```

**OUTPUT SCREENSHOT:**

**GUI:**

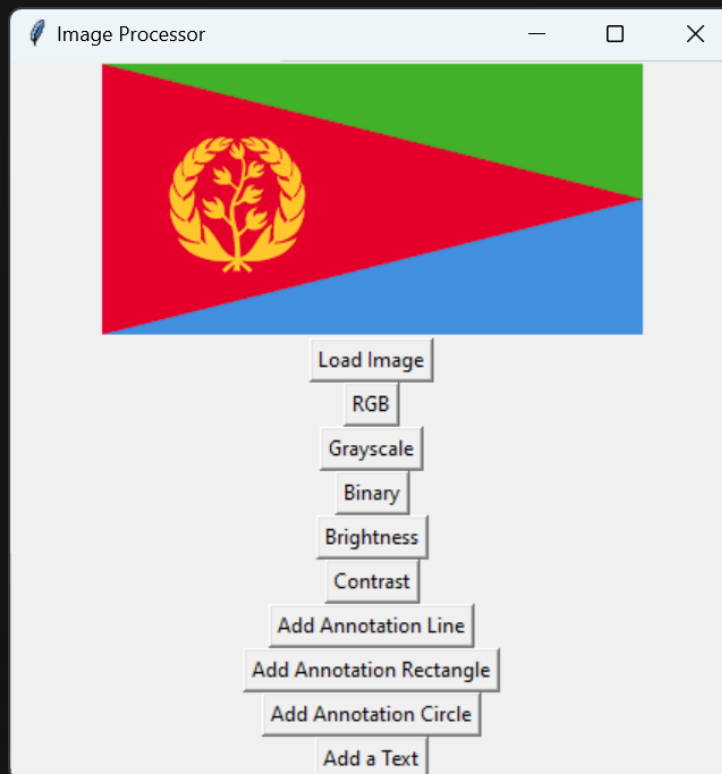


```
PS C:\Users\91882\OneDrive\Desktop\CV> python -u "c:\Users\91882\OneDrive\Desktop\CV\Basic_Image_Processing\bip.py"
```



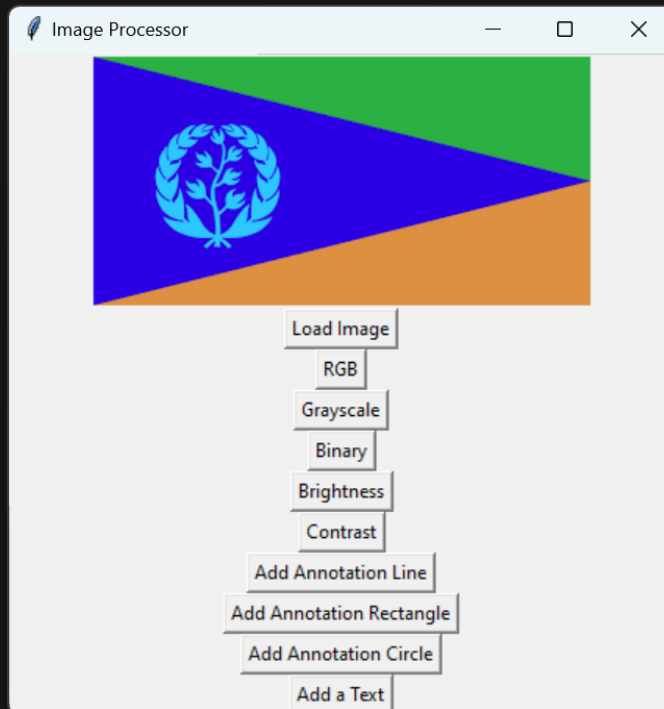
FIRST OF ALL, LOAD THE IMAGE.

```
PS C:\Users\91882\OneDrive\Desktop\CV> python -u "c:\Users\91882\OneDrive\Desktop\CV\Basic_Image_Processing\bip.py"
```



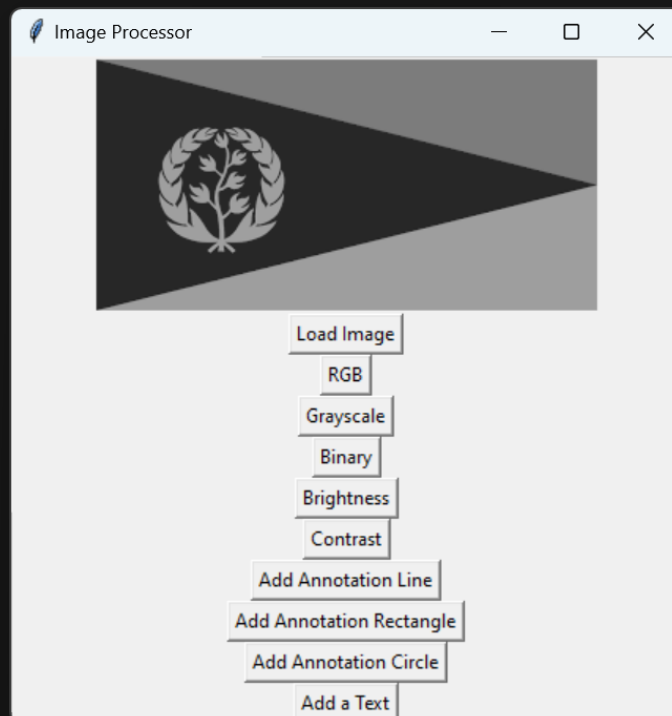
CONVERT INTO RGB IMAGE BY CLICKING ON THE BUTTON OF RGB.

```
PS C:\Users\91882\OneDrive\Desktop\CV> python -u "c:\Users\91882\OneDrive\Desktop\CV\Basic_Image_Processing\bip.py"
```



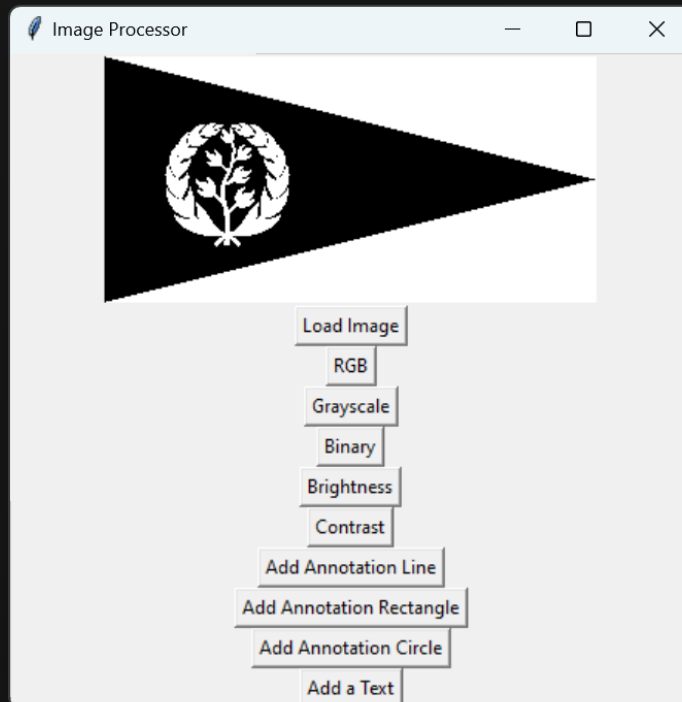
CONVERT INTO GRAYSCALE BY CLICKING ON THE GRAYSCALE BUTTON.

```
PS C:\Users\91882\OneDrive\Desktop\CV> python -u "c:\Users\91882\OneDrive\Desktop\CV\Basic_Image_Processing\bip.py"
```



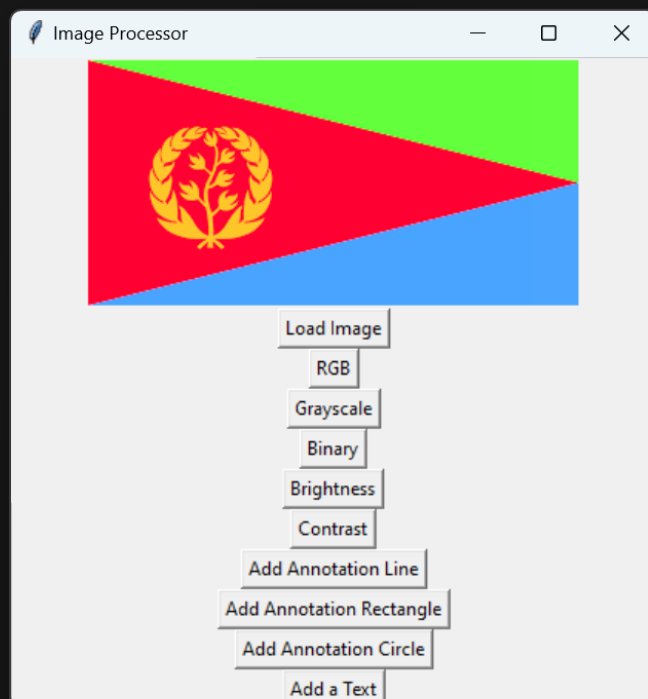
CONVERT INTO BINARY BY CLICKING ON THE BINARY BUTTON.

```
PS C:\Users\91882\OneDrive\Desktop\CV> python -u "c:\Users\91882\OneDrive\Desktop\CV\Basic_Image_Processing\bip.py"
```

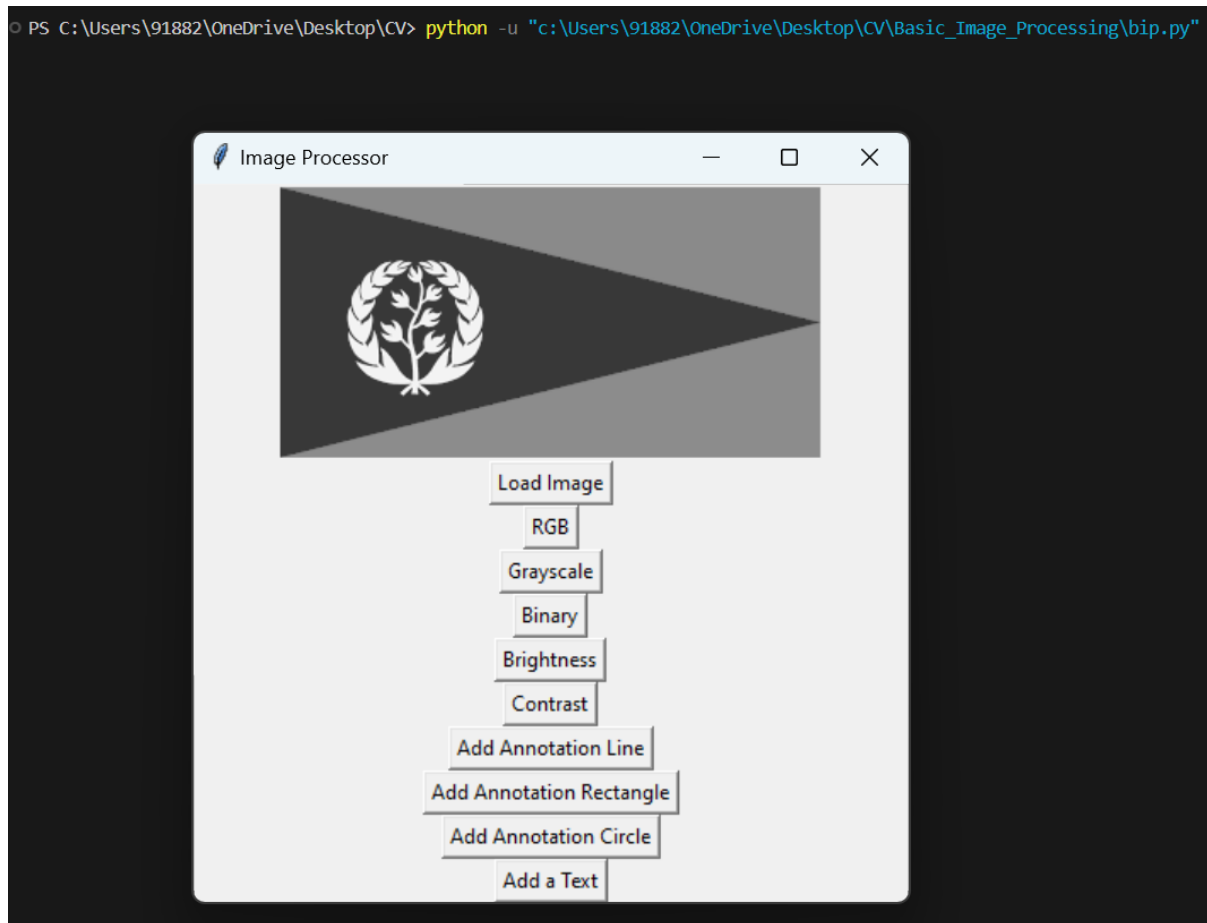


WE CAN ADJUST BRIGHTNESS BY CLICKING REPEATEDLY ON BRIGHTNESS BUTTON.

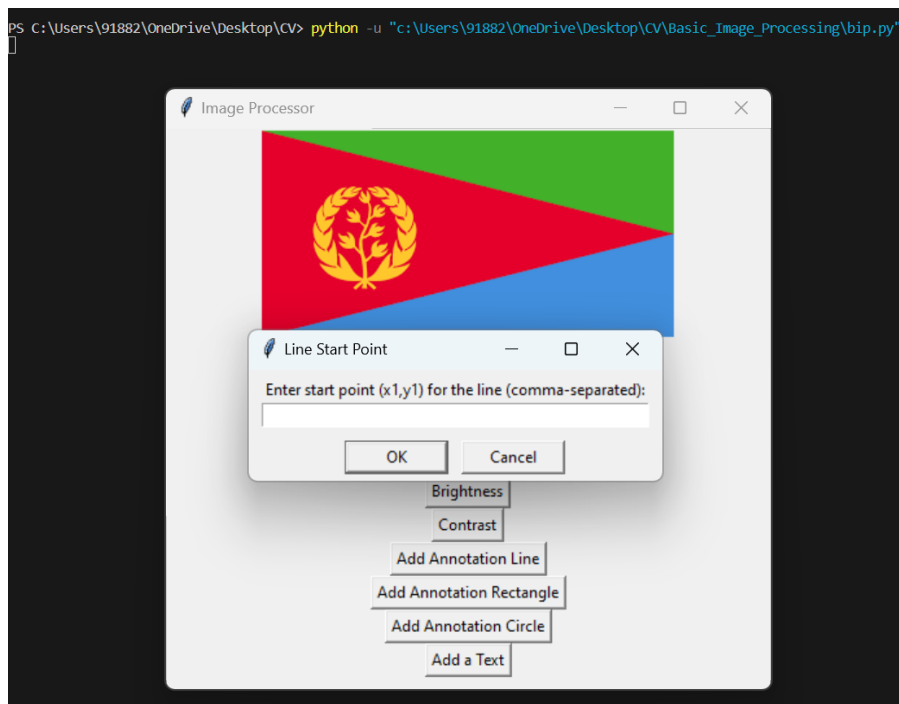
```
PS C:\Users\91882\OneDrive\Desktop\CV> python -u "c:\Users\91882\OneDrive\Desktop\CV\Basic_Image_Processing\bip.py"
```



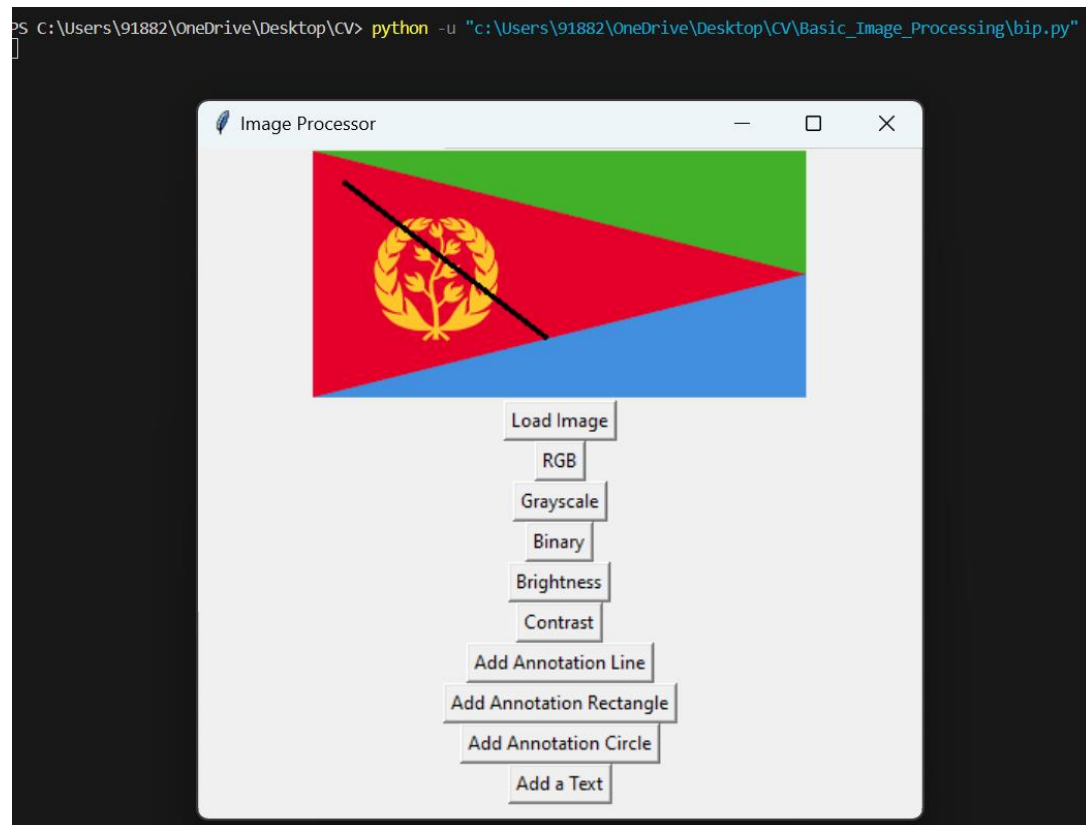
WE CAN ADJUST CONTRAST BY CLICKING REPEATEDLY ON CONTRAST  
BUTTON



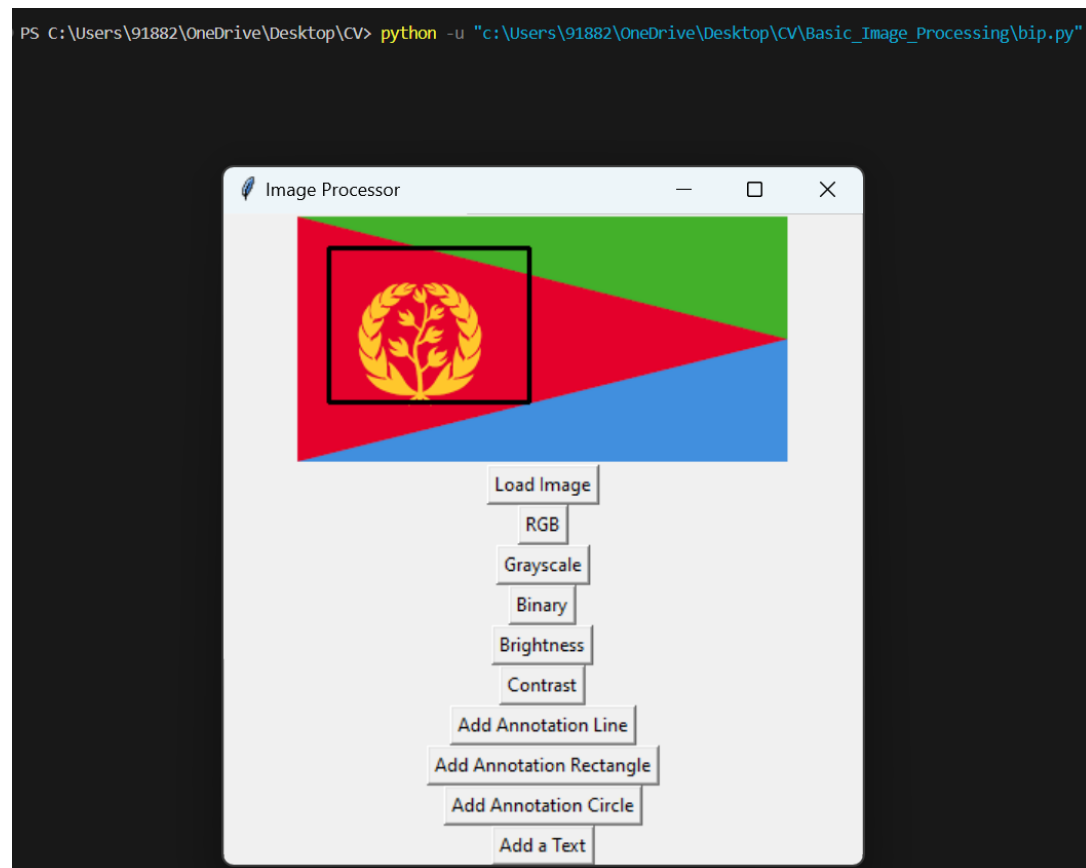
NOW,WE CAN ADD LINE ,RECTANGLE AND CIRCLE AT DESIRED POSITION BY  
PROVIDING CORDINATES AS:



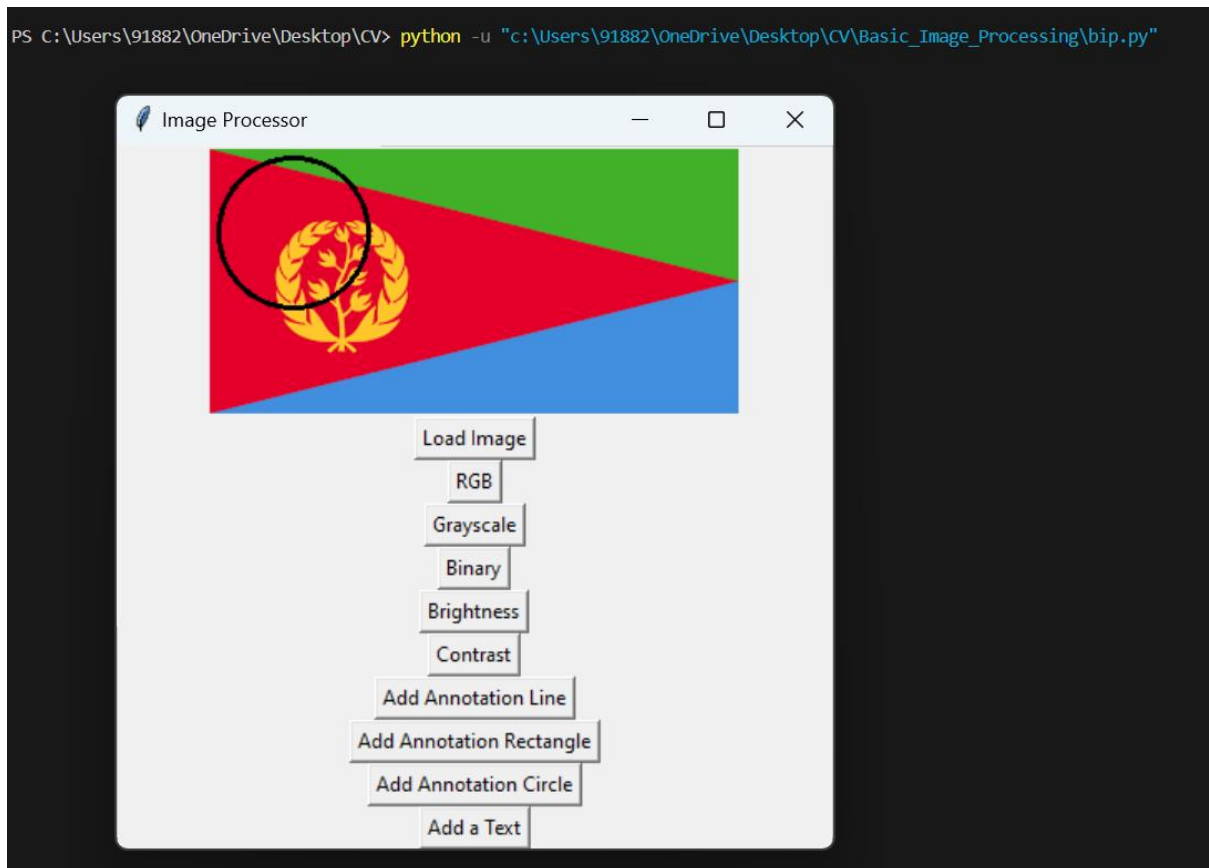
LINE:



RECTANGLE:



CIRCLE:



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

