

PRICE

CALCULATOR

OBJECTIVES

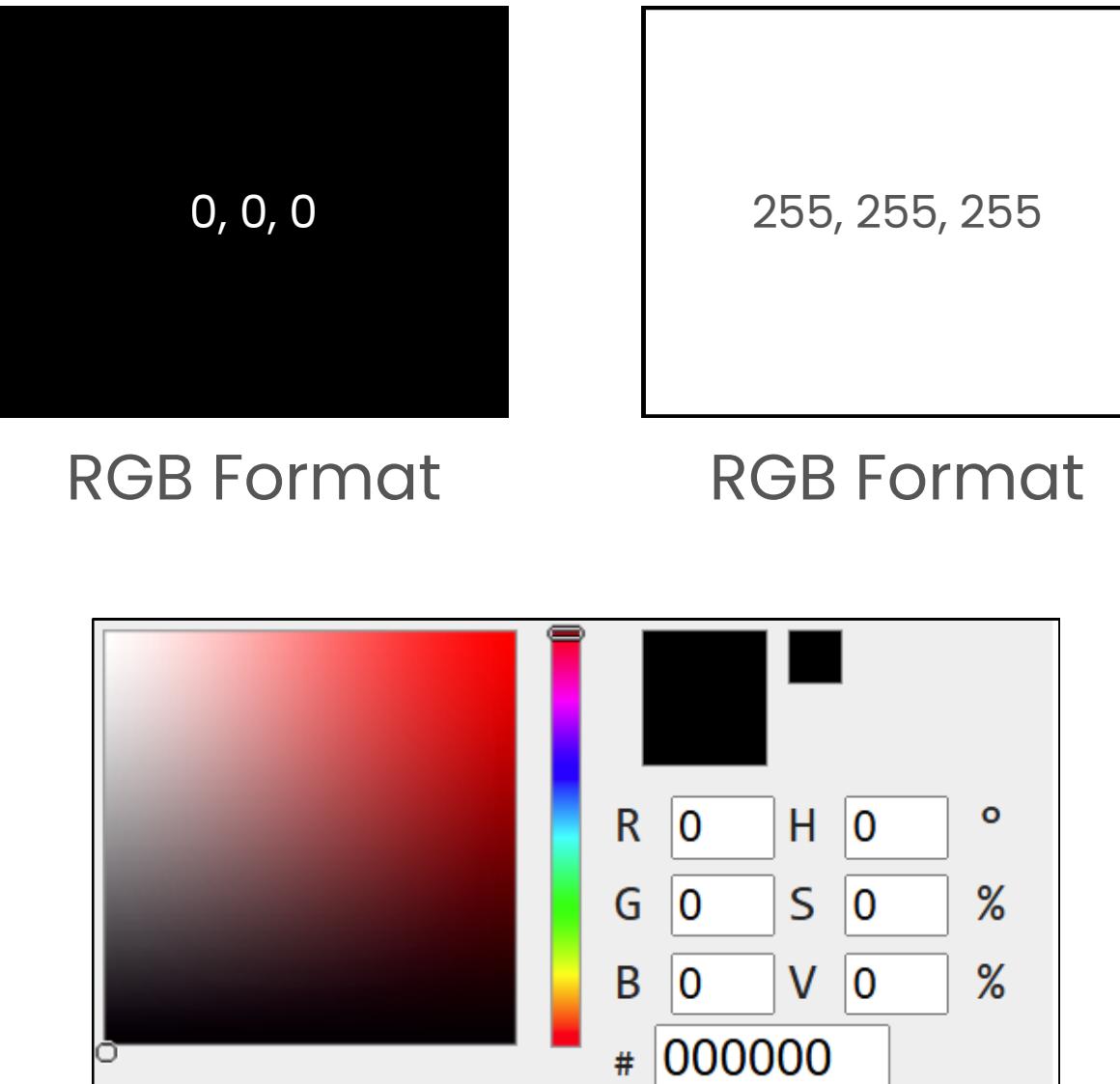
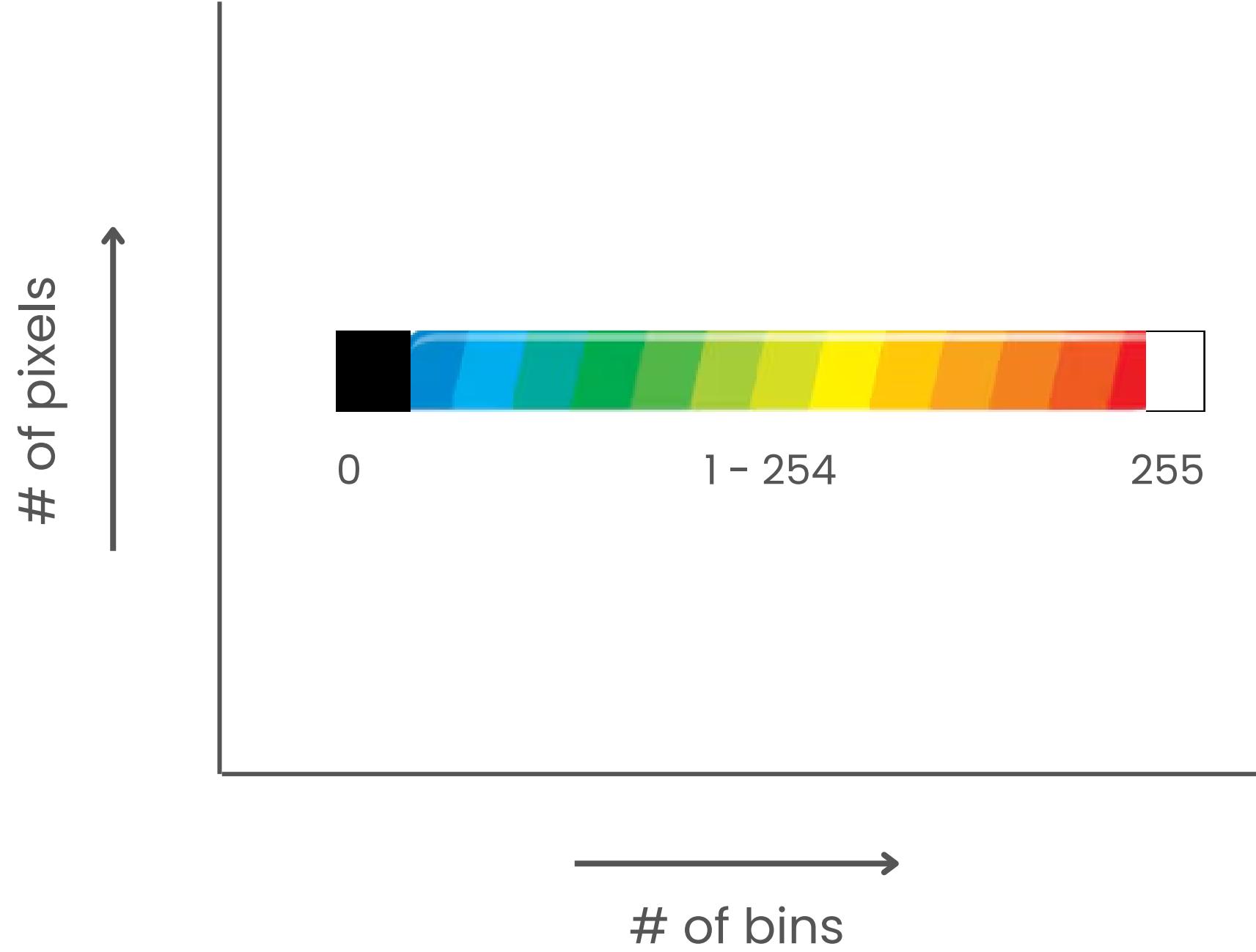
- Solve the **dilemma** of every student or customer when they want to print files – with the use of a **histogram**

PRICE RANGE

- ₱5.00 - Black Text Color
- ₱5.01 → ₱19.99 - Colored
- ₱20.00 - Whole Page Colored

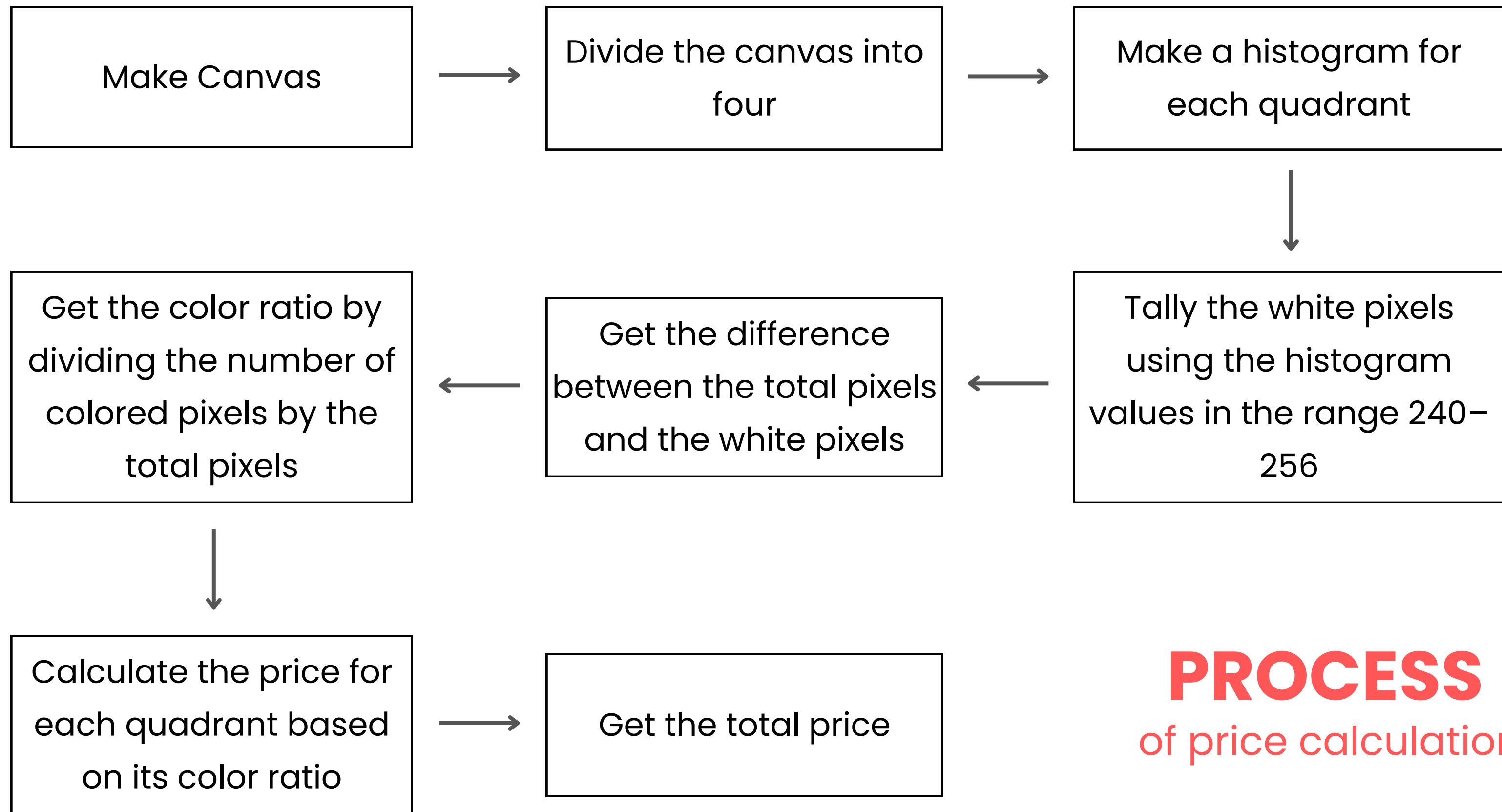
Note: ₱5.00 - ₱20.00 Price Range

HISTOGRAM



Most of the students in MSU-IIT (or even in other universities) experienced going to computer cafes to print their documents. Typically, a page with black text costs 2 pesos. However, if a student wants to print a colored page, the teller in the cafe will put up a price based on how large or small the colored portions are on the page. If the colored portion on the page is around a quarter, the price will be 5 pesos. If it is around half the page, the price will be around 10 pesos. If the whole page contains colored figures, the price will go up to 20 pesos. As a student-customer, there will be instances that we find the heuristic determination of the price "unfair" due to a lack of standards in the computation.

To address this dilemma, you will create a solution to calculate the price of the colored page according to the distribution of colors (or color histograms). Let us assume that the image resolution will be 600 by 900 pixels. Generate sample images for your test bench. Create a strategy on how to price the images. For uniformity, the minimum price will be 5 pesos and the maximum price will be 20 pesos. Also, your calculated price must be in two decimal places only, such as 11.23, 10.02, 5.00, etc.



PROCESS
of price calculation

Make canvas

```
image = cv2.imread(args["image"])
image = cv2.resize(image, (900, 600))
cv2.imshow("Original", image)
```

height = 600

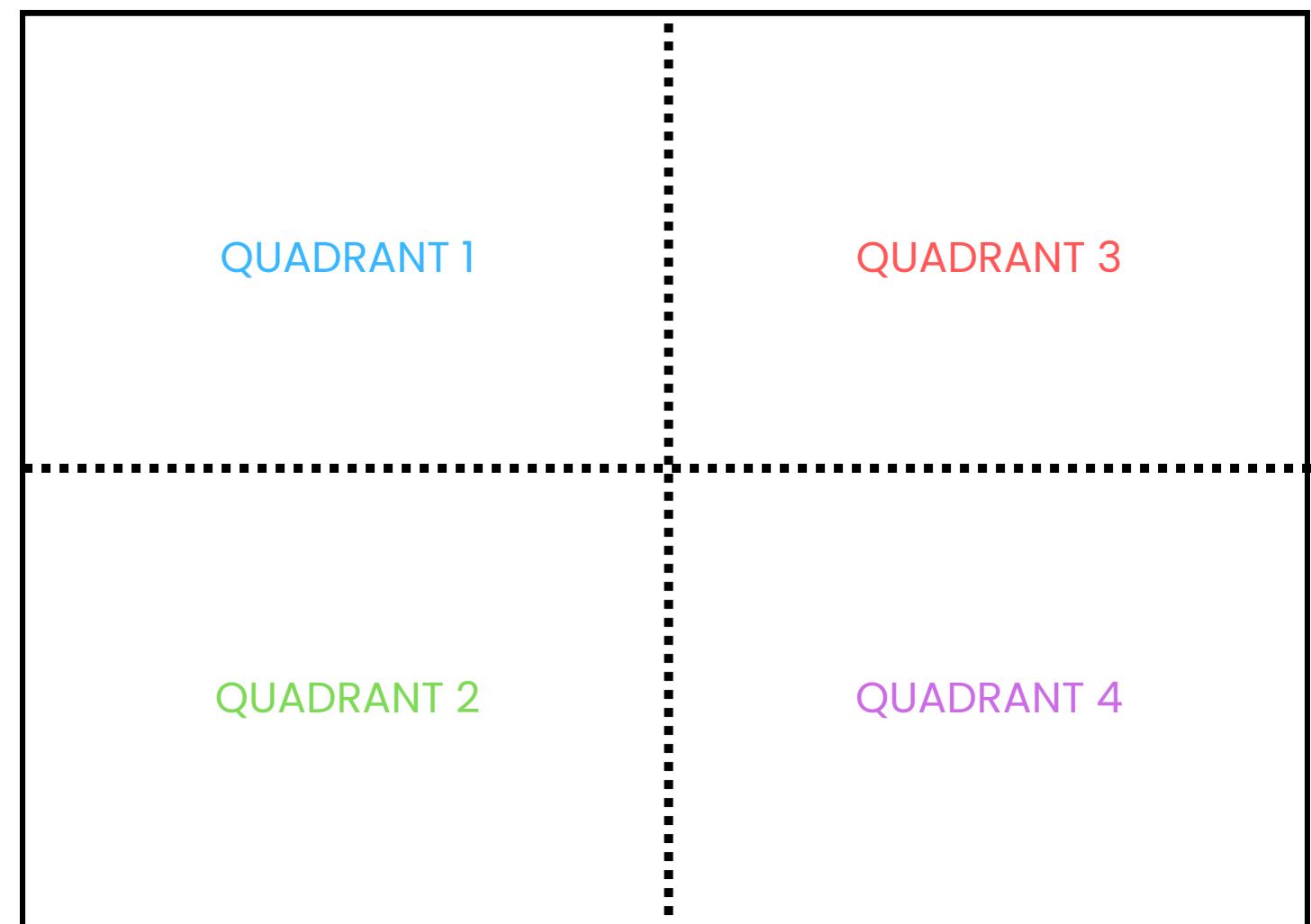


width = 900

Divide into 4

```
quadrants= {  
    "Quadrant 1 (Top-Left)": image[0:300, 0:450],  
    "Quadrant 2 (Bottom-Left)": image[300:600, 0:450],  
    "Quadrant 3 (Top-Right)": image[0:300, 450:900],  
    "Quadrant 4 (Bottom-Right)": image[300:600, 450:900]  
}
```

```
image[0:300, 0:450] = (225, 0, 0)  
image[300:600, 0:450] = (0, 255, 0)  
image[0:300, 450:900] = (0, 0, 225)  
image[300:600, 450:900] = (225, 0, 225)
```

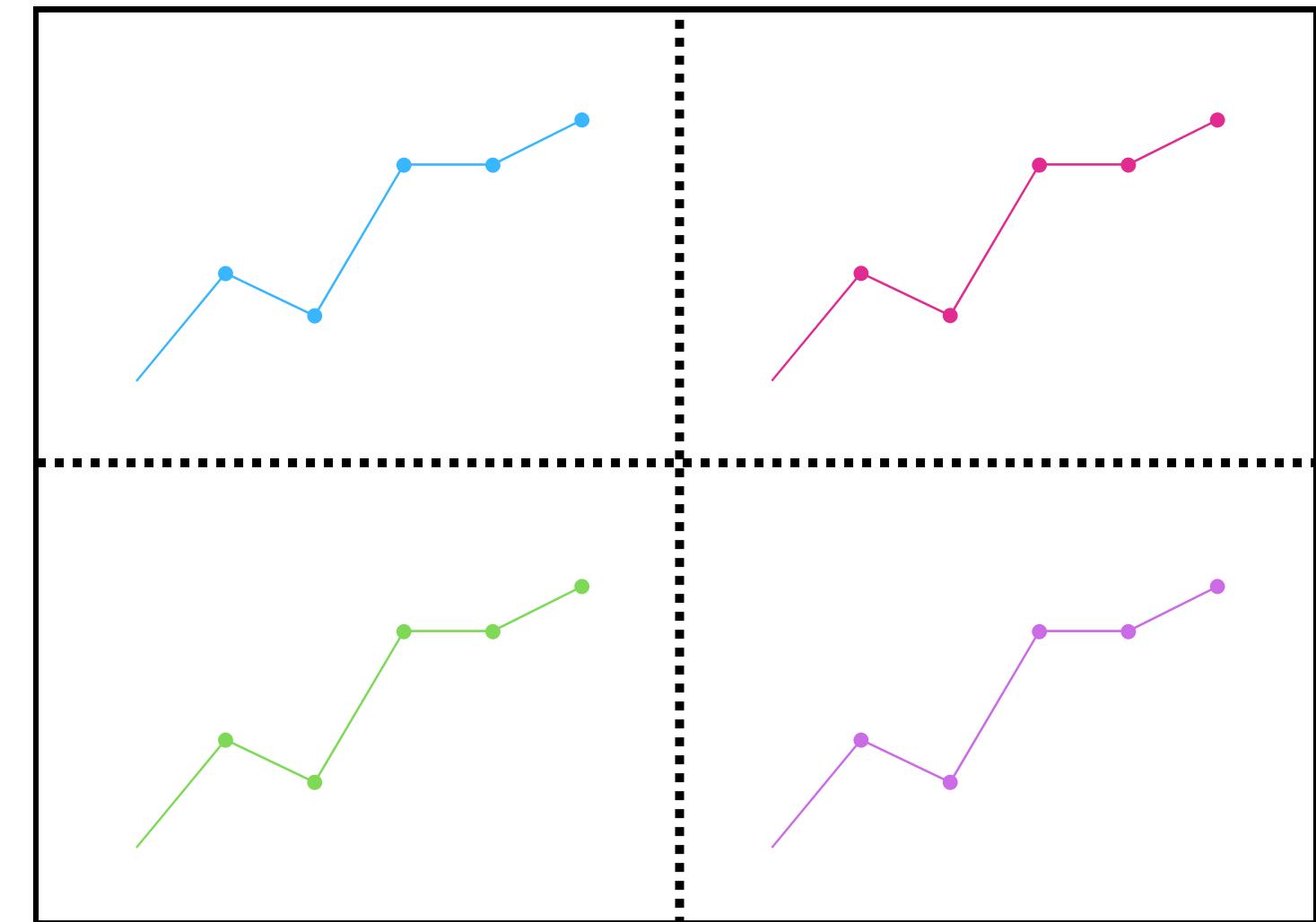


Make a histogram

```
chans = cv2.split(corner)
white_pixels = 0

plt.figure()
plt.title("Flattened ColorHistogram")
plt.xlabel("Bins")
plt.ylabel("# of Pixels")

for (chan, color) in zip (chans, ("b", "g", "r")):
    hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
    white_pixels += hist[240:256].sum()
    plt.plot(hist, color=color)
plt.xlim([0, 256])
```



Price calculation

```
white_pixels /= 3.0
```

```
total_pixels = corner.shape[0] * corner.shape[1]
```

```
colored_pixels = total_pixels - white_pixels
```

```
color_ratio = colored_pixels / total_pixels
```

```
price = 1.25 + color_ratio * (5.0 - 1.25)
```

```
price = round(price, 2)
```

```
print(f"title} = Price: {price:.2f}")
```

```
return price
```

Price Range

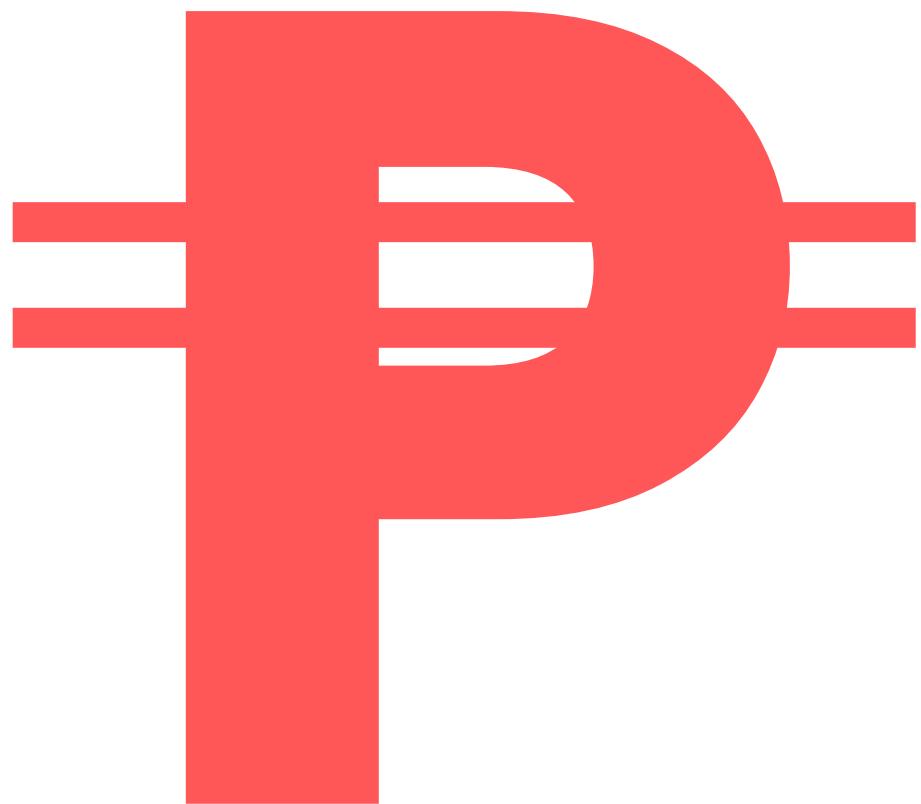
- ₱5.00 – ₱20.00

$$\frac{20}{4} = 5 \longrightarrow \frac{5}{4} = 1.25$$

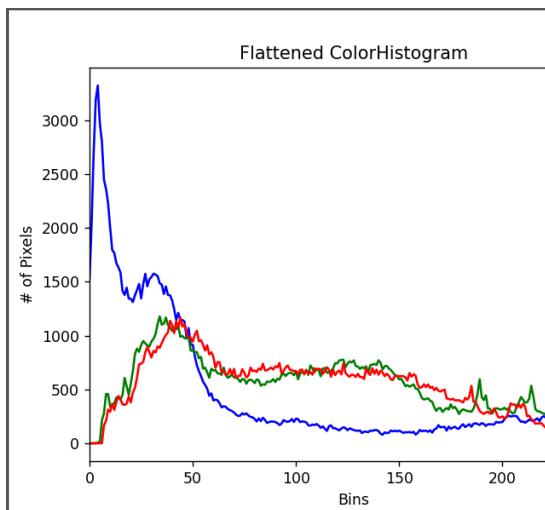
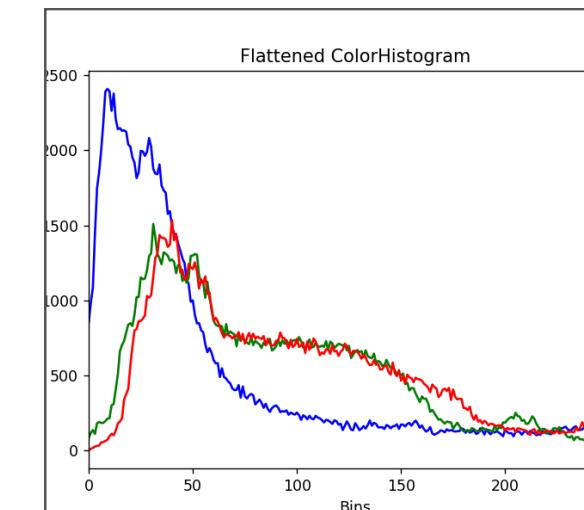
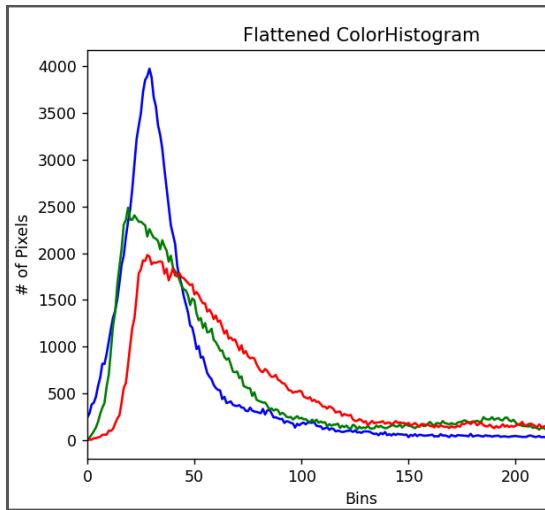
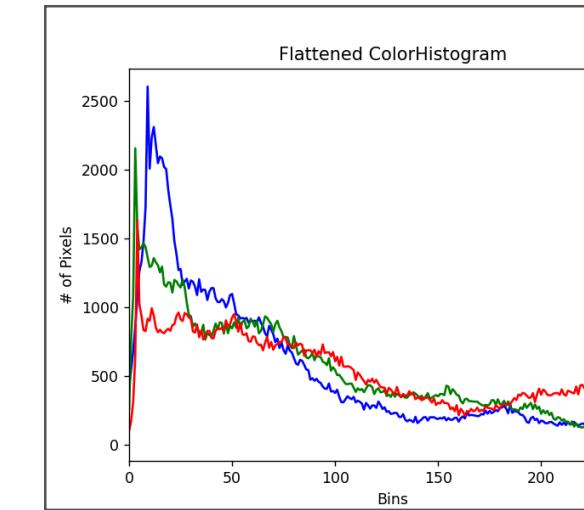
Total Price

```
for name, corner in quadrants.items():
    price = price_calculator(corner, name)
    total_price += price

print(f"\nTOTAL PRICE for the page = {total_price:.2f} pesos")
```



TESTBENCH





TESTBENCH 1

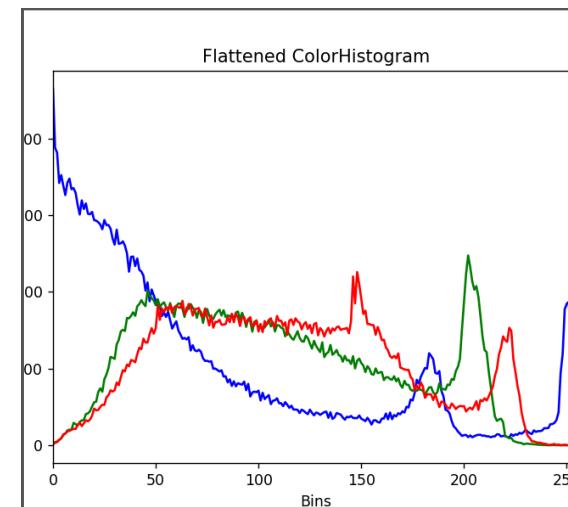
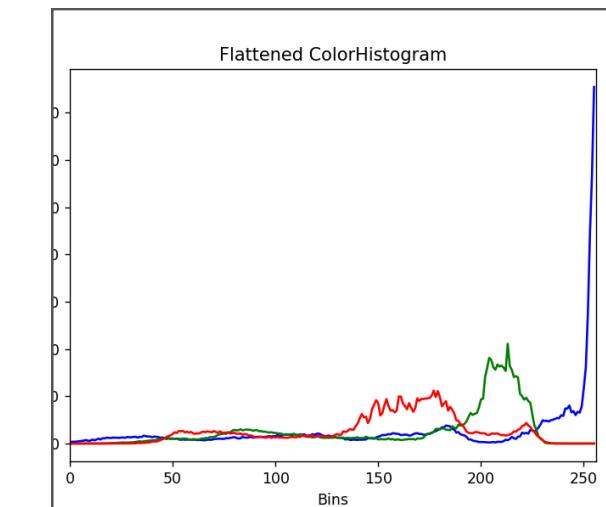
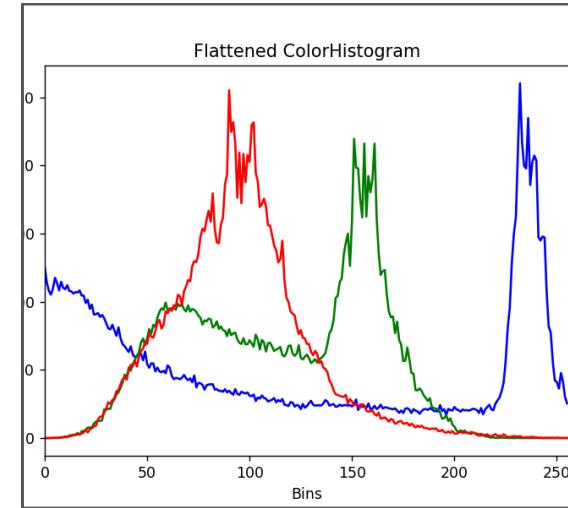
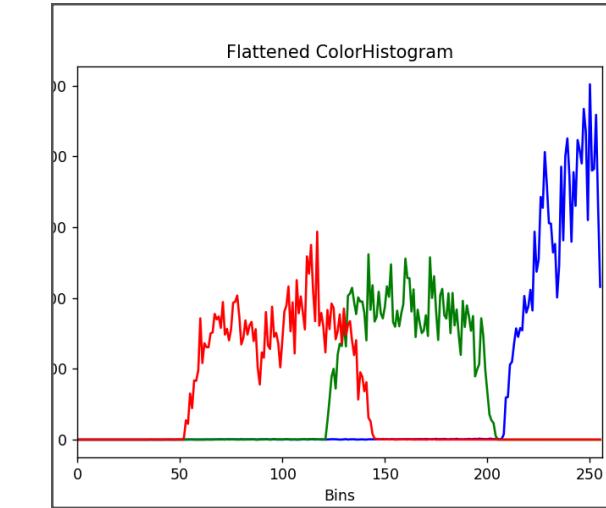
Quadrant 1 (Top-Left) = Price: 4.94

Quadrant 2 (Bottom-Left) = Price: 4.95

Quadrant 3 (Top-Right) = Price: 4.96

Quadrant 4 (Bottom-Right) = Price: 4.8

TOTAL PRICE for the page = 19.72 pesos





TESTBENCH 2

Quadrant 1 (Top-Left) = Price: 4.44

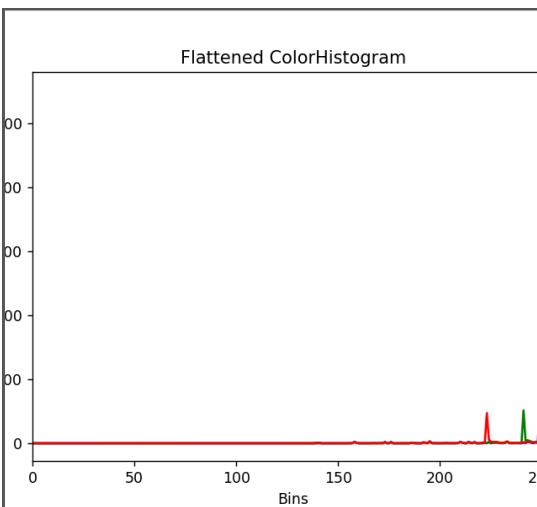
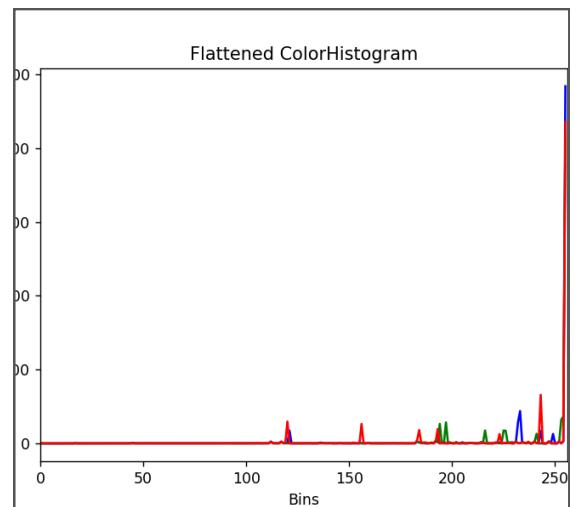
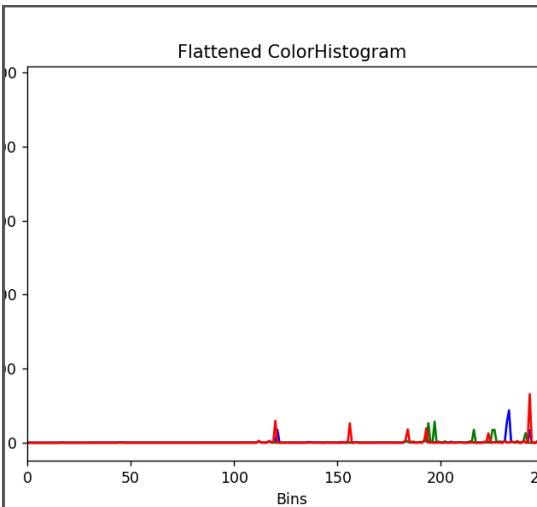
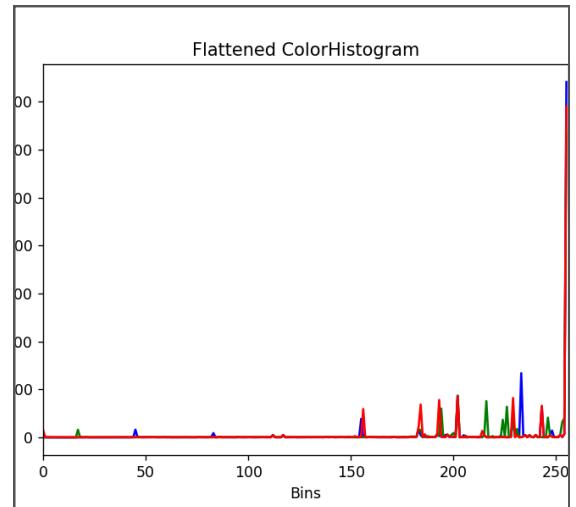
Quadrant 2 (Bottom-Left) = Price: 4.44

Quadrant 3 (Top-Right) = Price: 4.87

Quadrant 4 (Bottom-Right) = Price: 4.94

TOTAL PRICE for the page = 18.69 pesos

Six Sigma Document Control Plan Template



Six Sigma Document Control Plan Template

TESTBENCH 3

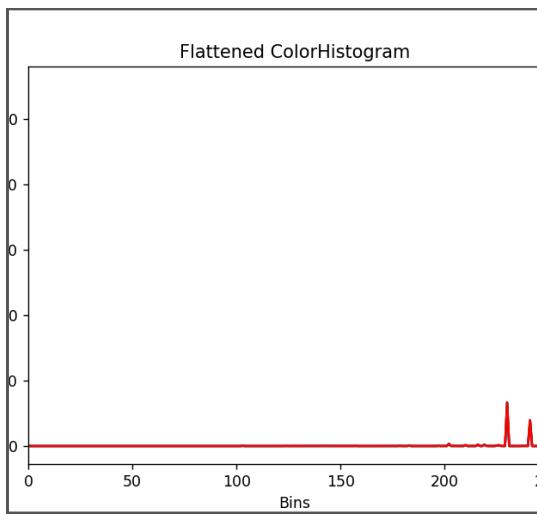
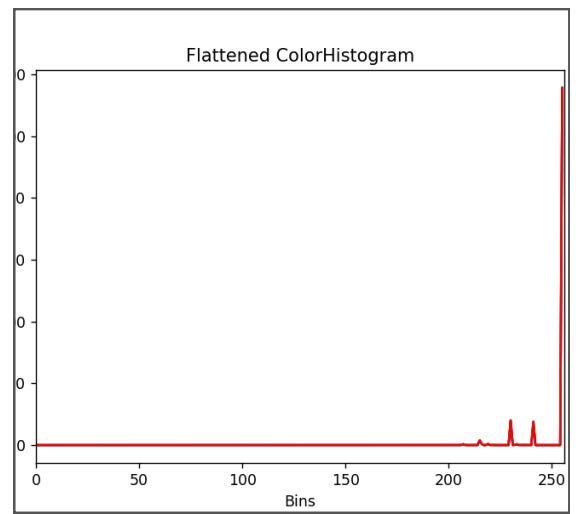
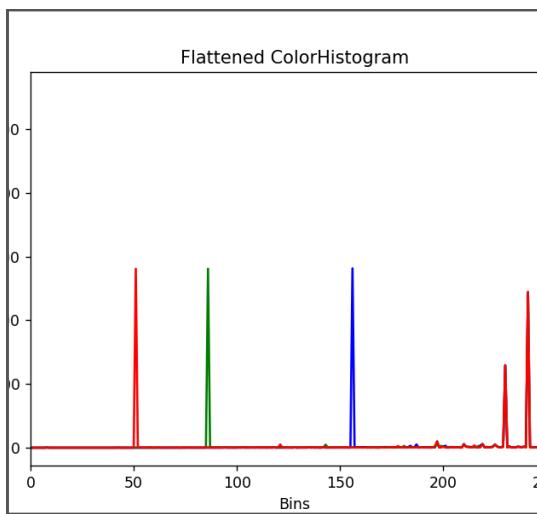
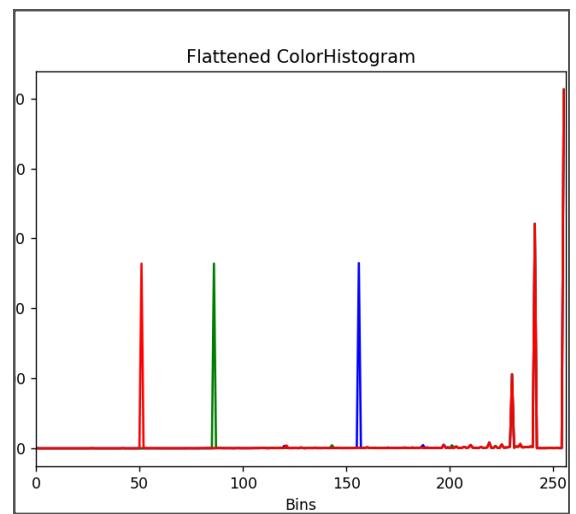
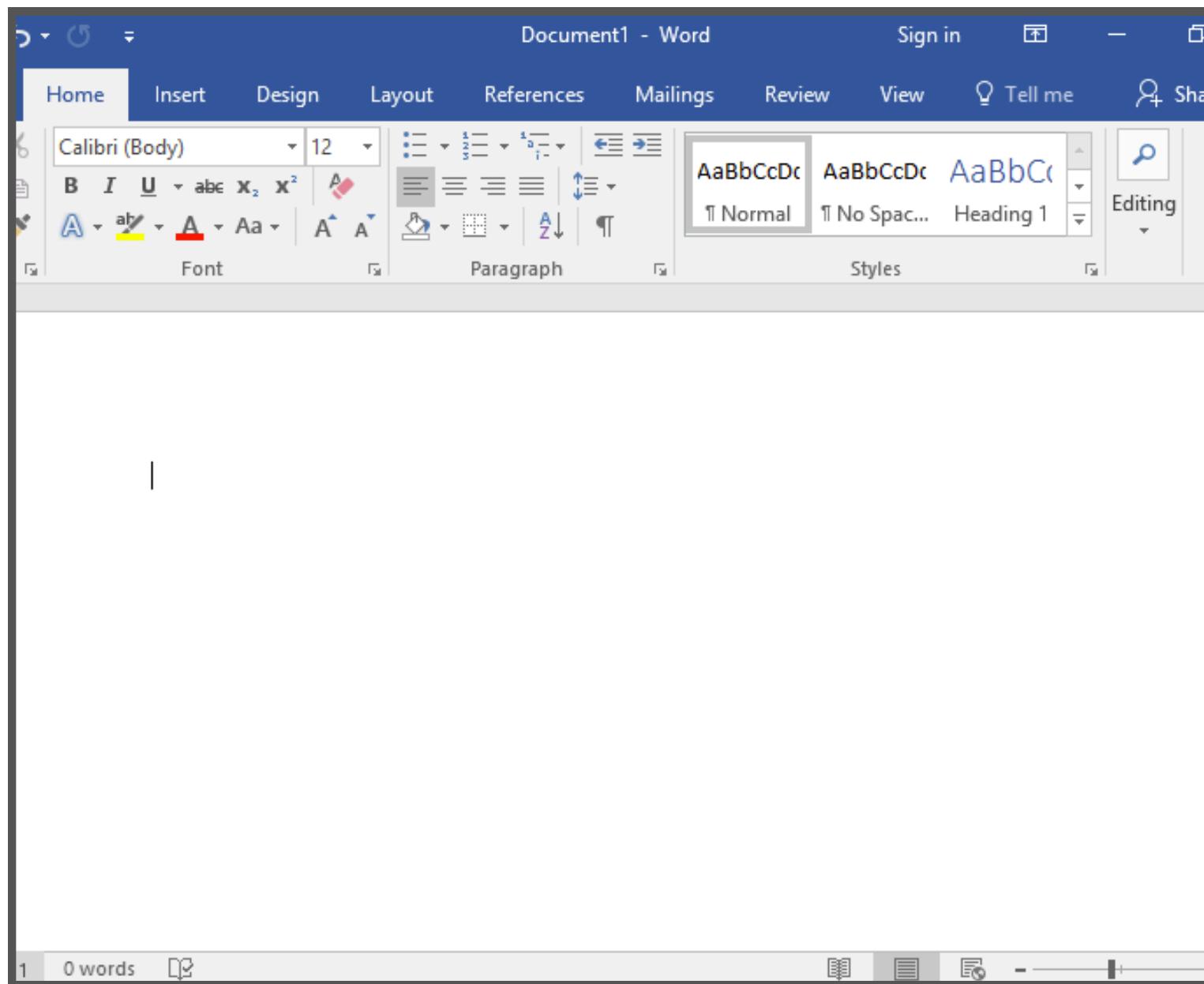
Quadrant 1 (Top-Left) = Price: 2.77

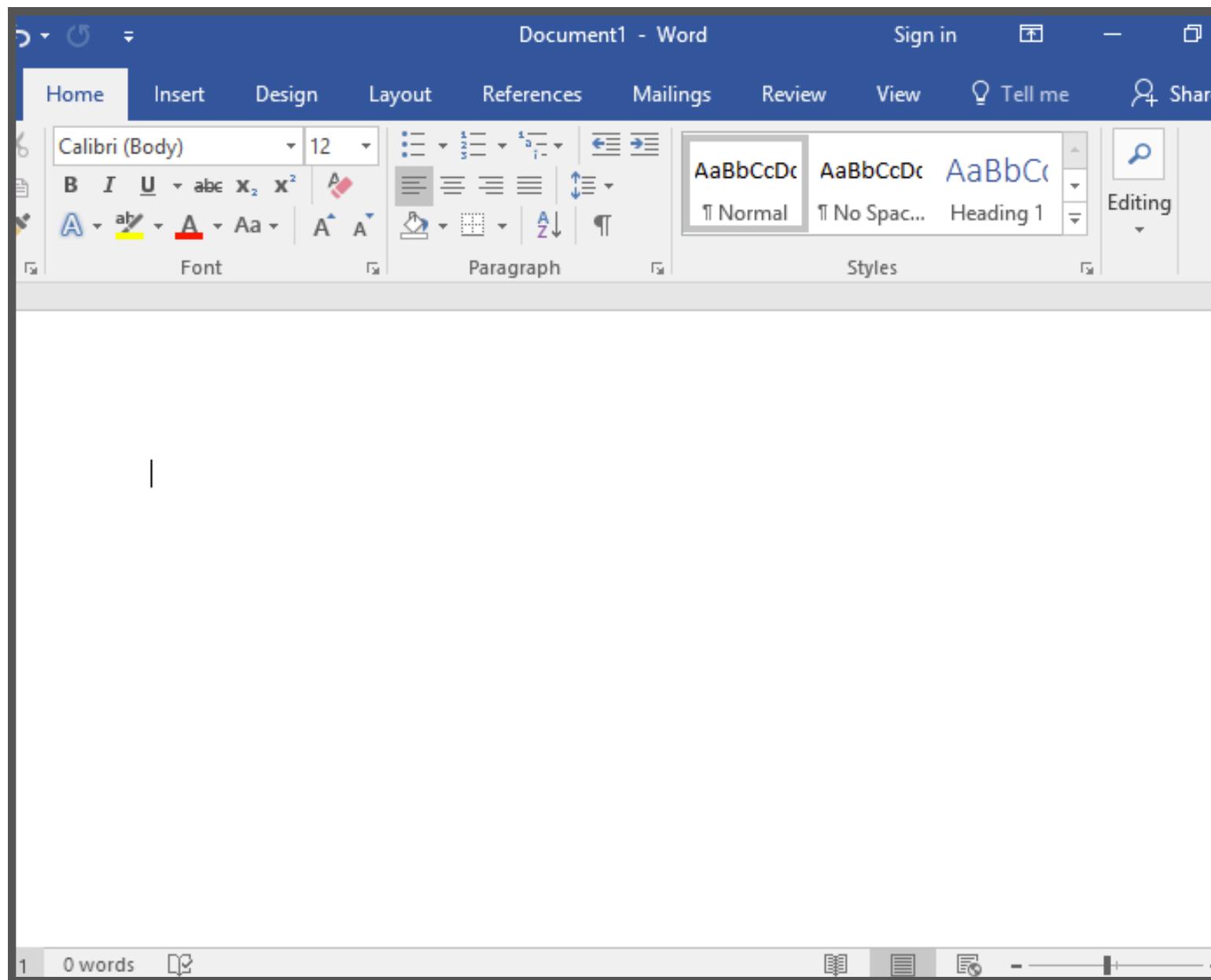
Quadrant 2 (Bottom-Left) = Price: 2.14

Quadrant 3 (Top-Right) = Price: 2.48

Quadrant 4 (Bottom-Right) = Price: 1.58

TOTAL PRICE for the page = 8.97 pesos





TESTBENCH 4

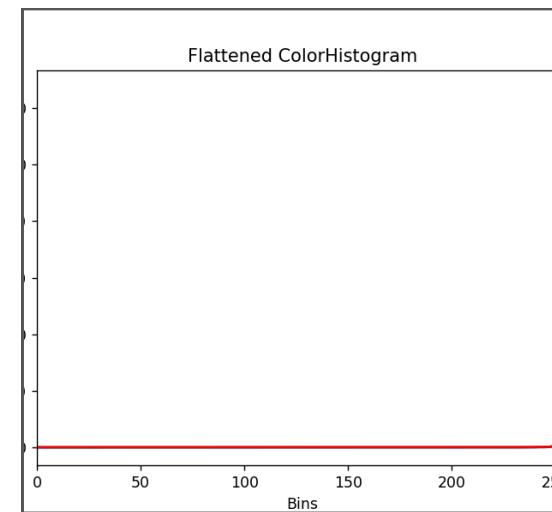
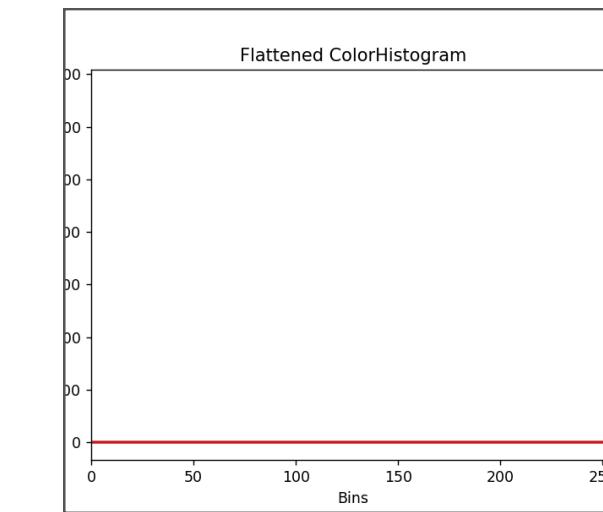
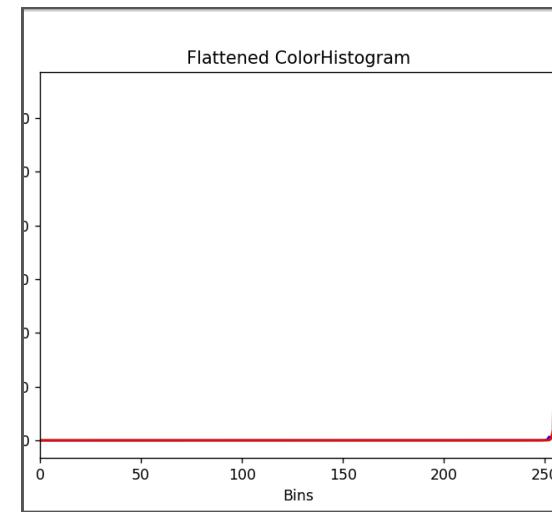
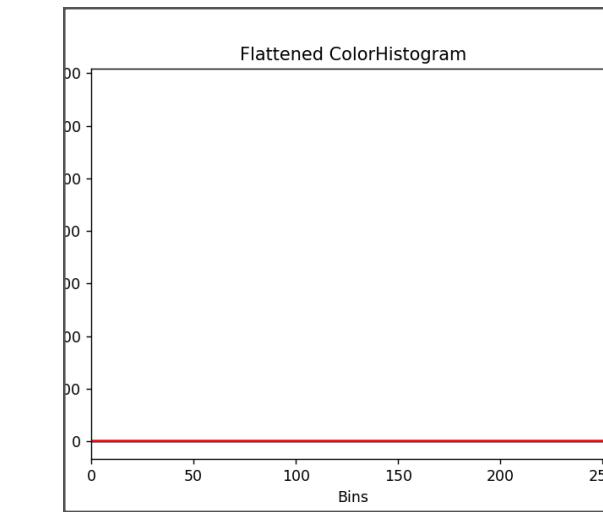
Quadrant 1 (Top-Left) = Price: 2.67

Quadrant 2 (Bottom-Left) = Price: 1.58

Quadrant 3 (Top-Right) = Price: 2.71

Quadrant 4 (Bottom-Right) = Price: 1.71

TOTAL PRICE for the page = 8.67 pesos





TESTBENCH 5

Quadrant 1 (Top-Left) = Price: 1.25

Quadrant 2 (Bottom-Left) = Price: 1.25

Quadrant 3 (Top-Right) = Price: 1.25

Quadrant 4 (Bottom-Right) = Price: 1.26

TOTAL PRICE for the page = 5.01 pesos

Study Plan

Take this ultimate guide for your daily learning goals.

Class & Study Schedule

| TIME | SUN | MON | TUE | WED | THU | FRI |
|------------------|---------|-----------|----------|----------|----------|----------|
| 06:00 - 08:00 AM | Wake Up | | | | | Charging |
| 08:00 - 09:00 AM | | Breakfast | | | | |
| 09:00 - 10:00 AM | | Lunch | | | | |
| 10:00 - 11:00 AM | | Break | | | | |
| 11:00 - 12:00 PM | | | Homework | | | |
| 12:00 - 01:00 PM | | | | Homework | | |
| 01:00 - 02:00 PM | | | | | Homework | |
| 02:00 - 03:00 PM | | | | | | Homework |
| 03:00 - 04:00 PM | | | | | | |
| 04:00 - 05:00 PM | | | | | | |
| 05:00 - 06:00 PM | | | | | | |
| 06:00 - 07:00 PM | | | | | | |
| 07:00 - 08:00 PM | | | | | | |
| 08:00 - 09:00 PM | | | | | | |
| 09:00 - 10:00 PM | | | | | | |
| 10:00 - 11:00 PM | | | | | | |
| 11:00 - 12:00 AM | | | | | | |

Onboarding Plan

Meet your teammates!

| Day | Breakfast | Lunch | Dinner | Snacks |
|-----------|-----------|-------|--------|--------|
| Wednesday | Breakfast | Lunch | Dinner | Snacks |
| Thursday | Breakfast | Lunch | Dinner | Snacks |
| Friday | Breakfast | Lunch | Dinner | Snacks |
| Saturday | Breakfast | Lunch | Dinner | Snacks |
| Sunday | Breakfast | Lunch | Dinner | Snacks |

Meal Planner

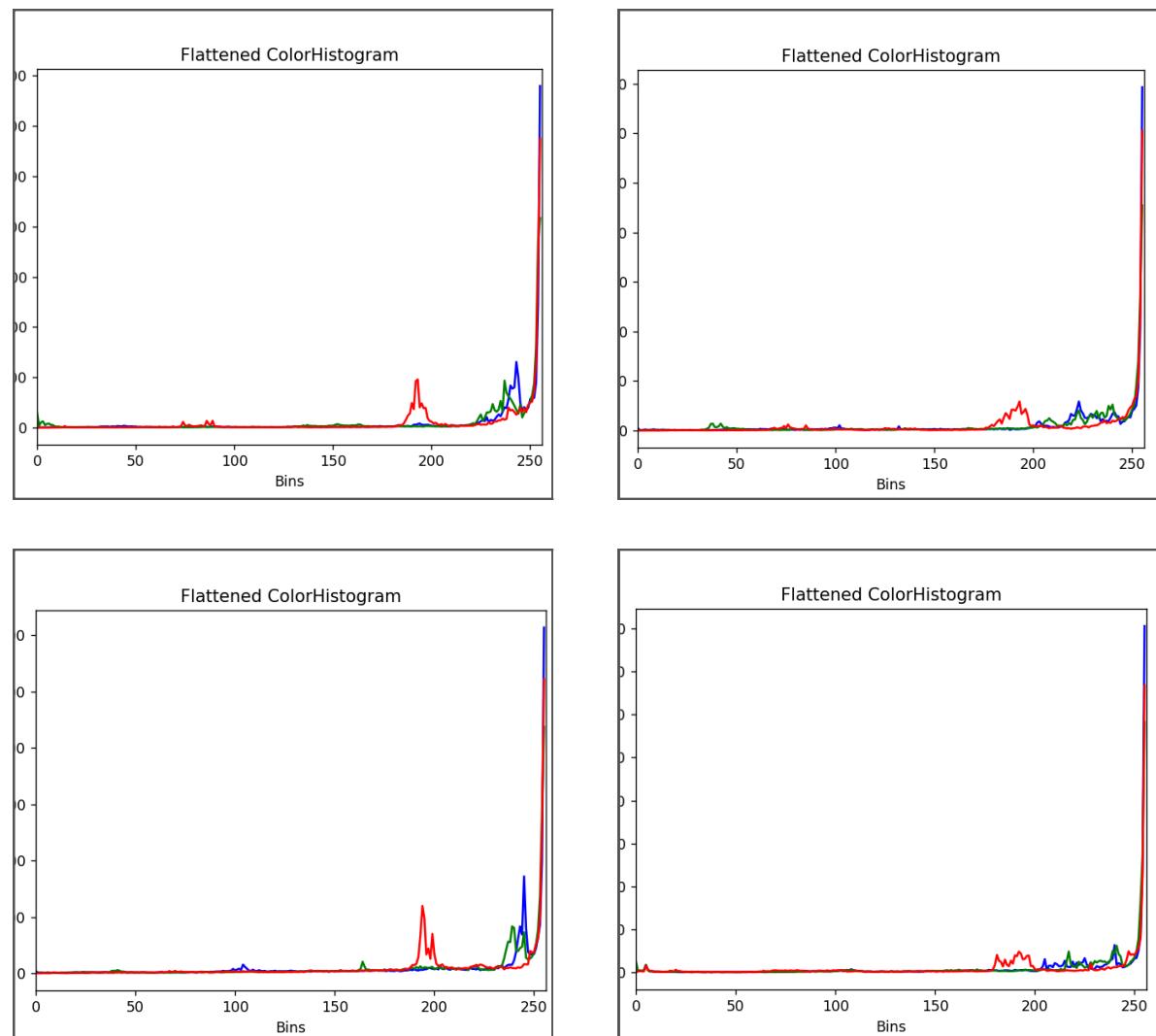
Map the most of your weekly meals.

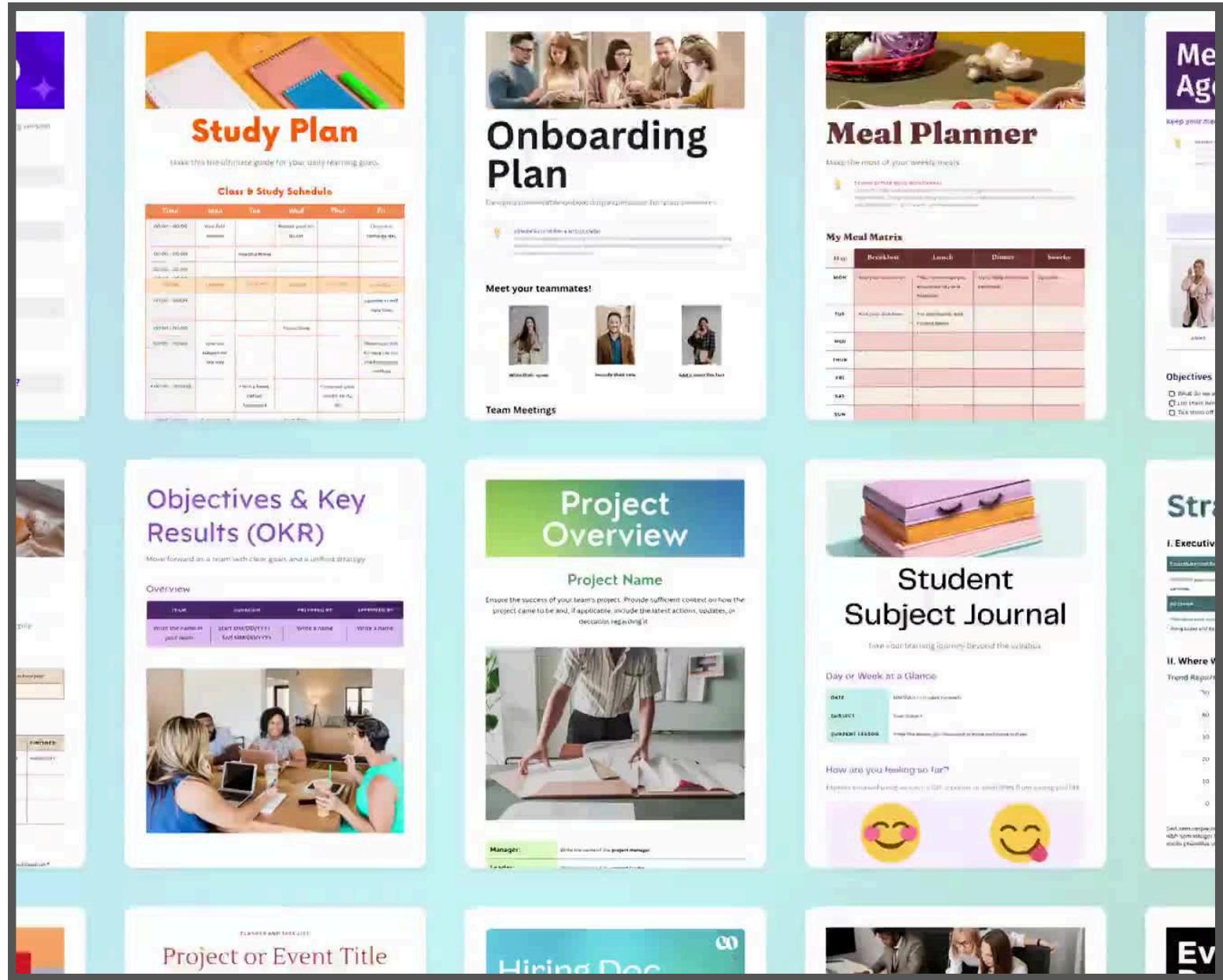
Objectives & Key Results (OKR)

Move forward as a team with clear goals and a unified strategy.

Project Overview

Student Subject Journal





TESTBENCH 6

Quadrant 1 (Top-Left) = Price: 2.71

Quadrant 2 (Bottom-Left) = Price: 2.99

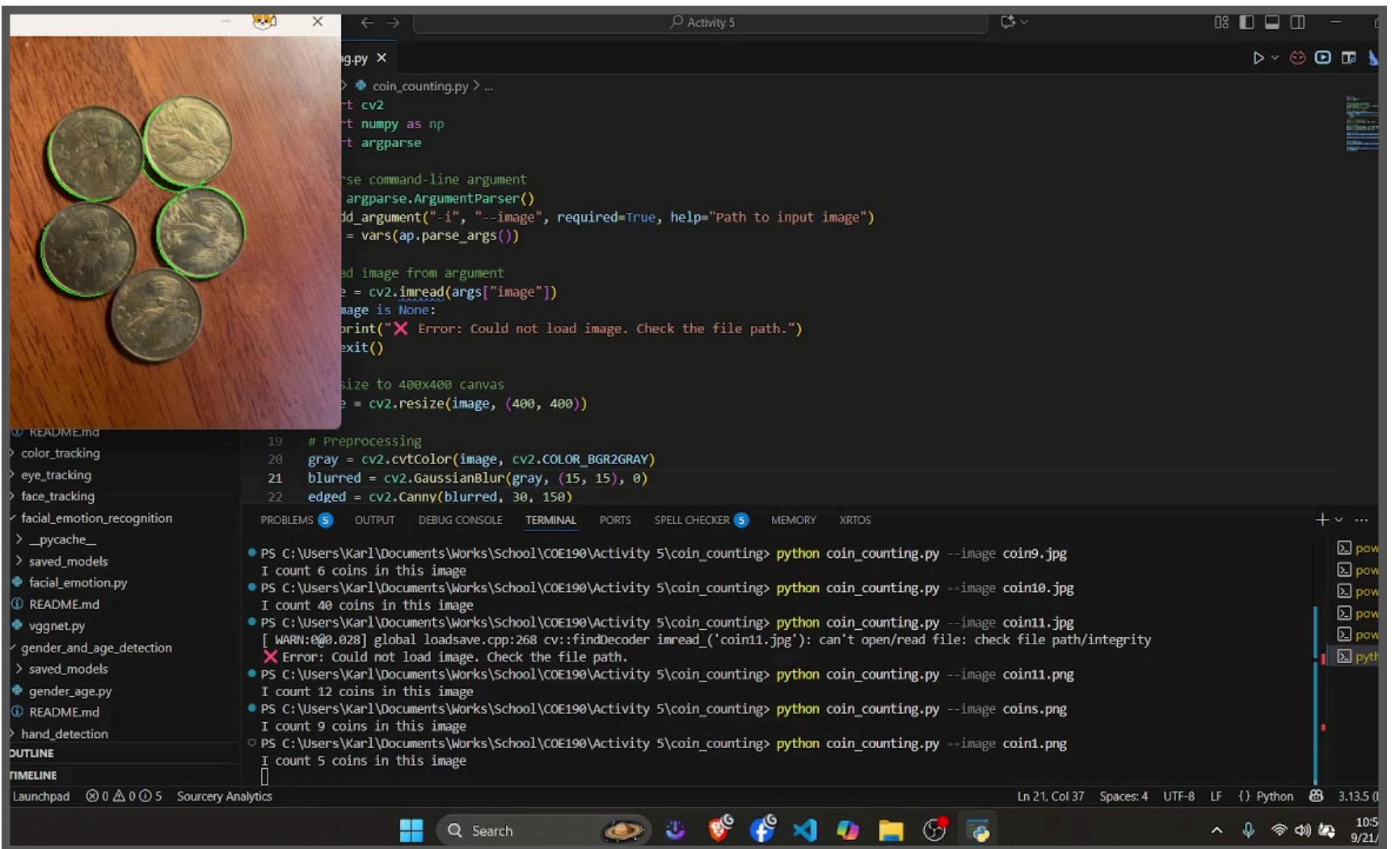
Quadrant 3 (Top-Right) = Price: 3.09

Quadrant 4 (Bottom-Right) = Price: 2.99

TOTAL PRICE for the page = 11.78 pesos

COIN COUNTING

COIN COUNTING TESTBENCH



```
coin_counting.py
> coin_counting.py > ...
> cv2
> numpy as np
> argparse

use command-line argument
argparse.ArgumentParser()
add_argument("-i", "--image", required=True, help="Path to input image")
args = vars(ap.parse_args())

load image from argument
image = cv2.imread(args["image"])
if image is None:
    print("Error: Could not load image. Check the file path.")
    exit()

size to 400x400 canvas
image = cv2.resize(image, (400, 400))

# Preprocessing
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
blurred = cv2.GaussianBlur(gray, (15, 15), 0)
edged = cv2.Canny(blurred, 30, 150)

# Count coins
cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnts = cnts[0] if len(cnts) == 2 else cnts[1]

for c in cnts:
    # Compute the bounding box of the contour
    (x, y, w, h) = cv2.boundingRect(c)
    # Only count coins that are roughly circular
    if w > 0 and h > 0 and abs(w - h) < 10:
        # Draw a green circle around the coin
        cv2.circle(image, (int(x + w / 2), int(y + h / 2)), 2, (0, 255, 0), 2)

# Show the image
cv2.imshow("Image", image)
cv2.waitKey(0)
```

```
image = cv2.resize(image, (400, 400))
```

Preprocessing

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
blurred = cv2.GaussianBlur(gray, (15, 15), 0)
```

```
edged = cv2.Canny(blurred, 30, 150)
```



5 coins



21 coins



10 Coins



10 coins



7 coins



6 Coins



10 coins



12 coins



9 coins



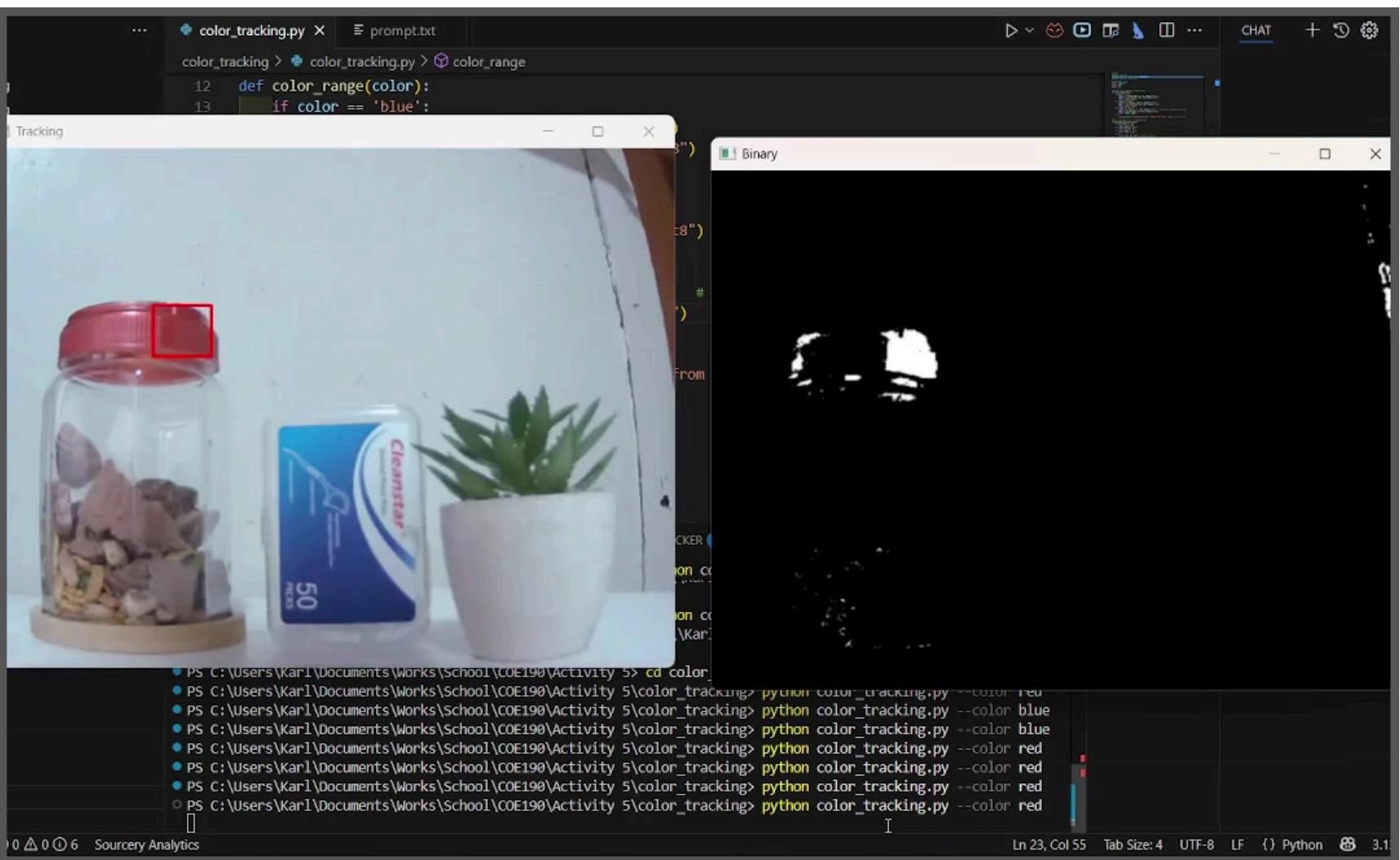
40 Coins

COLOR TRACKING



```
def color_range(color):  
    if color == 'blue':  
        lower = np.array([100, 67, 0], dtype="uint8")  
        upper = np.array([255, 128, 50], dtype="uint8")  
        return (lower, upper)  
    elif color == 'green':  
        lower = np.array([0, 100, 0], dtype="uint8")  
        upper = np.array([100, 255, 100], dtype="uint8")  
        return (lower, upper)  
    elif color == 'red':  
        lower = np.array([0, 0, 100], dtype="uint8") # low blue,  
        low green, high red  
        upper = np.array([80, 80, 255], dtype="uint8")  
        return (lower, upper)  
    else:  
        raise ValueError("Unsupported color. Choose from  
'blue', 'green', or 'red'.")
```

COLOR TRACKING TESTBENCH



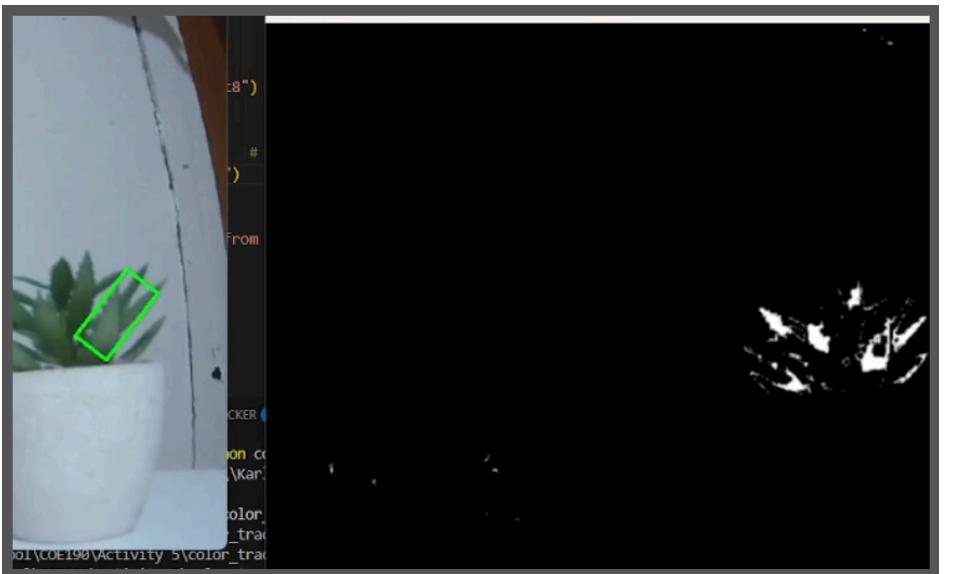
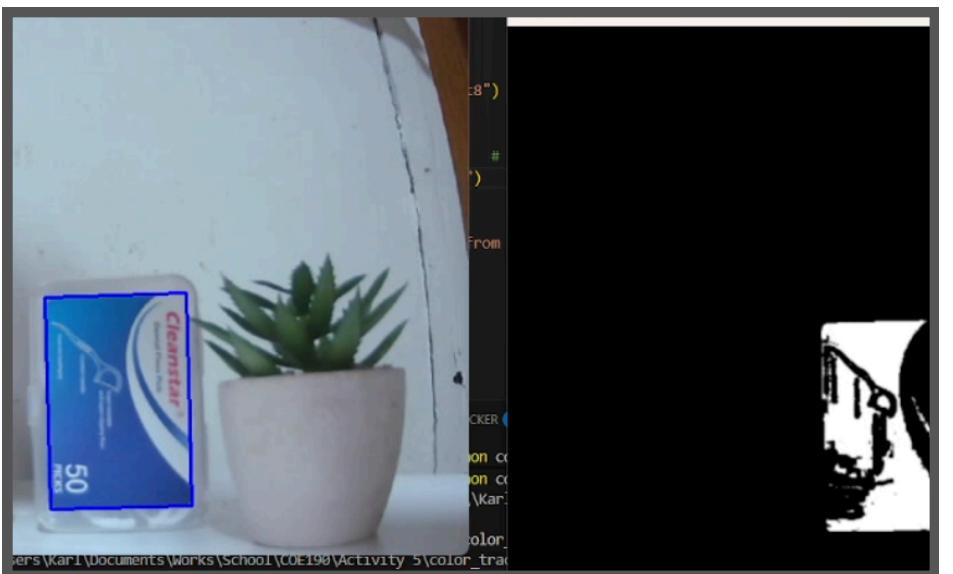
color_tracking.py

```
color_tracking > color_tracking.py > color_range
12 def color_range(color):
13     if color == 'blue':
14         return True
15     else:
16         return False
```

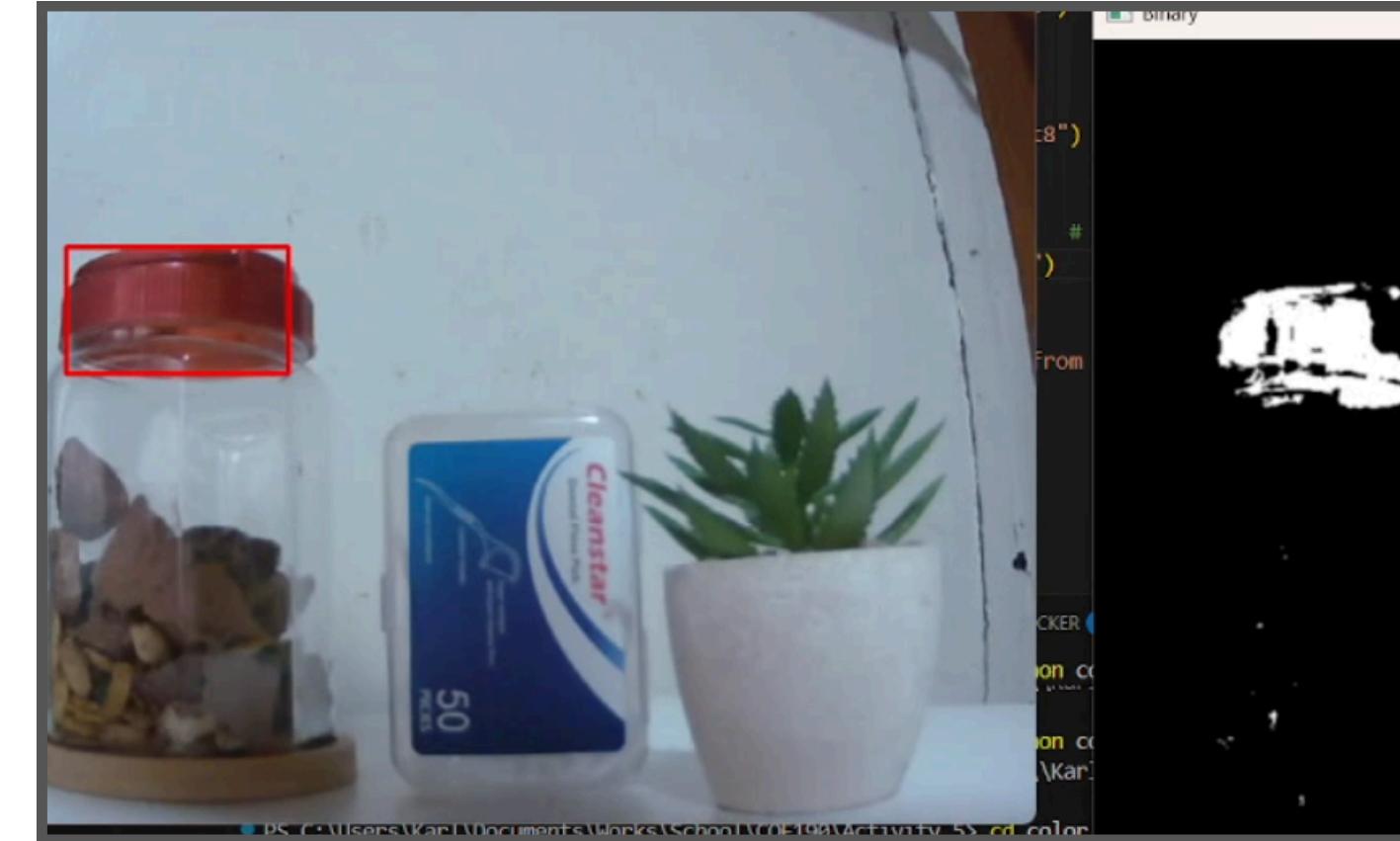
Binary

```
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5> cd color
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5\color_tracking> python color_tracking.py --color red
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5\color_tracking> python color_tracking.py --color blue
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5\color_tracking> python color_tracking.py --color blue
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5\color_tracking> python color_tracking.py --color red
```

Sourcery Analytics



LIGHTING DIFFERENCE (red)

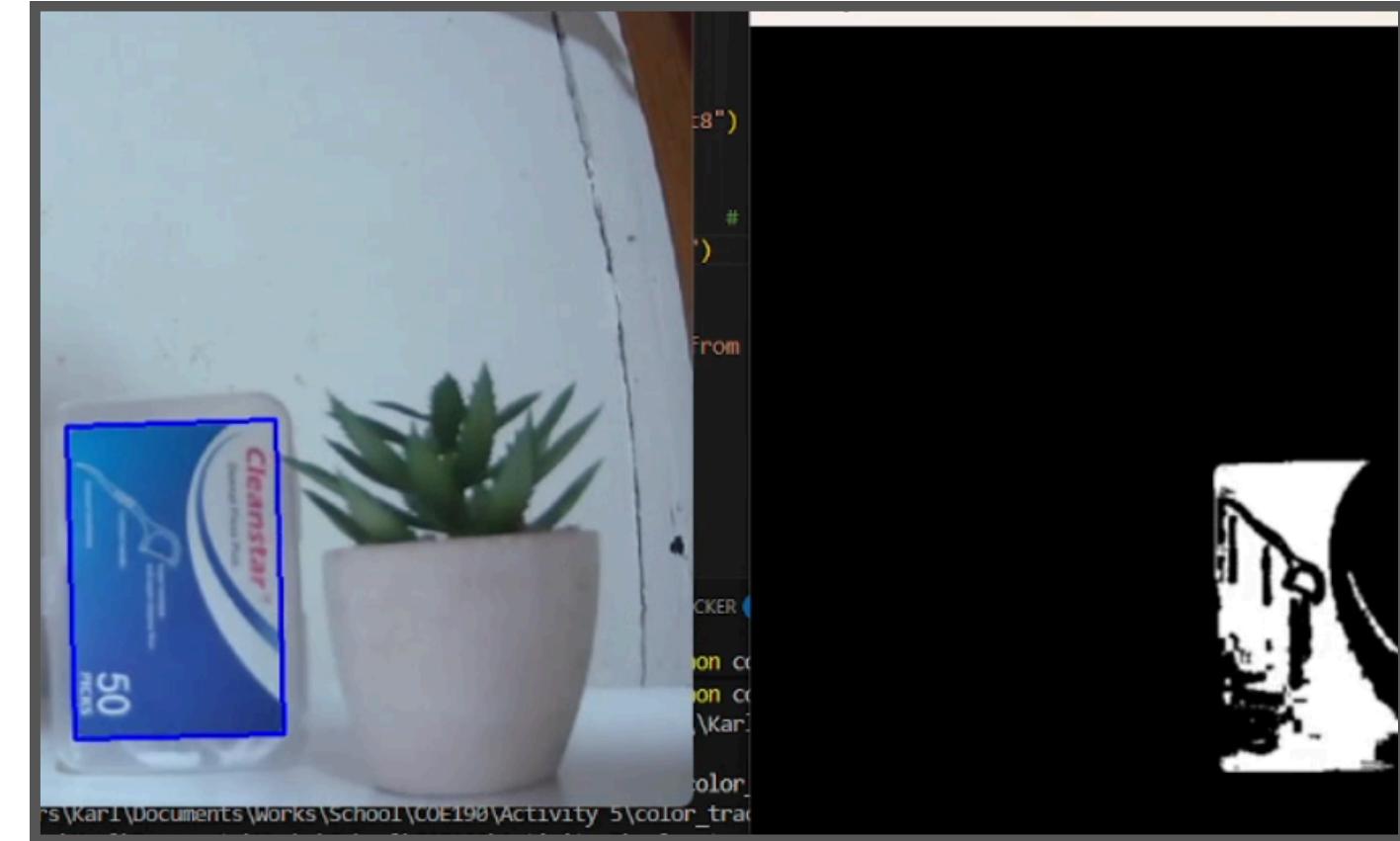
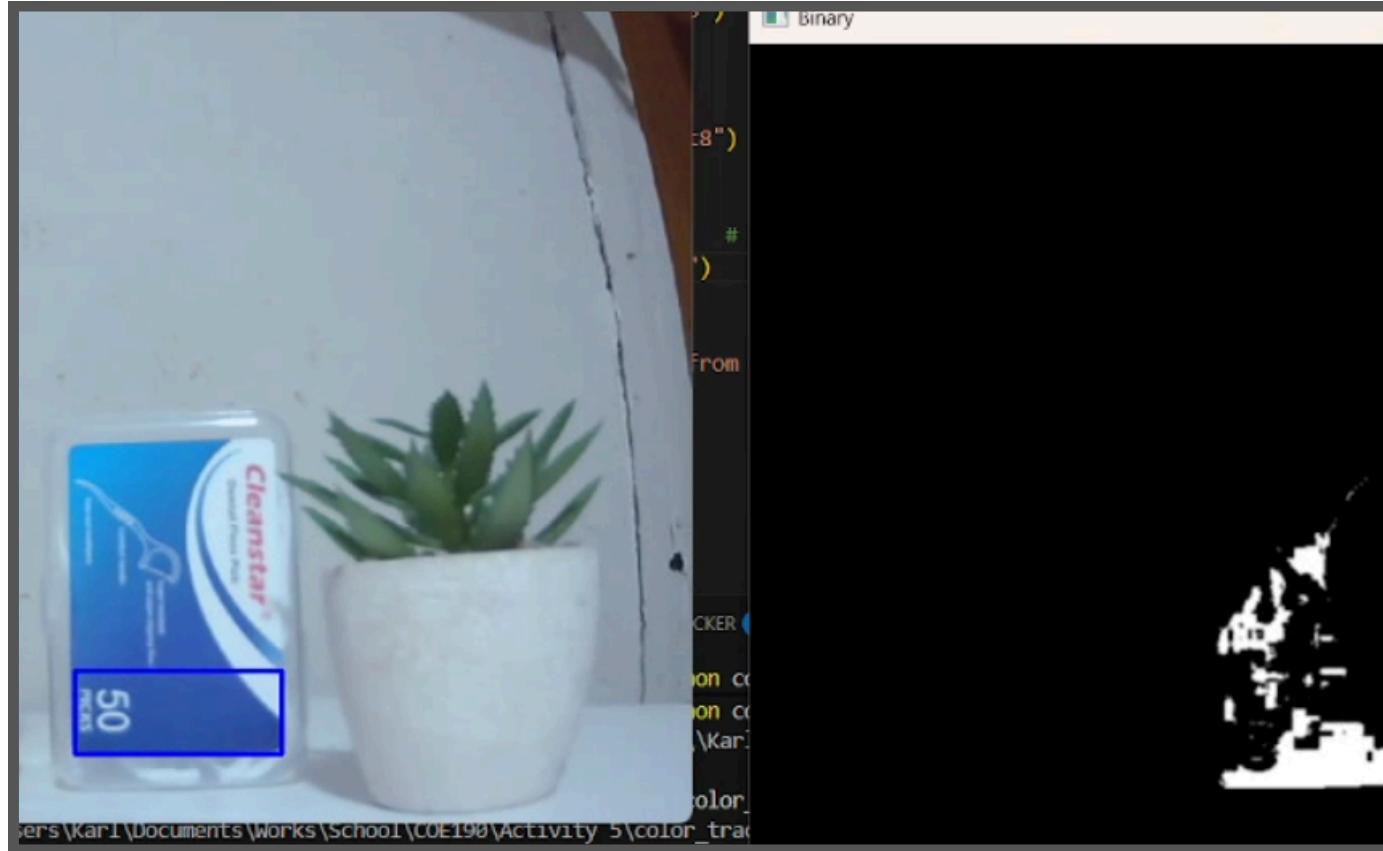


BRIGHT



LOW LIGHT

LIGHTING DIFFERENCE (blue)

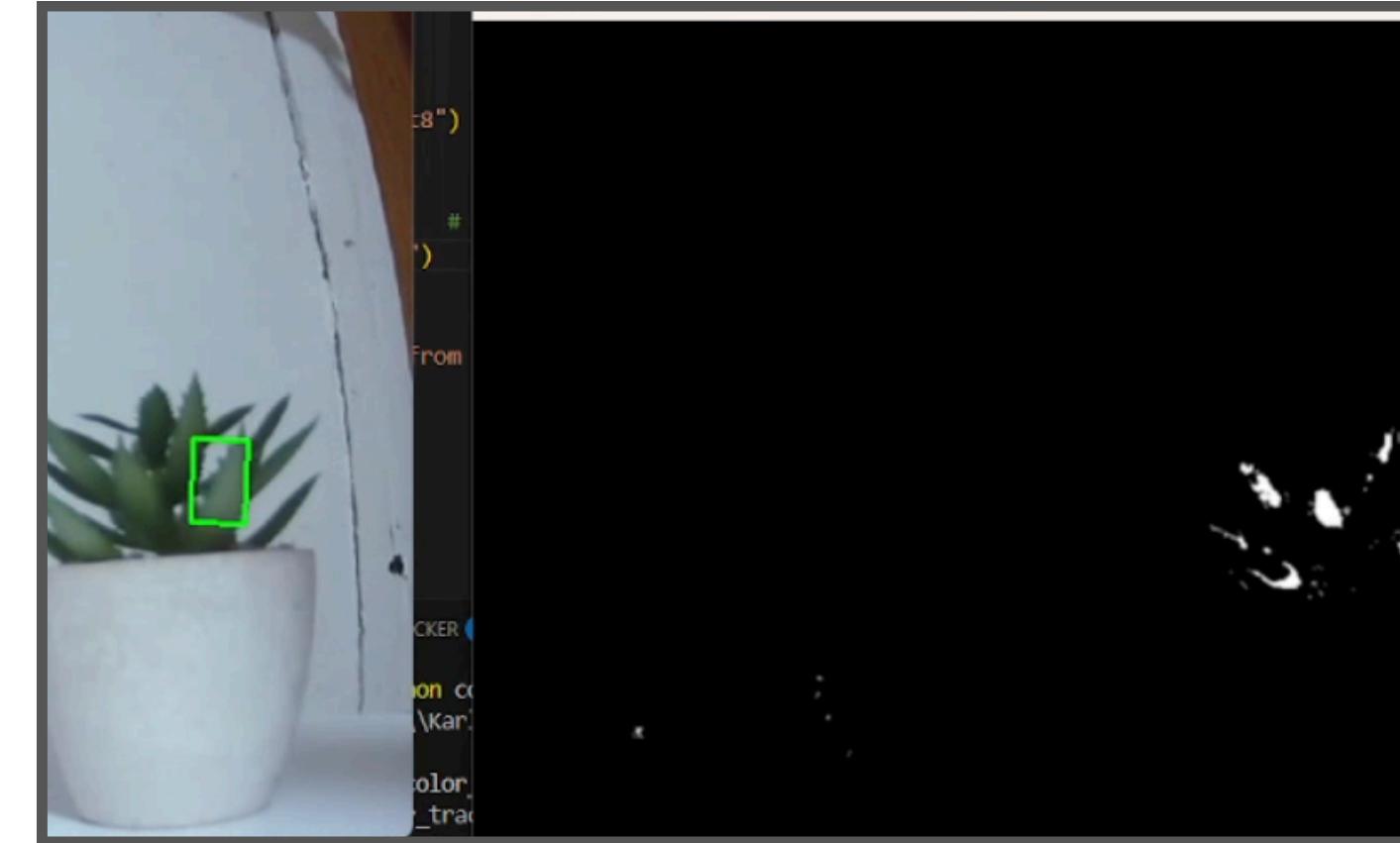
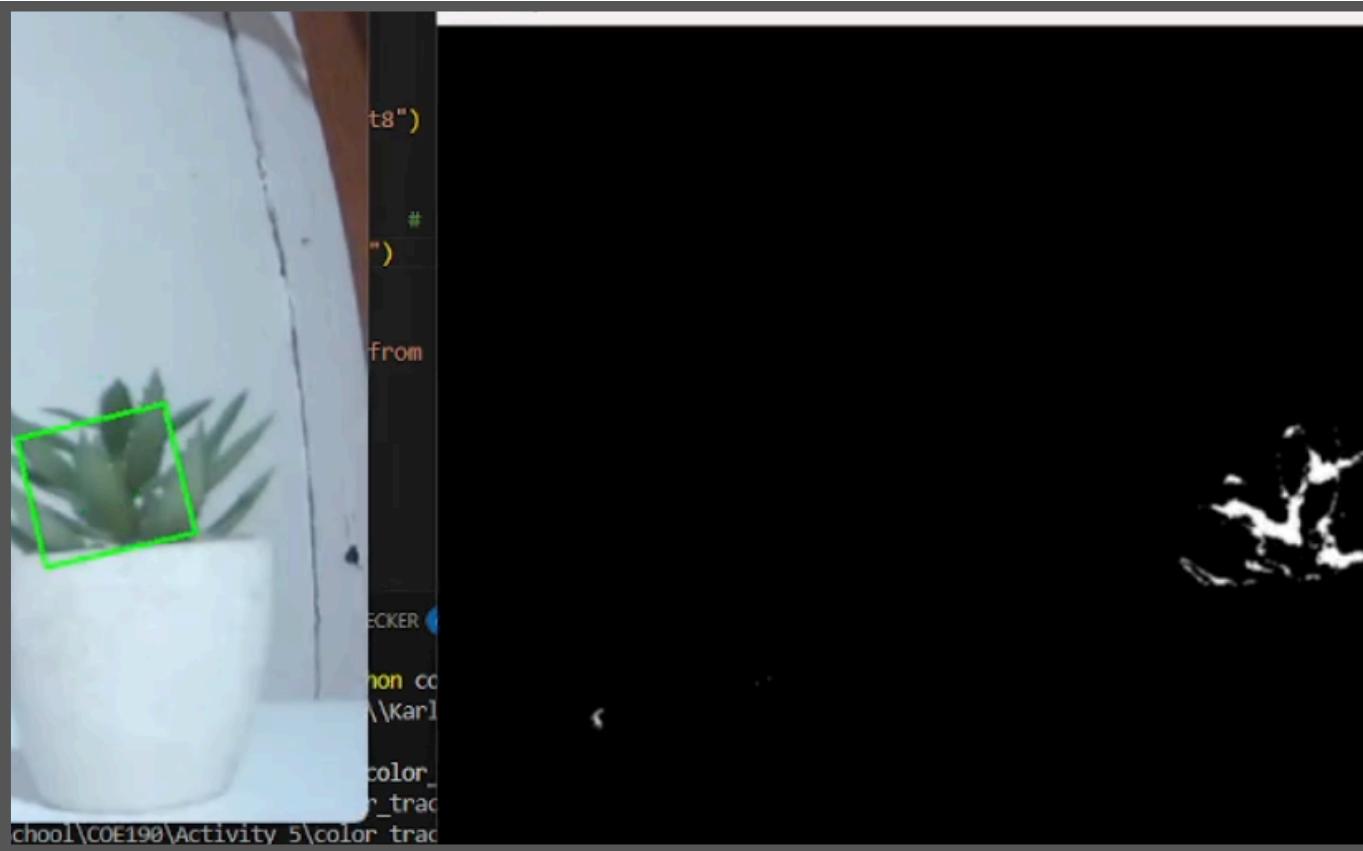


BRIGHT



LOW LIGHT

LIGHTING DIFFERENCE (*green*)



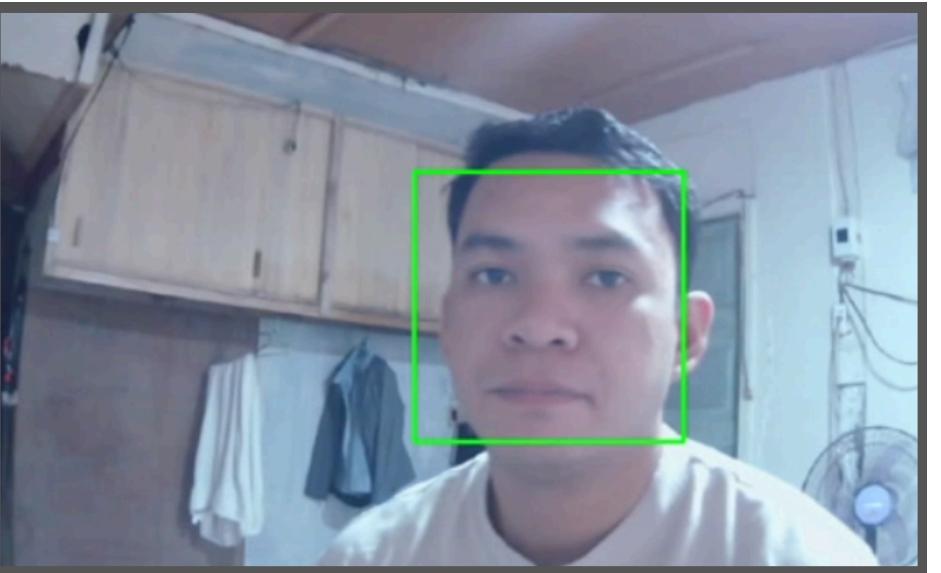
BRIGHT



LOW LIGHT

FACE TRACKING

FACE TRACKING TESTBENCH



A screenshot of a Windows terminal window titled "Activity 5". The terminal shows the command:

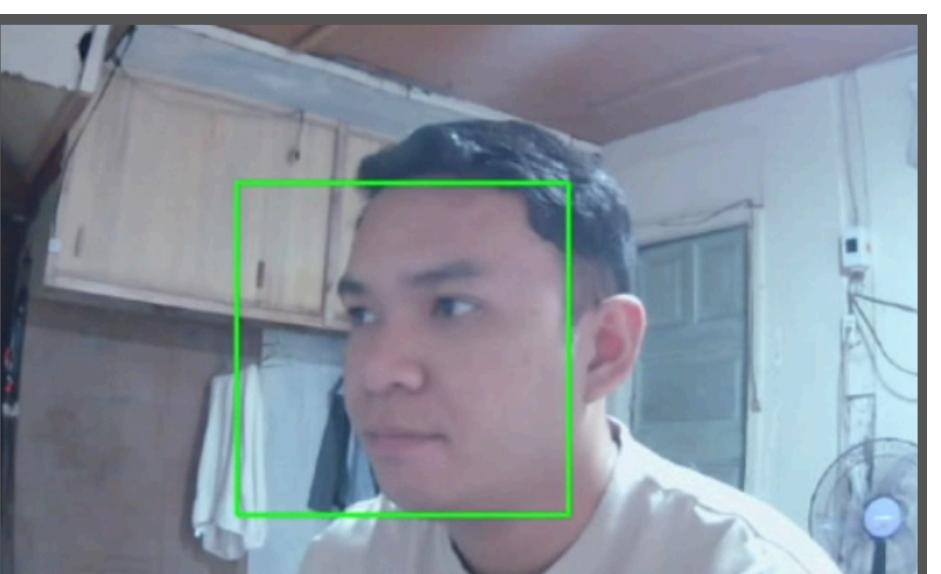
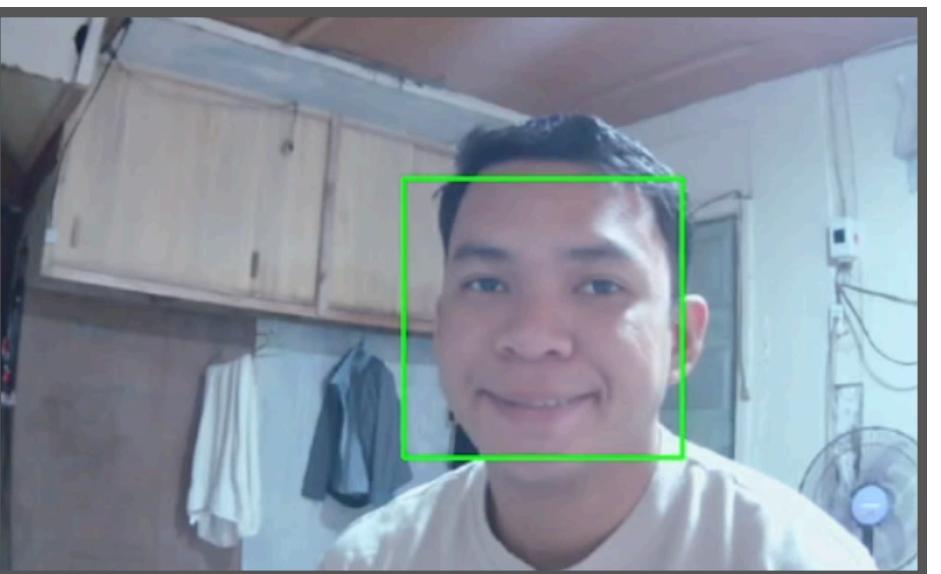
```
python facetracking.py --face cascades/haarcascade_frontalface_default.xml
```

The terminal output shows the command being run and a warning message:

```
[...]
frontalface_default.xml --eye cascades/haarcascade_eye
frontalface_default.xml --eye cascades/haarcascade_eye
frontalface_default.xml
[...]
! facetracking
C:\Users\Karl\Documents\Works\School\COE190\Activity 5\facetracking' because it does not exist.

+ CategoryInfo          : ObjectNotFound: (C:\Users\Karl\Documents\Works\School\COE190\Activity 5\facetracking:String) [Set-Location], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand
```

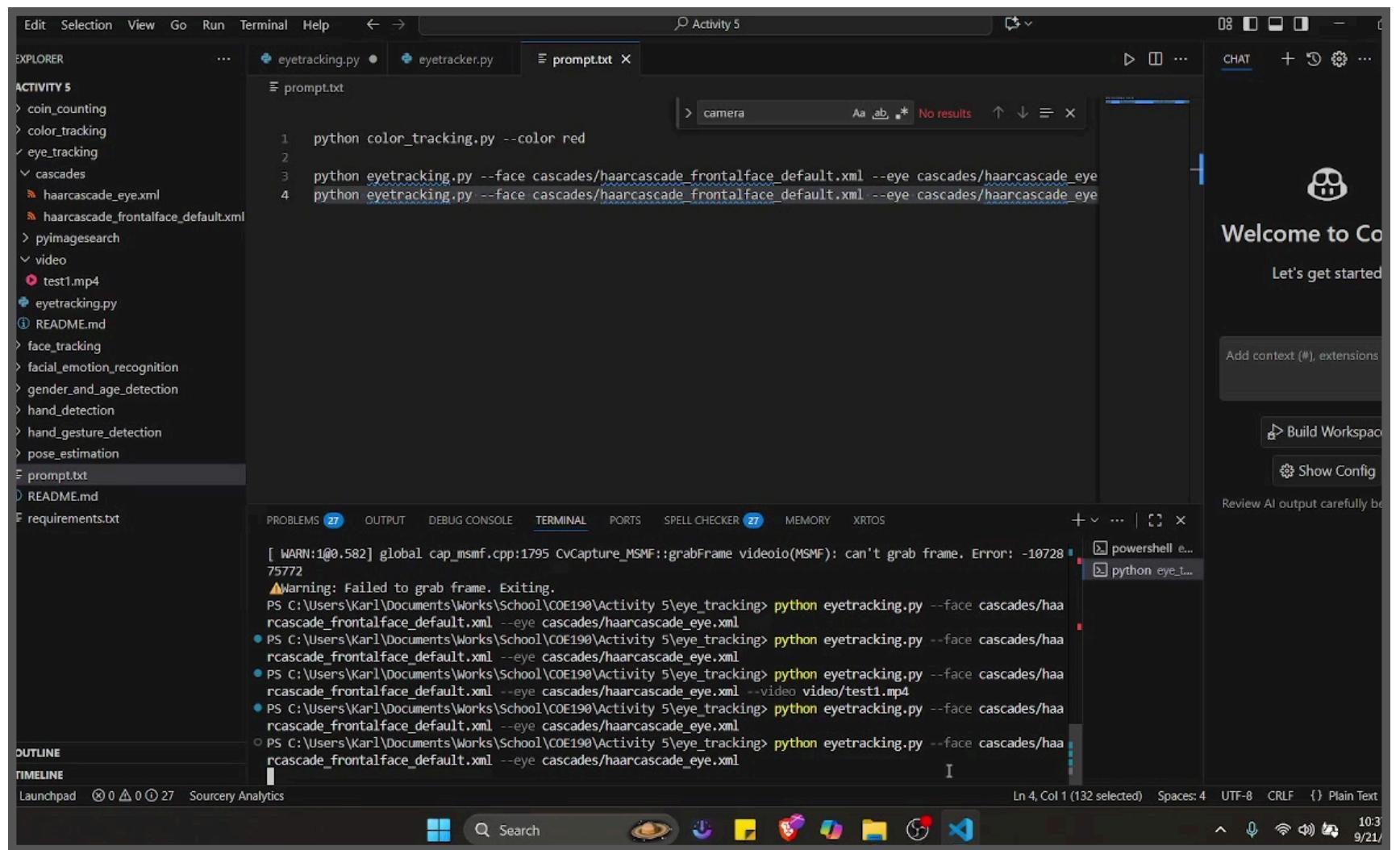
Below the terminal, a video feed from a camera shows the same young man's face, with a green bounding box highlighting the tracked area. The background shows a kitchen with wooden cabinets and a fan.



EYE TRACKING

EYE TRACKING

TESTBENCH

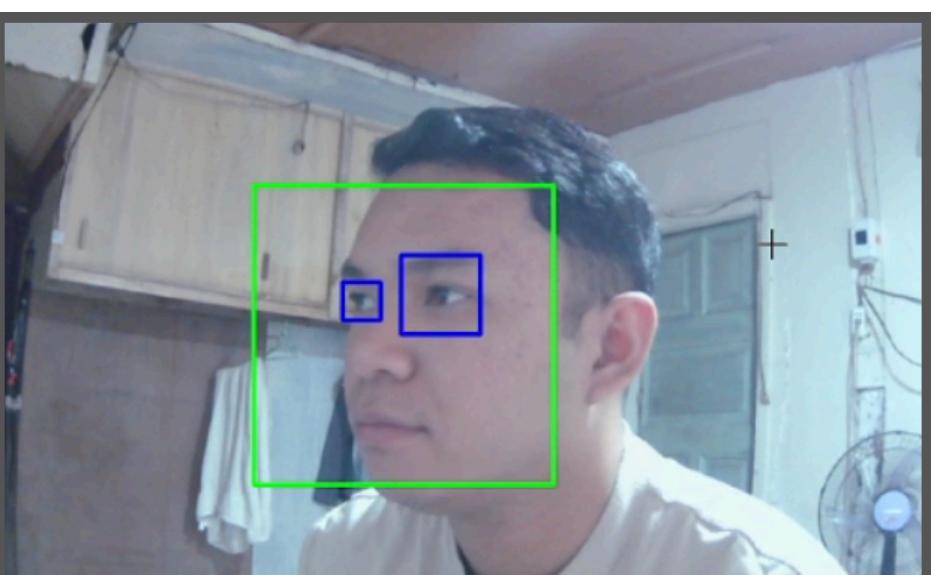
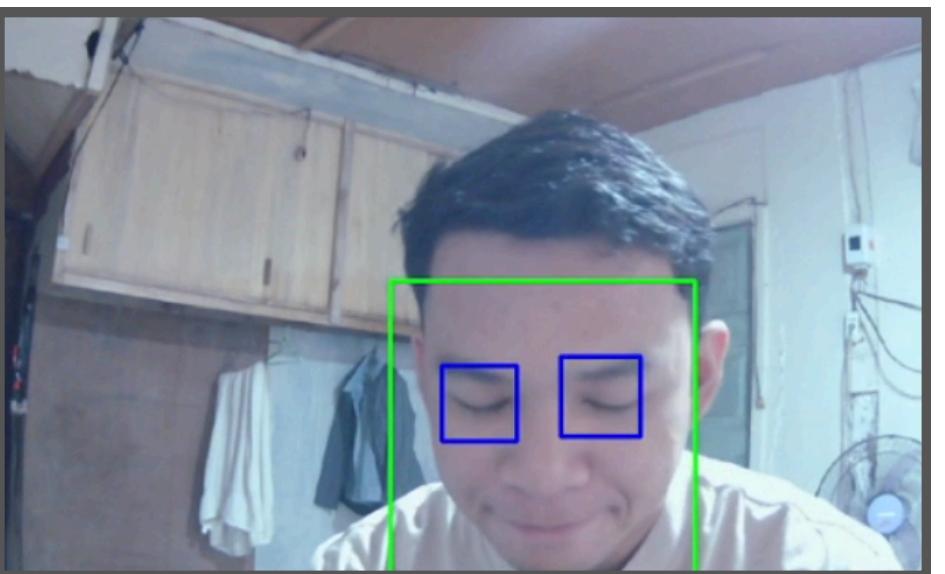
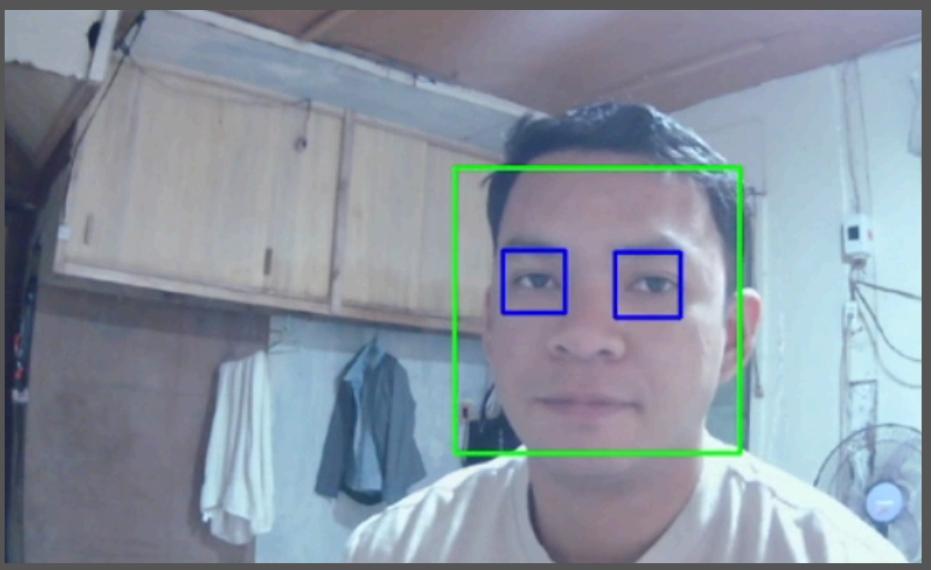


The screenshot shows a code editor interface with several files open:

- EXPLORER: Shows a folder structure for "ACTIVITY 5" containing "coin_counting", "color_tracking", "eye_tracking", "cascades" (with "haarcascade_eye.xml" and "haarcascade_frontalface_default.xml"), "pyimagesearch", "video" (with "test1.mp4"), "eyetracking.py", "README.md", "face_tracking", "facial_emotion_recognition", "gender_and_age_detection", "hand_detection", "hand_gesture_detection", "pose_estimation", "prompt.txt", and "requirements.txt".
- TERMINAL: Displays command-line logs for running eye tracking scripts. The logs show multiple attempts to run the script with different command-line arguments, including errors related to frame grabbing.

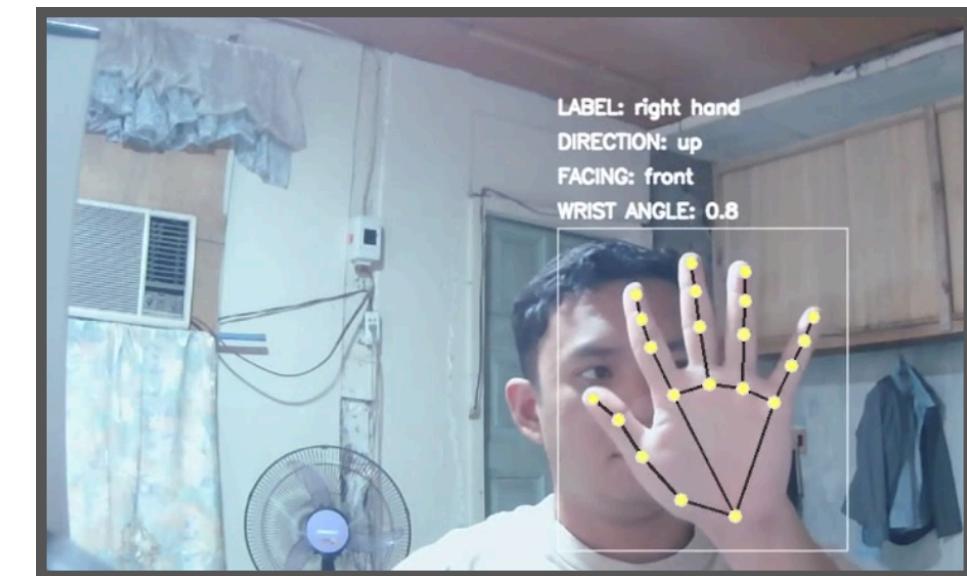
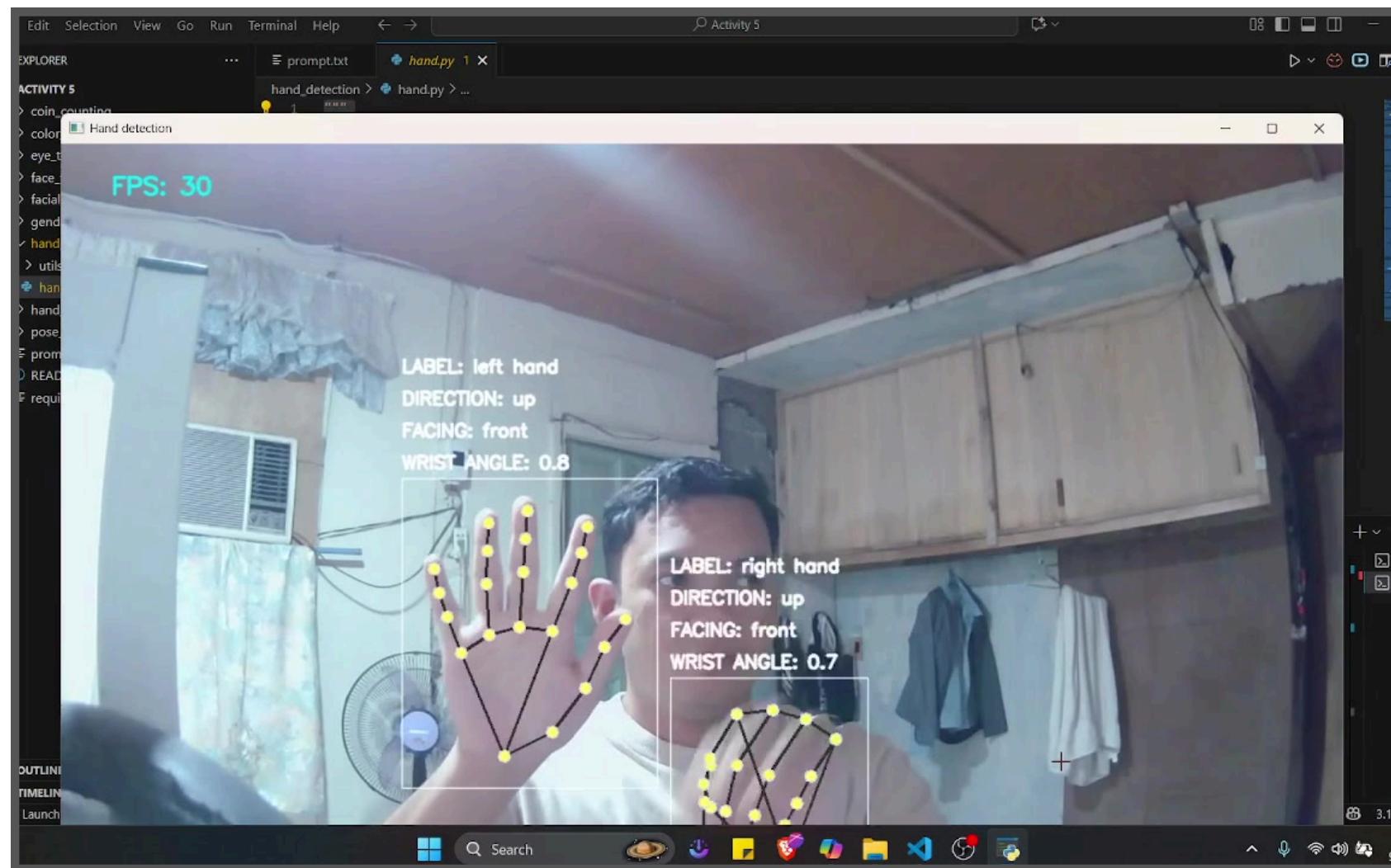
```
python color_tracking.py --color red
python eyetracking.py --face cascades/haarcascade_frontalface_default.xml --eye cascades/haarcascade_eye.xml
python eyetracking.py --face cascades/haarcascade_frontalface_default.xml --eye cascades/haarcascade_eye.xml
```

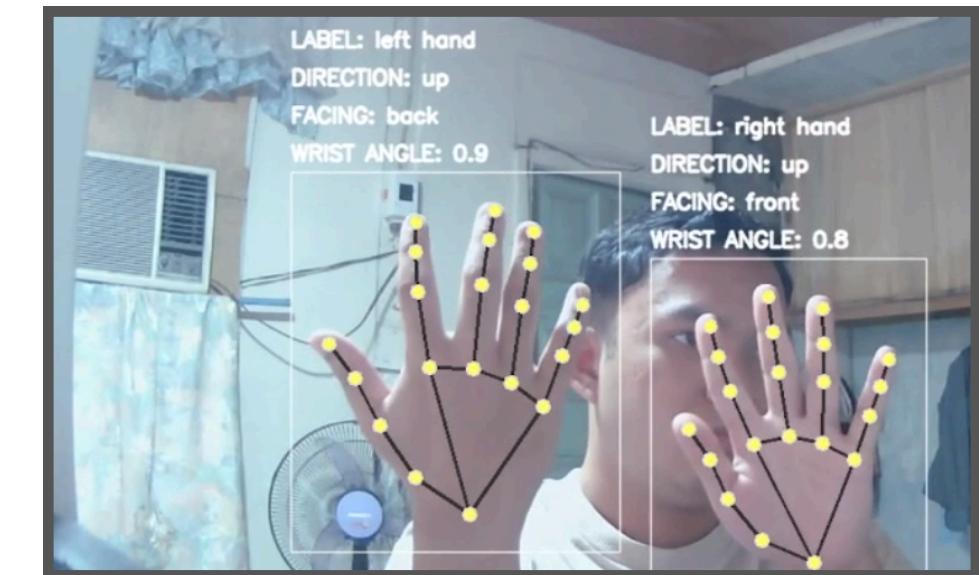
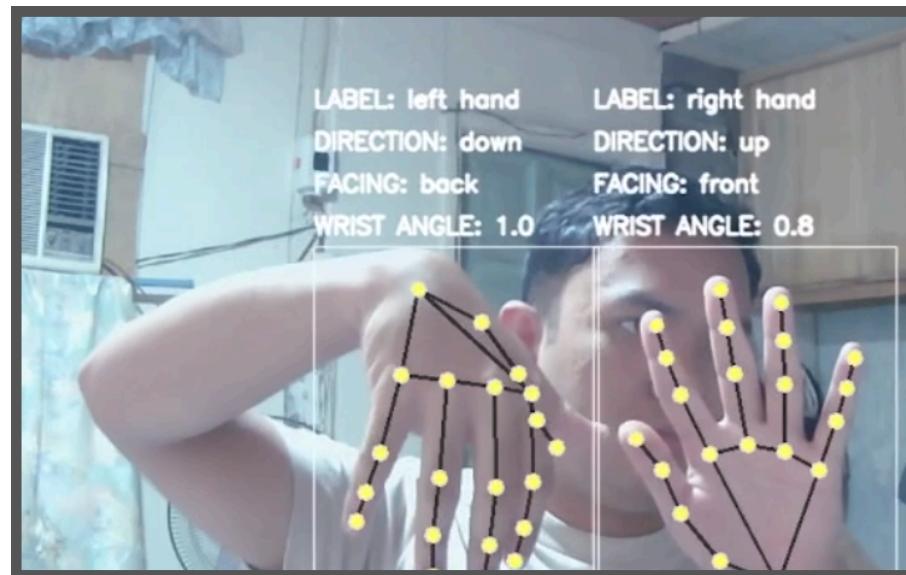
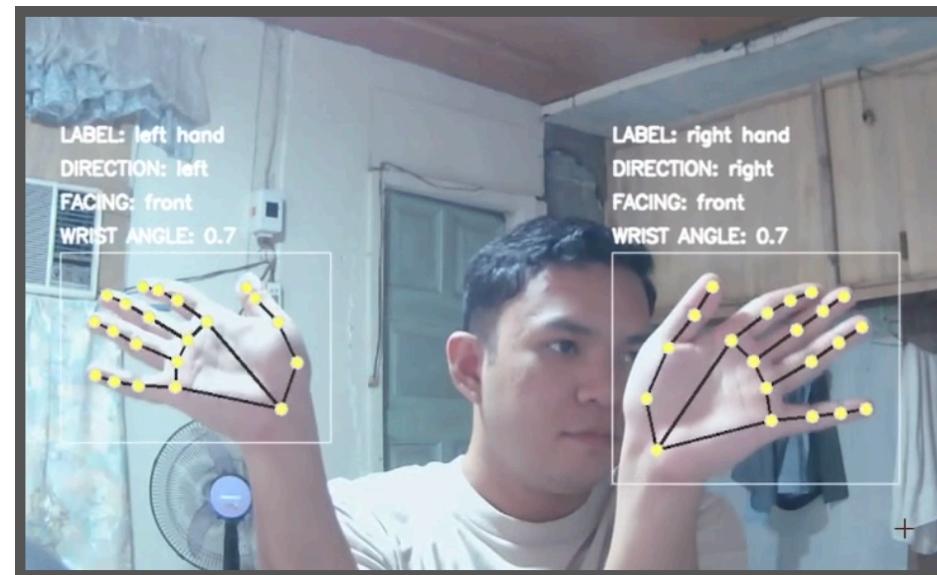
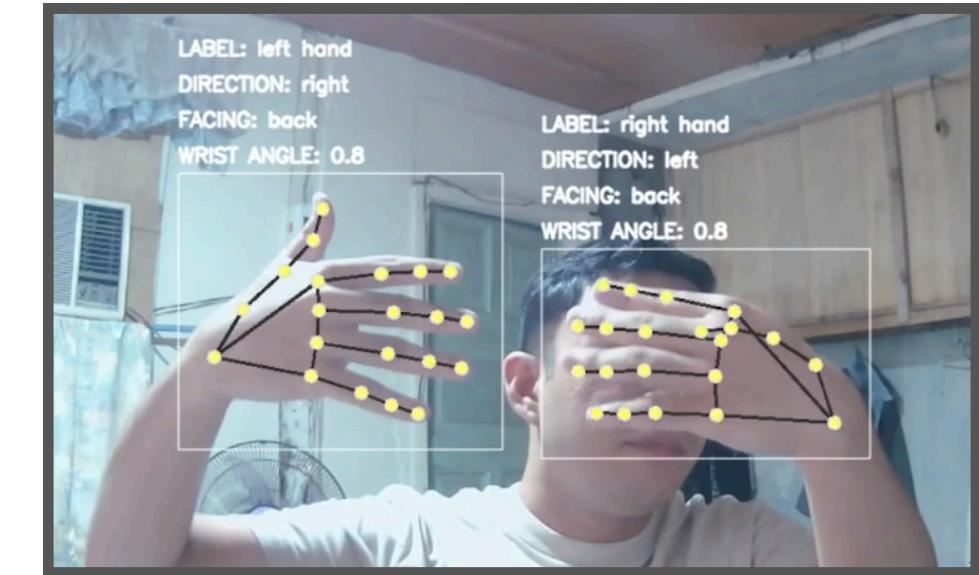
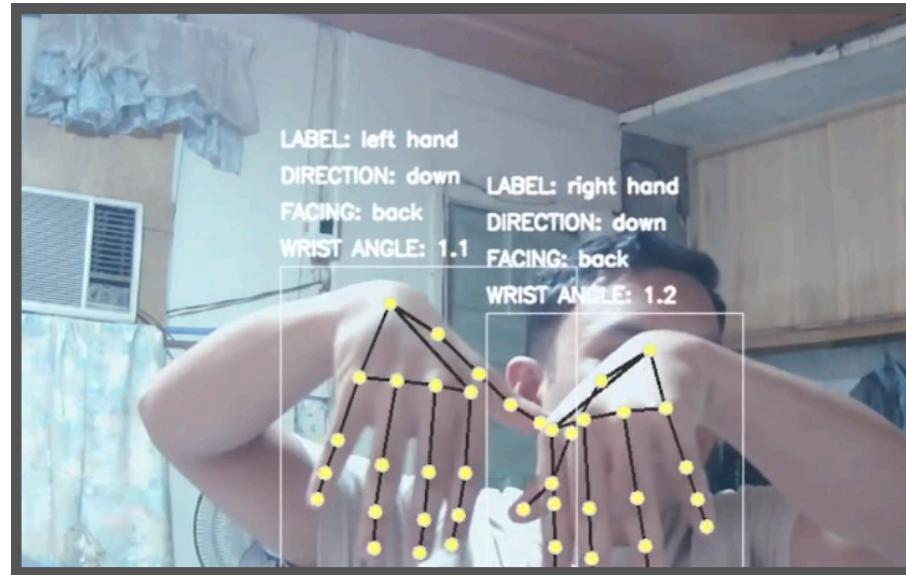
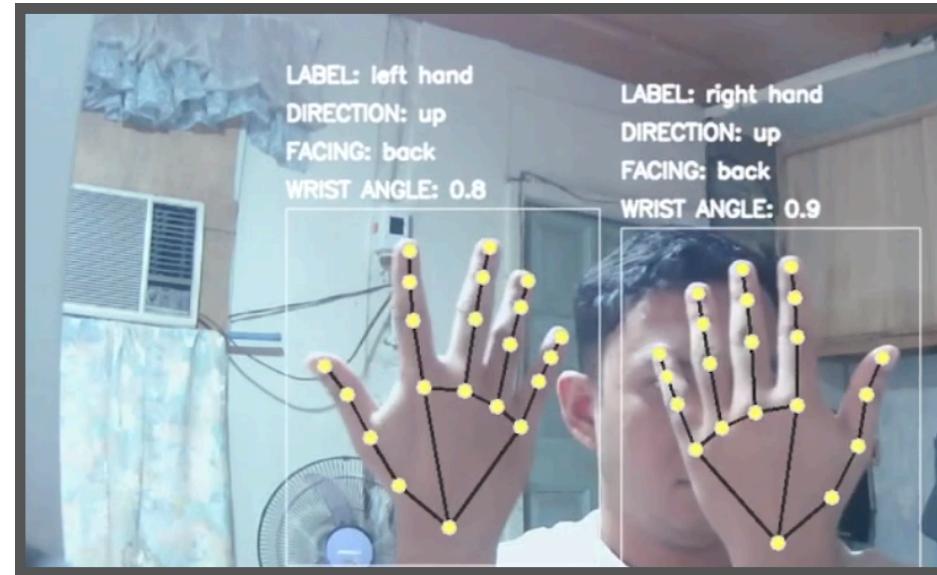
```
[ WARN:100.582] global cap_msdf.cpp:1795 CvCapture_MSMF::grabFrame videoio(MSMF): can't grab frame. Error: -10728
75772
⚠️Warning: Failed to grab frame. Exiting.
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5\eye_tracking> python eyetracking.py --face cascades/haarcascade_frontalface_default.xml --eye cascades/haarcascade_eye.xml
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5\eye_tracking> python eyetracking.py --face cascades/haarcascade_frontalface_default.xml --eye cascades/haarcascade_eye.xml
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5\eye_tracking> python eyetracking.py --face cascades/haarcascade_frontalface_default.xml --eye cascades/haarcascade_eye.xml --video video/test1.mp4
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5\eye_tracking> python eyetracking.py --face cascades/haarcascade_frontalface_default.xml --eye cascades/haarcascade_eye.xml
PS C:\Users\Karl\Documents\Works\School\COE190\Activity 5\eye_tracking> python eyetracking.py --face cascades/haarcascade_frontalface_default.xml --eye cascades/haarcascade_eye.xml
```



HAND DETECTION

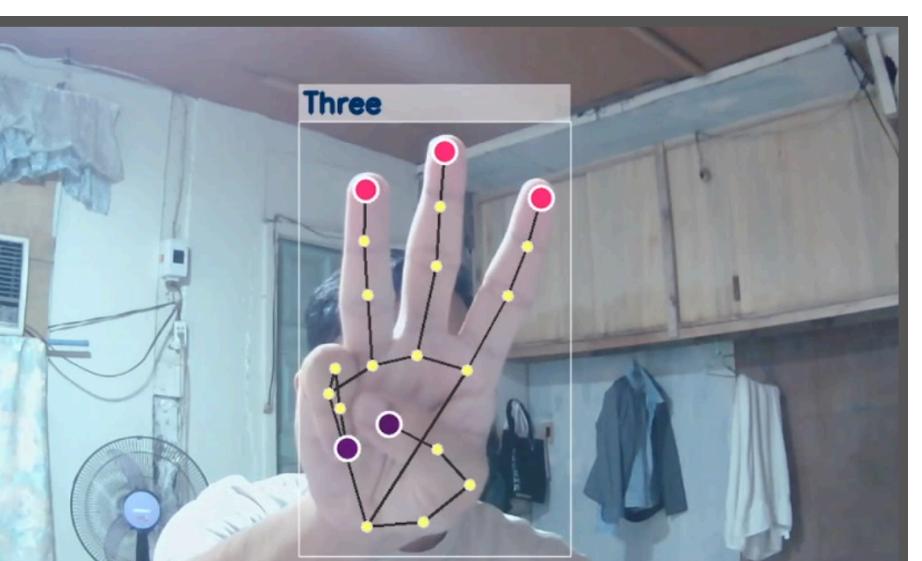
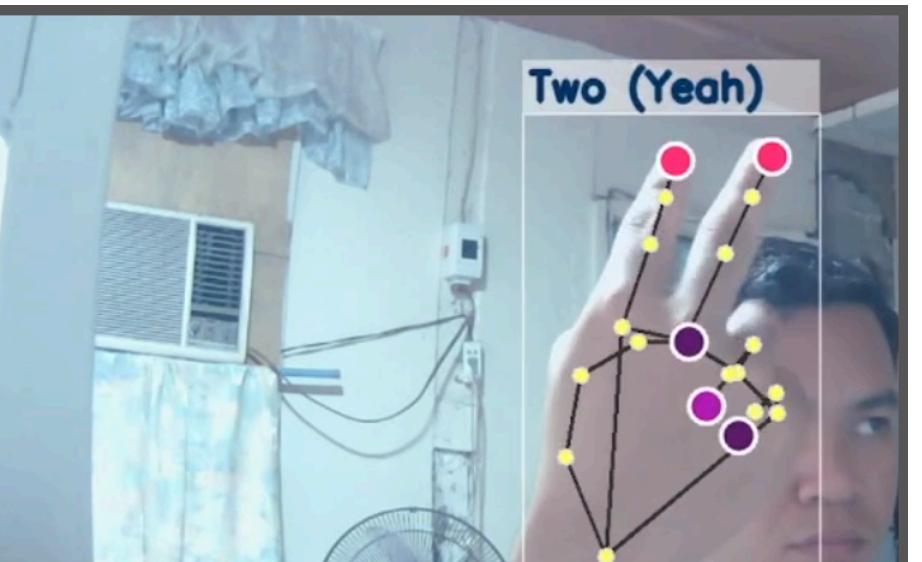
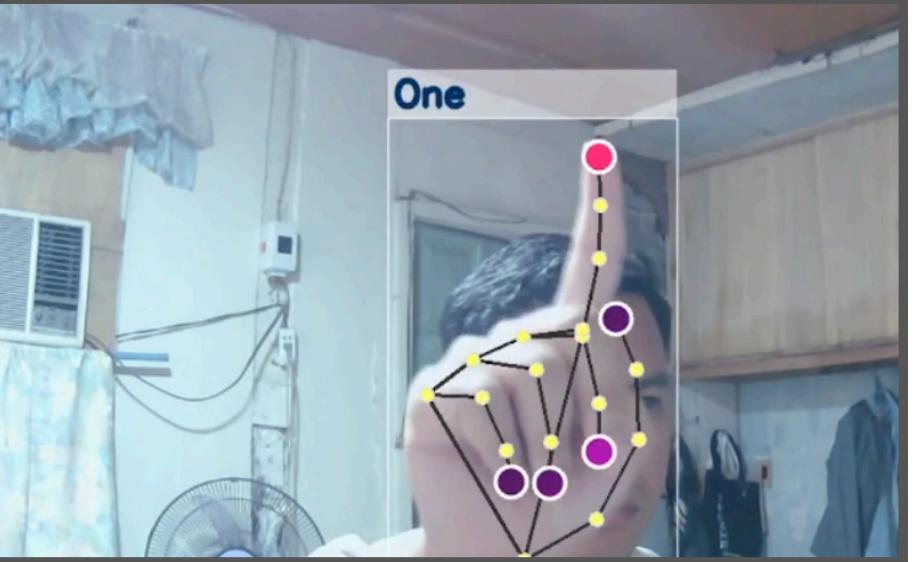
HAND DETECTION TESTBENCH





HAND GESTURE

HAND GESTURE TESTBENCH



A screenshot of a hand gesture recognition application interface. The main window shows a video feed of a person's hand with a skeleton overlay. A callout box labeled "One" highlights the index finger. The application has a dark theme with a top navigation bar and a sidebar containing project files like "gesture.py". The terminal tab shows Python command-line output related to TensorFlow and oneDNN custom operations.

```
def main(mode='single', target_gesture='all'):
```

```
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--mode', type=str, default='single',
                        help='single/double-hand gestures (default: single)')
    parser.add_argument('--target_gesture', type=str, default='all',
                        help='detect a specific gesture (default: all)')
    opt = parser.parse_args()
```

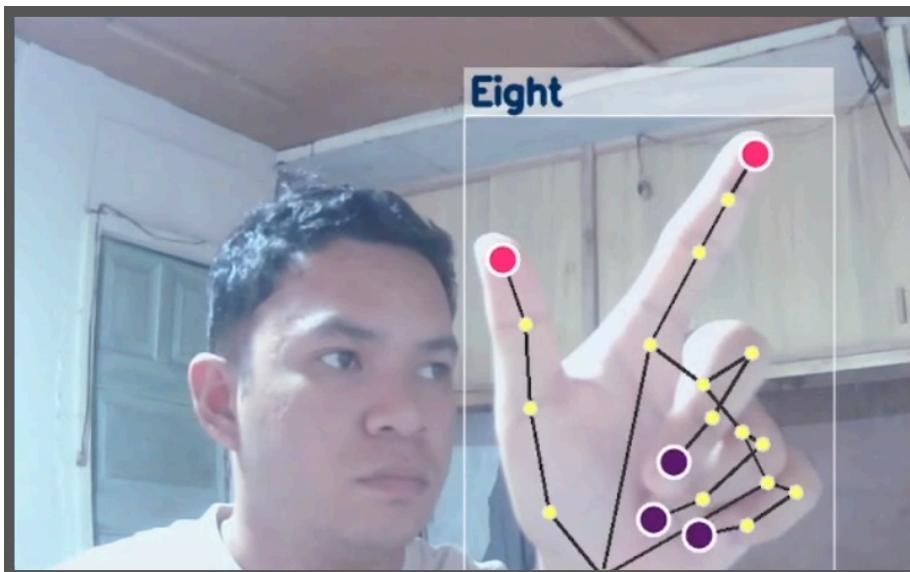
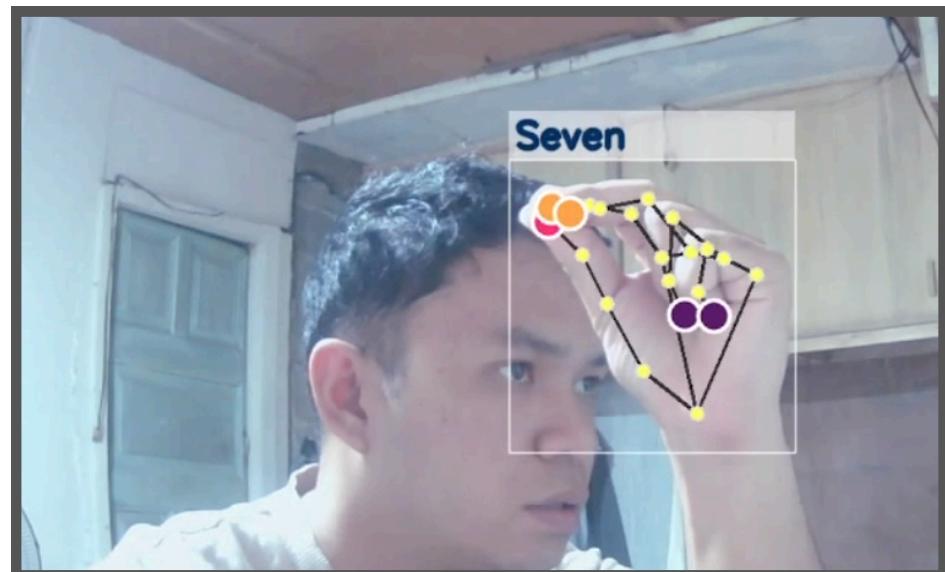
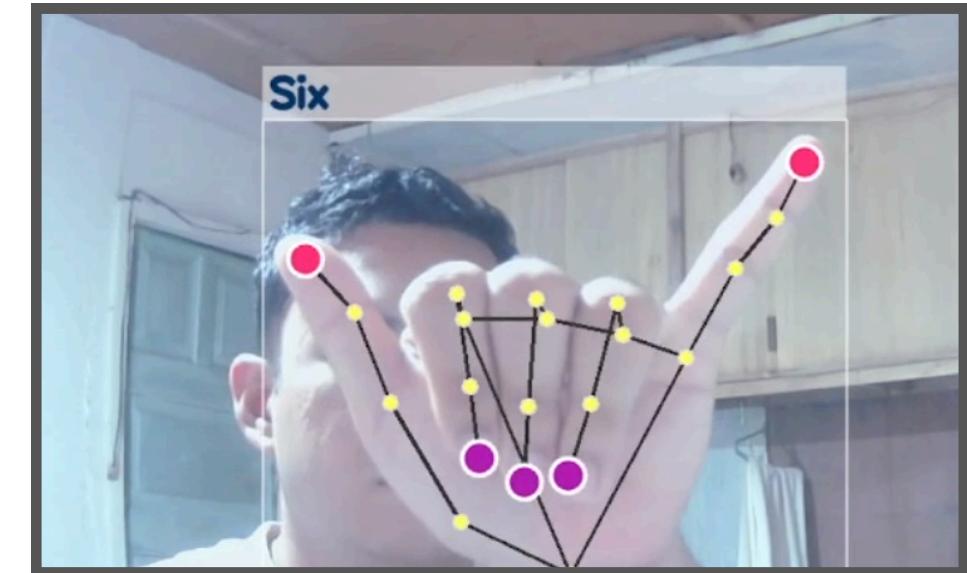
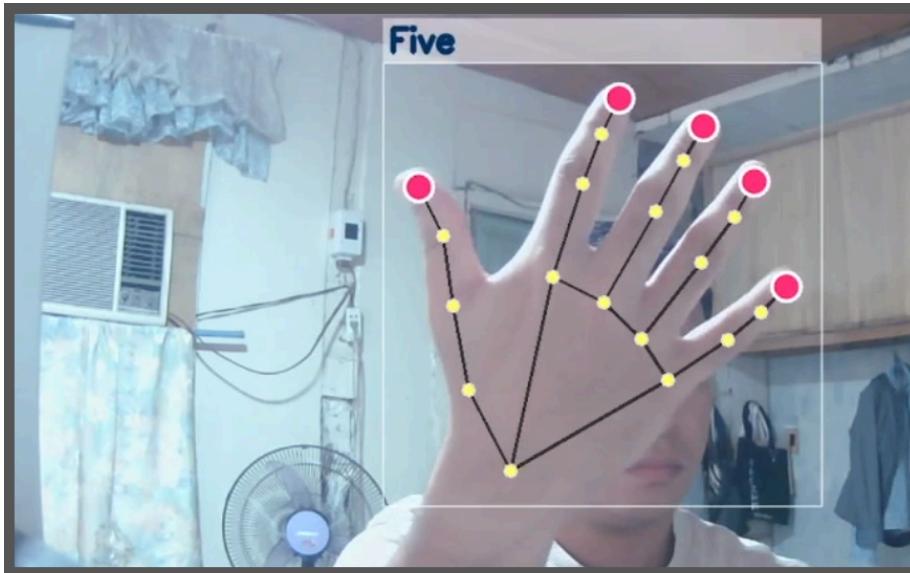
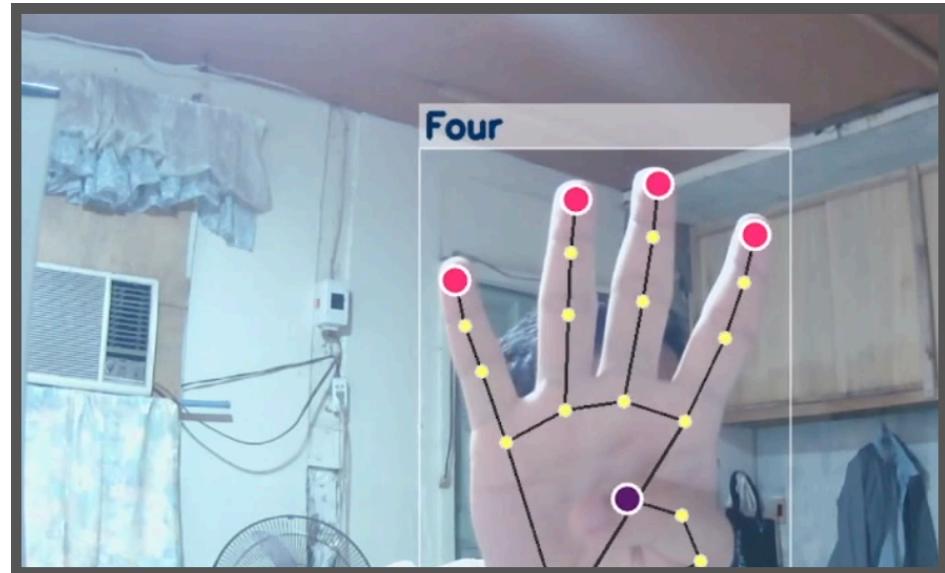
```
INFO: Created TensorFlow XNNPACK delegate for CPU.
```

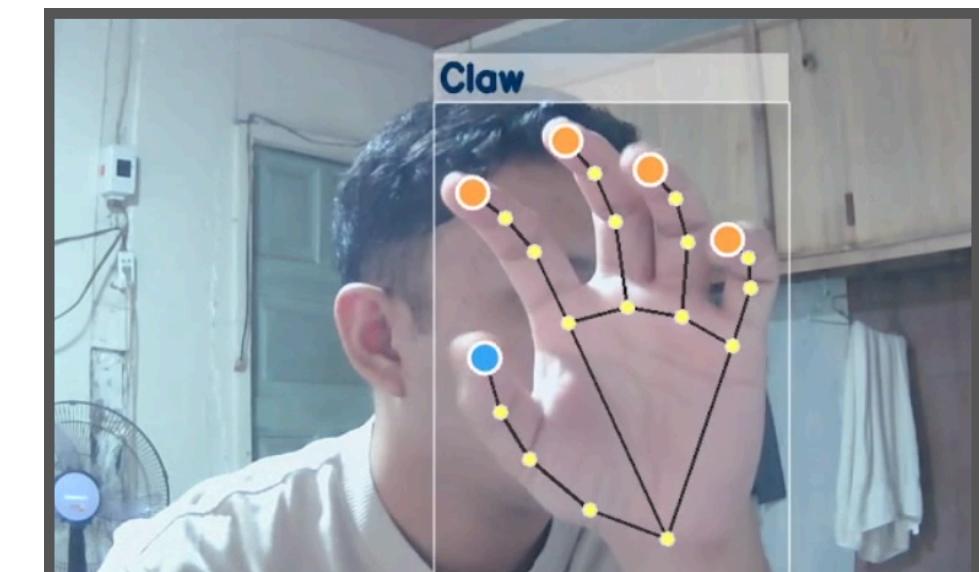
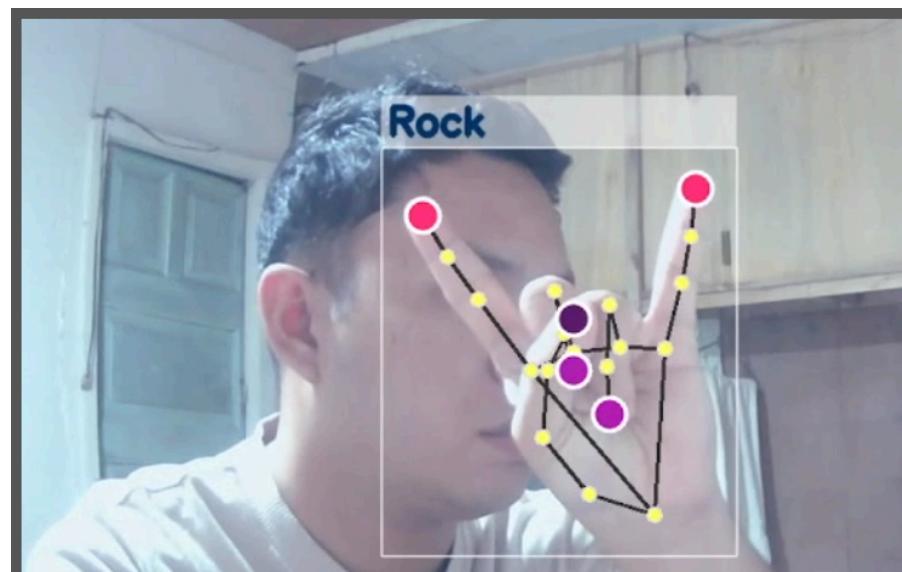
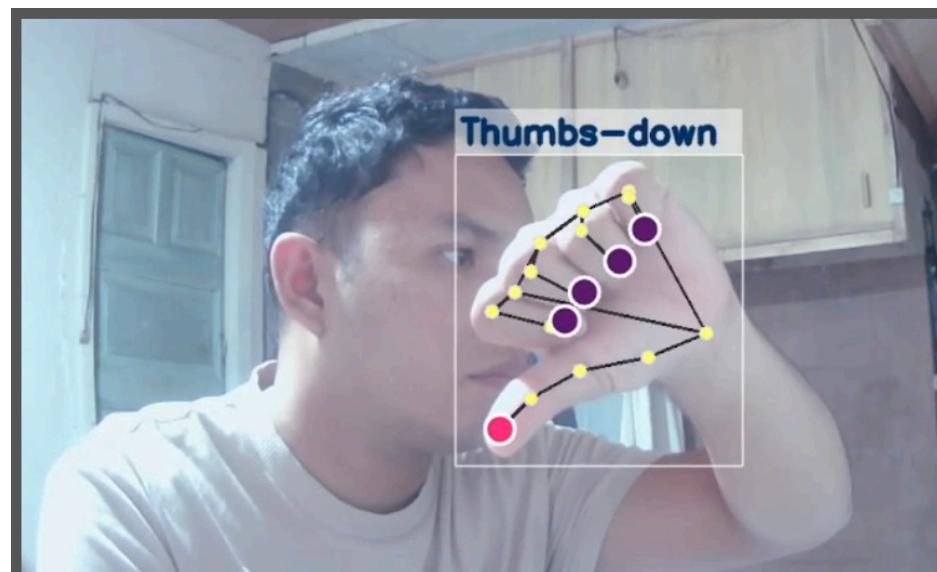
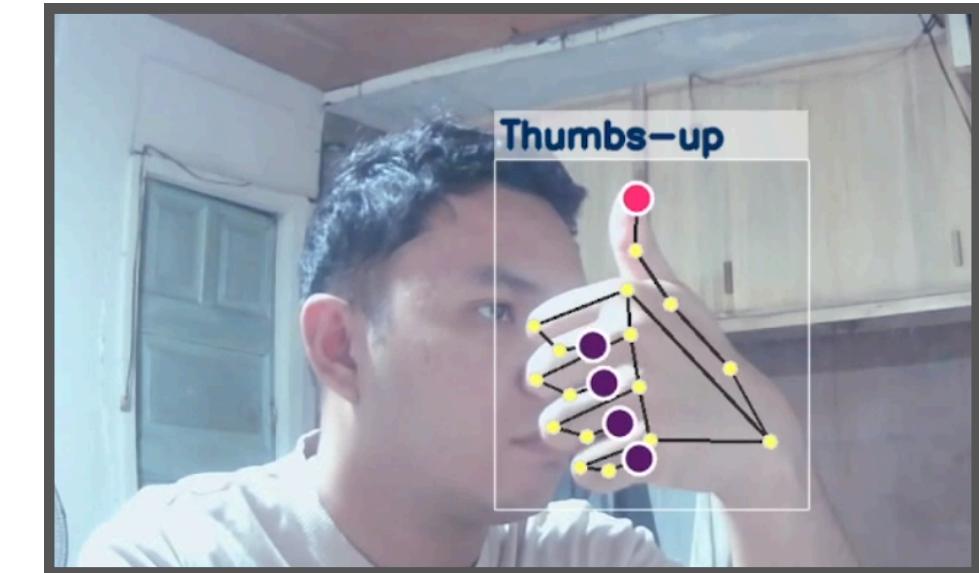
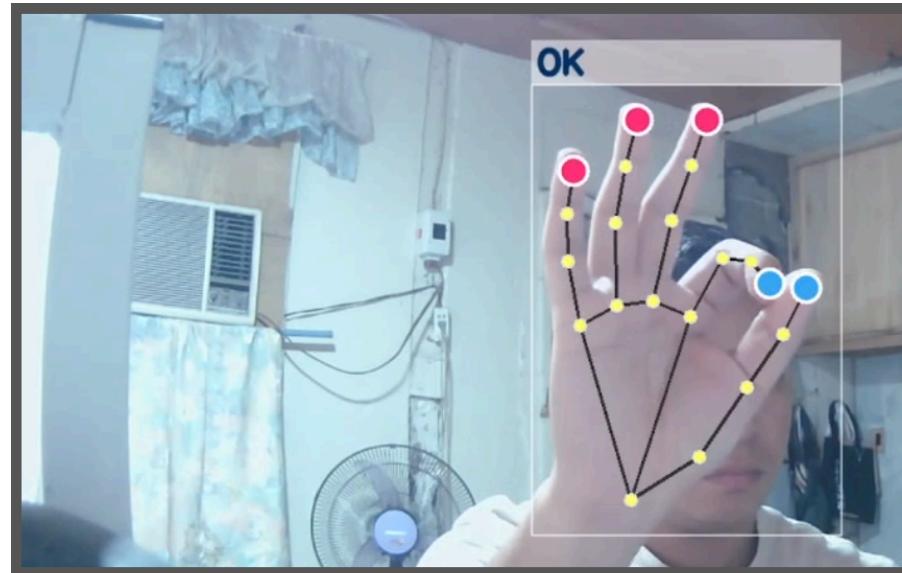
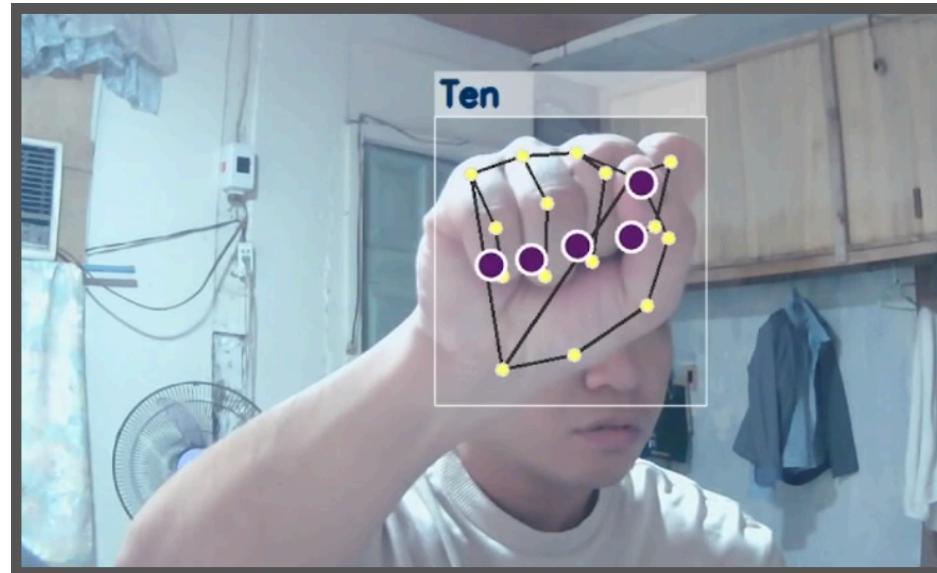
```
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
```

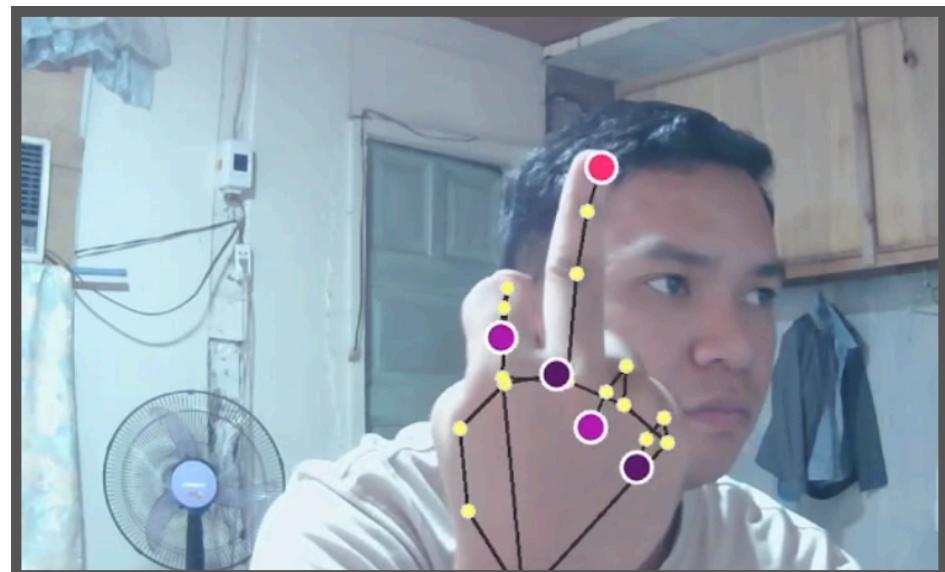
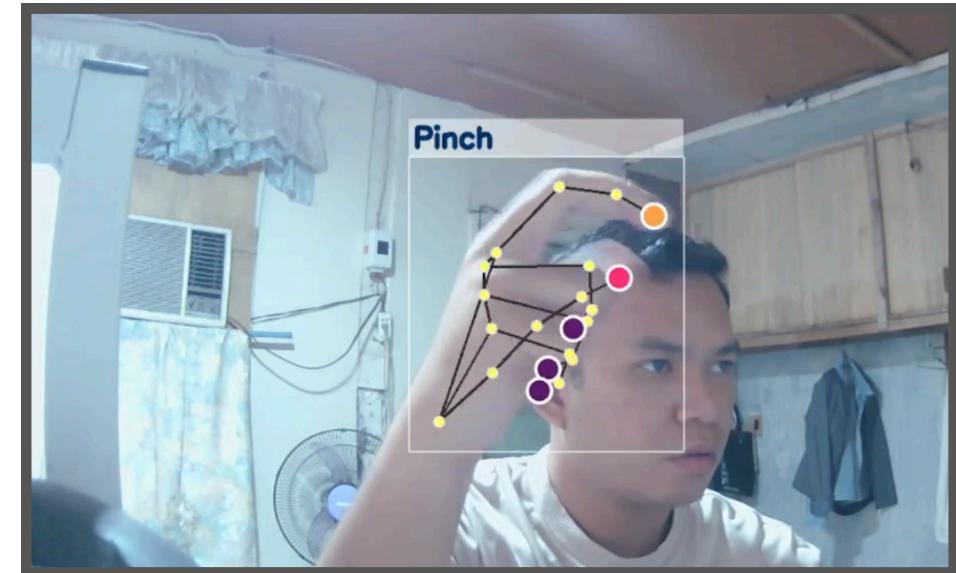
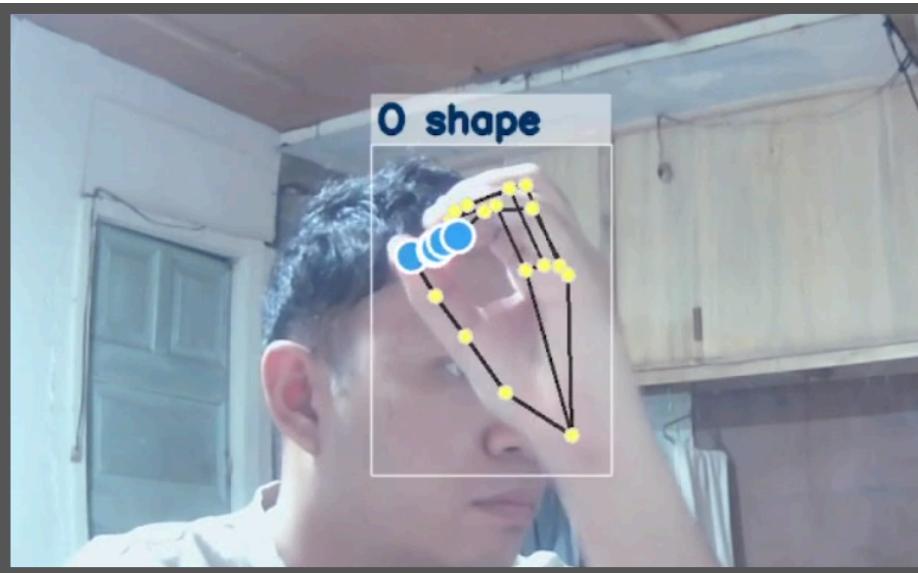
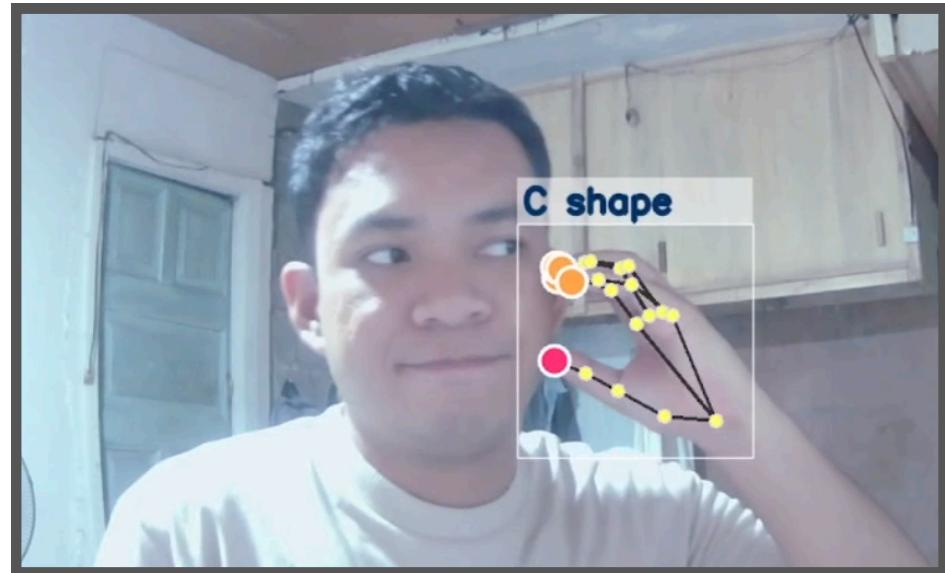
```
W0000 00:00:1758424995.031758 9552 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback tensors.
```

```
W0000 00:00:1758424995.070249 9552 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback tensors.
```

```
W0000 00:00:1758424996.576791 5076 landmark_projection_calculator.cc:186] Using NORM_RECT without IMAGE_DIMENSIONS is only supported for the square ROI. Provide IMAGE_DIMENSIONS or use PROJECTION MATRIX.
```

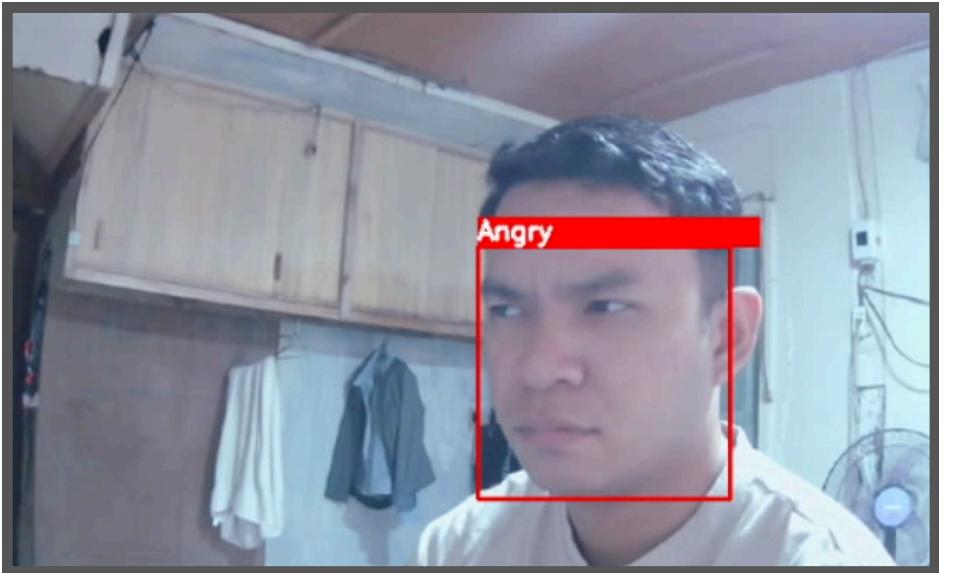






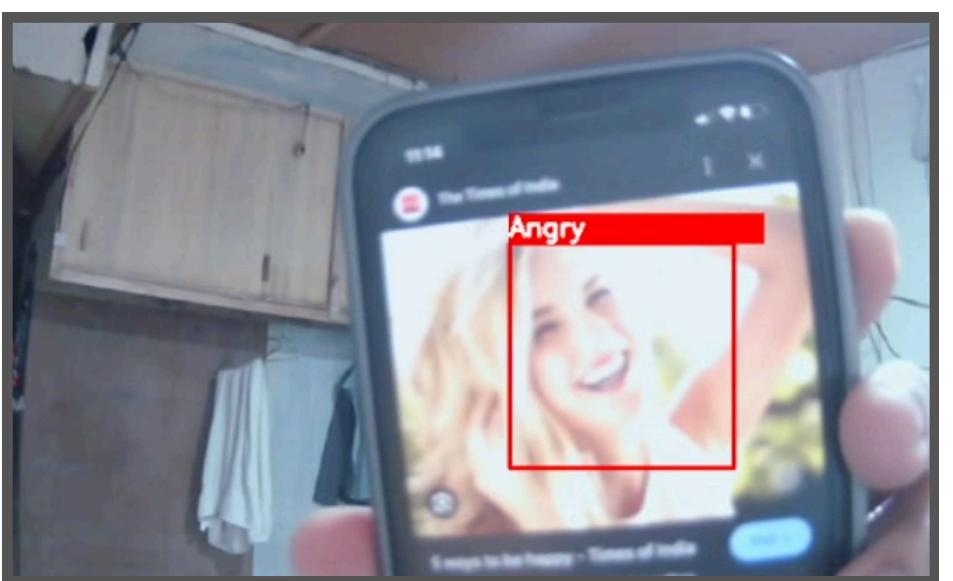
FACIAL EMOTION

FACIAL EMOTION TESTBENCH



A screenshot of a code editor interface. The terminal window shows the command `python facial_emotion.py` being run. The output indicates the model is using TensorFlow Lite XNNPACK delegate for CPU. The code editor shows the `facial_emotion.py` file and other related files like `vgg` and `facial`. The bottom status bar shows the Python version is 3.13.5.

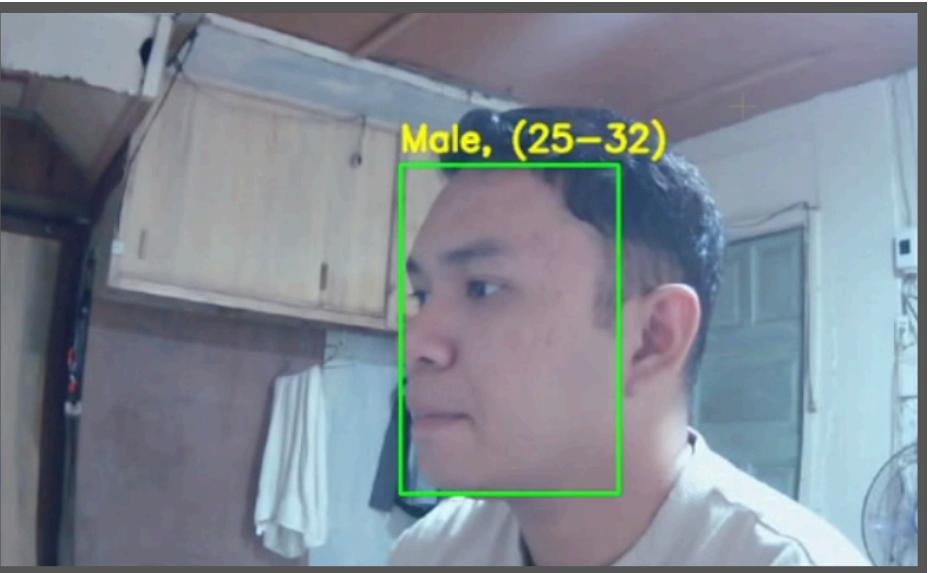
```
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.  
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR  
W0000 00:00:1758424471.631704 [13640 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disabling support for feedback tensors.  
1/1 ┈┈┈┈ 0s 317ms/step  
1/1 ┈┈┈┈ 0s 181ms/step  
1/1 ┈┈┈┈ 0s 48ms/step  
1/1 ┈┈┈┈ 0s 46ms/step  
1/1 ┈┈┈┈ 0s 38ms/step
```



GENDER AND AGE

GENDER AND AGE

TESTBENCH



A screenshot of a terminal window titled "Activity 5" showing Python code for gender and age detection. The code includes constants for model mean values and age groups, and lists for gender and age. The terminal also shows the command to run the script and its execution.

```
gender_and_age_detection > gender_and_age.py > ...
5  # Define constants
6 MODEL_MEAN_VALUES = (78.4263377603, 87.7689143744, 114.895847746)
7 ageList = [(0-2), '(4-6)', '(8-12)', '(15-20)', '(25-32)', '(38-43)', '(48-53)', '(60-100)']
8 genderList = ['Male', 'Female']

300), MODEL_MEAN_VALUES, swapRB=True)

MEMORY XRTOS python - gender_and_age_detection + ~ ...
```

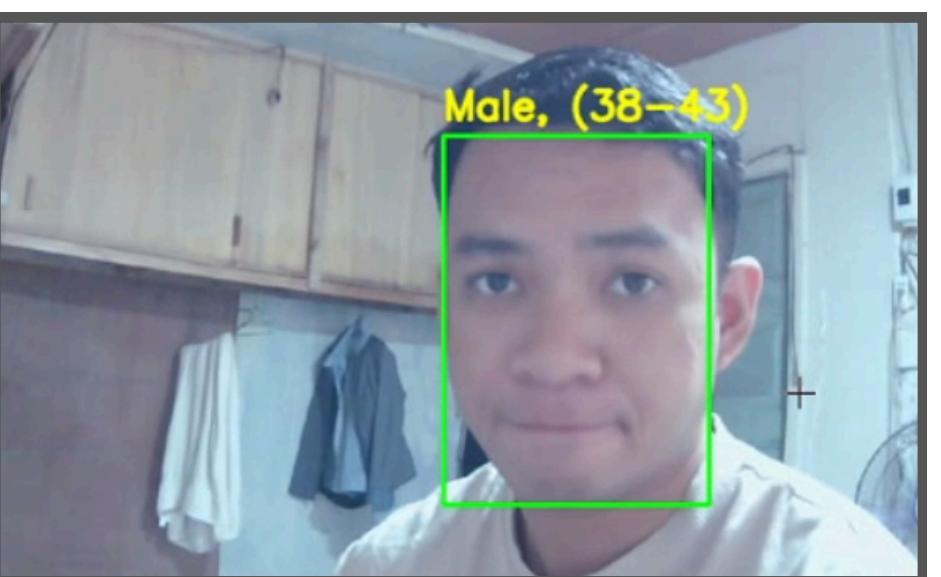
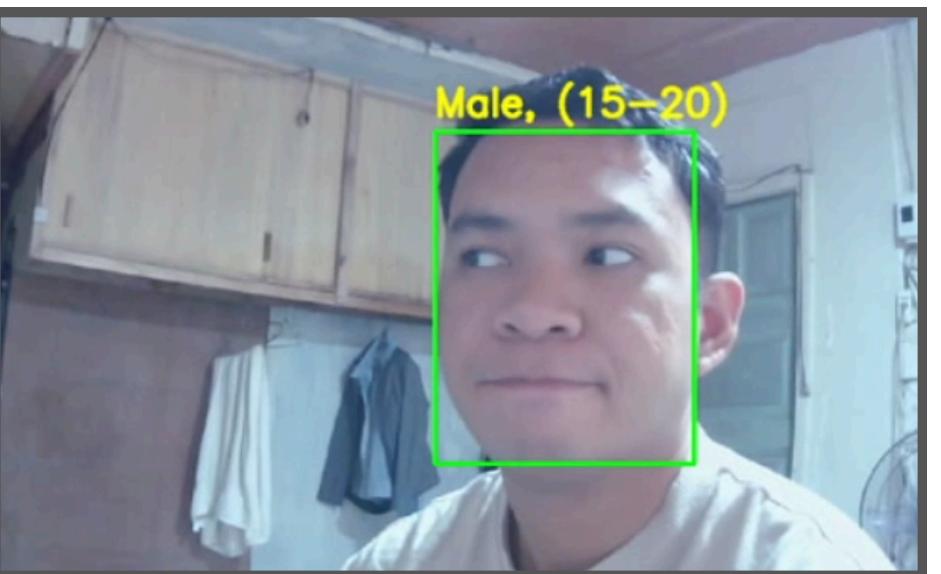
```
r1/anaconda3/Scripts/activate
base
nd_age_detection
ge_detection> python gender_and_age.py
ge_detection> python gender_and_age.py
```

OUTLINE
TIMELINE
Launchpad ⚡ 0 ⚡ 0 28 Sourcery Analytics

Search

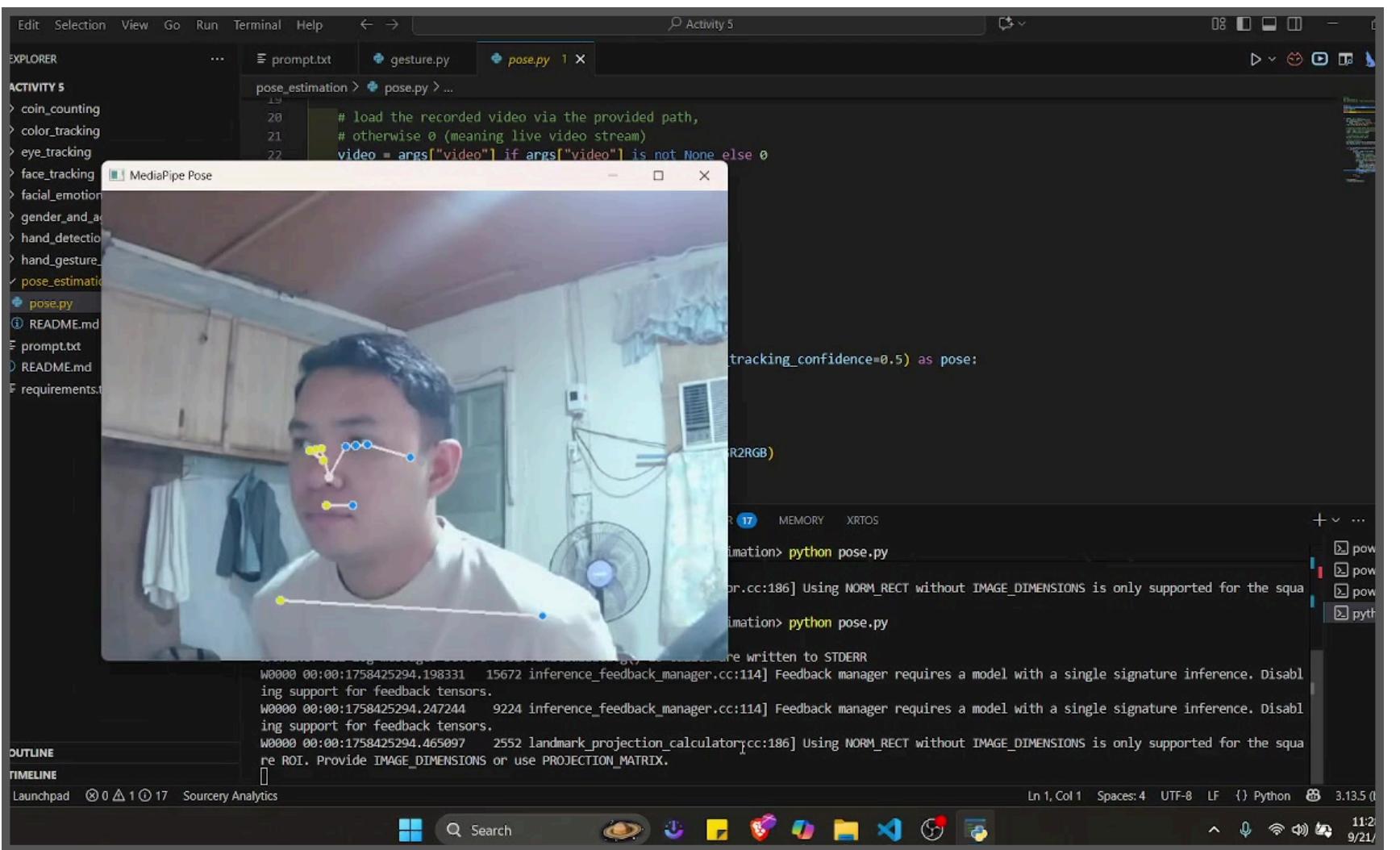
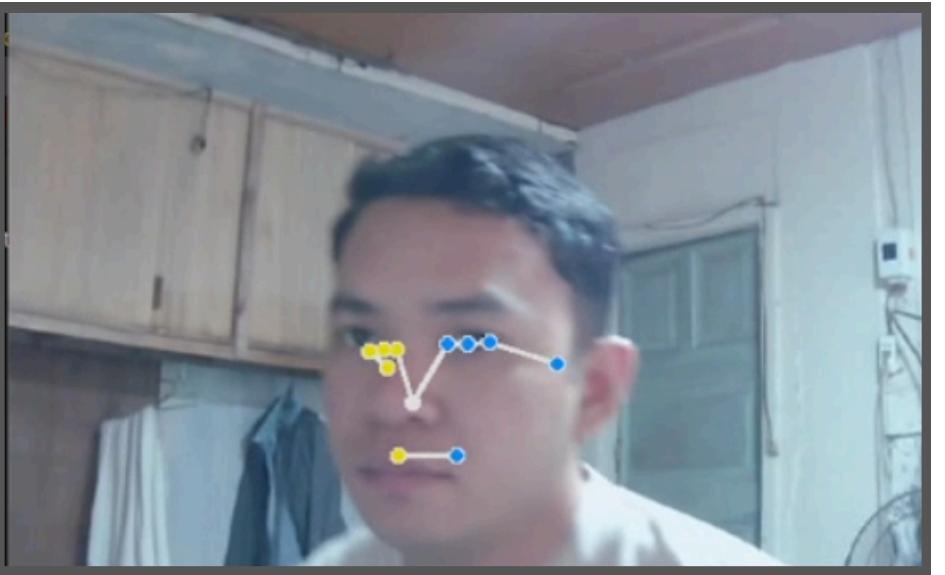
Ln 84, Col 28 Spaces: 4 UTF-8 LF {} Python 3.13.5 ()

11:1 9/21/



POSE ESTIMATION

POSE ESTIMATION TESTBENCH



The screenshot shows a terminal window titled "Activity 5" with the following Python code:

```
pose_estimation > pose.py > ...
20     # load the recorded video via the provided path,
21     # otherwise 0 (meaning live video stream)
22     video = args["video"] if args["video"] is not None else 0
23
24     cap = cv2.VideoCapture(video)
25
26     with mp_pose.Pose(
27         static_image_mode=False,
28         model_complexity=1,
29         enable_segmentation=False,
30         min_detection_confidence=0.5) as pose:
31
32         while cap.isOpened():
33             success, image = cap.read()
34             if not success:
35                 print("Ignoring empty camera frame.")
36                 continue
37
38             # Convert the BGR image to RGB.
39             image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
40
41             # Get landmark results
42             results = pose.process(image)
43
44             # Draw landmark predictions on the image
45             mp_drawing.draw_landmarks(
46                 image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS)
47
48             # Convert the RGB image back to BGR
49             image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
50
51             # Show the image
52             cv2.imshow('MediaPipe Pose', image)
53
54             if cv2.waitKey(5) & 0xFF == 27:
55                 break
56
57         cap.release()
58         cv2.destroyAllWindows()
```

The terminal also displays some log messages related to MediaPipe.

