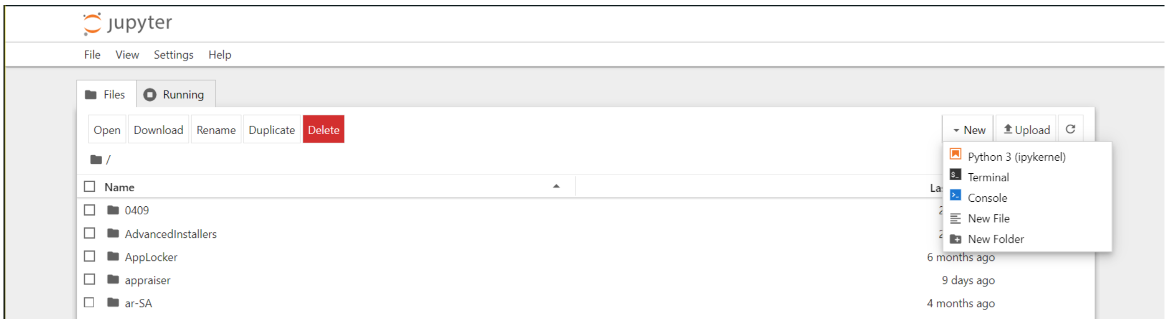
INFOSYS PROJECT - OBJECT TRACKING IN SURVEILLANCE VIDEOS

INSTALL OPENCV IN JUPYTER NOTEBOOK BY USING THE COMMAND

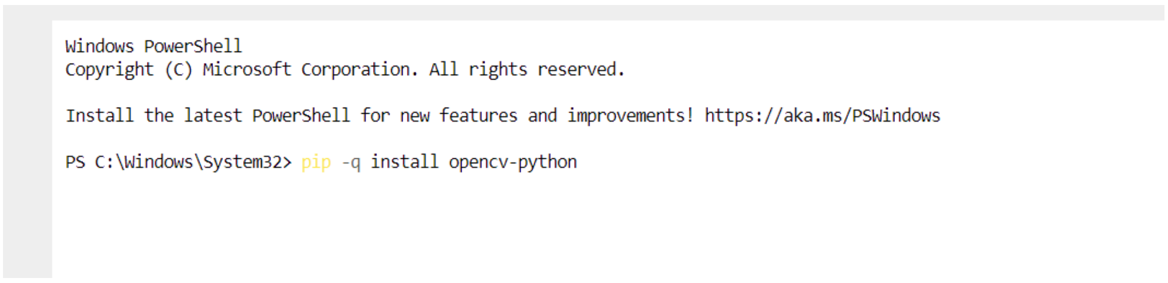
COMMAND -

```
pip -q install opencv-python
```

YOU CAN USE THE TERMINAL OPTION ON JUPYTER NOTEBOOK.



THEN INSTALL OPENCV IN THAT TERMINAL USING THE COMMAND

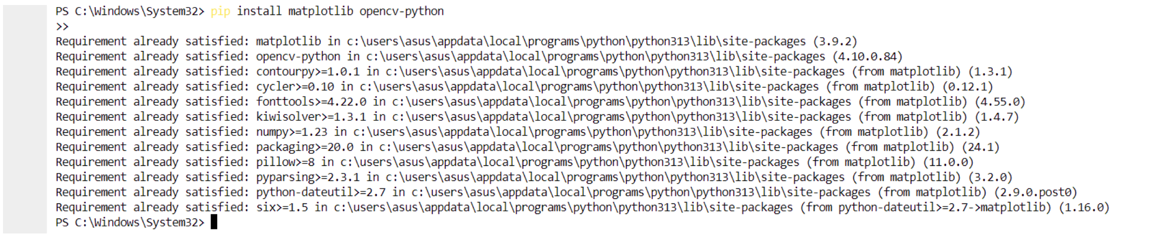


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\System32> pip -q install opencv-python
```

THEN BEFORE PROCEEDING TOWARDS NEXT STEP INSTALL MATPLOTLIB USING -

pip install matplotlib opencv-python

IN MY NOTEBOOK IT IS ALREADY INSTALLED .

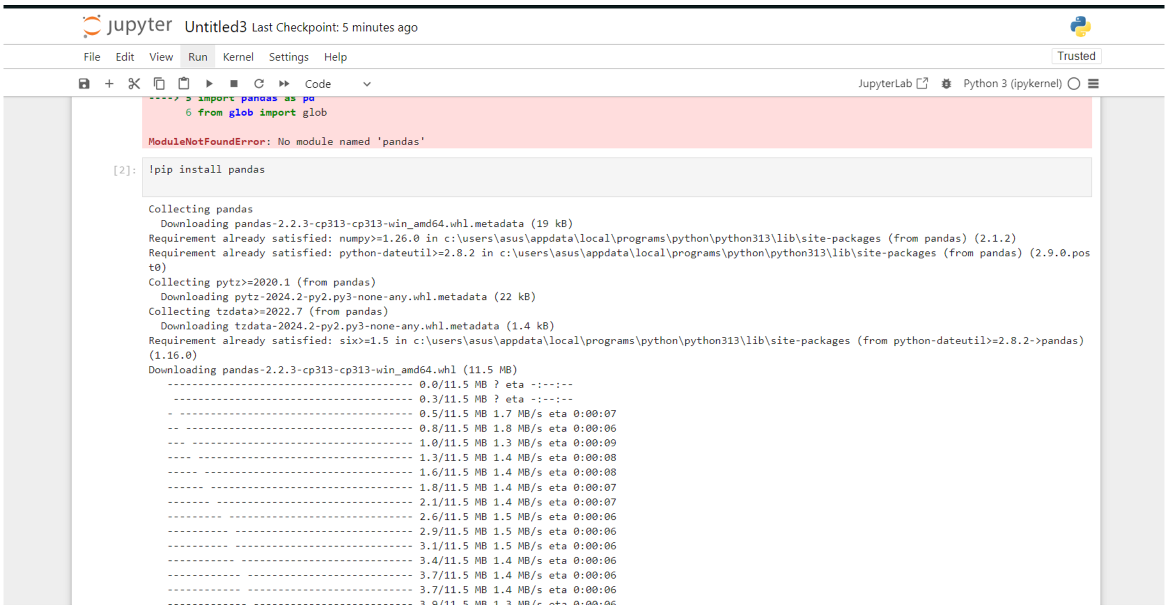


```
PS C:\Windows\System32> pip install matplotlib opencv-python
>>
Requirement already satisfied: matplotlib in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (3.9.2)
Requirement already satisfied: opencv-python in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (4.10.0.84)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.3.1)
Requirement already satisfied: cycler>=0.10 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (4.55.0)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (1.4.7)
Requirement already satisfied: numpy>=1.23 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (2.1.2)
Requirement already satisfied: packaging>=20.0 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
PS C:\Windows\System32>
```

NOW INSTALL PANDA ON THE NOTEBOOK -

!pip install pandas

HERE IS A CHANGE YOU DO NOT REQUIRE TO USE TERMINAL OF THE NOTEBOOK YOU CAN SIMPLY USE THE GIVEN COMMAND ON THE NOTEBOOK ITSELF

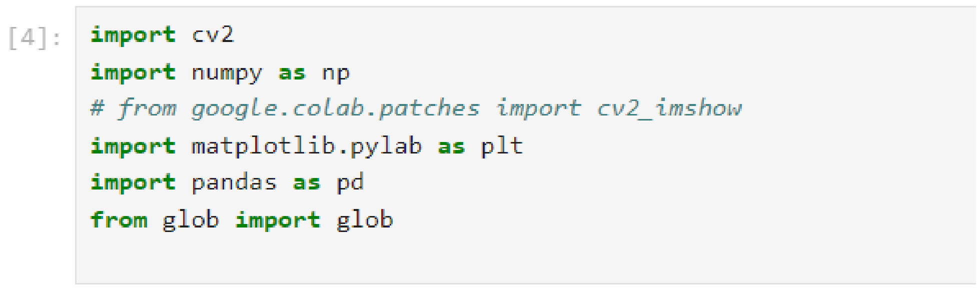

jupyter Untitled3 Last Checkpoint: 5 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
----> 5 import pandas as pd
      6 from glob import glob

ModuleNotFoundError: No module named 'pandas'
```

```
[2]: !pip install pandas
```

```
Collecting pandas
  Downloading pandas-2.2.3-cp313-cp313-win_amd64.whl.metadata (19 kB)
Requirement already satisfied: numpy>=1.26.0 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.1.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from pandas) (2.9.0.pos
t0)
Collecting pytz>=2020.1 (from pandas)
  Downloading pytz-2024.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas)
  Downloading tzdata-2024.2-py2.py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: six>=1.5 in c:\users\asus\appdata\local\programs\python\python313\lib\site-packages (from python-dateutil>=2.8.2->pandas)
(1.16.0)
Downloading pandas-2.2.3-cp313-cp313-win_amd64.whl (11.5 MB)
   ---------------------------------------- 0.0/11.5 MB ? eta -:--:--
    --------------------------------------- 0.3/11.5 MB ? eta -:--:--
   - -------------------------------------- 0.5/11.5 MB 1.7 MB/s eta 0:00:07
   -- ------------------------------------- 0.8/11.5 MB 1.8 MB/s eta 0:00:06
   --- ------------------------------------ 1.0/11.5 MB 1.3 MB/s eta 0:00:09
   ---- ----------------------------------- 1.3/11.5 MB 1.4 MB/s eta 0:00:08
   ----- ---------------------------------- 1.6/11.5 MB 1.4 MB/s eta 0:00:08
   ------ --------------------------------- 1.8/11.5 MB 1.4 MB/s eta 0:00:07
   ------- -------------------------------- 2.1/11.5 MB 1.4 MB/s eta 0:00:07
   --------- ------------------------------ 2.6/11.5 MB 1.5 MB/s eta 0:00:06
   ---------- ----------------------------- 2.9/11.5 MB 1.5 MB/s eta 0:00:06
   ---------- ----------------------------- 3.1/11.5 MB 1.5 MB/s eta 0:00:06
   ----------- ---------------------------- 3.4/11.5 MB 1.4 MB/s eta 0:00:06
   ------------ --------------------------- 3.7/11.5 MB 1.4 MB/s eta 0:00:06
   ------------ --------------------------- 3.7/11.5 MB 1.4 MB/s eta 0:00:06
   ------------- -------------------------- 3.9/11.5 MB 1.3 MB/s eta 0:00:06
```

AS YOU MAY HAVE NOTICE THE RED BOC ABOVE , THAT IS BECAUSE THE PANDA MODULE WAS NOT INSTALLED ON MY NOTEBOOK BUT AFTER INSTALLANTION YOU HAVE TO RUN THE SAME CODE AGAIN AND HERE IS THE NEW RESULT -



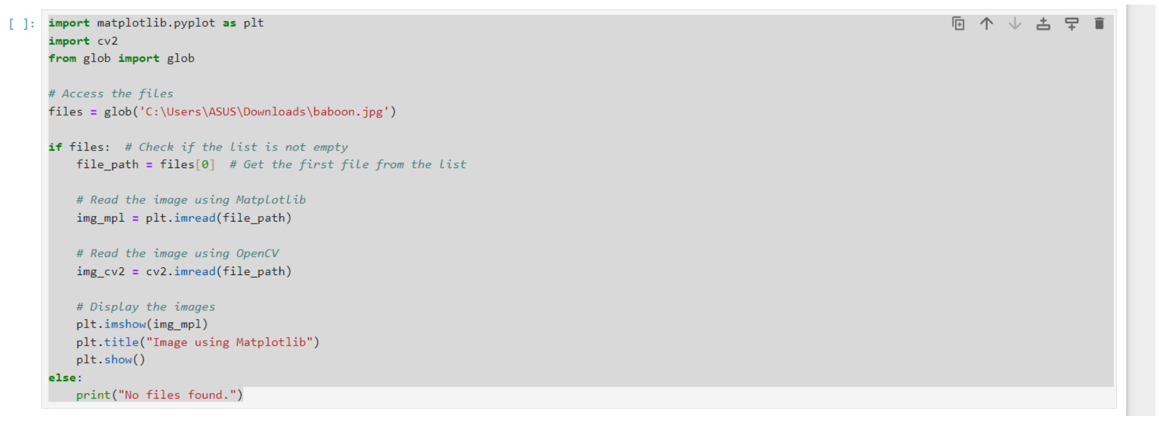
```
[4]: import cv2
     import numpy as np
     # from google.colab.patches import cv2_imshow
     import matplotlib.pylab as plt
     import pandas as pd
     from glob import glob
```

THEN IN THE MAIN PROCESS USE THE GIVEN CODE BELOW -

```
import cv2
import numpy as np
# from google.colab.patches import cv2_imshow
import matplotlib.pylab as plt
import pandas as pd
from glob import glob
```

THEN IN THE NEXT STEP YOU WILL USE THE GIVEN CODE -

```
import matplotlib.pyplot as plt
import cv2
from glob import glob

# Access the files
files = glob('C:\Users\ASUS\Downloads\baboon.jpg')

if files:  # Check if the list is not empty
    file_path = files[0]  # Get the first file from the list

    # Read the image using Matplotlib
    img_mpl = plt.imread(file_path)

    # Read the image using OpenCV
    img_cv2 = cv2.imread(file_path)

    # Display the images
    plt.imshow(img_mpl)
    plt.title("Image using Matplotlib")
    plt.show()
else:
    print("No files found.")BUT IN THIS GIVEN CODE ON THE LINE - files = gl
```



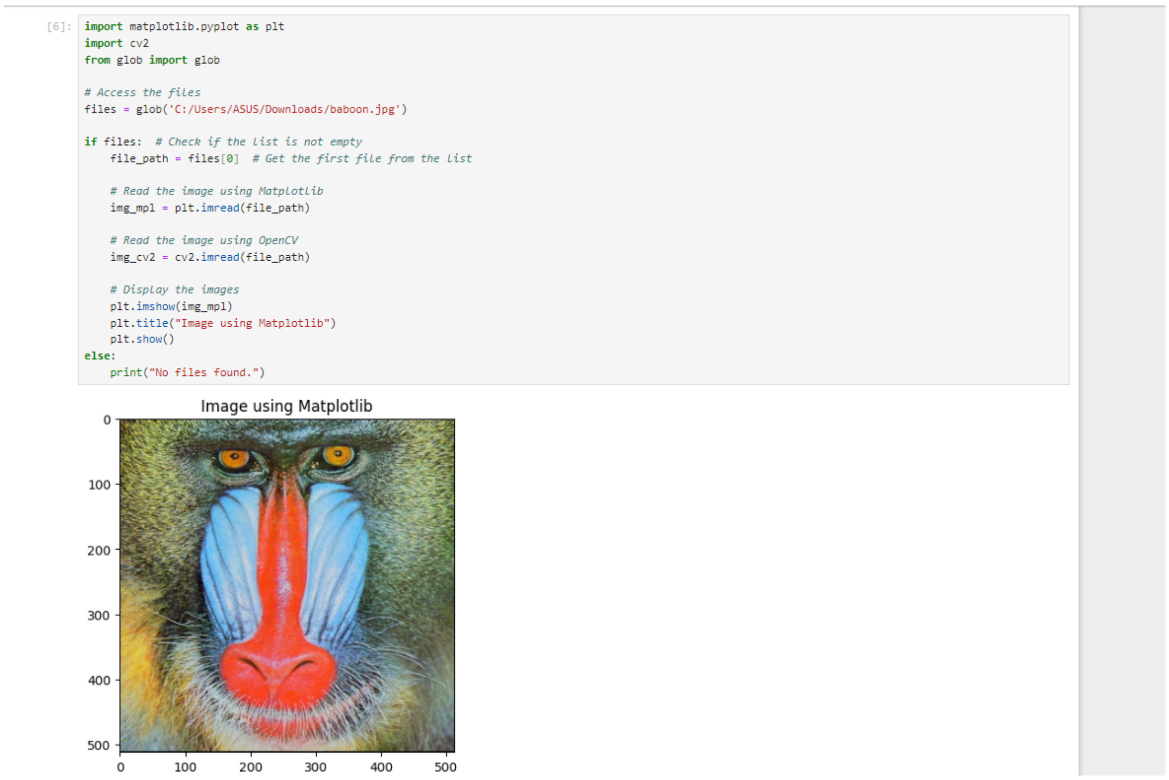
```
[ ]: import matplotlib.pyplot as plt
     import cv2
     from glob import glob

     # Access the files
     files = glob('C:\Users\ASUS\Downloads\baboon.jpg')

     if files:  # Check if the list is not empty
         file_path = files[0]  # Get the first file from the list

         # Read the image using Matplotlib
         img_mpl = plt.imread(file_path)

         # Read the image using OpenCV
         img_cv2 = cv2.imread(file_path)

         # Display the images
         plt.imshow(img_mpl)
         plt.title("Image using Matplotlib")
         plt.show()
     else:
         print("No files found.")
```

BUT BEFORE USING THE COMMAND YOU CAN SEE THE LINE - files = glob (‘C:\Users\ASUS\Downloads\baboon.jpg’)

HERE YOU HAVE TO CHANGE THE LOCATION OF THE PATH WHERE YOU HAVE SAVED THE PHOTO WHICH YOU WANTED TO USE IN THE PROJECT

FOR EXAMPLE I HAVE USED THIS PICTURE -

AFTER USING THE GIVEN CODE THE PHOTO WHILE CHANGE INTO THE GIVEN PHOTO -

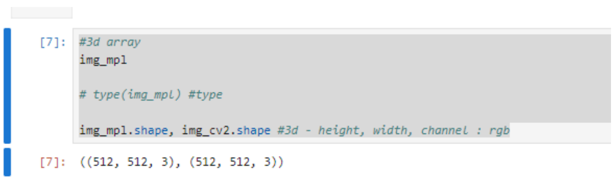


```
[6]: import matplotlib.pyplot as plt
     import cv2
     from glob import glob

     # Access the files
     files = glob('C:/Users/ASUS/Downloads/baboon.jpg')

     if files:  # Check if the list is not empty
         file_path = files[0]  # Get the first file from the list

         # Read the image using Matplotlib
         img_mpl = plt.imread(file_path)

         # Read the image using OpenCV
         img_cv2 = cv2.imread(file_path)

         # Display the images
         plt.imshow(img_mpl)
         plt.title("Image using Matplotlib")
         plt.show()
     else:
         print("No files found.")
```

AFTER GETTING THIS OUTPUT RUN ANOTHER SHELL AND USE THIS CODE -

```
#3d array
img_mpl

# type(img_mpl) #type

img_mpl.shape, img_cv2.shape #3d - height, width, channel : rgb
```

AND THIS IS THE OUTPUT -

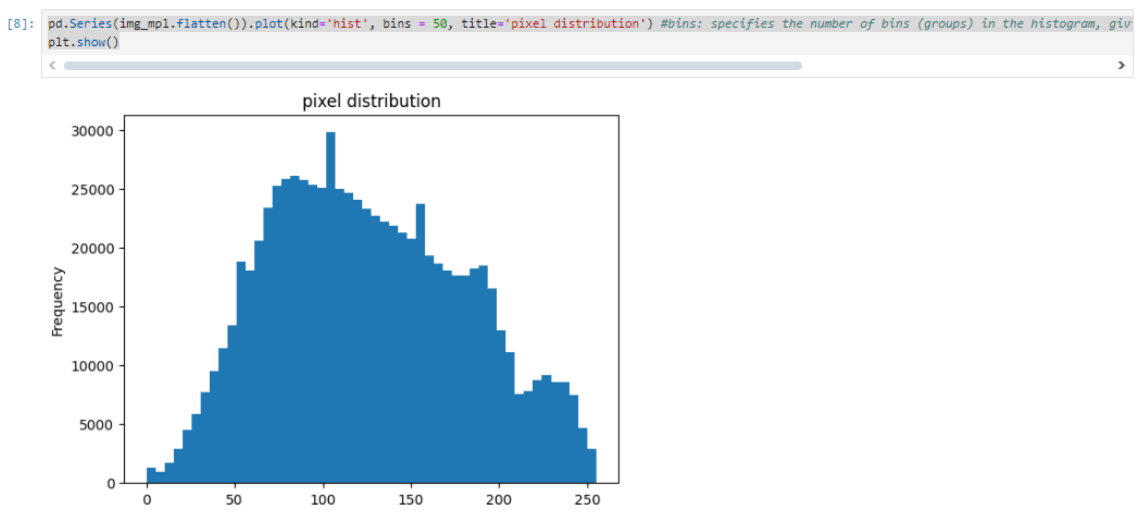


```
[7]: #3d array
     img_mpl

     # type(img_mpl) #type

     img_mpl.shape, img_cv2.shape #3d - height, width, channel : rgb

[7]: ((512, 512, 3), (512, 512, 3))
```

NOW AFTER THAT RUN ANOTHER CODE -

```
pd.Series(img_mpl.flatten()).plot(kind='hist', bins = 50, title='pixel dist
plt.show()
```

AND HERE IS THE PIXEL DISTRIBUTION OUTPUT -

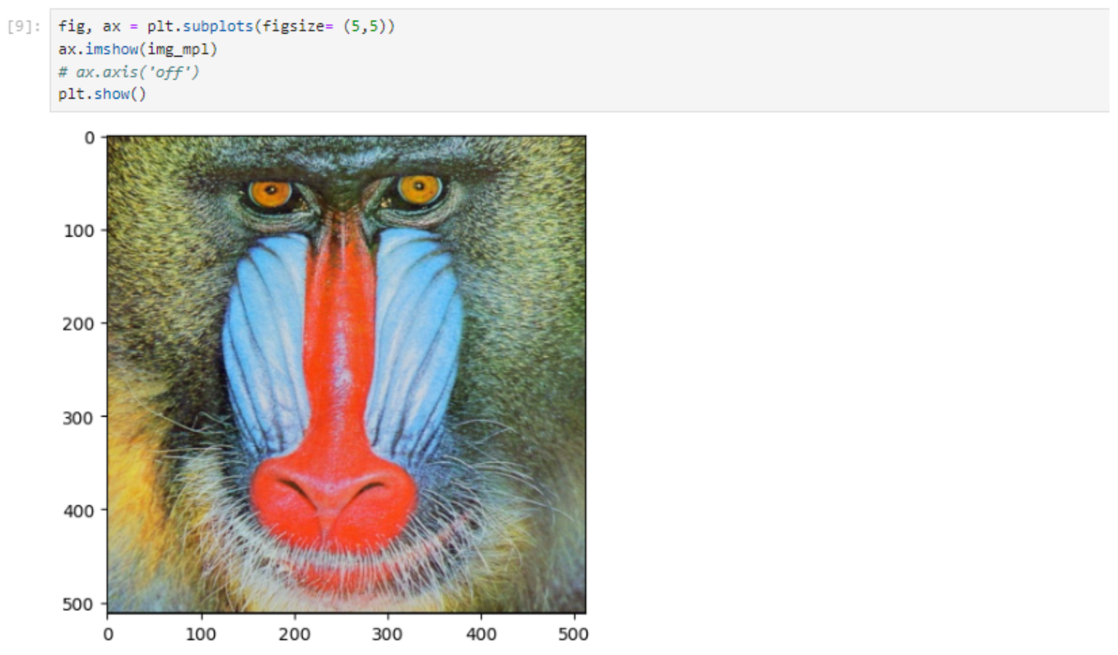


```
[8]: pd.Series(img_mpl.flatten()).plot(kind='hist', bins = 50, title='pixel distribution') #bins: specifies the number of bins (groups) in the histogram, giv
plt.show()
```

AFTER THAT THE NEXT STEP IS TO CREATE FIGURE SIZE OF THE PICTURE IN AXIS TABLE -

```
fig, ax = plt.subplots(figsize= (5,5))
ax.imshow(img_mpl)
# ax.axis('off')
plt.show()
```

AND HERE IS THE OUTPUT -



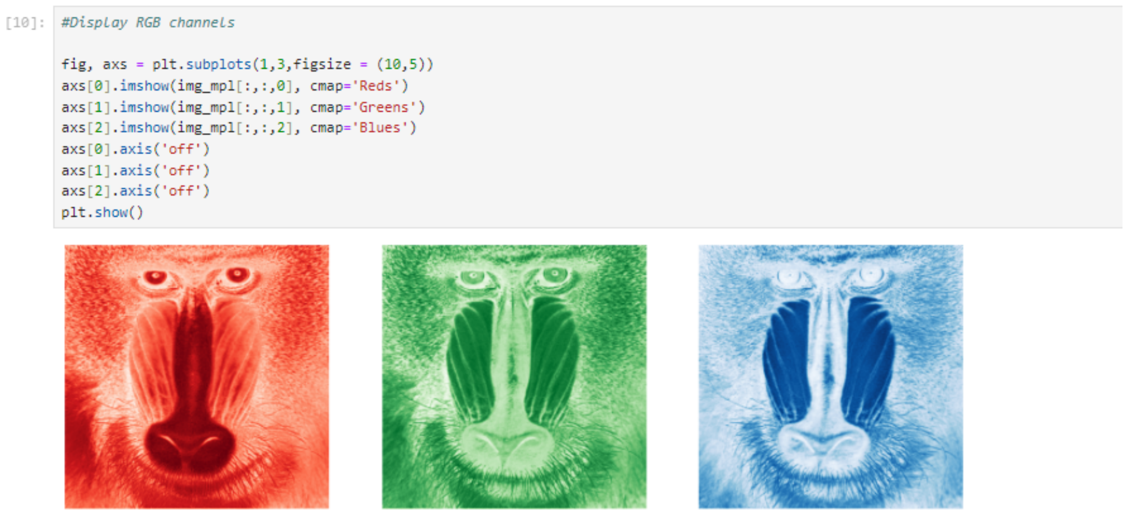
```
[9]: fig, ax = plt.subplots(figsize= (5,5))
ax.imshow(img_mpl)
# ax.axis('off')
plt.show()
```

AFTER THAT , THE NEXT STEP IS TO PUT RGB COLOURS IN THE PHOTO -

```
#Display RGB channels
```

```
fig, axs = plt.subplots(1,3,figsize = (10,5))
axs[0].imshow(img_mpl[:,:,0], cmap='Reds')
axs[1].imshow(img_mpl[:,:,1], cmap='Greens')
axs[2].imshow(img_mpl[:,:,2], cmap='Blues')
axs[0].axis('off')
axs[1].axis('off')
axs[2].axis('off')
plt.show()
```

OUTPUT -

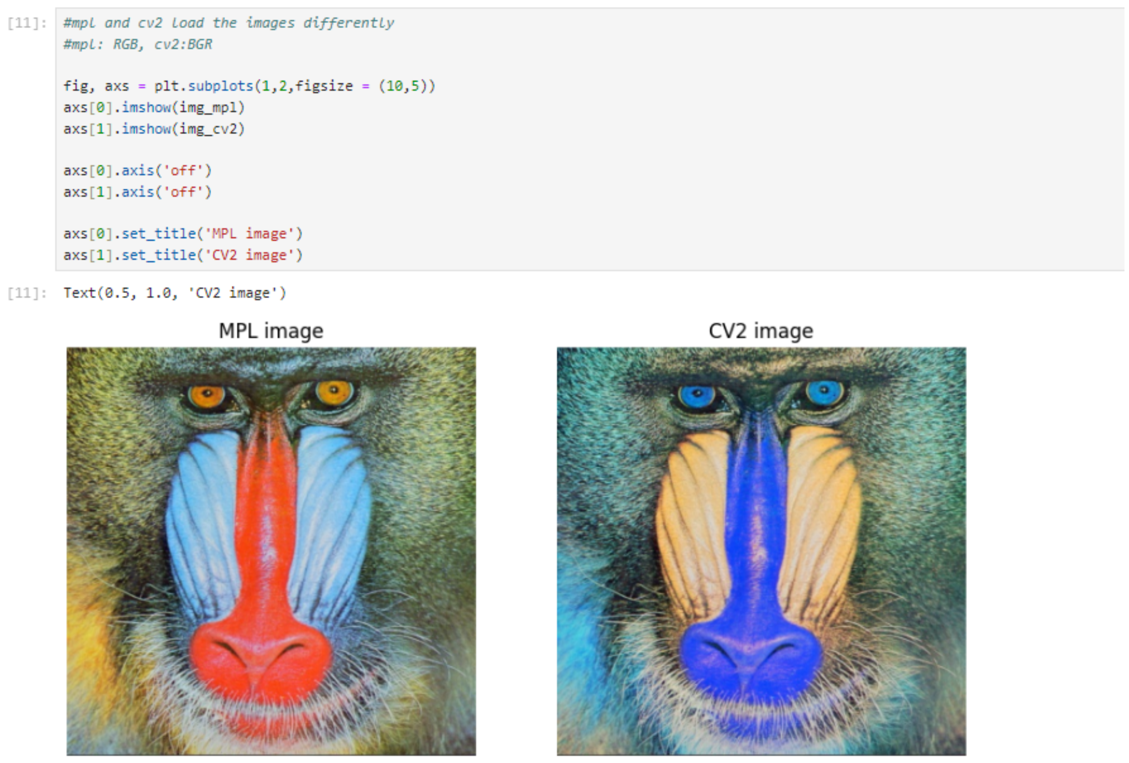


```
[10]: #Display RGB channels

      fig, axs = plt.subplots(1,3,figsize = (10,5))
      axs[0].imshow(img_mpl[:,:,0], cmap='Reds')
      axs[1].imshow(img_mpl[:,:,1], cmap='Greens')
      axs[2].imshow(img_mpl[:,:,2], cmap='Blues')
      axs[0].axis('off')
      axs[1].axis('off')
      axs[2].axis('off')
      plt.show()
```

IN THE NEXT PROCESS WE USE THIS CODE -

```
#mpl and cv2 load the images differently
#mpl: RGB, cv2:BGR

fig, axs = plt.subplots(1,2,figsize = (10,5))
axs[0].imshow(img_mpl)
axs[1].imshow(img_cv2)

axs[0].axis('off')
axs[1].axis('off')

axs[0].set_title('MPL image')
axs[1].set_title('CV2 image')
```

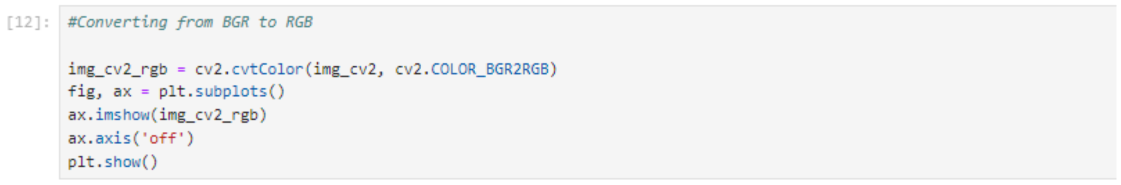
THE OUTPUT -



```
[11]: #mpl and cv2 load the images differently
      #mpl: RGB, cv2:BGR

      fig, axs = plt.subplots(1,2,figsize = (10,5))
      axs[0].imshow(img_mpl)
      axs[1].imshow(img_cv2)

      axs[0].axis('off')
      axs[1].axis('off')

      axs[0].set_title('MPL image')
      axs[1].set_title('CV2 image')

[11]: Text(0.5, 1.0, 'CV2 image')
```

```
#Converting from BGR to RGB

img_cv2_rgb = cv2.cvtColor(img_cv2, cv2.COLOR_BGR2RGB)
fig, ax = plt.subplots()
ax.imshow(img_cv2_rgb)
ax.axis('off')
plt.show()
```

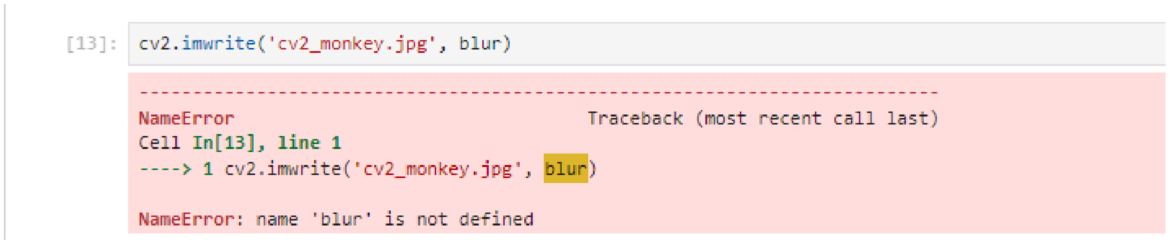
OUTPUT -



```
[12]: #Converting from BGR to RGB

      img_cv2_rgb = cv2.cvtColor(img_cv2, cv2.COLOR_BGR2RGB)
      fig, ax = plt.subplots()
      ax.imshow(img_cv2_rgb)
      ax.axis('off')
      plt.show()
```

AFTER THIS , IN THE NEXT STEP WE WILL CHECK IF THE IMAGE IS BLUR OR NOT , SO USE THIS CODE - cv2.imwrite('cv2_monkey.jpg', blur)

BUT IF AFTER USING THIS CODE , IF YOU GOT A PROBLEM LIKE THIS -



```
[13]: cv2.imwrite('cv2_monkey.jpg', blur)

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 cv2.imwrite('cv2_monkey.jpg', blur)

NameError: name 'blur' is not defined
```

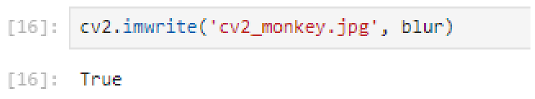
THEN USE THIS CODE TO RECTIFY THIS PROCESS -

```
import cv2

# Read the image
img = cv2.imread('C:/Users/ASUS/Downloads/baboon.jpg')

if img is not None:  # Ensure the image was successfully loaded
    # Apply Gaussian Blur
    blur = cv2.GaussianBlur(img, (15, 15), 0)

    # Save the blurred image
    save_status = cv2.imwrite('C:/Users/ASUS/Downloads/cv2_monkey.jpg', blu
    if save_status:
        print("Image successfully saved as 'cv2_monkey.jpg'")
    else:
        print("Error: Could not save the image.")
else:
    print("Error: Could not load the image. Check the file path.")
```

THIS CODE WILL FIX THE ERROR AND ALSO DO NOT FORGET TO CHANGE THE IMAGE PATH WHICH I HAVE MENTIONED ON THE CODE .

BUT THERE WAS NO ERROR THEN THEIR IS NO NEED TO USE THIS CODE .

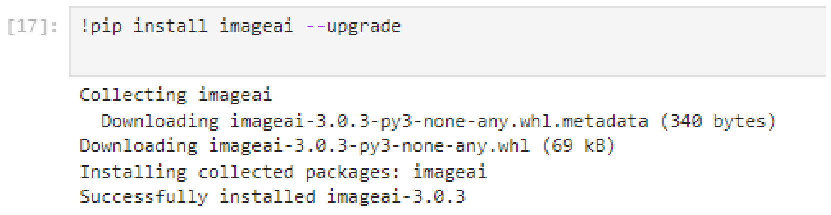
THE OUTPUT OF THE CODE - cv2.imwrite('cv2_monkey.jpg', blur) IS -



```
[16]: cv2.imwrite('cv2_monkey.jpg', blur)

[16]: True
```

IN THIS STEP WE WILL UPGRADE THE PICTURE USING -

!pip install imageai --upgrade



```
[17]: !pip install imageai --upgrade

      Collecting imageai
        Downloading imageai-3.0.3-py3-none-any.whl.metadata (340 bytes)
      Downloading imageai-3.0.3-py3-none-any.whl (69 kB)
      Installing collected packages: imageai
      Successfully installed imageai-3.0.3
```

THE YOLO FILE PROJECT

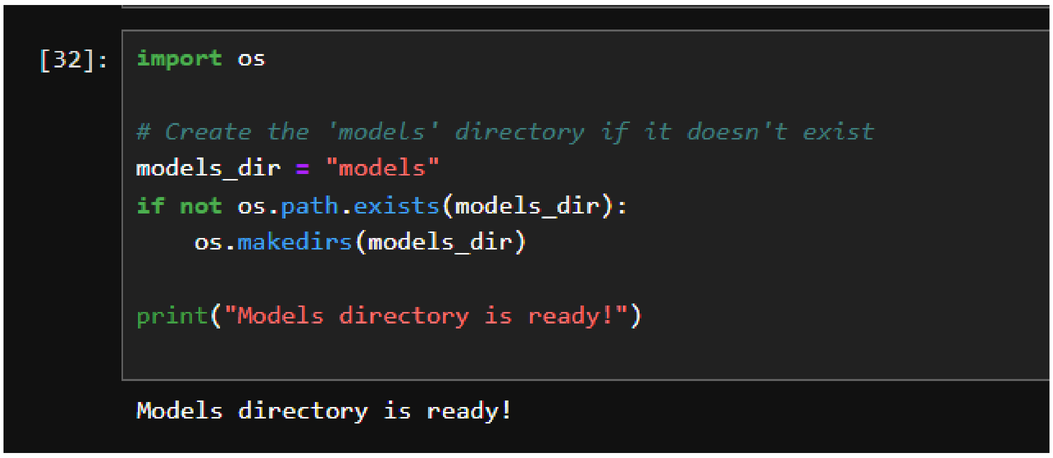
HERE YOU HAVE TO DOWNLOAD THE YOLO FILE - tiny-yolov3.pt

download

I HAVE EMBEDED A LINK WHICH WILL REDIRECT YOU TO THE DOWNLOADING PAGE OF THE YOLOV3 FILE.

AFTER DOWNLOADING THE FILE , IN THE NEXT STEP YOU HAVE TO CREATE THE MODELS FOLDER ON JUPYTER.

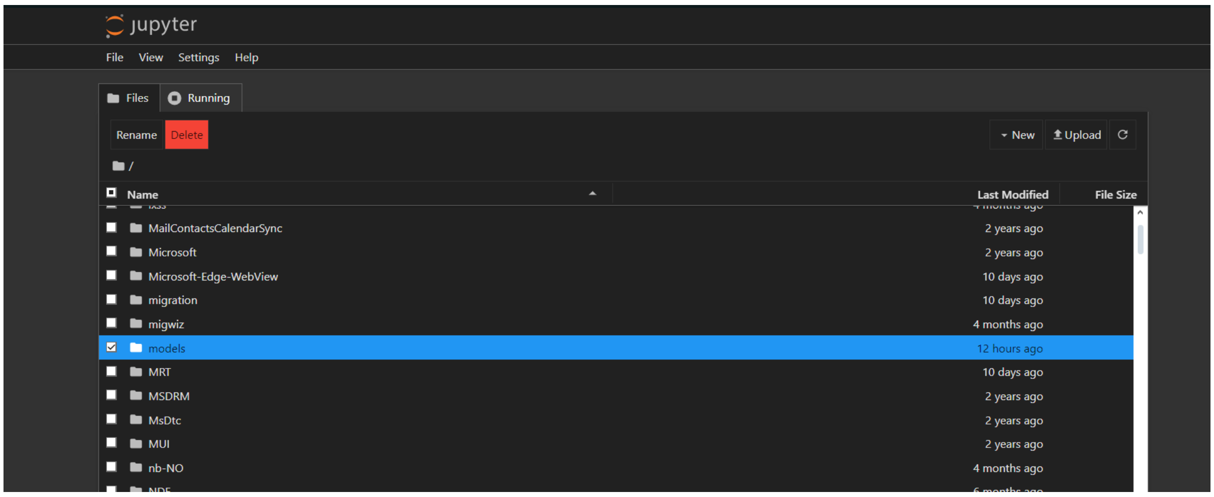
```
import os

# Create the 'models' directory if it doesn't exist
models_dir = "models"
if not os.path.exists(models_dir):
    os.makedirs(models_dir)

print("Models directory is ready!")
```



```
[32]: import os

      # Create the 'models' directory if it doesn't exist
      models_dir = "models"
      if not os.path.exists(models_dir):
          os.makedirs(models_dir)

      print("Models directory is ready!")

      Models directory is ready!
```

USING THE GIIVEN CODE YOUR MODELS FOLDER WILL BE READY .

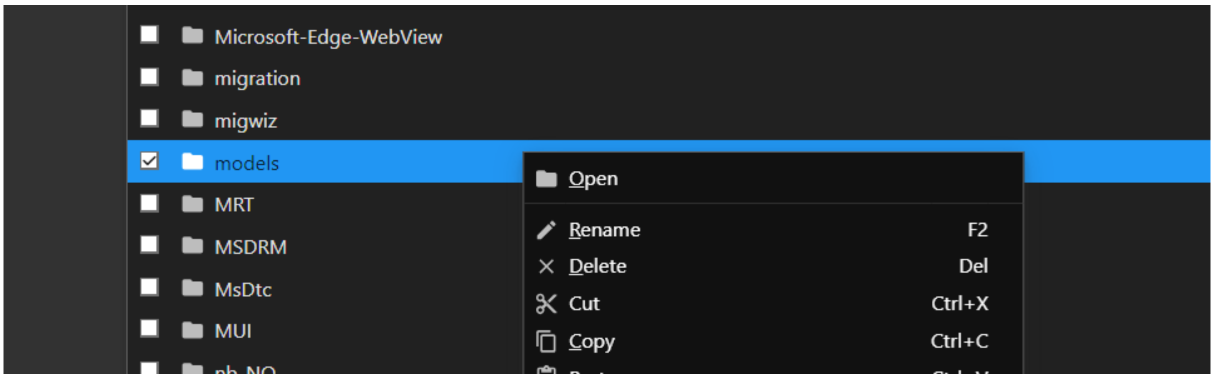
NOW YOU CAN ACCESS THE MODEL FOLDER ON THE HOME OF JUPYTER NOTEBOOK.



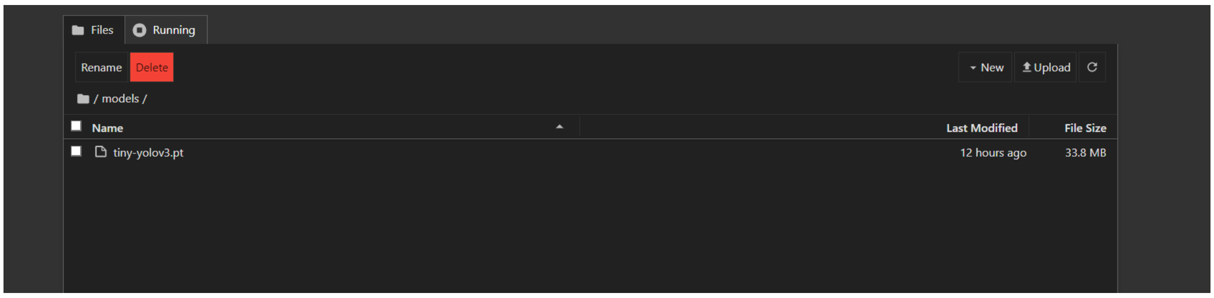
Name	Last Modified	File Size
MailContactsCalendarSync	2 years ago	
Microsoft	2 years ago	
Microsoft-Edge-WebView	10 days ago	
migration	10 days ago	
migwiz	4 months ago	
✓ models	12 hours ago	
MRT	10 days ago	
MSDRM	2 years ago	
MsDtc	2 years ago	
MUI	2 years ago	
nb-NO	4 months ago	

YOU CAN SEE A ✔️ MARK ON THE SIDE OF THE FOLDER

THEN RIGHT-CLICK ON THE FOLDER THEN OPEN THE FOLDER

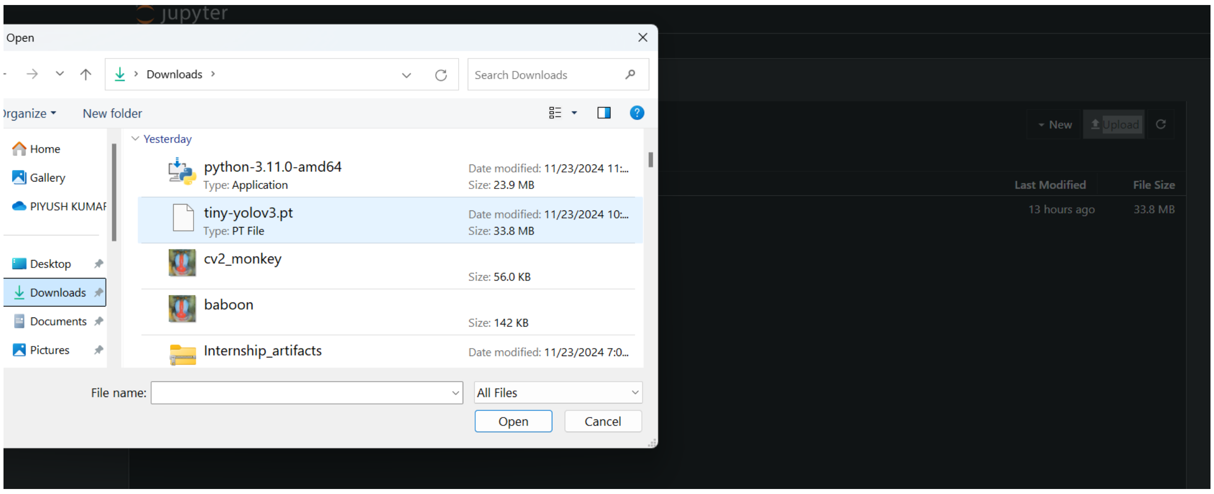


AFTER THIS YOU CAN SEE I HAVE ALREADY UPLOADED THE DOWNLOADED YOLO FILE IN THE MODEL FOLDER



Name	Last Modified	File Size
tiny-yolov3.pt	12 hours ago	33.8 MB

AFTER OPENING THE MODEL FOLDER THEN CLICK UPLOAD AND IT WILL NAVIGATE YOU TO THE FILE EXPLORER THEN CHOSE THE DOWNLOADED FILE AND IT WILL BE UPLOADED TO THE FOLDER.



Name	Type	Date modified	Size
python-3.11.0-amd64	Application	11/23/2024 11:...	23.9 MB
tiny-yolov3.pt	PT File	11/23/2024 10:...	33.8 MB
cv2_monkey			56.0 KB
baboon			142 KB
Internship_artifacts		11/23/2024 7:0...	

THEN YOU ARE READY TO GO !

IN THIS VERY NEXT STEP YOU NEED TO DO SOME MORE INSTALLATION OF MODULES LIKE - torch torchvision torchaudio

```
!pip install torch torchvision torchaudio
```

AFTER INSTALLATION USE THE GIVEN CODE FOR CONFIRMATION -

```
import torch
print(torch.__version__)
print("CUDA available:", torch.cuda.is_available())
```

IN THIS PROCESS WE WILL INSTALL OPEN-CV - PYTHON - HEADLESS

```
!pip install opencv-python-headless
```

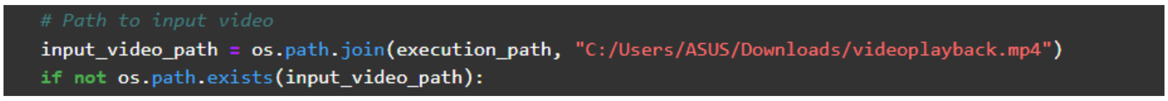
WHY IT IS NEED BECAUSE IT IS THE FULL VERSION OF OPEN-CV WITH ADDITIONAL FEATURES

LAST AND FINAL STEP -

CODE -

```
!pip install imageai --upgrade

from imageai.Detection import VideoObjectDetection
import os
import time

# Start timing
start_time = time.time()

# Set the execution path
execution_path = os.getcwd()

# Create the 'models' directory if it doesn't exist
os.makedirs("models", exist_ok=True)

def forFrame(frame_number, output_array, output_count):
    print("FOR FRAME ", frame_number)
    print("Output for each object : ", output_array)
    print("Output count for unique objects : ", output_count)
    print("------------END OF A FRAME --------------")

def forSeconds(second_number, output_arrays, count_arrays, average_output_c
    print("SECOND : ", second_number)
    print("Array for the outputs of each frame ", output_arrays)
    print("Array for output count for unique objects in each frame : ", cou
    print("Output average count for unique objects in the last second: ", a
    print("------------END OF A SECOND --------------")

def forMinute(minute_number, output_arrays, count_arrays, average_output_co
    print("MINUTE : ", minute_number)
    print("Array for the outputs of each frame ", output_arrays)
    print("Array for output count for unique objects in each frame : ", cou
    print("Output average count for unique objects in the last minute: ", a
    print("------------END OF A MINUTE --------------")

# Initialize the VideoObjectDetection
video_detector = VideoObjectDetection()
video_detector.setModelTypeAsTinyYOLOv3()

# Path to the model file
model_path = os.path.join(execution_path, "models/tiny-yolov3.pt")
if not os.path.exists(model_path):
    print("Model file not found. Please download tiny-yolov3.pt and place i
else:
    video_detector.setModelPath(model_path)
    video_detector.loadModel()

    # Path to input video
    input_video_path = os.path.join(execution_path, "C:/Users/ASUS/Download
    if not os.path.exists(input_video_path):
        print(f"Input video file not found at {input_video_path}. Please pr
    else:
        # Detect objects in the video
        output_video_path = os.path.join(execution_path, "output_video.mp4"
        video_detector.detectObjectsFromVideo(
            input_file_path=input_video_path,
            output_file_path=output_video_path,
            frames_per_second=10,
            per_second_function=forSeconds,
            per_frame_function=forFrame,
            per_minute_function=forMinute,
            minimum_percentage_probability=30
        )

        # End timing and calculate the duration
        end_time = time.time()
        execution_duration = end_time - start_time

        print("Video saved at:", output_video_path)
        print("Time taken to run the code:", execution_duration, "seconds")
```

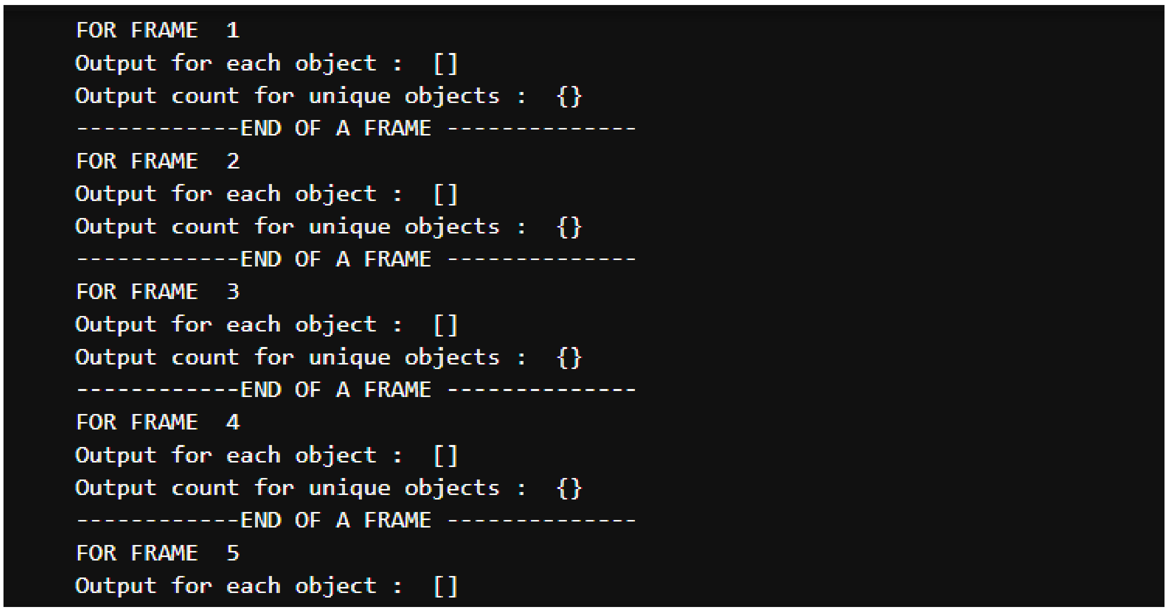
BEFORE EXECUTING THE ENTIRE CODE YOU NEED TO DOWNLOAD ANY SAMPLE VIDEO FROM ONLINE AND THEN YOU HAVE TO COPY THE PATH OF THE VIDEO FROM THE FILE EXPLORER ON YOUR SYSTEM AND PASTE IT ONE THE CODE WHERE I HAVE MENTIONED THE CODE -



```
# Path to input video
input_video_path = os.path.join(execution_path, "C:/Users/ASUS/Downloads/videoplayback.mp4")
if not os.path.exists(input_video_path):
```

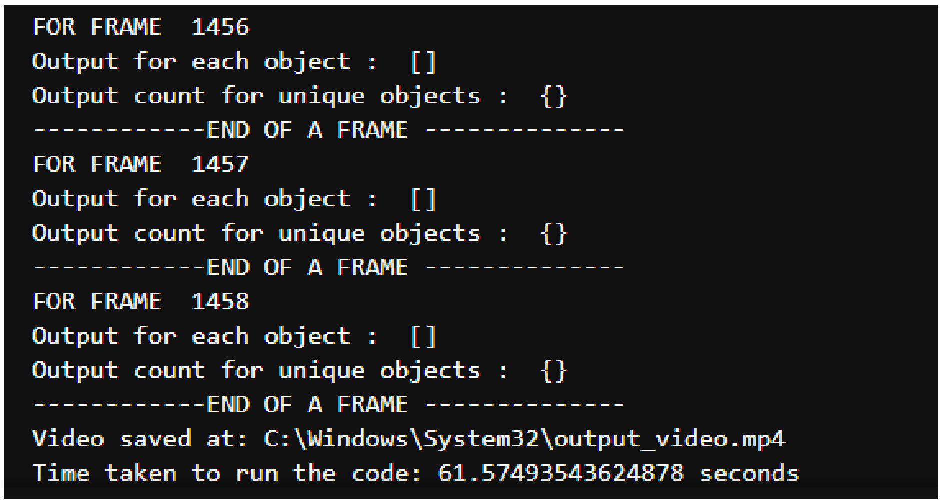
"C:\Users\ASUS\Downloads\videoplayback.mp4" - THIS IS THE PATH OF MY VIDEO WHICH I HAVE INSERTED ON THE CODE MENTIONED ABOVE .

NOW AFTER RUNNING THE WHOLE PROGRAM IF YOU GOT THE OUTPUT LIKE THIS -



```
FOR FRAME  1
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  2
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  3
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  4
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  5
Output for each object :  []
```

THEN YOU ARE GOOD TO GO BUT THIS IS THE BEGINNING OUTPUT , AFTER SCROLLING DOWN TO THE END YOU WILL SEE -



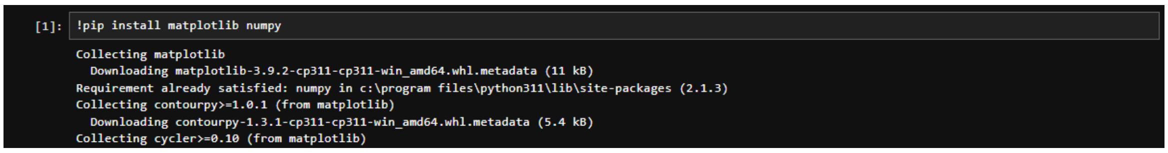
```
FOR FRAME  1456
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  1457
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  1458
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
Video saved at: C:\Windows\System32\output_video.mp4
Time taken to run the code: 61.57493543624878 seconds
```

THIS SHOWS THE CODE EXECUTION IS COMPLETED.

DNN MODULE PROJECT

IN ORDER TO DO THIS MODEL PROJECT YOU NEED TO HAVE SOME MODULES INSTALLED ON YOUR JUPYTER NOTEBOOK.

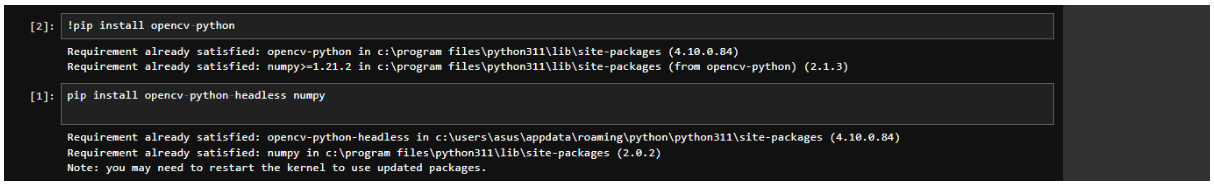
FIRST ONE IS - !pip install matplotlib numpy



```
[1]: !pip install matplotlib numpy

     Collecting matplotlib
       Downloading matplotlib-3.9.2-cp311-cp311-win_amd64.whl.metadata (11 kB)
     Requirement already satisfied: numpy in c:\program files\python311\lib\site-packages (2.1.3)
     Collecting contourpy>=1.0.1 (from matplotlib)
       Downloading contourpy-1.3.1-cp311-cp311-win_amd64.whl.metadata (5.4 kB)
     Collecting cycler>=0.10 (from matplotlib)
```

SECOND IS - !pip install opencv-python

pip install opencv-python-headless numpy



```
[2]: !pip install opencv-python

     Requirement already satisfied: opencv-python in c:\program files\python311\lib\site-packages (4.10.0.84)
     Requirement already satisfied: numpy>=1.21.2 in c:\program files\python311\lib\site-packages (from opencv-python) (2.1.3)

[1]: pip install opencv-python-headless numpy

     Requirement already satisfied: opencv-python-headless in c:\users\asus\appdata\roaming\python\python311\site-packages (4.10.0.84)
     Requirement already satisfied: numpy in c:\program files\python311\lib\site-packages (2.0.2)
     Note: you may need to restart the kernel to use updated packages.
```

AFTER THAT THERE ARE SOME CHANGES YOU NEED TO DO IN YOUR CODE , BUT FIRST YOU NEED TO DOWNLOAD 3 FILES

coco.names.txt

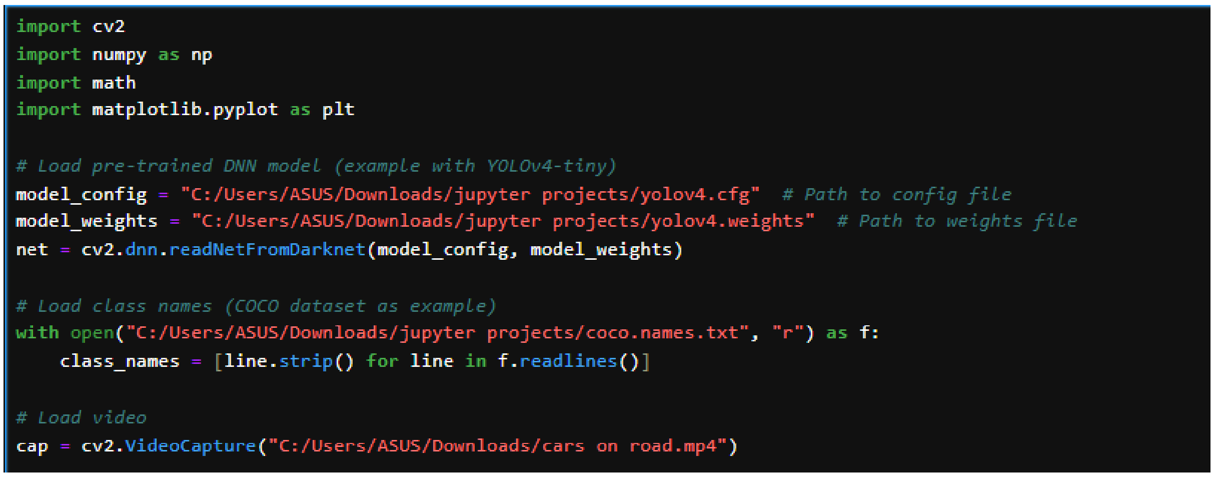
yolov4.cfg

AND THE NEXT IS OF YOLO WEIGHTS FILES - https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights

USE THIS GIVEN LINK AS THE FILE SIZE IS 245 MB

NOW CHANGES NEED TO BE DONE ON THE CODE -

AS YOU CAN SEE ON THE GIVEN PICTURE IN —


```python
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt

# Load pre-trained DNN model (example with YOLOv4-tiny)
model_config = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.cfg"  # Path to config file
model_weights = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.weights"  # Path to weights file
net = cv2.dnn.readNetFromDarknet(model_config, model_weights)

# Load class names (COCO dataset as example)
with open("C:/Users/ASUS/Downloads/jupyter projects/coco.names.txt", "r") as f:
    class_names = [line.strip() for line in f.readlines()]

# Load video
cap = cv2.VideoCapture("C:/Users/ASUS/Downloads/cars on road.mp4")
```

model_config = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.cfg" # Path to config file
model_weights = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.weights" # Path to weights file

with open("C:/Users/ASUS/Downloads/jupyter projects/coco.names.txt", "r") as f:

cap = cv2.VideoCapture("C:/Users/ASUS/Downloads/cars on road.mp4")

YOU NEED TO SET THE PATH OF THE DOWNLOADED FILES FROM YOUR SYSTEM THEN THE CODE WILL WORK. AND FOR THE 4TH LINE ABOUT VIDEO CAPTURE YOU HAVE TO USE A SAMPLE VIDEO FOR THAT CODE BUT IT SHOULD BE DIFFERENT FROM THE PREVIOUS ONE .

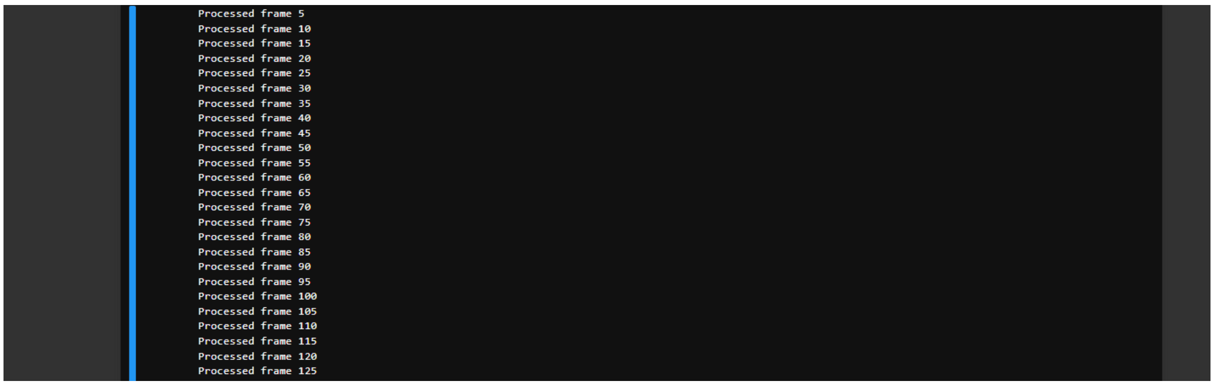
AFTER THE CODE IS EXECUTED YOU CAN SEE THAT THE CODE WILL CAPTURE THE OBJECTS FROM THE VIDEO AND IT WILL CONVERT THE VIDEO IN MULTIPLE PICTURES.

NOW USE THIS CODE -

```
import cv2
import numpy as np
import math

# Initialize video capture
input_video_path = "C:/Users/ASUS/Downloads/jupyter projects/Car Road Trans
output_video_path = "output.avi"

cap = cv2.VideoCapture(input_video_path)
if not cap.isOpened():
    print("Error: Unable to open video file.")
    exit()

frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
```

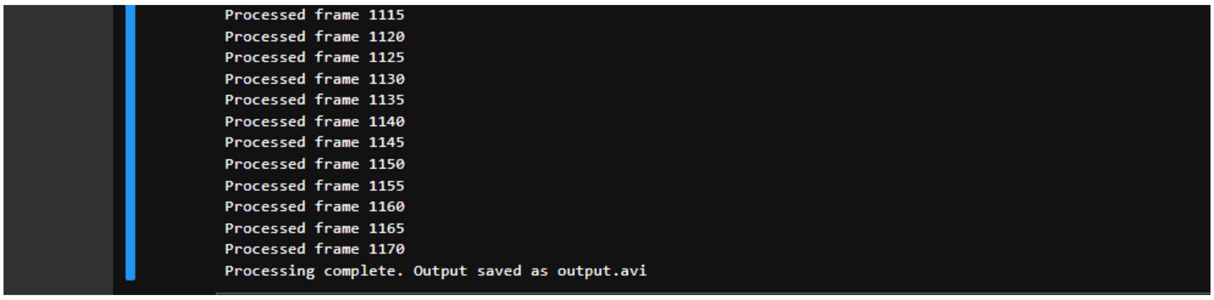
```
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(cv2.CAP_PROP_FPS))

# Initialize video writer
fourcc = cv2.VideoWriter_fourcc(*'XVID')
output_video = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_width

# Load YOLO model
model_cfg = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.cfg"
model_weights = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.weights"
labels_path = "C:/Users/ASUS/Downloads/jupyter projects/coco.names.txt"

net = cv2.dnn.readNetFromDarknet(model_cfg, model_weights)
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)

with open(labels_path, "r") as f:
    labels = f.read().strip().split("\n")

# Parameters for detection
conf_threshold = 0.6
nms_threshold = 0.3
blob_size = (320, 320)

# Object tracking variables
tracked_objects = {}
object_id = 0

# Function to calculate Euclidean distance
def euclidean_distance(p1, p2):
    return math.sqrt((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2)

# Function to perform object detection
def detect_objects(frame):
    blob = cv2.dnn.blobFromImage(frame, scalefactor=1 / 255.0, size=blob_si
    net.setInput(blob)
    layer_names = net.getUnconnectedOutLayersNames()
    layer_outputs = net.forward(layer_names)

    boxes = []
    confidences = []
    class_ids = []

    for output in layer_outputs:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]

            if confidence > conf_threshold:
                box = detection[0:4] * np.array([frame_width, frame_height,
                (center_x, center_y, width, height) = box.astype("int")

                x = int(center_x - (width / 2))
                y = int(center_y - (height / 2))

                boxes.append([x, y, int(width), int(height)])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_thre
    filtered_boxes = []
    filtered_class_ids = []
    filtered_confidences = []

    if len(indices) > 0:
        for i in indices.flatten():
            filtered_boxes.append(boxes[i])
            filtered_class_ids.append(class_ids[i])
            filtered_confidences.append(confidences[i])

    return filtered_boxes, filtered_class_ids, filtered_confidences

# Main loop
frame_count = 0
frame_skip = 5

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    frame_count += 1
    if frame_count % frame_skip != 0:
        continue

    boxes, class_ids, confidences = detect_objects(frame)

    for box, class_id, confidence in zip(boxes, class_ids, confidences):
        (x, y, w, h) = box
        center_x, center_y = x + w // 2, y + h // 2

        # Tracking logic
        min_distance = float("inf")
        matched_id = None

        for obj_id, (prev_x, prev_y) in tracked_objects.items():
            distance = euclidean_distance((center_x, center_y), (prev_x, pr
            if distance < min_distance and distance < 50:
                min_distance = distance
                matched_id = obj_id

        if matched_id is None:
            object_id += 1
            matched_id = object_id

        tracked_objects[matched_id] = (center_x, center_y)

        # Draw bounding box and label
        label = f"{labels[class_id]} {matched_id} ({confidence:.2f})"
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(frame, label, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.

    # Write processed frame
    output_video.write(frame)
    print(f"Processed frame {frame_count}")

cap.release()
output_video.release()
print("Processing complete. Output saved as output.avi")
```

THE OUTPUT AFTER EXECUTION OF CODE -



```
Processed frame 5
Processed frame 10
Processed frame 15
Processed frame 20
Processed frame 25
Processed frame 30
Processed frame 35
Processed frame 40
Processed frame 45
Processed frame 50
Processed frame 55
Processed frame 60
Processed frame 65
Processed frame 70
Processed frame 75
Processed frame 80
Processed frame 85
Processed frame 90
Processed frame 95
Processed frame 100
Processed frame 105
Processed frame 110
Processed frame 115
Processed frame 120
Processed frame 125
```

```
Processed frame 695
Processed frame 700
Processed frame 705
Processed frame 710
Processed frame 715
Processed frame 720
Processed frame 725
Processed frame 730
Processed frame 735
Processed frame 740
Processed frame 745
Processed frame 750
Processed frame 755
Processed frame 760
Processed frame 765
Processed frame 770
Processed frame 775
Processed frame 780
Processed frame 785
Processed frame 790
Processed frame 795
```



```
Processed frame 1115
Processed frame 1120
Processed frame 1125
Processed frame 1130
Processed frame 1135
Processed frame 1140
Processed frame 1145
Processed frame 1150
Processed frame 1155
Processed frame 1160
Processed frame 1165
Processed frame 1170
Processing complete. Output saved as output.avi
```