opencv installation can be done in 2 ways

- conda install -c conda-forge opencv
- pip install opencv-python

In [1]:
```
import cv2
import numpy as np
```

Load/Read image

In [2]:
```
# Load an image using 'imread' specifying the path to image
img = cv2.imread('image_examples/Modi.jpg',1)
```

Let's take a closer look at how images are stored

In [3]:
```
print(img)
```

```
[[[255 255 255]
  [255 255 255]
  [255 255 255]
  ...
  [255 255 255]
  [255 255 255]
  [255 255 255]]

 [[255 255 255]
  [255 255 255]
  [255 255 255]
  ...
  [255 255 255]
  [255 255 255]
  [255 255 255]]

 [[255 255 255]
  [255 255 255]
  [255 255 255]
  ...
  [255 255 255]
  [255 255 255]
  [255 255 255]]

 ...

 [[255 255 255]
  [255 255 255]
  [255 255 255]
  ...
  [255 255 255]
  [255 255 255]
  [255 255 255]]

 [[255 255 255]
  [255 255 255]
  [255 255 255]
  ...
  [255 255 255]
  [255 255 255]
  [255 255 255]]

 [[255 255 255]
  [255 255 255]
  [255 255 255]
```

```
...
[255 255 255]
[255 255 255]
[255 255 255]]]
```

Shape gives the dimensions of the image array

In [4]:
```
img.shape

#The 3D dimensions are 1358 pixels in height * 1500 pixels wide
#3 means that there are 3 components (RGB) that make up this image
```

Out[4]:
```
(1358, 1500, 3)
```

Display the image

In [5]:
```
# To display our image variable, we use 'imshow'
# The first parameter will be title shown on image window
# The second parameter is the image varialbe
cv2.imshow('PM', img)

# 'waitKey' allows us to input information when a image window is open
# By leaving it blank it just waits for anykey to be pressed before continuing.
# By placing numbers (except 0), we can specify a delay for
# how long you keep the window open (time is in milliseconds here)
cv2.waitKey(2000)

# This closes all open windows
# Failure to place this will cause your program to hang
cv2.destroyAllWindows()
```

In [6]:
```
img = cv2.imread('image_examples/Modi.jpg')
cv2.imshow('PM', img)
cv2.waitKey()
cv2.destroyAllWindows()
```

Save image

In [7]:
```
# Simply use 'imwrite' specificing the file name and the image to be saved
cv2.imwrite("pm_b_w.jpg", img)
```

Out[7]:
```
True
```

Resize Image

In [8]:
```
img = cv2.imread('image_examples/Modi.jpg')

resized_image = cv2.resize(img,(500,500))

gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)

cv2.imshow('Modi Image', gray)

cv2.waitKey()
cv2.destroyAllWindows()
```

In [9]:
```
img.shape[0]*0.5
```

Out[9]:
```
679.0
```

In [10]:
```
img.shape[1]*0.5
```

Out[10]:
```
750.0
```

Face Detection using HAAR Cascade Classifiers

In [11]:
```
# We point OpenCV's CascadeClassifier function to where our classifier (XML file format) is stored
face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

# Load our image then convert it to grayscale
image = cv2.imread('image_examples/Modi.jpg')
image = cv2.resize(img,(500,500))
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

### Tuning Cascade Classifiers - detectMultiScale(input image, **Scale Factor** , **Min Neighbors**)
faces = face_classifier.detectMultiScale(gray, 1.05, 5)
```

```
# Scale Factor - Specifies how much we reduce the image size each time we scale.
# E.g. in face detection we typically use 1.3. This means we reduce the image by 30% each time it's s
# Smaller values, like 1.05 will take longer to compute, but will increase the rate of detection.

## Min Neighbors**
# Specifies the number of neighbors each potential window should have in order to consider it a positive detection.
# Typically set between 3-6.
# It acts as sensitivity setting, low values will sometimes detect multiples faces over a single face.
# High values will ensure less false positives, but you may miss some faces.
```

In [12]:
```
print(faces)
```

```
[[205  79 217 217]]
```

In [13]:
```
# We point OpenCV's CascadeClassifier function to where our classifier (XML file format) is stored
face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

# Load our image then convert it to grayscale
image = cv2.imread('image_examples/Modi.jpg')
image = cv2.resize(image,(500,500))
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

### Tuning Cascade Classifiers - detectMultiScale(input image, **Scale Factor** , **Min Neighbors**)
faces = face_classifier.detectMultiScale(gray, 1.05, 5)

# Scale Factor - Specifies how much we reduce the image size each time we scale.
# E.g. in face detection we typically use 1.3. This means we reduce the image by 30% each time it's scaled.
# Smaller values, like 1.05 will take longer to compute, but will increase the rate of detection.

## Min Neighbors**
# Specifies the number of neighbors each potential window should have in order to consider it a positive detection.
# Typically set between 3-6.
# It acts as sensitivity setting, low values will sometimes detect multiples faces over a single face.
# High values will ensure less false positives, but you may miss some faces.

# When no faces detected, face_classifier returns and empty tuple
if faces is ():
    print("No faces found")

# We iterate through our faces array and draw a rectangle
# over each face in faces
for (x,y,w,h) in faces:
    cv2.rectangle(image, (x,y), (x+w,y+h), (0,255,100), 1)

cv2.imshow('Face Detection', image)
```

```
cv2.waitKey()
cv2.destroyAllWindows()
```

```
<>:23: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:23: SyntaxWarning: "is" with a literal. Did you mean "=="?
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_13632\2325480536.py:23: SyntaxWarning: "is" with a literal. Did you mean "=
="?
  if faces is ():
```

Face & Eye Detection using HAAR Cascade Classifiers in Image

In [14]:

```
face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')
eye_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_eye.xml')

img = cv2.imread('image_examples/modi.jpg')
resized_image = cv2.resize(img,(500,500))
gray = cv2.cvtColor(resized_image, cv2.COLOR_BGR2GRAY)

faces = face_classifier.detectMultiScale(gray, 1.3, 5)

# When no faces detected, face_classifier returns and empty tuple
if faces is ():
    print("No Face Found")

for (x,y,w,h) in faces:
    cv2.rectangle(resized_image,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = resized_image[y:y+h, x:x+w]
    eyes = eye_classifier.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',resized_image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

```
<>:11: SyntaxWarning: "is" with a literal. Did you mean "=="?
<>:11: SyntaxWarning: "is" with a literal. Did you mean "=="?
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_13632\4008476180.py:11: SyntaxWarning: "is" with a literal. Did you mean "=
="?
  if faces is ():
```

Capture a video

In [15]:
```
# Doing some Face Recognition with the webcam

import cv2
video = cv2.VideoCapture(0) #0--webcam

while True:
    check, frame = video.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.imshow('Video',gray)
    if cv2.waitKey(1) == ord('q'):
        break

video.release()
cv2.destroyAllWindows()
```

Face & Eye Detection using HAAR Cascade Classifiers

In [16]:
```
# Defining a function that will do the detections
face_cascade = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')
eye_cascade  = cv2.CascadeClassifier('Haarcascades/haarcascade_eye.xml')

def detect(gray, frame):
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_color = frame[y:y+h, x:x+w]
        eyes = eye_cascade.detectMultiScale(roi_gray, 1.1, 3)
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
    return frame

# Doing some Face Recognition with the webcam
video = cv2.VideoCapture(0)

while True:
    check, frame = video.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    canvas = detect(gray, frame)
```

```
    cv2.imshow('Video', canvas)
    if cv2.waitKey(1)==ord('q'):
        break

video.release()
cv2.destroyAllWindows()
```

Pedistrian Detection

In [17]:
```
import cv2

# Create our body classifier
body_classifier = cv2.CascadeClassifier('Haarcascades\haarcascade_fullbody.xml')

# Initiate video capture for video file
cap = cv2.VideoCapture('image_examples/walking.avi')

# Loop once video is successfully loaded
while cap.isOpened():

    # Read first frame
    check, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Pass frame to our body classifier
    bodies = body_classifier.detectMultiScale(gray, 1.2, 3)

    # Extract bounding boxes for any bodies identified
    for (x,y,w,h) in bodies:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 255), 2)
        cv2.imshow('Pedestrians', frame)

    if cv2.waitKey(1)==ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

In [18]:
```
import time

# Create our body classifier
car_classifier = cv2.CascadeClassifier('Haarcascades\haarcascade_car.xml')
```

```
# Initiate video capture for video file
cap = cv2.VideoCapture('image_examples/cars.avi')


# Loop once video is successfully loaded
while cap.isOpened():

    time.sleep(.05)
    # Read first frame
    check, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Pass frame to our car classifier
    cars = car_classifier.detectMultiScale(gray, 1.4, 2)

    # Extract bounding boxes for any bodies identified
    for (x,y,w,h) in cars:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 255), 2)
        cv2.imshow('Cars', frame)

    if cv2.waitKey(1)==ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```