Sure, Pratik! Let's go through both of your Python files **line by line**, in simple terms. The overall purpose of your project is to **control system volume using hand gestures** with OpenCV, MediaPipe, and Pycaw.

**📄 File 1: HandTrackingModule.py**

This is a custom module to detect and track hands using MediaPipe.

**1. Imports**

import cv2

import mediapipe as mp

import time

import math

* cv2: OpenCV, for video and image processing.
* mediapipe: Google's library for face, hand, and pose detection.
* time: For measuring FPS or delays.
* math: For distance calculations.

**2. Class Definition**

class handDetector():

* You define a class to wrap all hand-tracking functions.

**3. Initialization**

def \_\_init\_\_(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):

* mode: Whether the input is a static image.
* maxHands: Max number of hands to detect.
* detectionCon: Minimum confidence for initial detection.
* trackCon: Minimum confidence for tracking the hand.

self.mpHands = mp.solutions.hands

self.hands = self.mpHands.Hands(...)

self.mpDraw = mp.solutions.drawing\_utils

self.tipIds = [4, 8, 12, 16, 20]

* Initializes MediaPipe hand detector and a drawing utility.
* tipIds: IDs of fingertips (thumb, index, middle, ring, pinky).

**4. Find Hands**

def findHands(self, img, draw=True):

* Converts BGR to RGB (MediaPipe needs RGB).
* Processes the image to detect hands.
* Optionally draws landmarks (dots and connections).

**5. Find Position**

def findPosition(self, img, handNo=0, draw=True):

* Extracts landmark positions from one hand (handNo=0 = first hand).
* Converts normalized landmark coordinates (0–1) to pixel positions.
* Stores them in self.lmList as [id, x, y].
* Calculates bounding box around hand.
* Optionally draws a rectangle around the hand.

**6. Which Fingers Are Up**

def fingersUp(self):

* Checks if thumb and other fingers are up (based on landmark positions).
* Returns a list of 0s and 1s (0 = down, 1 = up).

**7. Distance Between Two Points**

def findDistance(self, p1, p2, img, draw=True):

* Calculates Euclidean distance between two landmarks.
* Draws a line and circles between them.
* Returns distance and visual info.

**📄 File 2: Main Application (Volume Control)**

This file uses the handDetector class to control system volume with gestures.

**1. Imports**

import cv2, time, numpy as np, math

import HandTrackingModule as htm

from ctypes import cast, POINTER

from comtypes import CLSCTX\_ALL

from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume

* Adds Pycaw (Python Core Audio Windows) to control system volume.

**2. Config and Initialization**

FRAME\_WIDTH, FRAME\_HEIGHT = 640, 480

cap = cv2.VideoCapture(0)

cap.set(3, FRAME\_WIDTH)

cap.set(4, FRAME\_HEIGHT)

* Sets up the webcam with resolution 640x480.

**3. Create Hand Detector**

detector = htm.handDetector(detectionCon=0.7)

* Uses your custom module with higher detection confidence.

**4. Setup Volume Control**

devices = AudioUtilities.GetSpeakers()

interface = devices.Activate(...)

volumeControl = cast(interface, POINTER(IAudioEndpointVolume))

* Initializes system volume control using Pycaw.

**5. Main Loop**

while True:

Runs the application in real time.

**6. Read Frame and Detect Hands**

success, img = cap.read()

img = detector.findHands(img)

lmList, \_ = detector.findPosition(img, draw=False)

* Captures webcam frame.
* Finds hand landmarks (positions of fingers).

**7. Gesture Detection (Thumb + Index Finger Distance)**

if len(lmList) >= 9:

x1, y1 = lmList[4][1], lmList[4][2] # Thumb tip

x2, y2 = lmList[8][1], lmList[8][2] # Index tip

* Gets the positions of thumb and index fingertip.
* Calculates the distance between them.

**8. Map Distance to Volume**

length = math.hypot(x2 - x1, y2 - y1)

volScalar = np.interp(length, [50, 250], [0.0, 1.0])

volumeControl.SetMasterVolumeLevelScalar(volScalar, None)

* Maps the hand distance to volume range (0.0 to 1.0).
* Sets the system volume accordingly.

**9. Visual Feedback**

volBar = np.interp(volScalar, [0.0, 1.0], [400, 150])

volPercent = smoothness \* round(targetVolPercent / smoothness)

* Displays a visual volume bar and smooth percentage.

**10. Show FPS and Display**

fps = 1 / (currTime - prevTime + 1e-6)

cv2.putText(img, f'FPS: {int(fps)}', ...)

cv2.imshow("Volume Control", img)

* Shows current frame rate (FPS) and live webcam view.

**11. Exit Condition**

if cv2.waitKey(1) & 0xFF == ord('q'):

break

* Press q to quit the app.

**✅ Summary:**

* **File 1 (HandTrackingModule.py)**: Contains the reusable logic for detecting hands and extracting finger positions/distances.
* **File 2 (main.py)**: Uses this logic to control volume by measuring the distance between thumb and index fingers.

Let me know if you want:

* Comments added directly into the code.
* To modify it for gestures like mute, pause, etc.
* To turn this into a full app with a GUI.