Video and frame capture:

- cap = cv2.VideoCapture(0)
    - Starts the camera, then returns the basic camera or "video capture object" to read frames.
    - 0 - That's the index of the camera it is used to select different cameras if you have more than one attached. By default 0 is your main one.
- ret, frame = cap.read()`
    - ret: it's a boolean, it's True if a frame was successfully read.
    - frame: the actual frame or image (numpy array) from the camera

Hand detection

- mp_hands = mp.solutions.hands contains MediaPipe's hand tracking model.
- hands = mp_hands.Hands(...) initializes the hand tracker
    - static_image_mode = False, it will just run continuously without static images - good for videos
    - max_num_hands = 1: just tracks one hand
    - min_detection_confidence: accuracy threshold for detecting new hands
    - min_tracking_confidence: accuracy for continuing to track the hand
- results = hands.process(image_rgb) runs hand detection on the input image
- results.multi_hand_landmarks - A list of hand landmarks (if any were detected)
    - remember: each hand (its item) has 21 landmarks with x, y, and z coordinates (normalized between 0 and 1)

Image Processing:

- image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    - <u>OpenCV reads images in BGR format BUT Mediapipe needs RGB!!!!!</u>
- frame = cv2.flip(frame, 1)
    - mirrors the camera horizontally, makes it less annoying

Drawing:

- mp_drawing.draw_landmarks(...) draws the hand skeleton on the frame
- cv2.imshow("Window", frame) shows the frame//image
- cv2.waitKey(1) & 0xFF makes it wait 1 millisecond for a key press
    - '& 0xFF' makes the output more flexible so that it is good for all systems
    - <u>RETURNS THE ASCII CODE NOT THE ACTUAL NUMBER!!</u> of the key pressed
- ord('1') coverts the character '1' to ASCII code ('49')
- gesture_label = key - ord('0') converts the ascii code in "key" to the actual number