Open IDE. (install Python 3.3+)
Download Packages:

- mediapipie
- opencv-python

Import the packages. Also, import NumPy to use NumPy arrays.

Opencv python has certain GUI features where we can access our computer web camera. To do this we need to create a VideoCapture object and give it an argument of 0 so that the program access the default camera option.

Full code to access the camera:

```
# To capture a video we need a create a VideoCapture object
capture = cv.VideoCapture(0)

# check if camera opened successfully or not
if not capture.isOpened():
    print('Cannot open Camera')
    exit()

while True:
    # capture frame by frame

    ret, frame=capture.read()
    # if frame is read correctly ret is True
    if not ret:
        print('cannot receive frame')
        break
    # convert image to fram to GreyScale
    grayScale=cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    # show the frame
    cv.imshow('frame', grayScale)
    cv.waitKey(0)

    # to open camera and keep it open:
    # success, img = capture.read()
    # imgRGB = cv.cvtColor(img, cv.COLOR_BGR2RGB) #convert to RGB
    # cv.imshow("Image", img)
    # cv.waitKey(1)


# When everything done, release the capture
capture.release()
cv.destroyAllWindows()
```

MediaPipe Pose is a machine learning solution for high-fidelity body pose tracking, inferring 33 3D landmarks on the whole body from RGB video frames.

We need to use the medipipe framework of pose to detect poses and also use drawing utilities to draw or connect detected landmarks on various poses

```
# use mediapipe framework of pose to detect pose
mpPose = mp.solutions.pose
```

```
# import drawing utilities
mpDrawing = mp.solutions.drawing_utils
```

.Pose() is what uses the detected processed images (converted to RGB) to produce results.

What are the results?

When we print(results) this is what we get:

```
<class 'mediapipe.python.solution_base.SolutionOutputs'>
<class 'mediapipe.python.solution_base.SolutionOutputs'>
```

It is just a class and it shows that it is detecting the image pose from the web camera.
We know that there should be 33 landmarks detected by the pose solution of mediapipe. How can we see the coordinates and the visibility of the landmarks?

```
print(results.pose_landmarks)
```

Printing this on our IDE will show the x,y, and z coordinate along with the visibility of every landmark of the pose.
For example:

```
landmark {
  x: 0.5749664306640625
  y: 3.1209166049957275
  z: 0.5941792726516724
  visibility: 0.00038935799966566265
}
```

Our task is to extract the coordinates of the necessary landmarks.

```
to extract all landmarks:
    try:
        landmarks = results.pose_landmarks.landmark
        print(landmarks)
^^the print will show us the cordinate of each landmark on our body and their connections

to find the index position of the landmarks:
    print(mpPOSE.PoseLandmark.<name_of_landmark>.value)
^^this will print the landmark index position

to extract a particular landmark:
    landmarks[mpPOSE.PoseLandmark.<name_of_landmark>]
    example: landmarks[mpPOSE.PoseLandmark.LEFT_SHOULDER]

^^this will give the cordinates of the landmark
    to check a particular cordinate:
    landmarks[mpPOSE.PoseLandmark.<name_of_landmark>.value].x/y/z
```

Now that we understood the above ways to extract landmarks we can do this for our required landmarks.

1. Left_hip
2. Left_shoulder
3. left_elbow

After we know the x and z coordinate of the extracted landmark our task is to compute the required angle.

```
def calculate_angle(a, b, c):
    a = np.array(a)  # First
    b = np.array(b)  # Mid
    c = np.array(c)  # End

    radians = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1], a[0] - b[0])
    angle = np.abs(radians * 180.0 / np.pi)

    if angle > 180.0:
        angle = 360 - angle

    return angle
```

We created the above function to compute the angle between the 3 landmarks (angle between left_hip to left_shoulder and left_shoulder to left_elbow).
^^Trigonometry needed.

Now that we know the angle it becomes easy for the program to predict the hand-state.

```
    if angle_left >= 90:
        hand_state='you ordered:'+text
    if angle_left < 20:
        hand_state = 'Please raise hand so that we can find you'
    if angle_left >= 20 and angle_left < 90 and hand_state == 'Please raise hand so that we can find you':
        hand_state = "your left hand is moving up"
    if angle_left >= 20 and angle_left < 90 and hand_state == 'you ordered:'+text:
        hand_state = "We found you. Thanks!"

except:
```

The above code is easy to understand (some if conditions)

The last step is to edit our GUI in a way to makes it representable. We use the cv.putText function to do so.

```
# GUI VISUALS and OPERATIONS
cv.rectangle(image, (0, 0), (700, 100), (245, 117, 16), -1)
cv.putText(image, 'Virtual waiter: ', (15, 25),
           cv.FONT_HERSHEY_SIMPLEX, 1, (0, 150, 200), 2, cv.LINE_AA)
cv.putText(image, hand_state,
           (60, 60),
           cv.FONT_HERSHEY_SIMPLEX, 0.75, (0, 150, 200), 2, cv.LINE_AA)
mpDrawing.draw_landmarks(image, results.pose_landmarks, mpPose.POSE_CONNECTIONS)

cv.imshow("POSE_GUESS", image)
if cv.waitKey(10) & 0xFF == ord('q'):
    break
```

Things I learned:

1) Extracting specific landmarks
2) How mediapipe solution for pose works
3) Computing angle between 2 lines (3 landmarks)
4) GUI Visuals and Operations
5) Use openCV GUI features

Resources:

1. Mediapipe: https://google.github.io/mediapipe/solutions/pose.html
2. Open CV GUI features https://docs.opencv.org/4.5.2/dc/d4d/tutorial_py_table_of_contents_gui.html

The things I learned were based on these particular projects only. I may not know all the OpenCV features like:

- Core operations
- Image processing
- Open CV Python Bindings

However, I learned and practiced all the required syntaxes.