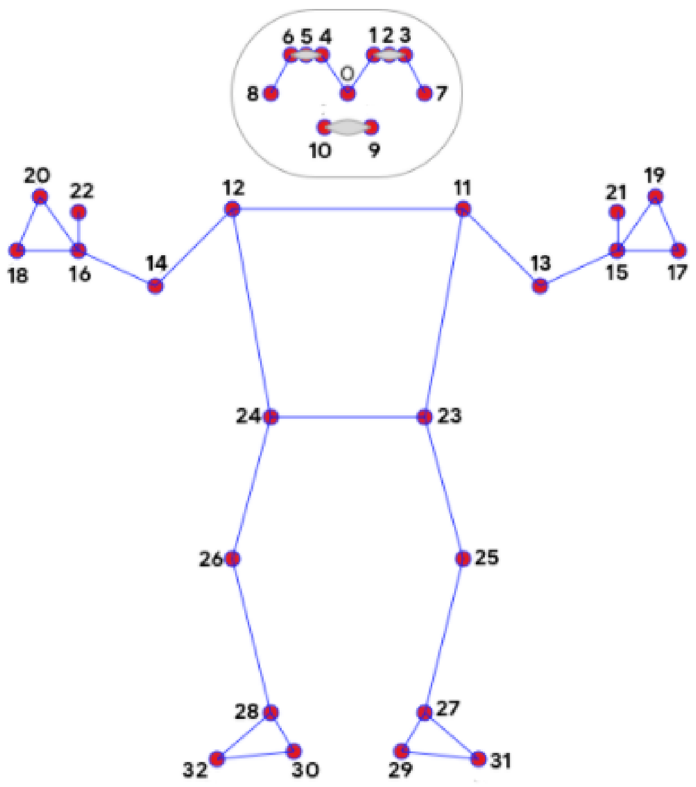
In [5]:

```
import cv2
import mediapipe as mp
```

In [6]:

```
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose
pose = mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5)
```



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

In [12]:

```
cap = cv2.VideoCapture('test_video.mp4')
while cap.isOpened():
    # read frame
    _, frame = cap.read()
    try:
        # resize the frame for portrait video
        frame = cv2.resize(frame, (350, 600))
        # convert to RGB
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # process the frame for pose detection
        pose_results = pose.process(frame_rgb)

        # print(pose_results.pose_landmarks)

        # draw skeleton on the frame
        mp_drawing.draw_landmarks(frame, pose_results.pose_landmarks, mp_pose.POSE_CONNECTI
        # display the frame
        cv2.imshow('Output', frame)
    except:
        break

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

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

In [10]:

```
# get landmark for a specific point
pose_results.pose_landmarks.landmark[2]
```

Out[10]:

```
x: 0.3880370259284973
y: 0.2901993691921234
z: -0.20075245201587677
visibility: 0.9999621510505676
```

In [ ]:

```
```

In [11]:

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
    # read frame
    _, frame = cap.read()
    try:
        # resize the frame for portrait video
        # frame = cv2.resize(frame, (350, 600))
        # convert to RGB
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # process the frame for pose detection
        pose_results = pose.process(frame_rgb)
        # print(pose_results.pose_landmarks)

        # draw skeleton on the frame
        mp_drawing.draw_landmarks(frame, pose_results.pose_landmarks, mp_pose.POSE_CONNECTI
        # display the frame
        cv2.imshow('Output', frame)
    except:
        break

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

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

In [ ]:

In [ ]:

In [ ]:

In [ ]: