# Pose Recognition Using Python

## 1] Introduction

**Pose recognition:** Pose recognition refers to the process of detecting key body landmarks, such as joints (e.g., shoulders, elbows, wrists, hips, knees, and ankles), and understanding the pose or position of a person in an image. Pose recognition involves identifying and analysing these key points to infer body posture (e.g., standing, sitting, walking). It is a key component of computer vision applications.

**Image Pose Recognition:** The focus of this project is on applying pose recognition to static images. In contrast to dynamic pose estimation, which works with video streams or live webcam feeds, static pose recognition involves detecting key body landmarks from a single frame or still image. The primary challenge lies in accurately detecting the landmarks despite varying body orientations, clothing, and environmental conditions.

**Landmark:** Landmarks are specific, predefined points on the human body such as the shoulders, elbows, wrists, hips, knees, ankles, and others. They help in understanding the structure and orientation of the body by identifying the spatial positions of these critical points.

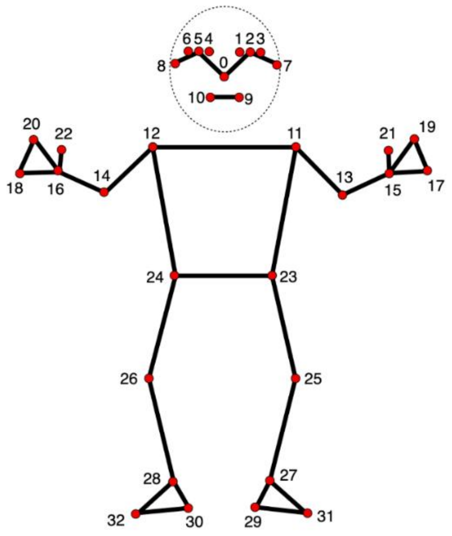


Figure: landmarks

## 2] Objective

The goal of this project is to implement pose recognition using Python of static images and pose estimation. The system should be able to detect 33 key body landmarks from an image and visualize the pose skeleton by connecting these landmarks. The system should handle varying body orientations and environments.

## 3] Technologies and Libraries Used

1. **Python:** Python is chosen for this project due to its simplicity and extensive libraries for image processing and computer vision. Its ease of use and flexibility make it ideal for rapid development and experimentation.

2. **OpenCV:** OpenCV (Open-Source Computer Vision Library) is used for image processing tasks such as:
    a. Loading, displaying, and saving images.
    b. Pre-processing the input image (e.g., converting to the RGB colour space) for compatibility with pose detection models.

3. **MediaPipe:** MediaPipe, developed by Google, provides a pre-trained pose estimation model that can detect 33 body landmarks in images. The model is robust and works well under various conditions, making it suitable for static image pose recognition. The library can extract both 2D and 3D key point coordinates.

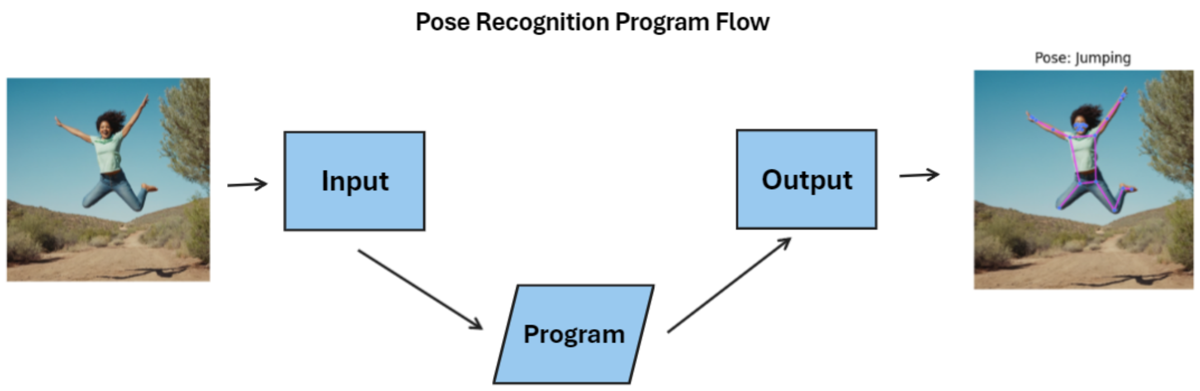
4. **NumPy:** NumPy is used for numerical operations, particularly for manipulating arrays and matrices. It plays a crucial role when working with image data and performing operations like matrix transformations and coordinate manipulation.

5. **Matplotlib:** Matplotlib is a plotting library used to display images with overlaid pose skeletons. It is particularly useful for visualizing results in an easy-to-understand format. Optionally, it can be used to save processed images with landmarks to output files.

## 4] How Pose Recognition Works

**Flow of Pose Recognition:**

1. Image Input: The system accepts an input image, which can be selected or uploaded by the user.
2. Pose Detection: The pre-trained MediaPipe Pose model detects the 33 body landmarks, such as the head, neck, shoulders, elbows, wrists, hips, knees, and ankles.
3. Landmark Extraction: The system extracts the key points as 2D coordinates from the image.
4. Skeleton Display: The landmarks are connected to form a skeleton, representing the body structure.
5. Output: The system displays or saves the image with the pose skeleton superimposed.



**Key Details:**

1. 2D Pose Estimation: MediaPipe's Pose model performs 2D pose estimation, returning the x and y coordinates for each body landmark.
2. Pose Skeleton: The detected landmarks are connected by lines, forming a skeletal structure that represents the pose.
3. Challenges: Static images may have issues with occlusion, poor lighting, or background clutter, which could affect landmark detection. Furthermore, the pose skeleton must be accurately overlaid on the image to reflect the true body structure.

## 5] Implementation Steps

**Step 1:** Set up the Directory structure

```
Pose-Recognition-Using-Python/
├── images/
│   ├── image-1.png
│   ├── image-2.png
│   └── ...
├── main.py
├── main.ipynb
├── pose_recognize_and_estimate.py
```

**images/:** Directory containing input images.
**main.py:** Main script to run the pose recognition.
**main.ipynb:** Can run main.py file from this file.
**pose_recognize_and_estimate.py:** Contains functions for pose detection and visualization.

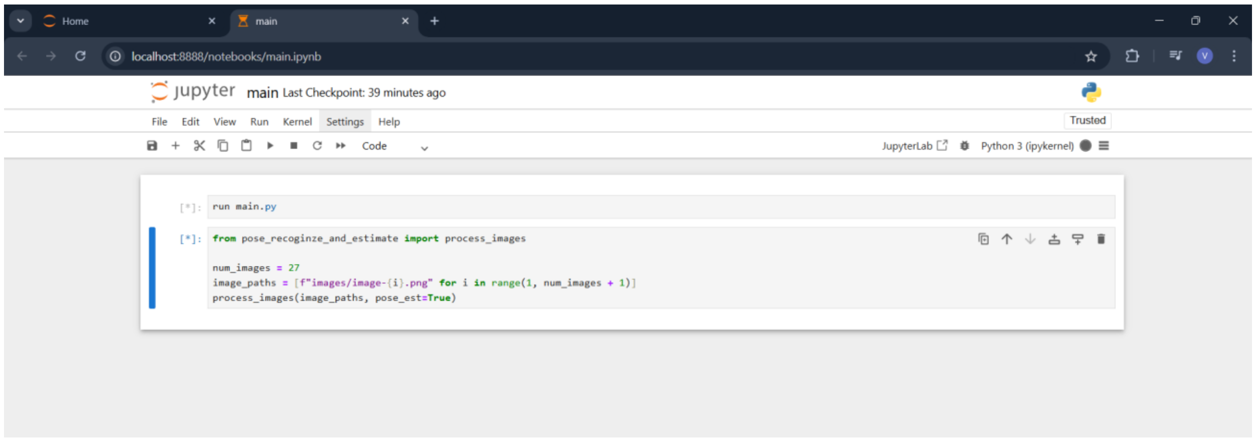
**Step 2:** Install Required Dependencies

```
pip install opencv-python mediapipe numpy matplotlib
```

**Step 3:** Coding
Create pose_recognize_and_estimate.py and main.py file
Source code: https://github.com/shivamvishwakarma2k2/Pose-Recognition-Using-Python

## 6] Results

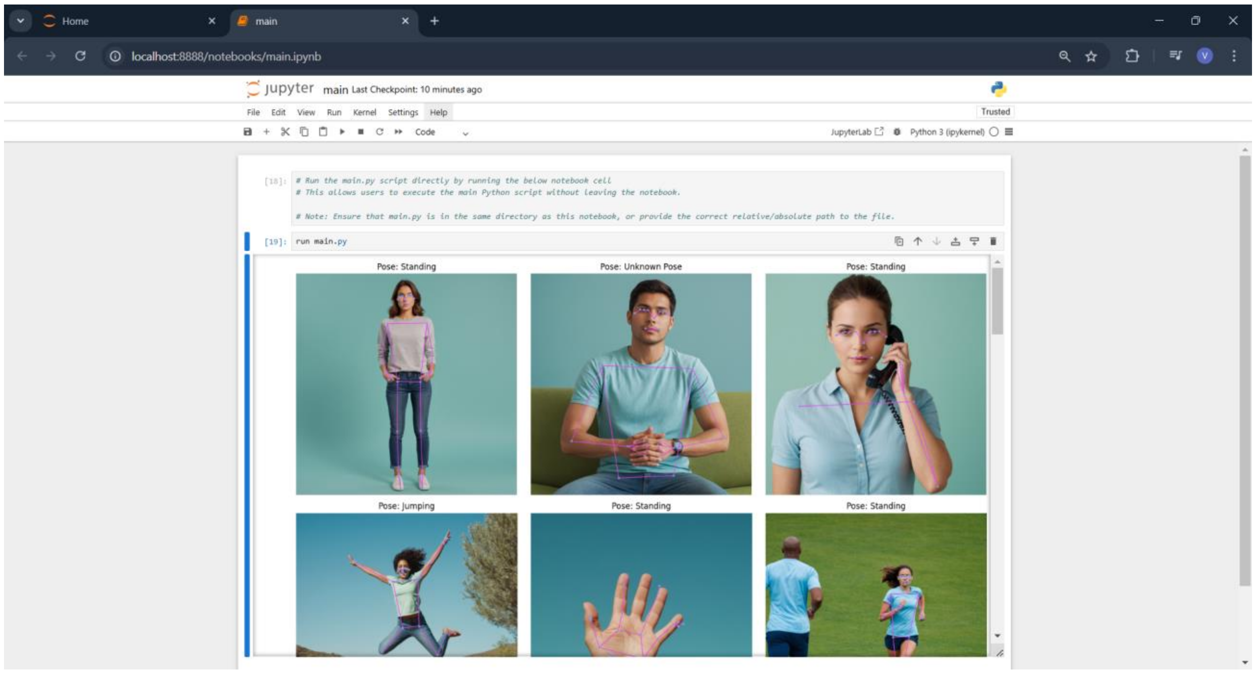
The final output images will show the detected 33 body landmarks connected by lines, forming a recognizable pose skeleton. Below is an example of what the processed image might look like:


Home
main
localhost:8888/notebooks/main.ipynb
jupyter main Last Checkpoint: 39 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[*]: run main.py
```

```
[*]: from pose_recoginze_and_estimate import process_images

     num_images = 27
     image_paths = [f"images/image-{i}.png" for i in range(1, num_images + 1)]
     process_images(image_paths, pose_est=True)
```

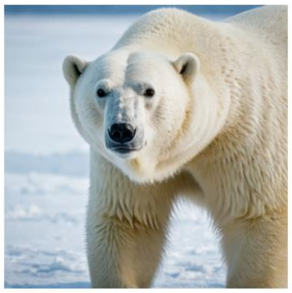


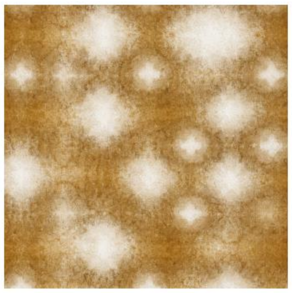
```
[18]: # Run the main.py script directly by running the below notebook cell
      # This allows users to execute the main Python script without leaving the notebook.

      # Note: Ensure that main.py is in the same directory as this notebook, or provide the correct relative/absolute path to the file.
```

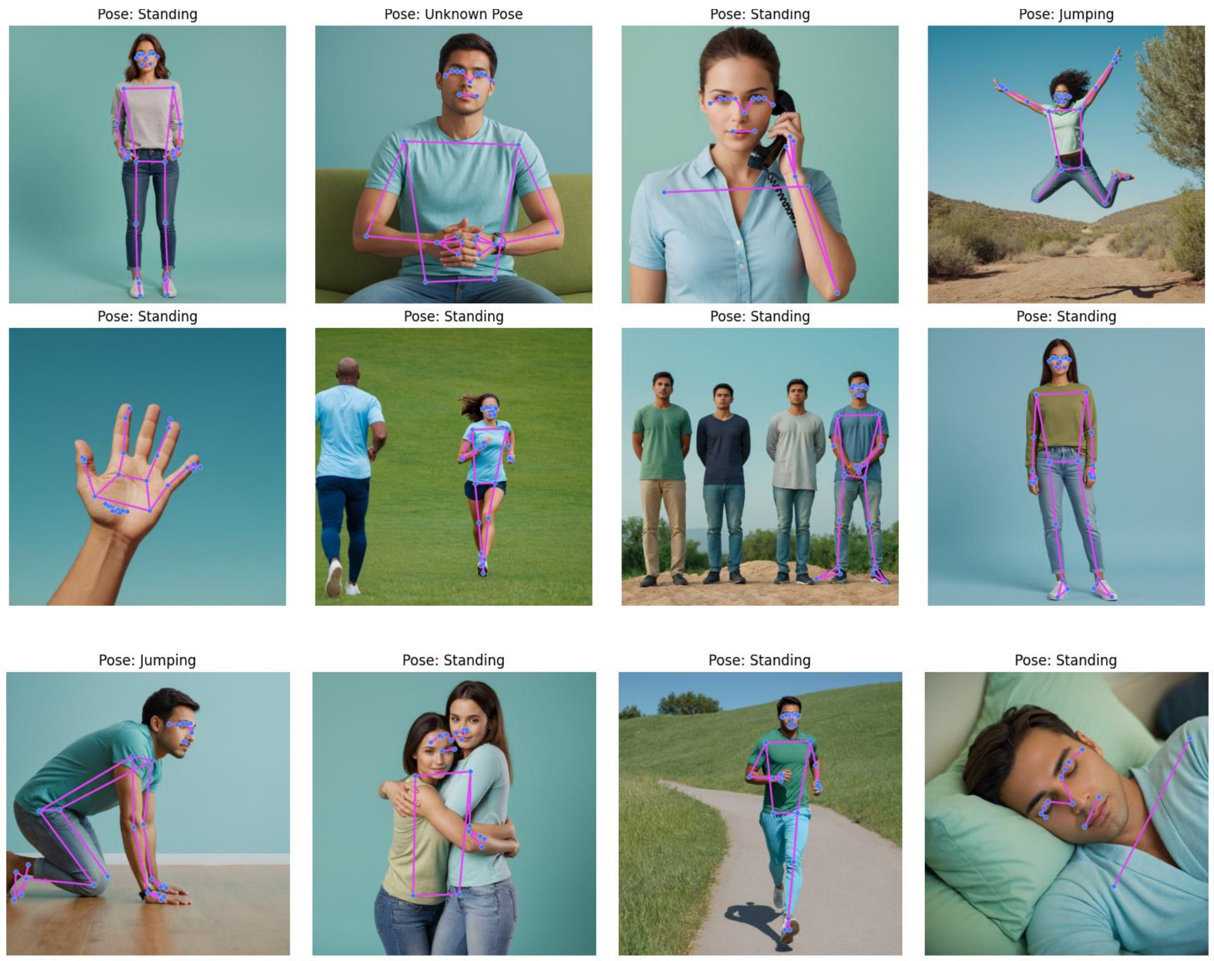
```
[19]: run main.py
```

Pose Estimation = False





Pose Estimation = True



## 7] Conclusion

This project successfully implemented a pose recognition system using Python, OpenCV, and MediaPipe for static images, pose estimation up to some level which can be more effective as bringing accuracy in calculation logic. The system detects and visualizes 33 body landmarks, forming a pose skeleton that represents the body structure.

## 8] References

- MediaPipe Pose Documentation: https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker
- OpenCV Documentation: https://opencv.org/
- OpenCV Python Tutorials: https://docs.opencv.org/master/d6/d00/tutorial_py_root.html
- GeeksforGeeks Pose Recognition Tutorial: https://www.geeksforgeeks.org/