Basic things to understand before running a code:

Keypoints:

- **Holistic**: Use for applications requiring comprehensive body tracking, including face, hands, and body pose.
- **Drawing Utils**: Use for visualizing detected landmarks on images or video frames.
- **Drawing Styles**: Use for applying consistent visual styles to the drawn landmarks and connections.
- **Face Mesh**: Use for applications focused on facial features and expressions.
- **Hands**: Use for hand gesture recognition and sign language translation.
- **Pose**: Use for full-body motion tracking and analysis.

- **min_detection_confidence** is applied during the initial detection phase, where the goal is to identify objects or landmarks in the input data.

- **min_tracking_confidence** is applied during the subsequent tracking phase, where the goal is to maintain continuity and reliability in tracking objects or landmarks over time.

- **fourcc**=Four character code , it is used by codec.

- **VideoWriter** is to specify the codec that is a video file

Modules that we can use in this mediapipe library

- **mp.solutions.holistic**: This refers to the holistic model provided by Mediapipe.

  MediaPipe Holistic consists of a pipeline of separate models for pose, face and hands,

- **mp.solutions.drawing_utils***:* This module contains utility functions to draw landmarks and connections on images, making it easier to visualize the results of the Mediapipe models.
- **mp.solutions.drawing_styles**: This module provides predefined styles for drawing landmarks and connections. It helps standardize the appearance of the visualizations produced by the Mediapipe models.
- **mp.solutions.face_mesh**: This is the face mesh model provided by Mediapipe. It detects and tracks facial landmarks with high precision, which can be used for applications like facial recognition

- **mp.solutions.pose:** This is the pose detection model from Mediapipe. It detects and tracks full-body landmarks,

Function initailzation parameters explanation:

Syntax : face_mesh = mp_face_mesh.FaceMesh(static_image_mode=False, max_num_faces=1, min_detection_confidence=0.5, min_tracking_confidence=0.5)

Note: static_image_mode This parameter indicates whether the face mesh detection is performed on static images (True) or video frames (False).

Why we need to convert the BGR to RGB of all images /videos:

Syntax : image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

Note: In OpenCV, images are often represented in the BGR (Blue, Green, Red) color space by default. However, many other libraries and applications, including MediaPipe, typically use the RGB (Red, Green, Blue) color space. So, if you're processing images using both OpenCV and MediaPipe, you might need to convert the images from BGR to RGB to ensure compatibility.

Drawing landmarks function explanation:

Syntax:
```
if face_results.multi_face_landmarks:
        for  face_landmarks in face_results.multi_face_landmarks:
            mp_drawing.draw_landmarks(
                        image=image,
                        landmark_list=face_landmarks,
                        connections=mp_face_mesh.FACEMESH_TESSELATION,
                        landmark_drawing_spec=None,

            connection_drawing_spec=mp_drawing_styles.get_default_face_mesh_tesselation_style())
```

- for face_landmarks in face_results.multi_face_landmarks:: This iterates over each detected face landmark. face_results.multi_face_landmarks likely contains a list of face landmarks detected in the image.
- mp_drawing.draw_landmarks: This function is used to draw landmarks on the image. It's part of the MediaPipe library.

Parameters used here are:

- **image=image:** This specifies the image on which the landmarks will be drawn.
- **landmark_list=face_landmarks:** This specifies the list of landmarks for the current face being processed.

- **connections=mp_face_mesh.FACEMESH_TESSELATION:** This parameter specifies which connections to draw between the landmarks. FACEMESH_TESSELATION likely refers to a predefined set of connections for drawing the face mesh.
- **landmark_drawing_spec=None**: This parameter can be used to specify the appearance of the landmarks (e.g., color, thickness). In this case, it seems to be set to None, which means that the default drawing specifications will be used

Summery of code explanation:

This script demonstrates how to extract and draw facial, hand, and body landmarks from a video file using MediaPipe's holistic modules and save the output with the landmarks drawn. Here's a step-by-step summary of the process:

1. **Import Libraries**:
    - Import cv2 for video handling and mediapipe for landmark detection.
    - Import os for path management.
2. **Initialize MediaPipe Solutions**:
    - Initialize MediaPipe solutions for holistic, drawing utilities, face mesh, hands, and pose tracking.
3. **Capture Video**:
    - Open the video file using cv2.VideoCapture.
    - Check if the video file is opened successfully.
    - Retrieve the video's frames per second (FPS), width, and height.
4. **Create Video Writer Object**:
    - Define the codec and create a cv2.VideoWriter object to save the output video with landmarks.
5. **Initialize MediaPipe Modules**:
    - Initialize face mesh, hands, and pose modules with the specified detection and tracking confidence levels.
6. **Process Video Frames**:
    - Read each frame from the video.
    - Convert the frame from BGR to RGB (MediaPipe processes images in RGB format).
    - Process the frame to detect face, hands, and pose landmarks.
    - Draw the detected landmarks on the frame:
        - **Face Landmarks**: Use mp_drawing.draw_landmarks with face mesh tessellation.
        - **Hand Landmarks**: Use mp_drawing.draw_landmarks with hand connections.
        - **Pose Landmarks**: Use mp_drawing.draw_landmarks with pose connections.
    - Write the frame with drawn landmarks to the output video.
7. **Display and Save Output**:
    - Display the frame with landmarks in a window using cv2.imshow.
    - Break the loop if 'q' key is pressed.
8. **Release Resources**:
    - Release the video capture and writer objects.
    - Close all OpenCV windows.