Installing Libraries

In [1]:
```
pip install  mediapipe opencv-python
```

```
Requirement already satisfied: mediapipe in c:\users\nidhi\anaconda3\lib\site-pack
ages (0.10.11)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: opencv-python in c:\users\nidhi\anaconda3\lib\site-
packages (4.9.0.80)
Requirement already satisfied: absl-py in c:\users\nidhi\anaconda3\lib\site-packag
es (from mediapipe) (2.0.0)
Requirement already satisfied: attrs>=19.1.0 in c:\users\nidhi\anaconda3\lib\site-
packages (from mediapipe) (23.1.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\nidhi\anaconda3\lib\si
te-packages (from mediapipe) (23.5.26)
Requirement already satisfied: jax in c:\users\nidhi\anaconda3\lib\site-packages
(from mediapipe) (0.4.23)
Requirement already satisfied: matplotlib in c:\users\nidhi\anaconda3\lib\site-pac
kages (from mediapipe) (3.7.2)
Requirement already satisfied: numpy in c:\users\nidhi\anaconda3\lib\site-packages
(from mediapipe) (1.24.3)
Requirement already satisfied: opencv-contrib-python in c:\users\nidhi\anaconda3\l
ib\site-packages (from mediapipe) (4.9.0.80)
Requirement already satisfied: protobuf<4,>=3.11 in c:\users\nidhi\anaconda3\lib\s
ite-packages (from mediapipe) (3.20.3)
Requirement already satisfied: sounddevice>=0.4.4 in c:\users\nidhi\anaconda3\lib
\site-packages (from mediapipe) (0.4.6)
Requirement already satisfied: CFFI>=1.0 in c:\users\nidhi\anaconda3\lib\site-pack
ages (from sounddevice>=0.4.4->mediapipe) (1.15.1)
Requirement already satisfied: ml-dtypes>=0.2.0 in c:\users\nidhi\anaconda3\lib\si
te-packages (from jax->mediapipe) (0.2.0)
Requirement already satisfied: opt-einsum in c:\users\nidhi\anaconda3\lib\site-pac
kages (from jax->mediapipe) (3.3.0)
Requirement already satisfied: scipy>=1.9 in c:\users\nidhi\anaconda3\lib\site-pac
kages (from jax->mediapipe) (1.11.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\nidhi\anaconda3\lib\si
te-packages (from matplotlib->mediapipe) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\nidhi\anaconda3\lib\site-p
ackages (from matplotlib->mediapipe) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\nidhi\anaconda3\lib\s
ite-packages (from matplotlib->mediapipe) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\nidhi\anaconda3\lib\s
ite-packages (from matplotlib->mediapipe) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\nidhi\appdata\roaming\p
ython\python311\site-packages (from matplotlib->mediapipe) (23.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\nidhi\anaconda3\lib\site-
packages (from matplotlib->mediapipe) (9.4.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\users\nidhi\anaconda3\l
ib\site-packages (from matplotlib->mediapipe) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\nidhi\appdata\roam
ing\python\python311\site-packages (from matplotlib->mediapipe) (2.8.2)
Requirement already satisfied: pycparser in c:\users\nidhi\anaconda3\lib\site-pack
ages (from CFFI>=1.0->sounddevice>=0.4.4->mediapipe) (2.21)
Requirement already satisfied: six>=1.5 in c:\users\nidhi\appdata\roaming\python\p
ython311\site-packages (from python-dateutil>=2.7->matplotlib->mediapipe) (1.16.0)
```

In [2]:
```
import cv2
import mediapipe as mp
import numpy as np
mp_drawing =mp.solutions.drawing_utils
mp_pose = mp.solutions.pose
```

```
WARNING:tensorflow:From C:\Users\nidhi\anaconda3\Lib\site-packages\keras\src\losse
s.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please u
se tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [3]:
```
#video capture
cap =cv2.VideoCapture(0)
while cap.isOpened():
    ret,frame =cap.read()
    cv2.imshow('Mediapipe', frame)
    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

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

In [5]:
```
cap =cv2.VideoCapture(0)
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pos
    while cap.isOpened():
        ret,frame =cap.read()

    #Recolouring bgr to rbg
        image = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
        image.flags.writeable =False

    #making detection
        results = pose.process(image)

    #recolouring rbg to bgr
        image.flags.writeable = True
        image = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    #Render detections
        mp_drawing.draw_landmarks(image,results.pose_landmarks,mp_pose.POSE_CONNECT
                                  mp_drawing.DrawingSpec(color=(245,117,66),thickne
                                  mp_drawing.DrawingSpec(color=(245,66,230),thickne
                                 )

        cv2.imshow('Mediapipe', image)
        if cv2.waitKey(10) & 0xFF == ord('q'):
         break

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

determining points

In [6]:
```
cap =cv2.VideoCapture(0)
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pos
    while cap.isOpened():
        ret,frame =cap.read()

    #Recolouring bgr to rbg
        image = cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
        image.flags.writeable =False

    #making detection
        results = pose.process(image)

    #recolouring rbg to bgr
        image.flags.writeable = True
        image = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

        try:
```

```
            landmarks = results.pose_landmarks.landmark
        except:
            pass
            Print(landmarks)
    #Render detections
        mp_drawing.draw_landmarks(image,results.pose_landmarks,mp_pose.POSE_CONNECT
                                  mp_drawing.DrawingSpec(color=(245,117,66),thickne
                                  mp_drawing.DrawingSpec(color=(245,66,230),thickne
                                 )

        cv2.imshow('Mediapipe', image)
        if cv2.waitKey(10) & 0xFF == ord('q'):
         break

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

In [7]:
```
for lndmrk in mp_pose.PoseLandmark:
    print(lndmrk)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
```

In [8]:
```
landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value]
```

Out[8]:
```
x: 0.708027
y: 0.9938754
z: -1.2707031
visibility: 0.9244154
```

In [9]:
```
landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value]
```

Out[9]:
```
x: 0.90105027
y: 1.2884082
z: -1.6343553
visibility: 0.17784917
```

In [10]:
```
landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value]
```

Out[10]:
```
x: 0.7688957
y: 1.3164604
z: -2.246477
visibility: 0.05887436
```

calculating angles

In [11]:
```
def calculate_angle(a,b,c):
    a = np.array(a)
    b = np.array(b)
    c = np.array(c)


    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
```

In [12]:
```
shoulder = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose
elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_pose.PoseL
wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_pose.PoseL
left_hip = [landmarks[mp_pose.PoseLandmark.LEFT_HIP.value].x,landmarks[mp_pose.Pose
left_knee = [landmarks[mp_pose.PoseLandmark.LEFT_KNEE.value].x,landmarks[mp_pose.Pc
left_ankle = [landmarks[mp_pose.PoseLandmark.LEFT_ANKLE.value].x,landmarks[mp_pose.
right_hip = [landmarks[mp_pose.PoseLandmark.RIGHT_HIP.value].x,landmarks[mp_pose.Pc
right_knee = [landmarks[mp_pose.PoseLandmark.RIGHT_KNEE.value].x,landmarks[mp_pose.
right_ankle = [landmarks[mp_pose.PoseLandmark.RIGHT_ANKLE.value].x,landmarks[mp_pos
```

In [13]:
```
shoulder, elbow , wrist
```

Out[13]:
```
([0.7080270051956177, 0.9938753843307495],
 [0.9010502696037292, 1.2884081602096558],
 [0.76889568567276, 1.316460371017456])
```

In [14]:
```
calculate_angle(left_hip, left_knee, left_ankle)
calculate_angle(right_hip, right_knee, right_ankle)
```

Out[14]:
```
123.94106293267085
```

In [15]:
```
calculate_angle(shoulder, elbow , wrist)
```

Out[15]:
```
68.74519516663077
```

In [18]:
```
tuple(np.multiply(elbow,[640,480]).astype(int))
```

Out[18]:
```
(576, 618)
```

DETECTION

In [17]:
```
import cv2
import mediapipe as mp
import numpy as np

mp_pose = mp.solutions.pose
mp_drawing = mp.solutions.drawing_utils

# Function to calculate angle
def calculate_angle(a, b, c):
    a = np.array(a)  # First point
    b = np.array(b)  # Mid point
    c = np.array(c)  # End point
    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

cap = cv2.VideoCapture(0)
desired_width = 1280
desired_height = 720
cap.set(cv2.CAP_PROP_FRAME_WIDTH, desired_width)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, desired_height)

# Initialize variables for bicep curl
bicep_counter = 0
bicep_sets = 0
bicep_stage = None

# Initialize variables for squat
squat_counter = 0
squat_sets = 0
squat_stage = None

# Flag to indicate when to switch from biceps to squats
switch_to_squats = False

# Initialize status box parameters
status_box_position = (10, 10)
status_box_size = (300, 150)
status_box_color = (0, 0, 0)  # Black
status_text_color = (255, 255, 255)  # White

with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pos
    while cap.isOpened():
        ret, frame = cap.read()

        # Recoloring BGR to RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Making detection
        results = pose.process(image)

        # Recoloring RGB to BGR
        image.flags.writeable = True
        image = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)

        try:
            landmarks = results.pose_landmarks.landmark
```

```
            # Bicep curl logic
            if not switch_to_squats:
                left_shoulder = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value
                                 landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value
                left_elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,
                              landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
                left_wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,
                              landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]
                bicep_angle = calculate_angle(left_shoulder, left_elbow, left_wrist

                if bicep_angle > 160:
                    bicep_stage = "down"
                if bicep_angle < 30 and bicep_stage == "down":
                    bicep_stage = "up"
                    bicep_counter += 1
                    print("Biceps Repetition:", bicep_counter)
                    if bicep_counter % 3 == 0:  # Check if 3 repetitions completed
                        bicep_sets += 1
                        print("Biceps Set:", bicep_sets)
                        if bicep_sets == 3:  # Check if 3 sets completed
                            print("Biceps workout completed. Squat exercise will nc
                            switch_to_squats = True
            else:
                # Squat logic
                # Your squat detection logic here...
                pass

        except:
            pass

        # Render detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNE
                                  mp_drawing.DrawingSpec(color=(245, 117, 66), thic
                                  mp_drawing.DrawingSpec(color=(245, 66, 230), thic

        # Display status box

        status_text = f"Bicep Curls: {bicep_counter}\nBicep Sets: {bicep_sets}\nBic

        if bicep_sets == 3:
            status_text += "\nBiceps workout completed. Starting next exercise..."

        cv2.putText(image, status_text, (status_box_position[0] + 10, status_box_po
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, status_text_color, 2, cv2.LINE_AA)


        cv2.imshow('Workout Detection', image)
        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

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

```
Biceps Repetition: 1
Biceps Repetition: 2
Biceps Repetition: 3
Biceps Set: 1
Biceps Repetition: 4
Biceps Repetition: 5
Biceps Repetition: 6
Biceps Set: 2
Biceps Repetition: 7
Biceps Repetition: 8
Biceps Repetition: 9
Biceps Set: 3
Biceps workout completed. Squat exercise will now be monitored.
```

In [ ]: