```python
#import the used libraries
import mediapipe as mp
import cv2
import time

#Create the hand detector class
class handDetector():
    #Initializate it's own variables
    def __init__( self, mode=False, maxHands=4, modelComplex = 1, detectionCon = 0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.modelComplex = modelComplex
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        #Initialize the hands solution object
        self.mpHands = mp.solutions.hands
        #Initialize the Hands object from inside the other object with the handDetector variables as operators
        self.hands = self.mpHands.Hands(self.mode, self.maxHands, self.modelComplex, self.detectionCon, self.trackCon)
        #Initialize the drawing object from mediapipe which draws on a image with cv2
        self.mpDraw = mp.solutions.drawing_utils

    #Create a function that process the image with the Landmarks of every point on the hands detected
    def findHands(self, img, drawCon = True, drawLan = True):
        #convert the bgr from cv2 to rgb that mediapipe is able to scan
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        #store the scanned image on a variable
        self.results = self.hands.process(imgRGB)
        #if there is some results on the variable, meaning that there is hands found on the image
        if self.results.multi_hand_landmarks:
            #For each Landmark in the list
            for handLms in self.results.multi_hand_landmarks:
                #If the operator says it, draws a scheme over the image to show the hand found
                if drawCon:
                    self.mpDraw.draw_landmarks(img, handLms, self.mpHands.HAND_CONNECTIONS)
                elif drawLan:
                    self.mpDraw.draw_landmarks(img, handLms)

        #return the image
        return img
    #Create a function that returns a list of coordinates from everi hand landmark on a image
    def findPosition(self, img, handNo = 0, draw = True):
        #initialize the list
        lmList = []
        #if there is any landmarks on the image
        if self.results.multi_hand_landmarks:
            #find the hand specified above landmarks
            myHand = self.results.multi_hand_landmarks[handNo]
            #for every landmark and it's id in the list of landmarks
            for id, lm in enumerate(myHand.landmark):
                #calculate the image shape
                h, w, c = img.shape
                #convert it to pixels
                cx, cy = int(lm.x*w), int(lm.y*h)
                #append a list of each id and coordinates to the main list
```

```
                lmList.append( [id, cx, cy])
                #if it has to draw a circle on a specific landmark it does
                if draw and id ==8 :
                    cv2.circle(img, (cx,cy), 15, (0,255,0), cv2.FILLED)
        #Return the list
        return lmList
#create the main function
def main():
    pTime = 0
    cTime = 0
    #initializate the webcam object
    cap = cv2.VideoCapture(0)

    #initializate the hand detector object
    detector = handDetector()

    #endlessly
    while True:
        #get a capture from the webcam
        success, img = cap.read()

        #find the hands that there could be on the image
        img = detector.findHands(img)

        #find the positions of that hands
        lmList = detector.findPosition(img)
        #if that list is full that means that there are some hands on the image
        if len(lmList) != 0:
            #Print it's coordinates
            print(lmList[4])

        #Calculate fps
        cTime = time.time()
        fps = 1/(cTime - pTime)
        pTime = cTime

        #insert the text of the fps in the images
        cv2.putText(img, str(int(fps)), (10,70), cv2.FONT_HERSHEY_PLAIN,3,(255,0,255),3)

        #show the images
        cv2.imshow("Image",img)
        cv2.waitKey(1)




#if it is the main call
if __name__ == "__main__":
    main()
```