```
import cv2

import numpy as np

import face_recognition

import os

from datetime import datetime



path = 'photo' # this contain known face images

images =[] # list to store the images

classNames = [] # list to store the names of the images

my_list = os.listdir(path) # list of all the images in the path
```

```
print(*my_list ) # print the list of images
for cls in my_list:
  curimg = cv2.imread(f'{path}/{cls}') # read the image
  images.append(curimg) # append the image to the list
  classNames.append(os.path.splitext(cls)[0]) # append the name of the image to the list
print(*classNames ) # print the names of the images
def findEncoding(image): # function to find the encoding of the image
  encodelist = []
  for img in images: # loop through the images
      img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # convert the image to RGB
```

```
        encode = face_recognition.face_encodings(img)[0] # find the encoding of the image
        encodelist.append(encode) # append the encoding to the list
    return encodelist # return the list of encodings


def markAttendance(name): # function to mark the attendance of the person
    with open('attendance.csv', 'r+') as f: # open the attendance file in read and write mode
        myDataList = f.readlines() # read the lines of the file
        nameList = [] # list to store the names of the people present
        for line in myDataList: # loop through the lines of the file
            entry = line.split(',') # split the line by comma
```

```
            nameList.append(entry[0]) # append the name to the list
        if name not in nameList: # if the name is not in the list
            now = datetime.now() # get the current date and time
            dtString = now.strftime('%H:%M:%S') # format the date and time
            f.writelines(f'\n{name},{dtString}') # write the name and date and time to the file


encodelistknown = findEncoding(images) # find the encodings of the known images
print(len(encodelistknown), "Encodings found") # print the number of encodings found
```

```
cap = cv2.VideoCapture(0) # open the webcam

while True: # loop until the webcam is closed

  success, img = cap.read()# read the frame from the webcam

  imgS = cv2.resize(img, (0,0),None,0.25,0.25) # resize the frame to 1/4th of the original size

  imgS = cv2.cvtColor(imgS,cv2.COLOR_BGR2RGB) # convert the frame to RGB

  facesCurFrame = face_recognition.face_locations(imgS) # find the faces in the frame

  encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame) # find the encodings of the faces in the frame



  for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame): # loop through the encodings and the face locations

      matches = face_recognition.compare_faces(encodelistknown, encodeFace)
```

```
    faceDistance = face_recognition.face_distance(encodelistknown, encodeFace)

    matchesIndex = np.argmin(faceDistance) # find the index of the minimum face distance

    if matches[matchesIndex]:

        name = classNames[matchesIndex].upper()

        y1, x2, y2, x1 = faceLoc # get the face location

        y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4 # scale the face location back to the original size

        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

        cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255,0), cv2.FILLED)

        cv2.putText(img, name, (x1 , y2 ), cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2) # draw a rectangle around the face and put
the name on the rectangle
```

```
        markAttendance(name)



cv2.imshow('Webcam', img) # display the frame with the face locations and names

if cv2.waitKey(1) & 0xFF == ord('d'): # if the 'd' key is pressedd

    break
```