```
import cv2
import face_recognition
import numpy as np
import os
import datetime

# Initialize variables
known_face_encodings = []
known_face_names = []
attendance = {}

# Load known faces
def load_known_faces():
    images_folder = "known_faces/"  # Folder containing images of known students
    for filename in os.listdir(images_folder):
        if filename.endswith(".jpg") or filename.endswith(".png"):
            image_path = os.path.join(images_folder, filename)
            image = face_recognition.load_image_file(image_path)
            face_encoding = face_recognition.face_encodings(image)[0]
            known_face_encodings.append(face_encoding)
            known_face_names.append(os.path.splitext(filename)[0])  # Use filename without extension as name

load_known_faces()

# Initialize video capture
video_capture = cv2.VideoCapture(0)
```

```
while True:
    # Capture frame-by-frame
    ret, frame = video_capture.read()

    # Resize frame for faster processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    # Convert the image from BGR color (which OpenCV uses) to RGB color
    rgb_small_frame = small_frame[:, :, ::-1]

    # Find all face locations and face encodings in the current frame
    face_locations = face_recognition.face_locations(rgb_small_frame)
    face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)

    current_names = []

    # Loop through each face found in the frame
    for face_encoding in face_encodings:
        # Check if the face is a match for any known face
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

        # Use the first match.
        if True in matches:
            first_match_index = matches.index(True)
            name = known_face_names[first_match_index]

        current_names.append(name)
```

```
        # Mark attendance
        if name != "Unknown":
            attendance[name] = attendance.get(name, 0) + 1

    # Display the results
    for (top, right, bottom, left), name in zip(face_locations, current_names):
        # Scale back up face locations since the frame we detected in was scaled down
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        # Draw a box around the face and label it
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
        cv2.putText(frame, name, (left + 6, bottom - 6), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255,
255, 255), 1)

    # Display the resulting image
    cv2.imshow('Video', frame)

    # Exit if 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Save attendance to a file
with open('attendance.txt', 'a') as f:
    f.write(f"{datetime.datetime.now()}: {attendance}\n")
```

```
# Release handle to the webcam
video_capture.release()
cv2.destroyAllWindows()
```