**CV Assignment 0**

**Aryan Jain 2019101056**

**\*the complete drive link which contains jupyter notebook and all inputs and outputs is available at:**

[**https://drive.google.com/drive/u/0/folders/1X-PREcb9vKmqzgEXL9jFlJM30rCgyPZn**](https://drive.google.com/drive/u/0/folders/1X-PREcb9vKmqzgEXL9jFlJM30rCgyPZn)

**Problem 1**

**Description:**

Given a video, it is asked to break the video into frames and save them in a specified folder. In the next part it is asked to combine the frames saved in the folder to generate a video whose frame rate can be varied based on user’s choice.

**Solution:**

Used cv2.VideoCapture() to get an object of the input video and then used read() function to get the frames. In my case the video used to generate the frames is named ‘input.mp4’ and the frames are stored in ‘images’ folder in the same directory. The video formed by combining the saved frames is stored as ‘output.mp4’ . For the 2nd part I took the frames per second as 30, which can be changed based on user’s preference by just changing the ‘fps’ variable in the code.

**Experiments:**

The ‘fps’ variable in the 2nd part (frames per second) is adjusted in order to get the continuous effect in the video, rather than a fast forward flow of frames.

**Challenges:**

As such no challenges were faced.

**Learning:**

The persistence of vision of human eye is 1/16 seconds, that means a human eye can distinguish 16 frames in a second. So to create an optical illusion effect or a continuous flow effect in videos, we need to keep frame rate to be more than 16 fps. So was did and I pick fps as 30 for my case.

**Results:**

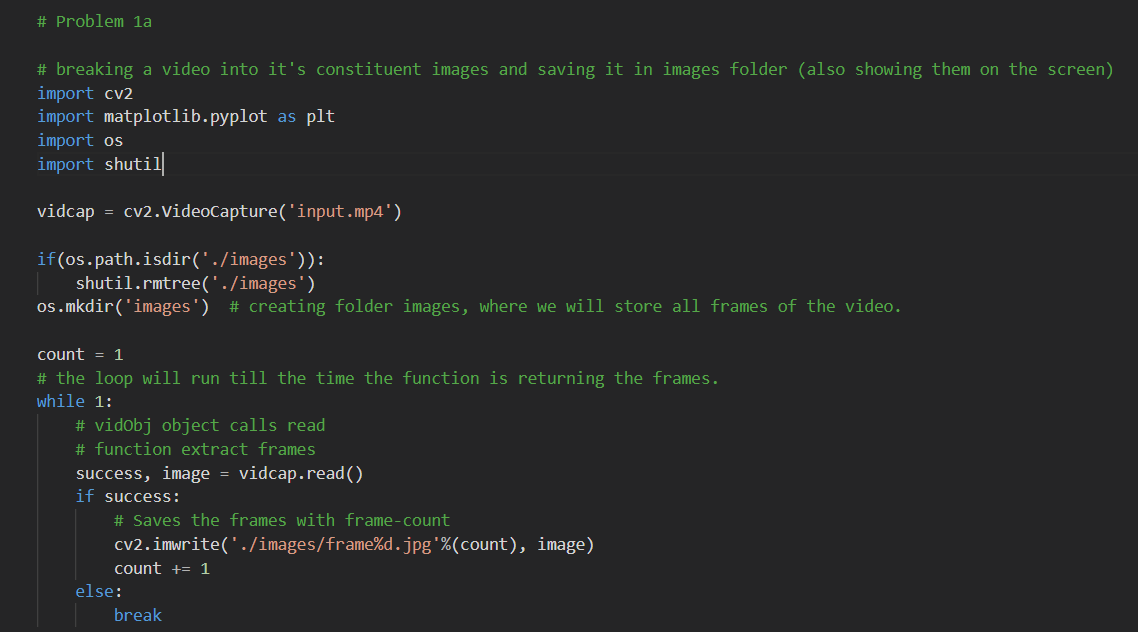
The frames formed using the input.mp4 video are available at this link (for 1a) :

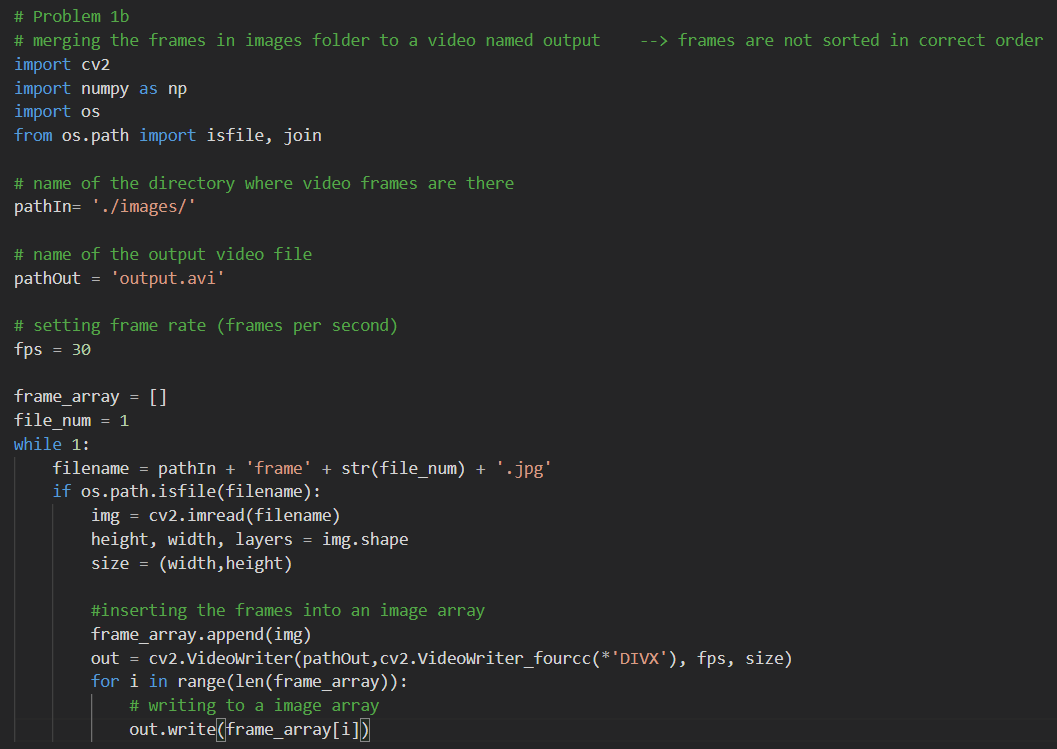
<https://drive.google.com/drive/u/0/folders/1thN-VW_31uWMAEm0cyhTRyVG1ig-P5fJ>

The video (output.avi) is available at this link (for 1b):

<https://drive.google.com/drive/u/0/folders/1X-PREcb9vKmqzgEXL9jFlJM30rCgyPZn>

**Code :**

****

****

**Problem 2**

**Description:**

In this problem, we are asked to capture the frames from a webcam connected to the computer and save them in some folder. Also, we need to show the video on the screen while recording.

**Solution:**

Here we create an object (named ‘cap’) using cv2.VideoCapture(0) and run an infinite ‘while loop’ using cap.isOpened() function. The loop will run for infinite time and will break only when the ‘q’ (as exit) is pressed. Meanwhile the frame are saved in ‘webcam\_images’ folder and video is shown on the screen using ‘cv2.imshow()’ function.

**Experiments:**

No experiment is involved.

**Challenges:**

No challenge is involved.

**Learning:**

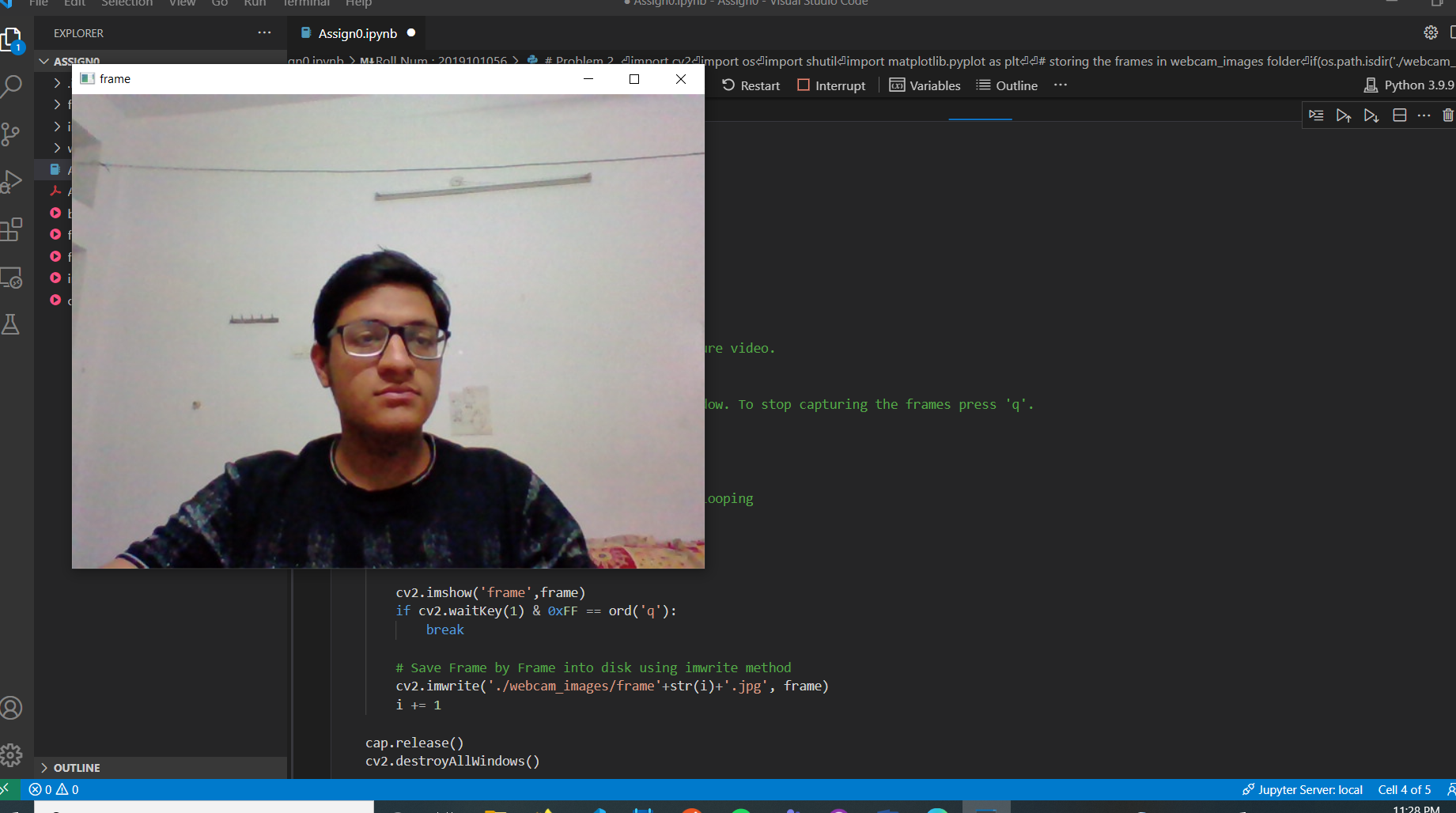
Learnt how to capture frames from a webcam connected to the computer and simultaneously display the video recorded on the screen.

**Results:**

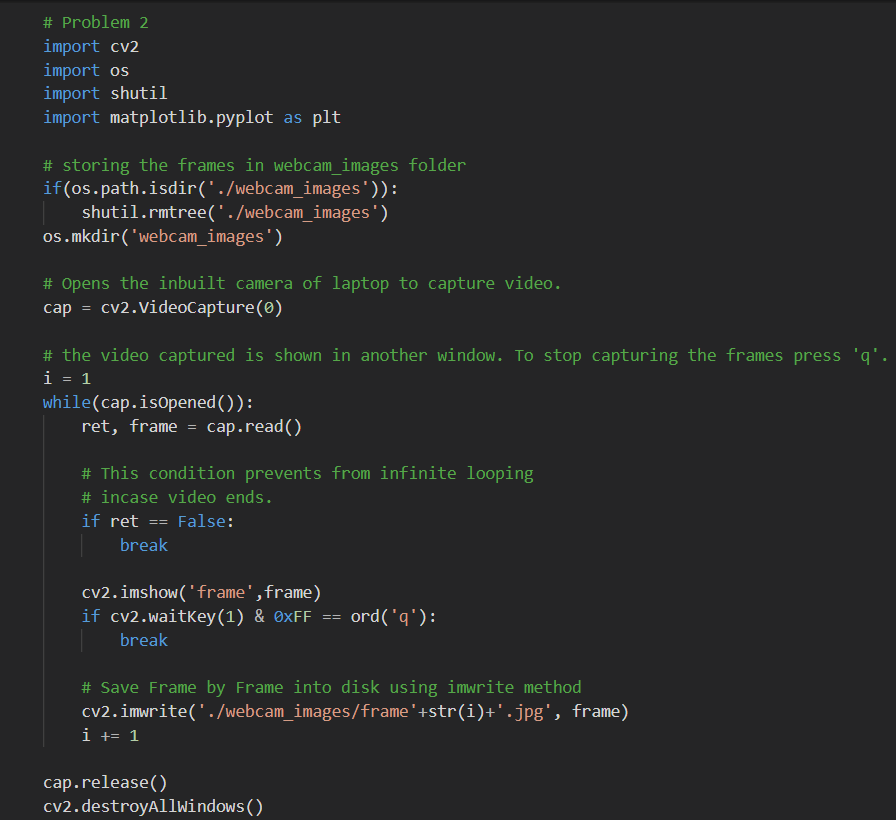
The frames captured by webcam are saved in a local folder and is made available at :

<https://drive.google.com/drive/u/0/folders/1TG7iQ8a0mUGDHOEKnctOWmYwq7SsYH0I>

Below is the proof of displaying the video on the screen while capturing the frames through webcam.

****

**Code:**



**Problem 3**

**Description:**

Using the Chroma Keying technique, we are asked to create an interesting composition of 2 videos.

**Solution:**

In the first video (in my case I called it as ‘foreground.mp4’), we will take a green screen background video. And we can take nice background as the 2nd video (called ‘background.mp4’ in my case). We will divide the videos into frames, and for the first video we will create a mask using which we will be remove the green screen from the background and extract the require portion. The same mask can be used for the background video frames to mask the region where the characters from the foreground video are coming.

After this we will combine both the images and store these new frames in another folder named ‘fg-bg\_images’. After this we will combine the frames into video just like we did in 2nd part of Problem 1.

**Challenges:**

The main challenge was in picking the values for ‘lower\_green’ and ‘upper\_green’ triplet, which is used for creating mask for the green screen video. (These triplets are defined in the code).

**Experiment:**

The values in these triplets(lower\_green and upper\_green) are needed to be chosen in such a way that it removes the green screen from the background of the main character in the foreground.mp4 video, which is a tricky task because all green screen background is not same in all videos. (Different green color cloth can be used to film the scenes at different places, different time). So we need to choose an appropriate value for it.

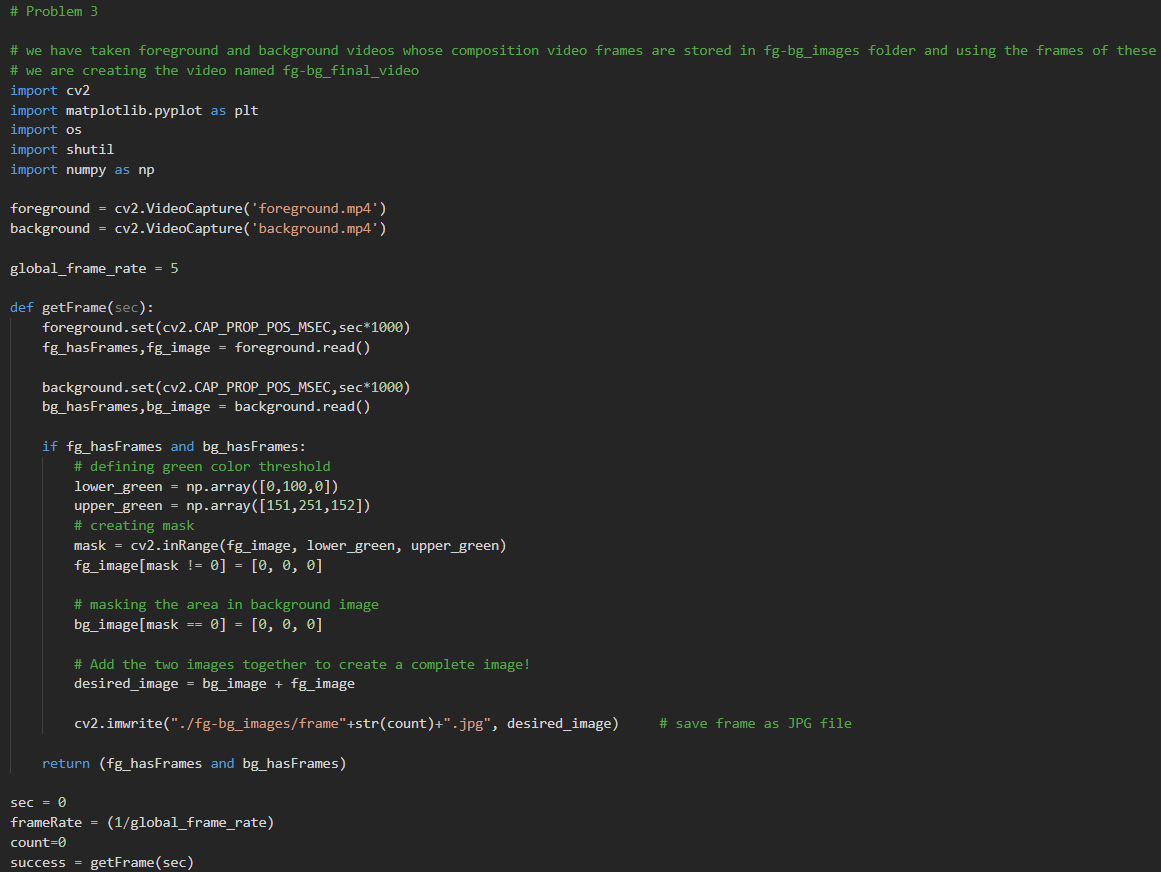
**Learning:**

Learnt how to apply Chroma Keying and understood how it is used in general to create videos and movies with nice background.

**Results:**

The final video (fg-bg\_final\_video.avi) formed by the composition of 2 videos using the Chroma Keying is available at :

<https://drive.google.com/drive/u/0/folders/1X-PREcb9vKmqzgEXL9jFlJM30rCgyPZn>

**Code:**