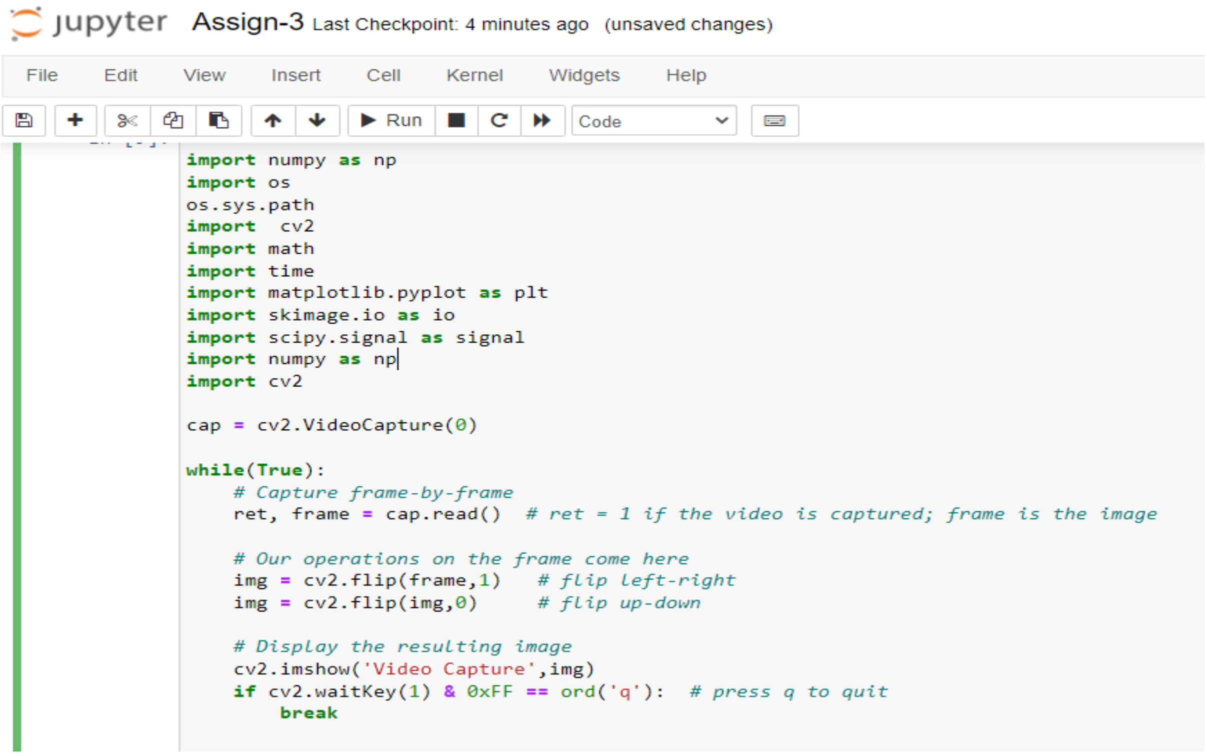
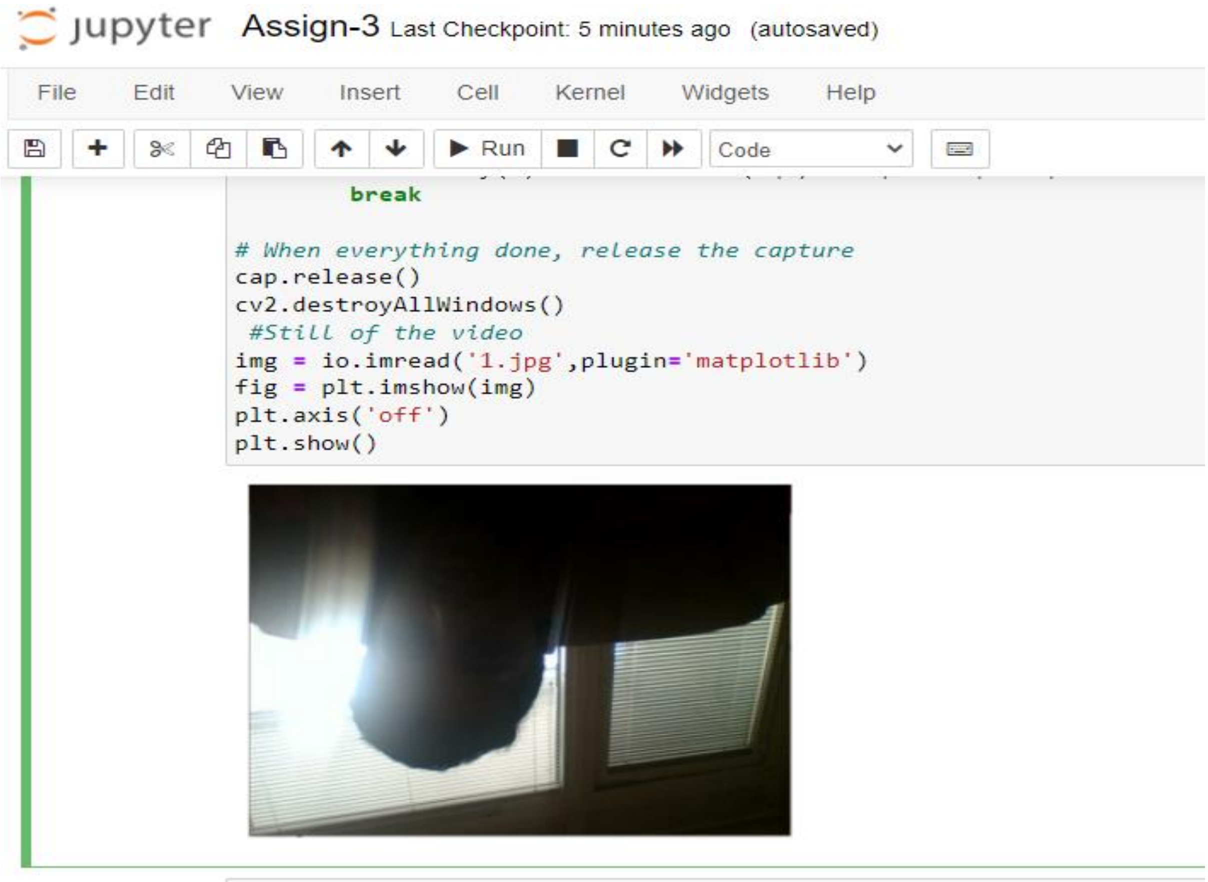
Assignment#3

Step:1



Assign-3 Last Checkpoint: 4 minutes ago (unsaved changes)

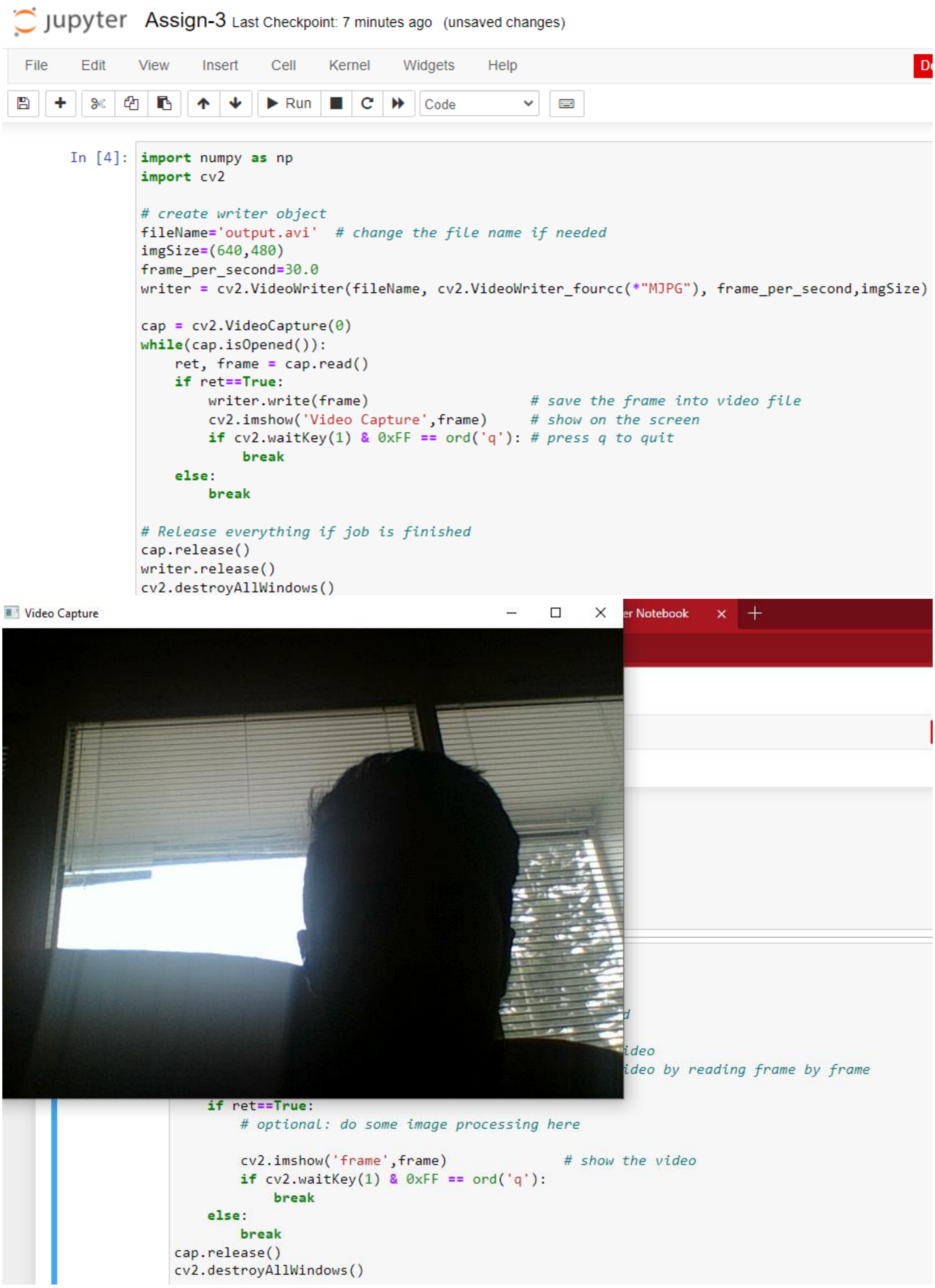
```
import numpy as np
import os
os.sys.path
import  cv2
import math
import time
import matplotlib.pyplot as plt
import skimage.io as io
import scipy.signal as signal
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()  # ret = 1 if the video is captured; frame is the image

    # Our operations on the frame come here
    img = cv2.flip(frame,1)   # flip left-right
    img = cv2.flip(img,0)     # flip up-down

    # Display the resulting image
    cv2.imshow('Video Capture',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):  # press q to quit
        break
```



Assign-3 Last Checkpoint: 5 minutes ago (autosaved)

```
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
 #Still of the video
img = io.imread('1.jpg',plugin='matplotlib')
fig = plt.imshow(img)
plt.axis('off')
plt.show()
```

Step:2

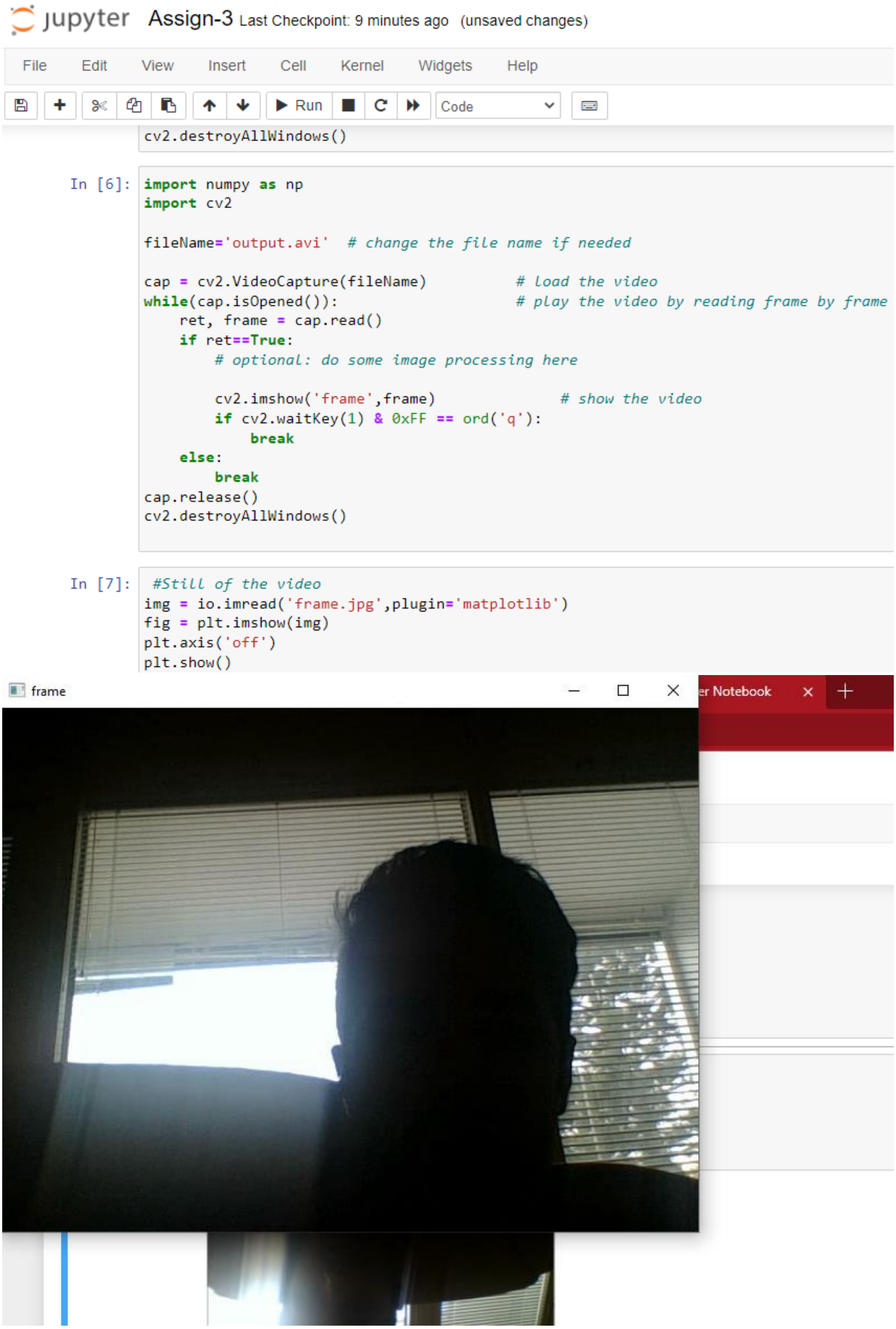


In [4]:

```
import numpy as np
import cv2

# create writer object
fileName='output.avi'  # change the file name if needed
imgSize=(640,480)
frame_per_second=30.0
writer = cv2.VideoWriter(fileName, cv2.VideoWriter_fourcc(*"MJPG"), frame_per_second,imgSize)

cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    ret, frame = cap.read()
    if ret==True:
        writer.write(frame)                     # save the frame into video file
        cv2.imshow('Video Capture',frame)       # show on the screen
        if cv2.waitKey(1) & 0xFF == ord('q'): # press q to quit
            break
    else:
        break

# Release everything if job is finished
cap.release()
writer.release()
cv2.destroyAllWindows()
```

```
d
ideo
ideo by reading frame by frame
    if ret==True:
        # optional: do some image processing here

        cv2.imshow('frame',frame)              # show the video
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```

Step:3



```
cv2.destroyAllWindows()
```

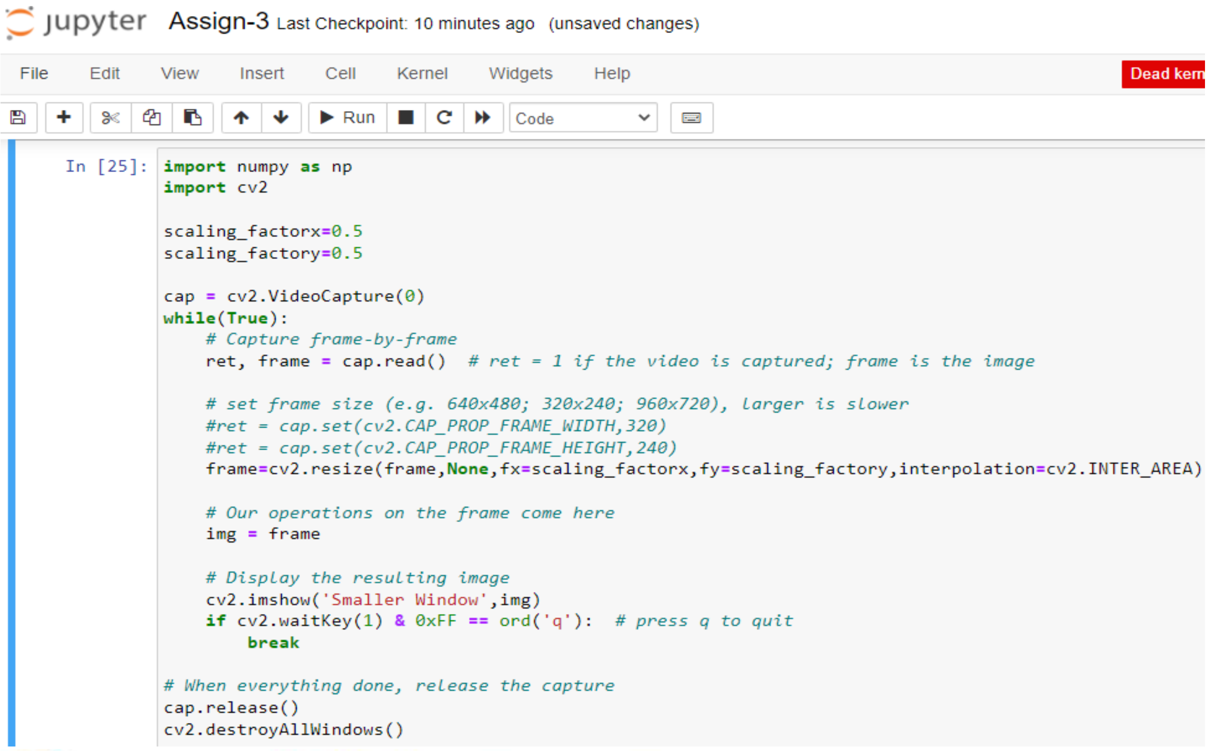
In [6]:

```
import numpy as np
import cv2

fileName='output.avi'  # change the file name if needed

cap = cv2.VideoCapture(fileName)          # load the video
while(cap.isOpened()):                    # play the video by reading frame by frame
    ret, frame = cap.read()
    if ret==True:
        # optional: do some image processing here

        cv2.imshow('frame',frame)              # show the video
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        break
cap.release()
cv2.destroyAllWindows()
```

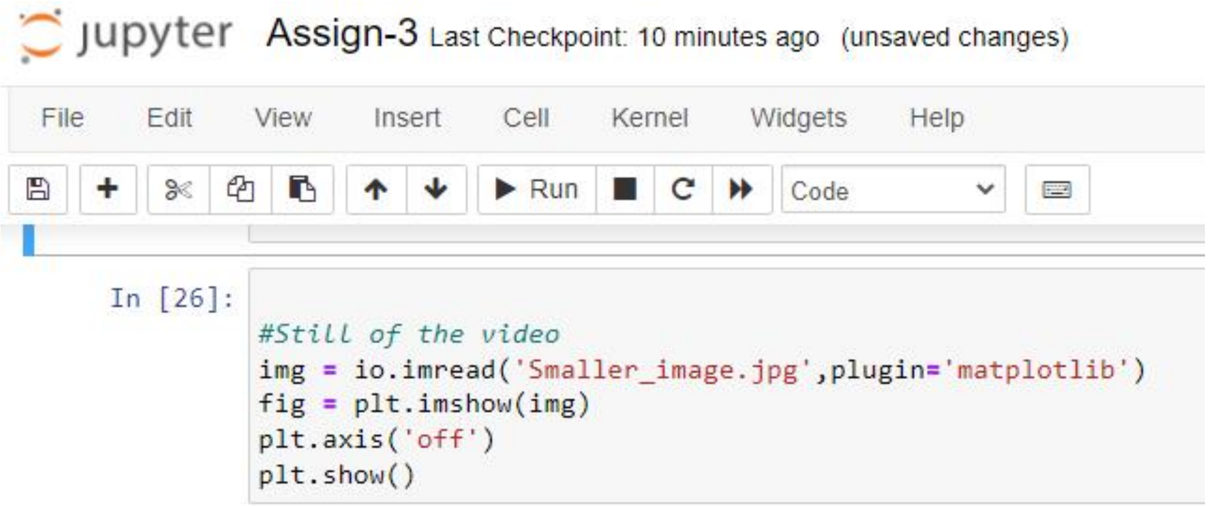
In [7]:

```
 #Still of the video
img = io.imread('frame.jpg',plugin='matplotlib')
fig = plt.imshow(img)
plt.axis('off')
plt.show()
```

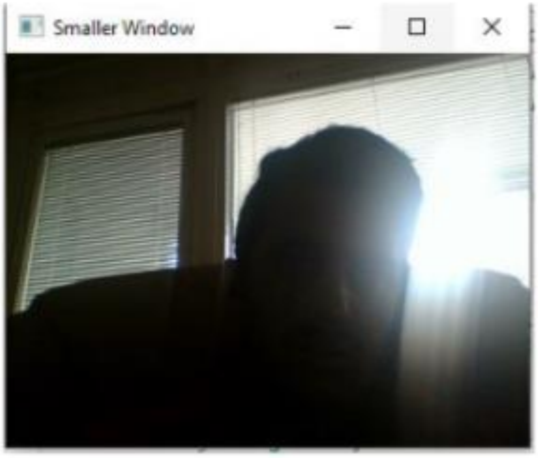
Step:4



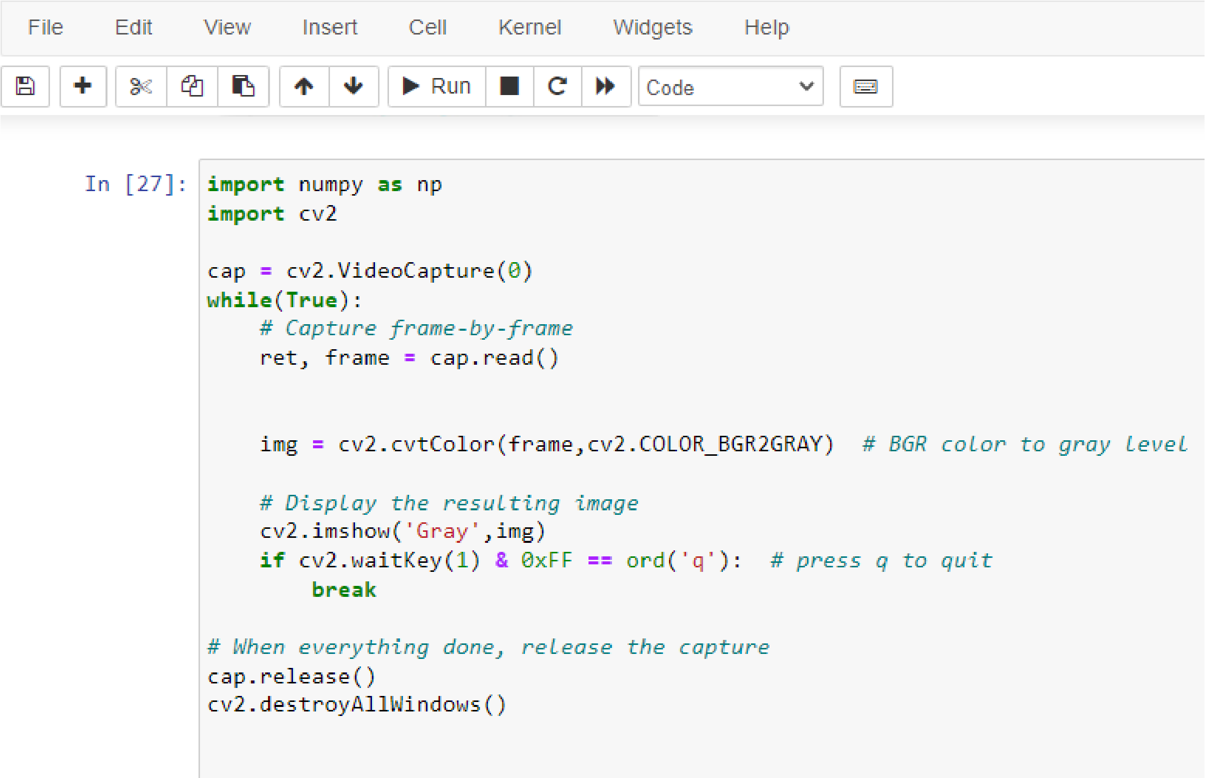
```
import numpy as np
import cv2

scaling_factorx=0.5
scaling_factory=0.5

cap = cv2.VideoCapture(0)
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()  # ret = 1 if the video is captured; frame is the image

    # set frame size (e.g. 640x480; 320x240; 960x720), larger is slower
    #ret = cap.set(cv2.CAP_PROP_FRAME_WIDTH,320)
    #ret = cap.set(cv2.CAP_PROP_FRAME_HEIGHT,240)
    frame=cv2.resize(frame,None,fx=scaling_factorx,fy=scaling_factory,interpolation=cv2.INTER_AREA)

    # Our operations on the frame come here
    img = frame

    # Display the resulting image
    cv2.imshow('Smaller Window',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):  # press q to quit
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```



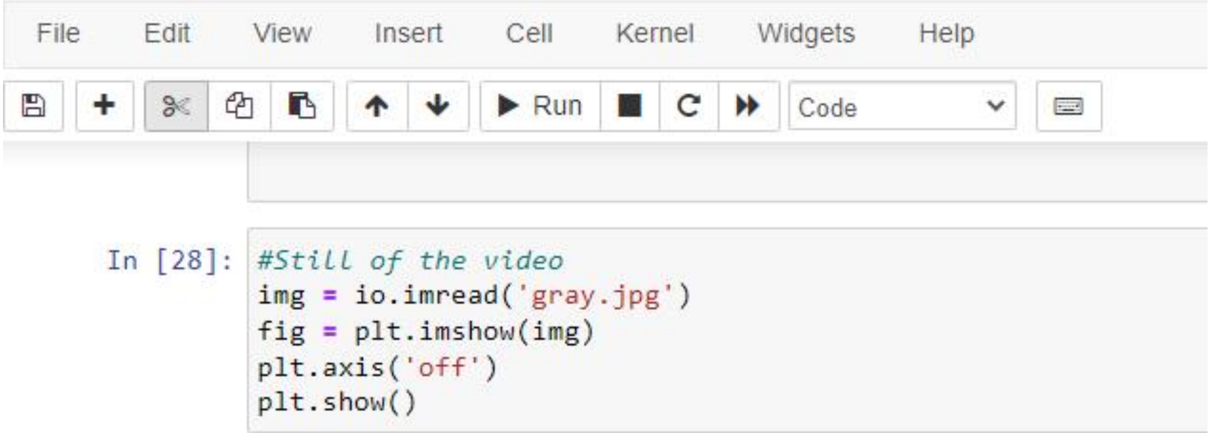
```
#Still of the video
img = io.imread('Smaller_image.jpg',plugin='matplotlib')
fig = plt.imshow(img)
plt.axis('off')
plt.show()
```



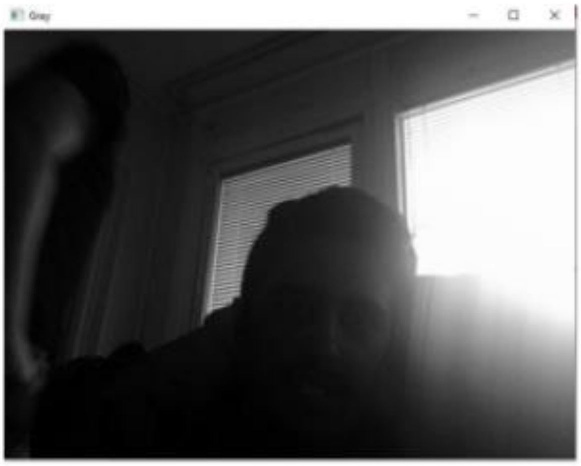
Step:5



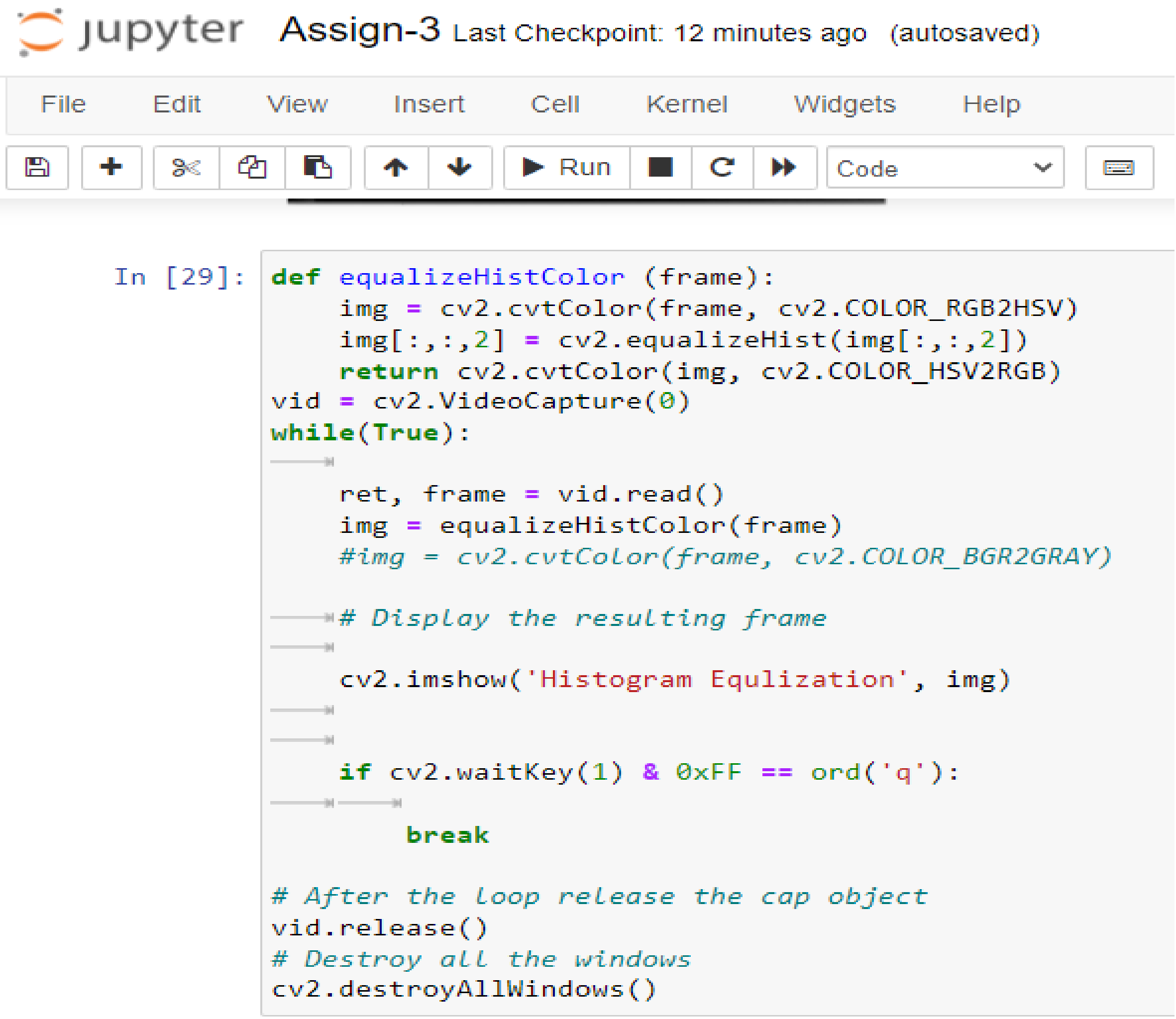
In [27]:
```python
import numpy as np
import cv2

cap = cv2.VideoCapture(0)
while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()


    img = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)  # BGR color to gray level

    # Display the resulting image
    cv2.imshow('Gray',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):  # press q to quit
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```



In [28]:
```python
#Still of the video
img = io.imread('gray.jpg')
fig = plt.imshow(img)
plt.axis('off')
plt.show()
```

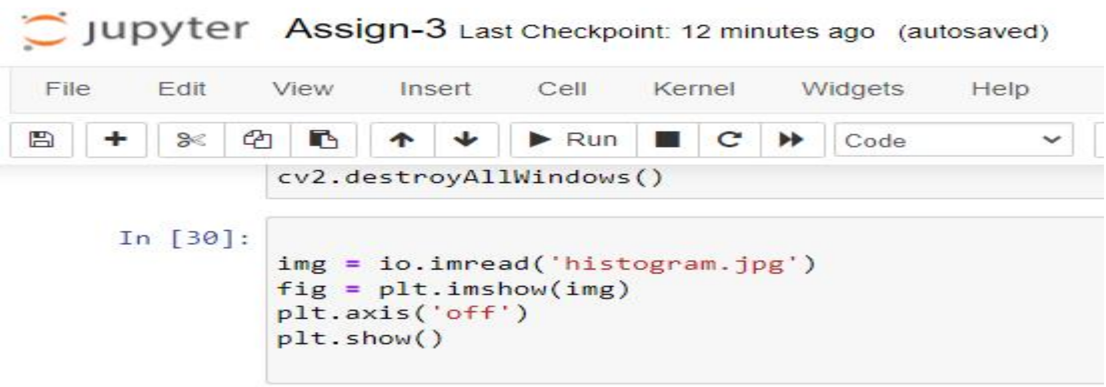


Step:6



In [29]:

```
def equalizeHistColor (frame):
    img = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)
    img[:,:,2] = cv2.equalizeHist(img[:,:,2])
    return cv2.cvtColor(img, cv2.COLOR_HSV2RGB)
vid = cv2.VideoCapture(0)
while(True):

    ret, frame = vid.read()
    img = equalizeHistColor(frame)
    #img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Display the resulting frame

    cv2.imshow('Histogram Equlization', img)


    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# After the loop release the cap object
vid.release()
# Destroy all the windows
cv2.destroyAllWindows()
```

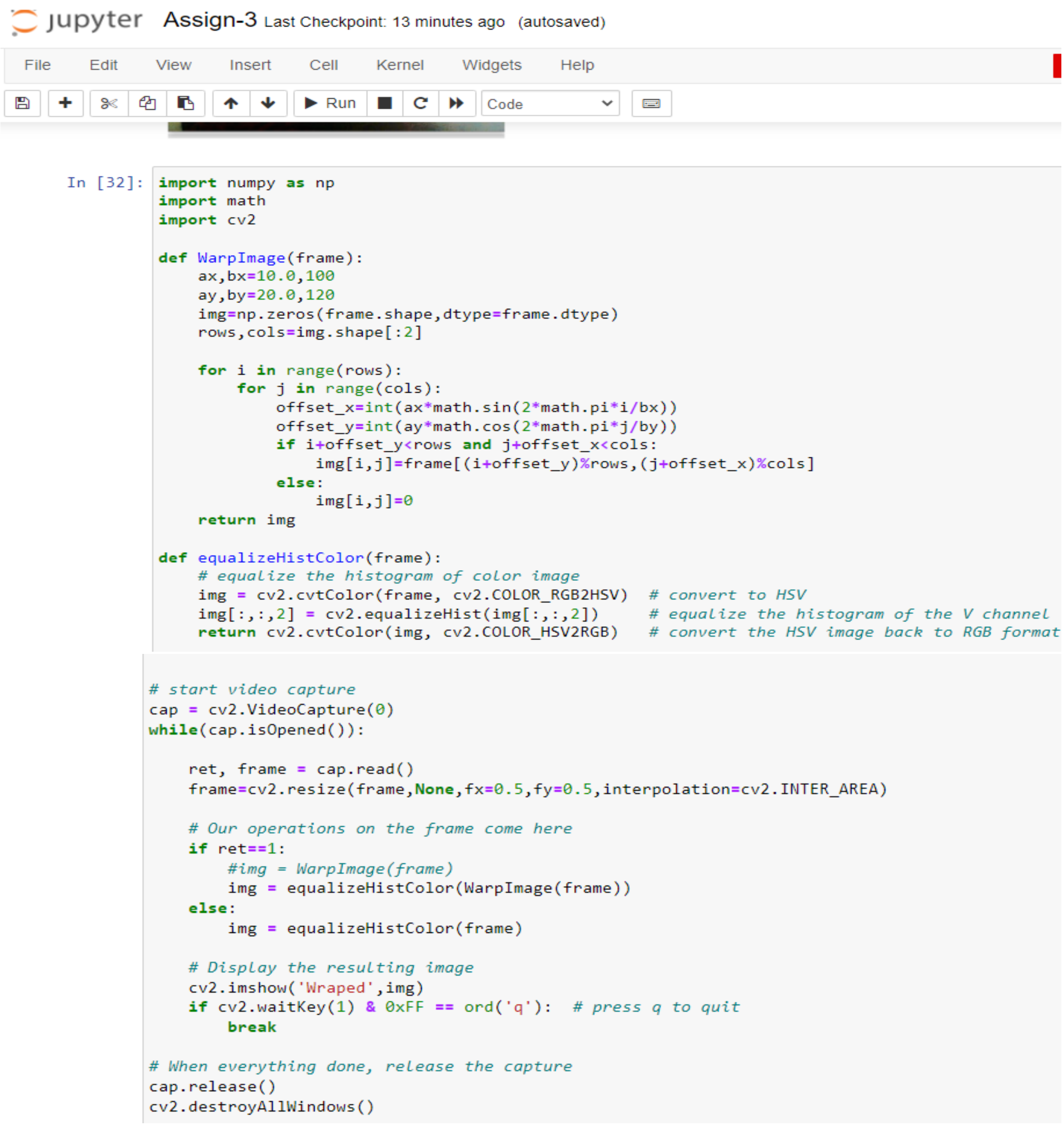


```
cv2.destroyAllWindows()
```

In [30]:

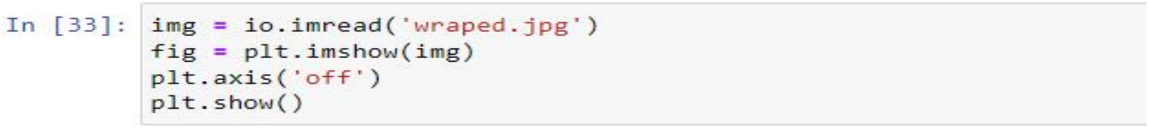
```
img = io.imread('histogram.jpg')
fig = plt.imshow(img)
plt.axis('off')
plt.show()
```

Step:7

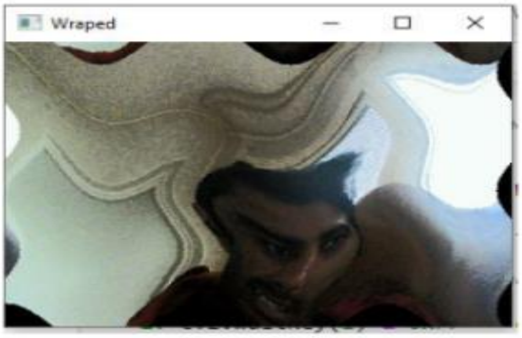


In [32]:
```
import numpy as np
import math
import cv2

def WarpImage(frame):
    ax,bx=10.0,100
    ay,by=20.0,120
    img=np.zeros(frame.shape,dtype=frame.dtype)
    rows,cols=img.shape[:2]

    for i in range(rows):
        for j in range(cols):
            offset_x=int(ax*math.sin(2*math.pi*i/bx))
            offset_y=int(ay*math.cos(2*math.pi*j/by))
            if i+offset_y<rows and j+offset_x<cols:
                img[i,j]=frame[(i+offset_y)%rows,(j+offset_x)%cols]
            else:
                img[i,j]=0
    return img

def equalizeHistColor(frame):
    # equalize the histogram of color image
    img = cv2.cvtColor(frame, cv2.COLOR_RGB2HSV)  # convert to HSV
    img[:,:,2] = cv2.equalizeHist(img[:,:,2])     # equalize the histogram of the V channel
    return cv2.cvtColor(img, cv2.COLOR_HSV2RGB)   # convert the HSV image back to RGB format
```

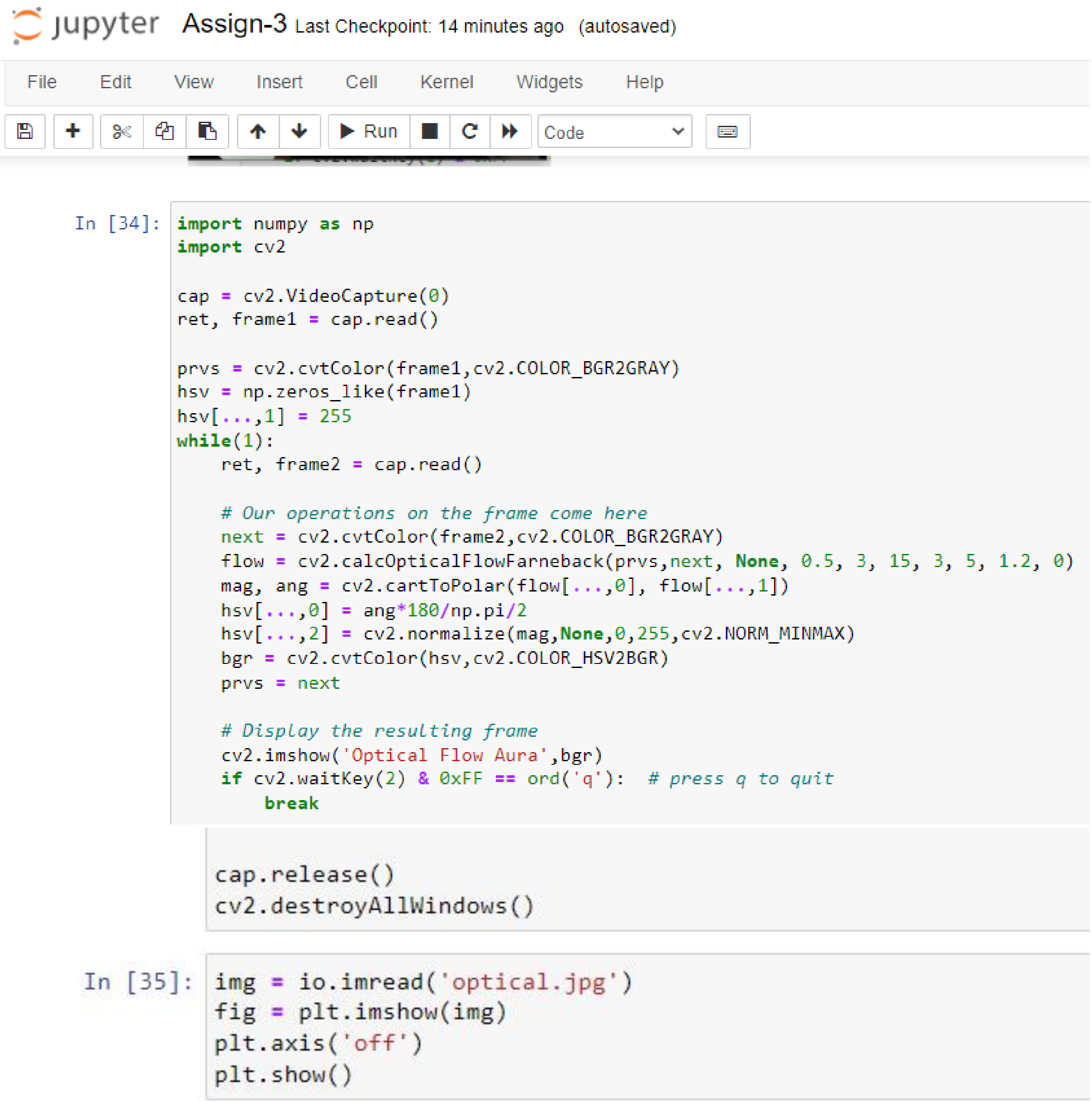
```
# start video capture
cap = cv2.VideoCapture(0)
while(cap.isOpened()):

    ret, frame = cap.read()
    frame=cv2.resize(frame,None,fx=0.5,fy=0.5,interpolation=cv2.INTER_AREA)

    # Our operations on the frame come here
    if ret==1:
        #img = WarpImage(frame)
        img = equalizeHistColor(WarpImage(frame))
    else:
        img = equalizeHistColor(frame)

    # Display the resulting image
    cv2.imshow('Wraped',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):  # press q to quit
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```



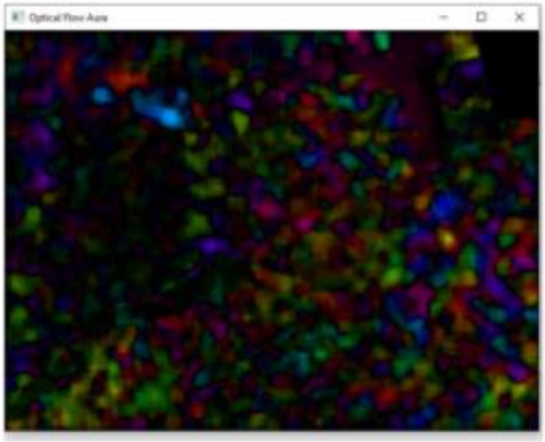
In [33]:
```
img = io.imread('wraped.jpg')
fig = plt.imshow(img)
plt.axis('off')
plt.show()
```



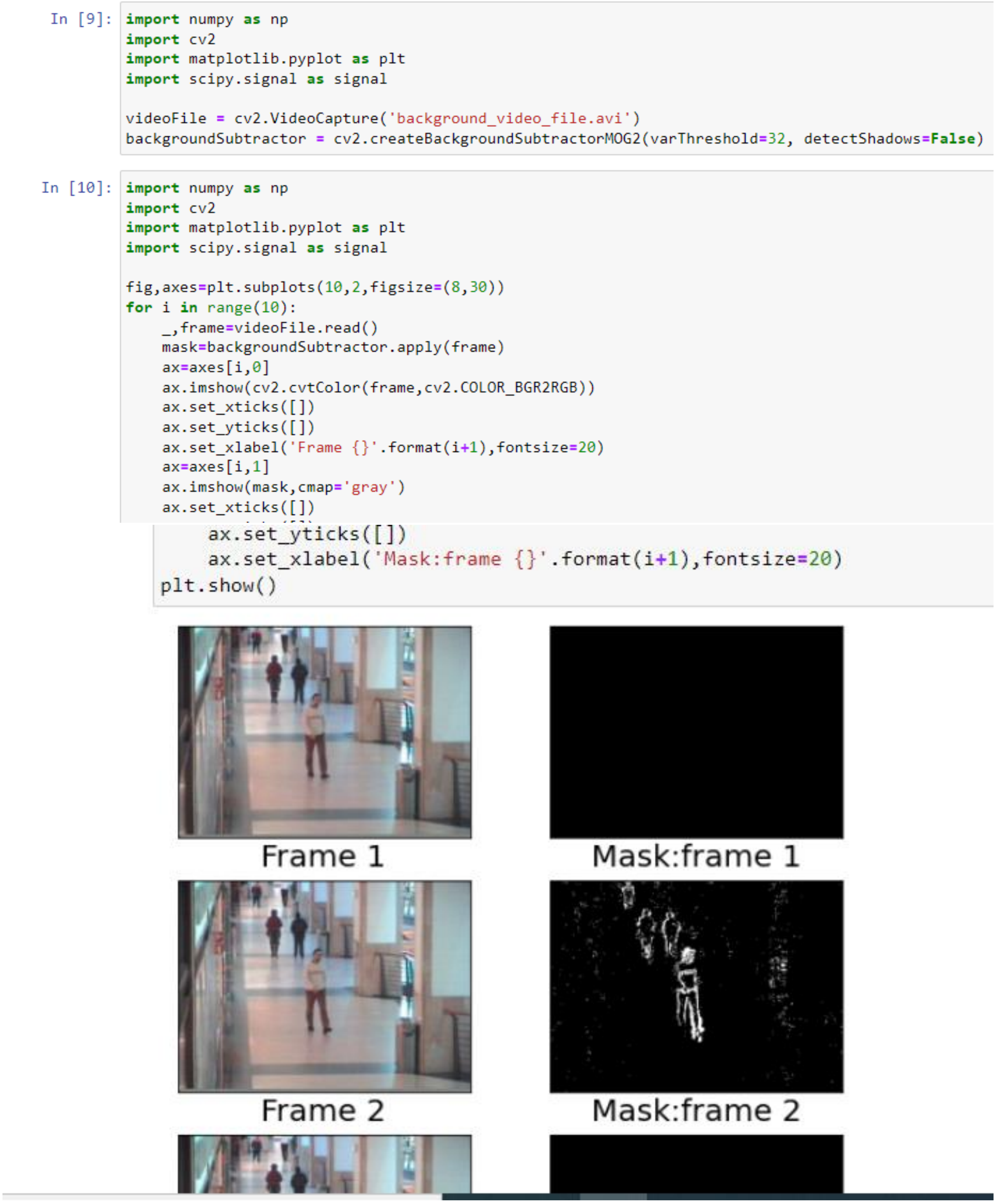
Step:8



```
In [34]: import numpy as np
         import cv2

         cap = cv2.VideoCapture(0)
         ret, frame1 = cap.read()

         prvs = cv2.cvtColor(frame1,cv2.COLOR_BGR2GRAY)
         hsv = np.zeros_like(frame1)
         hsv[...,1] = 255
         while(1):
             ret, frame2 = cap.read()

             # Our operations on the frame come here
             next = cv2.cvtColor(frame2,cv2.COLOR_BGR2GRAY)
             flow = cv2.calcOpticalFlowFarneback(prvs,next, None, 0.5, 3, 15, 3, 5, 1.2, 0)
             mag, ang = cv2.cartToPolar(flow[...,0], flow[...,1])
             hsv[...,0] = ang*180/np.pi/2
             hsv[...,2] = cv2.normalize(mag,None,0,255,cv2.NORM_MINMAX)
             bgr = cv2.cvtColor(hsv,cv2.COLOR_HSV2BGR)
             prvs = next

             # Display the resulting frame
             cv2.imshow('Optical Flow Aura',bgr)
             if cv2.waitKey(2) & 0xFF == ord('q'):  # press q to quit
                 break

         cap.release()
         cv2.destroyAllWindows()

In [35]: img = io.imread('optical.jpg')
         fig = plt.imshow(img)
         plt.axis('off')
         plt.show()
```

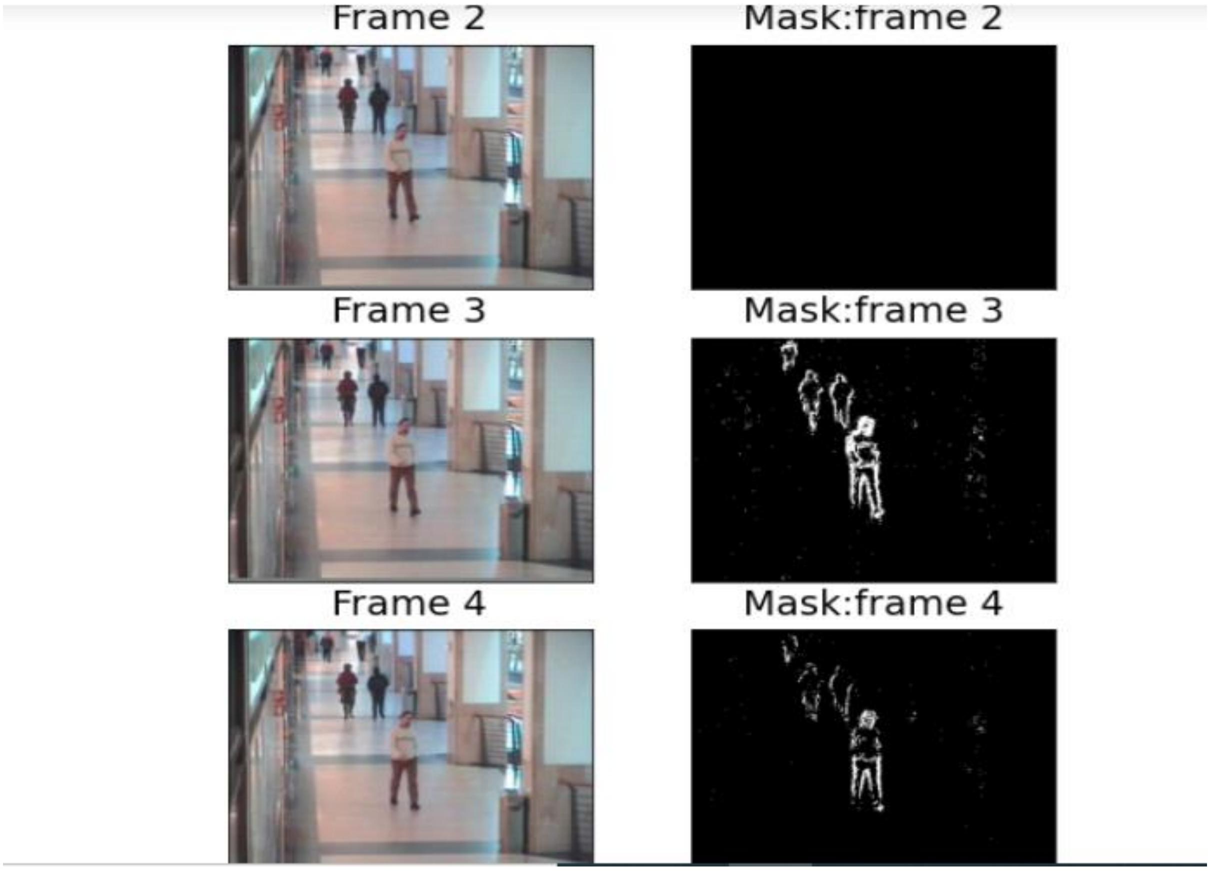


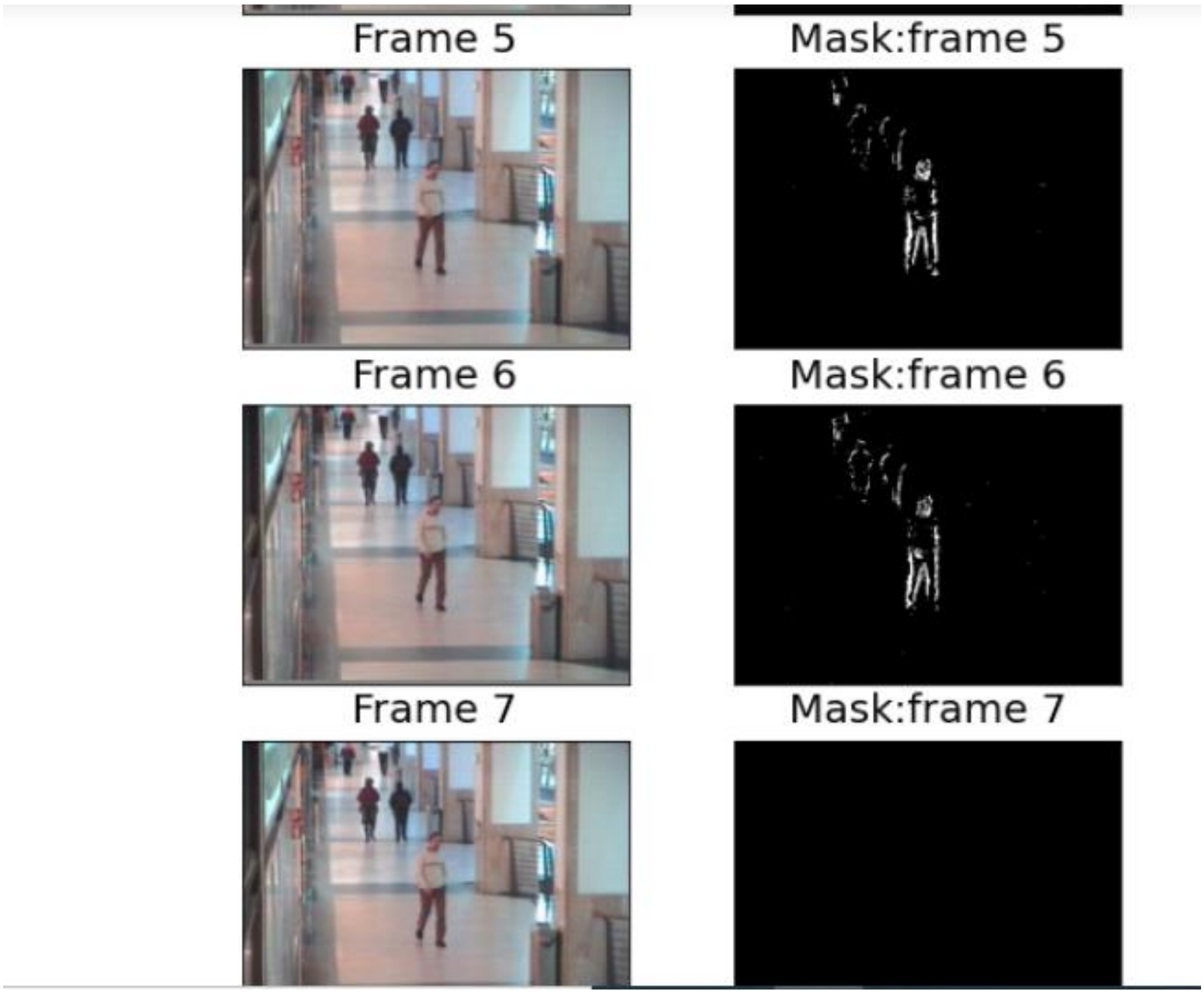
Step:9

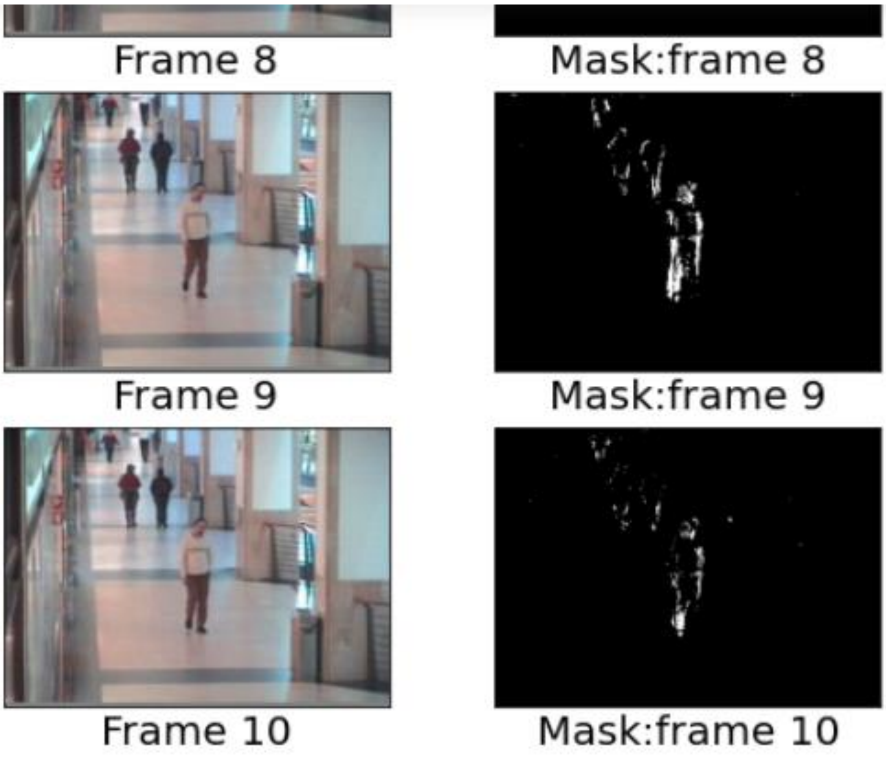


In [9]:
```python
import numpy as np
import cv2
import matplotlib.pyplot as plt
import scipy.signal as signal

videoFile = cv2.VideoCapture('background_video_file.avi')
backgroundSubtractor = cv2.createBackgroundSubtractorMOG2(varThreshold=32, detectShadows=False)
```

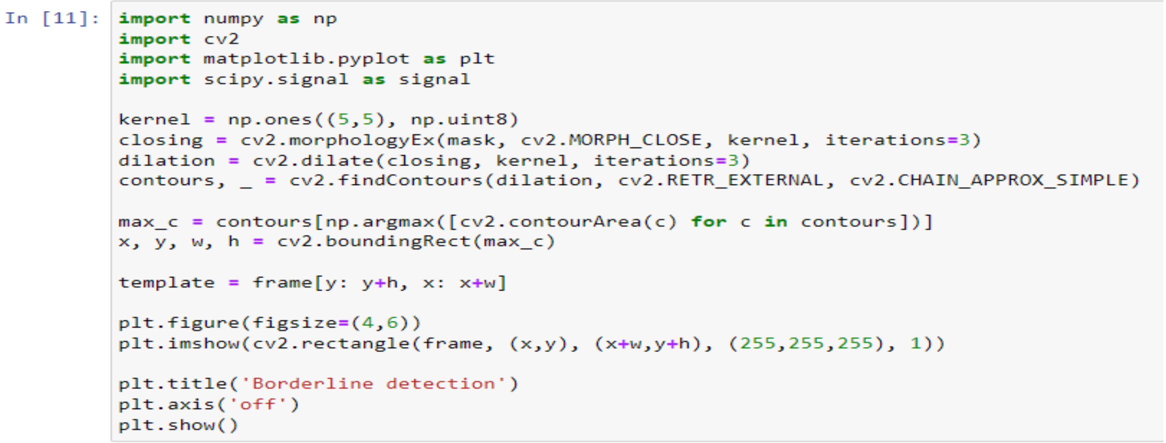
In [10]:
```python
import numpy as np
import cv2
import matplotlib.pyplot as plt
import scipy.signal as signal

fig,axes=plt.subplots(10,2,figsize=(8,30))
for i in range(10):
    _,frame=videoFile.read()
    mask=backgroundSubtractor.apply(frame)
    ax=axes[i,0]
    ax.imshow(cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))
    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_xlabel('Frame {}'.format(i+1),fontsize=20)
    ax=axes[i,1]
    ax.imshow(mask,cmap='gray')
    ax.set_xticks([])
    ax.set_yticks([])
    ax.set_xlabel('Mask:frame {}'.format(i+1),fontsize=20)
plt.show()
```


Frame 2
Mask:frame 2
Frame 3
Mask:frame 3
Frame 4
Mask:frame 4

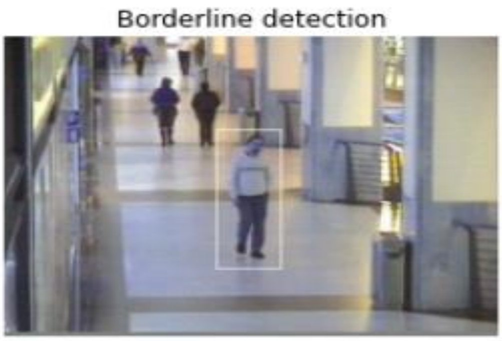

Frame 5
Mask:frame 5
Frame 6
Mask:frame 6
Frame 7
Mask:frame 7



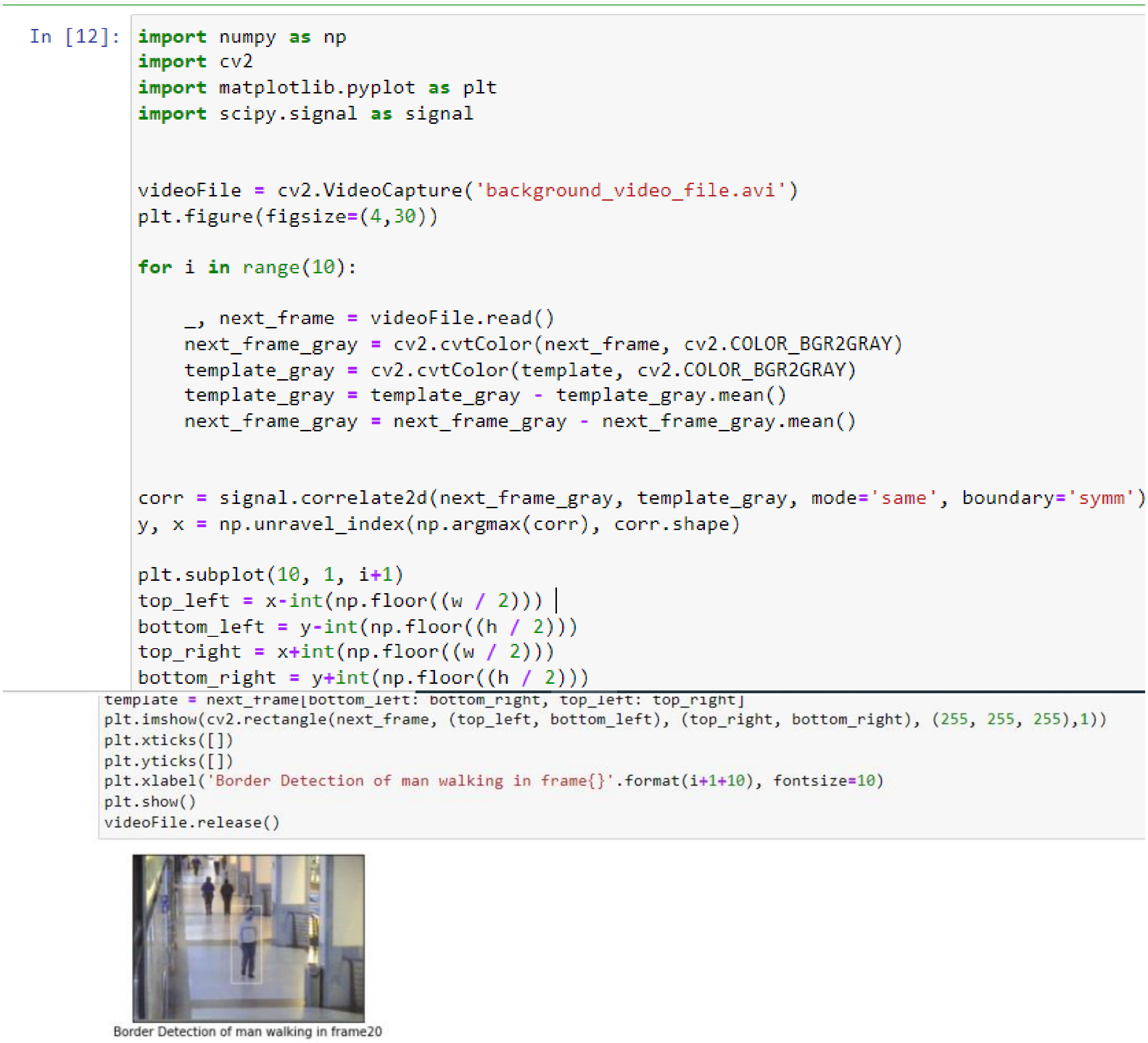
Step:10



```
In [11]: import numpy as np
         import cv2
         import matplotlib.pyplot as plt
         import scipy.signal as signal

         kernel = np.ones((5,5), np.uint8)
         closing = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel, iterations=3)
         dilation = cv2.dilate(closing, kernel, iterations=3)
         contours, _ = cv2.findContours(dilation, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

         max_c = contours[np.argmax([cv2.contourArea(c) for c in contours])]
         x, y, w, h = cv2.boundingRect(max_c)

         template = frame[y: y+h, x: x+w]

         plt.figure(figsize=(4,6))
         plt.imshow(cv2.rectangle(frame, (x,y), (x+w,y+h), (255,255,255), 1))

         plt.title('Borderline detection')
         plt.axis('off')
         plt.show()
```



Step:11



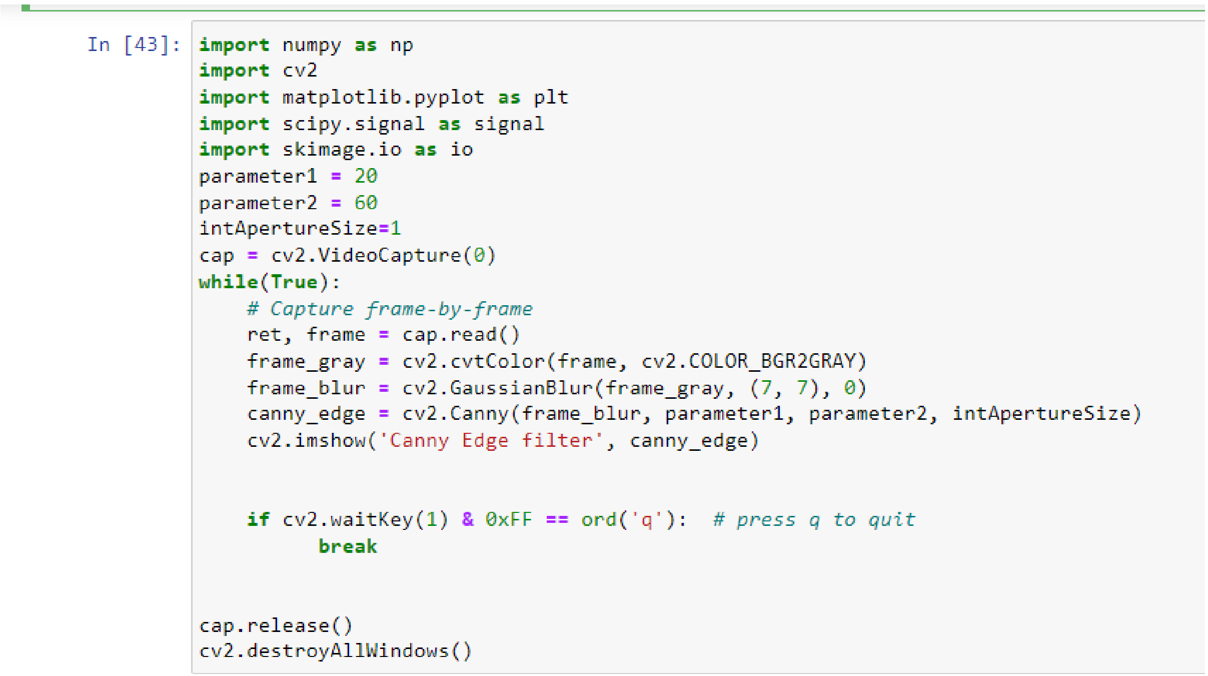
```
In [12]: import numpy as np
         import cv2
         import matplotlib.pyplot as plt
         import scipy.signal as signal


         videoFile = cv2.VideoCapture('background_video_file.avi')
         plt.figure(figsize=(4,30))

         for i in range(10):

             _, next_frame = videoFile.read()
             next_frame_gray = cv2.cvtColor(next_frame, cv2.COLOR_BGR2GRAY)
             template_gray = cv2.cvtColor(template, cv2.COLOR_BGR2GRAY)
             template_gray = template_gray - template_gray.mean()
             next_frame_gray = next_frame_gray - next_frame_gray.mean()


         corr = signal.correlate2d(next_frame_gray, template_gray, mode='same', boundary='symm')
         y, x = np.unravel_index(np.argmax(corr), corr.shape)

         plt.subplot(10, 1, i+1)
         top_left = x-int(np.floor((w / 2)))
         bottom_left = y-int(np.floor((h / 2)))
         top_right = x+int(np.floor((w / 2)))
         bottom_right = y+int(np.floor((h / 2)))
```

```
template = next_frame[bottom_left: bottom_right, top_left: top_right]
plt.imshow(cv2.rectangle(next_frame, (top_left, bottom_left), (top_right, bottom_right), (255, 255, 255),1))
plt.xticks([])
plt.yticks([])
plt.xlabel('Border Detection of man walking in frame{}'.format(i+1+10), fontsize=10)
plt.show()
videoFile.release()
```

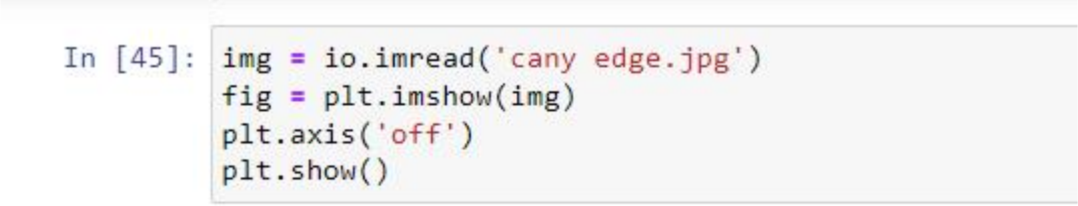
Border Detection of man walking in frame20

Challenging Question Answers

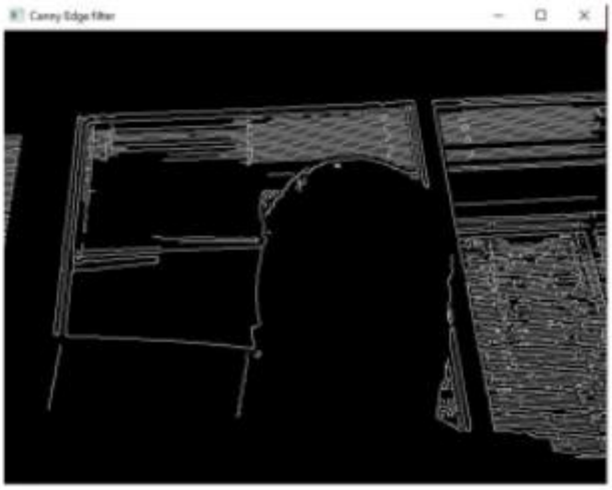
Question-1:



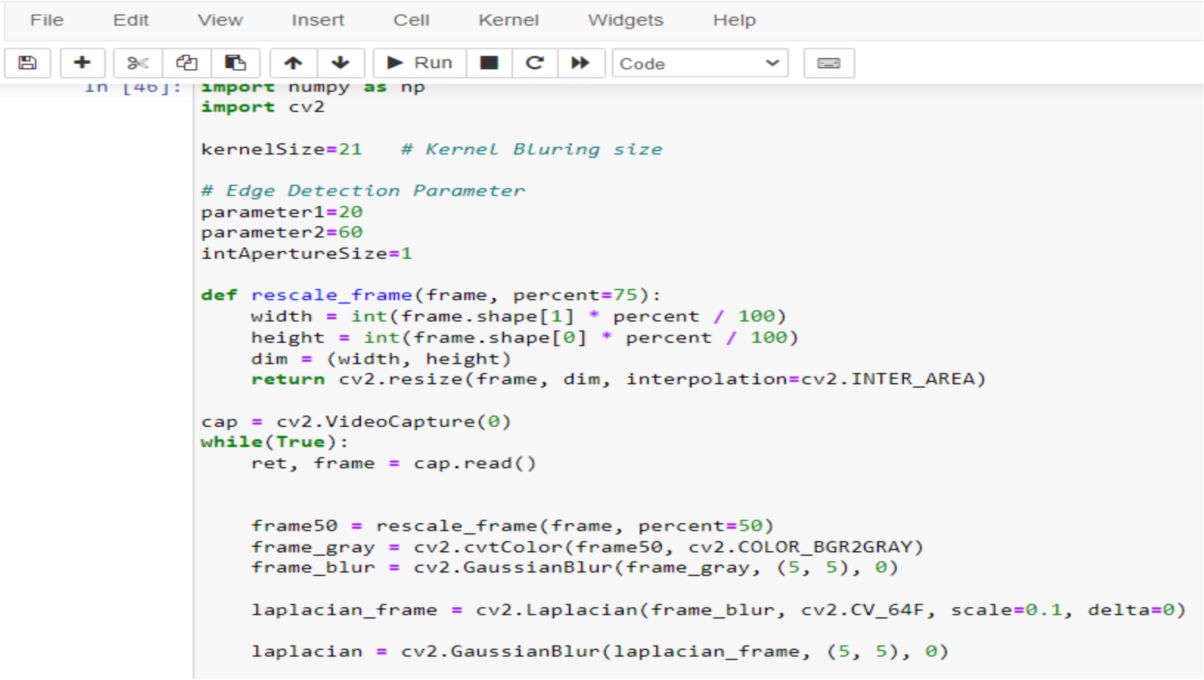
```
In [43]: import numpy as np
         import cv2
         import matplotlib.pyplot as plt
         import scipy.signal as signal
         import skimage.io as io
         parameter1 = 20
         parameter2 = 60
         intApertureSize=1
         cap = cv2.VideoCapture(0)
         while(True):
             # Capture frame-by-frame
             ret, frame = cap.read()
             frame_gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
             frame_blur = cv2.GaussianBlur(frame_gray, (7, 7), 0)
             canny_edge = cv2.Canny(frame_blur, parameter1, parameter2, intApertureSize)
             cv2.imshow('Canny Edge filter', canny_edge)


             if cv2.waitKey(1) & 0xFF == ord('q'):  # press q to quit
                   break


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



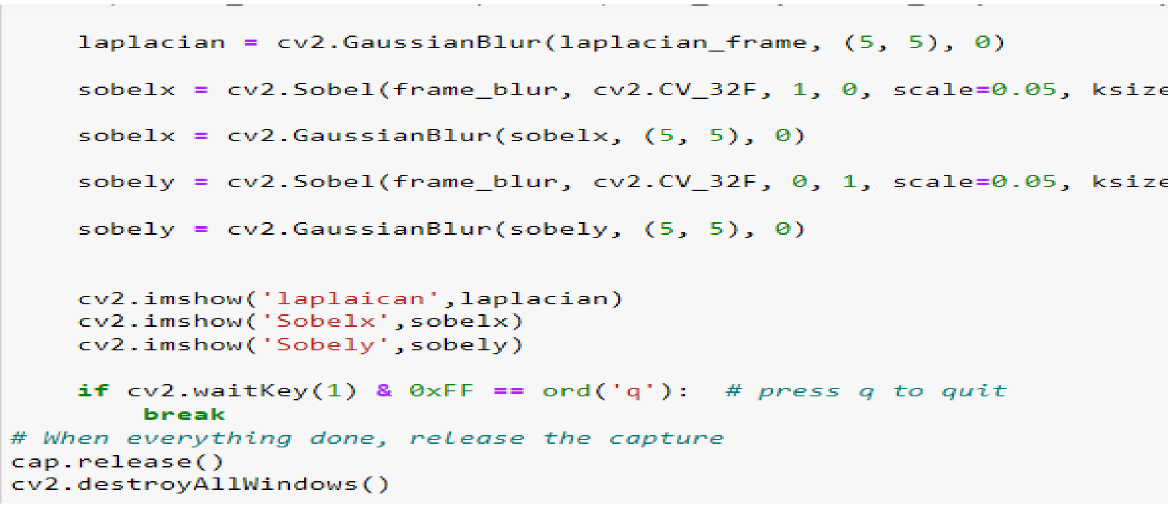
```
In [45]: img = io.imread('cany edge.jpg')
         fig = plt.imshow(img)
         plt.axis('off')
         plt.show()
```



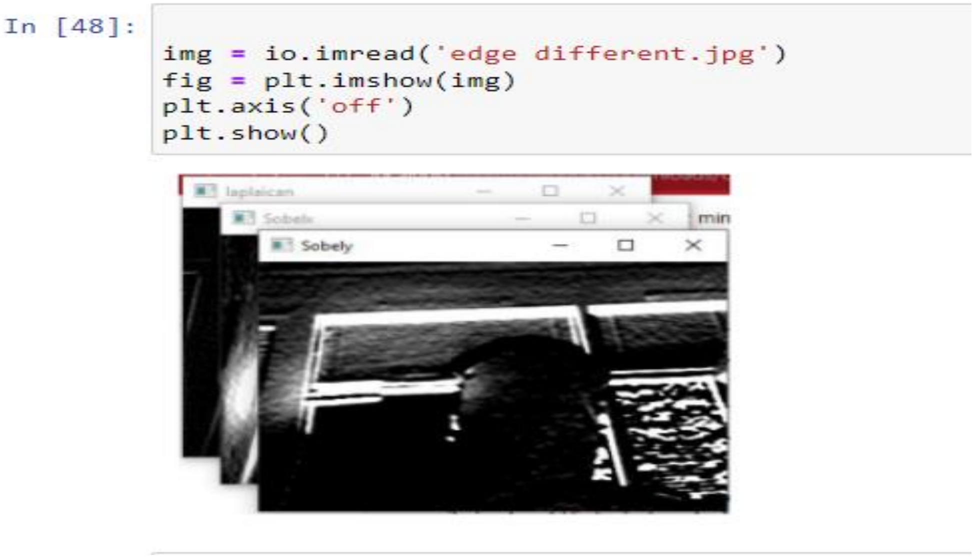
Question:2



```
In [46]: import numpy as np
         import cv2

         kernelSize=21    # Kernel Bluring size

         # Edge Detection Parameter
         parameter1=20
         parameter2=60
         intApertureSize=1

         def rescale_frame(frame, percent=75):
             width = int(frame.shape[1] * percent / 100)
             height = int(frame.shape[0] * percent / 100)
             dim = (width, height)
             return cv2.resize(frame, dim, interpolation=cv2.INTER_AREA)

         cap = cv2.VideoCapture(0)
         while(True):
             ret, frame = cap.read()


             frame50 = rescale_frame(frame, percent=50)
             frame_gray = cv2.cvtColor(frame50, cv2.COLOR_BGR2GRAY)
             frame_blur = cv2.GaussianBlur(frame_gray, (5, 5), 0)

             laplacian_frame = cv2.Laplacian(frame_blur, cv2.CV_64F, scale=0.1, delta=0)

             laplacian = cv2.GaussianBlur(laplacian_frame, (5, 5), 0)
```

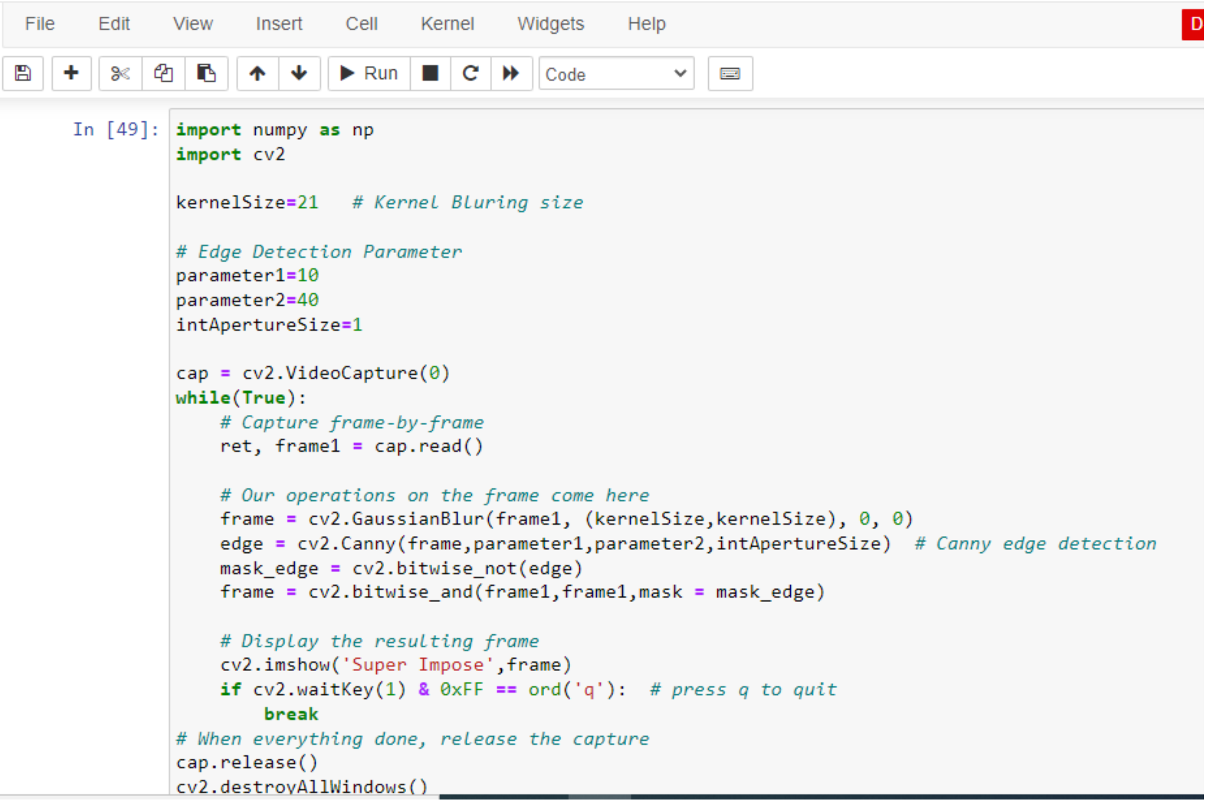


```
    laplacian = cv2.GaussianBlur(laplacian_frame, (5, 5), 0)

    sobelx = cv2.Sobel(frame_blur, cv2.CV_32F, 1, 0, scale=0.05, ksize

    sobelx = cv2.GaussianBlur(sobelx, (5, 5), 0)

    sobely = cv2.Sobel(frame_blur, cv2.CV_32F, 0, 1, scale=0.05, ksize

    sobely = cv2.GaussianBlur(sobely, (5, 5), 0)


    cv2.imshow('laplaican',laplacian)
    cv2.imshow('Sobelx',sobelx)
    cv2.imshow('Sobely',sobely)

    if cv2.waitKey(1) & 0xFF == ord('q'):   # press q to quit
        break
# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```



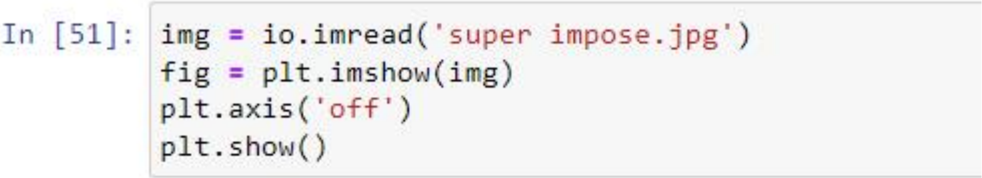
```
In [48]:
          img = io.imread('edge different.jpg')
          fig = plt.imshow(img)
          plt.axis('off')
          plt.show()
```

Question:3



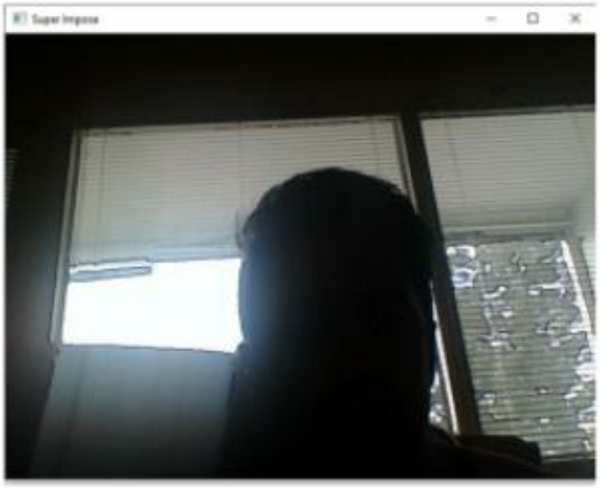
In [49]:

```
import numpy as np
import cv2

kernelSize=21   # Kernel Bluring size

# Edge Detection Parameter
parameter1=10
parameter2=40
intApertureSize=1

cap = cv2.VideoCapture(0)
while(True):
    # Capture frame-by-frame
    ret, frame1 = cap.read()

    # Our operations on the frame come here
    frame = cv2.GaussianBlur(frame1, (kernelSize,kernelSize), 0, 0)
    edge = cv2.Canny(frame,parameter1,parameter2,intApertureSize)  # Canny edge detection
    mask_edge = cv2.bitwise_not(edge)
    frame = cv2.bitwise_and(frame1,frame1,mask = mask_edge)

    # Display the resulting frame
    cv2.imshow('Super Impose',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):  # press q to quit
        break
# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```



In [51]:

```
img = io.imread('super impose.jpg')
fig = plt.imshow(img)
plt.axis('off')
plt.show()
```



Question-4:

First, we download the necessary packages for that part.

Then we use createsubtractBackgroundMOG2 method to subtract specific part of a video.

cv2.dilution method will dilute the white parts.

cv2.findContours method will find the contours or boundaries of white space.

In cv2.max_contours maximum value will be detected which forms the walking man.

cv2.boundingRect will find the edges of the rectangle around the man.

The rest of the code will generate the rectangle.

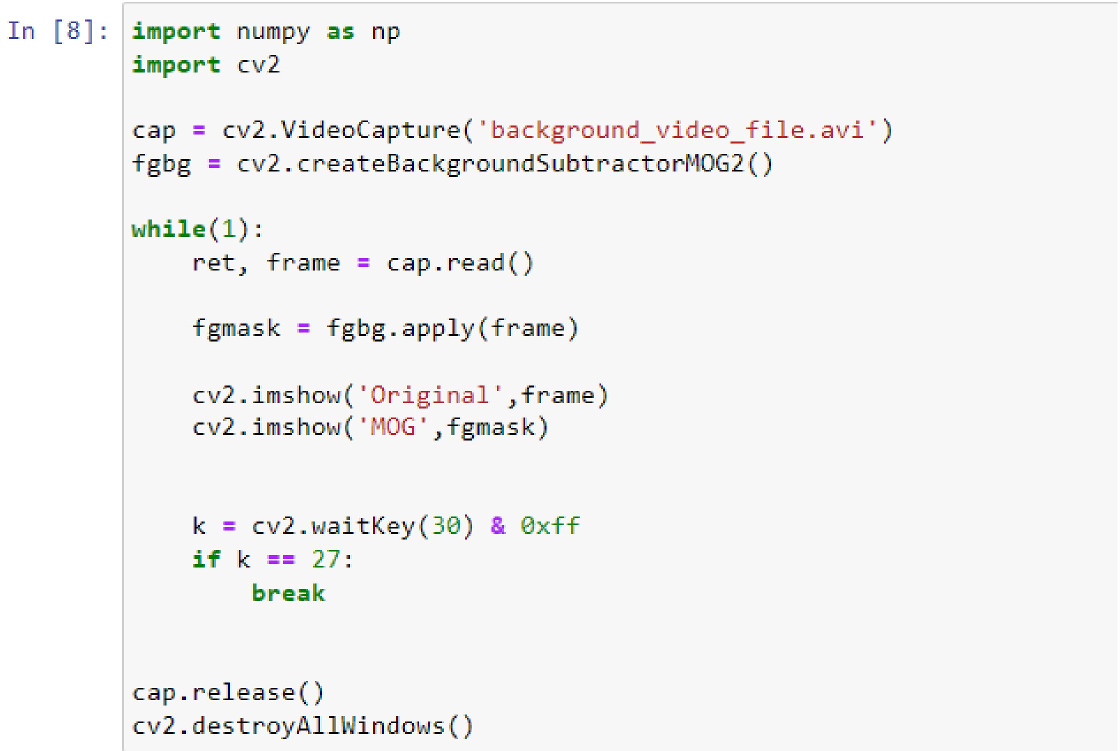
Question:5

MOG2 subtraction method

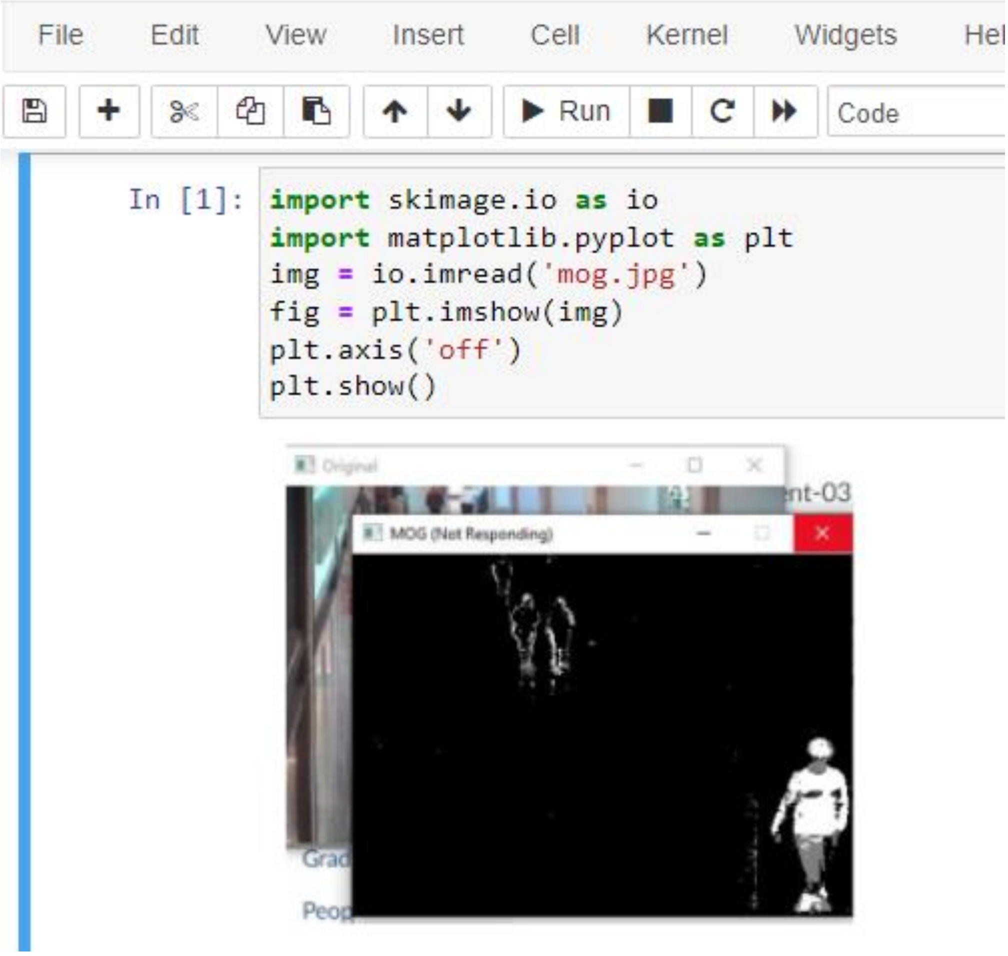
Background subtraction algorithm is used to distinguish between foreground and background objects in computer vision. There are

different approaches used for background subtraction and one of them is MOG2.This method works in OpenCV by having four inputs.

These inputs are source. learn rate, blur and threshold parameters.



```
In [8]: import numpy as np
        import cv2

        cap = cv2.VideoCapture('background_video_file.avi')
        fgbg = cv2.createBackgroundSubtractorMOG2()

        while(1):
            ret, frame = cap.read()

            fgmask = fgbg.apply(frame)

            cv2.imshow('Original',frame)
            cv2.imshow('MOG',fgmask)


            k = cv2.waitKey(30) & 0xff
            if k == 27:
                break


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



```
In [1]: import skimage.io as io
        import matplotlib.pyplot as plt
        img = io.imread('mog.jpg')
        fig = plt.imshow(img)
        plt.axis('off')
        plt.show()
```