In [5]:

```
import nltk
texts = "I am the only person who can run this program too much efficiently."
for text in texts:
    sentences = nltk.sent_tokenize(texts)
    for sentence in sentences:
        words = nltk.word_tokenize(sentence)
        tagged = nltk.pos_tag(words)
        print(tagged)
for text in texts:
    words=nltk.word_tokenize(texts)
    for word in words:
        word
```

```
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
```

```
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
```

```
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
```

```
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
[('I', 'PRP'), ('am', 'VBP'), ('the', 'DT'), ('only', 'JJ'), ('person', 'NN'), ('wh
o', 'WP'), ('can', 'MD'), ('run', 'VB'), ('this', 'DT'), ('program', 'NN'), ('too',
'RB'), ('much', 'JJ'), ('efficiently', 'RB'), ('.', '.')]
```

In [1]:

```
import cv2
import numpy as np
import math
import time
import datetime



cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    ret, img = cap.read()
    cv2.rectangle(img,(300,300),(100,100),(0,255,0),0)
    crop_img = img[100:300, 100:300]
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)
    _, thresh1 = cv2.threshold(blurred, 127, 255,cv2.THRESH_BINARY_INV+cv2.THRESH_OT
    cv2.imshow('Thresholded', thresh1)
    contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
            cv2.CHAIN_APPROX_NONE)
    max_area = -1
    for i in range(len(contours)):
        cnt=contours[i]
        area = cv2.contourArea(cnt)
        if(area>max_area):
            max_area=area
            ci=i
    cnt=contours[ci]
    x,y,w,h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img,(x,y),(x+w,y+h),(0,0,255),0)
    hull = cv2.convexHull(cnt)
    drawing = np.zeros(crop_img.shape,np.uint8)
    cv2.drawContours(drawing,[cnt],0,(0,255,0),0)
    cv2.drawContours(drawing,[hull],0,(0,0,255),0)
    hull = cv2.convexHull(cnt,returnPoints = False)
    defects = cv2.convexityDefects(cnt,hull)
    count_defects = 0
    cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)
    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        start = tuple(cnt[s][0])
        end = tuple(cnt[e][0])
        far = tuple(cnt[f][0])
        a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
        b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
        c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
        if angle <= 90:
            count_defects += 1
            cv2.circle(crop_img,far,1,[0,0,255],-1)
```

```
        dist = cv2.pointPolygonTest(cnt,far,True)
        cv2.line(crop_img,start,end,[0,255,0],2)
        cv2.circle(crop_img,far,5,[0,0,255],-1)
    if count_defects == 1:
        cv2.putText(img,"Fan speed 2", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    elif count_defects == 2:
        str = "Fan speed 3"
        cv2.putText(img, str, (5,50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
    elif count_defects == 3:
        cv2.putText(img,"Fan speed 4", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    elif count_defects == 4:
        cv2.putText(img,"fan speed 5", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    else:
        cv2.putText(img,"Fan speed 1", (50,50),\
                    cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    cv2.imshow('drawing', drawing)
    cv2.imshow('end', crop_img)
    cv2.imshow('Gesture', img)
    all_img = np.hstack((drawing, crop_img))
    cv2.imshow('Contours', all_img)
    k = cv2.waitKey(10)
    if k == 27:
        break
```

In [2]:

```
import cv2
import numpy as np
import math
import time
import datetime



cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    ret, img = cap.read()
    cv2.rectangle(img,(300,300),(100,100),(0,255,0),0)
    crop_img = img[100:300, 100:300]
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)
    _, thresh1 = cv2.threshold(blurred, 127, 255,cv2.THRESH_BINARY_INV+cv2.THRESH_OT
    cv2.imshow('Thresholded', thresh1)
    contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
            cv2.CHAIN_APPROX_NONE)
    max_area = -1
    for i in range(len(contours)):
        cnt=contours[i]
        area = cv2.contourArea(cnt)
        if(area>max_area):
            max_area=area
            ci=i
    cnt=contours[ci]
    x,y,w,h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img,(x,y),(x+w,y+h),(0,0,255),0)
    hull = cv2.convexHull(cnt)
    drawing = np.zeros(crop_img.shape,np.uint8)
    cv2.drawContours(drawing,[cnt],0,(0,255,0),0)
    cv2.drawContours(drawing,[hull],0,(0,0,255),0)
    hull = cv2.convexHull(cnt,returnPoints = False)
    defects = cv2.convexityDefects(cnt,hull)
```

```
    count_defects = 0
    cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)
    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        start = tuple(cnt[s][0])
        end = tuple(cnt[e][0])
        far = tuple(cnt[f][0])
        a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
        b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
        c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
        if angle <= 90:
            count_defects += 1
            cv2.circle(crop_img,far,1,[0,0,255],-1)
        dist = cv2.pointPolygonTest(cnt,far,True)
        cv2.line(crop_img,start,end,[0,255,0],2)
        cv2.circle(crop_img,far,5,[0,0,255],-1)
    if count_defects == 1:
        cv2.putText(img,"Fan speed 2", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    elif count_defects == 2:
        str = "Fan speed 3"
        cv2.putText(img, str, (5,50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
    elif count_defects == 3:
        cv2.putText(img,"Fan speed 4", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    elif count_defects == 4:
        cv2.putText(img,"fan speed 5", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    else:
        cv2.putText(img,"Fan speed 1", (50,50),\
                    cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    cv2.imshow('drawing', drawing)
    cv2.imshow('end', crop_img)
    cv2.imshow('Gesture', img)
    all_img = np.hstack((drawing, crop_img))
    cv2.imshow('Contours', all_img)
    k = cv2.waitKey(10)
    if k == 27:
        break
```

In [0]:

In [3]:

```
import cv2
import numpy as np
import math
import time
import datetime



cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    ret, img = cap.read()
    cv2.rectangle(img,(300,300),(100,100),(0,255,0),0)
    crop_img = img[100:300, 100:300]
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)
    _, thresh1 = cv2.threshold(blurred, 127, 255,cv2.THRESH_BINARY_INV+cv2.THRESH_OT
    cv2.imshow('Thresholded', thresh1)
    contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
```

```
            cv2.CHAIN_APPROX_NONE)
    max_area = -1
    for i in range(len(contours)):
        cnt=contours[i]
        area = cv2.contourArea(cnt)
        if(area>max_area):
            max_area=area
            ci=i
    cnt=contours[ci]
    x,y,w,h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img,(x,y),(x+w,y+h),(0,0,255),0)
    hull = cv2.convexHull(cnt)
    drawing = np.zeros(crop_img.shape,np.uint8)
    cv2.drawContours(drawing,[cnt],0,(0,255,0),0)
    cv2.drawContours(drawing,[hull],0,(0,0,255),0)
    hull = cv2.convexHull(cnt,returnPoints = False)
    defects = cv2.convexityDefects(cnt,hull)
    count_defects = 0
    cv2.drawContours(thresh1, contours, -1, (0,255,0), 3)
    for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0]
        start = tuple(cnt[s][0])
        end = tuple(cnt[e][0])
        far = tuple(cnt[f][0])
        a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
        b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
        c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
        angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
        if angle <= 90:
            count_defects += 1
            cv2.circle(crop_img,far,1,[0,0,255],-1)
        dist = cv2.pointPolygonTest(cnt,far,True)
        cv2.line(crop_img,start,end,[0,255,0],2)
        cv2.circle(crop_img,far,5,[0,0,255],-1)
    if count_defects == 1:
        cv2.putText(img,"Fan speed 2", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    elif count_defects == 2:
        str = "Fan speed 3"
        cv2.putText(img, str, (5,50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
    elif count_defects == 3:
        cv2.putText(img,"Fan speed 4", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    elif count_defects == 4:
        cv2.putText(img,"fan speed 5", (50,50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    else:
        cv2.putText(img,"Fan speed 1", (50,50),\
                    cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
    cv2.imshow('drawing', drawing)
    cv2.imshow('end', crop_img)
    cv2.imshow('Gesture', img)
    all_img = np.hstack((drawing, crop_img))
    cv2.imshow('Contours', all_img)
    k = cv2.waitKey(10)
    if k == 27:
        break
```

In [3]:

```
import networkx as nx
G = nx.Graph()
G.add_edge('A', 'B', weight=4)
G.add_edge('B', 'D', weight=2)
G.add_edge('A', 'C', weight=3)
```

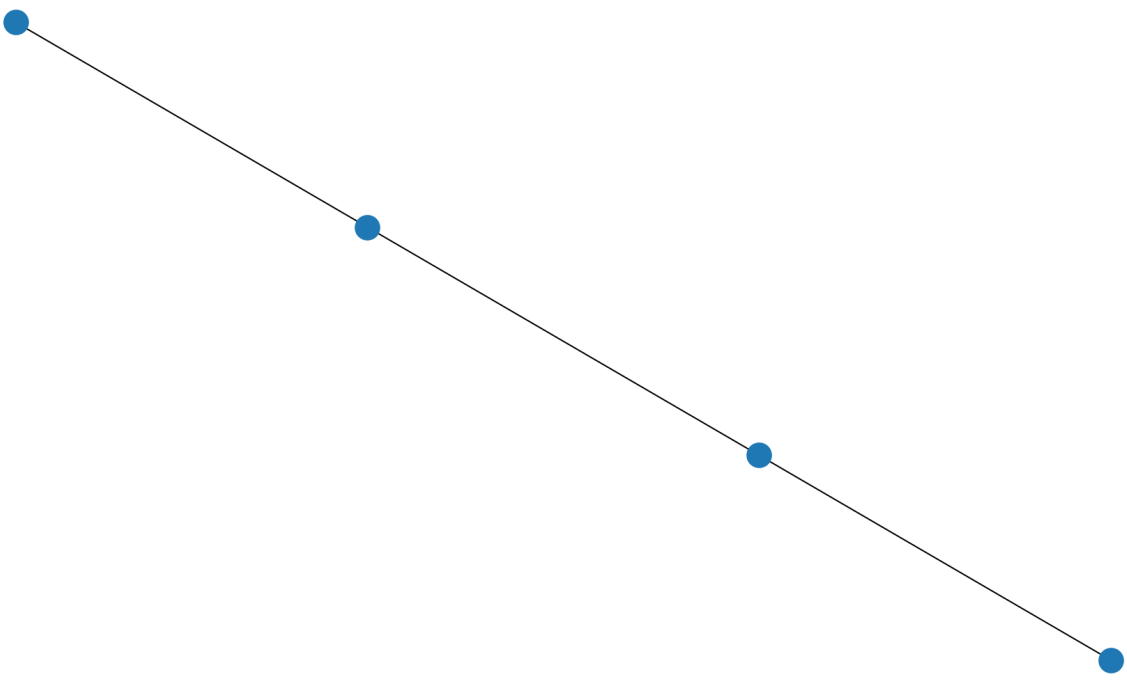
```
G.add_edge('C', 'D', weight=4)
nx.shortest_path(G, 'A', 'D', weight='weight')
```

Out[3]:
```
['A', 'B', 'D']
```

In [5]:
```
import networkx as nx
import matplotlib.pyplot as plt

G=nx.path_graph(4)
cities = {0:"Toronto",1:"London",2:"Berlin",3:"New York"}

H=nx.relabel_nodes(G,cities)

print("Nodes of graph: ")
print(H.nodes())
print("Edges of graph: ")
print(H.edges())
nx.draw(H)
plt.savefig("path_graph_cities.png")
plt.show()
```

```
Nodes of graph:
['Toronto', 'London', 'Berlin', 'New York']
Edges of graph:
[('Toronto', 'London'), ('London', 'Berlin'), ('Berlin', 'New York')]
```

Out[5]:



In [1]:
```
import csv
import networkx as nx
from operator import itemgetter
import community
# Read in the nodelist file
with open('quakers_nodelist.csv', 'r') as nodecsv:
 nodereader = csv.reader(nodecsv)
 nodes = [n for n in nodereader][1:]
# Get a list of just the node names (the first item in each row)
node_names = [n[0] for n in nodes]
# Read in the edgelist file
with open('quakers_edgelist.csv', 'r') as edgecsv:
```

```
 edgereader = csv.reader(edgecsv)
 edges = [tuple(e) for e in edgereader][1:]
# Print the number of nodes and edges in our two lists
print(len(node_names))
print(len(edges))
G = nx.Graph() # Initialize a Graph object
G.add_nodes_from(node_names) # Add nodes to the Graph
G.add_edges_from(edges) # Add edges to the Graph
print(nx.info(G)) # Print information about the Graph
hist_sig_dict = {}
gender_dict = {}
birth_dict = {}
death_dict = {}
id_dict = {}
for node in nodes: # Loop through the list, one row at a time
 hist_sig_dict[node[0]] = node[1]
 gender_dict[node[0]] = node[2]
 birth_dict[node[0]] = node[3]
 death_dict[node[0]] = node[4]
 id_dict[node[0]] = node[5]
nx.set_node_attributes(G, hist_sig_dict, 'historical_significance')
nx.set_node_attributes(G, gender_dict, 'gender')
nx.set_node_attributes(G, birth_dict, 'birth_year')
nx.set_node_attributes(G, death_dict, 'death_year')
nx.set_node_attributes(G, id_dict, 'sdfb_id')
for n in G.nodes(): # Loop through every node, in our data "n" will be the name of t
    print(n, G.node[n]['birth_year']) # Access every node by its name, and then by t
"birth_year"
density = nx.density(G)
print("Network density:", density)
fell_whitehead_path = nx.shortest_path(G, source="Margaret Fell", target="GeorgeWhit
print("Shortest path between Fell and Whitehead:", fell_whitehead_path)
print("Length of that path:", len(fell_whitehead_path)-1)
print(nx.is_connected(G))
# Next, use nx.connected_components to get the list of components,
# then use the max() command to find the largest one:
components = nx.connected_components(G)
largest_component = max(components, key=len)
# Create a "subgraph" of just the largest component
# Then calculate the diameter of the subgraph, just like you did with density.
#
subgraph = G.subgraph(largest_component)
diameter = nx.diameter(subgraph)
print("Network diameter of largest component:", diameter)
triadic_closure = nx.transitivity(G)
print("Triadic closure:", triadic_closure)
degree_dict = dict(G.degree(G.nodes()))
nx.set_node_attributes(G, degree_dict, 'degree')
print(G.node['William Penn'])
sorted_degree = sorted(degree_dict.items(), key=itemgetter(1), reverse=True)
print("Top 20 nodes by degree:")
for d in sorted_degree[:20]:
 print(d)
betweenness_dict = nx.betweenness_centrality(G) # Run betweenness centrality
eigenvector_dict = nx.eigenvector_centrality(G) # Run eigenvector centrality
# Assign each to an attribute in your network
nx.set_node_attributes(G, betweenness_dict, 'betweenness')
nx.set_node_attributes(G, eigenvector_dict, 'eigenvector')
sorted_betweenness = sorted(betweenness_dict.items(), key=itemgetter(1), reverse=Tru
sorted_betweenness1 = sorted(eigenvector_dict.items(), key=itemgetter(1), reverse=Tr
print("Top 20 nodes by betweenness centrality:")
```

```
for b in sorted_betweenness[:20]:
 print(b)
print("Top 20 nodes by eigenvector centrality:")
for b in sorted_betweenness1[:20]:
 print(b)
#First get the top 20 nodes by betweenness as a list
top_betweenness = sorted_betweenness[:20]
#First get the top 20 nodes by betweenness as a list
top_betweenness = sorted_betweenness1[:20]
#Then find and print their degree
for tb in top_betweenness: # Loop through top_betweenness
 degree = degree_dict[tb[0]] # Use degree_dict to access a node's degree, see footno
 print("Name:", tb[0], "| Betweenness Centrality:", tb[1], "| Degree:", degree)
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-1-713b5a3632dd> in <module>
      4 import community
      5 # Read in the nodelist file
----> 6 with open('quakers_nodelist.csv', 'r') as nodecsv:
      7  nodereader = csv.reader(nodecsv)
      8  nodes = [n for n in nodereader][1:]

FileNotFoundError: [Errno 2] No such file or directory: 'quakers_nodelist.csv'
```

In [0]: