0. Install and Import Dependencies

```
#!pip install easyocr
#!pip install imutils
```

```
import cv2
from matplotlib import pyplot as plt
import numpy as np
import imutils
import easyocr
```

1. Read in Image, Grayscale and Blur

```
def grey_scale_image(img):
  gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  return gray
```

```
gray = grey_scale_image(img)
plt.imshow(cv2.cvtColor(gray, cv2.COLOR_BGR2RGB))
```

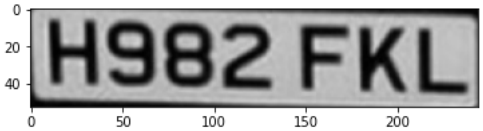
```
<matplotlib.image.AxesImage at 0x7f46d5645ca0>
```

2. Create a function that identify the target object in the image

```
def find_target_data_in_image(gray,img):
  bfilter = cv2.bilateralFilter(gray, 11, 17, 17) #Noise reduction
  edged = cv2.Canny(bfilter, 30, 200) #Edge detection
  keypoints = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
  contours = imutils.grab_contours(keypoints)
  contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]
  location = None
  for contour in contours:
      approx = cv2.approxPolyDP(contour, 10, True)
      if len(approx) == 4:
          location = approx
          break
  mask = np.zeros(gray.shape, np.uint8)
  new_image = cv2.drawContours(mask, [location], 0,255, -1)
  new_image = cv2.bitwise_and(img, img, mask=mask)
  #plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_BGR2RGB))
  (x,y) = np.where(mask==255)
  (x1, y1) = (np.min(x), np.min(y))
  (x2, y2) = (np.max(x), np.max(y))
  cropped_image = gray[x1:x2+1, y1:y2+1]
  return cropped_image
```

```
cropped_image = find_target_data_in_image(gray,img)
plt.imshow(cv2.cvtColor(cropped_image, cv2.COLOR_BGR2RGB))
```

```
<matplotlib.image.AxesImage at 0x7f46d48cc2b0>
```



3. create a function that identify the target data and impose in the image

```
def identify_target_data(cropped_image,img):
  reader = easyocr.Reader(['en'])
  result = reader.readtext(cropped_image)
  text = result[0][-2]
  font = cv2.FONT_HERSHEY_SIMPLEX
  res = cv2.putText(img, text=text, org=(approx[0][0][0], approx[1][0][1]+60), fontFace=font, fontScale=1, color=(0,255,0), thick
  res = cv2.rectangle(img, tuple(approx[0][0]), tuple(approx[2][0]), (0,255,0),3)
  return res
```

Final : Create the main function and leverage above created function to identify desire data

```
if __name__ == '__main__':

  img = cv2.imread('image4.jpg')

  gray = grey_scale_image(img)
  cropped_image = find_target_data_in_image(gray,img)
  res = identify_target_data(cropped_image,img)

  plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
```

```
WARNING:easyocr.easyocr:CUDA not available - defaulting to CPU. Note: This module is much f
```

Colab paid products - Cancel contracts here

0s completed at 9:45 PM