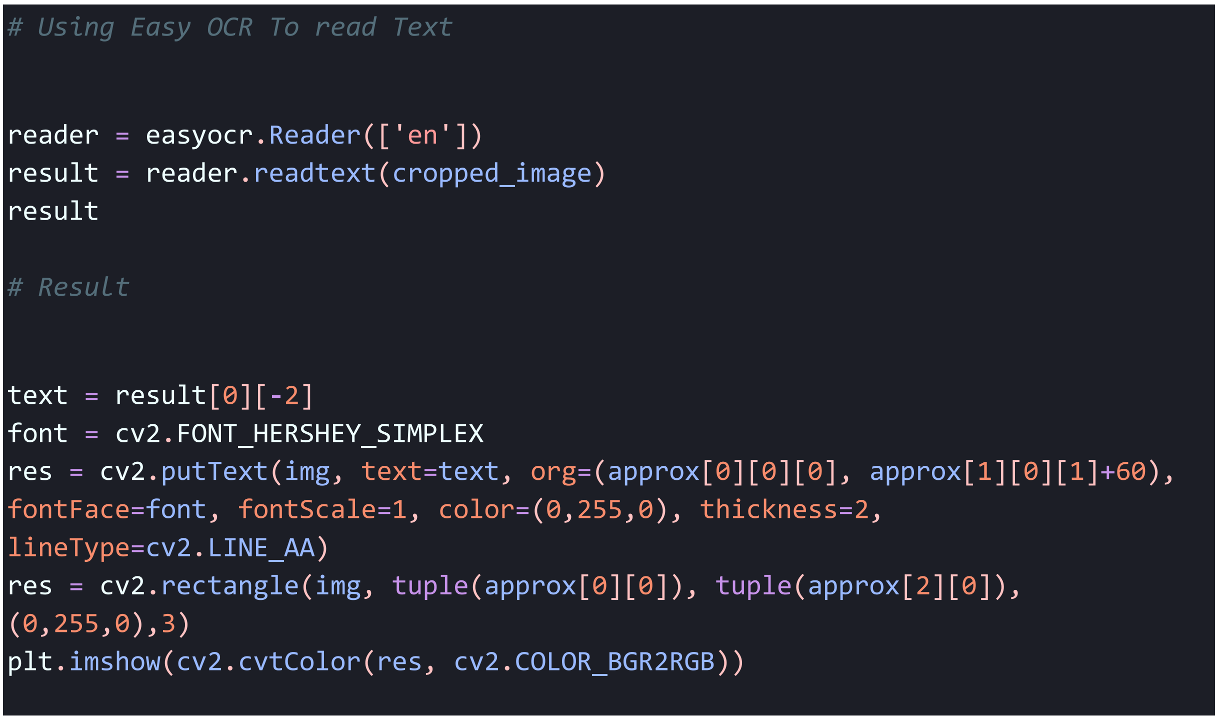
```python
import cv2
from matplotlib import pyplot as plt
import numpy as np
import easyocr
import imutils


# Read in Image, Grayscale and Blur

img = cv2.imread('/content/Project-No-35-Automatic-Number-Plate-
Recognition/image dataset/images (1).jpeg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(cv2.cvtColor(gray, cv2.COLOR_BGR2RGB))

# Apply filter and find edges for localization


bfilter = cv2.bilateralFilter(gray, 11, 17, 17) #Noise reduction
edged = cv2.Canny(bfilter, 30, 200) #Edge detection
plt.imshow(cv2.cvtColor(edged, cv2.COLOR_BGR2RGB))

# Find Contours and Apply Mask


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)


reader = easyocr.Reader(['en'])
result = reader.readtext(cropped_image)
result


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



```
# Using Easy OCR To read Text


reader = easyocr.Reader(['en'])
result = reader.readtext(cropped_image)
result

# Result


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), thickness=2,
lineType=cv2.LINE_AA)
res = cv2.rectangle(img, tuple(approx[0][0]), tuple(approx[2][0]),
(0,255,0),3)
plt.imshow(cv2.cvtColor(res, cv2.COLOR_BGR2RGB))
```

Code Explained:

The code you provided is an implementation of an Automatic Number Plate Recognition (ANPR) system using Python and various libraries. Let's go through the code step by step:

1. Importing Libraries:

- `cv2` is the OpenCV library for computer vision tasks.
- `pyplot` from `matplotlib` is used for displaying images and plots.
- `numpy` is a library for numerical computations.
- `easyocr` is a library for optical character recognition (OCR).
- `imutils` provides convenience functions for working with OpenCV.

2. Reading and Preprocessing the Image:

- The code reads an image from the file path specified in `img` using `cv2.imread`.
- The image is then converted to grayscale using `cv2.cvtColor`.
- The grayscale image is displayed using `pyplot.imshow`.

3. Applying Filter and Finding Edges:

- A bilateral filter is applied to the grayscale image to reduce noise using `cv2.bilateralFilter`.
- The Canny edge detection algorithm is applied to detect edges in the filtered image using `cv2.Canny`.

- The edged image is displayed using `pyplot.imshow`.

4. Finding Contours and Applying Mask:

   - Contours are found in the edged image using `cv2.findContours`.

   - The contours are sorted based on their area in descending order using `sorted` and `cv2.contourArea`.

   - The largest contour, assumed to be the license plate, is approximated using `cv2.approxPolyDP`.

   - A mask is created based on the location of the license plate using `cv2.drawContours` and `cv2.bitwise_and`.

   - The masked image is displayed using `pyplot.imshow`.

5. Reading Text from License Plate:

   - An OCR reader is initialized using `easyocr.Reader` with the language set to English.

   - The cropped license plate image is passed to the OCR reader using `reader.readtext`.

   - The recognized text is stored in the `result` variable.

6. Displaying the Result:

   - The original image is annotated with the recognized text and a bounding box around the license plate using `cv2.putText` and `cv2.rectangle`.

   - The annotated image is displayed using `pyplot.imshow`.

Overall, this code performs license plate localization, character recognition using OCR, and displays the result on the original image.