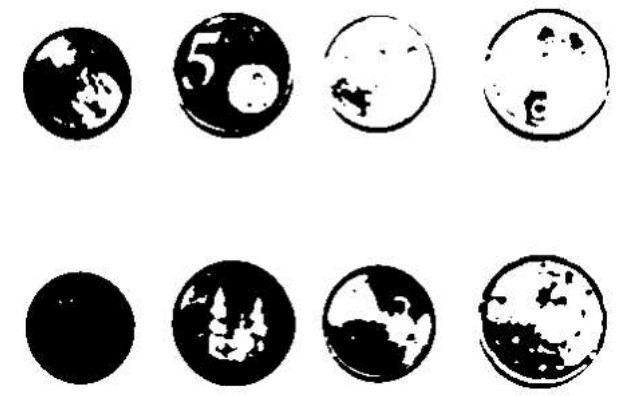
```
# 1-2. Sorunun cevabı
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Resmi BGR formatında okuma (para.jpg)
bgr_image = cv2.imread("para.jpg")

# BGR formatından RGB formatina
rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)

# RGB formatındaki resmi imshow ile gösterme
plt.imshow(rgb_image)
plt.axis("off")
plt.show()
```

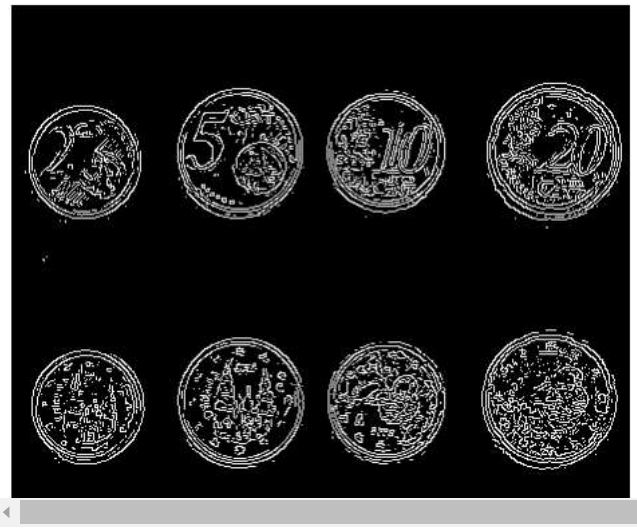
```
# 3.Sorunun cevabı
gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY)

# Gri tonlamalı resmi ekranda gösterme
plt.imshow(gray_image, cmap="gray")
plt.axis("off")  # Eksenleri kapatma (isteğe bağlı)
plt.show()
```

```
# 4.Sorunun cevabı
# Gauss Bulanıklaştırma uygulama
blurred_image = cv2.GaussianBlur(gray_image, (13, 13), 0)

# Bulanık resmi ekranda gösterme
plt.imshow(blurred_image, cmap="gray")
plt.axis("off")  # Eksenleri kapatma (isteğe bağlı)
plt.show()
```

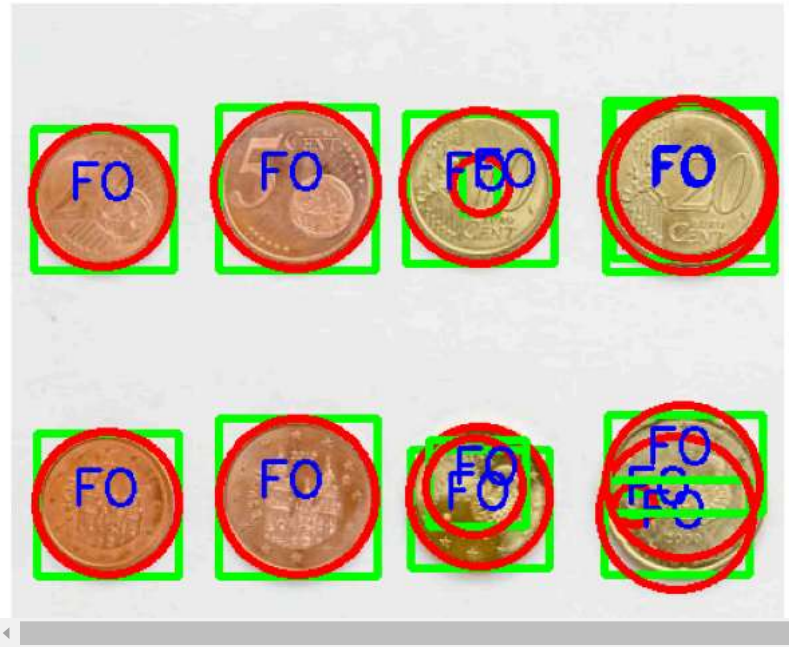
```
# 5. Sorunun cevabı
# Gauss Bulanıklaştırma uygulama (kernel boyutunu biraz küçültüyoruz)
blurred_image = cv2.GaussianBlur(gray_image, (7, 7), 0)

# Eşikleme (Threshold) uygulama (eşik değerini 100-130 arasında ayarladım)
_, binary_image = cv2.threshold(blurred_image, 160, 255, cv2.THRESH_BINARY)

# Binary resmi ekranda gösterme
plt.imshow(binary_image, cmap="gray")
plt.axis("off")  # Eksenleri kapatma (isteğe bağlı)
plt.show()
```



```
# 6. Sorunun cevabı
# Canny Kenar Bulma işlemi
edge_image = cv2.Canny(gray_image, 95, 95)

# Kenarların bulunduğu resim
plt.imshow(edge_image, cmap="gray")
plt.axis("off")
plt.show()
```



```
# 7. Sorunun cevabı
contours, _ = cv2.findContours(edges_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
output_image = rgb_image.copy()

for contour in contours:
    if cv2.contourArea(contour) > 67:
        # Bounding box hesaplama
        x, y, width, height = cv2.boundingRect(contour)
        cv2.rectangle(output_image, (x, y), (x + width, y + height), (0, 255, 0), 3)
        # Daire çizme
        (center_x, center_y), radius = cv2.minEnclosingCircle(contour)
        center = (int(center_x), int(center_y))
        radius = int(radius)
        cv2.circle(output_image, center, radius, (255, 0, 0), 3)

        # "FO" metni yazma
        text_pos_x, text_pos_y = x + width // 4, y + height // 2
        cv2.putText(output_image, "FO", (text_pos_x, text_pos_y), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

# Sonucu gösterme
plt.figure(figsize=(10, 6))
plt.imshow(output_image)
plt.axis("off")
plt.show()
```




```
# Bonus Soru 8. Sorunun cevabı

# Canny kenar tespiti
edges_image = cv2.Canny(gray_image, 100, 200)

# Kontur tespiti
contours, _ = cv2.findContours(edges_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

```
# Konturların çizileceği kopya görüntü
output_image = rgb_image.copy()

# Konturları alanlarına göre sırala (büyükten küçüğe)
sorted_contours = sorted(contours, key=cv2.contourArea, reverse=True)

# En büyük ve en küçük paraları bulma
largest_contour = sorted_contours[0]
smallest_contour = sorted_contours[-1]

# En büyük paranın içine "8" yazma
x, y, w, h = cv2.boundingRect(largest_contour)
text_pos_x, text_pos_y = x + w // 2, y + h // 2
cv2.putText(output_image, "8", (text_pos_x, text_pos_y), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

# En küçük paranın içine "1" yazma
x, y, w, h = cv2.boundingRect(smallest_contour)
text_pos_x, text_pos_y = x + w // 2, y + h // 2
cv2.putText(output_image, "1", (text_pos_x, text_pos_y), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

# Sonucu gösterme
plt.figure(figsize=(10, 6))
plt.imshow(output_image)
plt.axis("off")
plt.show()
```