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

CHUYỂN ĐỔI GIỮA CÁC KHÔNG GIAN MÀU

Hàm cv2.cvtColor() cho phép thực hiện chuyển đổi giữa các không gian màu.
Có hơn 150 phương thức chuyển đổi không gian màu dựng sẵn trong openCV. Chúng ta sẽ làm quen với một số phương thức chuyển đổi không gian màu sau đây.

❖ Cú pháp chung:

cv2.cvtColor(src, code[, dst[, dstCn]]) Trong đó:

src: ảnh đầu vào, toàn bộ không gian màu của ảnh sẽ thay đổi

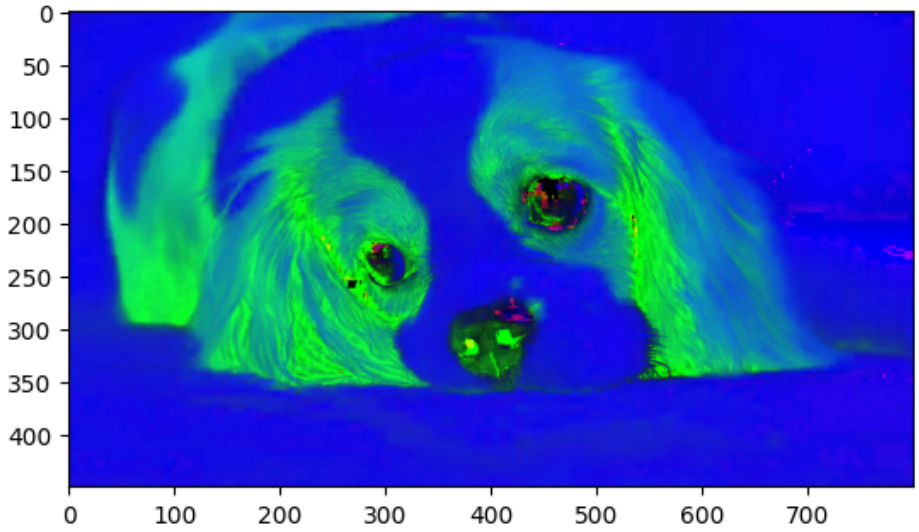
code: mã không gian màu sẽ chuyển

dst: ảnh đầu ra có cùng kích thước và độ sâu bit với ảnh đầu vào, (tùy chọn).

dstCn: số lượng kênh trong ảnh đầu ra. Nếu tham số có giá trị 0 thì số lượng kênh được lấy tự động từ src và code, (tùy chọn)

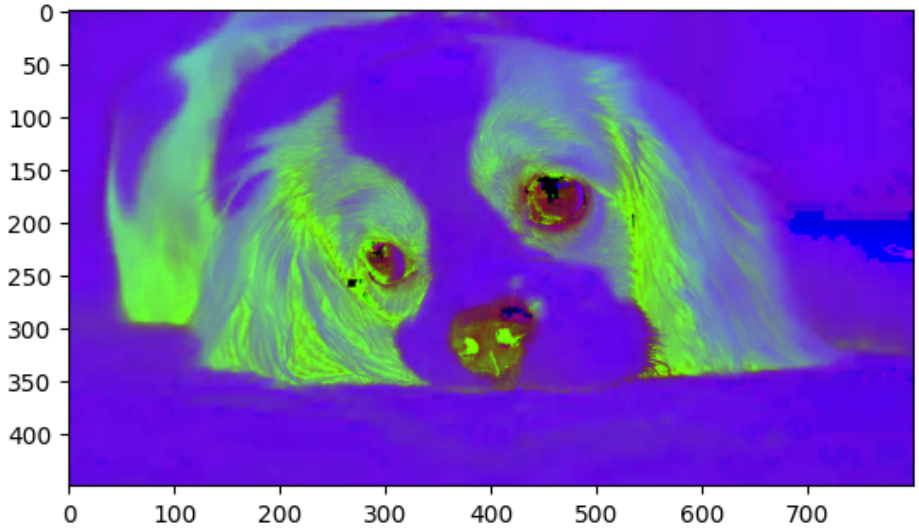
In [ ]:
```
img = cv2.imread('hinh2.jpg')
hsv_img= cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
plt.imshow(hsv_img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x24a7b80a170>
```



In [ ]:
```
hsv_img2= cv2.cvtColor(img,cv2.COLOR_RGB2HSV)
plt.imshow(hsv_img2)
```

Out[ ]: <matplotlib.image.AxesImage at 0x24a7bd9c880>



2.2. Hãy phân tích và giải thích nguyên nhân sự khác biệt giữa 2 kết quả chuyển đổi không gian màu trên? Đâu sẽ là ảnh kết quả đúng trong không gian màu HSV?

-->Ảnh hsv_img thì do là chuyển từ hệ BGR sang HSV còn hsv_img2 là từ RGB sang HSV,
kết quả đúng sẽ là hsv_img2 do thứ tự sắp xếp khác nhau 2.3. Đề xuất hướng giải quyết cho phương pháp chuyển đổi không gian màu còn lại để thu được ảnh đúng trong không gian màu HSV? (Lưu ý, Câu lệnh 1(2) không thay đổi)
-->Đối với ảnh màu, trong OpenCV, không gian màu được hiểu mặc định là BGR.
Do đó, để chuyển đổi đúng sang không gian màu HSV, cần sử dụng COLOR_BGR2HSV thay vì COLOR_RGB2HSV.

TRỘN VÀ DÁN ẢNH (BLENDING AND PASTING) – CÁC PHÉP TOÁN SỐ HỌC TRÊN ẢNH

Trộn ảnh với hàm cv2.addWeighted() Ý tưởng của trộn ảnh chỉ đơn giản là thực hiện thêm trọng số cho mỗi pixel trên cả 2 ảnh và kết hợp chúng lại với nhau. Để trộn ảnh, chúng ta sử dụng công thức đơn giản sau: $new\_pixel = \boldsymbol{\alpha} \times pixel\_1 + \boldsymbol{\beta} \times pixel\_2 + \boldsymbol{\gamma}$

In [ ]:
```
img1 = cv2.imread('dog_backpack.png')
img2 = cv2.imread('watermark_no_copy.png')
img1= cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
img2= cv2.cvtColor(img2,cv2.COLOR_BGR2RGB)
```

In [ ]:
```
# import matplotlib.pyplot as plt
# import matplotlib.image as mpimg
# Tạo một figure với hai subplots
plt.figure(figsize=(10, 5))

# Hiển thị ảnh thứ nhất ở subplot thứ nhất với tiêu đề
plt.subplot(1, 2, 1)
plt.imshow(img1)
plt.axis('off')  # Ẩn trục
plt.title('Ảnh 1')  # Đặt tiêu đề cho ảnh thứ nhất

# Hiển thị ảnh thứ hai ở subplot thứ hai với tiêu đề
plt.subplot(1, 2, 2)
plt.imshow(img2)
plt.axis('off')  # Ẩn trục
plt.title('Ảnh 2')  # Đặt tiêu đề cho ảnh thứ hai

# Hiển thị plot
plt.show()
```

Ảnh 1

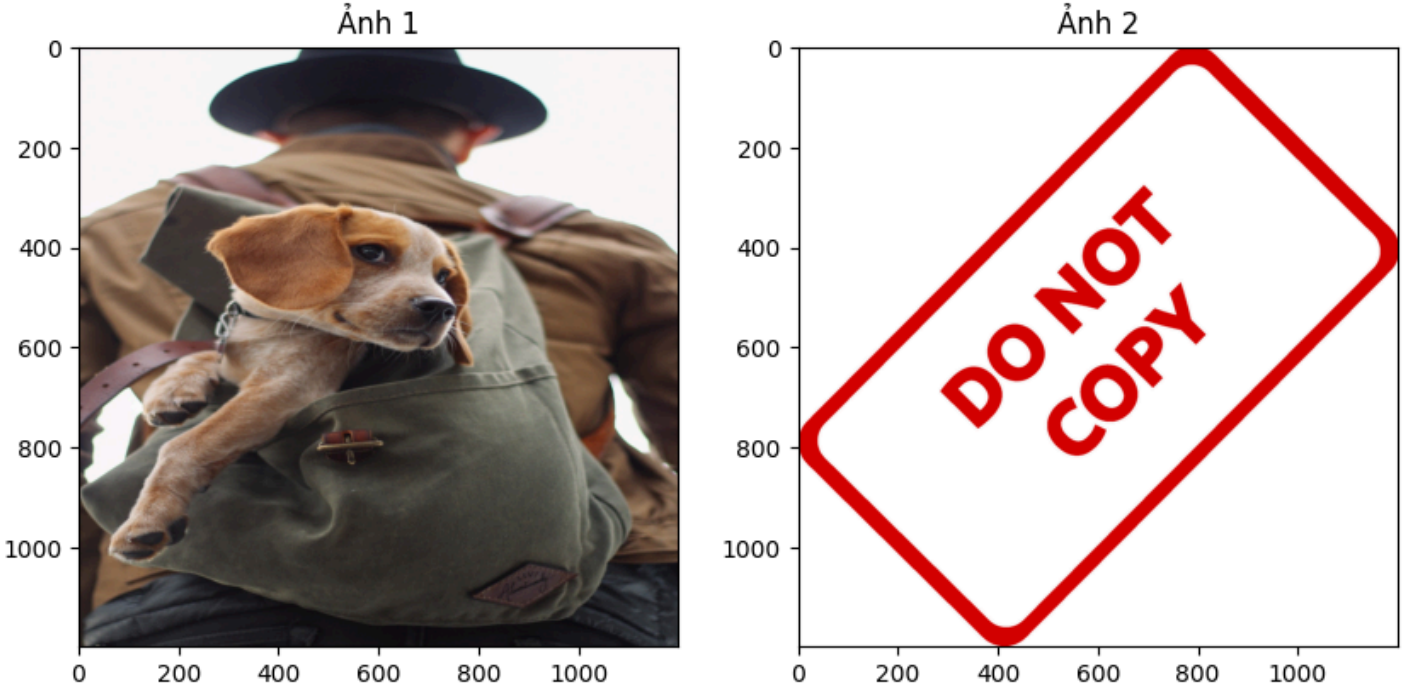
Ảnh 2

In [ ]:
```
img1= cv2.resize(img1,(1200,1200))
img2= cv2.resize(img2,(1200,1200))
plt.figure(figsize=(10, 5))

# Hiển thị ảnh thứ nhất ở subplot thứ nhất với tiêu đề
plt.subplot(1, 2, 1)
plt.imshow(img1)
plt.title('Ảnh 1')  # Đặt tiêu đề cho ảnh thứ nhất

# Hiển thị ảnh thứ hai ở subplot thứ hai với tiêu đề
plt.subplot(1, 2, 2)
plt.imshow(img2)
plt.title('Ảnh 2')  # Đặt tiêu đề cho ảnh thứ hai

# Hiển thị plot
plt.show()
```

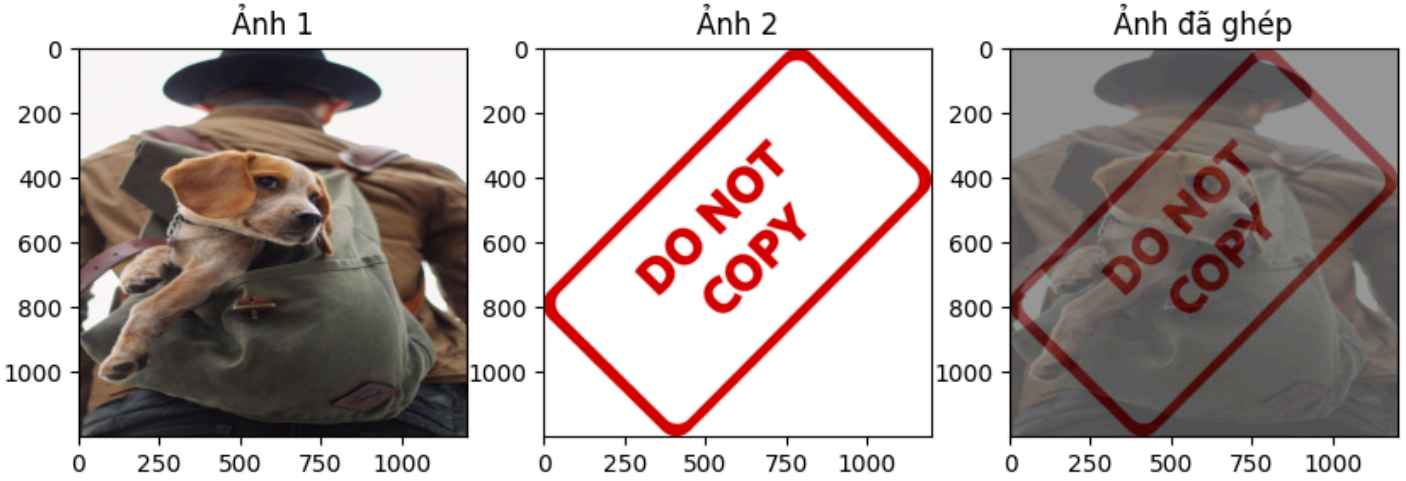


In [ ]:
```
blended = cv2.addWeighted(src1=img1,alpha=0.3,src2=img2,beta=0.3,gamma=0)
plt.figure(figsize=(10, 5))

# Hiển thị ảnh thứ nhất ở subplot thứ nhất với tiêu đề
plt.subplot(1, 3, 1)
plt.imshow(img1)
plt.title('Ảnh 1')  # Đặt tiêu đề cho ảnh thứ nhất

# Hiển thị ảnh thứ hai ở subplot thứ hai với tiêu đề
plt.subplot(1, 3, 2)
plt.imshow(img2)
plt.title('Ảnh 2')  # Đặt tiêu đề cho ảnh thứ hai

plt.subplot(1, 3, 3)
plt.imshow(blended)
plt.title('Ảnh đã ghép')
# Hiển thị plot
plt.show()
```



Xếp chồng các hình ảnh khác kích thước lên nhau

- Đọc lại 2 ảnh
- Thay đổi kích thước ảnh img2 về kích thước 600 × 600
- Xử lý chuyển đổi không gian màu sang RGB
- Gán ảnh large_img (ảnh lớn) bằng ảnh img1 và ảnh small_img (ảnh nhỏ) bằng ảnh img2

In [ ]:
```
# Đọc ảnh từ tệp
img1 = cv2.imread('dog_backpack.png')  # Thay thế bằng đường dẫn đến ảnh thứ nhất
img2 = cv2.imread('watermark_no_copy.png')  # Thay thế bằng đường dẫn đến ảnh thứ hai

# Thay đổi kích thước ảnh thứ hai
img2 = cv2.resize(img2, (600, 600))

# Chuyển đổi màu từ BGR sang RGB để hiển thị đúng màu sắc với matplotlib
img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)

# Thiết lập ảnh lớn và ảnh nhỏ
large_img = img1
small_img = img2

# Xác định vị trí để đặt ảnh nhỏ lên ảnh lớn
x_offset = 0
y_offset = 180
```

```
# Chèn ảnh nhỏ vào ảnh lớn tại vị trí đã xác định
large_img[y_offset:y_offset + small_img.shape[0], x_offset:x_offset + small_img.shape[1]] = small_img

# Hiển thị ảnh kết quả
plt.imshow(large_img)
plt.axis('off')  # Ẩn trục
plt.title('Ảnh Kết Hợp')  # Đặt tiêu đề cho ảnh kết hợp
plt.show()
```

Trộn ảnh với kích thước khác nhau

1/ Tạo vùng quan tâm (Region of Interest - ROI)

In [ ]:
```
# Đọc ảnh từ tệp
img1 = cv2.imread('dog_backpack.png')  # Thay thế bằng đường dẫn đến ảnh thứ nhất
img2 = cv2.imread('watermark_no_copy.png')  # Thay thế bằng đường dẫn đến ảnh thứ hai
img2=cv2.resize(img2,(600,600))
img1= cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
img2= cv2.cvtColor(img2,cv2.COLOR_BGR2RGB)
```

In [ ]:
```
img1.shape
```

Out[ ]:
```
(1401, 934, 3)
```

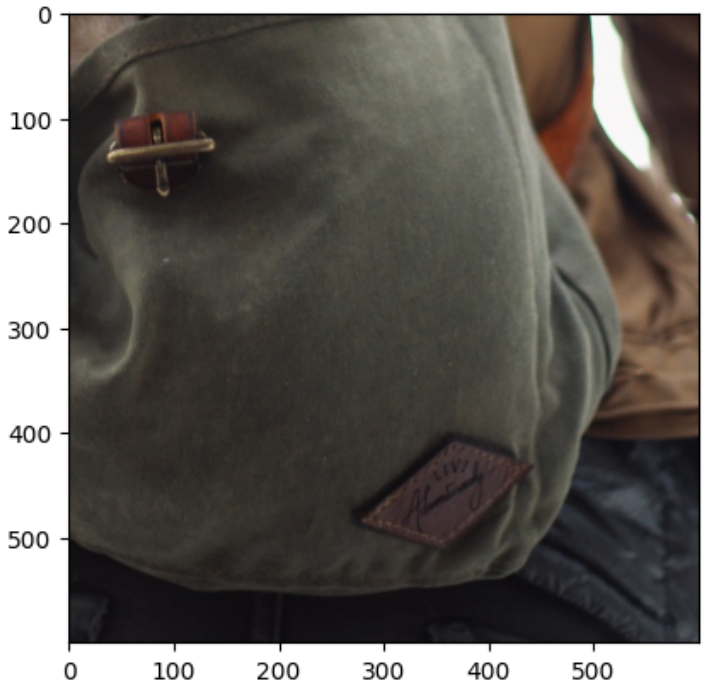
In [ ]:
```
# Thiết lập vị trí góc trên bên trái của ROI:
#Tạo vùng quan tâm có kích thước bằng kích thước của ảnh foreground (img2, ảnh có kích thước nhỏ hơn sẽ nằm ở phía trên ảnh lớn hơn)
x_offset = 934-600
y_offset = 1401-600
rows, cols, channels = img2.shape
roi = img1[y_offset:1401,x_offset:934 ]
```

In [ ]:
```
roi.shape
```

Out[ ]:
```
(600, 600, 3)
```

In [ ]:
```
plt.imshow(roi)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x24a00c235b0>
```



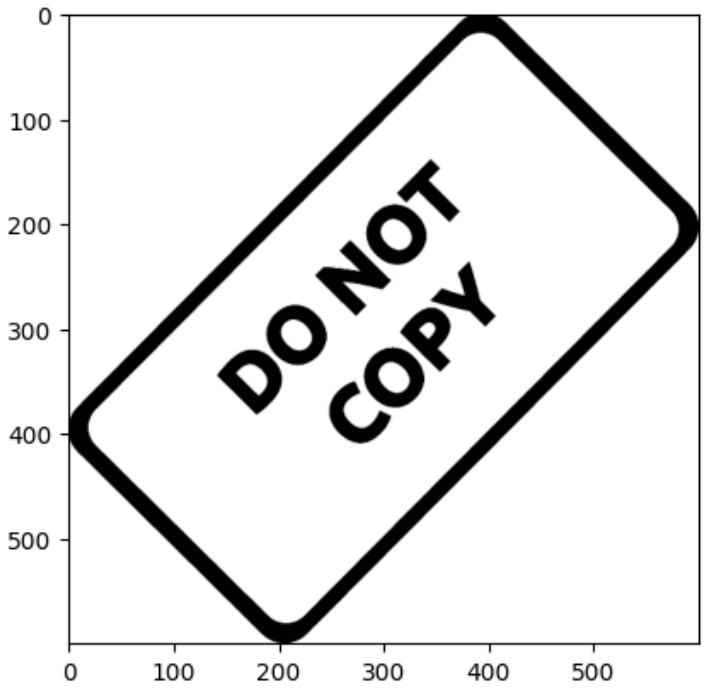
2/ Tạo mặt nạ (Mask)

In [ ]:
```
#Tạo một mặt nạ của logo và tạo mặt nạ bù của nó- Tạo măt nạ là ảnh đơn sắc từ ảnh img2
img2gray = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
img2gray.shape
```

Out[ ]:
```
(600, 600)
```

In [ ]:
```
plt.imshow(img2gray,cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x24a0125b640>
```

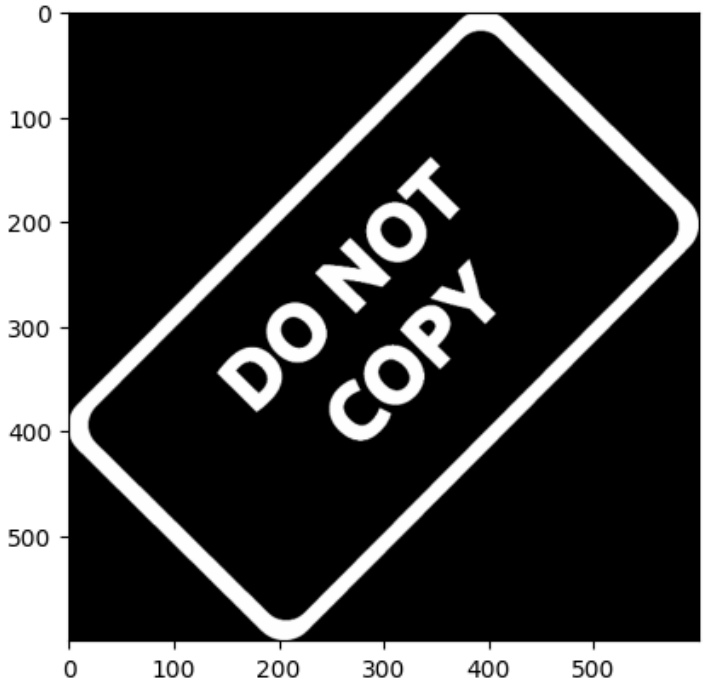


In [ ]:
```
mask_inv = cv2.bitwise_not(img2gray)
mask_inv.shape
```

Out[ ]:
```
(600, 600)
```

In [ ]:
```
plt.imshow(mask_inv,cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x24a012525f0>
```



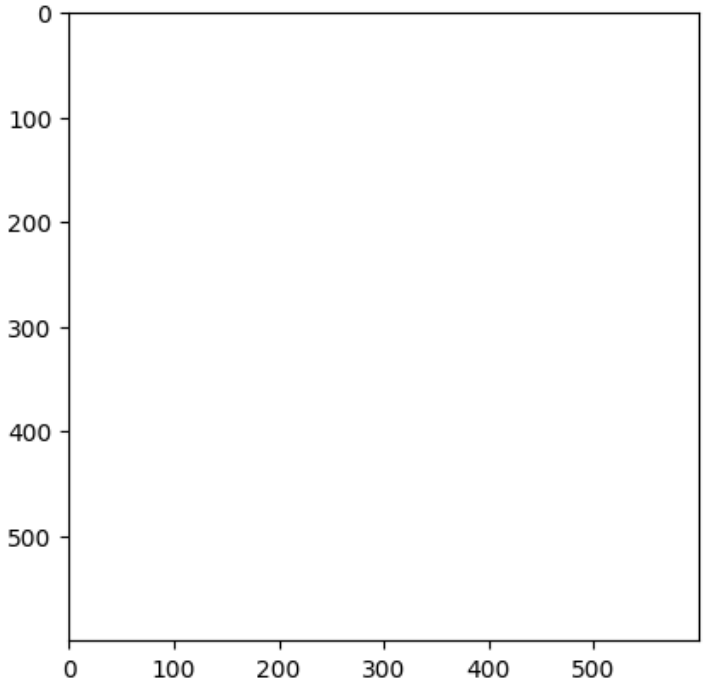
3/ Chuyển mặt nạ về ảnh có ba kênh

In [ ]:
```
white_bg = np.full(img2.shape, 255, dtype=np.uint8 )
white_bg.shape
```

Out[ ]:
```
(600, 600, 3)
```

In [ ]:
```
plt.imshow(white_bg)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x24a016bf550>
```



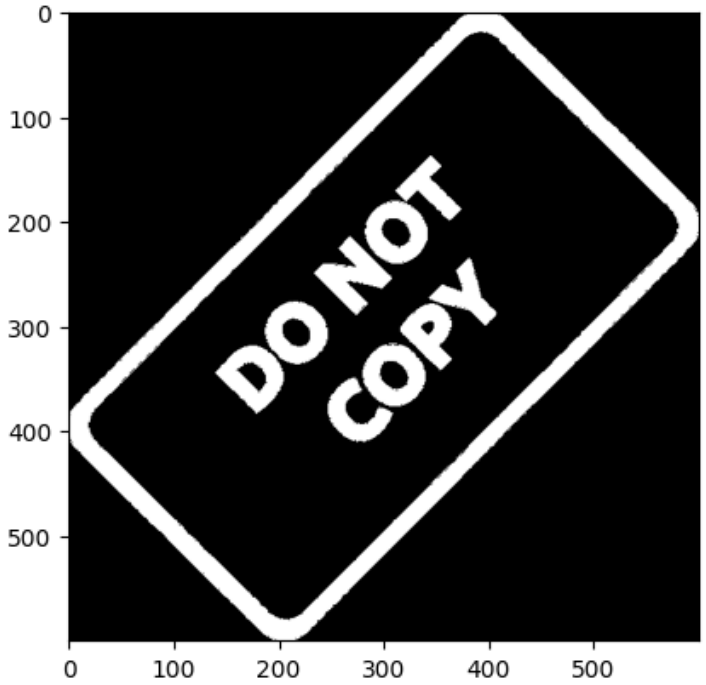
In [ ]:
```
# Công việc tiếp theo là sử dụng hàm cv2,bitwise_or() với mặt nạ sử dụng là
# mask_inv để giữ lại những thành phần của ảnh mặt nạ mà chúng ta mong muốn
# (các đường nét của chữ và đường viền) trên nền trắng đã tạo.

bk = cv2.bitwise_or(white_bg, white_bg, mask=mask_inv)
bk.shape
```

Out[ ]:
```
(600, 600, 3)
```

In [ ]:
```
plt.imshow(bk)
```

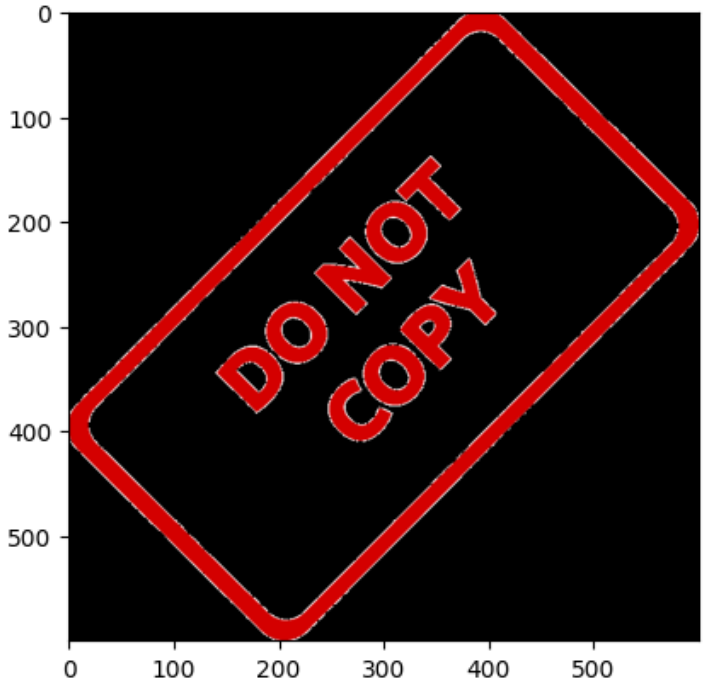
Out[ ]:
```
<matplotlib.image.AxesImage at 0x24a0172e200>
```



4 /Đặt ảnh foreground gốc lên trên mặt nạ

In [ ]:
```
#- Hiển thị lại ảnh bù của mặt nạ:
# Phân tách thành phần của ảnh img2 bằng mặt nạ mask_inv
fg= cv2.bitwise_or(img2, img2, mask=mask_inv)
plt.imshow(fg,cmap= 'gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x24a020565f0>
```



5/ Lấy ROI và trộn mặt nạ với ROI

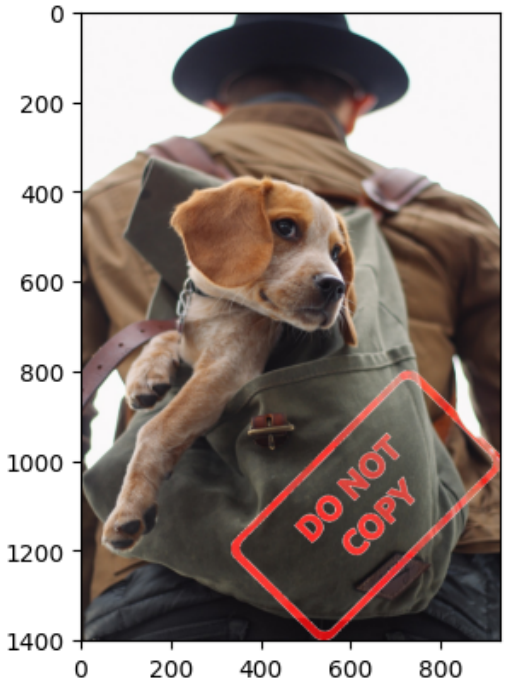
In [ ]:
```
final_roi = cv2.bitwise_or(roi,fg)
plt.imshow(final_roi,cmap= 'gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x24a02075c90>
```

6/ Thêm ROI vào lại phần còn lại của ảnh ban đầu

In [ ]:
```
large_img = img1
small_img = final_roi
large_img[y_offset:y_offset + small_img.shape[0], x_offset:x_offset + small_img.shape[1]] = small_img
plt.imshow(large_img)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x24a02133430>
```


0
200
400
600
800
1000
1200
1400
0
200
400
600
800
DO NOT
COPY