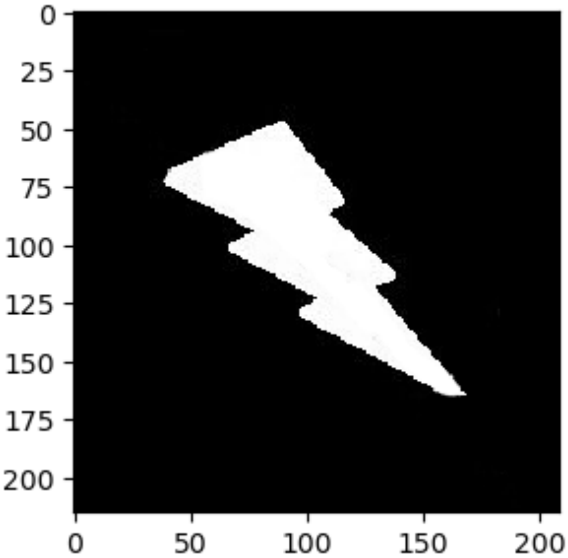
In [3]:
```
%matplotlib inline
from __future__ import print_function
# import ganymede
# ganymede.configure('uav.beaver.works')
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
```

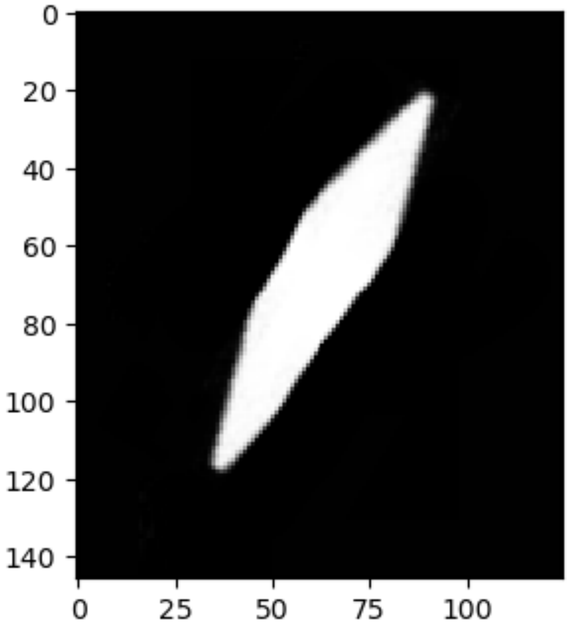
In [4]:
```
def check(p): pass
check(0)
```

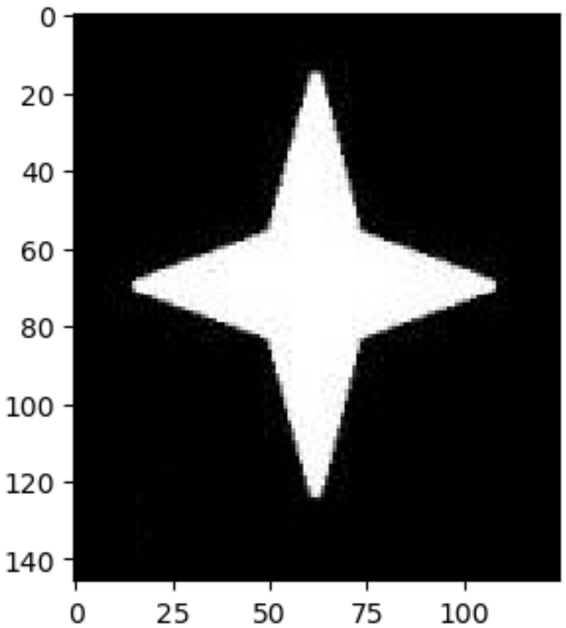
Note

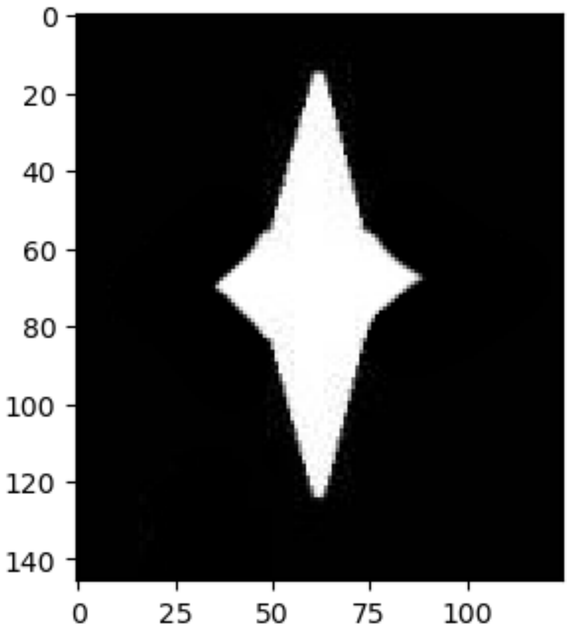
`cv2.imshow()` will not work in a notebook, even though the OpenCV tutorials use it. Instead, use `plt.imshow` and family to visualize your results.

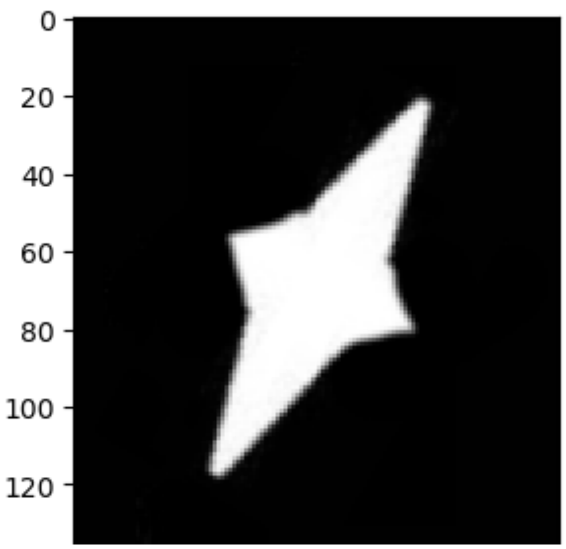
In [5]:
```
lightningbolt      = cv2.imread('shapes/lightningbolt.png', cv2.IMREAD_GR
blob               = cv2.imread('shapes/blob.png', cv2.IMREAD_GRAYSCALE)
star               = cv2.imread('shapes/star.png', cv2.IMREAD_GRAYSCALE)
squishedstar       = cv2.imread('shapes/squishedstar.png', cv2.IMREAD_GRA
squishedturnedstar = cv2.imread('shapes/squishedturnedstar.png', cv2.IMRE
letterj            = cv2.imread('shapes/letterj.png', cv2.IMREAD_GRAYSCAL

images = [lightningbolt, blob, star, squishedstar, squishedturnedstar, le

fig,ax = plt.subplots(nrows=3, ncols=2)
for a,i in zip(ax.flatten(), images):
    a.imshow(i, cmap='gray', interpolation='none');
fig.set_size_inches(7,14);
```

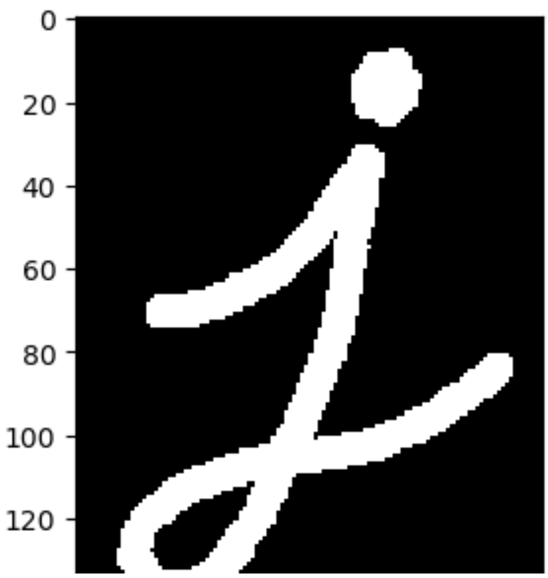

0
25
50
75
100
125
150
175
200
0 50 100 150 200

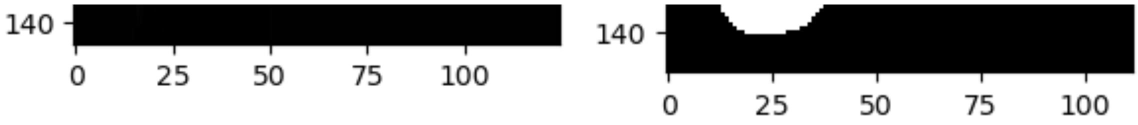

0
20
40
60
80
100
120
140
0 25 50 75 100


0
20
40
60
80
100
120
140
0 25 50 75 100


0
20
40
60
80
100
120
140
0 25 50 75 100


0
20
40
60
80
100
120


0
20
40
60
80
100
120



In [6]:
```
intensity_values = set(lightningbolt.flatten())
print(len(intensity_values))
```

75

Question:

What would you expect the value to be, visually? What explains the actual value?

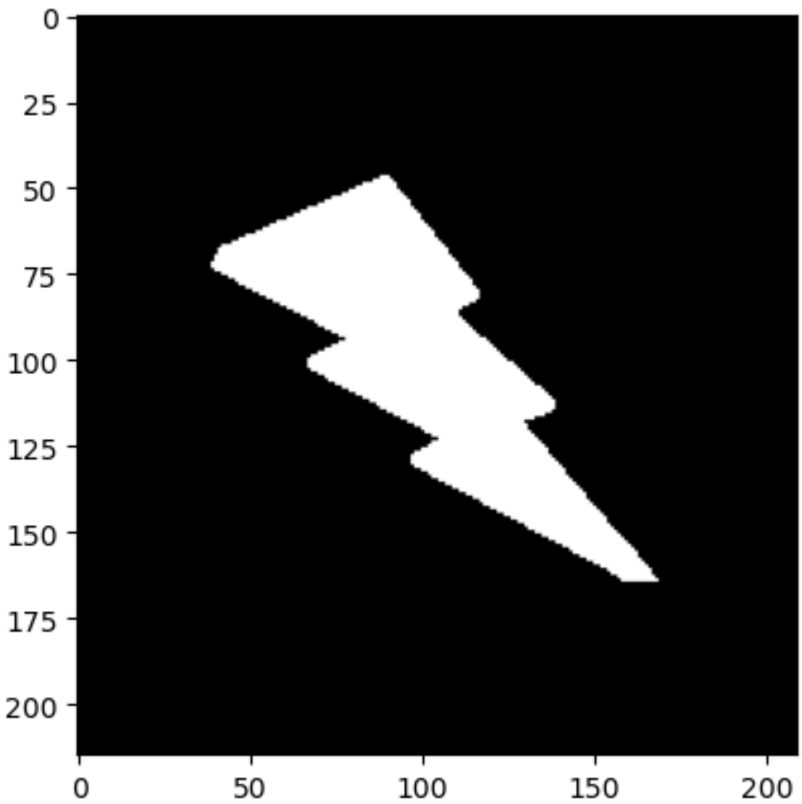
In [7]:
```
# TODO
# Your Answer

# Visually, I expect the answer to be around 60. The actual value might b
```

Thresholding

https://docs.opencv.org/3.4.1/d7/d4d/tutorial_py_thresholding.html

In [8]:
```
_, lightningbolt = cv2.threshold(lightningbolt,100,255,cv2.THRESH_BINARY)

intensity_values = set(lightningbolt.flatten())
print(len(intensity_values))

plt.imshow(lightningbolt, cmap='gray');
```

2



Question

What happens when the above values are used for thresholding? What is a "good" value for thresholding the above images? Why?

In [9]:
```
## TODO
## Your answer

# some pixels are assumed to be white when it should be black, so a bette
```

Exercises

Steps

1. Read each tutorial
   - Skim all parts of each tutorial to understand what each operation does
   - Focus on the part you will need for the requested transformation
2. Apply the transformation and visualize it

1. Blend lightningbolt and blob together

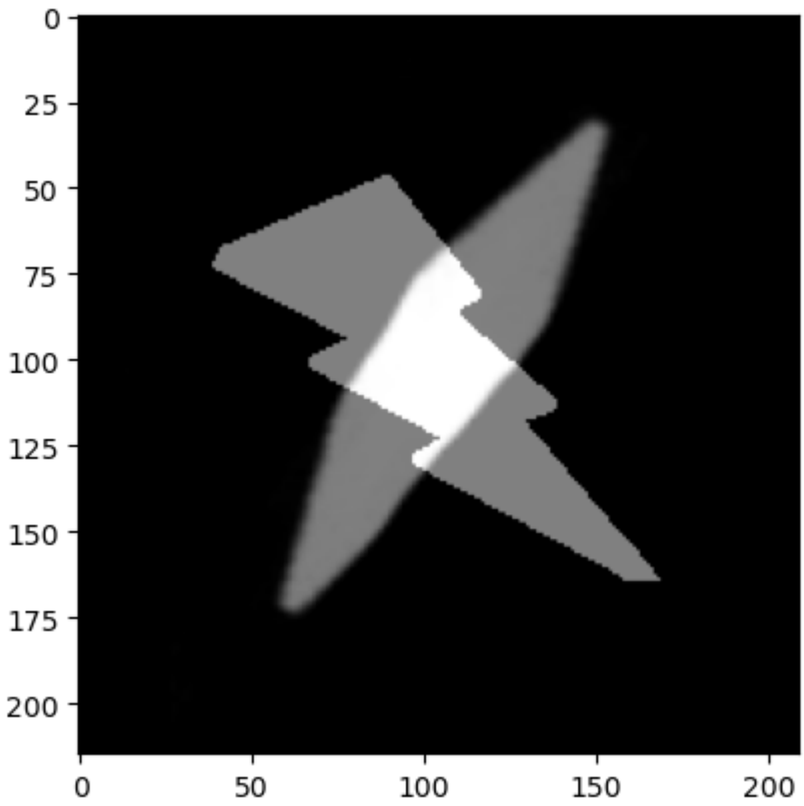
https://docs.opencv.org/3.4.1/d0/d86/tutorial_py_image_arithmetics.html

Remember: Don't use `imshow` from OpenCV, use `imshow` from `matplotlib`

In [10]:
```
# 1. Blend
# TODO

print(lightningbolt.shape)
blob_2 = cv2.resize(blob,(209,215))
dst = cv2.addWeighted(blob_2,0.5,lightningbolt, 0.5,0)
plt.imshow(dst, cmap='gray')
```

```
(215, 209)
```

Out[10]: <matplotlib.image.AxesImage at 0x7aafb4ffbd00>

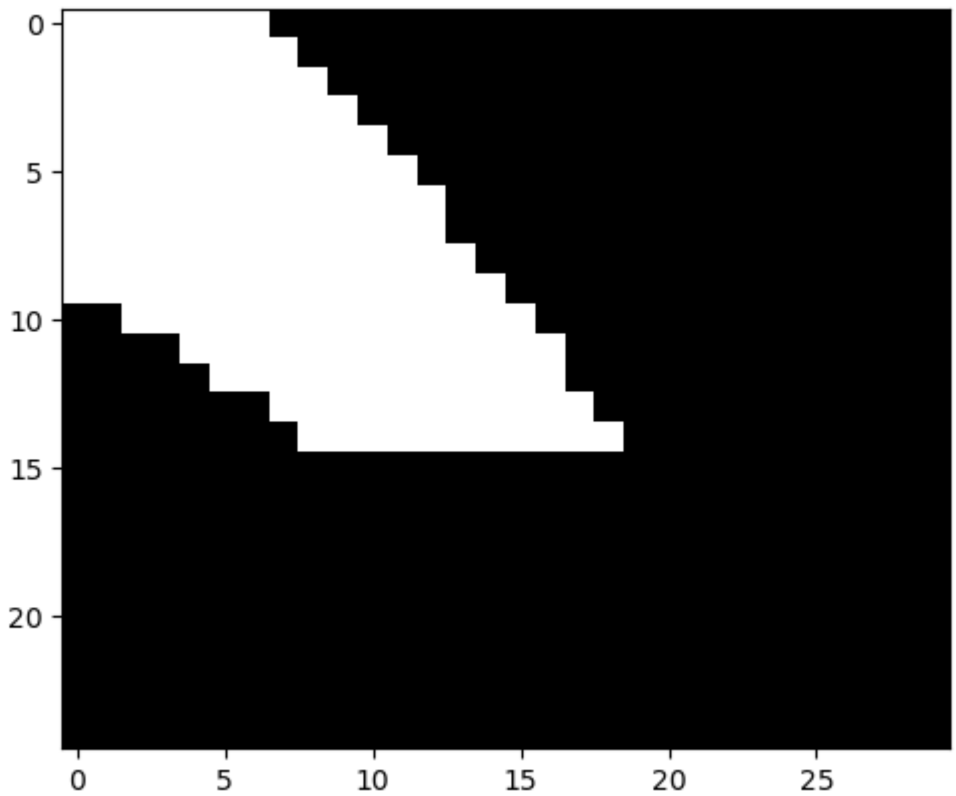


2. Find a ROI which contains the point of the lightning bolt

https://docs.opencv.org/3.4.1/d3/df2/tutorial_py_basic_ops.html

In [11]:
```
# 2. ROI
# TODO

light = lightningbolt[150:175, 150:180]
plt.imshow(light, cmap='gray')
```

Out[11]: <matplotlib.image.AxesImage at 0x7aafb505f730>

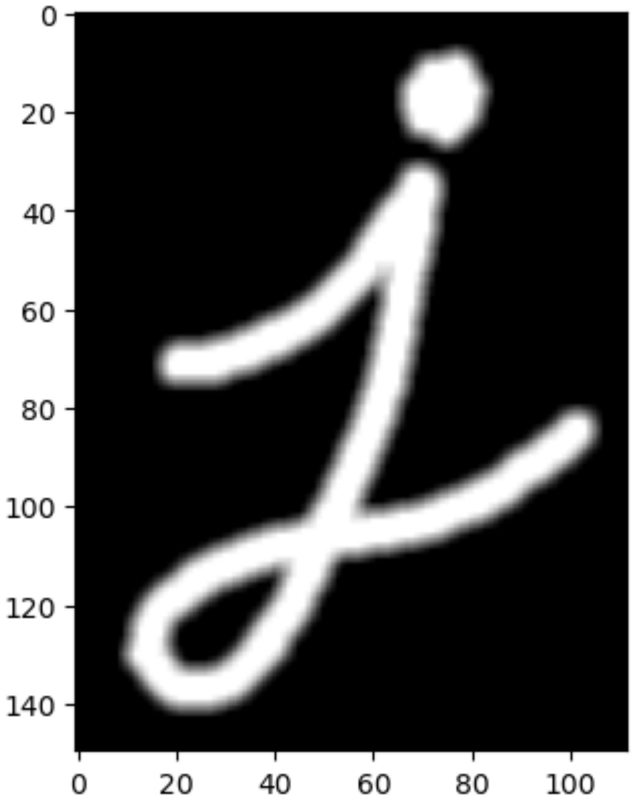


3. Use an averaging kernel on the letter j

https://docs.opencv.org/3.4.1/d4/d13/tutorial_py_filtering.html

In [12]:
```
# 3.
# TODO
blur = cv2.blur(letterj,(4,4))
plt.imshow(blur, cmap = 'gray')
```

Out[12]: <matplotlib.image.AxesImage at 0x7aafb4ed9e40>



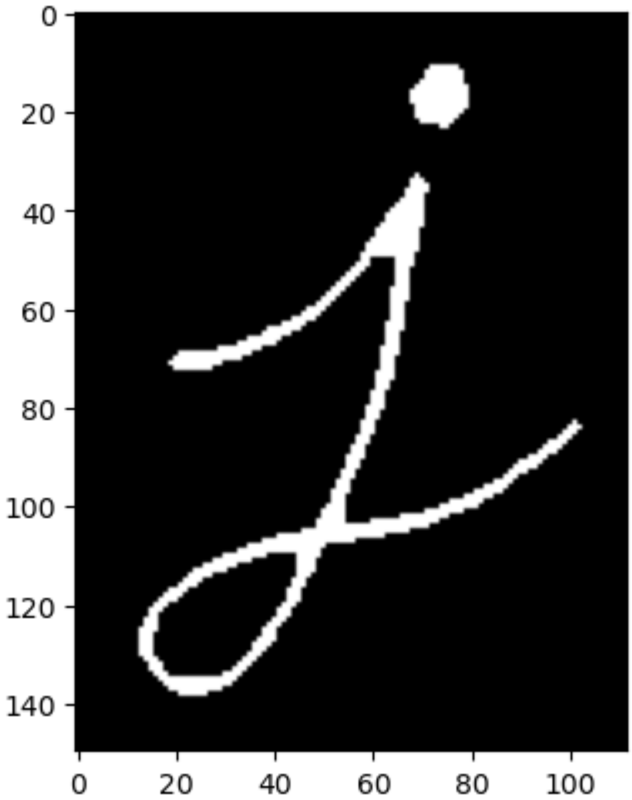
Morphology

https://docs.opencv.org/3.4.1/d9/d61/tutorial_py_morphological_ops.html

4. Perform erosion on j with a 3x3 kernel

In [13]:
```
# 4
# TODO

kernel = np.ones((3,3),np.uint8)
erosion = cv2.erode(letterj,kernel,iterations = 2)
plt.imshow(erosion, cmap = 'gray')
```

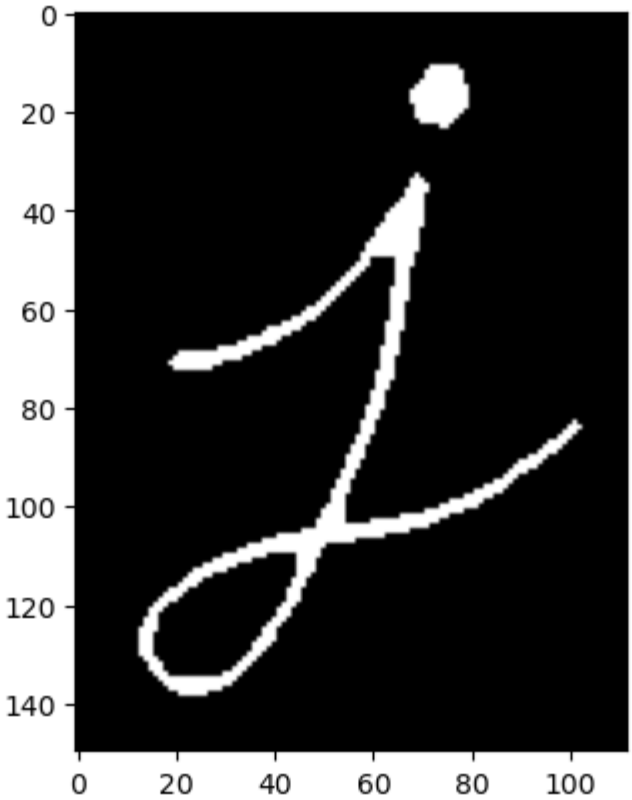
Out[13]: <matplotlib.image.AxesImage at 0x7aafb4f415a0>



5. Perform erosion on j with a 5x5 kernel

In [14]:
```
# 5
# TODO

kernel = np.ones((5,5),np.uint8)
erosion = cv2.erode(letterj,kernel,iterations = 1)
plt.imshow(erosion, cmap = 'gray')
```

Out[14]: <matplotlib.image.AxesImage at 0x7aafb4fb13c0>



6. Perform erosion on j with **two** iterations, using a kernel size of your choice

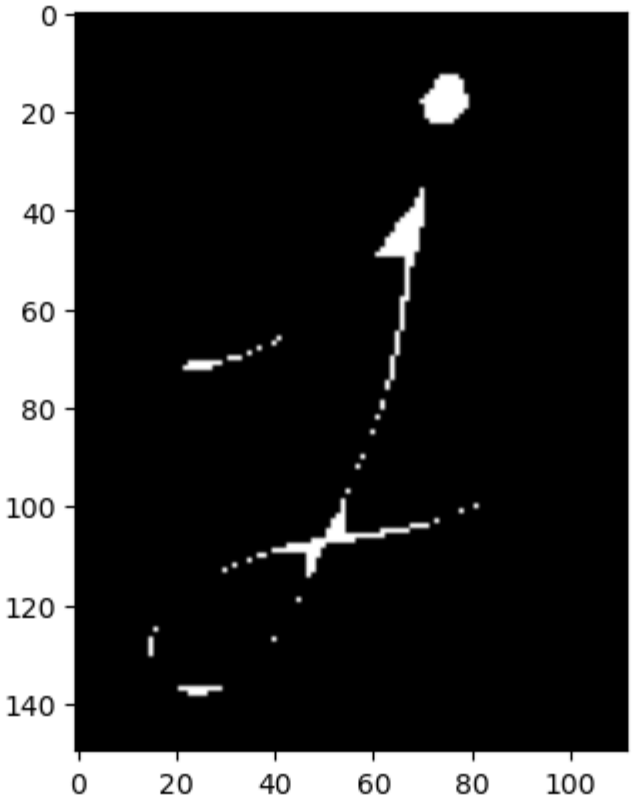
Hint: look at the OpenCV API documentation. It is possible to perform two iterations of erosion in one line of Python!

https://docs.opencv.org/3.4.1/d4/d86/group__imgproc__filter.html#gaeb1e0c1033e3f6b891a25d0511362aeb

In [15]:

```
# 6
# TODO
kernel = np.ones((4,4),np.uint8)
erosion = cv2.erode(letterj,kernel,iterations = 2)
plt.imshow(erosion, cmap = 'gray')
```

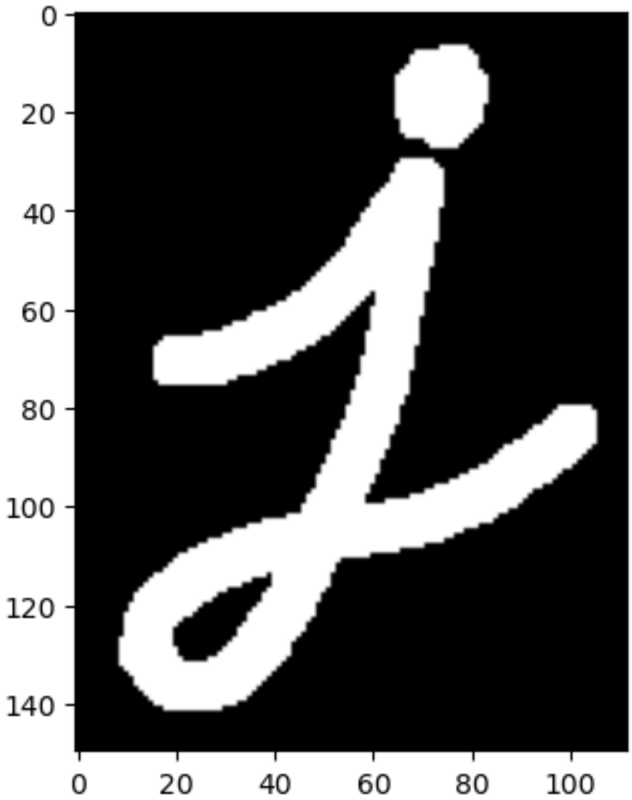
Out[15]: <matplotlib.image.AxesImage at 0x7aafb4e18970>



7. Perform dilation on j with a 3x3 kernel

In [16]:
```
# 7
# TODO
kernel = np.ones((3,3),np.uint8)
dilation = cv2.dilate(letterj,kernel,iterations = 1)
plt.imshow(dilation, cmap = 'gray')
```

Out[16]: <matplotlib.image.AxesImage at 0x7aafb4e84880>



8. Perform dilation on j with a 5x5 kernel

In [17]:
```
# 8
# TODO

kernel = np.ones((5,5),np.uint8)
dilation = cv2.dilate(letterj,kernel,iterations = 1)
plt.imshow(dilation, cmap = 'gray')
```

Out[17]: <matplotlib.image.AxesImage at 0x7aafb4cf21d0>

9. What is the effect of kernel size on morphology operations?

In [18]:
```
# 9
# TODO

# The greater the kernel, the greater the effect of the morphology
```

10. What is the difference betweeen repeated iterations of a morphology operation with a small kernel, versus a single iteration with a large kernel?

In [19]:
```
# 10
# TODO

# They can be the same.
```

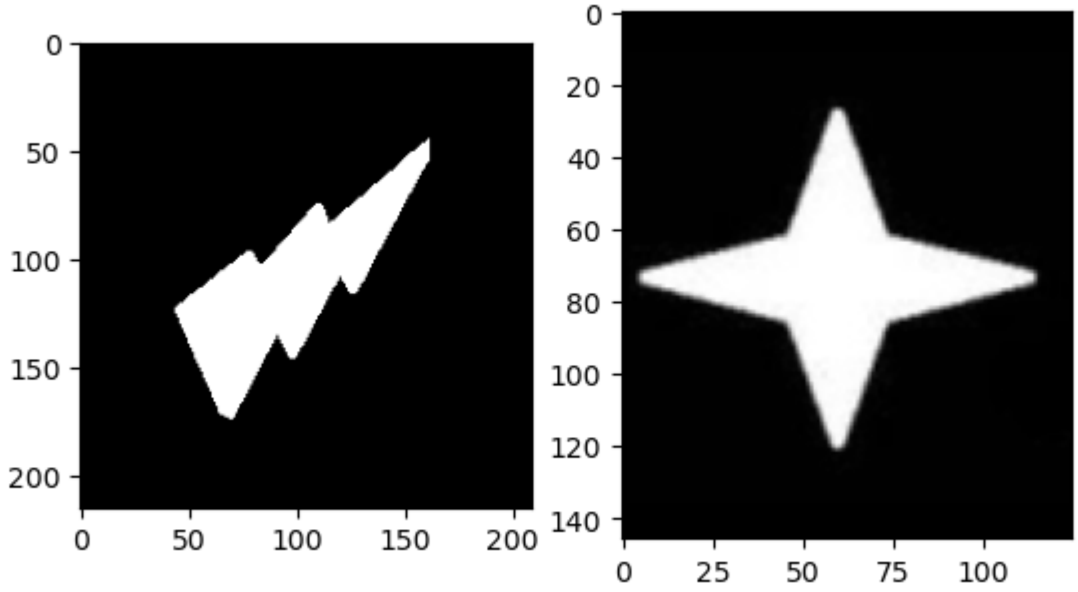
11. Rotate the lightningbolt and star by 90 degrees

https://docs.opencv.org/3.4.1/da/d6e/tutorial_py_geometric_transformations.html

In [41]:
```
# 11
# TODO

rows,cols = lightningbolt.shape
```

```
rows2,cols2 = star.shape
L = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
S = cv2.getRotationMatrix2D((cols2/2,rows2/2),90,1)

dst1 = cv2.warpAffine(lightningbolt,L,(cols,rows))
dst2 = cv2.warpAffine(star,S,(cols2,rows2))

fig, ax = plt.subplots(ncols=2, nrows=1)

ax[0].imshow(dst1,cmap='gray')
ax[1].imshow(dst2,cmap='gray')
```

Out[41]: <matplotlib.image.AxesImage at 0x7aafae7d35b0>



12. STRETCH GOAL:

Visualize the result of Laplacian, Sobel X, and Sobel Y on all of the images. Also, produce a combined image of both Sobel X and Sobel Y for each image. Is Exercise 1 the best way to do this? Are there other options?

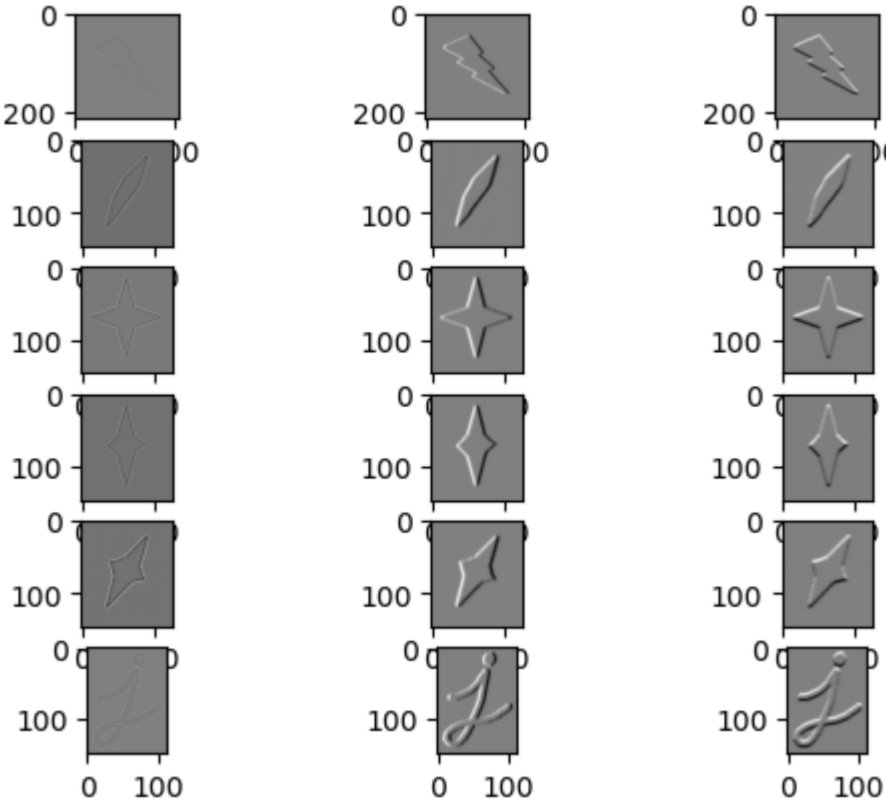
You should have 4 outputs (Laplacian, SobelX, SobelY, and the combination) for each input image visualized at the end.

https://docs.opencv.org/3.4.1/d5/d0f/tutorial_py_gradients.html

In [21]:
```
# 12
# TODO

fig, ax = plt.subplots(ncols=3, nrows=6)

#lightning bolt

light_laplacian = cv2.Laplacian(lightningbolt,cv2.CV_64F)
light_sobelx = cv2.Sobel(lightningbolt,cv2.CV_64F,1,0,ksize=7)
light_sobely = cv2.Sobel(lightningbolt,cv2.CV_64F,0,1,ksize=7)

ax[0,0].imshow(light_laplacian, cmap='gray')
ax[0,1].imshow(light_sobelx, cmap='gray')
```

```
ax[0,2].imshow(light_sobely, cmap='gray')

#blob

blob_laplacian = cv2.Laplacian(blob,cv2.CV_64F)
blob_sobelx = cv2.Sobel(blob,cv2.CV_64F,1,0,ksize=7)
blob_sobely = cv2.Sobel(blob,cv2.CV_64F,0,1,ksize=7)

ax[1,0].imshow(blob_laplacian, cmap='gray')
ax[1,1].imshow(blob_sobelx, cmap='gray')
ax[1,2].imshow(blob_sobely, cmap='gray')

#star

star_laplacian = cv2.Laplacian(star,cv2.CV_64F)
star_sobelx = cv2.Sobel(star,cv2.CV_64F,1,0,ksize=7)
star_sobely = cv2.Sobel(star,cv2.CV_64F,0,1,ksize=7)

ax[2,0].imshow(star_laplacian, cmap='gray')
ax[2,1].imshow(star_sobelx, cmap='gray')
ax[2,2].imshow(star_sobely, cmap='gray')

#squished star

ss_laplacian = cv2.Laplacian(squishedstar,cv2.CV_64F)
ss_sobelx = cv2.Sobel(squishedstar,cv2.CV_64F,1,0,ksize=7)
ss_sobely = cv2.Sobel(squishedstar,cv2.CV_64F,0,1,ksize=7)

ax[3,0].imshow(ss_laplacian, cmap='gray')
ax[3,1].imshow(ss_sobelx, cmap='gray')
ax[3,2].imshow(ss_sobely, cmap='gray')

#squished turned star

sts_laplacian = cv2.Laplacian(squishedturnedstar,cv2.CV_64F)
sts_sobelx = cv2.Sobel(squishedturnedstar,cv2.CV_64F,1,0,ksize=7)
sts_sobely = cv2.Sobel(squishedturnedstar,cv2.CV_64F,0,1,ksize=7)

ax[4,0].imshow(sts_laplacian, cmap='gray')
ax[4,1].imshow(sts_sobelx, cmap='gray')
ax[4,2].imshow(sts_sobely, cmap='gray')

#letter j

j_laplacian = cv2.Laplacian(letterj,cv2.CV_64F)
j_sobelx = cv2.Sobel(letterj,cv2.CV_64F,1,0,ksize=7)
j_sobely = cv2.Sobel(letterj,cv2.CV_64F,0,1,ksize=7)

ax[5,0].imshow(j_laplacian, cmap='gray')
ax[5,1].imshow(j_sobelx, cmap='gray')
ax[5,2].imshow(j_sobely, cmap='gray')
```

```
Out[21]: <matplotlib.image.AxesImage at 0x7aafaecb3bb0>
```



When you are done:

You should have one or more images for each exercise.

1. Double-check that you filled in your name at the top of the notebook!
2. Click `File -> Export Notebook As -> PDF`
3. Email the PDF to `YOURTEAMNAME@beaver.works`