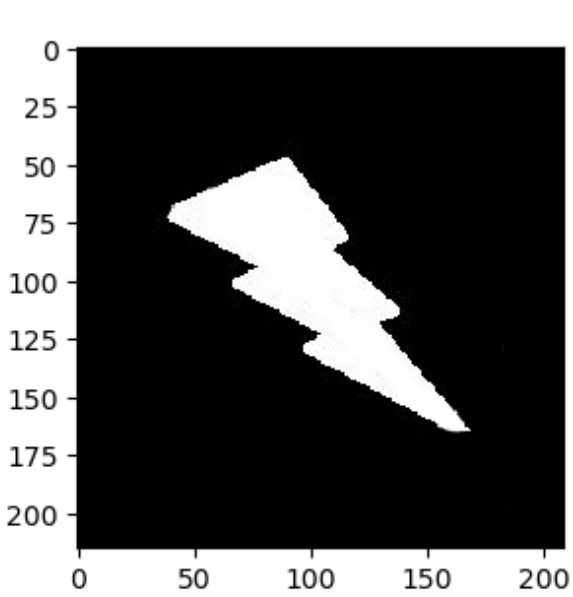
In [36]:
```
%matplotlib inline
from __future__ import print_function
#import ganymede
#ganymede.configure('uav.beaver.works')
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
```

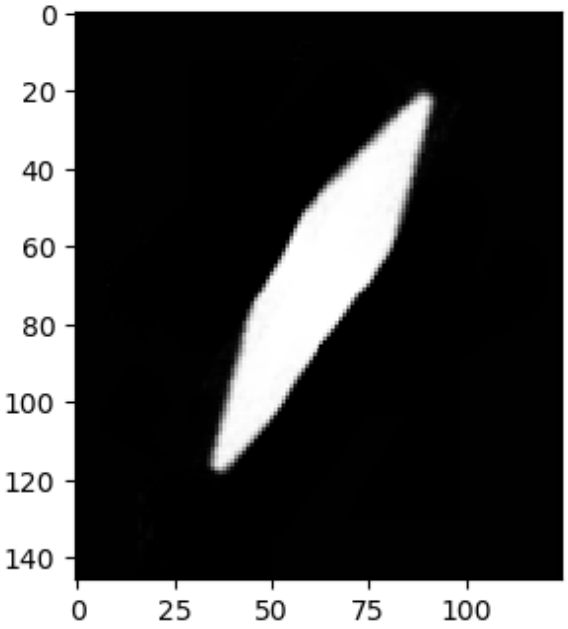
In [37]:
```
def check(p): pass
check(0)
```

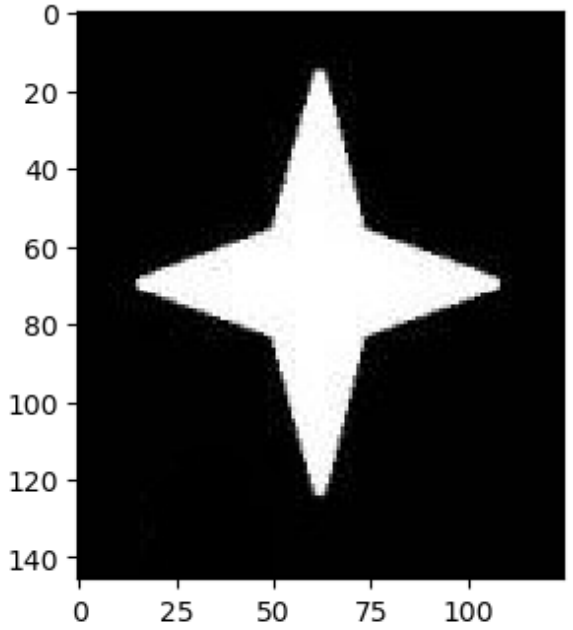
Note

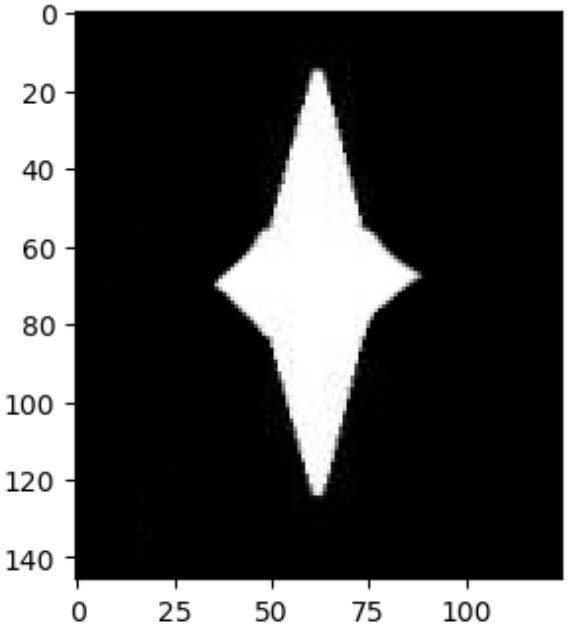
`cv2.imshow()` will not work in a notebook, even though the OpenCV tutorials use it. Instead, use `plt.imshow` and family to visualize your results.

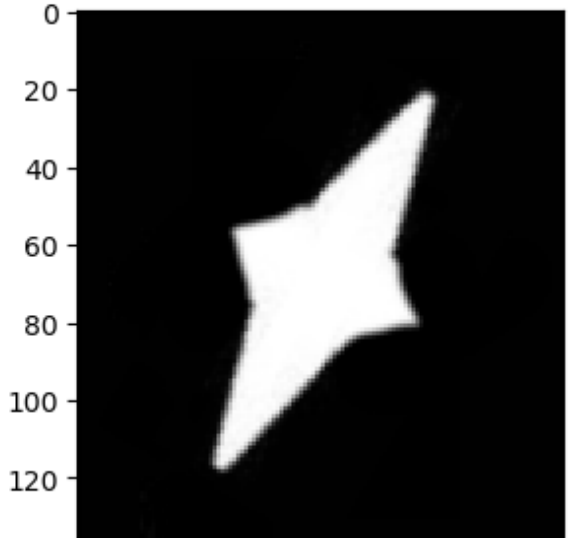
In [38]:
```
lightningbolt      = cv2.imread('shapes/lightningbolt.png', cv2.IMREAD_GR
blob               = cv2.imread('shapes/blob.png', cv2.IMREAD_GRAYSCALE)
star               = cv2.imread('shapes/star.png', cv2.IMREAD_GRAYSCALE)
squishedstar       = cv2.imread('shapes/squishedstar.png', cv2.IMREAD_GRA
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);
```

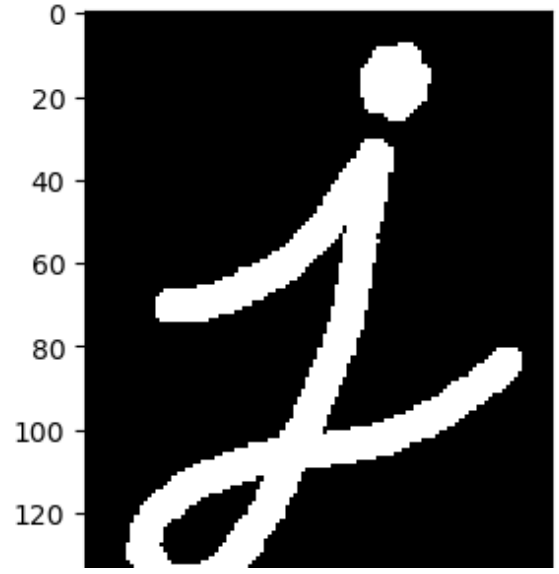

0
25
50
75
100
125
150
175
200
0
50
100
150
200

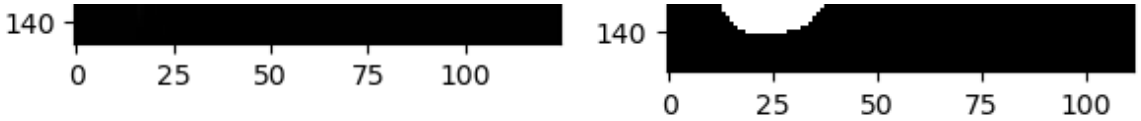

0
20
40
60
80
100
120
140
0
25
50
75
100


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 [39]:
```
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?

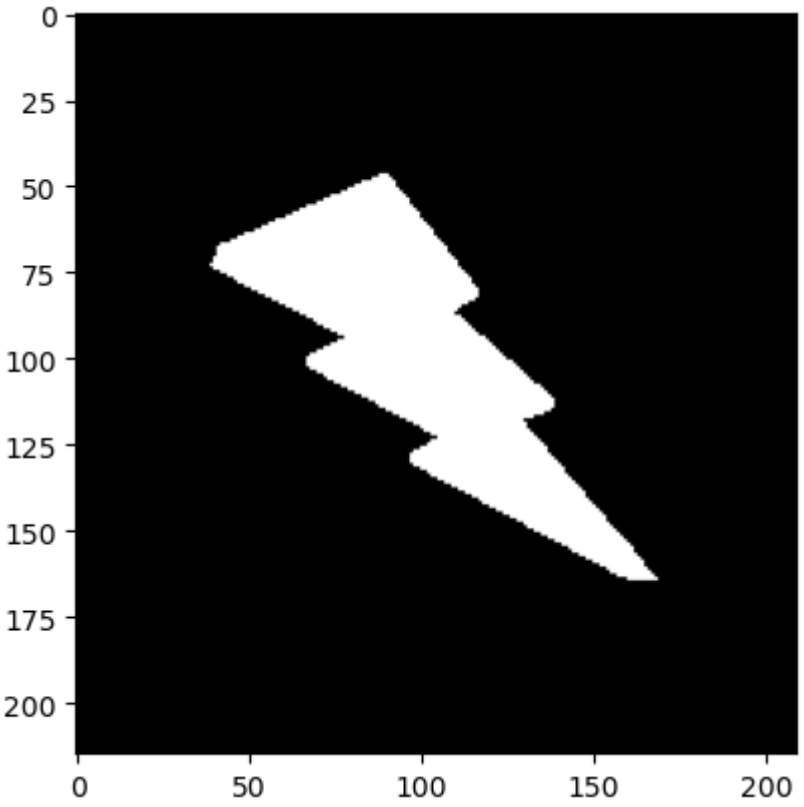
In [40]:
```
#I expect it to be 2 because the image appears to be just black and white
```

Thresholding

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

In [41]:
```
_, lightningbolt = cv2.threshold(lightningbolt,170,255,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 [42]:
```
## TODO
## Thresholding binary makes the image black and white, but because it wa
```

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

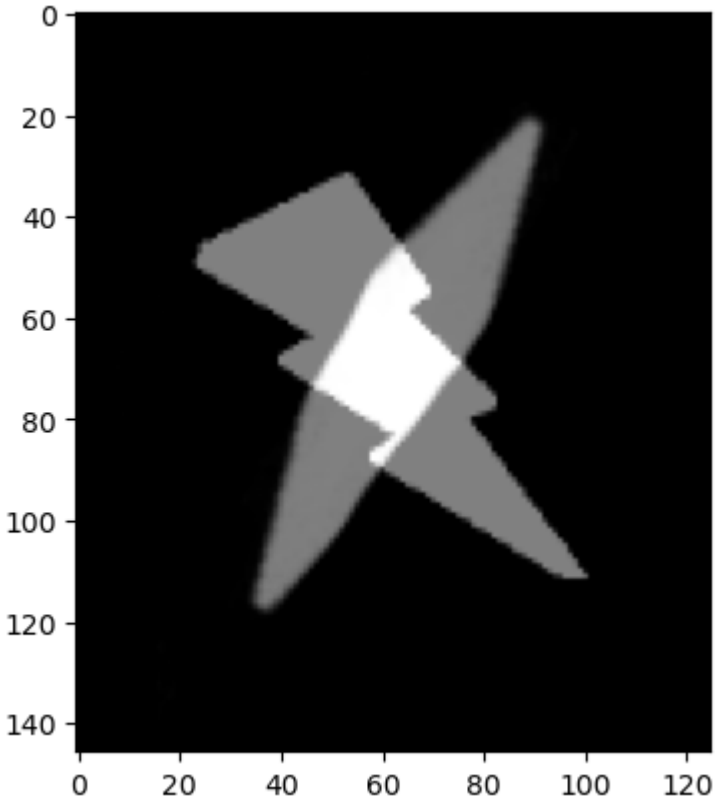
https://docs.opencv.org/3.4.1/d0/d86/tutorial_py_image_arithmetics.html

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

In [80]:
```
import cv2 as cv
print(blob.shape)
resizedlightningbolt = cv.resize(lightningbolt,(125,146))
dst = cv.addWeighted(resizedlightningbolt,0.5,blob,0.5,0)
plt.imshow(dst,cmap="gray")
```

```
(146, 125)
```

Out[80]:
```
<matplotlib.image.AxesImage at 0x75bb80399c90>
```



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

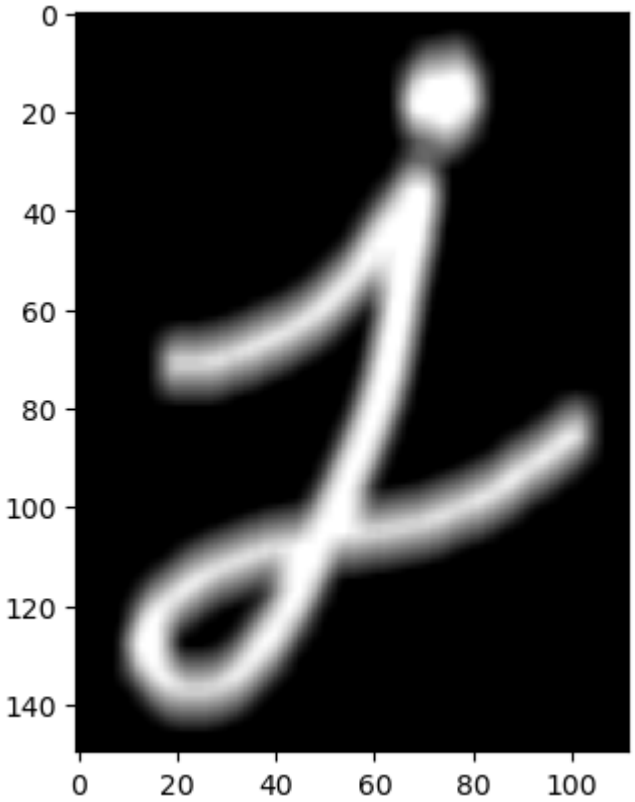
In [ ]:
```
lightningbolt[150:170,150:175]
```

3. Use an averaging kernel on the letter j

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

In [62]:
```
blur = cv.blur(letterj,(5,10))
plt.imshow(blur,cmap="gray")
```

Out[62]:
```
<matplotlib.image.AxesImage at 0x75bb8072a6b0>
```



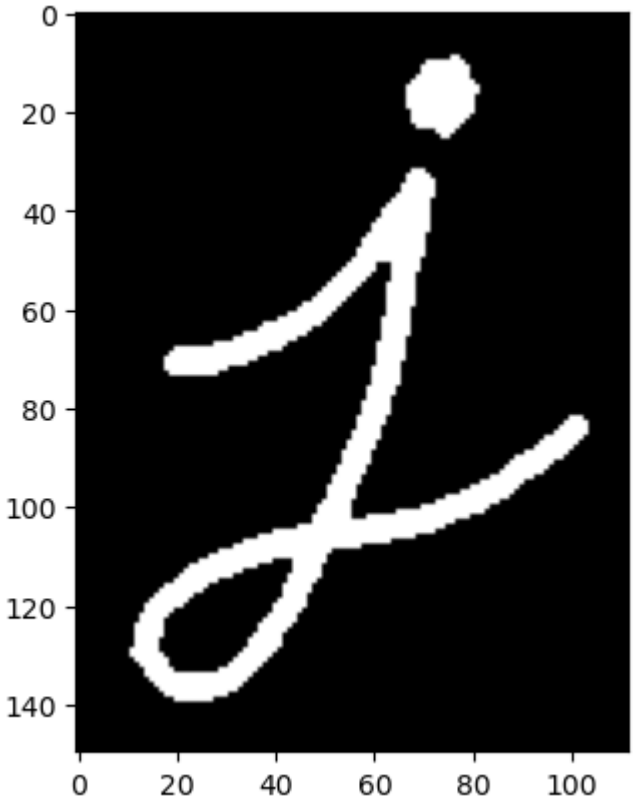
Morphology

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

4. Perform erosion on j with a 3x3 kernel

In [50]:
```
kernel = np.ones((3,3),np.uint8)
erosion = cv.erode(letterj,kernel,iterations = 1)
plt.imshow(erosion,cmap="gray")
```

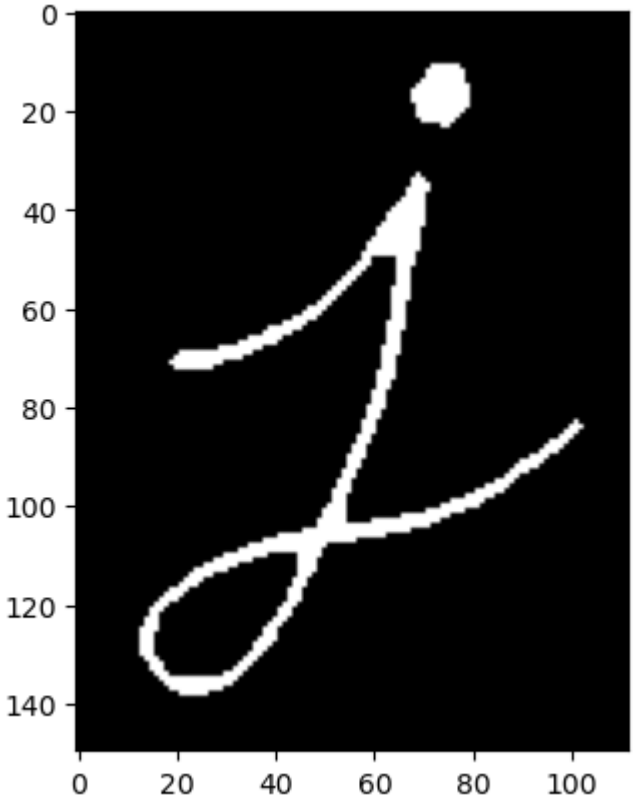
Out[50]:
```
<matplotlib.image.AxesImage at 0x75bb808dd990>
```



5. Perform erosion on j with a 5x5 kernel

In [53]:
```
kernel = np.ones((5,5),np.uint8)
erosion = cv.erode(letterj,kernel,iterations = 1)
plt.imshow(erosion,cmap="gray")
```

Out[53]: <matplotlib.image.AxesImage at 0x75bb80814850>



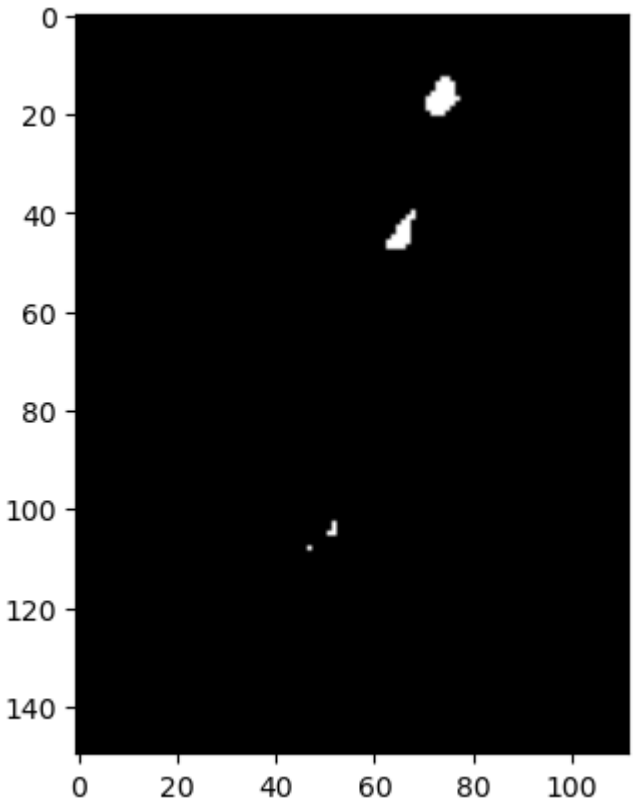
6. Perform erosion on j with **two** iterations, using a kernel size of your choice

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

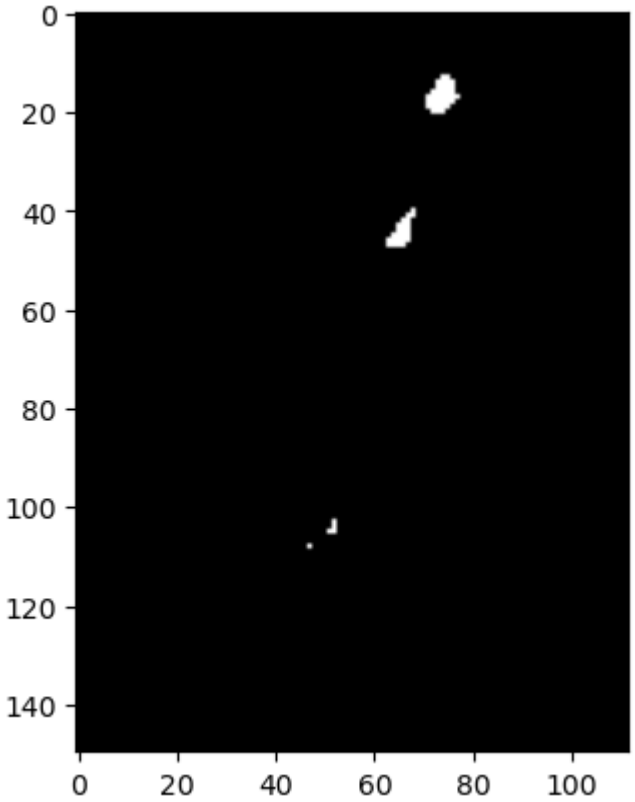
In [61]:
```
kernel = np.ones((5,5),np.uint8)
new_erode = cv.erode(letterj,kernel,iterations = 2)
plt.imshow(new_erode,cmap="gray")
```

Out[61]: <matplotlib.image.AxesImage at 0x75bb806dffd0>



In [74]:
```
kernel = np.ones((9,9),np.uint8)
new_erode = cv.erode(letterj,kernel,iterations = 1)
plt.imshow(new_erode,cmap="gray")
```

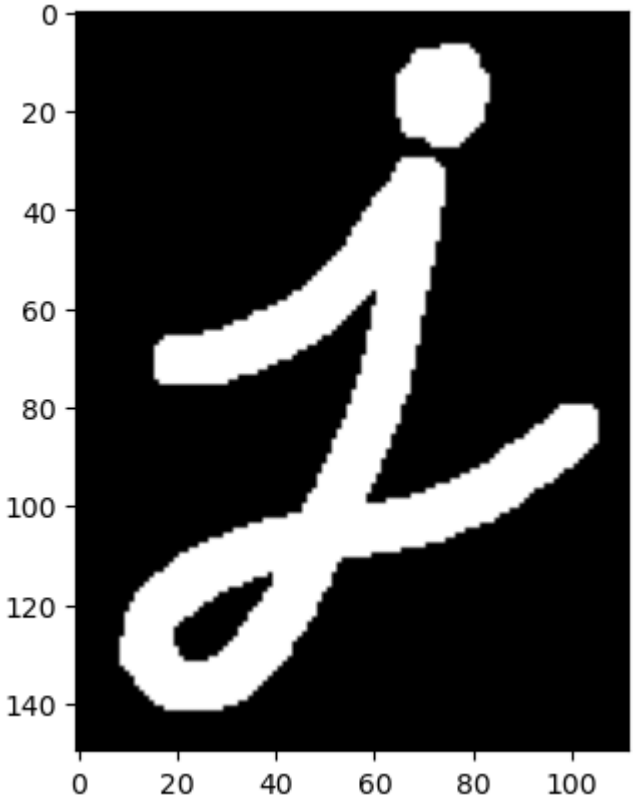
Out[74]: <matplotlib.image.AxesImage at 0x75bb80478d90>



7. Perform dilation on j with a 3x3 kernel

In [67]:
```
kernel = np.ones((3,3),np.uint8)
dilation = cv.dilate(letterj,kernel,iterations = 1)
plt.imshow(dilation,cmap="gray")
```

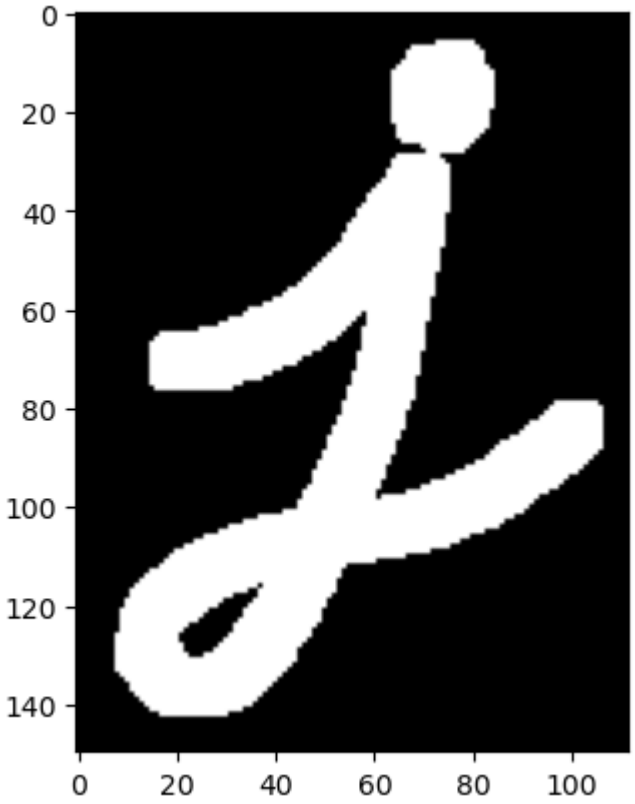
Out[67]: <matplotlib.image.AxesImage at 0x75bb8062be50>



8. Perform dilation on j with a 5x5 kernel

In [69]:
```
kernel = np.ones((5,5),np.uint8)
dilation = cv.dilate(letterj,kernel,iterations = 1)
plt.imshow(dilation,cmap="gray")
```

Out[69]: <matplotlib.image.AxesImage at 0x75bb804ca260>



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

In [ ]:
```
#the kernal larger the kernal size, the larger the dialtion/erosion/blur
```

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 [ ]:
```
# the effect of the small kernal multiple iterations can be changed into
```

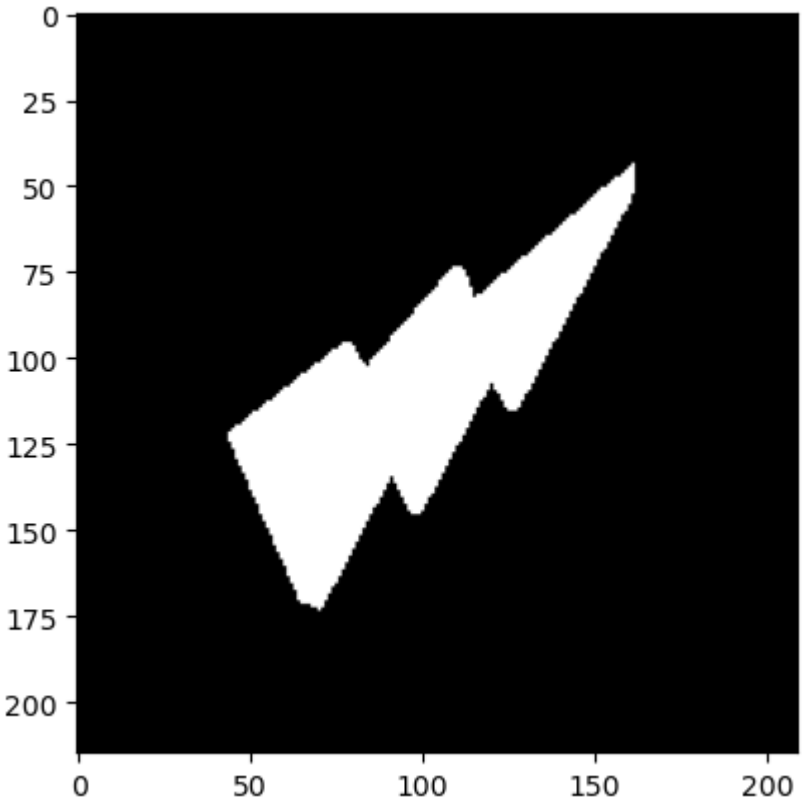
11. Rotate the lightningbolt and star by 90 degrees

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

In [72]:
```
rows,cols = lightningbolt.shape
```

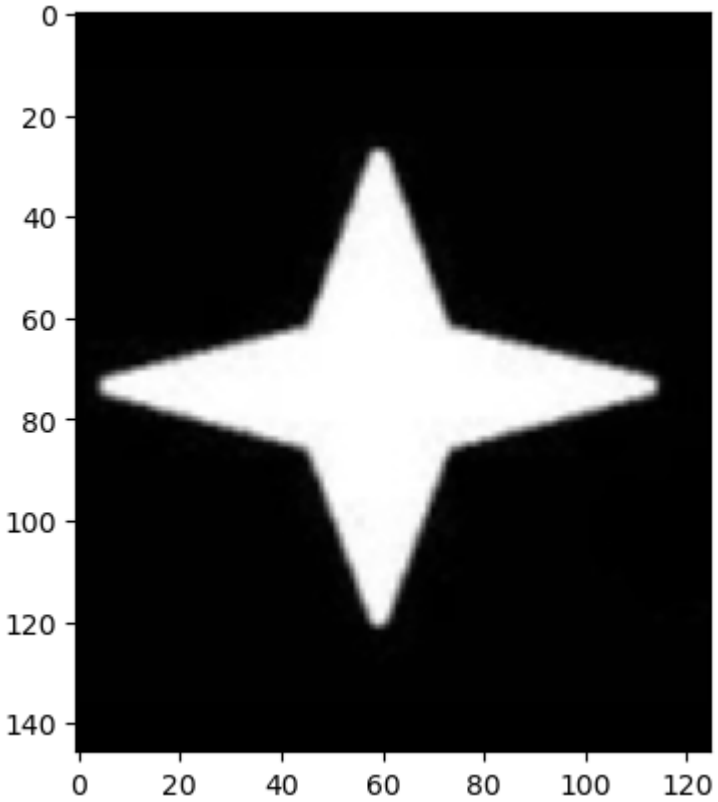
```
M = cv.getRotationMatrix2D((cols/2,rows/2),90,1)
dst = cv.warpAffine(lightningbolt,M,(cols,rows))
plt.imshow(dst,cmap="gray")
```

Out[72]: <matplotlib.image.AxesImage at 0x75bb805a15d0>



In [73]:
```
rows,cols = star.shape
M = cv.getRotationMatrix2D((cols/2,rows/2),90,1)
dst = cv.warpAffine(star,M,(cols,rows))
plt.imshow(dst,cmap="gray")
```

Out[73]: <matplotlib.image.AxesImage at 0x75bb804110f0>



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

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`