```
In [ ]: Krisha Mangrola
```

```python
%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
```

```
/home/krisha/.local/lib/python3.10/site-packages/matplotlib/projections/__init__.py:63: UserWarning: Unable to import Axes3D. This may be due to multiple versions of Matplotlib being installed (e.g. as a system package and as a pip package). As a result, the 3D projection is not availabl
e.
  warnings.warn("Unable to import Axes3D. This may be due to multiple versions of "
```

```python
def check(p): pass
check(0)
```

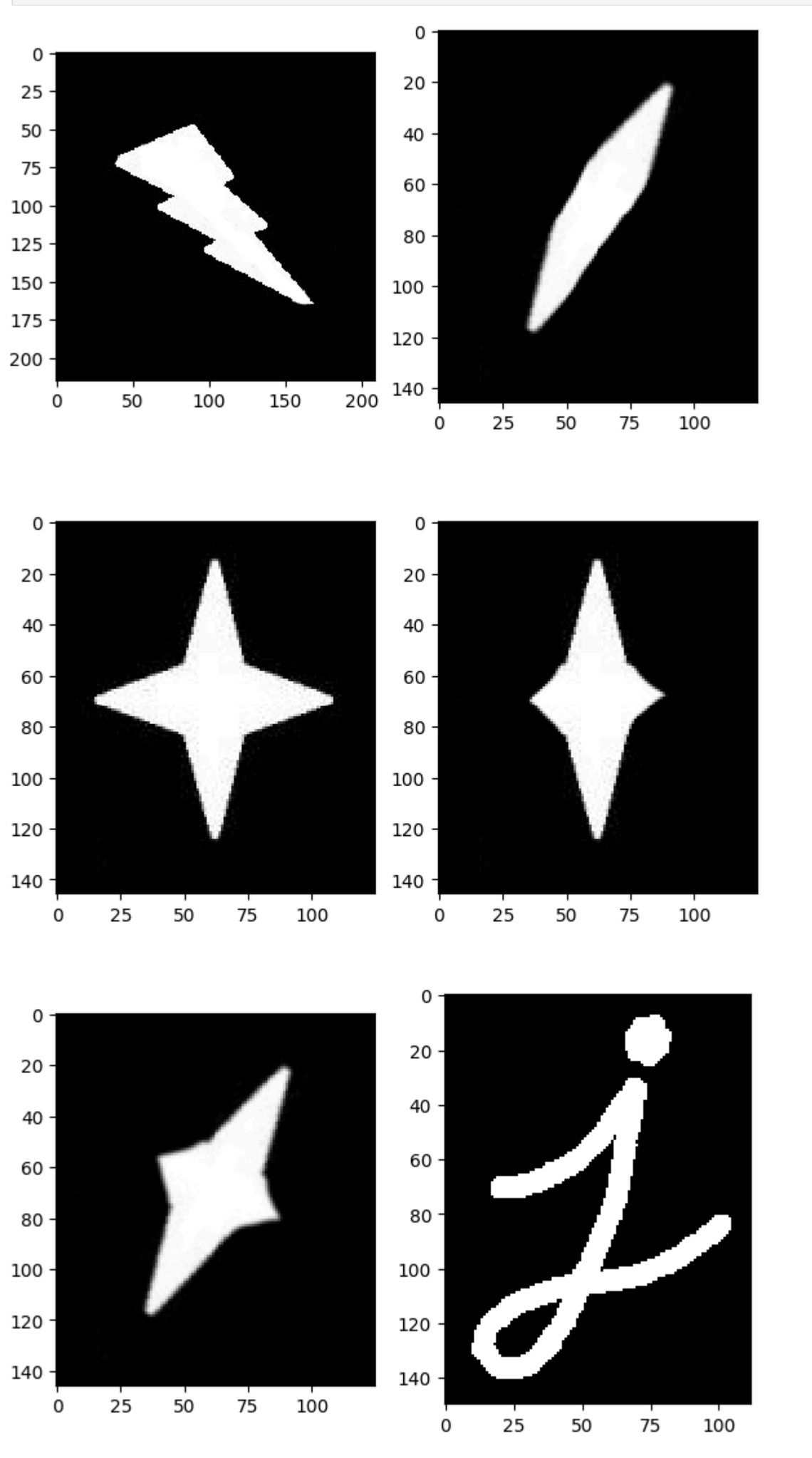
Note

`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.

```python
lightningbolt      = cv2.imread('shapes/lightningbolt.png', cv2.IMREAD_GRAYSCALE)
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_GRAYSCALE)
squishedturnedstar = cv2.imread('shapes/squishedturnedstar.png', cv2.IMREAD_GRAYSCALE)
letterj            = cv2.imread('shapes/letterj.png', cv2.IMREAD_GRAYSCALE)

images = [lightningbolt, blob, star, squishedstar, squishedturnedstar, letterj]

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);
```



```python
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?

```
Expected value: more than 75 because since its flattened it has more pixels of different shades of grey
Actual: 75 becuase the set function only accounts for unique intensity values, of which there are 75
```

```
  Cell In[22], line 1
    Expected value: more than 75 because since its flattened it has more pixels of different shades of grey
             ^
SyntaxError: invalid syntax
```

Thresholding

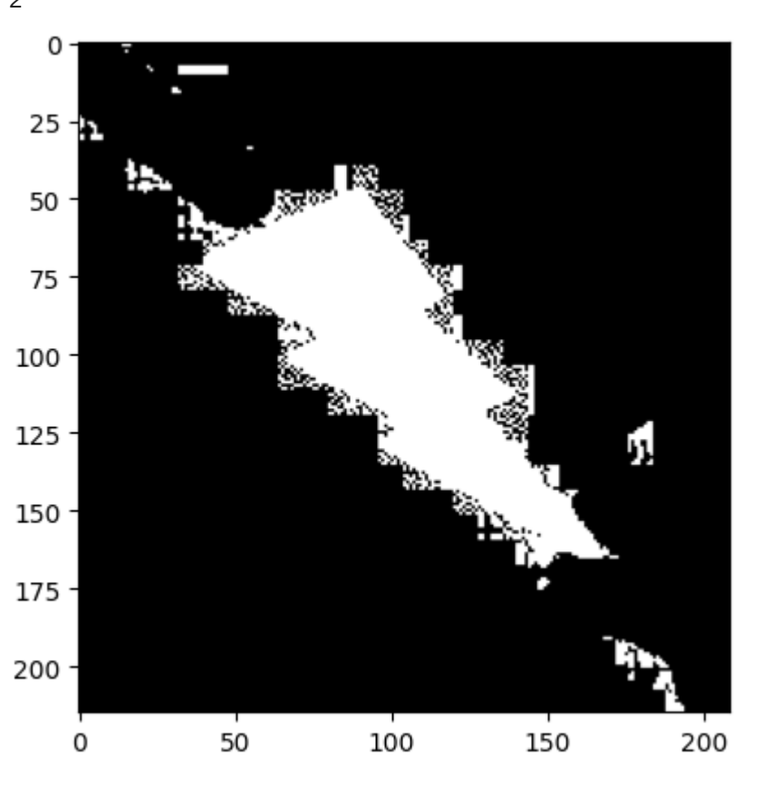
https://docs.opencv.org/3.4.1/d7/d4d/tutorial_py_thresholding.html

```python
_, lightningbolt = cv2.threshold(lightningbolt,0,255,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?

```
When the above values are used for thresholding, 0-127 threshold grey values turn to black and 128-255 grey values turn to white.
A good threshold value is 127 because it is right in the middle.
```

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

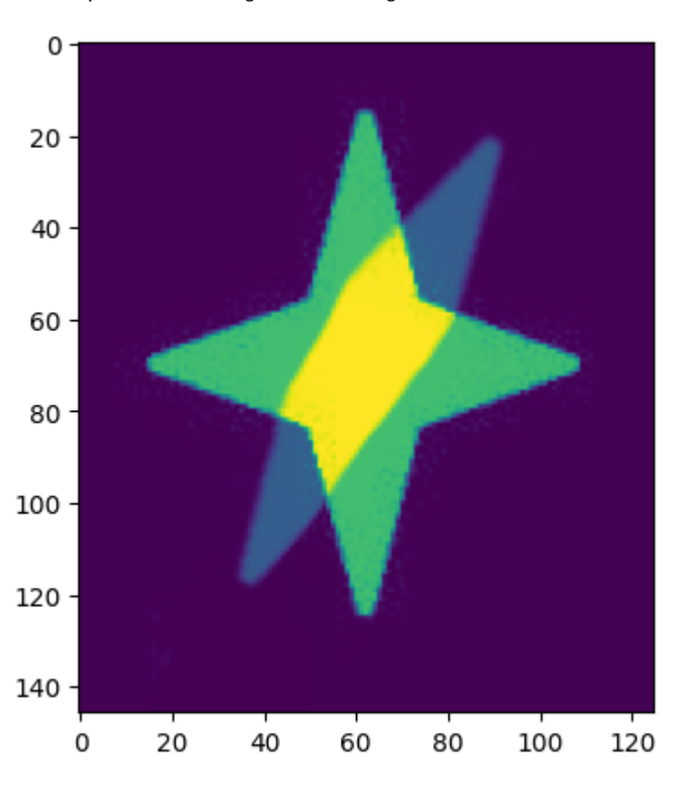
1. Blend lightningbolt and blob together

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`

```python
dst = cv2.addWeighted(star,0.7,blob,0.3,0)
plt.imshow(dst)
```

```
Out[10]: &lt;matplotlib.image.AxesImage at 0x7dd612238640&gt;
```

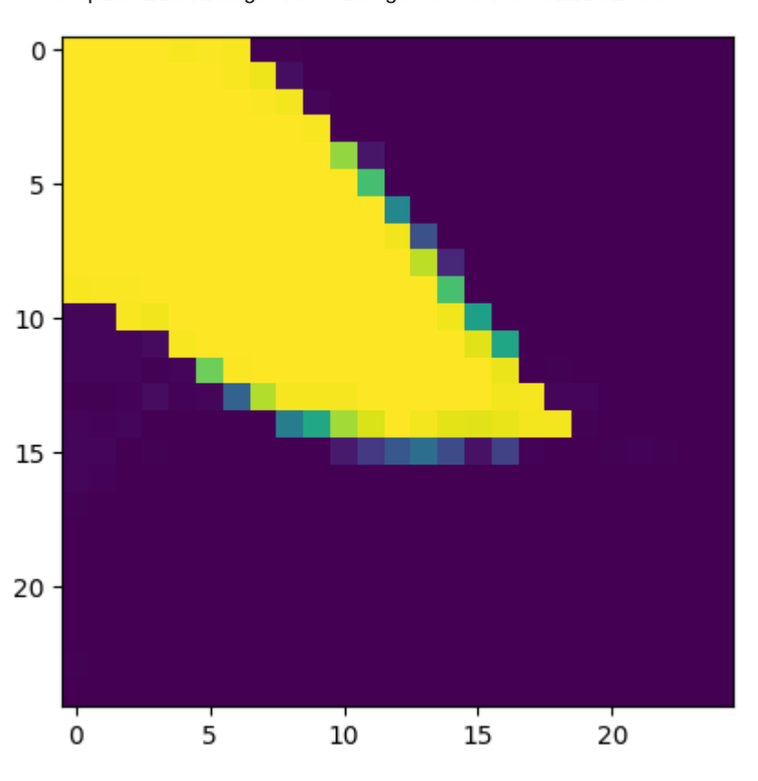


2. Find a ROI which contains the point of the lightning bolt

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

```python
ROI = lightningbolt [150:175, 150:175]
plt.imshow(ROI)
```

```
Out[11]: &lt;matplotlib.image.AxesImage at 0x7dd612291fc0&gt;
```

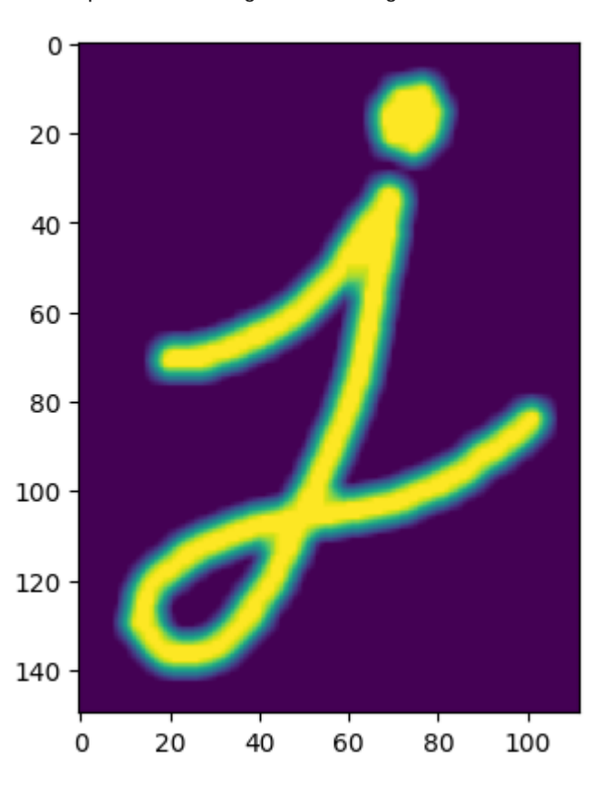


3. Use an averaging kernel on the letter j

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

```python
blur = cv2.blur(letterj,(5,5))
plt.imshow(blur)
```

```
Out[15]: &lt;matplotlib.image.AxesImage at 0x7dd614cf7c10&gt;
```



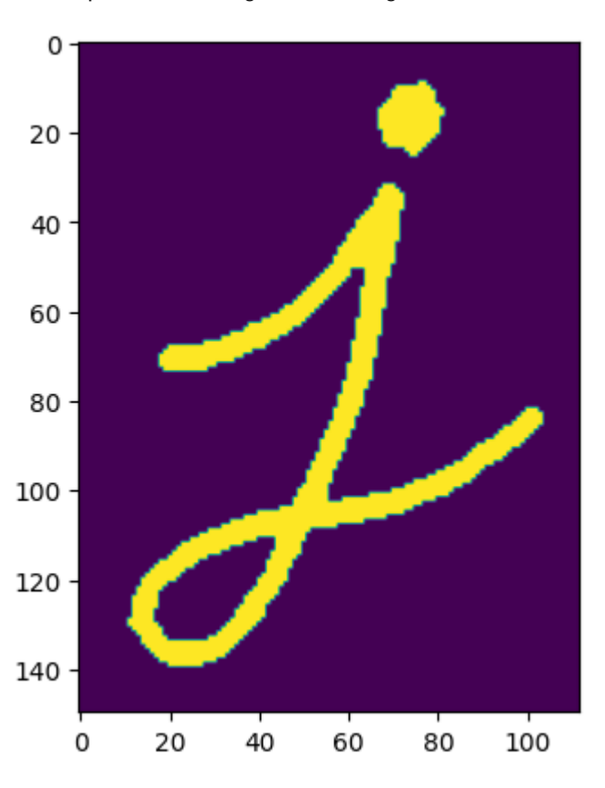
Morphology

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

4. Perform erosion on j with a 3x3 kernel

```python
kernel = np.ones((3,3),np.uint8)
erosion = cv2.erode(letterj,kernel,iterations = 1)
plt.imshow(erosion)
```

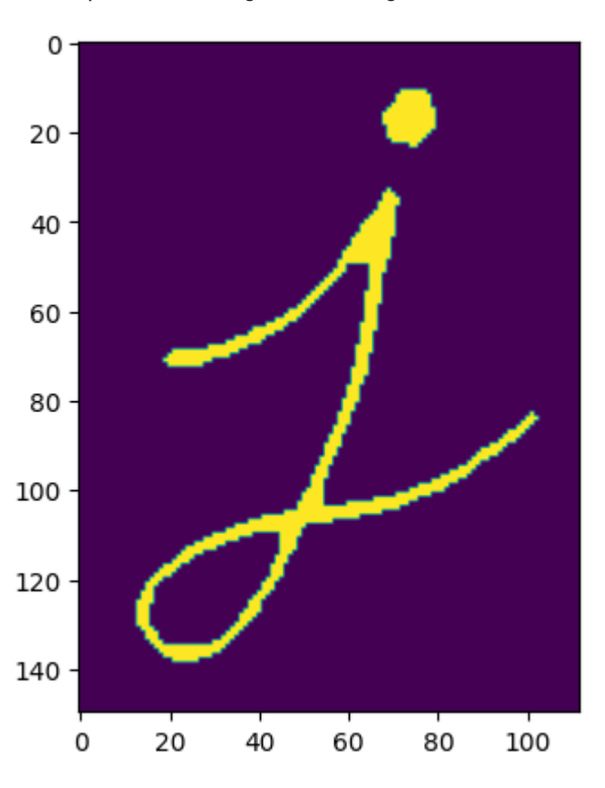
```
Out[21]: &lt;matplotlib.image.AxesImage at 0x7dd6108bd210&gt;
```



5. Perform erosion on j with a 5x5 kernel

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

```
Out[22]: &lt;matplotlib.image.AxesImage at 0x7dd61091f250&gt;
```



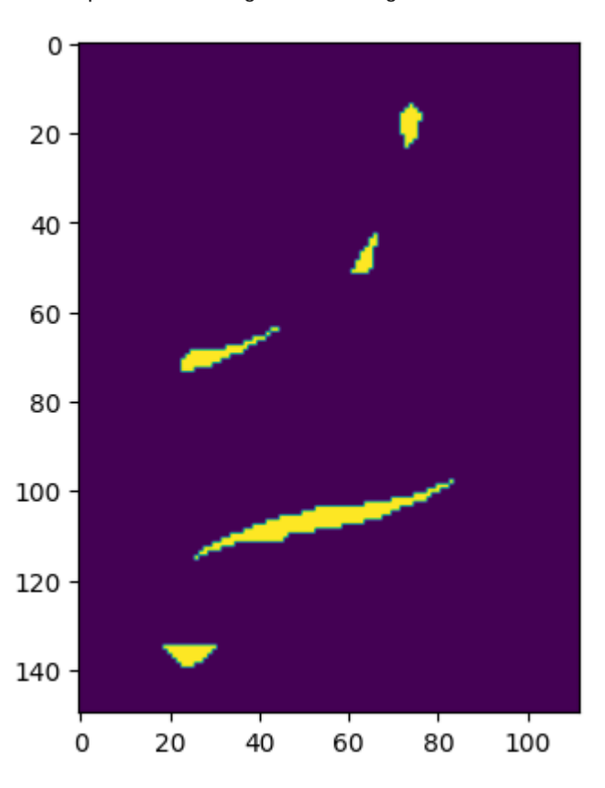
6. Perform erosion on j with **two** iterations, using a kernel size of your choice

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

```python
kernel = np.ones((2,7),np.uint8)
erosion = cv2.erode(letterj,kernel,iterations = 2)
plt.imshow(erosion)
```

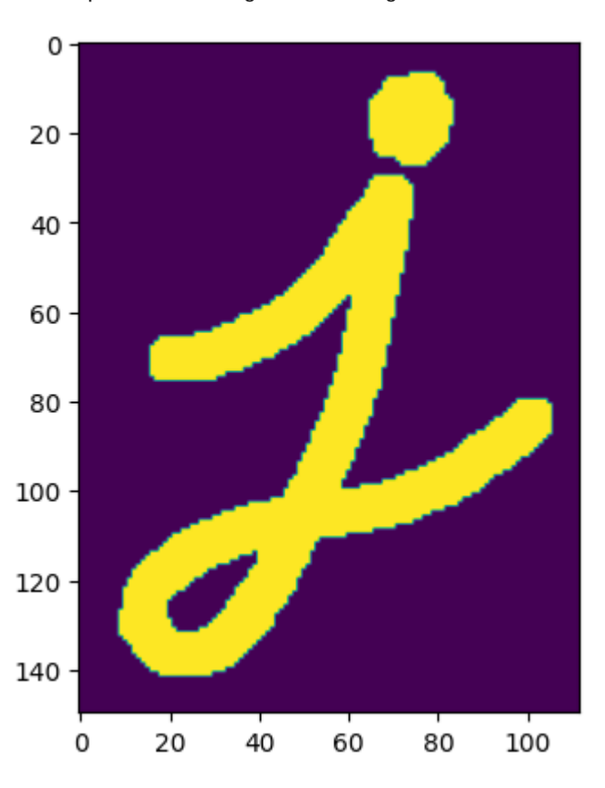
```
Out[32]: &lt;matplotlib.image.AxesImage at 0x7dd61048ac20&gt;
```



7. Perform dilation on j with a 3x3 kernel

```python
kernel = np.ones((3,3),np.uint8)
dilation = cv2.dilate(letterj,kernel,iterations = 1)
plt.imshow(dilation)
```

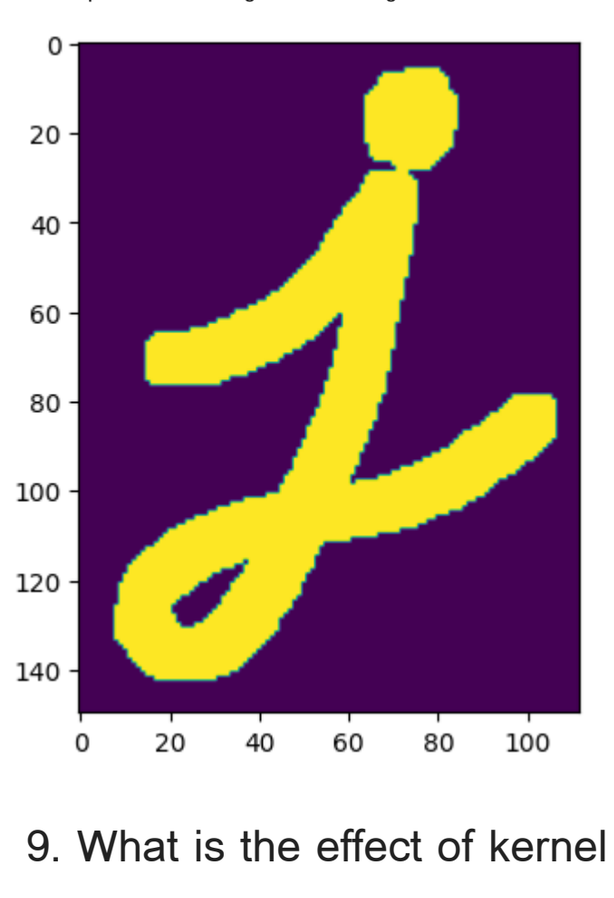
```
Out[29]: &lt;matplotlib.image.AxesImage at 0x7dd614ddd2d0&gt;
```



8. Perform dilation on j with a 5x5 kernel

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

```
Out[30]: &lt;matplotlib.image.AxesImage at 0x7dd6106da710&gt;
```



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

```
The kernel size intensifies the morph. For example, the smaller the size for dilation, the bigger the image appears.
```

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

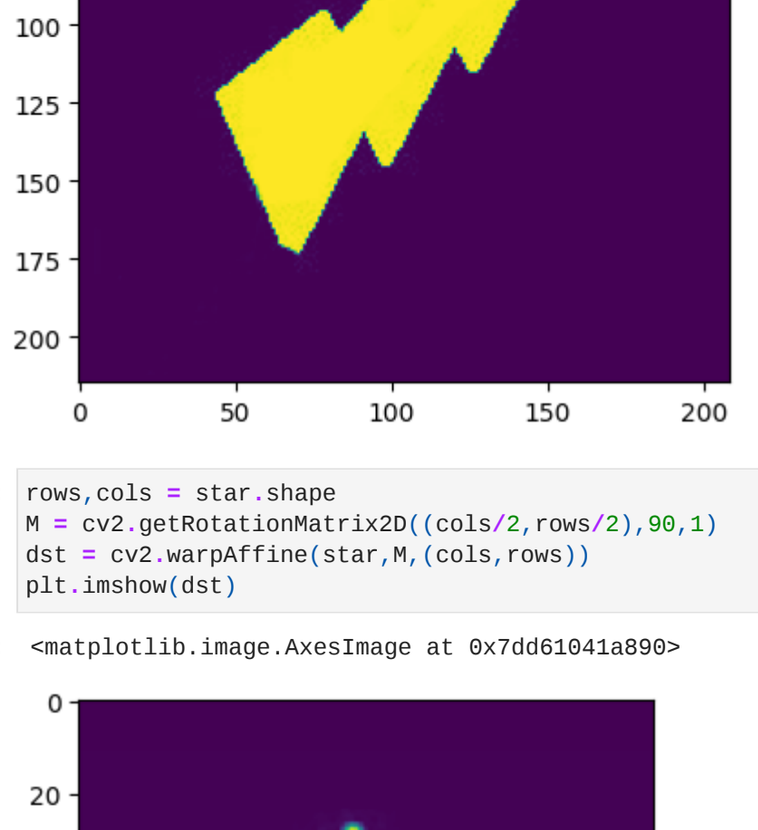
```
Repeated iterations with a small kernel, it would cause the change to happen multiple times. One iteration in a big kernel will cause the morph to occur once, intensley.
```

11. Rotate the lightningbolt and star by 90 degrees

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

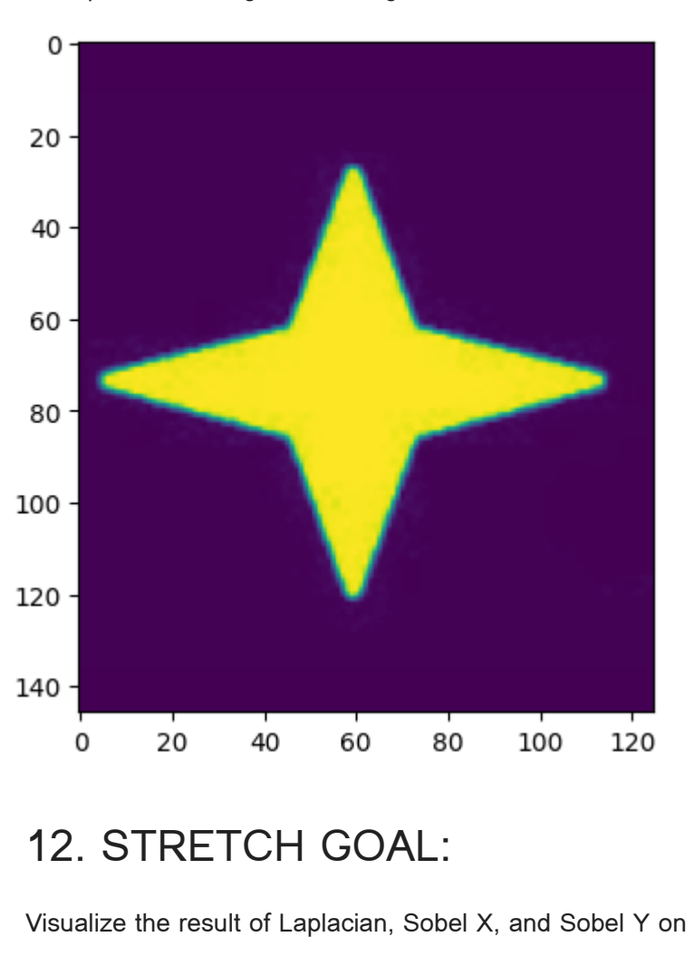
```python
rows,cols = lightningbolt.shape
M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
dst = cv2.warpAffine(lightningbolt,M,(cols,rows))
plt.imshow(dst)
```

```
Out[39]: &lt;matplotlib.image.AxesImage at 0x7dd6102a9000&gt;
```



```python
rows,cols = star.shape
M = cv2.getRotationMatrix2D((cols/2,rows/2),90,1)
dst = cv2.warpAffine(star,M,(cols,rows))
plt.imshow(dst)
```

```
Out[38]: &lt;matplotlib.image.AxesImage at 0x7dd61041a890&gt;
```



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`