1. Points are at (2, 2) and (5, 5). All of the distance parts (1-5) are in distance.py
2. I used scipy.spatial.distance.euclidean to get the euclidean distance automatically. Both mine and SciPpy give 4.2426~. And this follows the distance equation based on the pythagorean theorem:

   $$\sqrt{(5-2)^2 + (5-2)^2} = \sqrt{18}$$

3. When multiplying the points to (4, 4) and (10, 10), both give a distance of 8.4852~:

   $$\sqrt{(10-4)^2 + (10-4)^2} = \sqrt{72}$$

   Multiplying each point by a scalar also multiplies the distance by that scalar value.

4. Here I used scipy.spatial.distance.cityblock, and both mine and SciPy give 6. This is just equal to the x difference plus the y difference:

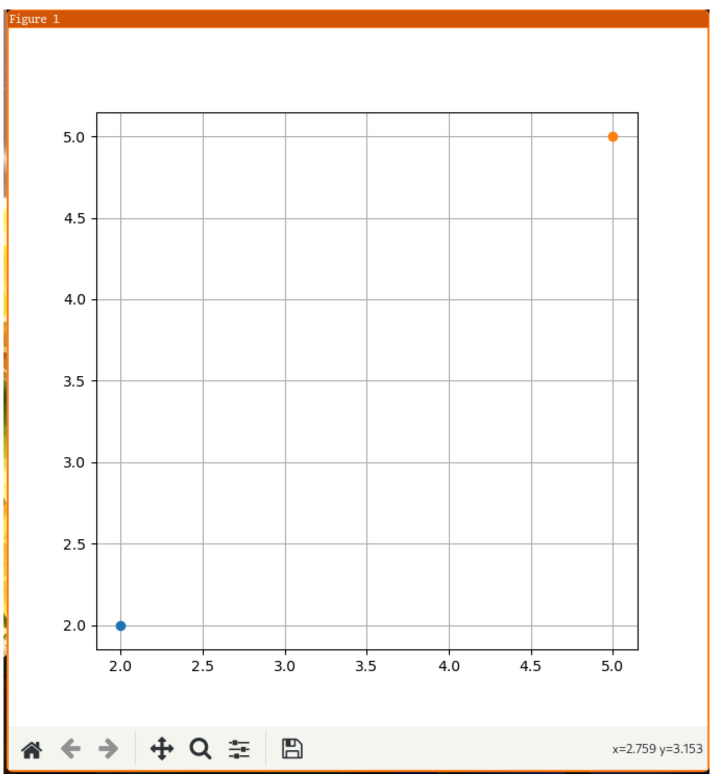
   $$|5-2| + |5-2| = 6$$

5. When multiplying the points to (4, 4) and (10, 10), both give a distance of 12.

   $|10-4| + |10-4| = 12$

   Again, multiplying each point by a scalar value also multiplies the distance by that scalar value.

Here is 2 screenshots of the output of distance.py:



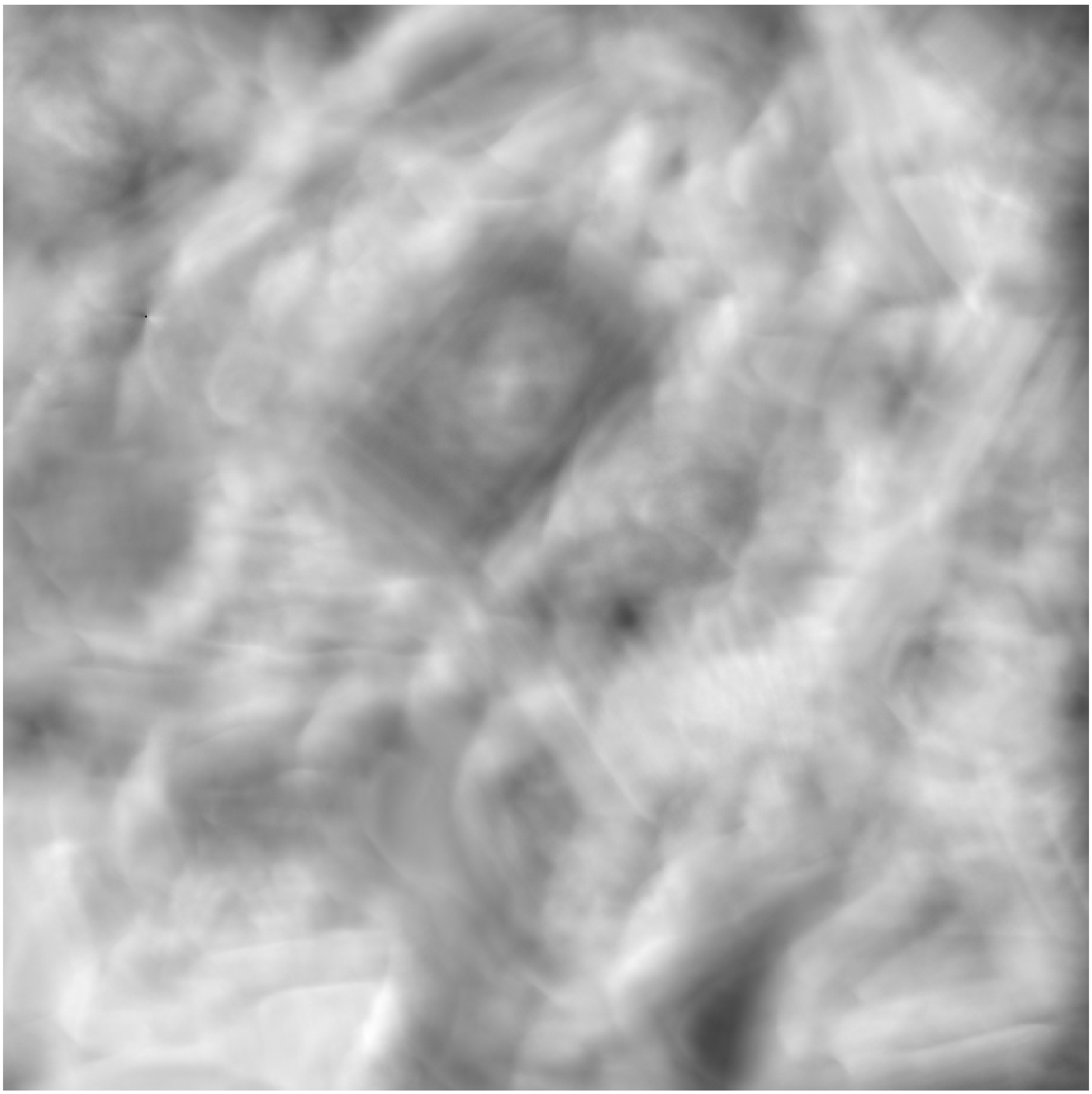

```
[keag@archbook Lab2]$ python distances.py
My euclidean distance: 4.242640687119285
SciPy euclidean distance: 4.242640687119285

My euclidean distance (Points scaled by 2): 8.48528137423857
SciPy euclidean distance (Points scaled by 2): 8.48528137423857

My manhattan distance: 6
SciPy manhattan distance: 6

My manhattan distance (Points scaled by 2): 12
SciPy manhattan distance (Points scaled by 2): 12
```

6. This part is in template_matching.py. You can run it like so:

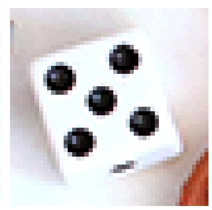
```
python template_matching.py <main_image_path> <template_path>
```

I have 2 sets of images in the images folder, which are what I used to make sure the script worked. When you run the script it will display the smallest difference value (closest match) out of the normalized difference matrix (values 0-255) and the position of that value in the matrix. It then creates and shows an image representation of the normalized difference matrix.

```
[keag@archbook Lab2]$ python template_matching.py images/i_spy.png images/i_spy_template.png
Closest match: 0 @ (75, 164)
```



From these images:



You can see the die near the top left, which lines up with where the program found it.
In the normalized difference matrix visualization, you can see one dark pixel that sticks out right where the top left of the die image is.