One last higher dimensionality, visualization-technique you should know how to use is MatPlotLib's `.imshow()` method. This command generates an image based off of the normalized values stored in a *matrix*, or rectangular array of float64s. The properties of the generated image will depend on the dimensions and contents of the array passed in:

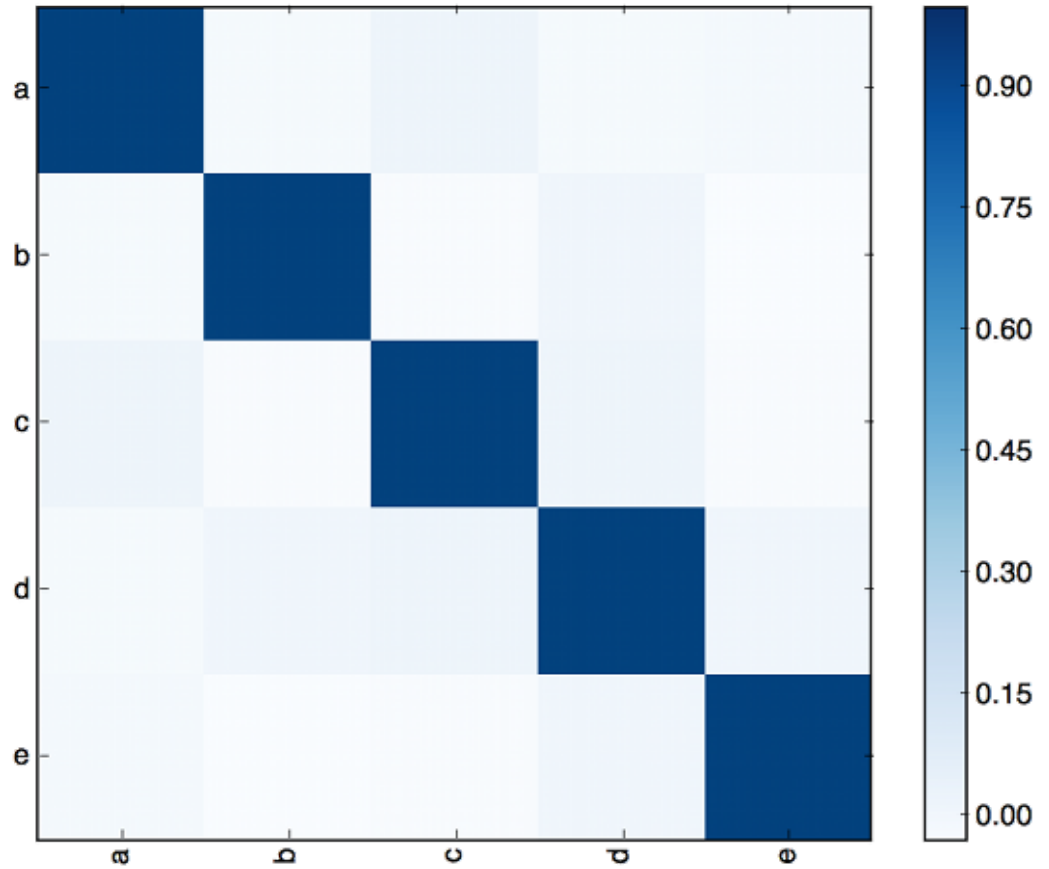
- An [X, Y] shaped array will result in a grayscale image being generated
- A [X, Y, 3] shaped array results in a full-color image: 1 channel for red, 1 for green, and 1 for blue
- A [X, Y, 4] shaped array results in a full-color image as before with an extra channel for alpha

Besides being a straightforward way to display .PNG and other images, the `.imshow()` method has quite a few other use cases. When you use the `.corr()` method on your dataset, Pandas calculates a correlation matrix for you that measures how close to being linear the relationship between any two features in your dataset are. Correlation values may range from -1 to 1, where 1 would mean the two features are perfectly positively correlated and have identical slopes for all values. -1 would mean they are perfectly negatively correlated, and have a negative slope for one another, again being linear. Values closer to 0 mean there is little to no linear relationship between the two variables at all (e.g., pizza sales and plant growth), and so the the further away from 0 the value is, the stronger the relationship between the features:

```
>>> df = pd.DataFrame(np.random.randn(1000, 5), columns=['a', 'b', 'c', 'd', 'e'])
>>> df.corr()

          a         b         c         d         e
a  1.000000  0.007568  0.014746  0.027275 -0.029043
b  0.007568  1.000000 -0.039130 -0.011612  0.082062
c  0.014746 -0.039130  1.000000  0.025330 -0.028471
d  0.027275 -0.011612  0.025330  1.000000 -0.002215
e -0.029043  0.082062 -0.028471 -0.002215  1.000000
```

The matrix is symmetric because the correlation between any two features **X** and **Y** is, of course, identical to that of features **Y** and **X**. It is invariant to scale, so even if one feature is measured in inches and the other is in centimeters, it makes no difference. This matrix and others like the covariance matrix, are useful for inspecting how the variance of a feature is explained by the variance in other feature, and verifying how much new information each feature provides. But even looking at this little, 5x5 matrix makes me dizzy, so you can imagine how easy it is to get lost in a higher dimensionality dataset. You can circumvent this by visualizing your correlation matrix by plotting it with `.imshow()`:

```
import matplotlib.pyplot as plt

plt.imshow(df.corr(), cmap=plt.cm.Blues, interpolation='nearest')
plt.colorbar()
tick_marks = [i for i in range(len(df.columns))]
plt.xticks(tick_marks, df.columns, rotation='vertical')
plt.yticks(tick_marks, df.columns)

plt.show()
```



`.imshow()` can help you any time you have a square matrix you want to visualize. Other matrices you might want to visualize include the covariance matrix, the confusion matrix, and in the future once you learn how to use certain machine learning algorithms that generate clusters which live in your feature-space, you'll also be able to use `.imshow()` to peek into the brain of your algorithms as they run, so long as your features represent a rectangular image!