In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Let's fabricate a problem and suppose the dominant eigenvalue/eigenvector is $\lambda = 1.0$ and $v = [1, 1]$. Then pick $\lambda = 0.5$ and some random vector for the other problem.

To generate a *problem* we want to make sure we don't mess up the eigenalues.

An *orthogonal* matrix a matrix such that $Q^T = Q$.

In [12]:

```
V = np.random.rand(2,2)
V[:,0] = 1
Q, R = np.linalg.qr(V)
D = np.diag([1.0, 0.5])
A = Q.dot(D).dot(Q.T)
```

In [11]:

```
np.linalg.eig(A)
```

Out[11]:

```
(array([ 0.5,  1. ]), array([[-0.70710678, -0.70710678],
        [ 0.70710678, -0.70710678]]))
```

In [21]:

```
x = np.array([1,0])
plt.arrow(0,0,x[0],x[1], lw=4, clip_on=False)
for i in range(10):
    x = A.dot(x)
    x /= x.max()
    plt.arrow(0,0,x[0],x[1], lw=4, clip_on=False)
```