```python
import numpy as np

# np.random.seed(1)
B = np.random.randn(4,4)
A = B + B.T

I = np.identity(4)
# initialize guess eigenvector
b = np.random.randn(4)


# initialize lambda guess
u = 5
# compare eigenvalue with theoretical eigenvalue's
print np.linalg.eig(A)[0]


# Power iteration's to find the largest absolute eigenvalue
def PowerIterate(M,N,I,u,b):
    # generate random eigenvector
    b = np.random.rand(M.shape[0])

    for i in range(N):
        # calculate  Mb
        b_1 = np.dot(M, b)

        # calculate the norm
        b_1_norm = np.linalg.norm(b_1)

        # re normalize the vector
        b = b_1 / b_1_norm
    # initialize list for tracking u (eigenvalue)

    ulist = [u]
    return RayleighQuotient(M,I,u,b,ulist,ctr=0)


# Rayleighquotient
def RayleighQuotient(M,I,u,b,ulist,ctr=0):
    ctr+=1
    # get the new value of the eigenvector
    d = np.linalg.inv(M-u*I).dot(b)
    c = np.linalg.norm(d,np.inf)
    b = d/c
    # get the new value of sigma
    u = (np.conj(b).dot(M).dot(b))/(np.conj(b).dot(b))
```

```
47      ulist.append(u)
48
49      # if no change in the residuals on order of 10^-4 return values
50      residuals = M.dot(b) - u * (b)
51      if(np.linalg.norm(residuals,ord=2)<10**(-4)):
52          return u,b,ctr
53
54      return RayleighQuotient(M,I,u,b,ulist,ctr)
55
56
57  u,b,ctr = PowerIterate(A,20,I,u,b)
58  print A
59  print "We have %0.3f as the largest absolute eigenvalue after %0.0f
    iterations" %(u,ctr)
60  print "And as the Eigenvector: "+ str(b)
61
```