```python
import numpy as np
import matplotlib.pyplot as plt
B = np.zeros((6,6))
B[0][2] = 1
B[1][0] = 1
B[1][3] = 1
B[1][4] = 1
B[2][1] = 1
B[2][5] = 1
B[3][0] = 1
B[3][2] = 1
B[4][3] = 1
B[5][1] = 1
B[5][2] = 1
B[5][4] = 1

# intialize matrix
def InitMatrix(B):

    A = np.zeros((6,6))
    for i in range(np.shape(B)[0]):
        normlist = []
        for j in range(np.shape(B)[0]):
            normlist.append(B[j][i])
        norm = sum(normlist)
        for j in range(np.shape(B)[0]):
            A[j][i] = B[j][i] / norm
    return A

# initialize random eigenvector
def InitVector():
    x = np.random.rand(6)
    z = np.sum(x)
    vectlist = []
    for i in x:
        vectlist.append(i/z)
    b = np.array(vectlist)
    return b

# Power iteration's to find the largest absolute eigenvalue
def PowerIterate(M,N,b):
    i = 0
    stoplist=[0]
    while i < N:
        # calculate  Mb
        b_1 = np.dot(M, b)
```

```
47          # calculate the norm
48          b_1_norm = np.linalg.norm(b_1)
49          # re normalize the vector
50          b = b_1 / b_1_norm
51
52          lambdas = CalcLambda(b,M)
53          # if no change in the residuals on order of 10^-4 return
            values
54          residuals = M.dot(b) - lambdas * (b)
55          if(np.linalg.norm(residuals,ord=2)<10**(-4)):
56              return b,i,residuals,lambdas
57          i+=1
58      return b,i,residuals,lambdas
59
60  #  modifiy matrix for question 3.c
61  def Convect(M,alpha,size=6):
62      e = np.ones(size)
63      y = (1./size)*e
64      P = alpha*M +((1-alpha)*y.dot(e.T))
65      return P
66
67  # Find the eigenvalue using the RayleighQuotient:
68  def CalcLambda(powervect,A):
69      return
            powervect.T.dot(A).dot(powervect)/((powervect.T).dot(powervect)
            )
70
71  #  Init matrices
72  A = InitMatrix(B)
73  B[3][2] = 0
74  B[0][2] = 0
75  C = InitMatrix(B)
76  b = InitVector()
77
78  # print everything and call the functions
79  print "-------------A-----------------"
80  powervect,iters,residuals,lambdas = PowerIterate(A,2000,b)
81  node = np.argmax(abs(powervect)) +1
82  print "Lambda is: "+ str(lambdas)+" After "+ str(iters) + "
            iterations"
83  print "Eigenvector is: "+ str(powervect)
84  print "Highest node is: " + str(node)
85  print "Residual is: " +str(residuals)
86
87  # C is the matrix where the nodes from 3 to 4 and 3 to 1 are
            removed
```

```
88  print "--------------------------------"
89  print C
90  print "--------------------------------"
91  powervect,iters,residuals,lambdas = PowerIterate(C,2000,b)
92  node = np.argmax(abs(powervect)) +1
93  print "--------------C-----------------"
94  print "Lambda is: "+ str(lambdas)+" After "+ str(iters) + "
    iterations"
95  print "Eigenvector is: "+ str(powervect)
96  print "Highest node is: " + str(node)
97  print "Residual is: " +str(residuals)
98
99  # P is the modified C matrix here for alpha = 0.95
100 P = Convect(C,0.95)
101 powervect,iters,residuals,lambdas = PowerIterate(P,2000,b)
102 node = np.argmax(abs(powervect)) +1
103 print "---------------P for alpha = 0.95----------------"
104 print "Lambda is: "+ str(lambdas)+" After "+ str(iters) + "
    iterations"
105 print "Eigenvector is: "+ str(powervect)
106 print "Highest node is: " + str(node)
107 print "Residual is: " +str(residuals)
108
109 # P is the modified C matrix here for alpha = 0.75
110 P = Convect(C,0.75)
111 powervect,iters,residuals,lambdas = PowerIterate(P,2000,b)
112 node = np.argmax(abs(powervect)) +1
113 print "---------------P for alpha = 0.75----------------"
114 print "Lambda is: "+ str(lambdas)+" After "+ str(iters) + "
    iterations"
115 print "Eigenvector is: "+ str(powervect)
116 print "Highest node is: " + str(node)
117 print "Residual is: " +str(residuals)
118
119 list95 = []
120 list75 = []
121 # compare iterations with an alpha of 0.95 and 0.75 over 1000 times
122 for i in range(1000):
123     b = InitVector()
124     P = Convect(C,0.95)
125     powervect,iters,residuals,lambdas = PowerIterate(P,2000,b)
126     list95.append(iters)
127     P = Convect(C,0.75)
128     powervect,iters,residuals,lambdas = PowerIterate(P,2000,b)
129     list75.append(iters)
130
```

```
# plot iteration comparison
plt.plot(list95, label= r"$\alpha$ = 0.95")
plt.plot(list75, label= r"$\alpha$ = 0.75")
plt.xlabel("FunctionCall",fontsize=20)
plt.ylabel("Number of Iterations",fontsize=20)
plt.tick_params(axis='both', labelsize=15)
plt.legend()
plt.title("Excercise 3, iteration comparison",fontsize=30)
plt.show()


# Perform matrix modification for alpha from 0 to 1 in steps of
0.01
alphalist = []
a = np.arange(0,1,0.01)
for i in a:
    P = Convect(C,i)
    powervect,iters,residuals,lambdas = PowerIterate(P,2000,b)
    alphalist.append(iters)
plt.scatter(a,alphalist)
plt.xlabel(r"$\alpha$",fontsize=20)
plt.ylabel("Number of Iterations",fontsize=20)
plt.tick_params(axis='both', labelsize=15)
plt.legend()
plt.title("Excercise 3 different Alpha's",fontsize=30)
plt.show()

```