```
#
# Topics from CHAPTER 6 of the book
# Eigenvalues and Eigenvectors
#
# The Relaxation Method for Nonlinear Equations
#




# ------------------------------------------------------------------
#                    EIGENVALUES and EIGENVECTORS
# ------------------------------------------------------------------

# The most common method to find the eigenvalues and eigenvectors of a
# real symmetric or Hermitian matrix on a computer is the
# QR algorithm
#
# SEE the PDF file Lec11_Eigenvalues to review how to
# find analytically the eigenvalues and eigenvectors of
# 2x2 and 3x3 matrices
#
# and to
#
# learn about the QR algorithm
#
#
# ---------------------------
# Using NUMPY.LINALG
# ---------------------------
#
# For real symmetric or Hermitian matrix, use
#
# eigh for eigenvalues and eigenvectors of a NxN matrix
# (First array is a vector of dimension N = eigenvalues,
#  second array is a matrix NxN, where each column is an eigenvector)
#
# eigvalsh for eigenvalues ONLY.


print('Eigenvalues and Eigenvectors')
import numpy as np
import numpy.linalg as npa
A = np.array( [  [1,2], [2,1]  ], float)
en, v = npa.eigh(A)
print(en)
print(v)

print()
print('Eigenvalues only')
enOnly = npa.eigvalsh(A)
```

```
print(enOnly)



#%%
# ------------------------------------------------------------------------
#                              GOOGLE MATRIX
# ------------------------------------------------------------------------

# See PDF files LA01, LA02, and especially LA03

# LA03 LA03 LA03 LA03 LA03 LA03 LA03 LA03 LA03 LA03
# Solve Problems 3 and 4 from the end of PDF file LA03

# ------------------------
# IMPORTANT COMMENTS!!!!!
# ------------------------
#
# (1) The Google Matrix is a POSITIVE column STOCHASTIC matrix
# *) entries are real number > 0
# *) sum of the entries in each column =1
# *) largest eigenvalue = 1
# *) for this eigenvalue, there exists a UNIQUE eigenvector, whose sum of the
# entries is 1

#
# (2) The Google Matrix is NOT symmetric, so we CANNOT use
# eigh from numpy.linalg
# INSTEAD,
# we use
# eig from numpy.linalg

print('Eigenvalues and Eigenvectors for the')
print('TRANSITION MATRIX in LA03.pdf')
import numpy as np
import numpy.linalg as npa
A = np.array( [  [0,0,1,1/2],[1/3,0,0,0],[1/3,1/2,0,1/2], [1/3,1/2,0,0]  ],
 float)
en, v = npa.eig(A)
print(en)
print(v)

#
# (3) NOTE that the eigenvector associated with the eigenvalue 1
# is normalized so that |x|^2+|y|^2+|z|^2+|w|^2 =1
# BUT for the PAGE RANK vector, we want
# x+y+z+w=1.
# We therefore need to renormalize the vector.

print()
```

```
vec = np.real([v[0,0],v[1,0],v[2,0],v[3,0]])
print("The eigenvector with eigenvalue 1")
print(vec)
som = vec[0]**2 + vec[1]**2 + vec[2]**2 + vec[3]**2
print("has |x|^2 + |y|^2 + |z|^2 + |w|^2 = ",som)

print()
print("We can renormalize it, so that the new vector")
som = vec[0] + vec[1] + vec[2] + vec[3]
NewVec = vec/som
print(NewVec)
som = NewVec[0] + NewVec[1] + NewVec[2] + NewVec[3]
print("has x + y + z + w = ",som)

print()
print("The new vector is the PAGE RANK!")

#
# (4) The PAGE RANK can be obtained either by diagonalization or by
# iteration. This is the
#                  Power Method Convergence Method

# This method is feasible even when we deal with huge matrices, which
# exact diagonalization cannot handle!

# Start with
print()
print("Get the PAGE RANK by iteration")
vec = [0.25,0.25,0.25,0.25]
A = np.array( [  [0,0,1,1/2],[1/3,0,0,0],[1/3,1/2,0,1/2], [1/3,1/2,0,0]  ],
 float)
# Proceed with iterations
for n in range(10):
    vec=np.dot(A,vec)
print(vec)



#
# (5) The GOOGLE MATRIX "M" is actually a sum of the
#     TRANSITION MATRIX "A" and a matrix "B" where all entries are 1/Np
#     with Np being the total number of pages
#
#     M = (1-p)*A + p*B
#     where 0<= p <= 1



# ------------------------------------------------
# Exercise
# ------------------------------------------------
# LA03 LA03 LA03 LA03 LA03 LA03 LA03 LA03 LA03 LA03
```

Solve Problems 3 and 4 from the end of PDF file LA03