```
import numpy as np
from numpy.linalg import qr
np.set_printoptions(precision=2, suppress=True) # It is set to display
& secure the output with 4 decimals

def qr_factorization(A):      #Defining the function
    m, n = A.shape
    Q = np.zeros((m, n))
    R = np.zeros((n, n))

    for j in range(n):
        v = A[:, j]

        for i in range(j):
            q = Q[:, i]
            R[i, j] = q.dot(v)
            v = v - R[i, j] * q

        norm = np.linalg.norm(v)
        Q[:, j] = v / norm
        R[j, j] = norm
    return Q, R

a = np.array([[0,2],
              [2,3]])

n = int(input("How many iterations you want?"))
print("\n Given Matrix : A = ",a)
print("\n QR Factorization of A : ",qr_factorization(a))

i=0  #Initial Iteration
while i<n:
      q,r = qr_factorization(a)
      a = np.dot(r,q)
      i=i+1
      print('\n Iteration Number : ',i)
      print(a)
eigenvalues = a.diagonal()

print("Eigenvalues of the given matrix : ",eigenvalues)

##OUTPUT:
##How many iterations you want?10
##
## Given Matrix : A =  [[0 2]
## [2 3]]
##
## QR Factorization of A :  (array([[0., 1.],
##       [1., 0.]]), array([[2., 3.],
##       [0., 2.]]))
##
## Iteration Number :  1
##[[3. 2.]
## [2. 0.]]
##
## Iteration Number :  2
```

```
##[[ 3.92  0.62]
## [ 0.62 -0.92]]
##
## Iteration Number :  3
##[[ 4.    0.16]
## [ 0.16 -1.  ]]
##
## Iteration Number :  4
##[[ 4.    0.04]
## [ 0.04 -1.  ]]
##
## Iteration Number :  5
##[[ 4.    0.01]
## [ 0.01 -1.  ]]
##
## Iteration Number :  6
##[[ 4.  0.]
## [ 0. -1.]]
##
## Iteration Number :  7
##[[ 4.  0.]
## [ 0. -1.]]
##
## Iteration Number :  8
##[[ 4.  0.]
## [ 0. -1.]]
##
## Iteration Number :  9
##[[ 4.  0.]
## [ 0. -1.]]
##
## Iteration Number :  10
##[[ 4.  0.]
## [ 0. -1.]]
##Eigenvalues of the given matrix :  [ 4. -1.]
```