In [6]:
```python
import io
import requests
import itertools
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA,TruncatedSVD
import matplotlib.pyplot as plt
%matplotlib inline
```

Problem 1

In [4]:
```python
# Get data
url = "https://statweb.stanford.edu/~tibs/ElemStatLearn/datasets/zip.digits/train.0"
source = requests.get(url)
X = np.array([np.array(x[:-1].split(',')).astype('float32')
for x in io.StringIO(source.content.decode('utf-8')).readlines()])
```

In [5]:
```python
# Compute sample mean and centralize
sample_mean = X.mean(axis=0)
tilde_X = X - sample_mean
```

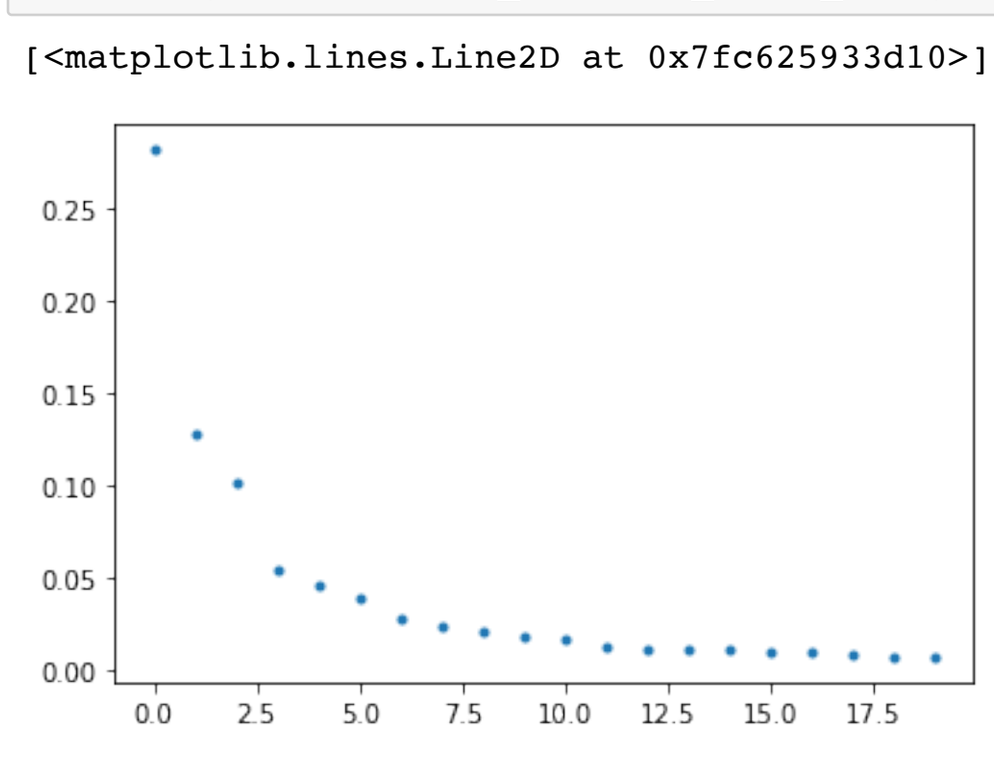
In [7]:
```python
# Compute the top $k$ components
k = 20
pca = PCA(n_components=20)
pca.fit(tilde_X)
```

Out[7]: PCA(n_components=20)

Plot eigenvalues

In [8]:
```python
plt.plot(pca.explained_variance_ratio_,'.')
```

Out[8]: [<matplotlib.lines.Line2D at 0x7fc625933d10>]

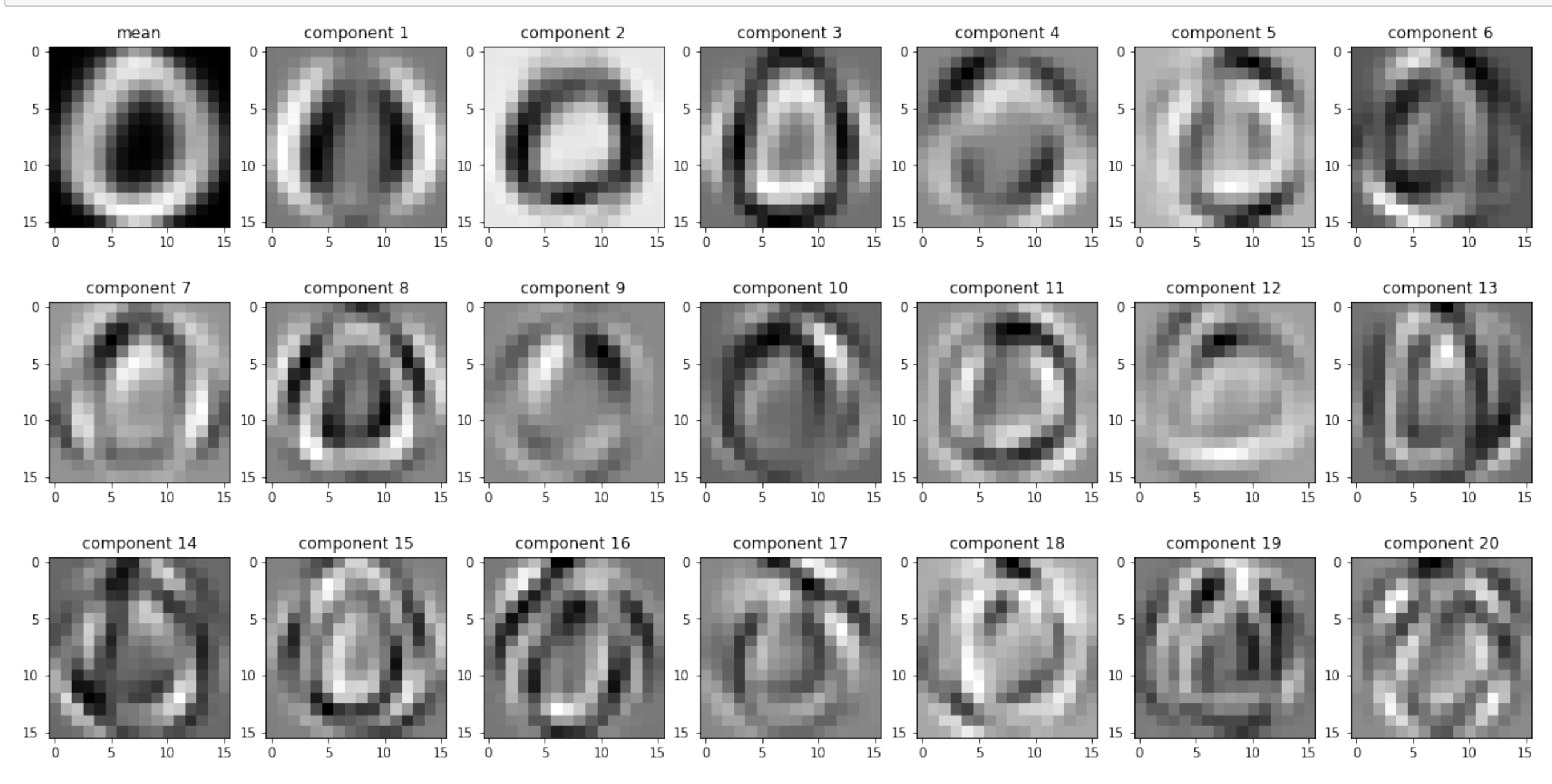


Show principle components

In [9]:
```python
U = pca.components_
U.shape
```

Out[9]: (20, 256)

In [11]:
```python
fig,ax=plt.subplots(3, 7, figsize=(20, 10))
ax[0][0].imshow(sample_mean.reshape(16, -1),cmap='gray')
ax[0][0].set_title('mean')
for j in range(k):
    row = (j + 1) // 7
    col = (j + 1) % 7
    ax[row][col].imshow(U[j,:].reshape(16, -1),cmap='gray')
    ax[row][col].set_title('component '+str(j+1))
```



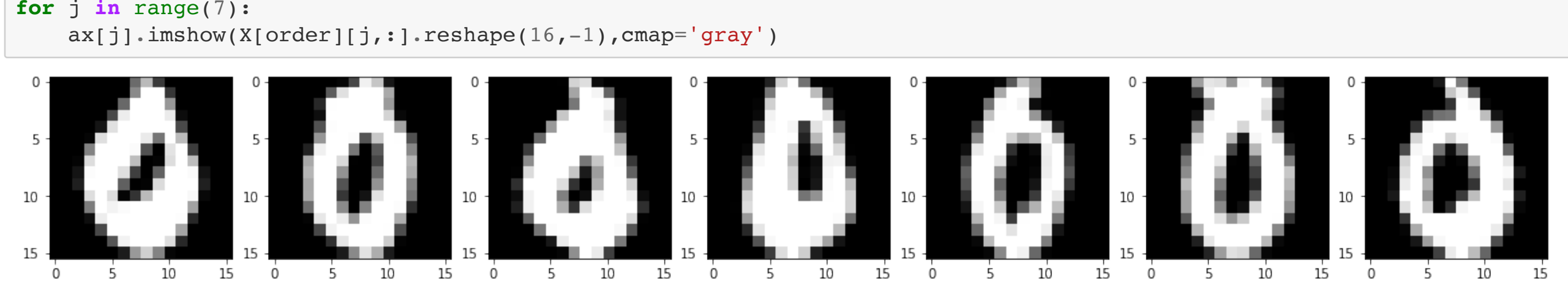
Ordering image according to the fisrt component

In [12]:
```python
V = pca.transform(X)
order = np.argsort(V[:,0])
```

In [13]:
```python
X[order].shape
```

Out[13]: (1194, 256)

In [14]:
```python
fig,ax=plt.subplots(1,7,figsize=(20,10))
for j in range(7):
    ax[j].imshow(X[order][j,:].reshape(16,-1),cmap='gray')
```

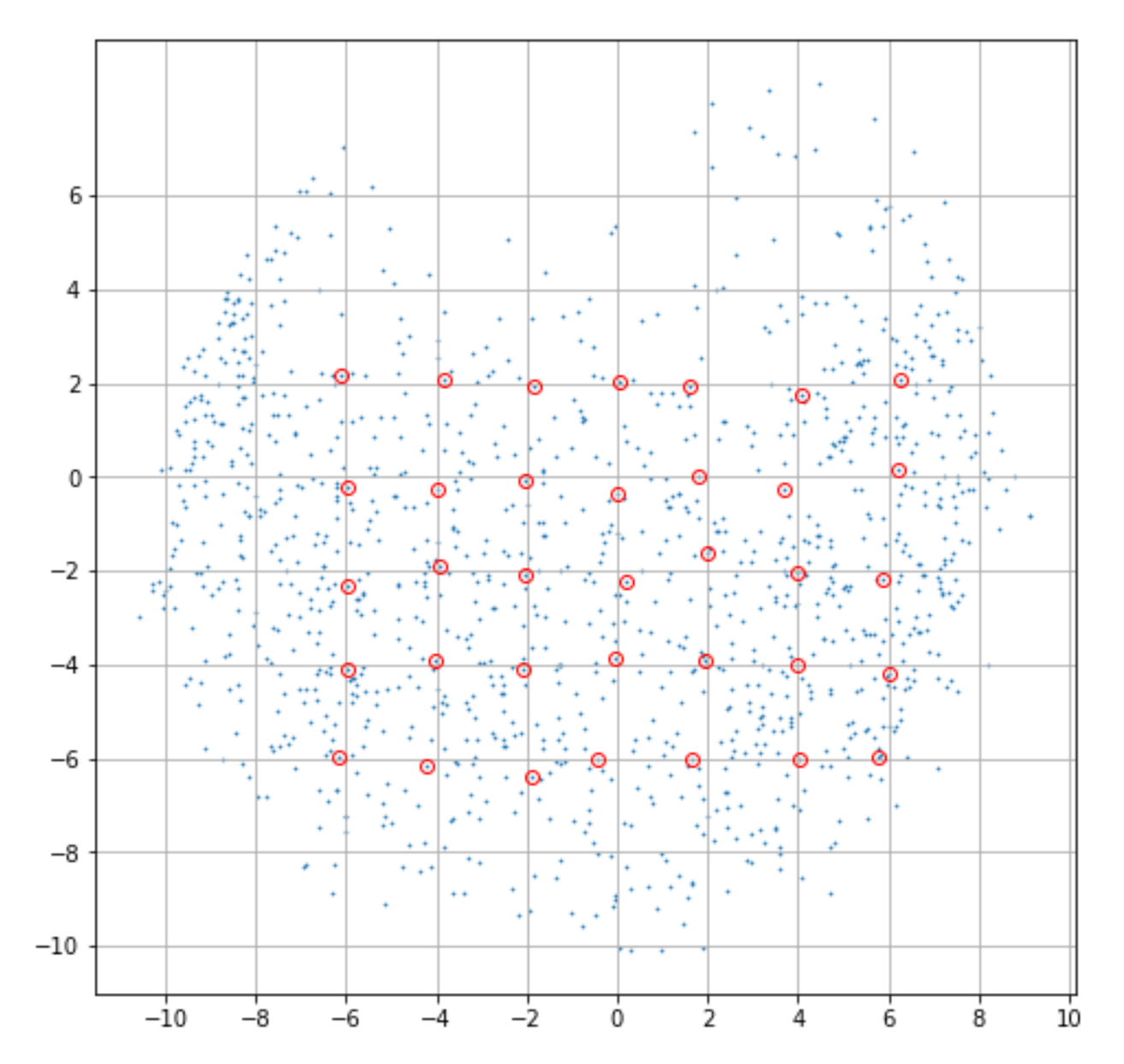


Draw the scatter plot of the first two components

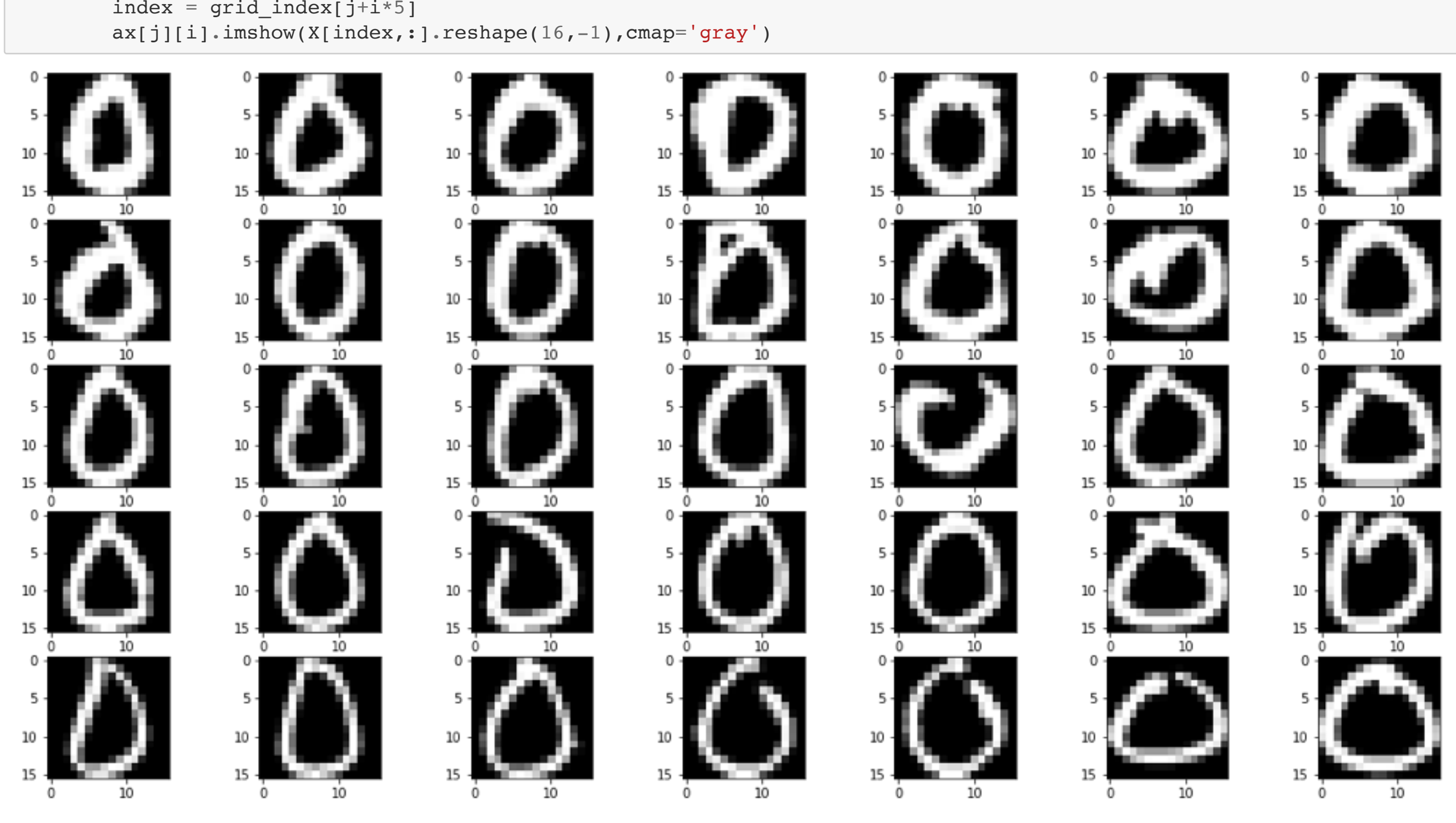
In [15]:
```python
grid_x = np.arange(-6,8,2)
grid_y = np.arange(-6,4,2)
grid_index = [
    np.argmin(np.abs(V[:,0]-x)+np.abs(V[:,1]-y))
    for x in grid_x
    for y in grid_y
]
```

In [16]:
```python
fig,ax = plt.subplots(1,figsize=(8,8))
ax.set_xticks(np.arange(-10,12,2))
ax.set_yticks(np.arange(-10,8,2))
ax.grid()
ax.plot(V[grid_index,0],V[grid_index,1],'ro',mfc='none')
ax.scatter(V[:,0],V[:,1],s=0.5)
```

Out[16]: <matplotlib.collections.PathCollection at 0x7fc624fb9f50>



In [17]:
```python
fig,ax = plt.subplots(5,7,figsize=(20, 10))
for i in range(7):
    for j in range(4,-1,-1):
        index = grid_index[j+i*5]
        ax[j][i].imshow(X[index,:].reshape(16,-1),cmap='gray')
```



Problem 2

(a)

Defines the cities to calculate pairwise distances.

In [25]:
```python
cities = ["Hong Kong", "Shanghai", "New York", "Philadelphia", "London", "Tokyo", "Moscow"]
```

Defines functions to calculate distances between two cities.

In [43]:
```python
from geopy.geocoders import Nominatim

geolocator = Nominatim(user_agent="Safari")

def get_coordinates(city):
    loc = geolocator.geocode(city)
    return (loc.latitude, loc.longitude)

def get_distance(city1, city2):
    coo1 = get_coordinates(city1)
    coo2 = get_coordinates(city2)
    print("Getting distance between", city1, "and", city2 + ".")
    return geodesic(coo1, coo2).km
```

Compute matrix D, the matrix of squares of distances.

In [103]:
```python
D = pd.DataFrame(columns=cities, index=cities, dtype=float)
```

In [104]:
```python
for pair in itertools.product(cities, cities):
    D.loc[pair[0], pair[1]] = get_distance(pair[0], pair[1]) ** 2
```

```
Getting distance between Hong Kong and Hong Kong.
Getting distance between Hong Kong and Shanghai.
Getting distance between Hong Kong and New York.
Getting distance between Hong Kong and Philadelphia.
Getting distance between Hong Kong and London.
Getting distance between Hong Kong and Tokyo.
Getting distance between Hong Kong and Moscow.
Getting distance between Shanghai and Hong Kong.
Getting distance between Shanghai and Shanghai.
Getting distance between Shanghai and New York.
Getting distance between Shanghai and Philadelphia.
Getting distance between Shanghai and London.
Getting distance between Shanghai and Tokyo.
Getting distance between Shanghai and Moscow.
Getting distance between New York and Hong Kong.
Getting distance between New York and Shanghai.
Getting distance between New York and New York.
Getting distance between New York and Philadelphia.
Getting distance between New York and London.
Getting distance between New York and Tokyo.
Getting distance between New York and Moscow.
Getting distance between Philadelphia and Hong Kong.
Getting distance between Philadelphia and Shanghai.
Getting distance between Philadelphia and New York.
Getting distance between Philadelphia and Philadelphia.
Getting distance between Philadelphia and London.
Getting distance between Philadelphia and Tokyo.
Getting distance between Philadelphia and Moscow.
Getting distance between London and Hong Kong.
Getting distance between London and Shanghai.
Getting distance between London and New York.
Getting distance between London and Philadelphia.
Getting distance between London and London.
Getting distance between London and Tokyo.
Getting distance between London and Moscow.
Getting distance between Tokyo and Hong Kong.
Getting distance between Tokyo and Shanghai.
Getting distance between Tokyo and New York.
Getting distance between Tokyo and Philadelphia.
Getting distance between Tokyo and London.
Getting distance between Tokyo and Tokyo.
Getting distance between Tokyo and Moscow.
Getting distance between Moscow and Hong Kong.
Getting distance between Moscow and Shanghai.
Getting distance between Moscow and New York.
Getting distance between Moscow and Philadelphia.
Getting distance between Moscow and London.
Getting distance between Moscow and Tokyo.
Getting distance between Moscow and Moscow.
```

In [105]:
```python
D
```

Out[105]:

	Hong Kong	Shanghai	New York	Philadelphia	London	Tokyo	Moscow
Hong Kong	0.000000e+00	1.510189e+06	1.684748e+08	1.702414e+08	9.298708e+07	8.350453e+06	5.116304e+07
Shanghai	1.510189e+06	0.000000e+00	1.411736e+08	1.425439e+08	8.495411e+07	3.124169e+06	4.666402e+07
New York	1.684748e+08	1.411736e+08	0.000000e+00	1.679663e+04	3.119504e+07	1.181759e+08	5.672405e+07
Philadelphia	1.702414e+08	1.425439e+08	1.679663e+04	0.000000e+00	3.265906e+07	1.188167e+08	5.862366e+07
London	9.298708e+07	8.495411e+07	3.119504e+07	3.265906e+07	0.000000e+00	9.188095e+07	6.293216e+06
Tokyo	8.350453e+06	3.124169e+06	1.181759e+08	1.188167e+08	9.188095e+07	0.000000e+00	5.627026e+07
Moscow	5.116304e+07	4.666402e+07	5.672405e+07	5.862366e+07	6.293216e+06	5.627026e+07	0.000000e+00

(b)

In [174]:
```python
def MDS(D, k):
    n = D.shape[0]
    # Compute centering matrix
    H = np.identity(n) -1 / n * np.ones((n, n))
    # Compute B
    B = -1/2 * np.matmul(np.matmul(H, (D**2).to_numpy()), H.T)
    # Compute eigenvalues and eigenvectors in descending order
    ei_val, ei_vec = np.linalg.eigh(B)
    ei_val = np.flip(ei_val)
    ei_vec = np.flip(ei_vec)
    # Diag matrix of eigenvalues
    Lambda = np.diag(ei_val[:k])
    return np.matmul(ei_vec[:k].T, Lambda ** (1/2)), ei_val
```

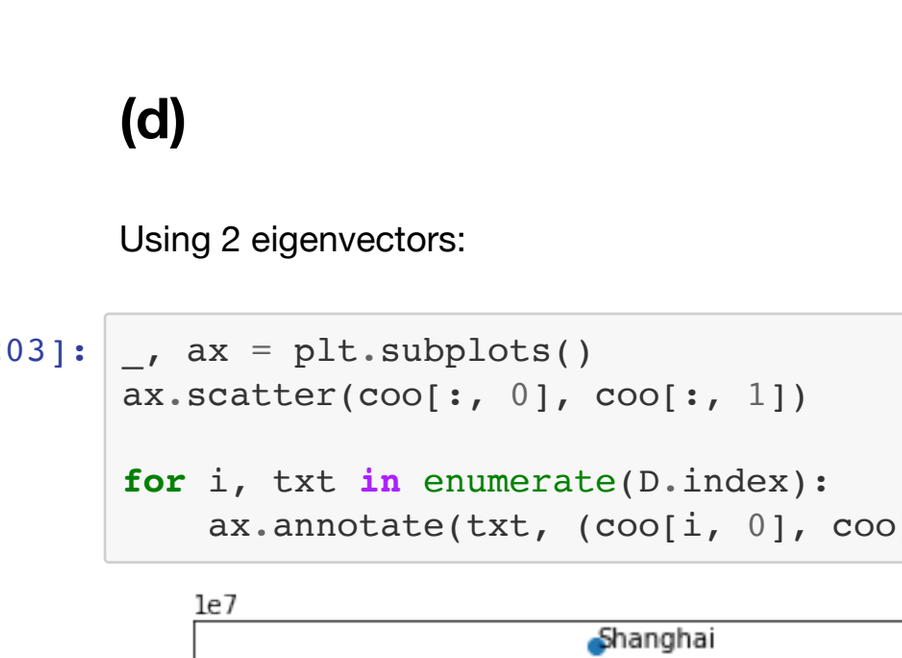
In [177]:
```python
coo, ei_val = MDS(D, 2)
coo
```

Out[177]:
```
array([[ 1.61710141e+06, -1.21219983e+07],
       [-3.21773503e+07,  3.24728923e+07],
       [ 6.25013722e+07,  1.73371217e+07],
       [ 6.54968615e+06,  6.73010899e+05],
       [ 1.85104807e+07,  1.36983389e+07],
       [-1.27058916e+08,  1.20612777e+07],
       [ 7.66226322e+07,  1.63844836e+07]])
```

(c)

In [183]:
```python
plt.plot(ei_val / ei_val.sum())
```

Out[183]: [<matplotlib.lines.Line2D at 0x7fecbf678c10>]



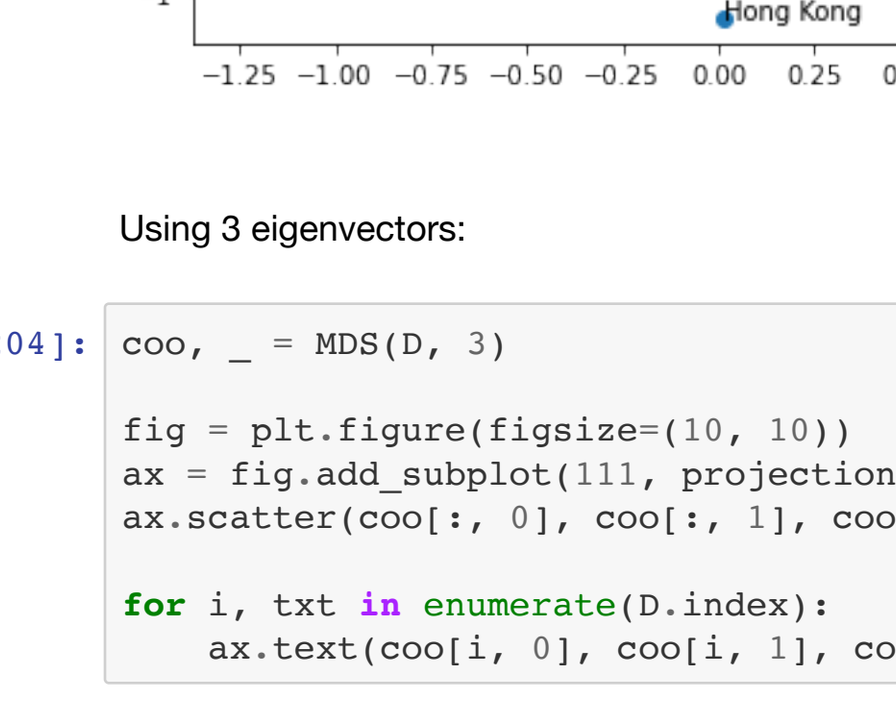
There are 4 negative eigenvalues, which means that the original distances are not Euclidean.

(d)

Using 2 eigenvectors:

In [203]:
```python
_, ax = plt.subplots()
ax.scatter(coo[:, 0], coo[:, 1])

for i, txt in enumerate(D.index):
    ax.annotate(txt, (coo[i, 0], coo[i, 1]))
```

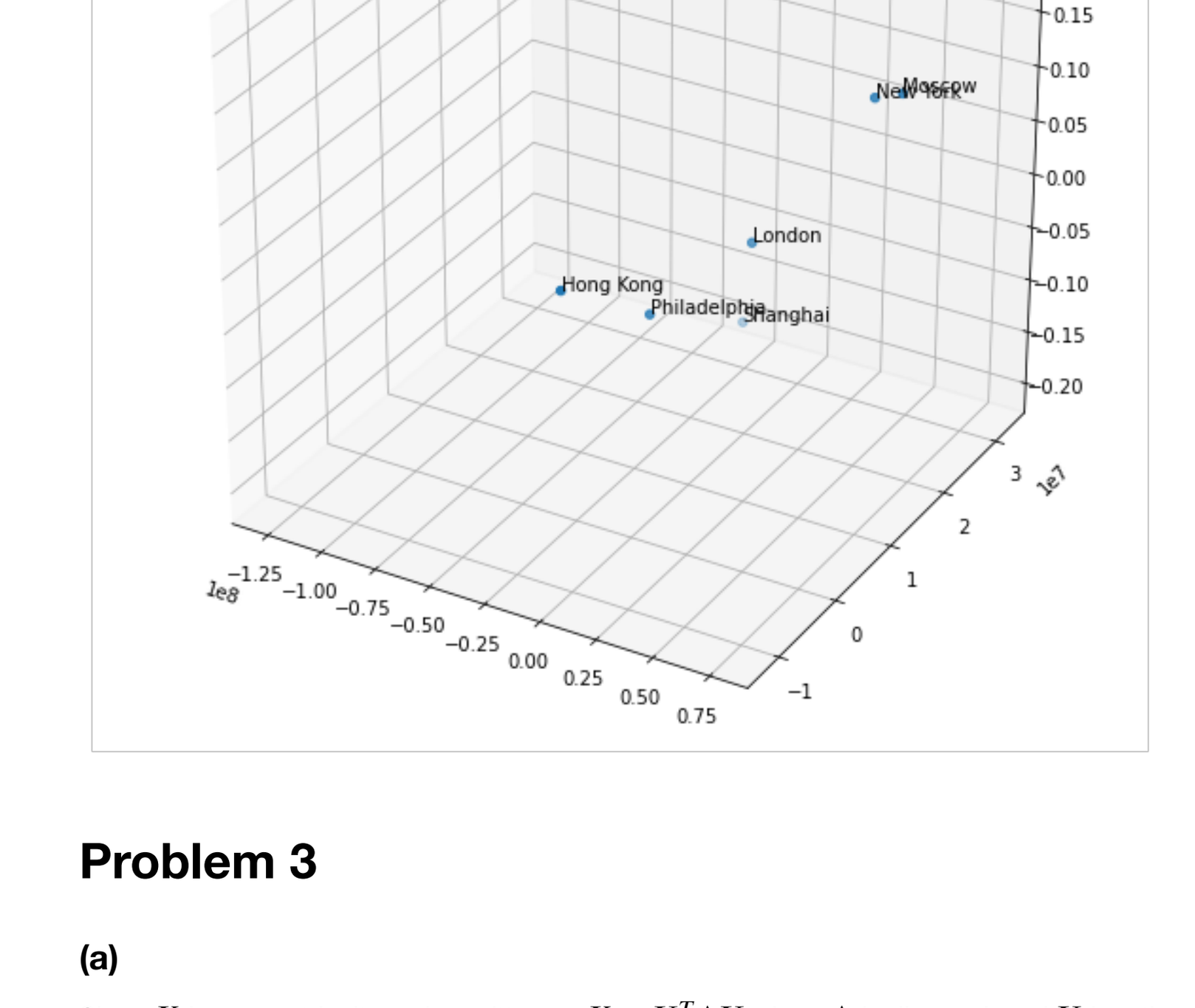


Using 3 eigenvectors:

In [204]:
```python
coo, _ = MDS(D, 3)

fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(coo[:, 0], coo[:, 1], coo[:, 2])

for i, txt in enumerate(D.index):
    ax.text(coo[i, 0], coo[i, 1], coo[i, 2], txt)
```



Problem 3

(a)

Since K is symmetric, it can be written as $K = U^T \Lambda U$ where Λ is diagonal, and U is orthogonal.

The elements on the diagonal of Λ can be written as $\Lambda_{ii} = (U^T e_i)^T K (U^T e_i)$, where e_i is the vector with the i'th value being 1 and others being zeros. We then have $x^T K x = \sum_i \Lambda_{ii} (e_i^T U x)^2$. Thus K is positive semi-definite if and only if all Λ_{ii} are nonnegative.

(b)

$K_{ij} = e_i^T K e_j \Rightarrow d_{ij} = (e_i - e_j)^T K (e_i - e_j)$. Then $d_{ij} = \|u_i - u_j\|^2$ where $u_i = diag\left(\left(\sqrt{\Lambda_{ii}}\right)_i\right) U e_i$.

(c)

Let $y = H_a^T x = x - a e^T x$, then $e^T x = 0$.

$y^T D y = \sum_{i,j} y_i y_j (u_i - u_j)^T (u_i - u_j) = 2 \sum_i e^T y y_i u_i^T u_i - 2 \sum_{i,j} y_i y_j u_i^T u_j = -2\left(\sum_i y_i u_i\right)^T \left(\sum_i y_i u_i\right) \leq 0$

(d)

$x^T (A + B) x = x^T A x + x^T B x \geq 0$.

$x^T (A \circ B) x = tr(A\ diag(x) B\ diag(x)) = tr(CC^T)$ where $C = \sqrt{A}\ diag(x) \sqrt{B}$ and is psd $\Rightarrow x^T (A \circ B) x \geq 0$.

Problem 4

(a)

No. Counter example: for Euclidean distance, let $d(x, y) = 1$, $d(x, z) = 2$, and $d(y, z) = 3$, d^2 violates the triangle inequality.

(b)

Yes. Trivial to prove the identity and symmetry properties.

$\sqrt{d(x, y)} \leq \sqrt{d(x, z) + d(z, y)} \leq \sqrt{d(x, z) + d(z, y) + \sqrt{4 * d(x, z) * d(z, y)}} = \sqrt{d(x, z)} + \sqrt{d(z, y)}$ proves the triangle inequality.