In [ ]:
```
%matplotlib inline
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
# Load the data matrix
A = np.loadtxt('mysterious_data.txt')
n,d = A.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
The matrix A contains 3000 points in dimension 1000
```

Out[ ]:
```
(3000, 1000)
```

Each row of A corresponds to a datapoint.

In [ ]:
```
### CENTER YOUR DATA ###
mean_row = A.mean(axis=0)
A_centered = A - mean_row
```

In [ ]:
```
### COMPUTE COVARIANCE ###
S = A.transpose() @ A
```

In [ ]:
```
### COMPUTE EIGEN DECOMP ###
from numpy import linalg as LA
w, v = LA.eigh(S)
## w is eigenvalues
## v is eigenvectors
```

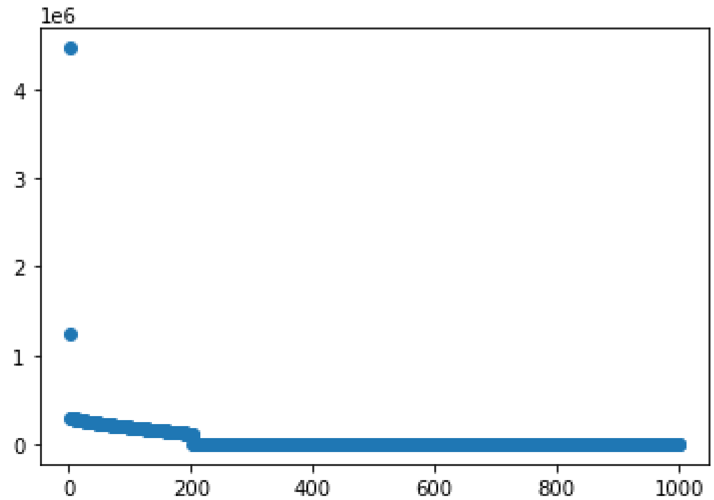
Out[ ]:
```
1243909.9724750144
```

In [ ]:
```
### FIND BEST K ###
import matplotlib.pyplot as plt
x_axis = np.arange(1,1001)

plt.scatter(x_axis, np.flip(w))

## Using method 2 from the notes, we choose k = 2
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x7fb782b359a0>
```



In [ ]:

```
### COMPUTE REDUCED VALUES ####

v1 = v[:,-1]
## We scale our second eigenvector in order to get the plot we want in the end
v2 = v[:,-2] * -1

eigenvectors = [v1,v2]

reduced_A = np.zeros((3000, 2))

i = 0
for i in range(0,3000):
    for j in range(0,2):
        reduced_A[i][j] = np.dot(eigenvectors[j], A_centered[i,:])
```

In [ ]:

```
### PLOT RESULT ###
x, y = reduced_A.T
plt.scatter(x,y)
```

Out[ ]: <matplotlib.collections.PathCollection at 0x7fb71026a460>