In [14]:
```
%matplotlib inline
import matplotlib
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import pandas as pd
```

In [15]:
```
# 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
```

Each row of A corresponds to a datapoint.

In [16]:
```
#Define PCA Algorithm
def pca(data,k=0,var=0):

    n,d = data.shape
    data_copy = data
    #Center Data at 0, Each Column must have mean 0
    #Loop through every column
    for i in range(d):
        data_copy[:,i] = data_copy[:,i] -  np.mean(data_copy[:,i])


    #Compute Covariance Matrix
    Cov_matrix = data_copy.T @ data_copy

    #Calculate eigendecomp for Cov_Matrix
    vals, vectors = np.linalg.eigh(Cov_matrix)

    #Sort the eigenvalues descending order
    vals = vals[::-1]
    vectors = vectors[:,::-1]
    total_var = vals.sum()

    #If we wanted the least dimensions for a certain amount of variance explained

    #See if Var was passed to function and its valid
    if var > 0 and var <=1:
        tracker = 0
        eig_vals_of_interest, eig_vectors_of_interest = [], []
        for i in range(len(vals)):
            if tracker < var:
                tracker += vals[i] / total_var
                eig_vals_of_interest.append(vals[i])
        eig_vectors_of_interest = vectors[:,:len(eig_vals_of_interest)]

    #Check if we just wanted k dimensions
    elif k > 0 and k <= d:

        #If we wanted k dimensions, set the appropriate amount
        eig_vals_of_interest = vals[:k]
        eig_vectors_of_interest = vectors[:,:k]

    else:

        #if no other arguments were passed, return the eigen decomp
        return vals, vectors

    #Make sure data types are compatible
    eig_vectors_of_interest = np.array(eig_vectors_of_interest)
    eig_vals_of_interest = np.array(eig_vals_of_interest)
    percent_of_variance = eig_vals_of_interest.sum() / total_var
    #Compute the Principle Component Matrix
    new_data = data_copy@eig_vectors_of_interest
```

```
    #Return completed pca, eigenvalues, and eigenvectors
    return new_data, eig_vals_of_interest,eig_vectors_of_interest, percent_of_variance
```

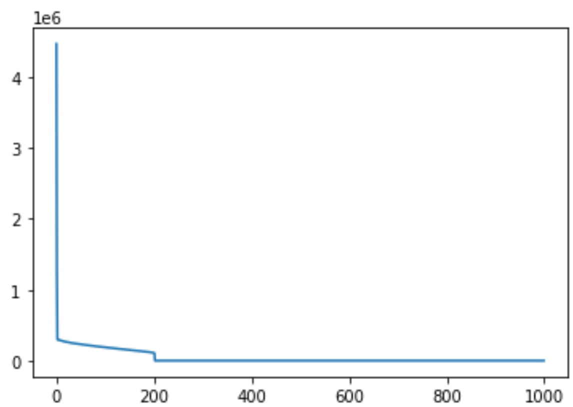
What does the distribution of eigenvalues of the covariance matrix look like?

In [17]:

```
vals, vectors = pca(A)
index = [i for i in range(len(vals))]
df = pd.DataFrame({'vals':vals})
sb.lineplot(index,vals)
print("It looks like the first few eigenvalues explain the majority of the variance")
print("It also appears that after 200 eigenvalues, the values are basically 0")
```

```
It looks like the first few eigenvalues explain the majority of the variance
It also appears that after 200 eigenvalues, the values are basically 0

C:\Users\Giulio\Anaconda3\lib\site-packages\seaborn\_decorators.py:43: FutureWarning: Pass t
he following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  FutureWarning
```



How many principle components would we need to capture 50% of the variance?

In [18]:

```
#Call function, and shape
output, values, vectors,var_percent = pca(A,var=.5)
print("We would need", output.shape[1], "Principle Components")
```

```
We would need 67 Principle Components
```

How many principle components would we need to capture 80% of the variance?

In [19]:

```
#Call function, and shape
output, values, vectors,var_percent = pca(A,var=.8)
print("We would need", output.shape[1], "Principle Components")
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would need 138 Principle Components
The matrix A contains 3000 points in dimension 138
```

How many principle components would we need to capture all of the variance?

In [20]:

```
#Call function, and shape
output, values, vectors,var_percent = pca(A,var=1)
print("We would need", output.shape[1], "Principle Components")
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would need 1000 Principle Components
The matrix A contains 3000 points in dimension 1000
```

What does the 202 Principle Components 100% of the variance dataset look like?

In [21]:
```
df = pd.DataFrame(output)
df
```

Out[21]:

	0	1	2	3	4	5	6	7	8	9	...
0	52.286909	9.834370	0.291642	-9.550960	-5.360088	0.845779	-8.844195	3.416257	-3.595199	17.273449	...
1	-19.311017	27.148588	16.740921	-3.105850	-13.432818	-5.870330	-0.775702	14.039010	-9.678481	20.248258	...
2	-9.624371	-22.207301	20.915428	2.860974	-12.175551	-2.437017	5.323750	-7.498591	1.266475	4.515910	...
3	52.129243	20.519438	14.572979	8.574215	5.743858	-1.218111	2.632488	-3.157580	-16.452796	15.562120	...
4	-68.571024	29.082733	5.485627	13.705943	1.410991	12.654287	2.235758	5.276971	-8.876717	1.297797	...
...	...	...	...	...	...	...	...	...	...	...	...
2995	34.642930	-33.888549	-5.493122	-4.180623	-1.623344	-12.501368	11.759640	-0.110850	-0.296381	0.059233	...
2996	-28.119669	-25.601034	-4.838316	0.617135	3.923323	1.424779	-4.446596	9.295150	-10.511800	-2.395244	...
2997	68.100775	24.906250	2.048599	-7.729703	3.528660	15.116484	13.054016	1.872415	1.670500	1.116577	...
2998	46.352552	-29.236058	10.495395	-5.126592	5.137411	10.984298	-4.400079	-11.196389	-6.420155	-12.158058	...
2999	-19.709776	-12.184919	6.041056	10.553933	13.507182	-7.264965	5.221503	-6.863069	-0.178527	3.854670	...

3000 rows × 1000 columns

How much of the variance does 25 Principle Components Capture?

In [22]:
```
#Call function, and shape
output, values, vectors, var_percent = pca(A,k=25)
print("We would explain", np.round(var_percent*100,decimals=4), "% of the variance with 25 Pri
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would explain 27.4701 % of the variance with 25 Principle Components
The matrix A contains 3000 points in dimension 25
```

How much of the variance does 50 Principle Components Capture?

In [23]:
```
#Call function, and shape
output, values, vectors, var_percent = pca(A,k=50)
print("We would explain", np.round(var_percent*100,decimals=4), "% of the variance with 50 Pri
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would explain 41.3561 % of the variance with 50 Principle Components
The matrix A contains 3000 points in dimension 50
```

How much of the variance does 75 Principle Components Capture?

In [24]:
```
#Call function, and shape
output, values, vectors, var_percent = pca(A,k=75)
print("We would explain", np.round(var_percent*100,decimals=3), "% of the variance with 75 Pri
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would explain 53.88 % of the variance with 75 Principle Components
The matrix A contains 3000 points in dimension 75
```

How much of the variance does 100 Principle Components Capture?

In [25]:
```
#Call function, and shape
output, values, vectors, var_percent = pca(A,k=100)
print("We would explain", np.round(var_percent*100,decimals=3), "% of the variance with 100 Pr
```

```
n,d = output.shape
print(f'The matrix A contains {n} points in dimension {d}')
```

```
We would explain 65.16 % of the variance with 100 Principle Components
The matrix A contains 3000 points in dimension 100
```

In [ ]:

In [ ]:

In [ ]: