```
import numpy as np
import pandas as pd
import networkx as nx
```

Entering in Data

```
X = pd.DataFrame([[1,-3.5],[2,-5.5],[3,-12]],columns=["x1","x2"])
X
```

	x1	x2
0	1	-3.5
1	2	-5.5
2	3	-12.0

1 - Finding Covariance Matrix

```
#averaging
x1_av = X['x1'].mean()
x2_av = X["x2"].mean()

#finding variances
var_x1 = ((X["x1"] - x1_av) ** 2).sum() / (len(X) - 1)
var_x2 = ((X["x2"] - x2_av) ** 2).sum() / (len(X) - 1)

#finding covariance
cov = ((X["x2"] - x2_av) * (X["x1"] - x1_av)).sum() / (len(X) - 1)

#putting it all together
data = {"x1": [var_x1, cov], "x2": [cov, var_x2]}
A = pd.DataFrame(data, columns=["x1", "x2"])

#displaying answer
A
```

	x1	x2
0	1.00	-4.25
1	-4.25	19.75

2 - Finding EigenValues

✓ 0s completed at 11:04 AM

```
#grabbing components
a = A.iloc[0,0]
b = A.iloc[0,1]
c = A.iloc[1,0]
d = A.iloc[1,1]

#getting coeffcients
coef1 = 1
coef2 = -1 * (a + d)
coef3 = ((a * d) - (b * c))

#using quadratic formula
eval1 = ((-1 * coef2) + np.sqrt(coef2 ** 2 - (4 * coef1 * coef3))) / (2 * coef1)
eval2 = ((-1 * coef2) - np.sqrt(coef2 ** 2 - (4 * coef1 * coef3))) / (2 * coef1)

#displaying answer
eval1, eval2
```

```
(20.66835343802009, 0.08164656197991071)
```

checking with numpy:

```
np.linalg.eig(A.values)
```

```
(array([ 0.08164656, 20.66835344]), array([[-0.97744102,  0.21120855],
        [-0.21120855, -0.97744102]]))
```

Sweet it worked!

3 - Finding Variance Explained

```
#calculating
variance_explained1 = eval1 / (eval1 + eval2)
variance_explained2 = eval2 / (eval1 + eval2)

#displaying
variance_explained1, variance_explained2
```

```
(0.9960652259286791, 0.003934774071320998)
```

4 finding the Eigen vector

4, finding the Eigen vector

Lets always assume that the first component of the egein vector is one. That is $v_1 = 0$ For the first equation, we have

$$A_{00} * 1 + A_{01} * v_2 = 0$$

then

$$v_2 = \frac{-A_{00} * v_1}{A_{01}}$$

```
def get_eigen_vector(A, eigen_val):
    A = A.copy().to_numpy()
    A_i = A - (np.identity(2) * eigen_val)
    v_1 = 1
    v_2 = (-1 * A_i[0][0] * v_1) / A_i[0][1]

    #normalzing
    v = pd.Series([v_1, v_2])
    v = v / np.sqrt((v ** 2).sum())
    return v
```

```
evec_1 = get_eigen_vector(A, eval1)
evec_2 = get_eigen_vector(A, eval2)
print(evec_1)
print(evec_2)
```

```
0    0.211209
1   -0.977441
dtype: float64
0    0.977441
1    0.211209
dtype: float64
```

Sweet this matches the numpy output!

5 - Applying onto Data

```
#Since eval1 is the biggest eigenvalue, we'll use that one
#all we have to do is take the dot product of X and our EigenVector
ans5 = np.dot(X, evec_1)
#displaying
ans5
```

```
array([ 3.63225212,  5.79834271, 12.36291789])
```