In [1]:
```
import numpy as np
import pylab
import scipy.stats as stats
```

In [2]:
```
def genarate_normal_data():
    return np.random.randn(1000)
```

In [3]:
```
def genarate_lognormal_data():
    return np.random.lognormal(0,1,1000)
```

use the functions genarate_normal_data, genarate_lognormal_data to get the two 1-d data sets.

ex: *normaldata = genarate_normal_data()*, *logdata = genarate_lognormal_data()*

Q1.

1. Plot the Q-Q plot between the normaldata (N) and logdata (L)
2. Find the covariance between N and L vectors
   - try to plot datpoints (N(i),L(i)) try to get the relation
   - use inbuilt functions to get this value
3. Do 1, 2 for Normalized vectors of N and L
4. Do 1, 2 for Standardized vectors of N and L

Q2.

1. Prove that the E[(X-μ)^2] = σ^2
2. Prove that the Expectation of a randam varaible X~N(μ,σ) is equal to μ

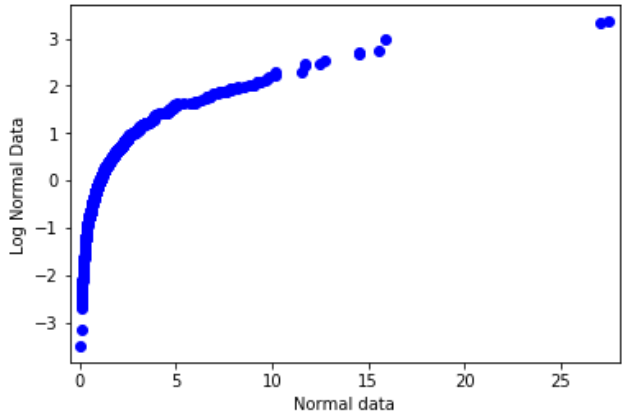
Plot the Q-Q plot between the normaldata (N) and logdata (L)

In [4]:
```
norm_data = genarate_normal_data()
log_normal_data = genarate_lognormal_data()
```

In [5]:
```
import statsmodels.api as sm
```

```
/home/ram/anaconda3/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56:
FutureWarning: The pandas.core.datetools module is deprecated and will be removed
in a future version. Please use the pandas.tseries module instead.
  from pandas.core import datetools
```

In [6]:
```
sm.qqplot_2samples(data1=norm_data, data2=log_normal_data, xlabel='Normal data
pylab.show()
```



In [7]:
```
1 covarience = np.cov(norm_data, log_normal_data)
2 covarience
```

Out[7]:
```
array([[ 1.08163304, -0.06347222],
       [-0.06347222,  5.36985925]])
```

plot the covarience

import matplotlib.pyplot as plt

plt.plot(covarience[0], covarience[1]) plt.show()

Normalized vectors of N and L

In [9]:
```
1 # normalization of norm_data
2 normalized_norm_data = (norm_data - norm_data.mean())/(norm_data.max()-norm_da
3 normalized_norm_data[:10]
```

Out[9]:
```
array([ 0.20516115, -0.18587064, -0.09360196, -0.03439995, -0.364173  ,
       -0.19756107, -0.21758882,  0.20356908, -0.06257045,  0.21051667])
```

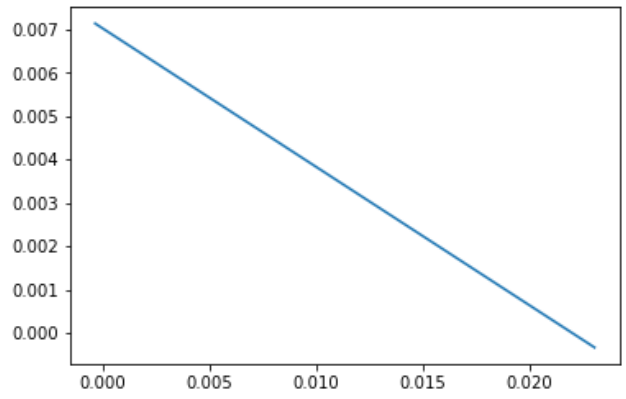
In [10]:
```
1 # normalization of log__normal data
2 normalized_log_normal_data = (log_normal_data - log_normal_data.mean())/(log_n
3 normalized_log_normal_data[:10]
```

Out[10]:
```
array([-0.0499916 , -0.00233615,  0.12249208,  0.06314359, -0.0406894 ,
       -0.0275636 ,  0.08223146, -0.04302253,  0.22016245, -0.04207334])
```

In [11]:
```
1 # plot covarience between these normalized vectors
2 covarience = np.cov(normalized_norm_data, normalized_log_normal_data)
3 covarience
```

Out[11]:
```
array([[ 0.02302004, -0.00033748],
       [-0.00033748,  0.00713269]])
```

In [12]:
```
1 # plot the covarience
2 plt.plot(covarience[0], covarience[1])
3 plt.show()
```



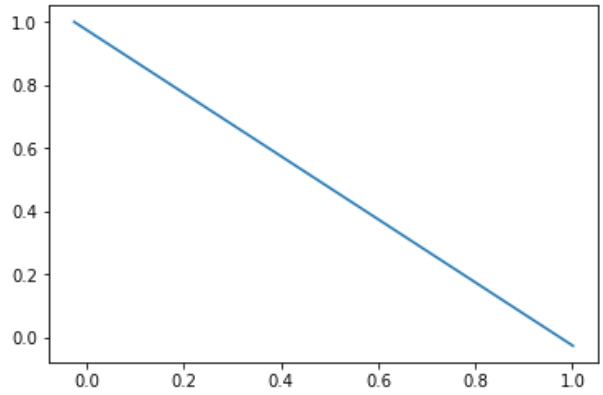
Standardized vectors of N and L

In [13]:
```
1 std_norm_data = (norm_data - np.mean(norm_data))/np.std(norm_data)
2 std_log_norm_data = (log_normal_data - np.mean(log_normal_data))/np.std(log_no
```

In [14]:
```
#find covarience
covariance = np.cov(std_norm_data, std_log_norm_data)
covarience
```

Out[14]:
```
array([[ 0.02302004, -0.00033748],
       [-0.00033748,  0.00713269]])
```

In [15]:
```
 # plot the covarience plot
plt.plot(covariance[0], covariance[1])
plt.show()
```



In [ ]:
```
1
```