Using Monte Carlo Simulation to predict the future stock of Fortum company in Finland

In [1]:

```
import numpy as np
import pandas as pd
import pandas_datareader.data as web
import datetime
from scipy.stats import norm
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

```
C:\Users\chumj\Anaconda3\Ben\lib\site-packages\pandas_datareader\compat\__init__.py:7:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at
pandas.testing instead.
  from pandas.util.testing import assert_frame_equal
```

In [2]:

```
start=datetime.datetime (2013,1,1)
end=datetime.datetime (2020,9,9)
```

We will work from the framework of Drift and Volatility.

Drift,is the best approximation of future return of a stock.

Drift=U-0.5.var or Drift=υ-0.5σ^2

Expected daily returns of stock.

U=mean,var=variance.

Volatity=Random variable

Random variable=σxZ(Rand(0;1))

Browian motion,

r=Drift + std*e^r

Price Today=Price Yesterday 〖 xe〗 ^((U-0.5σ^(2))+σz[Rand(0;1)])

In [3]:

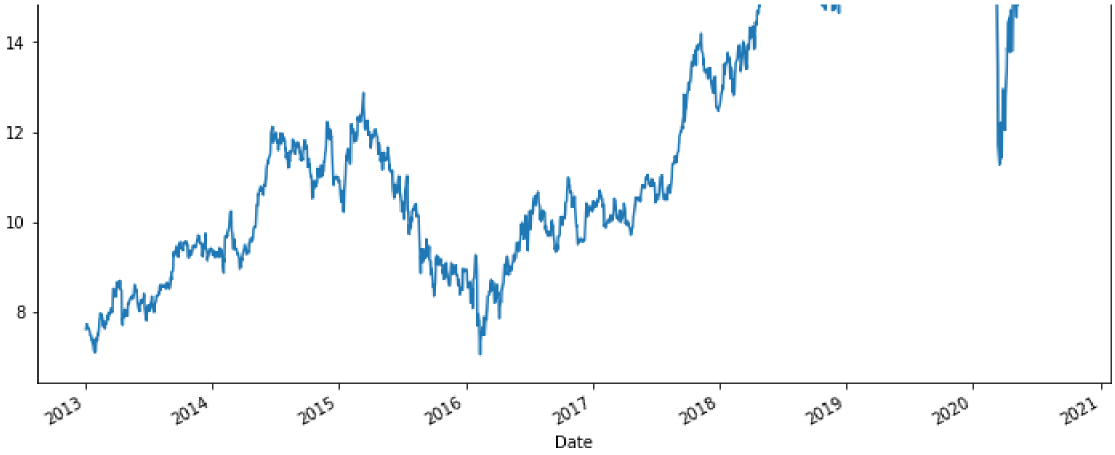
```
Fortum=web.DataReader('FORTUM.HE','yahoo',start,end)['Adj Close']
```

In [4]:

```
Fortum.plot(figsize=(12,8))
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c874350388>
```



In [5]:

```
log_ret=np.log(Fortum/Fortum.shift(1))
```

In [6]:

```
log_ret.head(4)
```

Out[6]:

```
Date
2013-01-02         NaN
2013-01-03    0.006248
2013-01-04    0.008956
2013-01-07   -0.008264
Name: Adj Close, dtype: float64
```

In [7]:

```
U=log_ret.mean()
U
```

Out[7]:

```
0.00042350135239911456
```

In [8]:

```
var=log_ret.var()
var
```

Out[8]:

```
0.00023686545496048953
```

In [9]:

```
std=log_ret.std()
std
```

Out[9]:

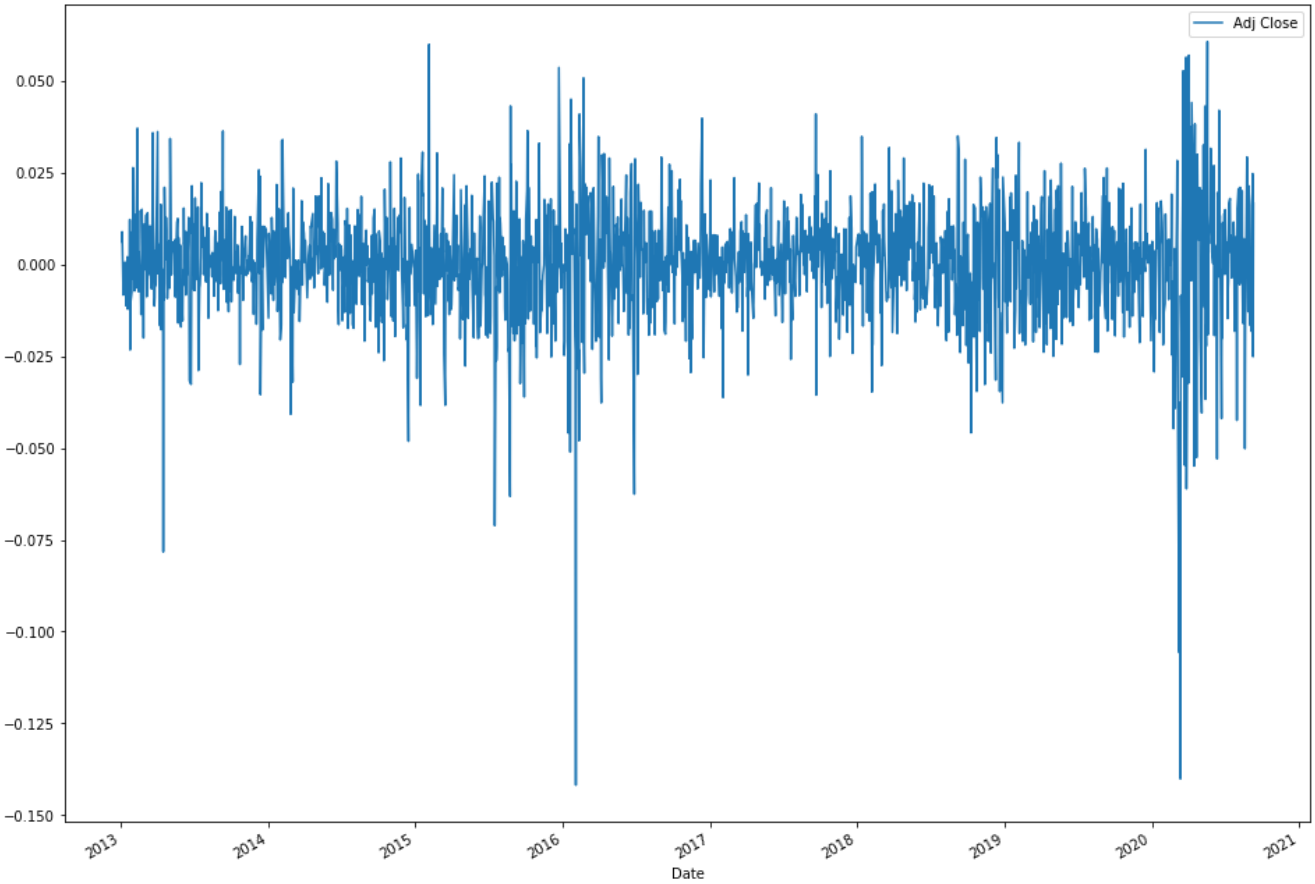
```
0.015390433878 24039
```

In [10]:

```
log_ret.plot(figsize=(16,12),legend=True)
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c87462e408>
```

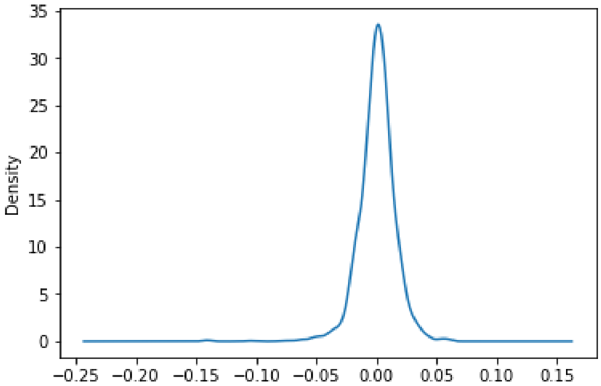


In [11]:

```
log_ret.plot(kind='kde')
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c87483b488>
```



In [12]:

```
drift=U-(0.5*var)
drift
```

Out[12]:

```
0.0003050686249188698
```

In [13]:

```
drif=np.array(drift)
drif
```

Out[13]:

```
array(0.00030507)
```

In [ ]:

```

```

In [14]:

```
sd=np.array(std)
sd
```

Out[14]:

```
array(0.01539043)
```

In [15]:

```
norm.ppf(0.95)
```

Out[15]:

```
1.6448536269514722
```

In [16]:

```
x=np.random.rand(10,2)
x
```

Out[16]:

```
array([[0.44018166, 0.8172463 ],
       [0.0983318 , 0.35456285],
       [0.64422325, 0.72474606],
       [0.10640619, 0.02144597],
       [0.69115401, 0.58837673],
       [0.83468723, 0.15790488],
       [0.24782871, 0.71538995],
       [0.42095991, 0.91793699],
       [0.62129818, 0.04552043],
       [0.65544681, 0.68556365]])
```

In [17]:

```
norm.ppf(x)
```

Out[17]:

```
array([[-0.15050867,  0.9049207 ],
       [-1.29111561, -0.37303057],
       [ 0.36977049,  0.59699926],
       [-1.24586929, -2.02476074],
       [ 0.49912407,  0.22337129],
       [ 0.97285467, -1.00310594],
       [-0.68133836,  0.56920047],
       [-0.1994384 ,  1.39132788],
       [ 0.30889205, -1.68993255],
       [ 0.40006807,  0.48331414]])
```

In [18]:

```
z=norm.ppf(np.random.rand(10,2))
z
```

Out[18]:

```
array([[-0.80131775, -0.76574802],
       [-0.44878055, -2.32206744],
       [ 2.11363954,  0.90281707],
```

```
       [ 1.33998112,  0.49801245],
       [-0.24215659, -1.70697299],
       [-1.57413964,  1.40093314],
       [-0.49101149,  1.44116878],
       [ 2.1289281 ,  0.66584854],
       [ 0.5418879 ,  0.53453952],
       [ 1.38588859, -1.33839192]])
```

In [19]:

```
t_intervals=350
iterations=10
```

In [20]:

```
daily_returns=np.exp(drif+sd*norm.ppf(np.random.rand(t_intervals,iterations)))
daily_returns
```

Out[20]:

```
array([[0.97927315, 0.9853733 , 1.01275675, ..., 0.99333584, 0.99578463,
        0.99786791],
       [1.0042481 , 1.01263921, 1.00920426, ..., 0.98258402, 1.01758068,
        0.99853448],
       [0.98129996, 1.00176591, 1.01276718, ..., 0.99267426, 0.9937563 ,
        0.98282945],
       ...,
       [1.01374625, 1.00665872, 0.97330718, ..., 0.98760873, 0.96878194,
        0.98438723],
       [1.01309399, 1.0369216 , 0.9733088 , ..., 0.98951514, 1.01340701,
        1.02443418],
       [0.98014154, 0.97437733, 0.99106171, ..., 1.02068582, 0.99987536,
        1.03569693]])
```

In [21]:

```
so=Fortum.iloc[-1]
so
```

Out[21]:

```
17.21500015258789
```

In [22]:

```
price_list=np.zeros_like(daily_returns)
price_list
```

Out[22]:

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [23]:

```
price_list[0]=so
price_list
```

Out[23]:

```
array([[17.21500015, 17.21500015, 17.21500015, ..., 17.21500015,
        17.21500015, 17.21500015],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
       [ 0.        ,  0.        ,  0.        , ...,  0.        ,
         0.        ,  0.        ],
```

```
        ...,
        [ 0.        ,  0.        ,  0.        , ...,  0.        ,
          0.        ,  0.        ],
        [ 0.        ,  0.        ,  0.        , ...,  0.        ,
          0.        ,  0.        ],
        [ 0.        ,  0.        ,  0.        , ...,  0.        ,
          0.        ,  0.        ]])
```

In [24]:

```
for t in range(1,t_intervals):
    price_list[t]=price_list[t-1]*daily_returns[t]
```

In [25]:

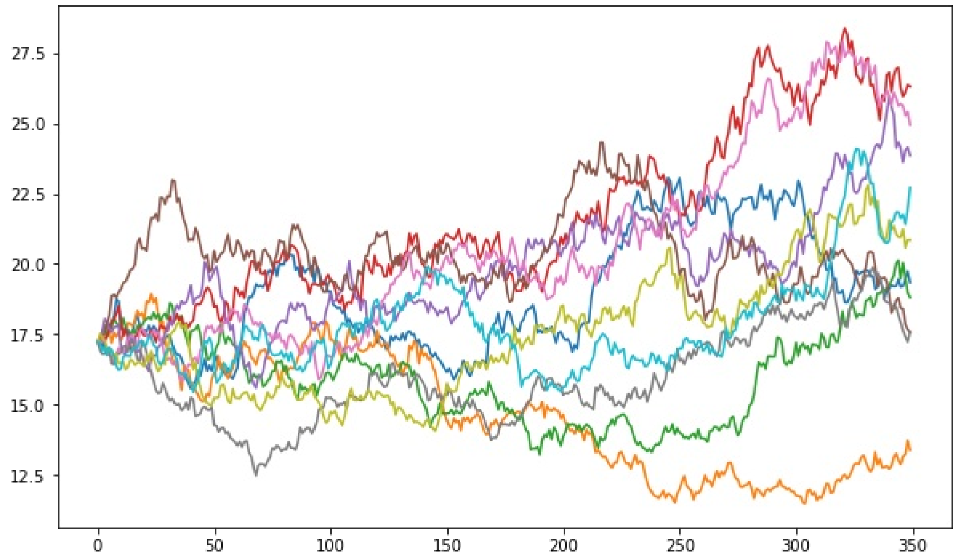
```
price_list
```

Out[25]:

```
array([[17.21500015, 17.21500015, 17.21500015, ..., 17.21500015,
         17.21500015, 17.21500015],
        [17.28813127, 17.43258424, 17.37345149, ..., 16.9151841 ,
         17.51765148, 17.1897713 ],
        [16.96484252, 17.46336863, 17.59526153, ..., 16.79126782,
         17.40827652, 16.89461341],
        ...,
        [19.47535882, 13.2362082 , 19.49643457, ..., 17.40520395,
         20.56946346, 21.4040791 ],
        [19.730369  , 13.7249102 , 18.97605129, ..., 17.22271286,
         20.8452385 , 21.92707022],
        [19.33855425, 13.37324134, 18.8064379 , ..., 17.57897873,
         20.84264042, 22.70979938]])
```

In [26]:

```
plt.figure(figsize=(10,6))
plt.plot(price_list);
```



In [ ]: