Sharpe Ratio and Portfolio Values

In [2]:
```
import pandas as pd
```

In [1]:
```
#import quandl
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import yfinance as yf
```

Create a Portfolio

In [3]:
```
start = pd.to_datetime('2012-01-01')
end = pd.to_datetime('2017-01-01')
```

In [4]:
```
# Grabbing a bunch of tech stocks for our portfolio
#aapl = quandl.get('WIKI/AAPL.11',start_date=start,end_date=end)
#cisco = quandl.get('WIKI/CSCO.11',start_date=start,end_date=end)
#ibm = quandl.get('WIKI/IBM.11',start_date=start,end_date=end)
#amzn = quandl.get('WIKI/AMZN.11',start_date=start,end_date=end)
```

In [3]:
```
# Alternative to reading the csv
# Quandl - try for HW, I use yf mostly nowadays
aapl = yf.download("AAPL", start="2012-01-01", end="2017-01-01")
csco = yf.download("CSCO", start="2012-01-01", end="2017-01-01")
amzn = yf.download("AMZN", start="2012-01-01", end="2017-01-01")
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

In [5]:
```
# Alternative
aapl = pd.read_csv('AAPL_CLOSE',index_col='Date',parse_dates=True)
cisco = pd.read_csv('CISCO_CLOSE',index_col='Date',parse_dates=True)
ibm = pd.read_csv('IBM_CLOSE',index_col='Date',parse_dates=True)
amzn = pd.read_csv('AMZN_CLOSE',index_col='Date',parse_dates=True)
```

In [6]:
```
# in case you want to save to csv from data frame for reference purpose
aapl.to_csv('AAPL_CLOSE')
cisco.to_csv('CISCO_CLOSE')
ibm.to_csv('IBM_CLOSE')
amzn.to_csv('AMZN_CLOSE')
```

Normalize Prices

This is the same as cumulative daily returns

In [6]:
```
# Example
aapl.iloc[0]['Adj. Close']
```

Out[6]:
```
53.063217800141
```

In [7]:
```
for stock_df in (aapl,cisco,ibm,amzn):
    stock_df['Normed Return'] = stock_df['Adj. Close']/stock_df.iloc[0]['Adj. Close']
```

In [9]:
```
aapl.head()
```

Out[9]:

Date	Adj. Close	Normed Return
2012-01-03	53.063218	1.000000
2012-01-04	53.348386	1.005374
2012-01-05	53.940658	1.016536
2012-01-06	54.504543	1.027162
2012-01-09	54.418089	1.025533

In [10]:
```
aapl.tail()
```

Out[10]:

Date	Adj. Close	Normed Return
2016-12-23	115.547742	2.177549
2016-12-27	116.281568	2.191378
2016-12-28	115.785740	2.182034
2016-12-29	115.755990	2.181473
2016-12-30	114.853583	2.164467

Allocations

Let's pretend we had the following allocations for our total portfolio:

- 30% in Apple
- 20% in Google/Alphabet
- 40% in Amazon
- 10% in IBM

Let's have these values be reflected by multiplying our Norme Return by out Allocations

In [12]:
```
for stock_df,allo in zip([aapl,cisco,ibm,amzn],[.3,.2,.4,.1]):
    stock_df['Allocation'] = stock_df['Normed Return']*allo
```

In [13]:
```
aapl.head()
```

Out[13]:

Date	Adj. Close	Normed Return	Allocation
2012-01-03	53.063218	1.000000	0.300000
2012-01-04	53.348386	1.005374	0.301612
2012-01-05	53.940658	1.016536	0.304961
2012-01-06	54.504543	1.027162	0.308149
2012-01-09	54.418089	1.025533	0.307660

Investment

Let's pretend we invested a million dollars in this portfolio

In [14]:
```
for stock_df in [aapl,cisco,ibm,amzn]:
    stock_df['Position Values'] = stock_df['Allocation']*1000000
```

Total Portfolio Value

In [15]:
```
portfolio_val = pd.concat([aapl['Position Values'],cisco['Position Values'],ibm['Position Values'],amzn['Positi
```

In [17]:
```
portfolio_val.head()
```

Out[17]:

Date	Position Values	Position Values	Position Values	Position Values
2012-01-03	300000.000000	200000.000000	400000.000000	100000.000000
2012-01-04	301612.236461	203864.734300	398368.223296	99150.980283
2012-01-05	304960.727573	203113.258186	396478.797638	99206.836843
2012-01-06	308148.724558	202361.782072	391926.999463	101999.664861
2012-01-09	307659.946988	203650.026838	389887.278583	99737.474166

In [18]:
```
portfolio_val.columns = ['AAPL Pos','CISCO Pos','IBM Pos','AMZN Pos']
```

In [19]:
```
portfolio_val.head()
```

Out[19]:

Date	AAPL Pos	CISCO Pos	IBM Pos	AMZN Pos
2012-01-03	300000.000000	200000.000000	400000.000000	100000.000000
2012-01-04	301612.236461	203864.734300	398368.223296	99150.980283
2012-01-05	304960.727573	203113.258186	396478.797638	99206.836843
2012-01-06	308148.724558	202361.782072	391926.999463	101999.664861
2012-01-09	307659.946988	203650.026838	389887.278583	99737.474166

In [20]:
```
portfolio_val['Total Pos'] = portfolio_val.sum(axis=1)
```

In [21]:
```
portfolio_val.head()
```

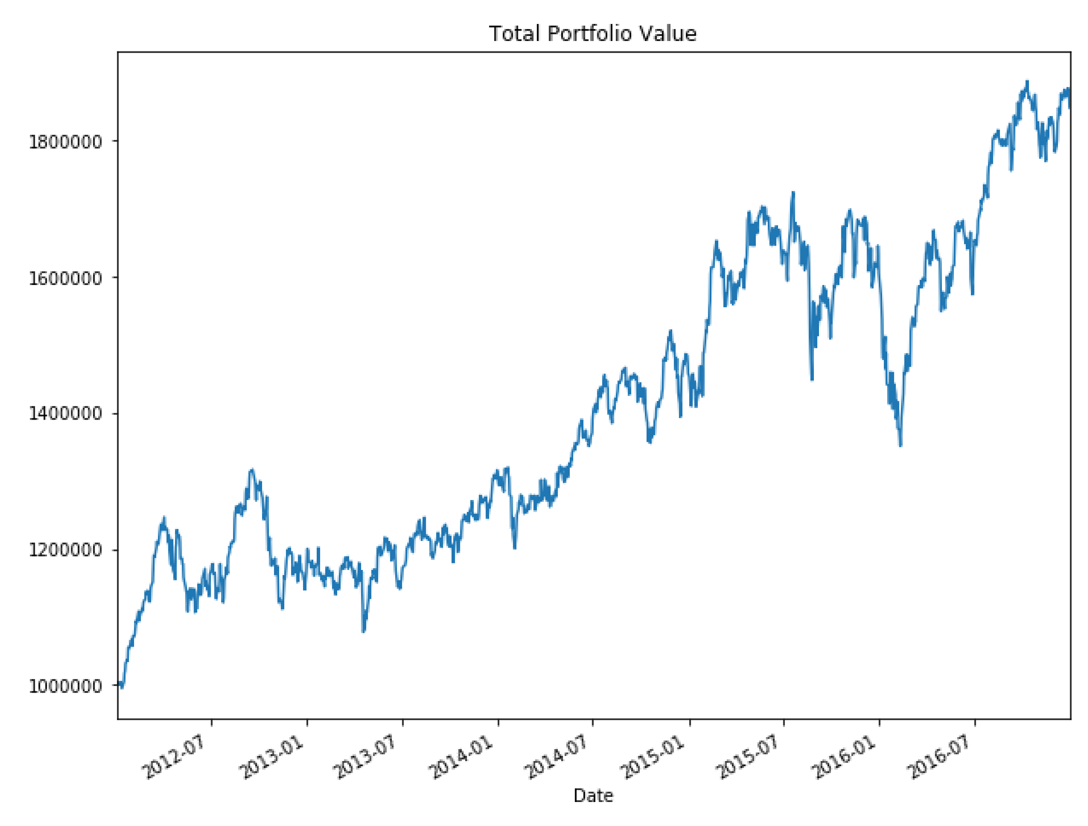
Out[21]:

Date	AAPL Pos	CISCO Pos	IBM Pos	AMZN Pos	Total Pos
2012-01-03	300000.000000	200000.000000	400000.000000	100000.000000	1.000000e+06
2012-01-04	301612.236461	203864.734300	398368.223296	99150.980283	1.002996e+06
2012-01-05	304960.727573	203113.258186	396478.797638	99206.836843	1.003760e+06
2012-01-06	308148.724558	202361.782072	391926.999463	101999.664861	1.004437e+06
2012-01-09	307659.946988	203650.026838	389887.278583	99737.474166	1.000935e+06

In [20]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

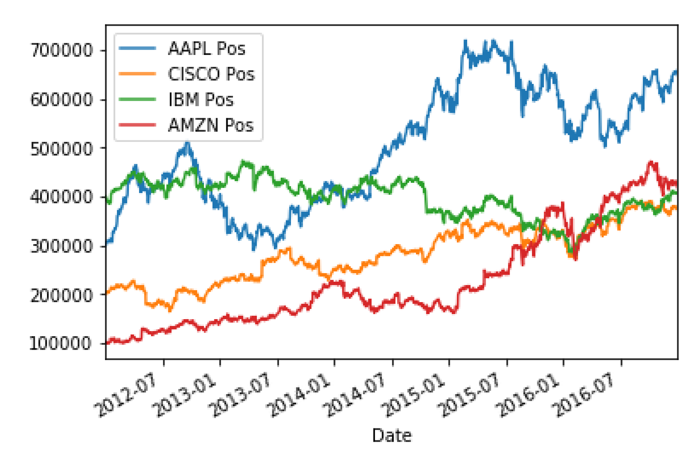
In [21]:
```
portfolio_val['Total Pos'].plot(figsize=(10,8))
plt.title('Total Portfolio Value')
```

Out[21]:
```
<matplotlib.text.Text at 0x23286633518>
```



In [22]:
```
portfolio_val.drop('Total Pos',axis=1).plot(kind='line')
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23286aa1908>
```



In [23]:
```
portfolio_val.tail()
```

Out[23]:

Date	AAPL Pos	CISCO Pos	IBM Pos	AMZN Pos	Total Pos
2016-12-23	653264.617079	377469.015679	407359.955612	424839.412389	1.862933e+06
2016-12-27	657413.396830	379323.596496	408410.671112	430877.506563	1.876025e+06
2016-12-28	654610.167268	376108.989746	406089.322915	431285.259454	1.868094e+06
2016-12-29	654441.973495	376603.544631	407091.167926	427386.471541	1.865523e+06
2016-12-30	649340.095692	373636.215323	405600.618032	418851.589119	1.847429e+06

Portfolio Statistics

Daily Returns

In [24]:
```
portfolio_val['Daily Return'] = portfolio_val['Total Pos'].pct_change(1)
```

Cumulative Return

In [25]:
```
cum_ret = 100 * (portfolio_val['Total Pos'][-1]/portfolio_val['Total Pos'][0] -1 )
print('Our return {} was percent!'.format(cum_ret))
```

```
Our return 84.74285181665459 was percent!
```

Avg Daily Return

In [26]:
```
portfolio_val['Daily Return'].mean()
```

Out[26]:
```
0.0005442330716215298
```

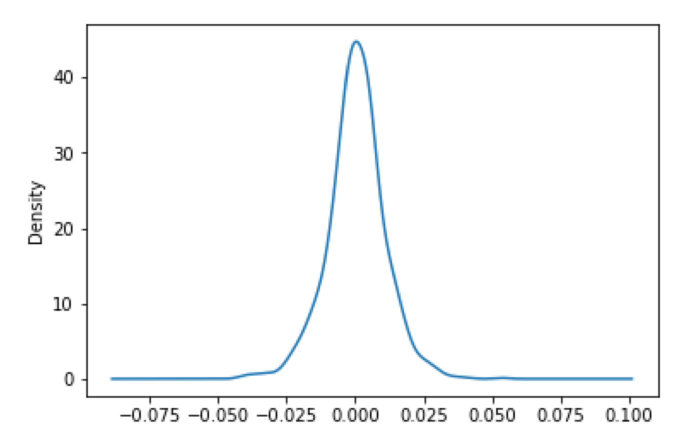
Std Daily Return

In [27]:
```
portfolio_val['Daily Return'].std()
```

Out[27]:
```
0.010568287769162561
```

In [28]:
```
portfolio_val['Daily Return'].plot(kind='kde')
```

Out[28]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x232869912e8>
```



Sharpe Ratio

The Sharpe Ratio is a measure for calculating risk-adjusted return, and this ratio has become the industry standard for such calculations.

Sharpe ratio = (Mean portfolio return − Risk-free rate)/Standard deviation of portfolio return

The original Sharpe Ratio

Annualized Sharpe Ratio = K-value * SR

K-values for various sampling rates:

- Daily = sqrt(252)
- Weekly = sqrt(52)
- Monthly = sqrt(12)

Since I'm based in the USA, I will use a very low risk-free rate (the rate you would get if you just put your money in a bank, its currently very low in the USA, let's just say its ~0% return). If you are in a different country with higher rates for your trading currency, you can use this trick to convert a yearly rate with a daily rate:

daily_rate = ((1.0 + yearly_rate)**(1/252))-1

Other values people use are things like the 3-month treasury bill or LIBOR.

Read more: Sharpe Ratio http://www.investopedia.com/terms/s/sharperatio

In [29]:
```
SR = portfolio_val['Daily Return'].mean()/portfolio_val['Daily Return'].std()
```

In [30]:
```
SR
```

Out[30]:
```
0.05149680662647732
```

In [31]:
```
ASR = (252**0.5)*SR
```

In [32]:
```
ASR
```

Out[32]:
```
0.8174864618858524
```

In [33]:
```
portfolio_val['Daily Return'].std()
```

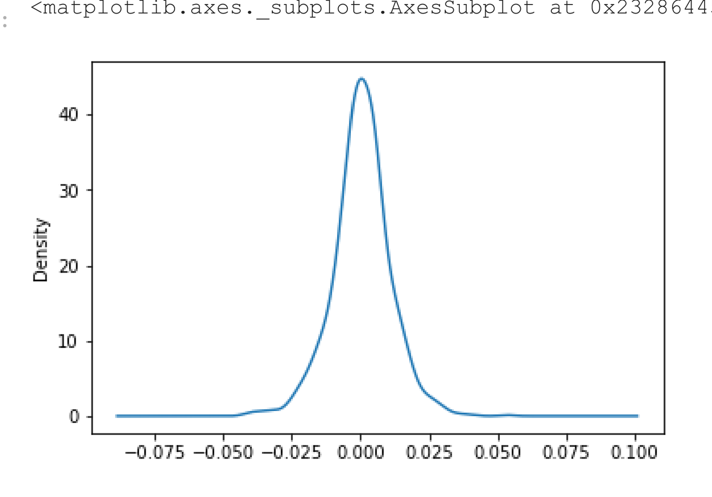
Out[33]:
```
0.010568287769162561
```

In [34]:
```
portfolio_val['Daily Return'].mean()
```

Out[34]:
```
0.0005442330716215298
```

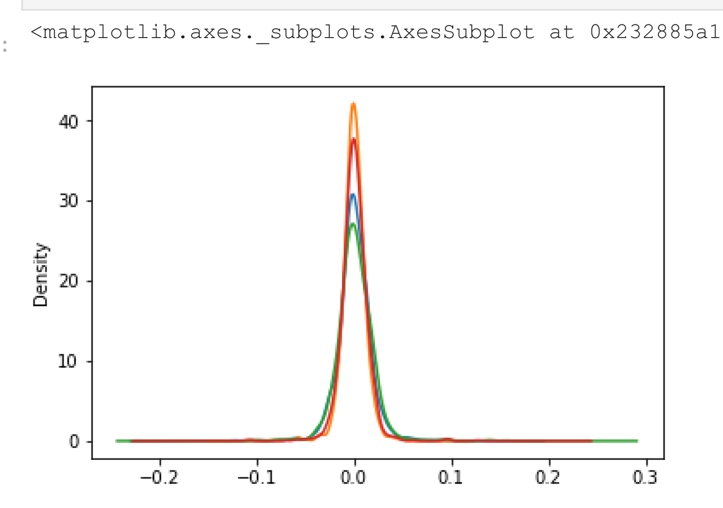
In [35]:
```
portfolio_val['Daily Return'].plot('kde')
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x23286445e80>
```



In [36]:
```
aapl['Adj. Close'].pct_change(1).plot('kde')
ibm['Adj. Close'].pct_change(1).plot('kde')
amzn['Adj. Close'].pct_change(1).plot('kde')
cisco['Adj. Close'].pct_change(1).plot('kde')
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x232885a1ef0>
```



In [37]:
```
import numpy as np
np.sqrt(252)* (np.mean(.001-0.0002)/.001)
```

Out[37]:
```
12.699606293110037
```

Great Job!