```
import pandas as pd
```

```
!pip install quandl
```

```
Requirement already satisfied: quandl in /usr/local/lib/python3.7/dist-packages (3.7.0)
Requirement already satisfied: inflection>=0.3.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from quanc
Requirement already satisfied: more-itertools in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.8 in /usr/local/lib/python3.7/dist-packages (frc
Requirement already satisfied: pandas>=0.14 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: requests>=2.7.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lit
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
```

```
import quandl
```

Create a portfolio

```
start=pd.to_datetime('2012-01-01')
end=pd.to_datetime('2021-01-01')
```

```
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)
```

```
aapl.to_csv('AAPL_CLOSE')
cisco.to_csv('CSCO_CLOSE')
ibm.to_csv('IBM_CLOSE')
amzn.to_csv('AMZN_CLOSE')
```

```
aapl.head()
```

	Adj. Close
Date	
2012-01-03	52.848787

```
cisco.head()
```

	Adj. Close
Date	
2012-01-03	15.617341
2012-01-04	15.919125
2012-01-05	15.860445
2012-01-06	15.801764
2012-01-09	15.902359

```
ibm.head()
```

	Adj. Close
Date	
2012-01-03	157.578371
2012-01-04	156.935540
2012-01-05	156.191208
2012-01-06	154.398046
2012-01-09	153.594506

Normalize Prices

Same as Cumulative Daily Returns

```
aapl.iloc[0]['Adj. Close']
```

```
52.848786580038
```

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

```
aapl.head()
```

Date	Adj. Close	Normed Return
2012-01-03	52.848787	1.000000
2012-01-04	53.132802	1.005374
2012-01-05	53.722681	1.016536
2012-01-06	54.284287	1.027162
2012-01-09	54.198183	1.025533

```
cisco.head()
```

Date	Adj. Close	Normed Return
2012-01-03	15.617341	1.000000
2012-01-04	15.919125	1.019324
2012-01-05	15.860445	1.015566
2012-01-06	15.801764	1.011809
2012-01-09	15.902359	1.018250

```
ibm.head()
```

Date	Adj. Close	Normed Return
2012-01-03	157.578371	1.000000
2012-01-04	156.935540	0.995921
2012-01-05	156.191208	0.991197
2012-01-06	154.398046	0.979817
2012-01-09	153.594506	0.974718

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

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

```
aapl.head()
```

Date	Adj. Close	Normed Return	Allocation
2012-01-03	52.848787	1.000000	0.300000
2012-01-04	53.132802	1.005374	0.301612
2012-01-05	53.722681	1.016536	0.304961
2012-01-06	54.284287	1.027162	0.308149
2012-01-09	54.198183	1.025533	0.307660

```
cisco.head()
```

Date	Adj. Close	Normed Return	Allocation
2012-01-03	15.617341	1.000000	0.200000
2012-01-04	15.919125	1.019324	0.203865
2012-01-05	15.860445	1.015566	0.203113
2012-01-06	15.801764	1.011809	0.202362
2012-01-09	15.902359	1.018250	0.203650

```
amzn.head()
```

	Adj. Close	Normed Return	Allocation
Date			

Investment

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

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

```
aapl.head()
```

	Adj. Close	Normed Return	Allocation	Position Values
Date				
2012-01-03	52.848787	1.000000	0.300000	300000.000000
2012-01-04	53.132802	1.005374	0.301612	301612.236461
2012-01-05	53.722681	1.016536	0.304961	304960.727573
2012-01-06	54.284287	1.027162	0.308149	308148.724558
2012-01-09	54.198183	1.025533	0.307660	307659.946988

Total Portfolio Value

```
portfolio_val=pd.concat([aapl['Position Values'],cisco['Position Values'],ibm['Position Value
                         amzn['Position Values']],axis=1)
```

```
portfolio_val.head()
```

	Position Values	Position Values	Position Values	Position Values
Date				
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

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

```
portfolio_val.head()
```

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

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

```
portfolio_val.head()
```

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-	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

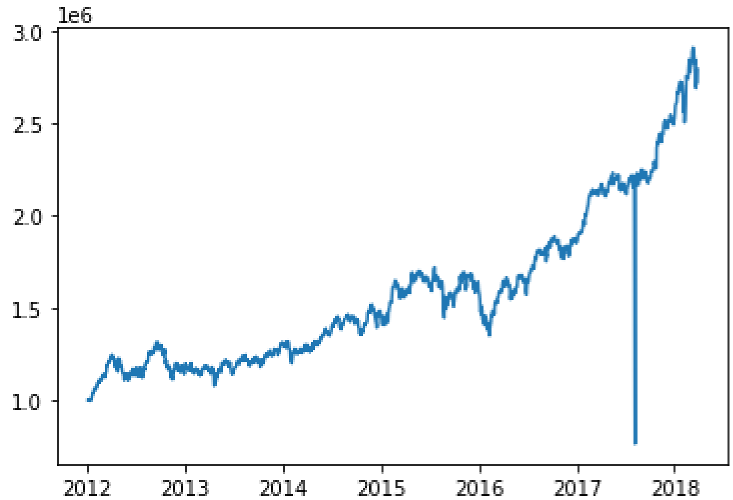
```
portfolio_val['Total Pos'].head()
```

```
Date
2012-01-03    1.000000e+06
2012-01-04    1.002996e+06
2012-01-05    1.003760e+06
```

```
2012-01-06    1.004437e+06
2012-01-09    1.000935e+06
Name: Total Pos, dtype: float64
```

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(portfolio_val['Total Pos'])
```

```
[<matplotlib.lines.Line2D at 0x7f1396f73a10>]
```

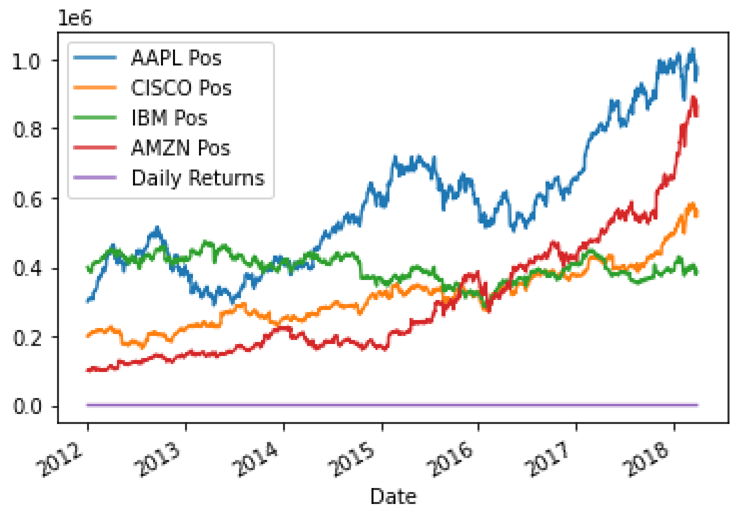



```
portfolio_val['Total Pos'].plot(figsize=(12,10))
plt.title('Total Portfolio Value')
```

```
Text(0.5, 1.0, 'Total Portfolio Value')
```


```
portfolio_val.drop('Total Pos',axis=1).plot(kind='line')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1396a53650>
```



Portfolio Statistics

Daily Returns

```
portfolio_val['Total Pos'].pct_change(1)
```

```
Date
2012-01-03         NaN
2012-01-04    0.002996
2012-01-05    0.000761
2012-01-06    0.000675
2012-01-09   -0.003487
                ...   
2018-03-21   -0.008665
2018-03-22   -0.022125
2018-03-23   -0.023826
2018-03-26    0.041036
2018-03-27   -0.028314
Name: Total Pos, Length: 1567, dtype: float64
```

```
portfolio_val['Daily Returns']=portfolio_val['Total Pos'].pct_change(1)
portfolio_val
```

Date	AAPL Pos	CISCO Pos	IBM Pos	AMZN Pos	Total Pos	R(
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	0.(
2012-01-05	304960.727573	203113.258186	396478.797638	99206.836843	1.003760e+06	0.(
2012-01-06	308148.724558	202361.782072	391926.999463	101999.664861	1.004437e+06	0.(
2012-01-09	307659.946988	203650.026838	389887.278583	99737.474166	1.000935e+06	-0.(
...	...	...	...	...	...	
2018-03-21	972226.673969	567446.158939	397744.940688	883572.585600	2.820990e+06	-0.(
2018-	958460.984214	551566.374757	386068.211304	862481.148411	2.758577e+06	-0.(

Average Daily Return

```
portfolio_val['Daily Returns'].mean()
```

```
0.0014927305900954441
```

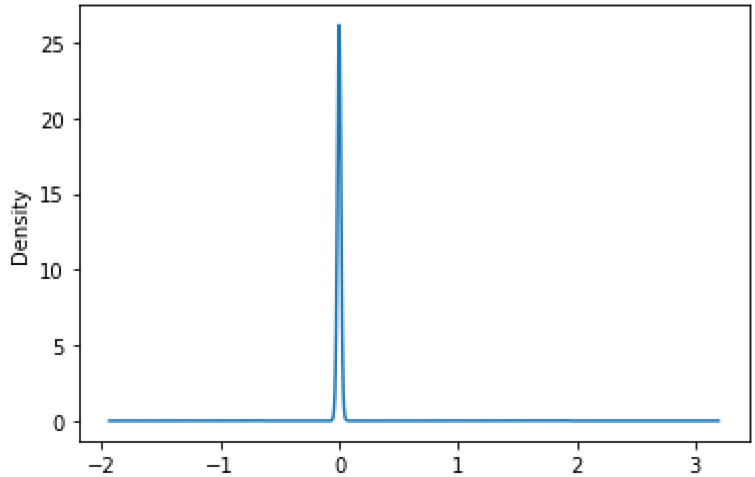
Standard Daily Return

```
portfolio_val['Daily Returns'].std()
```

```
0.05213018140551365
```

```
portfolio_val['Daily Returns'].plot(kind='kde')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f139708d650>



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

```
SR=portfolio_val['Daily Returns'].mean()/portfolio_val['Daily Returns'].std()
SR
```

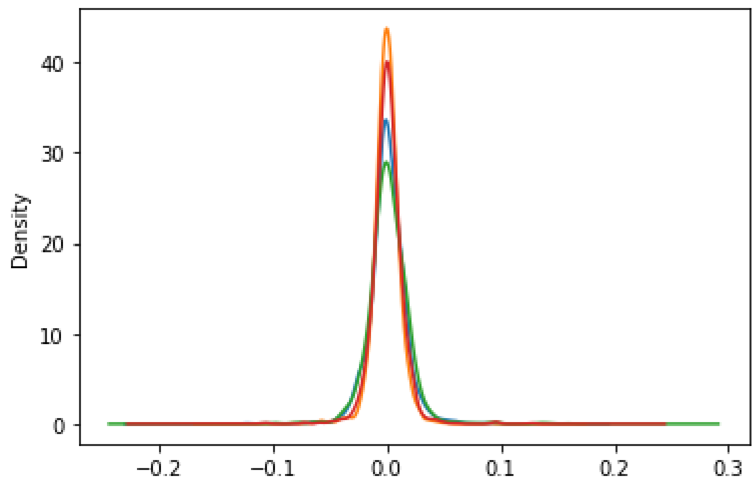
    0.028634670930524762

```
ASR=(252**0.5)*SR
ASR
```

```
0.4545613089380341
```

```
aapl['Adj. Close'].pct_change(1).plot(kind='kde')
ibm['Adj. Close'].pct_change(1).plot(kind='kde')
amzn['Adj. Close'].pct_change(1).plot(kind='kde')
cisco['Adj. Close'].pct_change(1).plot(kind='kde')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f13972bf490>
```




```
import numpy as np
np.sqrt(252)*(np.mean(0.001-0.0002)/0.001)
```

```
12.699606293110037
```