In [ ]:

Rolling and Expanding

A very common process with time series is to create data based off of a rolling mean. Let's show you how to do this easily with pandas!

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
# Best way to read in data with time series index!
df = pd.read_csv('time_data/walmart_stock.csv',index_col='Date',parse_dates=True)
```

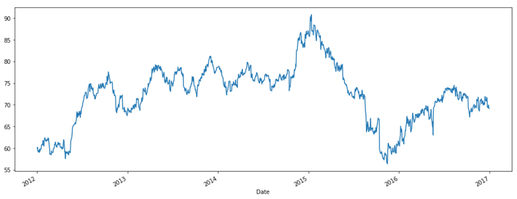
In [4]:
```
df.head(20)
```

Out[4]:

Date	Open	High	Low	Close	Volume	Adj Close
2012-01-03	59.970001	61.060001	59.869999	60.330002	12668800	52.619235
2012-01-04	60.209999	60.349998	59.470001	59.709999	9593300	52.078475
2012-01-05	59.349998	59.619999	58.369999	59.419998	12768200	51.825539
2012-01-06	59.419998	59.450001	58.869999	59.000000	8069400	51.459220
2012-01-09	59.029999	59.549999	58.919998	59.180000	6679300	51.616215
2012-01-10	59.430000	59.709999	58.980000	59.040001	6907300	51.494109
2012-01-11	59.060001	59.529999	59.040001	59.400002	6365600	51.808098
2012-01-12	59.790001	60.000000	59.400002	59.500000	7236400	51.895316
2012-01-13	59.180000	59.610001	59.009998	59.540001	7729300	51.930204
2012-01-17	59.869999	60.110001	59.520000	59.849998	8500000	52.200581
2012-01-18	59.790001	60.029999	59.650002	60.009998	5911400	52.340131
2012-01-19	59.930000	60.730000	59.750000	60.610001	9234600	52.863447
2012-01-20	60.750000	61.250000	60.669998	61.009998	10378800	53.212321
2012-01-23	60.810001	60.980000	60.509998	60.910000	7134100	53.125104
2012-01-24	60.750000	62.000000	60.750000	61.389999	7362800	53.543754
2012-01-25	61.180000	61.610001	61.040001	61.470001	5915800	53.613531
2012-01-26	61.799999	61.840000	60.770000	60.970001	7436200	53.177436
2012-01-27	60.860001	61.119999	60.540001	60.709999	6287300	52.950665
2012-01-30	60.470001	61.320000	60.349998	61.299999	7636900	53.465257
2012-01-31	61.529999	61.570000	60.580002	61.360001	9761500	53.517590

In [5]:
```
df['Open'].plot(figsize=(16,6))
```

Out[5]:
```
<AxesSubplot:xlabel='Date'>
```



Now let's add in a rolling mean! This rolling method provides row entries, where every entry is then representative of the window.

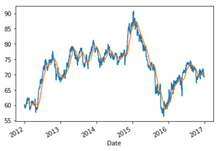
In [6]:
```
# 7 day rolling mean
df.rolling(7).mean().head(20)
```

Out[6]:

Date	Open	High	Low	Close	Volume	Adj Close
2012-01-03	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-04	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-05	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-06	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-09	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-10	NaN	NaN	NaN	NaN	NaN	NaN
2012-01-11	59.495714	59.895714	59.074285	59.440000	9.007414e+06	51.842984
2012-01-12	59.469999	59.744285	59.007143	59.321429	8.231357e+06	51.739567
2012-01-13	59.322857	59.638571	58.941428	59.297143	7.965071e+06	51.718386
2012-01-17	59.397143	59.708571	59.105714	59.358572	7.355329e+06	51.771963
2012-01-18	59.450000	59.791428	59.217143	59.502857	7.047043e+06	51.897808
2012-01-19	59.578572	59.960000	59.335715	59.707143	7.412086e+06	52.075984
2012-01-20	59.767143	60.180000	59.577143	59.988571	7.908014e+06	52.321443
2012-01-23	60.017143	60.387143	59.787143	60.204285	8.017800e+06	52.509586
2012-01-24	60.154286	60.672857	59.979999	60.474285	8.035857e+06	52.745077
2012-01-25	60.440000	60.958572	60.270000	60.749999	7.776786e+06	52.985553
2012-01-26	60.715714	61.205714	60.448571	60.910000	7.624814e+06	53.125103
2012-01-27	60.868572	61.361429	60.575714	61.010000	7.678514e+06	53.212323
2012-01-30	60.945715	61.445714	60.661428	61.108571	7.450271e+06	53.298295
2012-01-31	61.057143	61.491429	60.648571	61.158571	7.362086e+06	53.341905

In [7]:
```
df['Open'].plot()
df.rolling(window=30).mean()['Close'].plot()
```

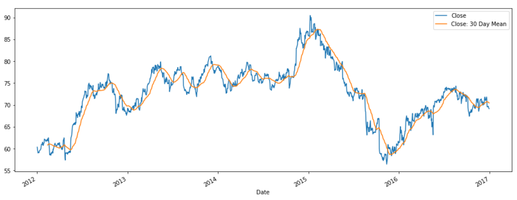
Out[7]:
```
<AxesSubplot:xlabel='Date'>
```



Easiest way to add a legend is to make this rolling value a new column, then pandas does it automatically!

In [8]:
```
df['Close: 30 Day Mean'] = df['Close'].rolling(window=30).mean()
df[['Close','Close: 30 Day Mean']].plot(figsize=(16,6))
```

Out[8]:
```
<AxesSubplot:xlabel='Date'>
```

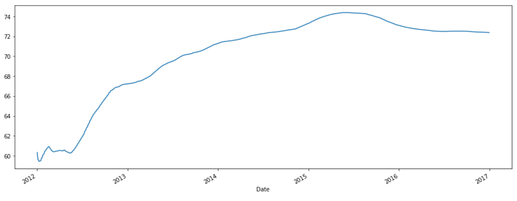


expanding

Now what if you want to take into account everything from the start of the time series as a rolling value? For instance, not just take into account a period of 7 days, or monthly rolling average, but instead, take into everything since the beginning of the time series, continuously:

In [9]:
```
# Optional specify a minimum number of periods
df['Close'].expanding(min_periods=1).mean().plot(figsize=(16,6))
```

Out[9]:
```
<AxesSubplot:xlabel='Date'>
```



Bollinger Bands

We will talk a lot more about financial analysis plots and technical indicators, but here is one worth mentioning!

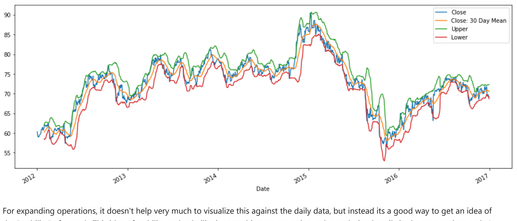
More info : http://www.investopedia.com/terms/b/bollingerbands.asp

Developed by John Bollinger, Bollinger Bands® are volatility bands placed above and below a moving average. Volatility is based on the standard deviation, which changes as volatility increases and decreases. The bands automatically widen when volatility increases and narrow when volatility decreases. This dynamic nature of Bollinger Bands also means they can be used on different securities with the standard settings. For signals, Bollinger Bands can be used to identify Tops and Bottoms or to determine the strength of the trend.

Bollinger Bands reflect direction with the 20-period SMA and volatility with the upper/lower bands. As such, they can be used to determine if prices are relatively high or low. According to Bollinger, the bands should contain 88-89% of price action, which makes a move outside the bands significant. Technically, prices are relatively high when above the upper band and relatively low when below the lower band. However, relatively high should not be regarded as bearish or as a sell signal. Likewise, relatively low should not be considered bullish or as a buy signal. Prices are high or low for a reason. As with other indicators, Bollinger Bands are not meant to be used as a stand alone tool.

In [10]:
```
df['Close: 30 Day Mean'] = df['Close'].rolling(window=20).mean()
df['Upper'] = df['Close: 30 Day Mean'] + 2*df['Close'].rolling(window=20).std()
df['Lower'] = df['Close: 30 Day Mean'] - 2*df['Close'].rolling(window=20).std()
df[['Close','Close: 30 Day Mean','Upper','Lower']].plot(figsize=(16,6))
```

Out[10]:
```
<AxesSubplot:xlabel='Date'>
```



For expanding operations, it doesn't help very much to visualize this against the daily data, but instead its a good way to get an idea of the "stability" of a stock. This idea of stability and volatility is something we are going to be exploring heavily in the next project, so let's jump straight into it!