5.2-Working with Time Series Data

KeytoDataScience.com

This section lays the foundations to leverage the powerful time series functionality made available by how Pandas represents dates, in particular by the DateTime Index.

We will learn how to create and manipulate date information and time series, and how to do calculations with time-aware DataFrames to shift your data in time or create period-specific returns.

Table of Contents

Example1: Air Travel Data

Create and plot a `Time Series`

In [1]:
```
# import libraries

import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv('AirPassengers.csv')

# Inspect data
data.head()
```

Out[2]:

	Month	#Passengers
0	1949-01	112
1	1949-02	118

	Month	#Passengers
2	1949-03	132
3	1949-04	129
4	1949-05	121

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Month        144 non-null    object
 1   #Passengers  144 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 2.4+ KB
```

In [4]:
```
# Convert the date column to datetime64
data.Month = pd.to_datetime(data.Month) #format='%Y-%m'
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype         
---  ------       --------------  -----         
 0   Month        144 non-null    datetime64[ns]
 1   #Passengers  144 non-null    int64         
dtypes: datetime64[ns](1), int64(1)
memory usage: 2.4 KB
None
```

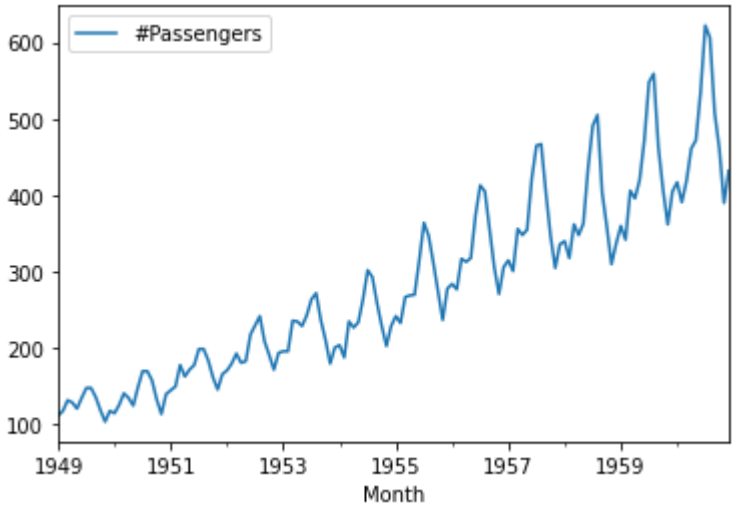
In [5]:
```
# Set date column as index
data.set_index('Month', inplace=True)

# Inspect data
data.head()
```

Out[5]:

	#Passengers
Month	
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

In [6]:
```
# Plot data
data.plot()
```

```
plt.show()
```



↥ back to top

Example 2: Annual Stock Price Trends Data

Indexing & resampling time series

2.1 Compare YoY Annual Stock Price Trends

We'll learn how to select sub-periods from a time series to compare the performance over years of `GOOGLE` stock prices.

Let's compare YoY(Year on Year) time series for `GOOGLE` stock prices.

In [7]:
```
google = pd.read_csv('GOOGL.csv', parse_dates=['Date'], index_col='Date')
```

In [8]:
```
google.head()
```

Out[8]:

Date	Open	High	Low	Close	Adj Close	Volume
2010-01-04	313.788788	315.070068	312.432434	313.688690	313.688690	3908400
2010-01-05	313.903900	314.234222	311.081085	312.307312	312.307312	6003300
2010-01-06	313.243256	313.243256	303.483490	304.434448	304.434448	7949400
2010-01-07	305.005005	305.305298	296.621613	297.347351	297.347351	12815700
2010-01-08	296.296295	301.926941	294.849854	301.311310	301.311310	9439100

In [9]:
```
# Create dataframe prices here
prices = pd.DataFrame()

# Select data for each year and concatenate with prices here
for year in ['2013', '2014', '2015']:
```

```
    price_per_year = google.loc[year, ['Open']].reset_index(drop=True)    #partial inde
    price_per_year.rename(columns={'Open': year}, inplace=True)
    prices = pd.concat([prices, price_per_year], axis=1)
```

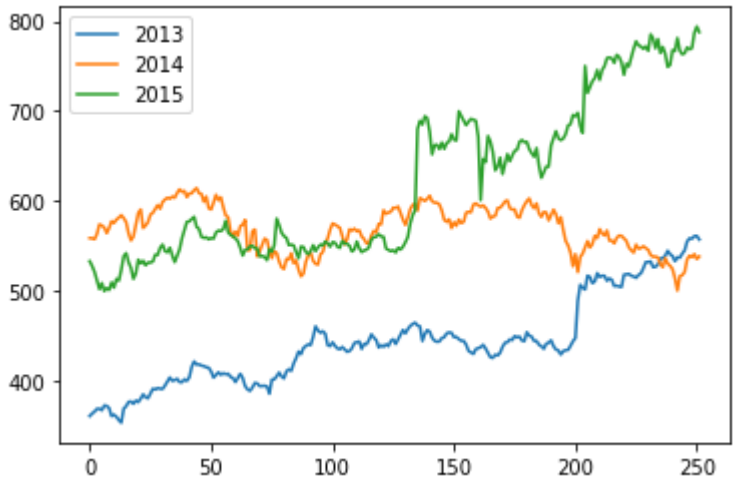
In [10]:
```
prices.head()
```

Out[10]:

	2013	2014	2015
0	360.070068	558.288269	532.599976
1	362.827820	558.058044	527.150024
2	365.035034	557.062073	520.500000
3	368.093079	563.063049	510.950012
4	368.138153	573.573547	501.510010

In [11]:
```
# Plot prices
prices.plot()
```

Out[11]:
```
<AxesSubplot:>
```



2.2 Set and change time series frequency

We will learn how to assign a frequency to a DateTimeIndex, and then change this frequency.

Let's change frequency to:

1. **Days**
2. **Months**

Note: If you want to do summarization, please use `resample` or `groupby` .

In [12]:
```
google.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2013 entries, 2010-01-04 to 2017-12-29
Data columns (total 6 columns):
```

```
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Open       2013 non-null   float64
 1   High       2013 non-null   float64
 2   Low        2013 non-null   float64
 3   Close      2013 non-null   float64
 4   Adj Close  2013 non-null   float64
 5   Volume     2013 non-null   int64
dtypes: float64(5), int64(1)
memory usage: 110.1 KB
```

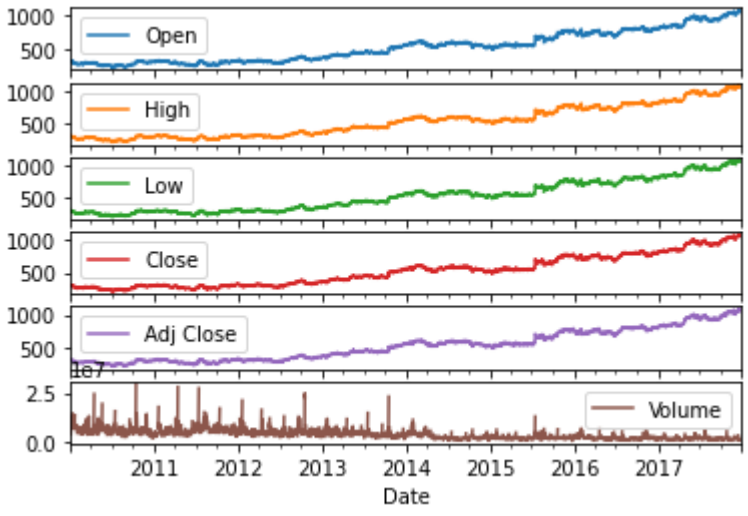
📌1. Days

In [13]:
```
google_D = google.asfreq('D')
google_D.head()
```

Out[13]:

Date	Open	High	Low	Close	Adj Close	Volume
2010-01-04	313.788788	315.070068	312.432434	313.688690	313.688690	3908400.0
2010-01-05	313.903900	314.234222	311.081085	312.307312	312.307312	6003300.0
2010-01-06	313.243256	313.243256	303.483490	304.434448	304.434448	7949400.0
2010-01-07	305.005005	305.305298	296.621613	297.347351	297.347351	12815700.0
2010-01-08	296.296295	301.926941	294.849854	301.311310	301.311310	9439100.0

In [14]:
```
# Plot the data
google_D.plot(subplots=True)
plt.show()
```



📌2. Months

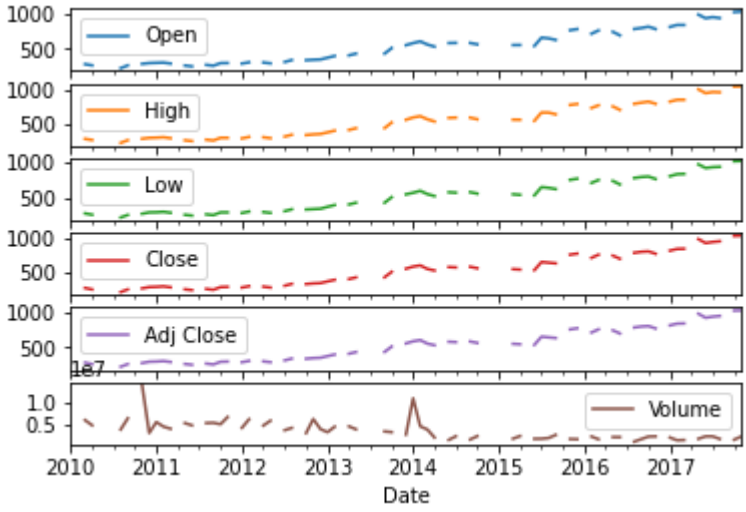
In [15]:
```
google_M = google.asfreq('M')
google_M.head()
```

Out[15]:

Date	Open	High	Low	Close	Adj Close	Volume

Date	Open	High	Low	Close	Adj Close	Volume
Date						
2010-01-31	NaN	NaN	NaN	NaN	NaN	NaN
2010-02-28	NaN	NaN	NaN	NaN	NaN	NaN
2010-03-31	282.807800	285.155151	281.686676	283.843842	283.843842	6055500.0
2010-04-30	265.830841	269.109100	262.982971	263.113098	263.113098	4865900.0
2010-05-31	NaN	NaN	NaN	NaN	NaN	NaN

In [16]:
```
# Plot the data
google_M.plot(subplots=True)
plt.show()
```



↥ back to top

2.3 Lags, changes, and returns for stock price series

Basic time series calculation:

- Shift or lag values back or forward back in time.
- Get the difference in a value for a given period of time.
- Compute the percent change over any number of periods.

Pandas built in methods rely on pd.DateTimeIndex

2.3.1 Shifting stock prices across time

Method to manipulate time series using `.shift()`, which allows you shift all values in a Series or DataFrame by a number of periods to a different time along the DateTimeIndex.

Let's use this to visually compare a stock price series for Google shifted 90 business days into both past and future.

In [17]:
```
# Import data here
google = pd.read_csv('GOOGL.csv', parse_dates=['Date'], index_col='Date')

# Set data frequency to business daily
google = google.asfreq('B')

# Create 'lagged' and 'shifted'
google['lagged'] = google.Close.shift(periods=-90)
google['shifted'] = google.Close.shift(periods=90)

# Plot the google price series
google[['Close','lagged','shifted']].plot()
plt.show()
```

In [18]:
```
google.head(10)
```

Out[18]:

Date	Open	High	Low	Close	Adj Close	Volume	lagged	shifted
2010-01-04	313.788788	315.070068	312.432434	313.688690	313.688690	3908400.0	261.086090	NaN
2010-01-05	313.903900	314.234222	311.081085	312.307312	312.307312	6003300.0	254.779785	NaN
2010-01-06	313.243256	313.243256	303.483490	304.434448	304.434448	7949400.0	252.947952	NaN
2010-01-07	305.005005	305.305298	296.621613	297.347351	297.347351	12815700.0	255.695694	NaN
2010-01-08	296.296295	301.926941	294.849854	301.311310	301.311310	9439100.0	254.019012	NaN
2010-01-11	302.532532	302.532532	297.317322	300.855865	300.855865	14411300.0	254.239243	NaN
2010-01-12	299.124115	299.379395	294.294281	295.535522	295.535522	9696800.0	249.434433	NaN
2010-01-13	288.533539	294.484497	287.237244	293.838837	293.838837	12980200.0	247.462463	NaN

Date	Open	High	Low	Close	Adj Close	Volume	lagged	shifted
2010-01-14	292.242249	297.397400	291.696686	295.220215	295.220215	8471700.0	237.742737	NaN
2010-01-15	296.966980	297.077087	289.309296	290.290283	290.290283	10858100.0	236.261261	NaN

2.3.2 Calculating stock price changes

We'll learn to calculate returns using current and shifted prices as input. Now we'll practice a similar calculation to calculate absolute changes from current and shifted prices, and compare the result to the function .diff().

In [19]:
```
# Import data here
google = pd.read_csv('GOOGL.csv', parse_dates=['Date'], index_col='Date')

# Created shifted_30 here
google['shifted_30'] = google.Open.shift(30)

# Subtract shifted_30 from price
google['change_30'] = google['Open'] - google['shifted_30']

# Get the 30-day price difference
google['diff_30'] = google.Open.diff(30)

# Inspect the last five rows of price
google.tail()
```

Out[19]:

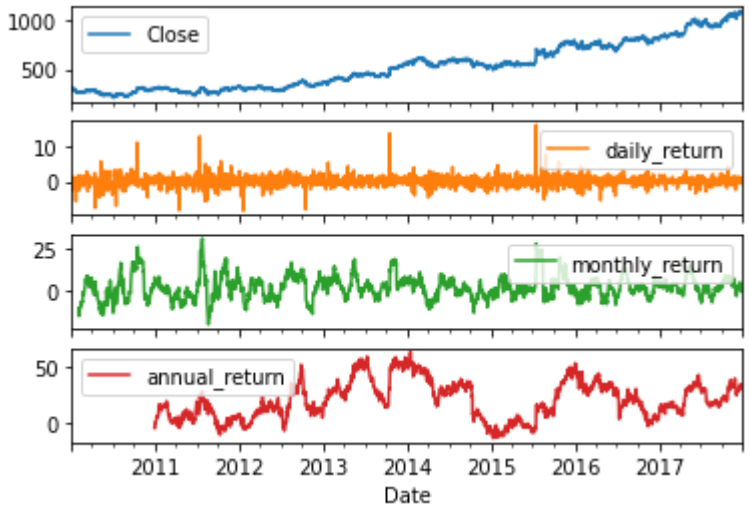
Date	Open	High	Low	Close	Adj Close	Volume	shifted_30	change_
2017-12-22	1070.000000	1071.719971	1067.640015	1068.859985	1068.859985	889400	1048.000000	22.0000
2017-12-26	1068.640015	1068.859985	1058.640015	1065.849976	1065.849976	918800	1043.869995	24.7700
2017-12-27	1066.599976	1068.270020	1058.380005	1060.199951	1060.199951	1116200	1040.800049	25.7999
2017-12-28	1062.250000	1064.839966	1053.380005	1055.949951	1055.949951	994200	1037.719971	24.5300
2017-12-29	1055.489990	1058.050049	1052.699951	1053.400024	1053.400024	1180300	1035.000000	20.4899

2.3.3 Plotting multi-period returns

The last time series method that we'll learn is .pct_change(). Let's use this function to calculate returns for various calendar day periods, and plot the result to compare the different patterns.

In [20]:
```
# Import data here
```

```
google = pd.read_csv('GOOGL.csv', parse_dates=['Date'], index_col='Date')

# Set data frequency to business daily
google = google.asfreq('D')

# Create daily_return
google['daily_return'] = google.Close.pct_change(1).mul(100)

# Create monthly_return
google['monthly_return'] = google.Close.pct_change(30).mul(100)

# Create annual_return
google['annual_return'] = google.Close.pct_change(360).mul(100)

# Plot the result
google[['Close','daily_return','monthly_return','annual_return']].plot(subplots=True)
plt.show()
```



↥ back to top

Great Job!
