Lab 6 Timeseries

Skills

- Reformat dates
- Extract parts of dates
- Visualize timeseries data

Data Source

- https://github.com/fivethirtyeight/uber-tlc-foil-response

Resources

- https://jakevdp.github.io/PythonDataScienceHandbook/03.11-working-with-time-series.html
- https://docs.python.org/2/library/time.html

In [1]:
```
# import modules
import pandas as pd
import numpy as np
import datetime
import matplotlib.pyplot as plt
%matplotlib inline
```

Part A: Formatting Dates

In [19]:
```
# read in data for uber from jul14 (uber-raw-data-jul14.csv), aug14 (uber-raw-data-aug
jul = pd.read_csv('uber-raw-data-jul14.csv')
aug = pd.read_csv('uber-raw-data-aug14.csv')
sep = pd.read_csv('uber-raw-data-sep14.csv')
# print first few lines of each data set
jul.head
aug.head
sep.head
```

Out[19]:
```
<bound method NDFrame.head of                     Date/Time      Lat      Lon    Base
0          9/1/2014 0:01:00  40.2201 -74.0021  B02512
1          9/1/2014 0:01:00  40.7500 -74.0027  B02512
2          9/1/2014 0:03:00  40.7559 -73.9864  B02512
3          9/1/2014 0:06:00  40.7450 -73.9889  B02512
4          9/1/2014 0:11:00  40.8145 -73.9444  B02512
...                     ...      ...      ...     ...
1028131  9/30/2014 22:57:00  40.7668 -73.9845  B02764
1028132  9/30/2014 22:57:00  40.6911 -74.1773  B02764
1028133  9/30/2014 22:58:00  40.8519 -73.9319  B02764
1028134  9/30/2014 22:58:00  40.7081 -74.0066  B02764
1028135  9/30/2014 22:58:00  40.7140 -73.9496  B02764

[1028136 rows x 4 columns]>
```

In [16]:
```
# Append/stack the 3 uber dataframes imported in the previous cell
uber = pd.merge(jul, aug, how = 'outer')
uber = pd.merge(uber, sep, how = 'outer')
uber
```

Out[16]:

	Date/Time	Lat	Lon	Base
0	7/1/2014 0:03:00	40.7586	-73.9706	B02512
1	7/1/2014 0:05:00	40.7605	-73.9994	B02512
2	7/1/2014 0:06:00	40.7320	-73.9999	B02512
3	7/1/2014 0:09:00	40.7635	-73.9793	B02512
4	7/1/2014 0:20:00	40.7204	-74.0047	B02512
...	...	...	...	...
2653527	9/30/2014 22:57:00	40.7668	-73.9845	B02764
2653528	9/30/2014 22:57:00	40.6911	-74.1773	B02764
2653529	9/30/2014 22:58:00	40.8519	-73.9319	B02764
2653530	9/30/2014 22:58:00	40.7081	-74.0066	B02764
2653531	9/30/2014 22:58:00	40.7140	-73.9496	B02764

2653532 rows × 4 columns

In [29]:
```
# separate date from time using string split (that is, using the 'Date/Time' column, c
uber[['Date','Time']] = uber['Date/Time'].str.split(' ', expand=True)
uber.head()
```

Out[29]:

	Date/Time	Lat	Lon	Base	Date	Time	Date2	Time2
0	7/1/2014 0:03:00	40.7586	-73.9706	B02512	7/1/2014	0:03:00	2014-07-01	12:03:00
1	7/1/2014 0:05:00	40.7605	-73.9994	B02512	7/1/2014	0:05:00	2014-07-01	12:05:00
2	7/1/2014 0:06:00	40.7320	-73.9999	B02512	7/1/2014	0:06:00	2014-07-01	12:06:00
3	7/1/2014 0:09:00	40.7635	-73.9793	B02512	7/1/2014	0:09:00	2014-07-01	12:09:00
4	7/1/2014 0:20:00	40.7204	-74.0047	B02512	7/1/2014	0:20:00	2014-07-01	12:20:00

In [30]:
```
# convert the values in the 'Date' column into the format Year-Month-Day, as in 2014-0
uber['Date2'] = uber['Date'].apply(lambda x: datetime.datetime.strptime(x, '%m/%d/%Y')
uber.head()
```

Out[30]:

	Date/Time	Lat	Lon	Base	Date	Time	Date2	Time2
0	7/1/2014 0:03:00	40.7586	-73.9706	B02512	7/1/2014	0:03:00	2014-07-01	12:03:00
1	7/1/2014 0:05:00	40.7605	-73.9994	B02512	7/1/2014	0:05:00	2014-07-01	12:05:00
2	7/1/2014 0:06:00	40.7320	-73.9999	B02512	7/1/2014	0:06:00	2014-07-01	12:06:00
3	7/1/2014 0:09:00	40.7635	-73.9793	B02512	7/1/2014	0:09:00	2014-07-01	12:09:00
4	7/1/2014 0:20:00	40.7204	-74.0047	B02512	7/1/2014	0:20:00	2014-07-01	12:20:00

In [25]:
```
# convert the values in the 'Time' column into the format 12hr:Min:Sec (i.e. 12 hour c
uber["Time2"] = uber["Time"].apply(lambda x:datetime.datetime.strptime(x,"%H:%M:%S").s
uber.head()
```

Out[25]:

	Date/Time	Lat	Lon	Base	Date	Time	Date2	Time2
0	7/1/2014 0:03:00	40.7586	-73.9706	B02512	7/1/2014	0:03:00	2014-07-01	12:03:00
1	7/1/2014 0:05:00	40.7605	-73.9994	B02512	7/1/2014	0:05:00	2014-07-01	12:05:00
2	7/1/2014 0:06:00	40.7320	-73.9999	B02512	7/1/2014	0:06:00	2014-07-01	12:06:00
3	7/1/2014 0:09:00	40.7635	-73.9793	B02512	7/1/2014	0:09:00	2014-07-01	12:09:00
4	7/1/2014 0:20:00	40.7204	-74.0047	B02512	7/1/2014	0:20:00	2014-07-01	12:20:00

Part B: Visualize Timeseries Data

Visualize by date

In [31]:
```
# extract the day of the week from the 'Date' variable in the uber data set and save t
# HINT: In the lecture, we extracted the hour from the time variable, using apply, lam
# example, you should use apply, lambda and "weekday()" instead
uber["DayOfWeek"] = uber["Date"].apply(lambda x:datetime.datetime.strptime(x,"%m/%d/%Y
uber.head()
```

Out[31]:

	Date/Time	Lat	Lon	Base	Date	Time	Date2	Time2	DayOfWeek
0	7/1/2014 0:03:00	40.7586	-73.9706	B02512	7/1/2014	0:03:00	2014-07-01	12:03:00	1
1	7/1/2014 0:05:00	40.7605	-73.9994	B02512	7/1/2014	0:05:00	2014-07-01	12:05:00	1
2	7/1/2014 0:06:00	40.7320	-73.9999	B02512	7/1/2014	0:06:00	2014-07-01	12:06:00	1
3	7/1/2014 0:09:00	40.7635	-73.9793	B02512	7/1/2014	0:09:00	2014-07-01	12:09:00	1
4	7/1/2014 0:20:00	40.7204	-74.0047	B02512	7/1/2014	0:20:00	2014-07-01	12:20:00	1

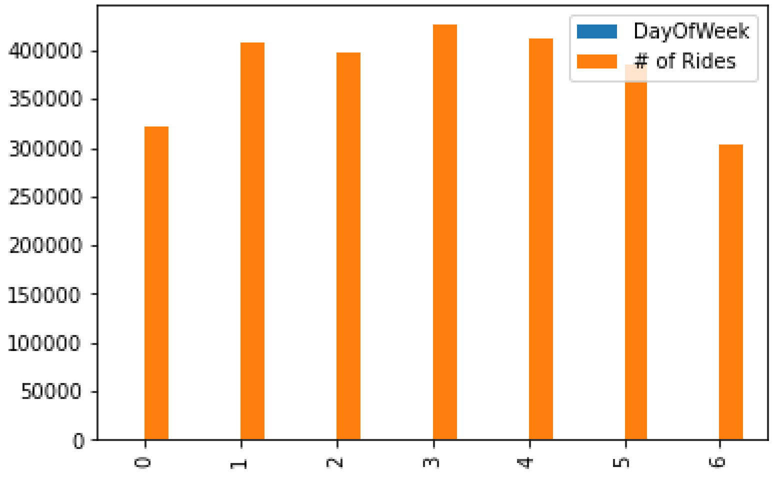
In [32]:
```
# aggregate the uber data set by day of the week (DayOfWeek) and count the number of r
# HINT: Use groupby (each row represents 1 ride, so you can use "Date" to identify 1 r
# HINT2: remember to reset the index and rename the columns
rides = uber[['DayOfWeek', 'Date']].groupby(['DayOfWeek']).count()
rides = rides.reset_index()
rides.columns = ['DayOfWeek', '# of Rides']
# print the first few rows
rides.head()
# To interpret the Day of Week remeber that 0 = Monday, 1 = Tuesday, 2 = Wednesday etc
```

Out[32]:

	DayOfWeek	# of Rides
0	0	322110
1	1	407808
2	2	398346
3	3	425832
4	4	411789

In [33]:
```
# create a barchart to display the number of trips per day of the week (DayOfWeek)
rides[['Day Of Week', '# of Rides']].plot(kind = 'bar')
```

Out[33]:
```
<AxesSubplot:>
```



In [34]:
```
# extract the hour from the Time variable in the uber data set and save this as a new
uber["Hour"] = uber["Time"].apply(lambda x:datetime.datetime.strptime(x,"%H:%M:%S").ho
uber.head()
```

Out[34]:

	Date/Time	Lat	Lon	Base	Date	Time	Date2	Time2	DayOfWeek	Hour
0	7/1/2014 0:03:00	40.7586	-73.9706	B02512	7/1/2014	0:03:00	2014-07-01	12:03:00	1	0
1	7/1/2014 0:05:00	40.7605	-73.9994	B02512	7/1/2014	0:05:00	2014-07-01	12:05:00	1	0
2	7/1/2014 0:06:00	40.7320	-73.9999	B02512	7/1/2014	0:06:00	2014-07-01	12:06:00	1	0
3	7/1/2014 0:09:00	40.7635	-73.9793	B02512	7/1/2014	0:09:00	2014-07-01	12:09:00	1	0
4	7/1/2014 0:20:00	40.7204	-74.0047	B02512	7/1/2014	0:20:00	2014-07-01	12:20:00	1	0

In [35]:
```
# aggregate the uber data set by hour and count the number of rides per hour
# HINT2: remember to reset the index and rename the columns
rides2 = uber[['Hour','Date/Time']].groupby('Hour').agg('count')
rides2 = rides2.reset_index()
rides2.columns = ['Hour Of Day','# of Rides']
```

```
# print the first few rows
rides2.head()
```

Out[35]:

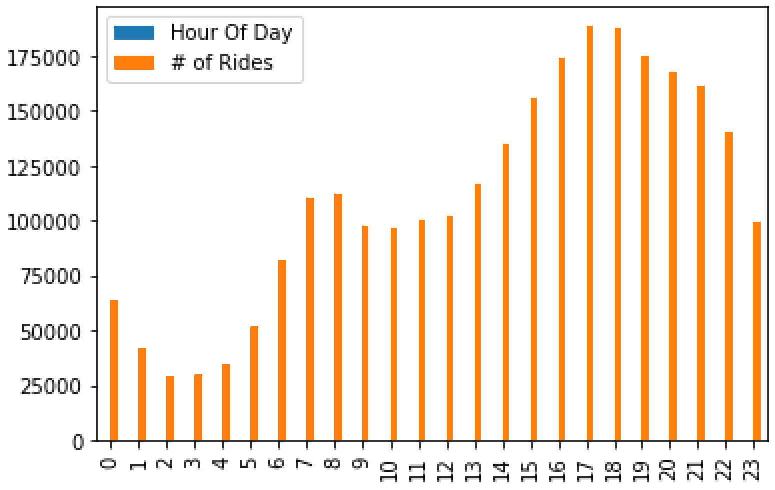
	Hour Of Day	# of Rides
0	0	63537
1	1	42105
2	2	29369
3	3	30364
4	4	34489

In [36]:

```
# create a barchart to display the number of trips per hour
rides2[['Hour Of Day', '# of Rides']].plot(kind = 'bar')
```

Out[36]:

```
<AxesSubplot:>
```



Aggregate at different time periods by setting date as index

In [37]:

```
# tell python to use the date variable as the index for the uber data. Call this new d
# by setting date as an index, we can then use special functions for aggregating datet
uber['Date'] = pd.to_datetime(uber['Date'])
uber_date_index = uber.set_index('Date')
uber_date_index.head()
```

Out[37]:

Date	Date/Time	Lat	Lon	Base	Time	Date2	Time2	DayOfWeek	Hour
2014-07-01	7/1/2014 0:03:00	40.7586	-73.9706	B02512	0:03:00	2014-07-01	12:03:00	1	0
2014-07-01	7/1/2014 0:05:00	40.7605	-73.9994	B02512	0:05:00	2014-07-01	12:05:00	1	0
2014-07-01	7/1/2014 0:06:00	40.7320	-73.9999	B02512	0:06:00	2014-07-01	12:06:00	1	0
2014-07-01	7/1/2014 0:09:00	40.7635	-73.9793	B02512	0:09:00	2014-07-01	12:09:00	1	0
2014-07-01	7/1/2014 0:20:00	40.7204	-74.0047	B02512	0:20:00	2014-07-01	12:20:00	1	0

In [38]:

```
# aggregate the uber_date_index dataframe by day to get the number of rides per day
# HINT: use resample
# HINT2: save this aggregate data frame as a new object e.g. uberperday
uberperday = uber_date_index.resample('D').count()
uberperday.head()
```

Out[38]:

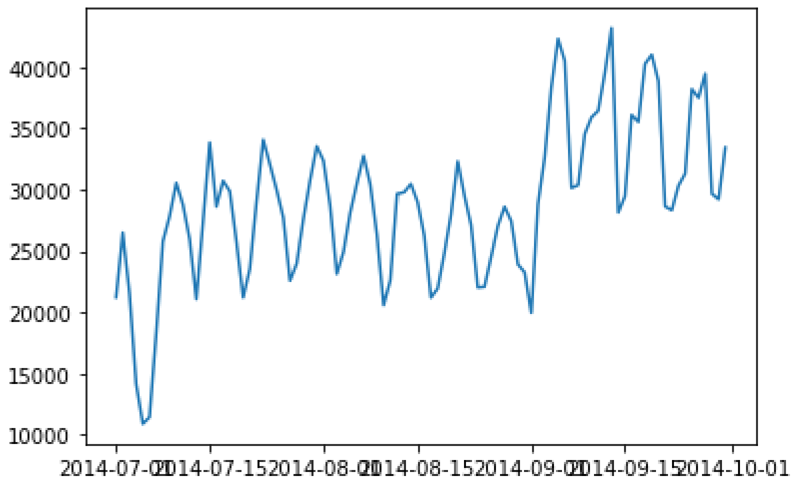
Date	Date/Time	Lat	Lon	Base	Time	Date2	Time2	DayOfWeek	Hour
2014-07-01	21228	21228	21228	21228	21228	21228	21228	21228	21228
2014-07-02	26480	26480	26480	26480	26480	26480	26480	26480	26480
2014-07-03	21597	21597	21597	21597	21597	21597	21597	21597	21597
2014-07-04	14148	14148	14148	14148	14148	14148	14148	14148	14148
2014-07-05	10890	10890	10890	10890	10890	10890	10890	10890	10890

In [39]:

```
# create a line graph of the number of rides per day using the aggregated uber data (u
plt.plot(uberperday.index,uberperday['Time'])
```

Out[39]:

```
[<matplotlib.lines.Line2D at 0x1b95c9c85e0>]
```



```
# aggregate the uberperday data set by week to get the average number of rides per day
```

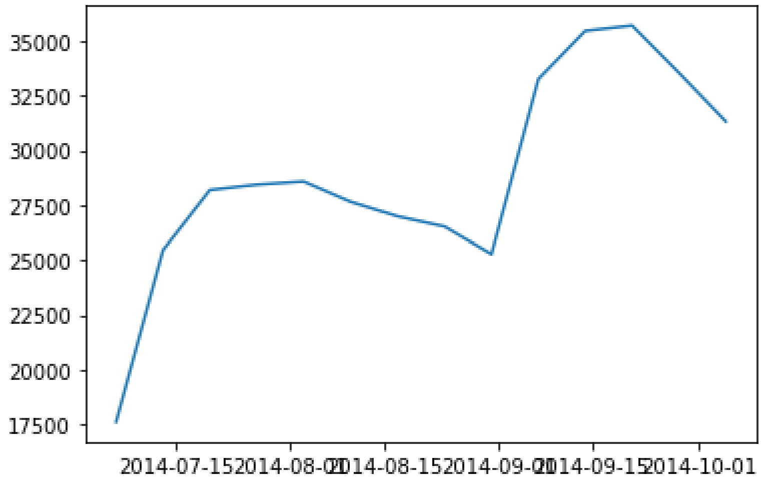
In [44]:
```
# HINT: use resample
# HINT2: save this aggregate data frame as a new object e.g. uberperweek
uberperweek = uberperday.resample('W').mean()
uberperweek.head()
```

Out[44]:

	Date/Time	Lat	Lon	Base	Time	Date2	Ti
Date							
2014-07-06	17631.000000	17631.000000	17631.000000	17631.000000	17631.000000	17631.000000	17631.00(
2014-07-13	25453.000000	25453.000000	25453.000000	25453.000000	25453.000000	25453.000000	25453.00(
2014-07-20	28187.142857	28187.142857	28187.142857	28187.142857	28187.142857	28187.142857	28187.142
2014-07-27	28429.000000	28429.000000	28429.000000	28429.000000	28429.000000	28429.000000	28429.00(
2014-08-03	28575.428571	28575.428571	28575.428571	28575.428571	28575.428571	28575.428571	28575.428

In [45]:
```
# create a line graph of the average number of rides per day per week using the aggreg
plt.plot(uberperweek.index, uberperweek['Time'])
```

Out[45]:
```
[<matplotlib.lines.Line2D at 0x1b9b37d34f0>]
```



In [ ]: