Check out some data

In this activity we will check out ipython install and the availability of some important modules. Furthermore, we'll visualize a dataset, learning a bit about the power of Pandas.

Note: Activity adapted from nbviewer (http://nbviewer.jupyter.org/github/jvns/pandas-cookbook/blob/v0.1/cookbook/Chapter%205%20-%20Combining%20dataframes%20and%20scraping%20Canadian%20weather%20data.ipynb). Montréal cycling data from Données Ouvertes Montréal (http://donnees.ville.montreal.qc.ca/dataset/velos-comptage)

First, let's make sure you have the right modules installed to run activities for this class.

In [1]:
```
import pandas as pd
import numpy as np
import scipy as sp
# Open graphs in new cells in the page rather than in a separate window
%matplotlib inline
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (15, 5)  # set a default figure size
```

You can read data from a CSV file using the `read_csv` function. By default, it assumes that the fields are comma-separated.

We're going to be looking some cyclist data from Montréal. Here's the original page (http://donnees.ville.montreal.qc.ca/dataset/velos-comptage) (in French), but it's already included in this repository. We're using the data from 2012.

This dataset is a list of how many people were on 7 different bike paths in Montreal, each day.

In [21]:
```
broken_df = pd.read_csv('data/bikes.csv','rt', encoding='latin1', engine='python'
broken_df
```

```
C:\Users\Checkout\AppData\Local\Programs\Python\Python310\lib\site-packages\IPy
thon\core\interactiveshell.py:3398: FutureWarning: In a future version of panda
s all arguments of read_csv except for the argument 'filepath_or_buffer' will b
e keyword-only.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[21]:

	Date;Berri 1;Brébeuf (données non disponibles);Côte-Sainte-Catherine;Maisonneuve 1;Maisonneuve 2;du Parc;Pierre-Dupuy;Rachel1;St-Urbain (données non disponibles)
0	01/01/2012;35;;0;38;51;26;10;16;
1	02/01/2012;83;;1;68;153;53;6;43;
2	03/01/2012;135;;2;104;248;89;3;58;
3	04/01/2012;144;;1;116;318;111;8;61;
4	05/01/2012;197;;2;124;330;97;13;95;
...	...
305	01/11/2012;2405;;1208;1701;3082;2076;165;2461
306	02/11/2012;1582;;737;1109;2277;1392;97;1888
307	03/11/2012;844;;380;612;1137;713;105;1302
308	04/11/2012;966;;446;710;1277;692;197;1374
309	05/11/2012;2247;;1170;1705;3221;2143;179;2430

310 rows × 1 columns

In [22]:
```
# Look at the first 3 rows
broken_df[:3]
```

Out[22]:

	Date;Berri 1;Brébeuf (données non disponibles);Côte-Sainte-Catherine;Maisonneuve 1;Maisonneuve 2;du Parc;Pierre-Dupuy;Rachel1;St-Urbain (données non disponibles)
0	01/01/2012;35;;0;38;51;26;10;16;
1	02/01/2012;83;;1;68;153;53;6;43;
2	03/01/2012;135;;2;104;248;89;3;58;

You'll notice that this is totally broken! `read_csv` has a bunch of options that will let us fix that, though. Here we'll

- change the column separator to a `;`
- Set the encoding to `'latin1'` (the default is `'utf8'`)
- Parse the dates in the 'Date' column
- Tell it that our dates have the date first instead of the month first
- Set the index to be the 'Date' column

In [7]:
```
fixed_df = pd.read_csv('data/bikes.csv', sep=';', encoding='latin1', parse_dates=
fixed_df[:3]
```

Out[7]:

Date	Berri 1	Brébeuf (données non disponibles)	Côte-Sainte-Catherine	Maisonneuve 1	Maisonneuve 2	du Parc	Pierre-Dupuy	Rachel1	St-Urb (donn disponibl
2012-01-01	35	NaN	0	38	51	26	10	16	N
2012-01-02	83	NaN	1	68	153	53	6	43	N
2012-01-03	135	NaN	2	104	248	89	3	58	N

When you read a CSV, you get a kind of object called a `DataFrame`, which is made up of rows and columns. You get columns out of a DataFrame the same way you get elements out of a dictionary.

Here's an example:

In [8]:
```
fixed_df['Berri 1']
```

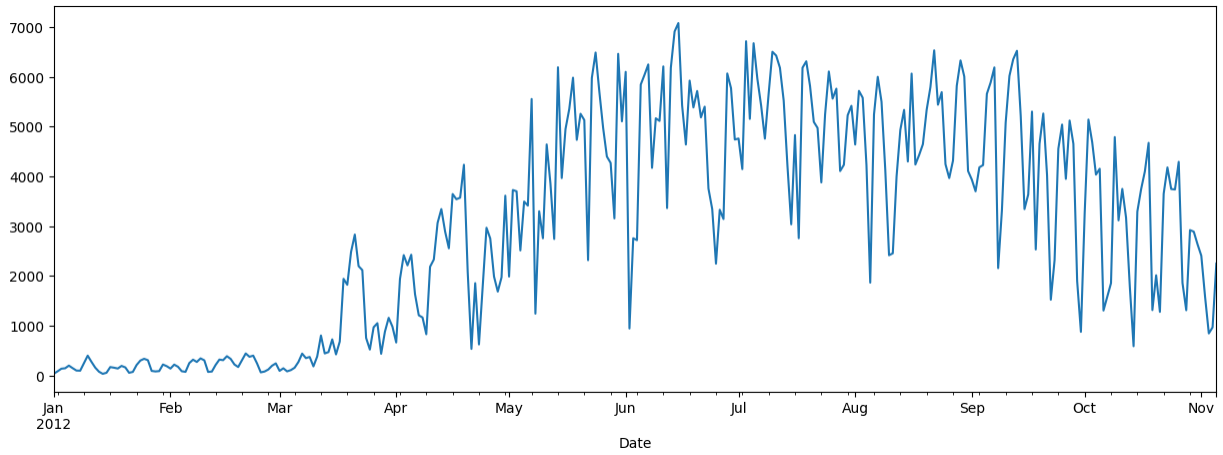
Out[8]:
```
Date
2012-01-01      35
2012-01-02      83
2012-01-03     135
2012-01-04     144
2012-01-05     197
               ...
2012-11-01    2405
2012-11-02    1582
2012-11-03     844
2012-11-04     966
2012-11-05    2247
Name: Berri 1, Length: 310, dtype: int64
```

Just add `.plot()` to the end to plot! How could it be easier? =)

We can see that, unsurprisingly, not many people are biking in January, February, and March,

In [9]:
```
fixed_df['Berri 1'].plot()
```

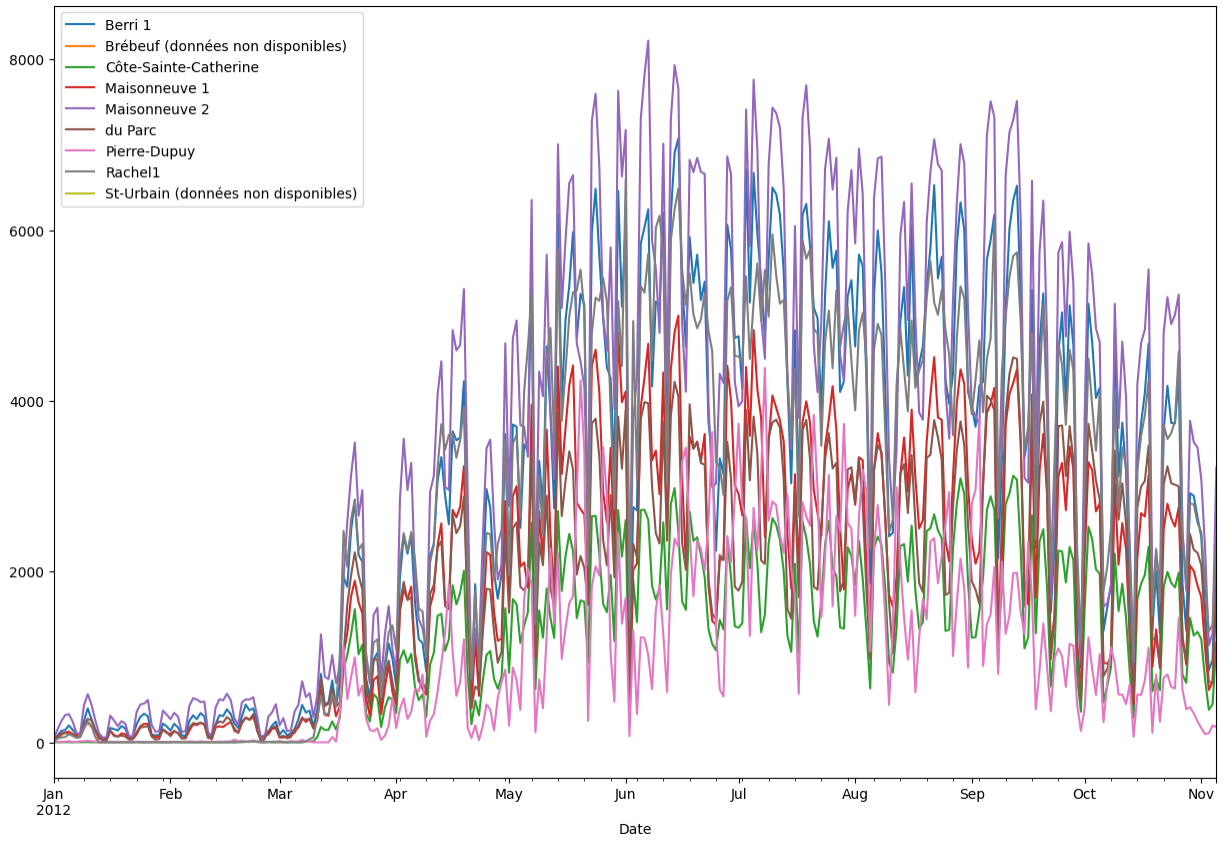
Out[9]:
```
<AxesSubplot:xlabel='Date'>
```



We can also plot all the columns just as easily. We'll make it a little bigger, too. You can see that it's more squished together, but all the bike paths behave basically the same -- if it's a bad day for cyclists, it's a bad day everywhere.

In [10]:
```
fixed_df.plot(figsize=(15, 10))
```

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



In [ ]:
```

```