pandas - Outline

- Overview
- Purpose
- Terminology
- Series
- DataFrame
- Functionality
- Data Loading
- Plotting
- What else can pandas do
- Question
pandas - Overview

- Python Data Analysis Library, similar to:
  - R
  - MATLAB
  - SAS
- Combined with the IPython toolkit
- Built on top of NumPy, SciPy, to some extent matplotlib
- **Panel Data System**
- Open source, BSD-licensed
- Key Components
  - Series
  - DataFrame
pandas - Purpose

- Ideal tool for data scientists
- Munging data
- Cleaning data
- Analyzing data
- Modeling data
- Organizing the results of the analysis into a form suitable for plotting or tabular display
pandas - Terminology

- **IPython** is a command shell for interactive computing in multiple programming languages, especially focused on the Python programming language, that offers enhanced introspection, rich media, additional shell syntax, tab completion, and rich history.

- **NumPy** is the fundamental package for scientific computing with Python.
SciPy (pronounced “Sigh Pie”) is a Python-based ecosystem of open-source software for mathematics, science, and engineering.

Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Data Munging or Data Wrangling means taking data that's stored in one format and changing it into another format.
Cython programming language is a superset of Python with a foreign function interface for invoking C/C++ routines and the ability to declare the static type of subroutine parameters and results, local variables, and class attributes.
pandas - Data Structures: Series

- One-dimensional array-like object containing data and labels (or index)
- Lots of ways to build a Series

```python
>>> import pandas as pd
>>> s = pd.Series(list('abcdef'))
>>> s
0    a
1    b
2    c
3    d
4    e
5    f
>>> s = pd.Series([2, 4, 6, 8])
>>> s
0    2
1    4
2    6
3    8
```
Series - Working with the index

- A series index can be specified
- Single values can be selected by index
- Multiple values can be selected with multiple indexes

```python
>>> s = pd.Series([2, 4, 6, 8], index=['f', 'a', 'c', 'e'])
>>> s
f   2
a   4
c   6
e   8
>>> s['a']
4
>>> s[['a', 'c']]
a   4
c   6
```
Series - Working with the index

- Think of a Series as a fixed-length, order dict
- However, unlike dict, index items don't have to be unique
Series - Operations

- Filtering
- NumPy-type operations on data
Series - Incomplete data

- pandas can accommodate incomplete data

```python
>>> sdata = {'b':100, 'c':150, 'd':200}
>>> s = pd.Series(sdata)
>>> s
b  100
 c  150
d  200
```

```python
>>> s = pd.Series(sdata, list('abcd'))
>>> s
a  NaN
b  100
 c  150
d  200
```

```python
>>> s*2
a  NaN
b  200
c  300
d  400
```
Series - Automatic alignment

- Unlike in NumPy ndarray, data is automatically aligned.
Data Structures: DataFrame

- Spreadsheet-like data structure containing an order collection of columns
- Has both a row and column index
- Consider as dict of Series (with shared index)
DataFrame

Creation with dict of equal-length lists

```python
>>> data = {'state': ['FL', 'FL', 'GA', 'GA', 'GA'],
          'pop': [18.8, 19.1, 9.7, 9.7, 9.8]}

>>> frame = pd.DataFrame(data)

>>> frame
   pop state year
0  18.8   FL  2010
1  19.1   FL  2011
2   9.7   GA  2008
3   9.7   GA  2010
4   9.8   GA  2011
```
DataFrame

Creation with dict of dicts

```python
>>> pop_data = {'FL': {2010: 18.8, 2011: 19.1},
>>> pop = pd.DataFrame(pop_data)
>>> pop
   FL  GA
2008 NaN 9.7
2010 18.8 9.7
2011 19.1 9.8
```
DataFrame

- Columns can be retrieved as Series
  - dict notation
  - attribute notation
- Rows can retrieved by position or by name (using ix attribute)
New Columns can be added (by computation or direct assignment)
DataFrame - Reindexing

- Creation of new object with the data conformed to a new index
Functionality

Summarizing and Descriptive Statistics

```python
>>> pop
   FL   GA
2008  NaN  9.7
2010  18.8  9.7
2011  19.1  9.8
>>> pop.sum()
    FL   GA
   37.9  29.2
>>> pop.mean()
    FL   GA
  18.95  9.7333
>>> pop.describe()
      FL   GA
count 2.0000 3.0000
mean  18.9500 9.7333
std   0.2121 0.0577
min   18.8000 9.7000
25%   18.8750 9.7000
50%   18.9500 9.7000
75%   19.0250 9.7500
max   19.1000 9.8000
```
Functionality

Boolean indexing

```python
>>> pop
   FL   GA
2008   NaN  9.7
2010  18.8  9.7
2011  19.1  9.8
>>> pop < 9.8
   FL   GA
2008    False  True
2010    False  True
2011    False  False
>>> pop[pop < 9.8] = 0
>>> pop
   FL   GA
2008   NaN   0.0
2010  18.8   0.0
2011  19.1   9.8
```
Data Loading

- pandas supports several ways to handle data loading
- Text file data
  - read_csv
  - read_table
- Structured data (JSON, XML, HTML)
  - works well with existing libraries
- Excel (depends upon xlrd and openpyxl packages)
- Database
  - pandas.io.sql module (read_frame)
Plotting

```python
>>> tips = pd.read_csv('/users/ah6/Desktop/pandas
talk/data/tips.csv')
>>> tips.ix[:2]
    total_bill  tip   sex  smoker day  time  size
0        16.99  1.01  Female    No  Sun  Dinner   2
1        10.34  1.66  Male     No  Sun  Dinner   3
2        21.01  3.50  Male     No  Sun  Dinner   3

>>> party_counts = pd.crosstab(tips.day, tips.size)

>>> party_counts
size  1  2  3  4  5  6
day
Fri  1  1  0  0
Sat  2  1  3  1  0
Sun  0  2  1  3  1
Thur 1  2  4  5  1  3

>>> sum_by_day = party_counts.sum(1).astype(float)
```
Plotting

```python
>>> party_pcts = party_counts.div(sum_by_day, axis=0)
>>> party_pcts
size 1 2 3 4 5 6
day Fri 0.052632 0.842105 0.052632 0.052632 0.000000 0.000000
Sat 0.022989 0.609195 0.206897 0.149425 0.011494 0.000000
Sun 0.000000 0.513158 0.197368 0.236842 0.039474 0.013158
Thur 0.016129 0.774194 0.064516 0.080645 0.016129 0.048387
>>> party_pcts.plot(kind='bar', stacked=True)
<matplotlib.axes.AxesSubplot at 0x6bf2>
```
Plotting

```python
>>> tips['tip_pct'] = tips['tip'] / tips['total_bill']
>>> tips['tip_pct'].hist(bins=50)
<matplotlib.axes._subplots.AxesSubplot at 0x6c10d30>
>>> tips['tip_pct'].describe()
count    244.000000
mean      0.160803
std       0.061072
min       0.035638
25%       0.129127
50%       0.154770
75%       0.191475
max       0.710345
```
What else?

- **Data Aggregation**
  - GroupBy
  - Pivot Tables

- **Time Series**
  - Periods/Frequencies
  - Operations with Time Series with Different Frequencies
  - Downsampling/Upsampling
  - Plotting with TimeSeries (auto-adjust scale)

- **Advanced Analysis**
  - Decile and Quartile Analysis
  - Signal Frontier Analysis
  - Future Contract Rolling
  - Rolling Correlation and Linear Regression
Questions?
pandas - Bibliography

- pandas - Python Data Analysis. [http://www.slideshare.net/AndrewHenshaw1/pandas-22984889](http://www.slideshare.net/AndrewHenshaw1/pandas-22984889)
- Getting started with pandas. [http://www.slideshare.net/maikroeder/getting-started-with-pandas](http://www.slideshare.net/maikroeder/getting-started-with-pandas)
pandas - Bibliography

- **NumPy.** [http://www.numpy.org/](http://www.numpy.org/)
- **Matplotlib.** [http://matplotlib.org/](http://matplotlib.org/)