



Python for data science

Alexandre Chevallier, Jérémy Richard, Geneviève Michaud, Baptiste Rouxel

► To cite this version:

Alexandre Chevallier, Jérémy Richard, Geneviève Michaud, Baptiste Rouxel. Python for data science. Doctoral. Summer School 2017, Trivedi Centre for Political Data (TCPD) Ashoka University, India. 2017. hal-03923285

HAL Id: hal-03923285

<https://sciencespo.hal.science/hal-03923285>

Submitted on 4 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Python for data science

#1 Introduction to Python and Jupyter

Trivedi Center for Political Data (TCPD) 2017 Summer School,
Technical module, July 11th 2017

Alexandre Chevallier (CDSP, SciencesPo and CNRS)
Jérémy Richard (SciencesPo)

Outline

Introduction

Adopt Python

Jupyter

Who are we?

Jérémy Richard, Sysadmin engineer.

Joined Sciences Po in 2012.

Missions: Building IT infrastructures for two laboratories of the scientific department:

- CDSP - Center For Socio-Political Data
- Médialab

Who are we?

Alexandre Chevallier, Web Developer.

Joined the CDSP of Sciences Po in 2014

Missions: Develop tools/applications for several projects, among which:

- ELIPSS project (Internet online panel for the Social Sciences)
- Bequali (Qualitative data bank)
- support for CDSP data and metadata related activities

Who are we?

- CDSP - Center for Socio Political Data
- CNRS and Sciences Po mixed service unit for social science researchers
- Helps researchers with their data and metadata, collect, curate and disseminated data and metadata.
- Involved in several projects such as:
 - ELIPSS (Digital panel for quantitative surveys)
 - beQuali (Archive & share qualitatives studies)
 - Vizlab (Visualization for french election data)

Why are we here?

Dispelling a few myths

- Myth 01: I have to go to university to learn how to code.
- Myth 02: I need to be a genius to code

Share IT knowledge

Helping you decide if our techs suit your use

Adopt Python - Brief

- Invented by Guido van Rossum, in the Netherlands early 90s
- Python is a general purpose open source programming language
- Often used as a scripting tool
- It could be also called an interpreted language (≠ compiled language)
- Swissknife language

Adopt Python - Scope

Data science

- Data analysis
- Data visualization

System information

- Scripting
- Deployed on most of Unix systems

Software development

- Web applications
- Testing scripts

Education

- Teaching programming

Adopt Python - Why use it ?

Object-oriented

Native indentation restriction

- Very clear readable syntax

Powerful

- Very high level data types
- Other language interoperoperation (e.g. C/C++)
- Python Package Index (pip)

Community support

- Open-Source
- Portable
- Free

Adopt Python - Data Structures

- List → [1, 2, 3, 4, 5, “hello”] : Ordered series of values
 - add data `list.append(1)`
- Dictionary → { “key” : “value”, “hello” : 1 } : Key/Value data structure
 - add data `dict[‘key’] = ‘value’`
- Tuple → (1, 2, 3, 4, “hello”) : Like list but immutable

Jupyter - Python libraries we use

Web Scraping: **BeautifulSoup**

- HTML/XML Parser
- Built on top of popular Python parsers like lxml and html5lib
- Designed for quick turnaround projects

Data Visualization: **Pandas**

- Similar to R, MATLAB, SAS
- Built on top of Numpy, Scipy and Matplotlib
- Handle a vast majority of typical use cases in finance, statistics, social science, and many areas of engineering

Jupyter - Brief

Web platform for Data Science.

Support for over 40 programming languages.

IPython is an interactive shell for Python.

Create and share notebooks that contains:

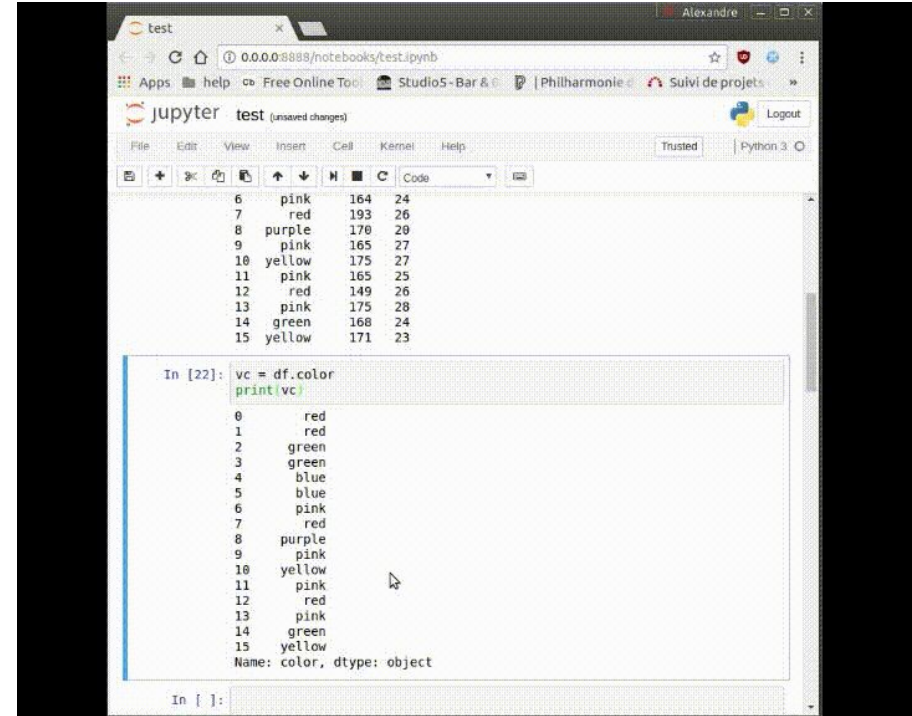
- live code
- equations
- visualizations
- text

Jupyter - Notebooks

Interactive way to learn, experiment and share your work.

Tool of choice for Data scientists.

Runs in every recent web browser.

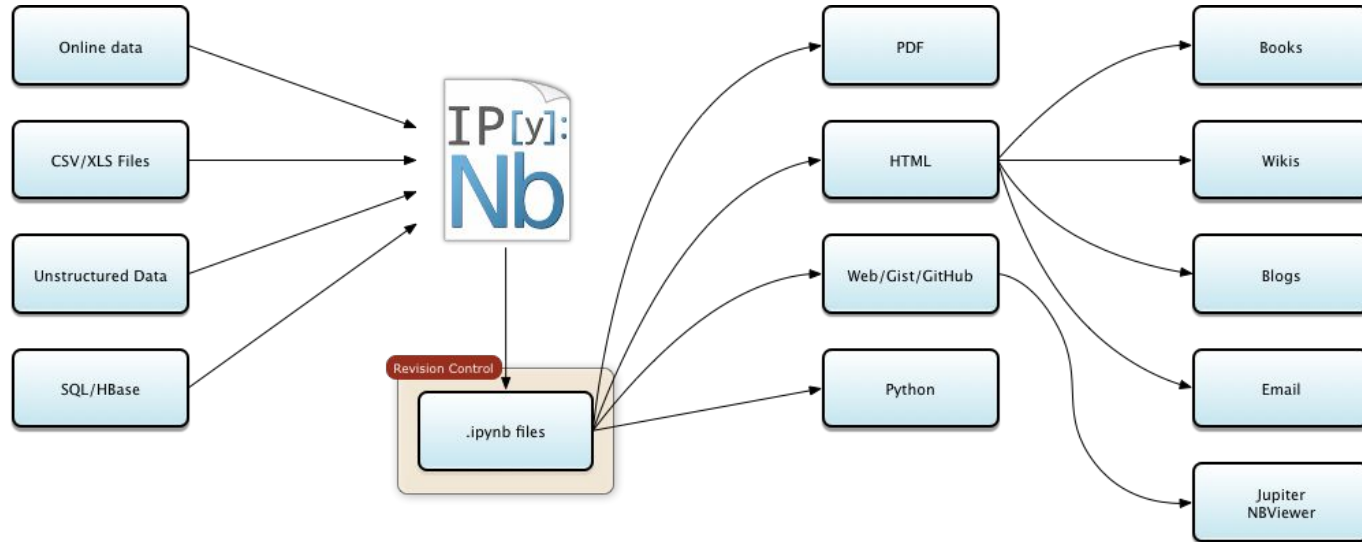


The screenshot shows a Jupyter Notebook running in a web browser. The top part of the notebook displays a data frame with 15 rows and 4 columns. The bottom part shows the execution of a code cell that prints the 'color' attribute of the data frame.

6	pink	164	24
7	red	193	26
8	purple	170	20
9	pink	165	27
10	yellow	175	27
11	pink	165	25
12	red	149	26
13	pink	175	28
14	green	168	24
15	yellow	171	23

```
In [22]: vc = df.color  
         print(vc)  
0      red  
1      red  
2    green  
3    green  
4     blue  
5     blue  
6     pink  
7      red  
8    purple  
9     pink  
10    yellow  
11     pink  
12     red  
13     pink  
14    green  
15    yellow  
Name: color, dtype: object
```

Jupyter - Notebook: input and output file formats



Alexandre Chevallier - Jérémy Richard
itcdsp-scpolst@sciencespo.fr

Python for data science

#2 Web Scraping

Trivedi Center for Political Data (TCPD) 2017 Summer School,
Technical module, July 11th 2017

Alexandre Chevallier (CDSP, SciencesPo and CNRS)
Jérémy Richard (SciencesPo)

Outline

Web scraping

Web page

BeautifulSoup Library

Practical Works

Web Scraping - What is it?

Data Scraping?

- Automated process
- Explore and download raw data
- Grab content
- Convert data in usable format for analysis
- Store data in database or text file

Web Scraping = Data Scraping of web pages

Web Scraping - What is a web page ?

Components of a web page

- HTML - Organizes and contains the main content of a web page
- CSS - Add styling to make the page looks nicer
- JS - Javascript files add interactivity to web pages
- Media files - Images, Sounds, Videos, etc.

Interesting content for web scraping = **HTML**

Web Scrapping - HTML

HTML is used to create documents on the Web

Very simple and logical

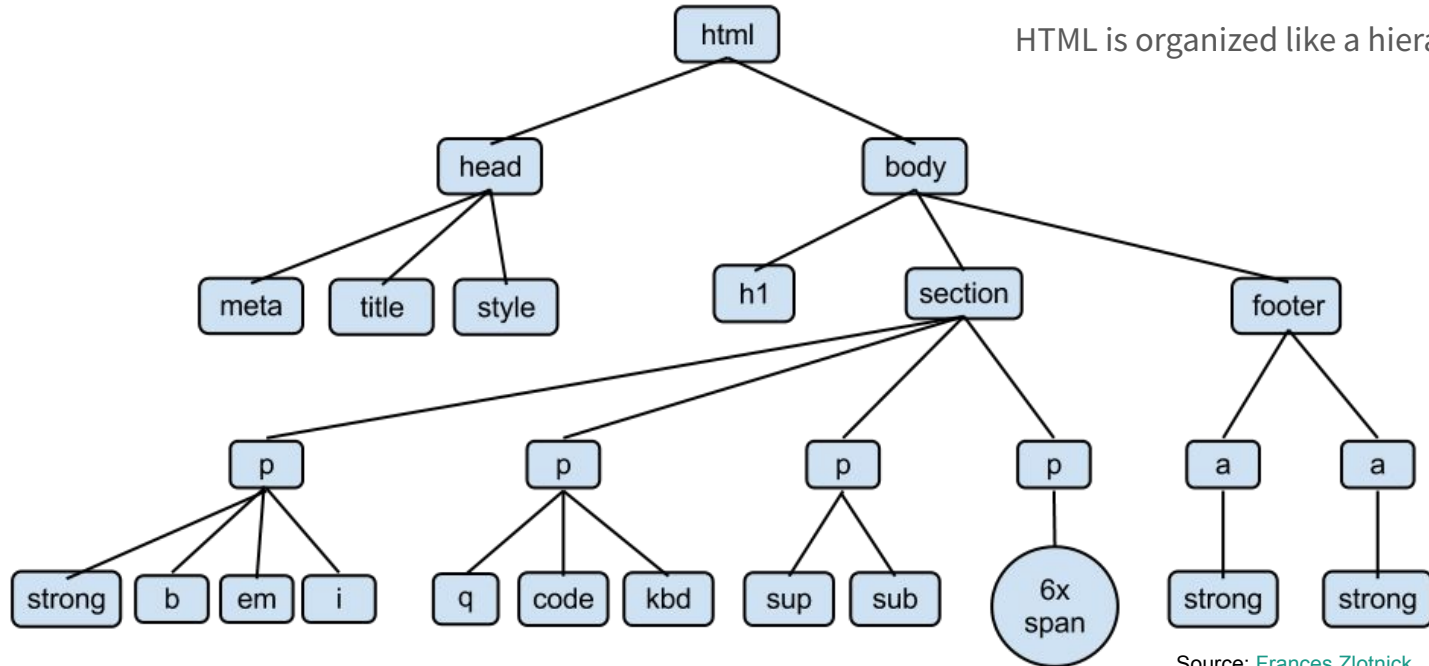
NOT a programming language but a **markup** language that uses <tags> like this

The websites you view are basically HTML files rendered by web browsers

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Example</title>
5          <link rel="stylesheet" href="style.css">
6      </head>
7      <body>
8          <h1>
9              <a href="/">Header</a>
10         </h1>
11         <nav>
12             <a href="/one/">One</a>
13             <a href="/two/">Two</a>
14             <a href="/three/">Three</a>
15         </nav>
```

Web Scraping - HTML

HTML is organized like a hierarchical tree



Source: [Frances Zlotnick](#)

Web Scraping - Inspect the source

Inspect element
Find HTML nodes

<table> defines a table
<tr> defines a row in a table
<th> defines a table header
cell
<td> defines a cell in table

Use BeautifulSoup to grab it

Delhi MCD Election Results 2017

MCD Election Results 2017

MCD Election Results – Party Wise

Party Name	Leading/Won (2017)	2012 Results
AAP	47	AAP did not contest in 2012.
BJP	184	138
Congress	30	77
Others	10	57

Ward Wise MCD Election 2017 Results

```
<div class="left-links-bx"></div>
<!--end left link box section-->
<div class="ad160" style="display: none !important;"></div>
<div class="ad160" style="display: none !important;"></div>
<div class="ad160" style="display: none !important;"></div>
</div>
<!--Side panel starts-->
<div id="content-main">
  <div class="content-panel">
    <div class="header-base"></div>
    <div id="dump"></div>
    <div class="cl"></div>
    <div class="text">
      <!--5535731/Elections-below-h1-new-->
      <div id="div-gpt-ad-1458386879591-0" style="height: 90px; width: 728px; display: none !important;"></div>
      <h2>MCD Election Results – Party Wise</h2>
      <table class="tableize-table" style="background:none!important" cellspacing="0">
        <tbody>
          <tr>
            <th="">Party Name</th>
            <th=""></th>
            <th>2012 Results</th>
          </tr>
          <tr class="L_w">
            <td class="atable">AAP</td>
            <td class="atable">47</td>
            <td class="atable">AAP did not contest in 2012.</td>
          </tr>
          <tr class="L_w">
            <td class="atable">BJP</td>
            <td class="atable">184</td>
            <td class="atable">138</td>
          </tr>
          <tr class="L_w">
            <td class="atable">Congress</td>
            <td class="atable">30</td>
            <td class="atable">77</td>
          </tr>
          <tr class="L_w">
            <td class="atable">Others</td>
            <td class="atable">10</td>
            <td class="atable">57</td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```

Web Scraping - BeautifulSoup

- A Python library
- Pull out data out of HTML/XML files
- Designed for quick turnaround projects
- Charged with some superb methods
- Open-source, free & well documented

Web Scraping - Jump into the code

```
#Grab node with BeautifulSoup
from BeautifulSoup import BeautifulSoup
import urllib

raw_html =
urllib.urlopen('http://www.elections.in/delhi/mcd-elections/').read()

soup = BeautifulSoup(raw_html)

attrs = { 'class': 'tableizer-table' }
tables = soup.findAll(attrs=attrs)
table = tables[0]
rows = table.findAll('tr')
```

} Import librairies

} Download data

} Instantiate
BeautifulSoup object

} Access the data

Use grabbed data to write a CSV file

Web Scraping - Jump into the code

```
import csv
```

```
with open('export.csv', 'wb') as f:
    writer = csv.writer(f, delimiter=';')
    for row in rows:
        csv_row = []
        headers = row.findAll('th')
        for header in headers:
            csv_row.append(header.text)
        cells = row.findAll('td')
        for cell in cells:
            csv_row.append(cell.text)
        writer.writerow(csv_row)
```

} Import the CSV library

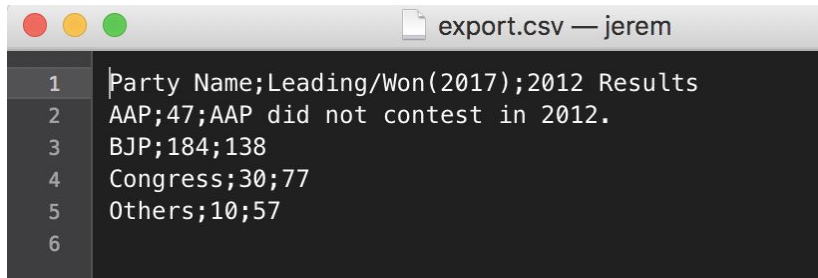
} Open a file with write permissions
} Handle it with CSV lib's methods

} Make loops for selecting data
inside table cells.
} Write it in a python list

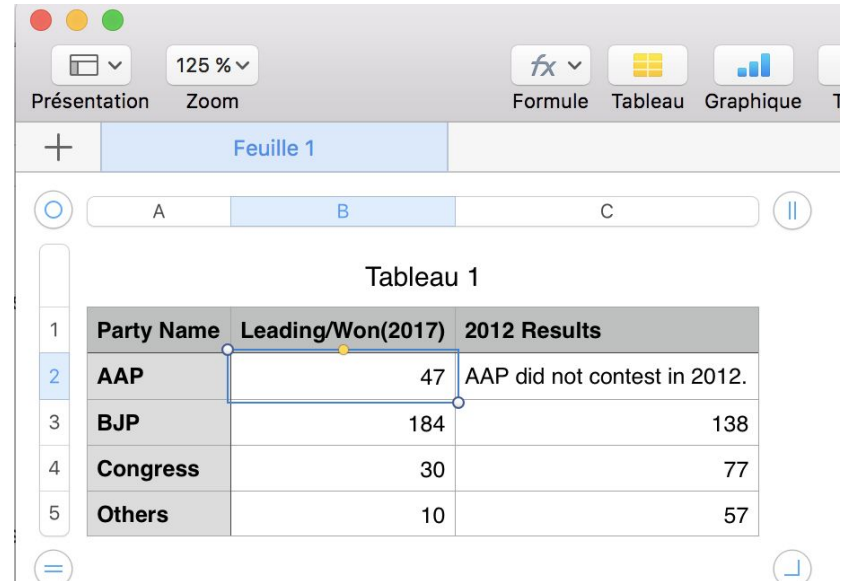
} Write list in CSV handle file

Web Scrapping - Jump into the code

Extraction Result



```
1 Party Name;Leading/Won(2017);2012 Results
2 AAP;47;AAP did not contest in 2012.
3 BJP;184;138
4 Congress;30;77
5 Others;10;57
6
```



	Party Name	Leading/Won(2017)	2012 Results
1			
2	AAP	47	AAP did not contest in 2012.
3	BJP	184	138
4	Congress	30	77
5	Others	10	57

Alexandre Chevallier - Jérémy Richard
itcdsp-scpolst@sciencespo.fr

Python for data science

#3 Data Visualization

Trivedi Center for Political Data (TCPD) 2017 Summer School,
Technical module, July 11th 2017

Alexandre Chevallier (CDSP, SciencesPo and CNRS)
Jérémy Richard (SciencesPo)

Outline

Pandas library

Visualization

Practical works

Data Visualization - Brief

Pandas - **P**anel **D**ata **S**ystem

Used in production in many companies, especially in financial industries

Suitable for many different kinds of data

Two primary data structures:

- Series (1 dimensional)
- DataFrame (2 dimensional). For R's users, it's like R's `data.frame` on steroids.

Data Visualization - Series

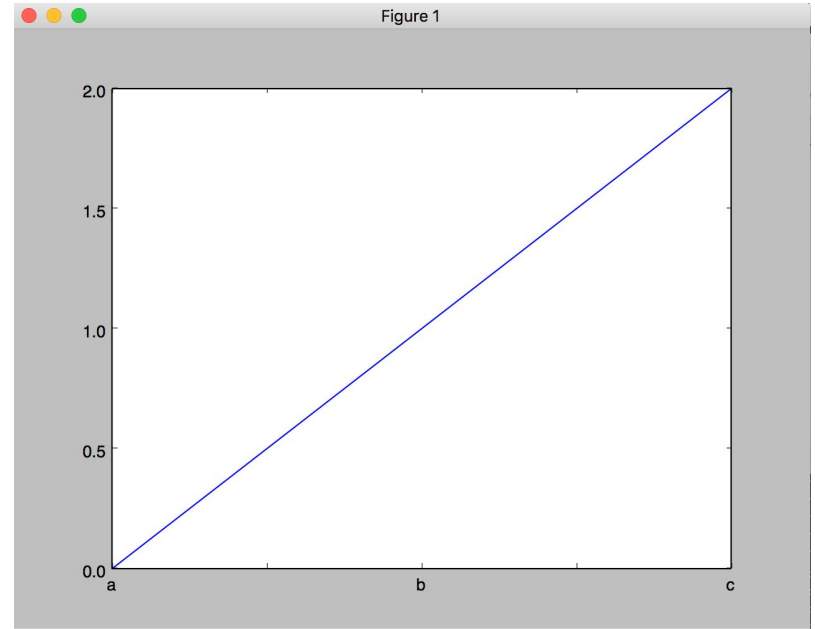
1 dimensional

```
from pandas import Series
data = {'a' : 0., 'b' : 1., 'c' : 2.}
s = Series(data)
print(s)
a      0.0
b      1.0
c      2.0
```

} Import pandas library
} Create python ordered dictionary with data
} Instantiate Series object
} Show variable content

Data Visualization - Series

```
import matplotlib.pyplot as plt  
s.plot()  
plt.show()
```



Data Visualization - Series

```
s = s.reindex(['c','a','b'])
```

```
print(s)
```

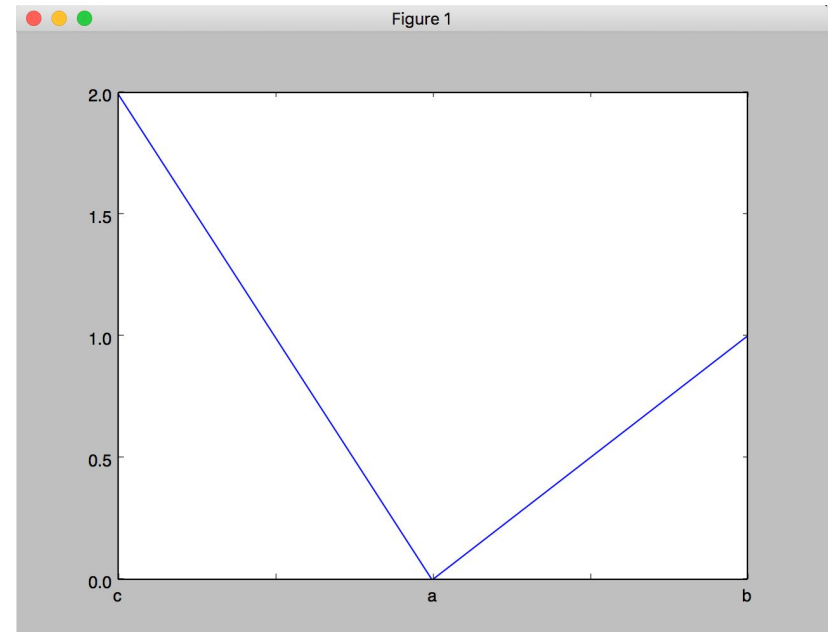
```
c    2.0
```

```
a    0.0
```

```
b    1.0
```

```
s.plot()
```

```
plt.show()
```



Data Visualization - DataFrame

2 dimensional table data structure

Like `data.frame` in R

Data manipulation with integrated indexing

Support heterogeneous type of columns

Data Visualization - DataFrame

```
#File input/output
import pandas as pd
data = pd.read_csv('2012-electoral-college.csv', sep=';',
index_col='State')
data.head()
```

	Name	Electors	Population
State			
AK	Alaska	3	710000
AL	Alabama	9	4780000
AR	Arkansas	6	2916000
AZ	Arizona	11	6392000
CA	California	55	37254000

The screenshot shows a spreadsheet application window titled "2012-electoral-college". The interface includes a menu bar with options like "Présentation", "Zoom", "Formule", "Tableau", "Graphique", "Texte", "Format", and "Trier et filtrer". Below the menu bar, there's a tab labeled "Feuille 1". The spreadsheet grid shows columns A, B, C, and D. The data is organized as follows:

	State	Name	Electors	Population
1				
2	AK	Alaska	3	710000
3	AL	Alabama	9	4780000
4	AR	Arkansas	6	2916000
5	AZ	Arizona	11	6392000
6	CA	California	55	37254000
7	CO	Colorado	9	5029000
8	CT	Connecticut	7	3574000
9	DC	Dist. of Col.	3	602000
10	DE	Delaware	3	898000
11	FL	Florida	29	18801000

Data Visualization - DataFrame

```
#Analysis
>>> data.Electors.mean()
10.549019607843137
>>> data.Electors.max()
55
>>> data.loc[data.Electors.argmax(), 'Name']
'California'
>>> data.Population.sum()
308746000
>>> data['ratio'] =
data['Electors']/data['Population']
>>> data
```

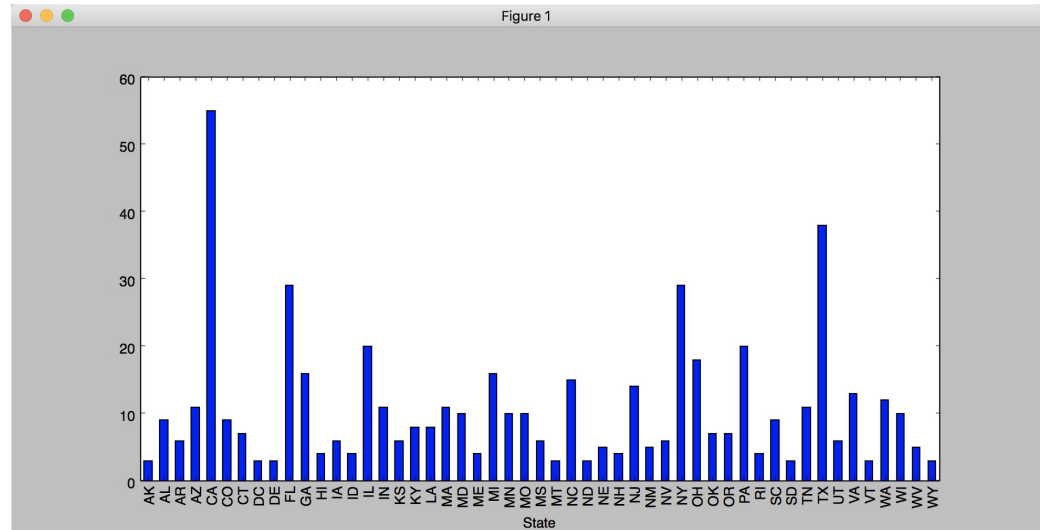
State	Name	Electors	Population	ratio
AK	Alaska	3	710000	0.000004
AL	Alabama	9	4780000	0.000002
AR	Arkansas	6	2916000	0.000002
AZ	Arizona	11	6392000	0.000002
[...]				

Data Visualization - DataFrame

```
#Visualization with matplotlib

import matplotlib.pyplot() as plt

data.Electors.plot.bar()
plt.show()
```



Data Visualization - Go further

And much more

- Group By
- Merge, join, aggregation
- Reshaping and Pivot Tables
- Time based series, date functions
- Multi-index
- ...

Alexandre Chevallier - Jérémy Richard
itcdsp-scpolst@sciencespo.fr