Python 数据分析实践（Data Analysis Action）

Chap 1 概述和环境配置（Preliminaries）

1. Python 环境配置和安装

2. 重要的 Python 库

3. Hello, Python

4. 几个Python的小示例

内容：

- Python 环境配置

实践：

- Anaconda 或 Pycharm 安装
- 基础环境 ipython / ipython notebook
- Install package / Import package
- Python的魔法函数
- 运行Python程序的方式
- 几个Python的小示例

1. Python 环境配置和安装

由于人们使用 Python 所做的事情不同，所以没有普适的 Python 及其插件包的安装方案。要想使 Python 的科学计算环境满足我们这个课程的需要，建议使用下面的 Python 安装包之一。对于初学者，**建议使用第一种 Anaconda 安装包。**

了解如何安装 IPython 到 http://ipython.org/install.html#i-am-getting-started-with-python (http://ipython.org/install.html#i-am-getting-started-with-python)

1). Anaconda

到 Anaconda 下载页面 (https://www.anaconda.com/distribution/)下载和安装适合系统的安装包，建议选择 Python 3 版本 (2020 年起，Python 2.7 版本不再维护)。

下载 Anaconda3，并安装到 你的驱动器:/Anaconda3/ **目录下，例如：C:/Anaconda3/ 或者 D:/Anaconda3/**

打开 Anaconda Command Prompt (C:/Anaconda3/)，在命令提示行中，执行如下的命令更新库:

- conda update conda # 更新conda
- conda update --all # 更新所有库
- conda update PACKAGE-NAME # 仅更新某个库package-name

如果当前版本默认没有安装库，需要先使用install命令安装库，执行如下命令:

- conda install PACKAGE-NAME

如果查看当前版本安装的库信息，执行如下命令:

- conda list PACKAGE

要启动 ipython notebook, 执行如下命令:

- cd <directory where you want to save notebook>，建议大家新建目录 C:/YourName/ 存放本课程的学习资料
- jupyter notebook 命令启动

```
C:\WINDOWS\system32>d:

D:\>cd iPythonNB

D:\iPythonNB>conda list numpy
# packages in environment at C:\Anaconda3:
#
# Name                    Version                   Build  Channel
numpy                     1.18.1           py37h93ca92e_0
numpy-base                1.18.1           py37hc3f5095_1
numpydoc                  0.9.2                      py_0

D:\iPythonNB>jupyter notebook
```

2). PyCharm

PyCharm 是一种 Python IDE，从 http://www.jetbrains.com/pycharm/download/ (http://www.jetbrains.com/pycharm/download/) 下载安装。

- PyCharm 带有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具，比如调试、语法高亮、Project 管理、代码跳转、智能提示、自动完成、单元测试、版本控制。此外，该 IDE 提供了一些高级功能，以用于支持 Django 框架下的专业 Web 开发。
- PyCharm 的安装需要注册码，适合 Python 高级使用者。PyCharm 提供教育优惠，可以通过学校后缀官方邮箱认证，认证地址 (https://www.jetbrains.com/zh/student/)。

3). Enthought Canopy

来自 Enthought 的面向科学计算的 Python 安装包 https://www.enthought.com/ (https://www.enthought.com/) 。从 http://www.enthought.com (http://www.enthought.com) 下载 EPDFree 的安装包，它可能是一个名字类似于 epd_free-7.3-1-win-x86.msi 的 MSI 安装包。EPDFull 安装包面向高校老师是免费的，需要使用官方邮箱注册。

- Enthought Python Distribution (EPD) 现在改名为 ** Enthought Canopy! **

2. 重要的 Python 库

1). NumPy

即 Numerical Python 的简称，是 Python 科学计算的基础包。数据分析的大部分内容都基于 NumPy 以及构建于其上的库。对于数值型数据，NumPy 数组在存储和处理数据时要比内置的 Python 中的列表数据结构高效得多，因为 NumPy 数组是针对某些对象进行了大量的优化工作。可以提供的功能包括（不限于此）：

- 快速高效的多维数组对象 ndarray
- 用于对数组执行元素级计算以及直接对数组执行数学运算的函数
- 用于读写硬盘上基于数组的数据集的工具
- 线性代数运算、傅立叶变换，以及随机数生成
- 用于将 C、C++、Fortran 代码集成到 Python 的工具（由 C 和 Fortran 编写的库可以直接操作 NumPy 数组中的数据，无需进行数据复制工作）
- 作为在算法之间传递数据的容器

2). pandas

名字源于 panel data（面板数据，是计量经济学中关于多维结构化数据集的一个术语）以及 Python data analysis（Python 数据分析），很适合金融数据分析应用的工具，也是本书中使用的主要工具

- 兼具 NumPy 高性能的数组计算功能以及电子表格和关系型数据库（如 SQL ）灵活的数据处理能力
- 提供了复杂精细的索引功能，以便更为便捷地完成重塑、切片和切块、聚合以及选取数据子集等操作
- 对于金融行业的用户，pandas 提供了大量适用于金融数据的高性能时间序列功能和工具
- 用的最多的 pandas 对象是 DataFrame，是一个面向列（column-oriented）的二维表结构，含有行标和列标

3). matplotlib

matplotlib 是最流行的用于绘制数据图表的 Python 库。它非常适合创建出版物上用的图表，跟 IPython 结合得很好，因而提供了一种非常好用的交互式数据绘图环境。绘制的图表也是交互式的，可以利用绘图窗口中的工具栏放大图表中的某个区域或对整个图表进行平移浏览。

4). IPython

IPython 是 Python 科学计算标准工具集的组成部分，它将其他所有的东西联系到了一起。为交互式和探索式计算提供了一个强健而高效的环境。

- 它是一个增强的 Python shell，目的是提高编写、运行、测试、调试 Python 代码的速度。
- 它主要用于交互式数据处理和利用 matplotlib 对数据进行可视化处理。

除了标准的基于终端的 IPython shell 外，该项目还提供了：

- 一个类似于 Mathematica 的 HTML 笔记本（通过 Web 浏览器连接 IPython）
- 一个基于 Qt 框架的 GUI 控制台，其中含有绘图、多行编辑以及语法高亮显示等功能
- 用于交互式并行和分布式计算的基础架构

5). SciPY

SciPy 是一组专门解决科学计算中各种标准问题域的包的集合。 NumPy 和 SciPy 的有机结合完全可以替代 MATLAB 的计算功能（包括其插件工具箱）。SciPy 的主要包括下面这些包：

- scipy.integrate: 数值积分例程和微分方程求解器
- scipy.linalg: 扩展了由 numpy.linalg 提供的线性代数例程和矩阵分解功能
- scipy.optimize: 函数优化器（最小化器）以及根查找算法
- scipy.signal: 信号处理工具
- scipy.sparse: 稀疏矩阵和稀疏线性系统求解器
- scipy.special: SPECFUN（这是一个实现了许多常用数学函数（如伽玛函数）的 Fortran 库）的包装器
- scipy.stats: 标准连续和离散概率分布（如密度函数、采样器、连续分布函数等）、各种统计检验方法，以及更好的描述统计法
- scipy.weave: 利用内联 C++ 代码加速数组计算的工具

有关 anaconda 的有用信息：

```
conda info  # 显示工具的有用信息
conda search pytables  # 搜索库和软件包，可以下载和安装以及已经安装（由 * 标出）的 pytables 版本
conda list pyt  # 显示pyt开始的所有软件包
```

常用模块的命名惯例

- `import numpy as np`
- `import pandas as pd`
- `import matplotlib.pyplot as plt`

当看到 np.arange 时，就应该想到它引用的是 NumPy 中的 arange 函数。这样做的原因是：在 Python 软件开发过程中，不建议直接引入类似 NumPy 这种大型库的全部内容（不建议 `from numpy import *`）

安装库的命令

- `pip install package-name`

或者在 anaconda 模式下，使用

- `conda install package-name`

本书需要下面的包：

- Python 中交互式 Python 解析器 `IPython`，科学计算基础库： `NumPy`，`SciPy`，`pandas`，和绘图库 `matplotlib` 等大多数常用库默认在 Anaconda 中安装
- 其他库如 `requests`，`beautifulsoup`，`statsmodels`，`PyTables`，`xlrd`，`xlwt` 等，它们被用在后面课程的不同示例学习中，在新的Anaconda版本中也已经被预先安装了

知乎：哪些 Python 库让你相见恨晚？ (https://www.zhihu.com/question/24590883)

本课程中一些重要的 Python 库包括以下：

- requests：人性化的 HTTP 请求库
- BeautifulSoup：以 Python 风格的方式来对 HTML 或 XML 进行解析，迭代，搜索和修改
- openpyxl：一个用来读写 Excel 2010 xlsx/xlsm/xltx/xltm 文件的库
- xlwt / xlrd：读写 Excel 文件的数据和格式信息
- python-docx：读取，查询以及修改 Microsoft Word 2007/2008 docx 文件
- XlsxWriter：一个用于创建 Excel .xlsx 文件的 Python 模块
- NLTK：一个先进的平台，用以构建处理人类语言数据的 Python 程序
- jieba：中文分词工具（没有预按照）
- pickleDB：一个简单，轻量级键值储存数据库（没有预按照）
- scikit-learn：基于 SciPy 构建的机器学习 Python 模块

绘图示例

Jupyter Notebook 的一个非常好的特性是，能够生成和插入高质量的可定制数据图，作为交互式工作流的一部分。特别是 matplotlib 包和其他科学计算工作对于 Jupyter Notebook 是可用的，功能非常强大，一旦理解了基本的流程就可以很轻松生成复杂的图表。

在启动 Jupyter Notebook 时启用绘图的方式是：

- `jupyter notebook --matplotlib=inline`

在 Python 的源代码中启动绘图的方式有两种：

- `%matplotlib inline`
- `import matplotlib.pyplot as plt`

或

- `%pylab inline`

Jupyter Notebook 中的魔术命令

magic函数主要包含两大类，一类是行魔法（Line magic）前缀为 `%` ，一类是单元魔法(Cell magic)前缀为 `%%`

最常用的几个魔法函数如下：

1. `%lsmagic`

打印当前可以用的魔法命令，可以使用%lsmagic来查询；

2. `%system shell`

如果你想使用shell，这个魔术命令可以帮到你。它非常适合快速检查当前目录和类似的东西。它看起来并不复杂，但是它是一个很好的工具。

3. `%run file.py`

在ipython中运行file.py程序

4. `%%python`

直接运行python源码

5. `%load`

加载代码（本地或网络）到当前

6. `%whos`

展示环境中的变量列表

7. `%time`，`%timeit` 和 `%% time`

代码的运行时间测试。这是对代码进行基准测试的快速方法，并向其他人表明他们需要多长时间来重新运行结果。

8. `%matplotlib`

将此魔术命令与内联参数结合使用。使用此命令可确保Jupyter Notebook显示绘图

In [1]:

```
%lsmagic
```

Out[1]:

```
Available line magics:
%alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave  %bookmark  %cd
%clear  %cls  %colors  %conda  %config  %connect_info  %copy  %ddir  %debug  %dhist
%dirs  %doctest_mode  %echo  %ed  %edit  %env  %gui  %hist  %history  %killbgscripts
%ldir  %less  %load  %load_ext  %loadpy  %logoff  %logon  %logstart  %logstate  %log
stop  %ls  %lsmagic  %macro  %magic  %matplotlib  %mkdir  %more  %notebook  %page  %
pastebin  %pdb  %pdef  %pdoc  %pfile  %pinfo  %pinfo2  %pip  %popd  %pprint  %precis
ion  %prun  %psearch  %psource  %pushd  %pwd  %pycat  %pylab  %qtconsole  %quickref
%recall  %rehashx  %reload_ext  %ren  %rep  %rerun  %reset  %reset_selective  %rmdir
%run  %save  %sc  %set_env  %store  %sx  %system  %tb  %time  %timeit  %unalias  %un
load_ext  %who  %who_ls  %whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%capture  %%cmd  %%debug  %%file  %%html  %%javascript
%%js  %%latex  %%markdown  %%perl  %%prun  %%pypy  %%python  %%python2  %%python3  %
%ruby  %%script  %%sh  %%svg  %%sx  %%system  %%time  %%timeit  %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

In [2]:

```
# 查看Python版本号
%system python -V
```

Out[2]:

```
['Python 3.7.6']
```

In [3]:

```
# 查看IPython版本号
%system ipython -V
```

Out[3]:

```
['7.12.0']
```

In [4]:

```
# 查看 jupyter notebook 版本号
%system jupyter notebook -V
```

Out[4]:

```
['6.0.3']
```

In [5]:

```
%system date
```

Out[5]:

['当前日期: 2020/02/25 周二 ', '输入新日期: (年月日)']

魔法命令执行Python文件的几种方式

1. 行魔法：%system python file.py
2. cell（单元块）魔法：%%system python file.py
3. %run 命令: %run file.py
4. %%python 直接写python源码

In [6]:

```
# 行魔法执行python命令，下面是cell魔法
%system python code/hello.py
```

Out[6]:

['Hello, Python 3!']

In [7]:

```
%%system
python code/hello.py
```

Out[7]:

['Hello, Python 3!']

In [8]:

```
%run code/hello.py
```

```
Hello, Python 3!
```

In [9]:

```
%%python
print("Hello,My Python!")
```

```
Hello,My Python!
```

加载代码到当前

执行下面代码

```
%load code/hello.py
```

之后，hello.py的内容就呈现出来，而原来的load语句自动加入注释

In [ ]:

```
# %load code/hello.py
print("Hello, Python 3!")
```

```
%load code/hello.py
```

```
# %load code/hello.py
print("Hello, Python!")
```

执行操作系统命令

In [11]:

```
%ls
```

```
 驱动器 D 中的卷是 Data
 卷的序列号是 7A4A-5B78

 D:\iPythonNB\0-专选课-数据分析实践Py3\L01-Preliminaries 的目录

2020/02/25  14:38    <DIR>          .
2020/02/25  14:38    <DIR>          ..
2020/02/25  14:14    <DIR>          .ipynb_checkpoints
2018/12/28  09:41    <DIR>          code
2020/02/25  14:18    <DIR>          data
2020/02/25  14:13    <DIR>          image
2020/02/25  14:38            28,003 L01-Preliminaries.ipynb
               1 个文件         28,003 字节
               6 个目录 47,744,782,336 可用字节
```

In [12]:

```
%whos
```

```
Interactive namespace is empty.
```

** %time，%timeit和 %% time**

代码的运行时间测试

In [13]:

```
import numpy as np
from numpy.random import randint

# A function to simulate one million dice throws.
def one_million_dice():
    return randint(low=1, high=7, size=10000000)

# Let's try %time first
%time throws = one_million_dice()
%time mean = np.mean(throws)
```

```
Wall time: 133 ms
Wall time: 12 ms
```

In [14]:

```
# Let's do the same with %timeit
%timeit throws = one_million_dice()
%timeit mean = np.mean(throws)
```

```
140 ms ± 10.2 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
13.6 ms ± 1.17 ms per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [15]:

```
%%time
throws = one_million_dice()
mean = np.mean(throws)
```

```
Wall time: 169 ms
```

3. Hello, Python

第一个 Python 代码，有多种运行方式：

In [16]:

```
# 运行方式1：  在 ipython notebook 模式下即现
print("Hello, Python 3!!")
```

```
Hello, Python 3!!
```

运行方式2

也可以将上面的代码存入到 code/hello.py 文件中，然后运行

```
%load code/hello.py
```

之后，hello.py的内容就呈现出来，而原来的load语句自动加入注释

In [17]:

```
# %load code/hello.py
print("Hello, Python 3!")
```

```
Hello, Python 3!
```

运行方式3

上面的代码存入到 code/hello.py 文件中，然后在系统命令行方式运行 `python hello.py`

In [18]:

```
# 运行方式4
%system python code/hello.py
```

Out[18]:

```
['Hello, Python 3!']
```

In [19]:

```
# 运行方式5
%run code/hello.py
```

```
Hello, Python 3!
```

运行方式6

In [20]:

```
%%python
print("Hello, Python 3!")
```

```
Hello, Python 3!
```

运行方式7

以system的单元格魔法函数方式运行 `python file.py`

In [21]:

```
%%system
python code/hello.py
```

Out[21]:

```
['Hello, Python 3!']
```

运行方式8

启动 ipython，然后在提示符后输入上述 python的print 语句

!conda install 可以检查是否安装了某个库

- 任何前置了！号的命令行都将发送给系统的 shell 来处理
- 可以使用变量来存储命令的输出结果

In [22]:

```
!conda install numpy
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.
```

注意：这个方式可能等待时间很久，因为没有安装的 package 开始在后台下载安装，慎用!

Python 帮助手册

- Tab 键的补全功能，输入函数名的前面字符，再按 Tab 键
- 在对象后面输入一个句点以便自动完成方法和属性的输入
- 调用 help 命令, help()
- 通过在函数名后面加上问号？进行查询。前提是要知道函数名，好处是不必输入 help 命令

Tab 键自动完成

这个功能是对是对标准 Python shell 的主要改进之一，大部分交互式数据分析环境都有这个功能。在 shell 中输入表达式时，只要按下 Tab 键，当前命名空间中任何与已输入的字符串相匹配的变量（对象、函数等）就会被找出来。

下面左图 IPython 将定义的两个变量都显示出来了，此外还显示了 Python 关键字 and 和内置函数 any。中图显示在任何对象后面输入一个句点以便自动完成方法和属性的输入。右图显示这个功能还可以应用在模块上。最右边的图显示，Tab 键自动完成功能不只可以用于搜索命名空间和自动完成对象或模块属性。当你输入任何看上去像是文件路径的东西时（即使是在一个 Python 字符串中），按下 Tab 键即可找出电脑文件系统中与之匹配的东西：

```
<table border="0">
<tr>
<th>
<img src="./image/tab1.png" width="120px" />
</th>
<th>
<img src="./image/tab2.png"   width="120px"/>
</th>
<th>
<img src="./image/tab3.png"   width="120px"/>
</th>
<th>
<img src="./image/tab4.png"   width="120px"/>
</th>
</tr>
</table>
```

如果再结合 %run 命令（参见后面内容），该功能将显著减少你敲键盘的次数。Tab 键自动完成功能还可用于函数关键字参数（包括等号（=）！）

注意：
> IPython 默认会隐藏那些以下划线开头的方法和属性，比如魔术方法（magic method）以及内部的“私有”方法和属性，其目的是避免在屏幕上显示一堆乱七八糟的东西（也为了避免把 Python 新人搞晕！）。其实这些也是可以通过 Tab 键自动完成的，只是你得先输入一个下划线才行。如果你就是喜欢能总是看到这些方法，直接修改 IPython 配置文件中的相关设置就可以了。

In [23]:

```
#range?
```

In [24]:

```
#help()
```

4. 几个 Python 的小示例

- 导入绘图库，画图
- 导入两个常用库，numpy 和 pandas
- 导入外部数据，显示前面几行
- 嵌入音乐、视频和网页

1. 导入绘图库，画图

In [25]:

```
# 在 python 代码中有如下两种启动绘图的方式
%matplotlib inline
import matplotlib.pyplot as plt
```

2.导入两个常用库，numpy 和 pandas

In [26]:

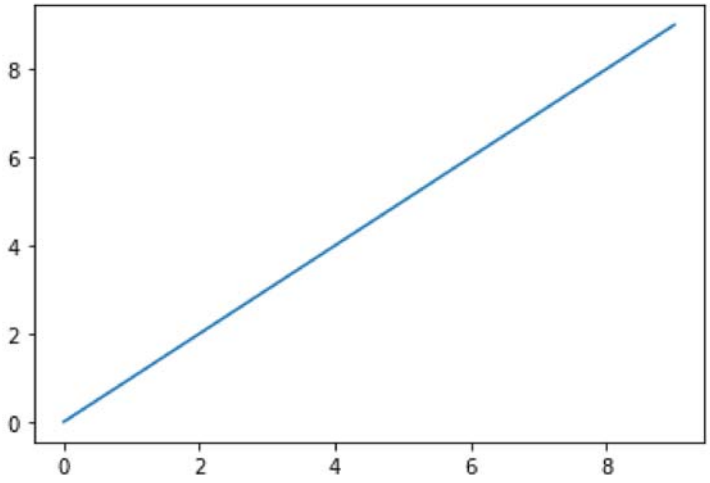
```
# 导入两个常用的库，在导入库之前先安装库
# numpy和pandas在anaconda中已经默认安装
import numpy as np
import pandas as pd

# 启动绘图
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(np.arange(10)) # 画出从0到9的曲线，使用numpy
#plt.plot(range(10))  #同上，使用python
```

Out[26]:

```
[<matplotlib.lines.Line2D at 0x23ba57b79c8>]
```



In [27]:

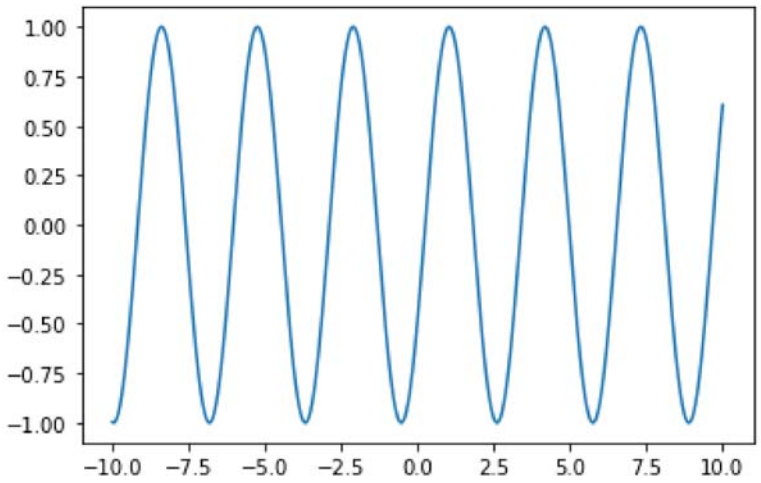
```
# 导入两个常用的库，在导入库之前先安装库
# numpy和pandas在anaconda中已经默认安装
import numpy as np
import pandas as pd

# 启动绘图
%matplotlib inline
import matplotlib.pyplot as plt

#绘制正弦曲线
x = np.linspace(-10, 10, 1000)
y = np.sin(2*x-0.5)
plt.plot(x,y)
```

Out[27]:

```
[<matplotlib.lines.Line2D at 0x23ba7347108>]
```



In [28]:

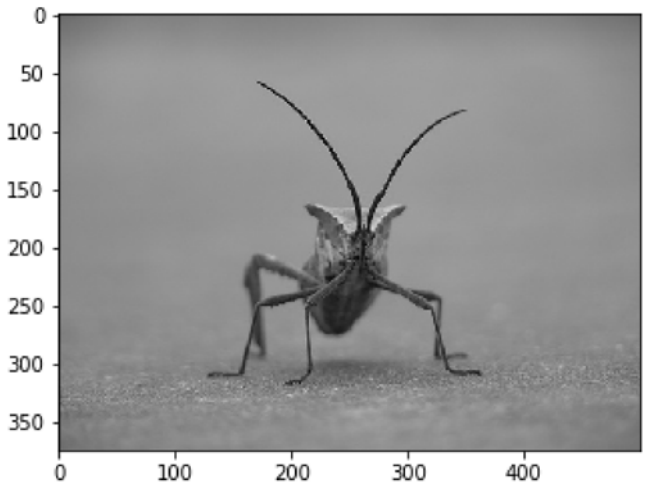
```
# 在python代码中有如下两种启动绘图的方式
# 启动绘图
%pylab inline
img = plt.imread('image/stinkbug.png')
imshow(img)
```

```
Populating the interactive namespace from numpy and matplotlib

C:\Anaconda3\lib\site-packages\IPython\core\magics\pylab.py:160: UserWarning: pylab
import has clobbered these variables: ['mean']
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

Out[28]:

```
<matplotlib.image.AxesImage at 0x23ba7617a48>
```



In [29]:

```
from IPython.display import Image
Image(filename='image/stinkbug.png')
```

Out[29]:

3.导入外部数据并显示

In [30]:

```
namelist = pd.read_csv('data/xiaoshan.csv')
namelist.head()# 最前五行，不含头行title
```

Out[30]:

	Address	Building	Organization
0	萧山区	明怡花苑	浙江旅游职业学院
1	杭州市	心意广场	浙江建设职业技术学院
2	浙江省	佳丰北苑	浙江同济科技职业学院
3	宁围镇	宁安社区	浙江师范大学萧山校区
4	北干街道	湘湖家园	戈雅公寓夏风园

In [31]:

```
namelist.tail() # 最后五行
```

Out[31]:

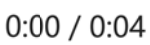
	Address	Building	Organization
27	林科技创业园	NaN	申通快递原华莺纺织
28	春晖路	NaN	众安村村委易德购超市旁右边门卫处
29	新屋门牌号	NaN	十三房村吉达来鞋厂
30	红枫路	NaN	(电话联系)
31	新感觉	NaN	伊锦苑美容美发

嵌入本地音乐和网络音乐

In [32]:

```
from IPython.display import Audio
Audio(filename='./data/si483.wav')
```

Out[32]:



In [33]:

```
from IPython.display import Audio
Audio(url='https://www.nch.com.au/acm/8k16bitpcm.wav')
```

Out[33]:

0:00 / 0:13

嵌入视频

In [34]:

```
from IPython.display import IFrame
url="http://kan.chinadaily.com.cn/content/WS5a7d4086a31019d721327d61.html"
IFrame(url, width='80%', height=400)
```

Out[34]:

移动新媒体 (//www.chinadaily.com.cn/mobile_cn/index.l

首页 (http://cn.chinadaily.com.cn/) | 时政 (http:
资讯 (http://world.chinadaily.com.cn/) | 财经 (h
生活 (http://fashion.chinadaily.com.cn/) | 图片 (
视频 (http://kan.chinadaily.com.cn/) | 专栏 (http
双语 (http://language.chinadaily.com.cn/) | 漫画

嵌入网页

In [36]:

```
from IPython.display import IFrame
url="https://www.ecnu.edu.cn/"
IFrame(url, width='80%', height=400)
```

Out[36]:

总结

- 配置并熟悉Python环境
- 掌握Python语法