

Python 数据分析

Cloudera Training Base Chongqing

重庆 Cloudera 授权大数据培训基地

2017 年 9 月 10 日

目 录

| | | |
|----------|----------------------------|----------|
| 1 | 实践练习：交互式计算和开发环境 | 1 |
| 1.1 | 交互式计算和开发环境安装 | 1 |
| 1.2 | IPython 基础 | 5 |
| 1.3 | 内省 | 8 |
| 1.4 | 使用命令历史 | 11 |
| 1.5 | jupyter notebook | 13 |
| 1.6 | NumPy 简介 | 21 |
| 1.7 | Numpy 数组对象 | 21 |
| 1.8 | 数组与标量之间的运算 | 26 |
| 1.9 | 基本的索引与切片 | 27 |
| 1.10 | 数组对象的相关操作 | 33 |

| | | |
|----------|-----------------------------------|-----------|
| 1.11 | NumPy 通用函数与方法 | 39 |
| 2 | 实践练习：数值计算类库 SciPy | 45 |
| 2.1 | Scipy 库简介 | 45 |
| 2.2 | 最小二乘拟合算法 | 46 |
| 2.3 | 统计函数算法 | 47 |
| 2.4 | 线性方程算法 | 48 |
| 2.5 | 插值算法 | 49 |
| 2.6 | 数值积分算法 | 50 |
| 2.7 | 常微分方程组算法 | 50 |
| 3 | 实践练习：高级数据结构和操作类库 Pandas 基础 | 52 |
| 3.1 | Pandas 的数据结构 | 52 |
| 3.2 | Pandas 的基本操作功能 | 58 |
| 3.3 | Pandas 的约简与汇总统计 | 64 |
| 3.4 | Pandas 缺失数据处理 | 66 |
| 3.5 | Pandas 层次化索引功能 | 67 |
| 4 | 实践练习：高级数据结构和操作类库 Pandas 进阶 | 69 |
| 4.1 | Pandas 读写文本格式的数据 | 69 |
| 4.2 | Pandas 读写二进制数据格式 | 73 |

| | | |
|----------|--------------------------------|-----------|
| 4.3 | Pandas 使用数据库 | 76 |
| 4.4 | Pandas 合并数据集 | 76 |
| 4.5 | Pandas 重塑和轴向旋转 | 82 |
| 4.6 | Pandas 数据转换 | 85 |
| 5 | 实践练习：可视化图表类库 Matplotlib | 91 |
| 5.1 | Matplotlib 类库快速绘图 | 91 |
| 5.2 | Figure 和 Subplot | 92 |
| 5.3 | Matplotlib 类库基本功能 | 95 |
| 5.4 | Pandas 绘图函数 | 102 |
| 5.5 | Matplotlib 类库绘图 | 107 |

第 1 章

实践练习：交互式计算和开发环境

1.1 交互式计算和开发环境安装

1. 交互式计算和开发环境介绍

IPython 是一个交互式计算系统。主要包含三个组件：增加的交互式 “Python shell”，解耦的双过程通信模型，交互式并行计算的架构。支持变量自动补全、自动缩进等。

IPython 官方网站：<http://www.ipython.org/>

Jupyter Notebook 是 IPython 团队开始开发一种基于 Web 技术的交互式计算文档格式 (此前被称为 IPython Notebook)，支持运行 40 多种编程语言。

Jupyter Notebook 官方网站：<https://www.jupyter.org/>

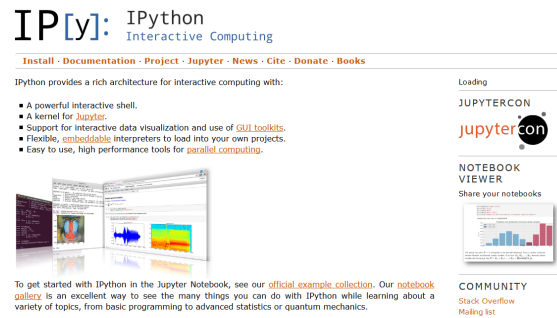


图 1.1: 'IPython 官网'

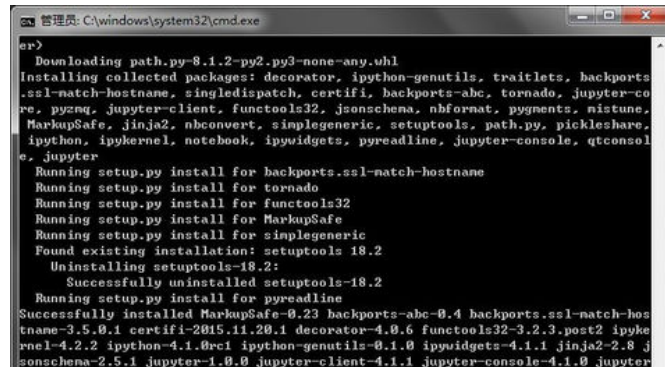


图 1.2: 'Jupyter Notebook 官网'

2. 交互式计算和开发环境安装方式

Windows 平台: 运行 cmd 命令窗口, 执行 `pip3 install Jupyter`

→ ○



```

er>
Downloading path.py-8.1.2-py2.py3-none-any.whl
Installing collected packages: decorator, ipython-genutils, traitlets, backports
.ssl-match-hostname, singledispatch, certifi, backports-abc, tornado, jupyter-co
re, pyzmq, jupyter-client, funtools32, jschema, nbformat, pygments, mistune,
MarkupSafe, Jinja2, nbconvert, simplegeneric, setuptools, path.py, pickleshare,
ipython, ipykernel, notebook, ipywidgets, pyreadline, jupyter-console, qtconsole,
e, jupyter
Running setup.py install for backports.ssl-match-hostname
Running setup.py install for tornado
Running setup.py install for funtools32
Running setup.py install for MarkupSafe
Running setup.py install for simplegeneric
Found existing installation: setuptools 18.2
Uninstalling setuptools-18.2:
Successfully uninstalled setuptools-18.2
Running setup.py install for pyreadline
Successfully installed MarkupSafe-0.23 backports-abc-0.4 backports.ssl-match-hos
tname-3.5.0.1 certifi-2015.11.20.1 decorator-4.0.6 funtools32-3.2.3.post2 ipyke
rnel-4.2.2 ipython-4.1.0rc1 ipython-genutils-0.1.0 ipywidgets-4.1.1 Jinja2-2.8 j
schema-2.5.1 jupyter-1.0.0 jupyter-client-4.1.1 jupyter-console-4.1.0 jupyter

```

图 1.3: 'Jupyter Notebook 安装过程'

3. 安装科学计算的基础库 Numpy

下载地址: <http://www.lfd.uci.edu/~gohlke/pythonlibs/#numpy>, 将文件下载至 D 盘 DataAnalysisPackage 文件夹中。

Windows 平台: 运行 cmd 命令窗口, 执行 `pip3 install numpy`

→ -1.13.1+mk1-cp36-cp36m-win_amd64.whl 本地安装。

```

1 D:\>cd DataAnalysisPackage
2 D:\DataAnalysisPackage>pip3 install numpy-1.13.1+mk1-cp36-
   → cp36m-win_amd64.whl
3 Processing d:\dataanalysispackage\numpy-1.13.1+mk1-cp36-
   → cp36m-win_amd64.whl
4 ...
5 Installing collected packages: numpy
6 Successfully installed numpy-1.13.1+mk1

```

4. 安装科学计算标准工具集合库 Scipy

下载地址:<http://www.lfd.uci.edu/~gohlke/pythonlibs/#scipy>, 将文件下载至 D 盘DataAnalysisPackage文件夹中。

Windows 平台: 运行 cmd 命令窗口, 执行 `pip3 install scipy`

↪ -0.19.1-cp36-cp36m-win_amd64.whl

```
1 D:\DataAnalysisPackage>pip3 install scipy-0.19.1-cp36-
  ↪ cp36m-win_amd64.whl
2 Processing d:\dataanalysispackage\scipy-0.19.1-cp36-cp36m-
  ↪ win_amd64.whl
3 ...
4 Installing collected packages: scipy
5 Successfully installed scipy-0.19.1
```

5. 安装高级数据结构和操作类库 Pandas

Windows 平台: 运行 cmd 命令窗口, 执行 `pip3 install pandas`

```
1 C:\Users\强>pip3 install pandas
2 Collecting pandas
3 Downloading pandas-0.20.3-cp36-cp36m-win_amd64.whl (8.3MB)
4 ...
5 Downloading pytz-2017.2-py2.py3-none-any.whl (484kB)
6 ...
7 Installing collected packages: pytz, pandas
8 Successfully installed pandas-0.20.3 pytz-2017.2
```

6. 安装可视化图表类库 Matplotlib

下载地址:<http://www.lfd.uci.edu/~gohlke/pythonlibs/#matplotlib>, 将文件下载至 D 盘DataAnalysisPackage文件夹中。

Windows 平台: 运行 cmd 命令窗口, 执行 `pip3 install matplotlib`

→ 2.0.2-cp36-cp36m-win_amd64.whl

```

1 D:\DataAnalysisPackage>pip3 install matplotlib-2.0.2-cp36-
   ↳ cp36m-win_amd64.whl
2 Processing d:\dataanalysispackage\matplotlib-2.0.2-cp36-
   ↳ cp36m-win_amd64.whl
3 Requirement already satisfied: pytz in c:\users\强
4 ....
5 Installing collected packages: cycler, matplotlib
6 Successfully installed cycler-0.10.0 matplotlib-2.0.2

```

1.2 IPython 基础

1. 如何启动 IPython

Windows 平台: 运行 cmd 命令窗口, 执行 `ipython`。

2. IPython 与 Python 进行比较

```

1 In [4]: from numpy.random import randn
2 In [5]: data = {i : randn() for i in range(3)}
3 In [6]: data
4 Out[6]:
5 {0: 1.6977986467936432,

```



```

IPython: C:\Users\强
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 保留所有权利。

C:\Users\强>ipython
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: num1 = 5

In [2]: num1
Out[2]: 5

In [3]:

```

图 1.4: ‘启动 IPython’

```

6 1: 0.22356532563683298,
7 2: -0.7346524723656888}

```

注意： 执行之前需要通过 pip3 安装 NumPy 类库。

```

1 >>> from numpy.random import randn
2 >>> data = {i : randn() for i in range(3)}
3 >>> print(data)
4 {0: 1.6977986467936432, 1: 0.22356532563683298, 2:
   ↪ -0.7346524723656888}

```

3. Tab 键自动完成功能

IPython的Tab键自动完成功能是对标准Python Shell的主要改进之一。只要按下Tab键,当前命名空间中任何与已输入的字符串想匹配的变量(对象、函数等)就会被找出来。

也可以在任何对象后面输入句号(.)以便自动完成方法和属性的输入。

还可以应用在模块上。



```
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation。保留所有权利。

C:\Users\强>ipython
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: an_num1 = 20

In [2]: an_num2 = 30

In [3]: an_
an_num1 any()
an_num2 and
```

图 1.5: ‘Tab 键自动完成方式 (1)’



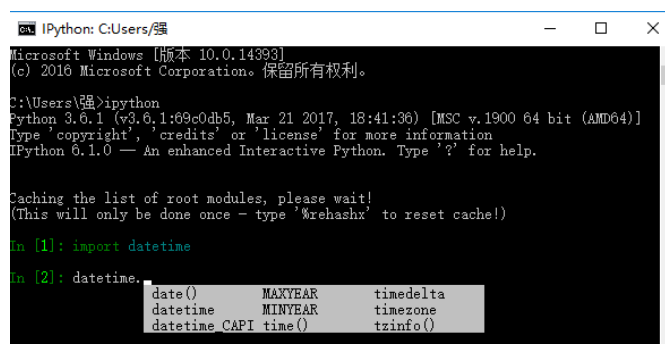
```
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation。保留所有权利。

C:\Users\强>ipython
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: list1 = [0,7,8,9]

In [2]: list1.
append() count() insert() reverse()
clear() extend() pop() sort()
copy() index() remove()
```

图 1.6: ‘Tab 键自动完成方式 (2)’



```
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation。保留所有权利。

C:\Users\强>ipython
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 18:41:36) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.

Caching the list of root modules, please wait!
(This will only be done once - type '%rehashx' to reset cache!)

In [1]: import datetime

In [2]: datetime.
date() MAXYEAR timedelta
datetime MINYEAR timezone
datetime CAPI time() tzinfo()
```

图 1.7: ‘Tab 键自动完成方式 (3)’

除了上述功能以外，还可以找出电脑文件系统中与之匹配的东西。

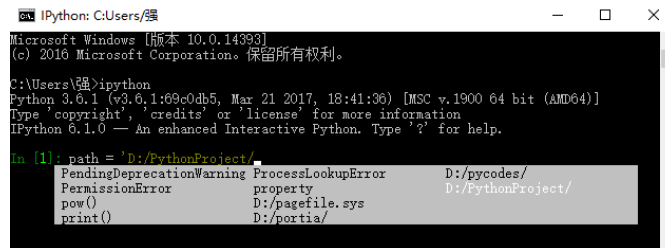


图 1.8: ‘Tab 键自动完成方式 (4)’

1.3 内省

1. 显示对象的通用信息

在变量的前面或者后面加上一个问号 (?) 就可以将有关该对象的一些通用信息进行显示。这称为对象内省。

```

1 In [1]: list1 = [6,7,8,9]
2 In [2]: list1?
3 Type:          list
4 String form: [6, 7, 8, 9]
5 Length:        4
6 Docstring:
7 list() -> new empty list
8 list(iterable) -> new list initialized from iterable's
   ↪ items

```

如果该对象是一个函数或实例方法，则其 **docstring** 也会被显示出来。

```

1 In [1]: def addition(a,b):
2     ...:     '''
3     ...:     返回执行加法的运算
4     ...:     '''
5     ...:     return a + b
6
7 In [2]: addition?
8 Signature: addition(a, b)
9 Docstring: 返回执行加法的运算
10 File:      c:\users\强\<ipython-input-2-29cdb915fb8a>
11 Type:      function

```

使用??还可以将该函数的源代码进行显示。

```

1 In [1]: def addition(a,b):
2     ...:     '''
3     ...:     返回执行加法的运算
4     ...:     '''
5     ...:     return a + b
6
7 In [2]: addition??
8 Signature: addition(a, b)
9 Source:
10 def addition(a,b):
11     '''
12     返回执行加法的运算
13     '''
14     return a + b

```

```

14 File:      c:\users\强\<ipython-input-1-29cdb915fb8a>
15 Type:      function

```

2. %run 命令

在 IPython 会话环境中，所有文件都可以通过 `%run` 命令当做 Python 程序来执行。例如：在 D 盘有 `test.py` 文件中存放了比较简单的脚本。

```

1 In [2]: %run D:\test.py
2 Hello World
3 Hello Python
4 这是一个段落
5 包含了多个语句
6 你好 您好 很好

```

3. 键盘常用快捷键

| 命令 | 说明 |
|------------------|------------------------|
| Ctrl + P 或上箭头键 | 后向搜索命令历史中以当前输入的文本开头的命令 |
| Ctrl + N 或下箭头键 | 前向搜索命令历史中以当前输入的文本开头的命令 |
| Ctrl + R | 按行读取的反向历史搜索 (部分匹配) |
| Ctrl + Shift + V | 从剪贴板粘贴文本 |
| Ctrl + C | 中止当前正在执行的代码 |
| Ctrl + A | 将光标移动到行首 |
| Ctrl + E | 将光标移动到行尾 |

| 命令 | 说明 |
|----------|---------------|
| Ctrl + K | 删除从光标开始到行尾的文本 |
| Ctrl + U | 清除当前行的所有文本 |
| Ctrl + F | 将光标句前移动一个字符 |
| Ctrl + B | 将光标句后移动一个字符 |
| Ctrl + L | 清屏 |

1.4 使用命令历史

1. 使用命令历史的作用

IPython维护着一个位于硬盘上的小型数据库，其中包含有我们执行过的每条命令的文本。这样做有几个目的。

- 减少按键次数、自动完成并执行之前已经执行过的命令。
- 在会话键持久化命令历史。
- 将输入/输出历史记录到日志文件。

2. 输入和输出变量

在实际开发过程中，如果忘记把函数结果赋值给变量是一件很郁闷的事情。而 IPython 会将输入和输出的引用保存在一些特殊变量中。

```

1 In [1]: 2 ** 6
2 Out[1]: 64
3 In [2]: 5 ** 4
4 Out[2]: 625
5 In [3]: _          # 最近的输出结果保存在_(一个下划线)
6 Out[3]: 625
7 In [4]: __         # 最近的输出结果保存在__(两个下划线)
8 Out[4]: 625

```

输入的变量保存在名为`_ix`的变量中，输出的变量保存在名为`_x`的变量中。其中`x`是输入或输出行的行号。

```

1 In [5]: username = 'freeman'
2 In [6]: username
3 Out[6]: 'freeman'
4 In [7]: _i6
5 Out[7]: 'username'
6 In [8]: _6
7 Out[8]: 'freeman'

```

3. 记录输入和输出

IPython 能够记录整个控制台的会话，包括输入和输出。执行`%logstart`即可开始记录日志。

```

1 In [9]: %logstart
2 Activating auto-logging. Current session state plus future
   ↪ input saved.
3 Filename      : ipython_log.py

```



```

4 Mode           : rotate
5 Output logging : False
6 Raw input log  : False
7 Timestamping   : False
8 State          : active

```

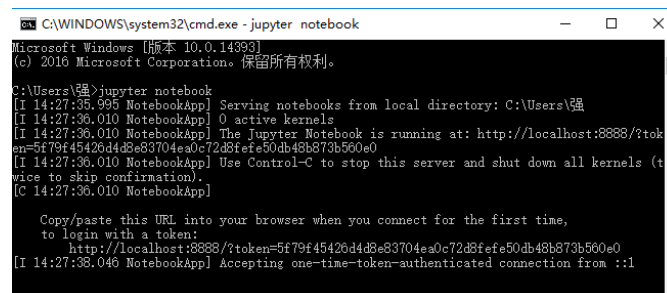
1.5 jupyter notebook

1. jupyter notebook 简介

Jupyter Notebook的本质是一个 **Web** 应用程序，便于创建和共享文学化程序文档，支持实时代码，数学方程，可视化和 markdown。用途包括：数据清理和转换，数值模拟，统计建模，机器学习等等。

启动 jupyter notebook

Windows 平台: 运行 cmd 命令窗口，执行 `jupyter notebook`。



```

C:\WINDOWS\system32\cmd.exe - jupyter notebook
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 保留所有权利。

C:\Users\强>jupyter notebook
[I 14:27:35.995 NotebookApp] Serving notebooks from local directory: C:\Users\强
[I 14:27:36.010 NotebookApp] 0 active kernels
[I 14:27:36.010 NotebookApp] The Jupyter Notebook is running at: http://localhost:8888/?token=5f79f45426d4d3e83704ea0c72d3efe50db48b873b560e0
[I 14:27:36.010 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
    http://localhost:8888/?token=5f79f45426d4d3e83704ea0c72d3efe50db48b873b560e0
[I 14:27:38.046 NotebookApp] Accepting one-time-token-authenticated connection from ::1

```

图 1.9: ‘JupyterNotebook 启动命令’

2. jupyter notebook 启动后 Web 应用程序

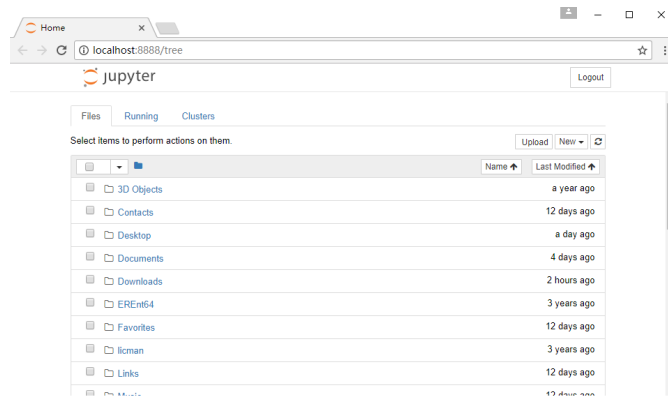


图 1.10: 'Web 应用程序'

3. 配置 Jupyter notebook 路径

Windows 平台: 运行 cmd 命令窗口, 执行 `jupyter notebook`

↪ `--generate-config`。

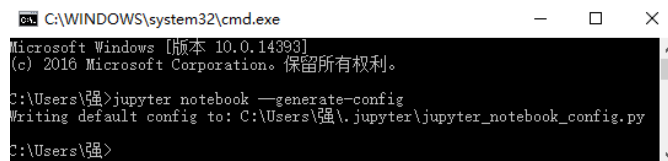


图 1.11: '执行命令生成配置文件'

4. 查看 Jupyter notebook 生成配置文件

打开“jupyter”文件夹, 可以看到里面有个 `jupyter_notebook_config`

↪ `.py` 配置文件。

5. 修改 `jupyter_notebook_config.py` 配置文件

- 打开这个配置文件, 找到 `"#c.NotebookApp.notebook_dir`

↪ `=....."`。

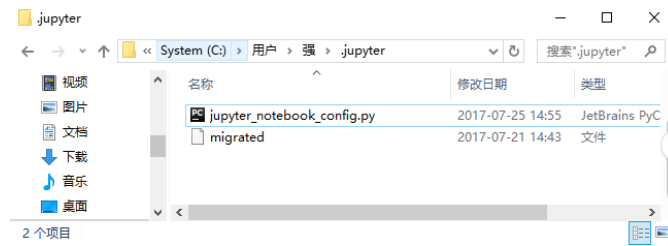


图 1.12: 'jupyter notebook 的配置文件'

- 路径改成自己的工作目录为:`c.NotebookApp.notebook_dir`
`↪ = 'D:\JupyterNoteBookCode'。`
- 重新通过 `cmd` 命令窗口启动 `jupyter notebook`。

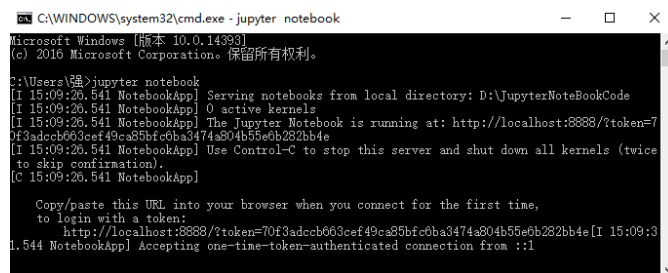


图 1.13: '修改配置文件后启动'

6. 修改配置文件后 jupyter notebook 的 Web 应用程序

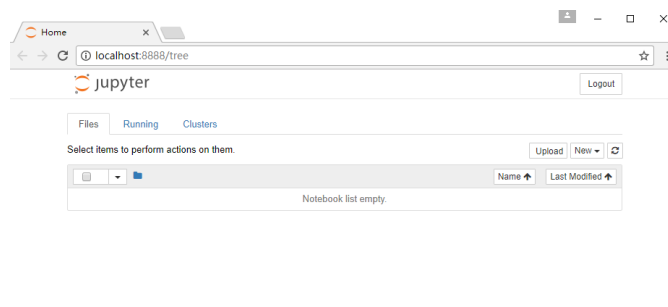


图 1.14: '修改后的 Web 应用程序'

7. jupyter notebook 如何新建文件

在主页面的右上角点new下列列表，即可新建想要的文件类型。

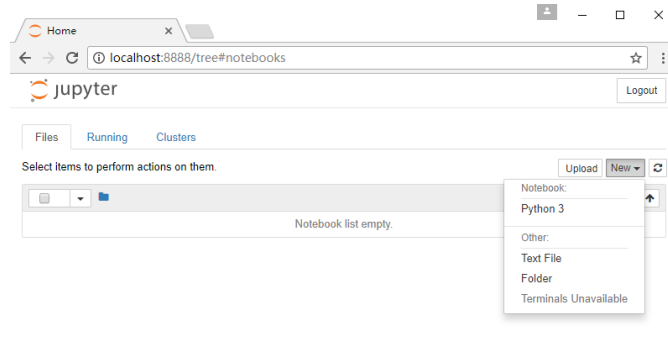


图 1.15: ‘文件类型’

8. 新建 Python3 文件

点击Python3后会在浏览器的新选项卡出现一下界面。

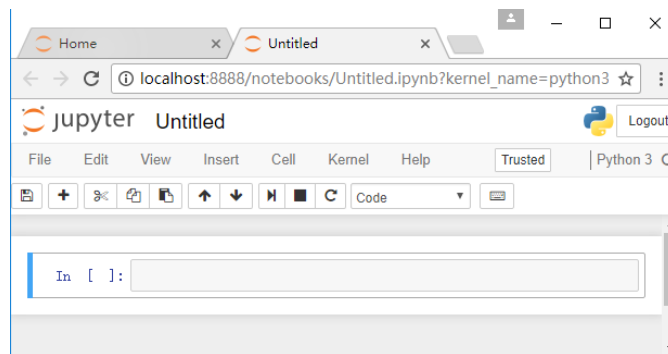


图 1.16: ‘新建 Python3 文件’

在 jupyter notebook 新建的 Python 文件后缀为 .ipynb

→ 。

9. 修改新建文件名称

方法一： 点击 **Jupyter** 图标旁的Untitled

方法二： 点击File >> rename

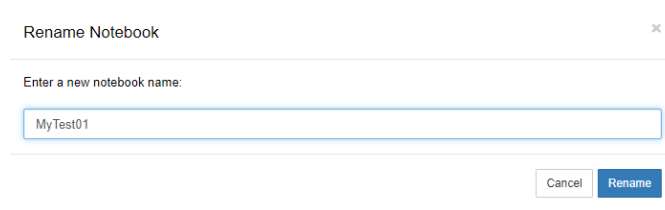


图 1.17: ‘修改名称对话框’

10. jupyter notebook 常用快捷键

- Enter：转入编辑模式。
- Ctrl + Enter：执行单元格代码。
- Shift + Enter：执行单元格代码并且移动到下一个单元格。
- Alt + Enter：执行单元格代码，新建并移动到下一个单元格。
- A：在上方插入新单元
- B：在下方插入新单元
- X：剪切选中的单元
- C：复制选中的单元
- Y：单元转入代码状态
- M：单元转入 markdown 状态

jupyter notebook 其它相关快捷键可以进行百度搜索。

11. 单元格格式

注意到快捷键栏中有一个 `code` 的下拉框，点击开发现有几个选项：

- Code格式就是正常的 python 代码格式。
- Markdown的一个 `text` 文档编辑格式，就像在 word 里编写一样。
- Heading就是给 Markdown 的句子设置标题等级。

12. 读取 CSV 文件

```
1 import pandas as pd
2 data = pd.read_csv('commodity.csv',encoding='gbk')
3 data
```

| | 产品名称 | 销售时间 | 销售数量 | 产品单价 |
|---|------|----------|------|--------|
| 0 | 手机 | 2017-3-4 | 7 | 1199.0 |
| 1 | 电脑 | 2017-3-4 | 3 | 3238.0 |
| 2 | 冰箱 | 2017-3-4 | 6 | 1599.0 |
| 3 | 洗衣机 | 2017-3-4 | 2 | 1088.0 |

图 1.18: ‘pandas 读取文件’

13. 绘制散点图

```
1 import numpy as np
2 from matplotlib import pyplot as plt
3 plt.figure(figsize=(10,7))
4 n=500
5 x=np.random.randn(1,n)
6 y=np.random.randn(1,n)
7 T=np.arctan2(x,y)
8 plt.scatter(x,y,c=T,s=50,alpha=0.5,marker='o')
9 plt.show()
```

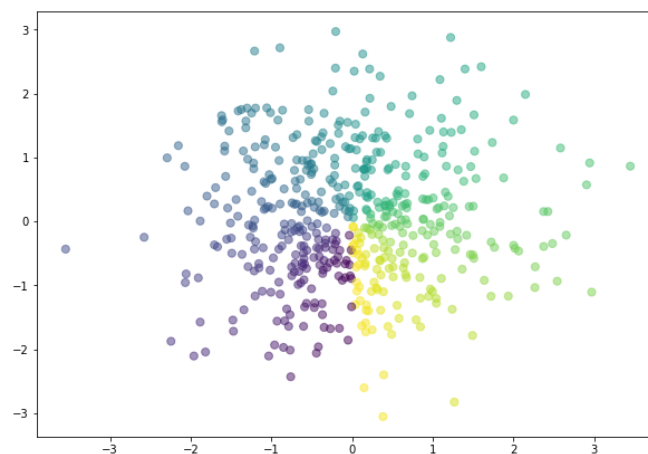


图 1.19: ‘绘制散点图’

14. jupyter notebook 编写 Markdown

```
1 # 一级标题
2 ## 二级标题
3 #### 三级标题
4 ##### 四级标题
5 ##### 五级标题
6 ##### 六级标题
```

```

7
8 - 无序列表1
9 - 无序列表2
10 - 无序列表3
11
12
13 1. 有序列表1
14 2. 有序列表2
15 3. 有序列表3
16
17 [百度](http://www.baidu.com)
18
19 ![ '大数据LOGO' ](http://www.raincent.com/uploadfile
    ↪ /2014/0728/20140728104351118.jpg)
20
21 **加粗显示**
22
23 > 提示

```



提示 # 实践练习：数组和 矢量计算类库 NumPy

1.6 NumPy 简介

1. NumPy 概述

NumPy(Numerical Python 的简称) 是高性能科学计算和数据分析的基础包。它几乎是所有高级工具的构建基础。其部分功能如下：

- `ndarray`，一个具有矢量算术运算和复杂广播能力的快速且节省空间的多维数组。
- 用于对整组数据进行快速运算的标准数学函数 (无需编写循环)。
- 用于读写磁盘数据的工具以及用于操作内存映射文件的工具。
- 线性代数、随机数生成以及傅里叶变换功能。
- 用于集成由 C，C++、Fortran 等语言编写的代码的工具。

1.7 Numpy 数组对象

1. NumPy 数组对象 `ndarray`

NumPy 最重要的一个特点就是其 N 维数组对象 `ndarray`，该对象是一个快速而灵活的大数据集容器。可以利用它对整块数据执行一些数学运算。

`ndarray`是一个多维数组对象，该对象由两部分组成：

- 实际的数据
- 描述这些数据的元数据

示例 1: Python 与 NumPy 向量相加对比

```
1 # Python的向量相加
2 def pyVector(n):
3     x = range(n)
4     y = range(n)
5     z = []
6     for i in range(len(x)):
7         x1 = i ** 2
8         y1 = i ** 3
9         z.append(x1 + y1)
10    return z
11 pyVector(5)
12
13 输出结果: [0, 2, 12, 36, 80]
```

```
1 # NumPy的向量相加
2 import numpy as np
3 def npVector(n):
4     x = np.arange(n) ** 2
5     y = np.arange(n) ** 3
6     z = x + y
7     return z
8 npVector(5)
```

```

9
10 输出结果: array([ 0,  2, 12, 36, 80], dtype=int32)

```

示例 2：创建多维数组及属性

```

1 data1 = [10,12,14,16,18]
2 arr1 = np.array(data1)
3 arr1
4
5 输出结果: array([10, 12, 14, 16, 18])

```

```

1 data2 = ((10,12,14,16,18), (11,13,15,17,19))
2 arr2 = np.array(data2)
3 arr2
4
5 输出结果: array([[10, 12, 14, 16, 18],
6              [11, 13, 15, 17, 19]])

```

```

1 arr2[0]
2
3 输出结果: array([10, 12, 14, 16, 18])

```

```

1 arr2[0][3]  # 或者 arr2[0,3]
2
3 输出结果: 16

```

```

1 arr2[0,3] = 32
2 arr2
3
4 输出结果: array([[10, 12, 14, 32, 18],

```

```
5 [11, 13, 15, 17, 19]])
```

```
1 arr1.dtype
```

```
2
```

```
3 输出结果: dtype('int32')
```

```
1 arr1.ndim,arr2.ndim
```

```
2
```

```
3 输出结果: (1, 2)
```

```
1 arr2.shape
```

```
2
```

```
3 输出结果: (2, 5)
```

2. 数组创建函数

示例

```
1 np.arange(20)
```

```
2
```

```
3 输出结果: array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
4              11, 12, 13, 14, 15, 16,17, 18, 19])
```

```
1 np.arange(1,20)
```

```
2
```

```
3 输出结果: array([ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
4              12, 13, 14, 15, 16, 17,18, 19])
```

```
1 np.arange(1,20,3)
```

```
2
```

```
3 输出结果: array([1, 4, 7, 10, 13, 16, 19])
```

3. NumPy 数据类型

`dtype`(数据类型) 是一个特殊的对象，它包含有 `ndarray` 将一块内存解释为特定数据类型所需的信息。

```
1 arr1 = np.array([10,12,14,16],dtype=np.float64)
2 arr2 = np.array([11,13,15,17],dtype=np.int32)
```

示例 3: NumPy 模拟商品数据类型

```
1 t = np.dtype([('name',np.str_,40),('numbers',np.int32),('
    ↳ price',np.float32)])
2 t
3
4 输出结果: dtype([('name', '<U40'), ('numbers', '<i4'),
5              ('price', '<f4')])
```

```
1 print(t['name'])
2
3 输出结果: <U40
```

```
1 items = np.array([('小米Mix手机',40,3099.9),
2                  ('小米Max2',20, 1799.9)],dtype=t)
3 items[1]
4
5 输出结果: ('小米Max2', 20, 1799.90002441)
```

1.8 数组与标量之间的运算

1. 数组与标量之间的运算介绍

数组很重要，因为它使你不用编写循环即可对数据执行批量运算。这通常就叫做矢量化。大小相等的数组之间的任何算术运算都会将运算应用到元素级。

```
1 import numpy as np
2 arr = np.array([[1,2,3],[4,5,6]])
3 arr
4
5 输出结果: array([[1, 2, 3],
6                [4, 5, 6]])
```

```
1 arr * arr
2
3 输出结果: array([[ 1,  4,  9],
4                [16, 25, 36]])
```

```
1 1 / arr
2
3 输出结果: array([[ 1. ,  0.5 ,  0.33333333],
4                [ 0.25 ,  0.2 ,  0.16666667]])
```

```
1 arr - arr
2
3 输出结果: array([[ 1. ,  0.5 ,  0.33333333],
4                [ 0.25 ,  0.2 ,  0.16666667]])
```

```
1 arr ** 0.5
2
3 输出结果: array([[ 1. ,  1.41421356,  1.73205081],
4                [ 2. ,  2.23606798,  2.44948974]])
```

1.9 基本的索引与切片

2. 一维数组的索引

NumPy 数组的索引是一个内容丰富的功能，它选取数据子集或单个元素的方式很多。

```
1 import numpy as np
2 arr1 = np.arange(20)
3 arr1[7:13] = 21
4 arr1
5
6 输出结果: array([ 0,  1,  2,  3,  4,  5,  6, 21, 21, 21,
7                21, 21, 21, 13, 14, 15, 16, 17, 18, 19])
```

3. 切片自动传播与复制副本

将一个标量值赋值给一个切片时，该值会自动传播到整个选区。

```
1 arr_section = arr1[7:13]
2 arr_section[2] = 678
```

```

3 arr1
4
5 输出结果: array([0, 1, 2, 3, 4, 5, 6, 21, 21, 678, 21,
6              21, 21, 13, 14, 15, 16, 17, 18, 19])

```

由于 NumPy 的设计目的是处理大数据。假如 NumPy 坚持要将数据进行反复复制将会产生何等的性能和内存问题。

复制副本

```

1 arr_section1 = arr1[7:13].copy()
2 arr_section1[:] = 22
3 arr1
4
5 输出结果: array([0, 1, 2, 3, 4, 5, 6, 30, 30, 30, 30,
6              30, 30, 13, 14, 15, 16, 17, 18, 19])

```

4. 多维数组的索引

在高维度数组中，能做更多的事情。在一个二维数组中，各索引位置上的元素不在是标量而是一维数组。

二维数组

```

1 arr2d = np.arange(1,24,2).reshape(3,4)
2 arr2d[1]
3
4 输出结果: array([9, 11, 13, 15])

```

三维数组


```

1 arr3d = np.arange(1,25).reshape(2,3,4)
2 arr3d[1,2,1]
3
4 输出结果: 22

```

5. 改变数组维度

- `ravel()` 函数

```

1 arr3d.ravel()
2
3 输出结果: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10,
4               ↪ 11, 12,
               13, 14, 15, 16, 17,18, 19, 20, 21, 22, 23,
               ↪ 24])

```

- `flatten()` 函数

```

1 arr3d.flatten()
2
3 输出结果: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10,
4               ↪ 11, 12,
               13, 14, 15, 16, 17,18, 19, 20, 21, 22, 23,
               ↪ 24])

```

- `shape()` 函数

```

1 arr3d.shape = (6,4)
2 arr3d
3
4 输出结果: array([[ 1,  2,  3,  4],
5                  [ 5,  6,  7,  8],
6                  [ 9, 10, 11, 12],
7                  [13, 14, 15, 16],
8                  [17, 18, 19, 20],
9                  [21, 22, 23, 24]])

```

- `transpose()` 函数

```

1 arr3d.transpose()
2
3 输出结果: array([[ 1,  5,  9, 13, 17, 21],
4                  [ 2,  6, 10, 14, 18, 22],
5                  [ 3,  7, 11, 15, 19, 23],
6                  [ 4,  8, 12, 16, 20, 24]])

```

- `resize()` 函数

```

1 arr3d.resize(2,12)
2 arr3d
3
4 输出结果:
5 array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12],
6        [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]])

```

6. 更多的索引

- 切片索引

ndarray 的切片与 Python 列表一维对象一样。而高维度对象的花样更多，可以在一个或多个轴上进行切片，也可以跟整数索引混合使用。

示例 4：二维数组切片索引

```
1 arr2d[:2]
2 arr2d[:2,1:]
3 arr2d[1,:2]
4 arr2d[2,:1]
5 arr2d[:,1]
6 arr2d[:,1::2]
```

| | | | |
|----|----|----|----|
| 1 | 3 | 5 | 7 |
| 9 | 11 | 13 | 15 |
| 17 | 19 | 21 | 23 |

(1)、arr2d[:2] 切片索引示意图

| | | | |
|----|----|----|----|
| 1 | 3 | 5 | 7 |
| 9 | 11 | 13 | 15 |
| 17 | 19 | 21 | 23 |

(4)、arr2d[2,:1] 切片索引示意图

| | | | |
|----|----|----|----|
| 1 | 3 | 5 | 7 |
| 9 | 11 | 13 | 15 |
| 17 | 19 | 21 | 23 |

(2)、arr2d[:2,1:] 切片索引示意图

| | | | |
|----|----|----|----|
| 1 | 3 | 5 | 7 |
| 9 | 11 | 13 | 15 |
| 17 | 19 | 21 | 23 |

(5)、arr2d[:,1] 切片索引示意图

| | | | |
|----|----|----|----|
| 1 | 3 | 5 | 7 |
| 9 | 11 | 13 | 15 |
| 17 | 19 | 21 | 23 |

(3)、arr2d[1,:2] 切片索引示意图

| | | | |
|----|----|----|----|
| 1 | 3 | 5 | 7 |
| 9 | 11 | 13 | 15 |
| 17 | 19 | 21 | 23 |

(6)、arr2d[:,1::2] 切片索引示意图

图 1.20: ‘切片索引结果示意图’

- 布尔型索引

布尔值索引指的是一个由布尔值组成的数组可以作为一个数组的索引，返回的数据为 **True** 值对应位置的值。

示例 5：布尔型索引模拟超标 PM2.5

```

1 moni_citys = np.array(['BJ', 'SH', 'BJ', 'CQ', 'SH', 'CQ', 'BJ'
    ↪ ])
2 moni_data = np.random.randn(7,8) * 2
3 more_citys = (moni_citys == 'BJ') | (moni_citys == 'SH')
4 moni_data = moni_data[more_citys]
5 moni_data[moni_data < 2.5] = 0
6 moni_data

```

- 花式索引

花式索引是指用整数数组进行索引。

示例 6：花式索引选取矩阵指定数据

```

1 arr3 = np.arange(1,61).reshape(10,6)
2 arr3[np.ix_([9,1,4,7],[2,0,5,3,1,4])]
3
4 输出结果: array([[57, 55, 60, 58, 56, 59],
5                [ 9,  7, 12, 10,  8, 11],
6                [27, 25, 30, 28, 26, 29],
7                [45, 43, 48, 46, 44, 47]])

```

1.10 数组对象的相关操作

1. 数组组合

NumPy 数组有水平组合、垂直组合和深度组合等多种组合方式，我们将使用 `vstack`、`dstack`、`hstack`、`column_stack`、`row_stack` 以及 `concatenate` 函数来完成数组的组合。

```
1 import numpy as np
2 arr1 = np.arange(1,17).reshape(4,4)
3 arr1
4
5 输出结果: array([[ 1,  2,  3,  4],
6                [ 5,  6,  7,  8],
7                [ 9, 10, 11, 12],
8                [13, 14, 15, 16]])
```

```
1 arr2 = 2 * arr1
2 arr2
3
4 输出结果: array([[ 2,  4,  6,  8],
5                [10, 12, 14, 16],
6                [18, 20, 22, 24],
7                [26, 28, 30, 32]])
```

水平组合

```
1 np.hstack((arr1,arr2)) # 或者使用np.concatenate((arr1,
    ↪ arr2), axis=1)
```

```

2
3 输出结果: array([[ 1,  2,  3,  4,  2,  4,  6,  8],
4                [ 5,  6,  7,  8, 10, 12, 14, 16],
5                [ 9, 10, 11, 12, 18, 20, 22, 24],
6                [13, 14, 15, 16, 26, 28, 30, 32]])

```

垂直组合

```

1 np.vstack((arr91,arr2)) # 或者使用np.concatenate((arr1,
   ↪ arr2), axis=0)
2
3 输出结果: array([[ 1,  2],
4                [ 2,  4],
5                [ 3,  6],
6                [ 4,  8]],
7
8                [[ 5, 10],
9                [ 6, 12],
10               [ 7, 14],
11               [ 8, 16]],
12
13               [[ 9, 18],
14                [10, 20],
15                [11, 22],
16                [12, 24]],
17
18               [[13, 26],
19                [14, 28],
20                [15, 30],

```

```
21 [16, 32]]])
```

列组合

```
1 np.column_stack((arr1,arr2))
2
3 输出结果: array([[ 1,  2,  3,  4,  2,  4,  6,  8],
4                [ 5,  6,  7,  8, 10, 12, 14, 16],
5                [ 9, 10, 11, 12, 18, 20, 22, 24],
6                [13, 14, 15, 16, 26, 28, 30, 32]])
7
8 np.column_stack((arr1,arr2)) == np.hstack((arr1,arr2))
9
10 输出结果:
11 array([[ True,  True,  True,  True,  True,  True,  True,  True,
12        ↪ True],
13        [ True,  True,  True,  True,  True,  True,  True,  True,
14        ↪ ],
15        [ True,  True,  True,  True,  True,  True,  True,  True,
16        ↪ ],
17        [ True,  True,  True,  True,  True,  True,  True,  True,
18        ↪ ]], dtype=bool)
```

行组合

```
1 np.row_stack((arr1,arr2))
2
3 输出结果: array([[ 1,  2,  3,  4],
4                [ 5,  6,  7,  8],
5                [ 9, 10, 11, 12],
```

```

6         [13, 14, 15, 16],
7         [ 2,  4,  6,  8],
8         [10, 12, 14, 16],
9         [18, 20, 22, 24],
10        [26, 28, 30, 32]])

```

2. 数组分割

NumPy 数组可以进行水平、垂直或深度分割，相关的函数有 `hsplit`、`vsplit`、`dsplit` 和 `split`。我们可以将数组分割成相同大小的子数组，也可以指定原数组中需要分割的位置。

水平分割

```

1 np.hsplit(arr1,4)  #使用np.split(arr1,4,axis=1)
2
3 输出结果: [array([[ 1],
4             [ 5],
5             [ 9],
6             [13]]), array([[ 2],
7             [ 6],
8             [10],
9             [14]]), array([[ 3],
10            [ 7],
11            [11],
12            [15]]), array([[ 4],
13            [ 8],
14            [12],
15            [16]])]

```


垂直分割

```
1 np.vsplit(arr1,4) # 使用np.split(arr1,4,axis=0)
2
3 输出结果: [array([[1, 2, 3, 4]]),
4            array([[5, 6, 7, 8]]),
5            array([[ 9, 10, 11, 12]]),
6            array([[13, 14, 15, 16]])]
```

深度分割

```
1 arr3 = np.arange(1,28).reshape(3,3,3)
2 np.dsplit(arr3,3)
3
4 输出结果: [array([[[ 1],
5                  [ 4],
6                  [ 7]],
7
8                  [[10],
9                  [13],
10                 [16]],
11
12                 [[19],
13                 [22],
14                 [25]])], array([[[ 2],
15                  [ 5],
16                  [ 8]],
17
18                  [[11],
19                  [14],
```

```
20         [17]],  
21  
22         [[20],  
23         [23],  
24         [26]]]), array([[[ 3],  
25         [ 6],  
26         [ 9]],  
27  
28         [[12],  
29         [15],  
30         [18]],  
31  
32         [[21],  
33         [24],  
34         [27]]]))]
```

3. 数组属性

除了 `shape` 和 `dtype` 属性以外，`ndarray` 对象还有很多其他的属性。

size

```
1 arr9.size  
2  
3 输出结果: 16
```

itemsize

```
1 arr9.itemsize
```

```
2
3 输出结果: 16
```

nbytes

```
1 arr9.nbytes # nbytes的值其实是itemsize 和 size 属性值的乘
   ↳ 积
2
3 输出结果: 64
4
5 arr9.size * arr9.itemsize
6
7 输出结果: 64
```

1.11 NumPy 通用函数与方法

1. 元素级数组函数

示例 7 绘制李萨茹曲线

```
1 import numpy as np
2 from matplotlib import pyplot as plt
3 a = 9
4 b = 8
5 t = np.linspace(-np.pi, np.pi, 201)
6 x = np.sin(a * t + np.pi/2)
7 y = np.sin(b * t)
8 plt.plot(x, y)
```

```
9 plt.show()
```

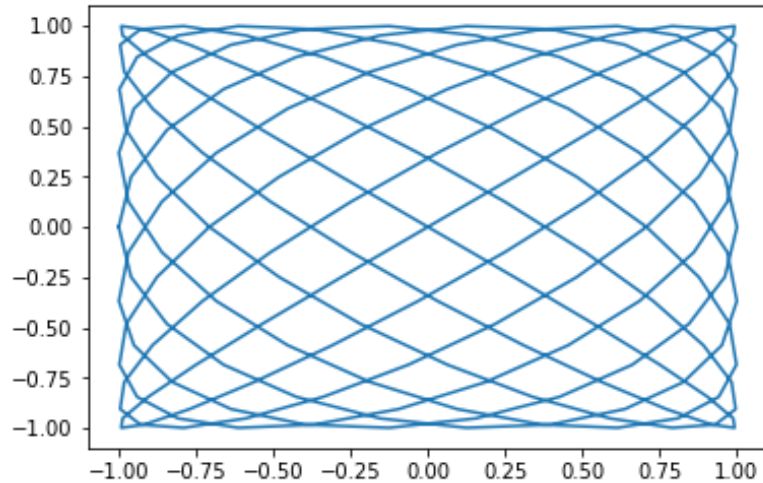


图 1.21: ‘绘制的李萨茹曲线图’

示例 8 返回两个数组元素级的最大值

```
1 xs = np.random.randn(10)
2 ys = np.random.randn(10)
3 np.maximum(xs, ys)
```

2. 运用数组进行数据处理

在 NumPy 中 `meshgrid()` 函数可以接受两个一维数组，并返回一个二维数组。

示例 9 绘制矢量化扩散图

```
1 points = np.arange(-5, 5, 0.01)
2 xs1, ys1 = np.meshgrid(points, points)
3 zs1 = np.sqrt(xs1 ** 2 + ys1 ** 2)
4 from matplotlib.font_manager import FontProperties
```

```

5 font_set = FontProperties(fname=r"c:\windows\fonts\simsum.
    ↳ ttc", size=14)
6 plt.imshow(zs1,cmap=plt.cm.Blues)
7 plt.colorbar()
8 plt.title("X与Y轴2次方矢量化扩散图",FontProperties=
    ↳ font_set)
9 plt.show()

```

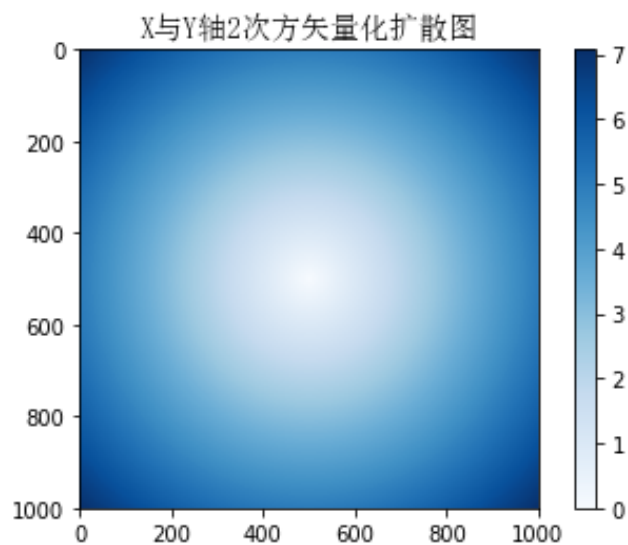


图 1.22: ‘矢量化扩散图’

3. 数组条件逻辑运算

示例 10 Where() 三元表达式选取数组的值

```

1 arrone = np.array([1.1,1.2,1.3,1.4,1.5])
2 arrtwo = np.array([2.1,2.2,2.3,2.4,2.5])
3 arrbool = ([True,False,True,True,False])
4 result = np.where(arrbool,arrone,arttwo)
5 result

```

4. 数学与统计方法

示例 11 绘制锯齿波和三角波

```
1 t = np.linspace(-np.pi, np.pi, 201)
2 k = np.arange(1, 100)
3 f = np.zeros_like(t)
4
5 for i in range(len(t)):
6     f[i] = np.sum(np.sin(2 * np.pi * k * t[i])/k)
7
8 f = (-2 / np.pi) * f
9 plt.plot(t, f, lw=1.0)
10 plt.plot(t, np.abs(f), lw=2.0)
11 plt.show()
```

5. 布尔型数组的方法

在统计方法中，布尔值会被强制转换为 1(True) 和 0(False)。
`sum()` 方法经常被用来对布尔值数组中的 True 值计数。

示例 12 统计随机数正值数量

```
1 arr5 = np.random.randn(100)
2 (arr5 > 0).sum()
```

6. 排序

跟 Python 内置的列表一样，NumPy 也可以通过 `sort()` 方法进行排序。

```

1 arr6 = np.random.randn((8))
2 arr6
3 arr6.sort()
4 arr6
5 arr7 = np.random.randn(3,10)
6 arr7
7 arr7.sort(1)
8 arr7

```

7. 唯一化与其他的集合逻辑

```

1 names = np.array(['ALBB','BD','TX','ALBB','ALBB','BD','MI'
    ↳ , 'HW','HW','TX'])
2 np.unique(names)
3 np.in1d(names, ['ALBB','BD','TX'])

```

8. 数组的文件输入输出

示例 13 读取 CSV 文件股票相关数据

```

1 c,v = np.loadtxt('data.csv',delimiter=',',usecols=(6,7),
    ↳ unpack=True)
2 c,v

```

9. 线性代数

```

1 x = np.array([[1,2,3],[4,5,6]])
2 y = np.array([[6,23],[-1,7],[8,9]])
3 x.dot(y)

```

10. 随机数生成

```
1 samples = np.random.normal(size=(4,4))  
2 samples
```


第2章

实践练习：数值计算类库 *SciPy*

2.1 Scripy 库简介

```
1 SciPy函数库在NumPy库的基础上增加了众多的数学、科学以及工程
   ↳ 计算中常用的库函数。例如线性代数、常微分方程数值求
   ↳ 解、信号处理、图像处理、稀疏矩阵等等。
2
3 | 模块 | 功能 |
4 | ---- | ---- |
5 | scipy.cluster | 矢量量化 / K-均值 |
6 | scipy.constants | 物理和数学常数 |
7 | scipy.fftpack | 傅里叶变换 |
8 | scipy.integrate | 积分程序 |
9 | scipy.interpolate | 插值 |
10 | scipy.io | 数据输入输出 |
```

```

11 | scipy.linalg | 线性代数程序 |
12 | scipy.ndimage | n维图像包 |
13 | scipy.odr | 正交距离回归 |
14 | scipy.optimize | 优化和拟合 |
15 | scipy.signal | 信号处理 |
16 | scipy.sparse | 稀疏矩阵 |
17 | scipy.spatial | 空间数据结构和算法 |
18 | scipy.special | 任何特殊数学函数 |
19 | scipy.stats | 统计 |

```

2.2 最小二乘拟合算法

```

1  **示例1 噪音曲线拟合**
2
3  ```python
4  import numpy as np
5  from scipy.optimize import leastsq
6  from matplotlib import pyplot as plt
7  from pylab import mpl
8  mpl.rcParams['font.sans-serif'] = ['SimHei']
9  mpl.rcParams['axes.unicode_minus'] = False
10
11 def func(x, p):
12     A, k, theta = p
13     return A*np.sin(2*np.pi*k*x+theta)
14
15 def residuals(p, y, x):

```

```

16     return y - func(x, p)
17
18 x = np.linspace(0, -2*np.pi, 100)
19 A, k, theta = 10, 0.34, np.pi/6
20 y0 = func(x, [A, k, theta])
21 y1 = y0 + 2 * np.random.randn(len(x))
22 p0 = [7, 0.2, 0]
23 plsq = leastsq(residuals, p0, args=(y1, x))
24 print("真实数据: ", [A, k, theta])
25 print("拟合参数: ", plsq[0])
26 plt.plot(x, y0, label="真实数据")
27 plt.plot(x, y1, label="带噪声的实验数据")
28 plt.plot(x, func(x, plsq[0]), label="拟合数据")
29 plt.legend()
30 plt.show()
31 ```

```

2.3 统计函数算法

```

1 **示例2 统计函数分析随机数**
2
3 ```python
4 from scipy import stats
5 gen= stats.norm.rvs(size=1200)
6 print("均值-标准差: ", stats.norm.fit(gen))
7 print("偏度-pvalue值: ", stats.skewtest(gen))
8 print("峰度-pvalue值: ", stats.kurtosistest(gen))

```

```

9 print("正态性检验-pvalue值: ", stats.normaltest(gen))
10 print("数据所在的区段中95%的数值: ", stats.
    ↳ scoreatpercentile(gen, 95))
11 print("数值1出发找到对应的百分比: ", stats.
    ↳ percentileofscore(gen, 1))
12 plt.hist(gen)
13 plt.show()
14 ```

```

2.4 线性方程算法

```

1 **示例3 求解线性方程组**
2
3 ```python
4 from scipy import linalg
5
6 m,n = 5,5
7 a = np.random.rand(m,n)
8 b = np.random.rand(m,n)
9
10 def linearFuncs():
11     x = linalg.solve(a,b)
12     return x
13
14 plt.title('线性方程组')
15 plt.plot(linearFuncs())
16 plt.show()

```

```
17  ` ` `
```

2.5 插值算法

```
1  **示例4 直线和B-Spline对正弦波上的点进行插值**
2
3  ```python
4  from scipy import interpolate
5
6  x = np.linspace(0, 2*np.pi+np.pi/4, 10)
7  y = np.sin(x)
8
9  x_new = np.linspace(0, 2*np.pi+np.pi/4, 100)
10 f_linear = interpolate.interpld(x, y)
11 tck = interpolate.splrep(x, y)
12 y_bspline = interpolate.splev(x_new, tck)
13
14 plt.plot(x, y, "o", label="原始数据")
15 plt.plot(x_new, f_linear(x_new), label="线性插值")
16 plt.plot(x_new, y_bspline, label="B-spline插值")
17 plt.legend()
18 plt.show()
19 ` ` `
```

2.6 数值积分算法

```

1  **示例5 累计计算积分**
2
3  ```python
4  from scipy import integrate
5  plt.figure(figsize=(8,4))
6  x = np.linspace(-2, 2, 40)
7  y=5*x**2-4*x
8  y_int = integrate.cumtrapz(y)
9  plt.plot(y_int, 'ro', label="积分值")
10 plt.plot(y, 'b*-', label="原函数")
11 plt.xlabel('横轴: 时间',fontproperties='STSong',fontsize
    ↪ =20)
12 plt.ylabel('纵轴: 位移',fontproperties='STSong',fontsize
    ↪ =20)
13 plt.legend(loc=0)
14 plt.show()
15 ```

```

2.7 常微分方程组算法

```

1  ```python
2  from scipy.integrate import odeint
3
4  def lorenz(w ,t, p, r, b):

```

```

5     x ,y, z = w
6     return np.array([p * (y -x), x * (r-z)-y, x * y - b *
7         ↪ z])
8
9 t = np.arange(0 , 40, 0.01)
10 track1 = odeint(lorenz, (0.0,1.00,0.0),t, args=(10.0,
11     ↪ 28.0,3.0))
12 track2 = odeint(lorenz, (0.0,1.01,0.0),t, args=(10.0,
13     ↪ 28.0,3.0))
14
15 from mpl_toolkits.mplot3d import Axes3D
16 fig = plt.figure()
17 ax = Axes3D(fig)
18 ax.plot(track1[:,0], track1[:,1], track1[:,2])
19 ax.plot(track2[:,0], track2[:,1], track2[:,2])
20 plt.title('洛仑兹吸引子的轨迹')
21 plt.show()
22 ```

```

第 3 章

实践练习：高级数据结构和操作类库 *Pandas* 基础

3.1 Pandas 的数据结构

1. Pandas 数据结构介绍

Pandas 引入方式

```
1 from pandas import Series, DataFrame
2 import pandas as pd
```

使用 Pandas 之前需要熟悉它的两个主要数据结构

- Series：类似一维数组的对象。
- DataFrame：一个表格型的数据结构。

2. Pandas 的 Series 对象

Series 对象相关运用

```
1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 rs = range(2,15,3)
6 ser1 = Series(rs)
7 ser1
8
9 ser1.values
10
11 ser1.index
12
13 index1 = ['c','e','a','b','d']
14 ser2 = Series(rs,index=index1)
15 ser2
16
17 ser2.index
18
19 ser1[0]
20
21 ser1[1:3]
22
23 ser2['d']
24
25 ser2[['a','c','e']]
```

```
26
27 ser2[ser2 > 7]
28
29 ser2 * 2
30
31 np.square(ser2)
32
33 'c' in ser2
34
35 'f' in ser2
36
37 data1 = {'SmartTV': 40700, 'Phone': 120000, 'Computer': 72000,
    ↪      'Freezer': 31200}
38 ser3 = Series(data1)
39 ser3
40
41 index2 = ['SmartTV', 'Phone', 'Computer', 'Washer']
42 ser4 = Series(data1, index=index2)
43 ser4
44
45 pd.isnull(ser4)
46
47 pd.notnull(ser4)
48
49 ser4.isnull()
50
51 ser3 + ser4
52
```

```

53 ser3.name = '网店销售数据'
54 ser3.index.name = '商品名称'
55 ser3
56
57 ser3.index = ['电脑','冰箱','手机','智能电视']
58 ser3

```

3. Pandas 的 DataFrame 对象

DataFrame 对象相关运用

```

1 data2 = {'商品名称':['电脑','手机','智能电视','冰箱'],
2         '销量':[72000,120000,40700,31200],
3         '单价':[3398,1599,4899,2199]}
4 df1 = DataFrame(data2)
5 df1
6
7 DataFrame(data2,columns=['商品名称','销量','单价'])
8
9 df2 = DataFrame(data2,columns=['商品名称','销量','单价','
    ↪ 库存量'],
10                  index=['a','b','c','d'])
11 df2
12
13 df2['销量']
14
15 df2.销量
16
17 df2.ix['b']

```

```

18
19 df2['库存量'] = 100
20 df2
21
22 df2['库存量'] = np.arange(7,32,8)
23 df2
24
25 val1=Series([11200,44440],index=['b','c'])
26 df2['库存量'] = val1
27 df2
28
29 df2['总量'] = df2['销量'] + df2['库存量']
30 df2
31
32 del df2['总量']
33 df2
34
35 data3= {'销量':{2014:20000,2015:30000,2016:22000},
36         '库存量'
37         ↪ :{2014:80000,2013:100000,2016:28000,2015:50000}}
38         ↪
39
40 df3 = DataFrame(data3)
41 df3
42
43 df3.T
44
45 df3.columns.name = '销存量'
46 df3.index.name = '年份'

```

```

44 df3
45
46 df3.values

```

DataFrame 构造函数所能接受的数据

- 二维 ndarray
- 由数组、列表或元组组成的字典
- NumPy 的结构化/记录数组
- 由 Series 组成的字典
- 由字典组成的字典
- 字典或 Series 的列表
- 由列表或元组组成的列表
- 另一个 DataFrame

4. 索引对象

```

1 ser4 = Series(range(1,4),index=['a','b','c'])
2 index3 = ser4.index
3 index3
4
5 index3[1:]
6
7 index3[1] = 'two'
8
9 index4 = pd.Index(np.arange(3))
10 ser5 = Series([4,5,6],index=index4)
11 ser5.index is index4

```

```
12
13 index5 = pd.Index(np.arange(10))
14 index5.delete(2)
15
16 index4.append(index5)
17
18 index5.insert(8,10)
```

3.2 Pandas 的基本操作功能

1. 重新索引

Pandas 对象重要方法 `reindex`，用于创建一个适应新索引的新对象。

Series 重新索引操作

```
1 ser1 = Series(np.arange(3,16,4),index=['b','d','a','c'])
2 ser1
3
4 index1 = ['a','b','c','d','e']
5 ser2 = ser1.reindex(index1)
6 ser2
7
8 ser1.reindex(index1,fill_value=0)
9
10 ser3 = Series(['18:20:11','19:00:17','19:40:13'],index
    ↪ = [1,3,5])
```

```

11 ser3.reindex(range(6))
12
13 ser3.reindex(range(6),method='ffill')

```

DataFrame 重新索引操作

```

... python arr2d = np.arange(1,13).reshape(3,4) df1 = DataFrame(arr2d,index=['a','c','d'],columns=['One','Two','Six','Four'])
df1

rowname = ['a','b','c','d'] df2 = df1.reindex(rowname) df2

colname = ['One','Two','Six','Four'] df1.reindex(columns=colname)

df1.reindex(index=rowname,columns=colname)

df1.ix[rowname,colname] ""

```

2. 丢弃指定轴上的项

丢弃 Series 轴上的项

```

1 ser4 = Series(np.arange(1,6),index=['a','b','c','d','e'])
2 new_ser = ser4.drop('c')
3 new_ser
4
5 ser4.drop(['a','c'])

```

丢弃 DataFrame 任意轴上的项

```

1 df1.drop(['a','d'])
2
3 df1.drop('Two',axis=1)
4

```

```
5 df1.drop(['One', 'Four'], axis=1)
```

3. 索引、选取和过滤

Series 索引的操作方式类似于 **NumPy** 数组的索引，只是索引值不只是整数。

```
1 ser4
2
3 ser4['b']
4
5 ser4[2:4]
6
7 ser4[['b', 'a']]
8
9 ser4[[1, 3]]
10
11 ser4[ser4 < 3]
12
13 ser4['b':'d']
14
15 ser4['b':'c'] = 10
16 ser4
```

DataFrame 进行索引就是获取一个或多个列。

```
1 df1
2
3 df1['Two']
4
```



```

5 df1[['Two', 'Four']]
6
7 df1[:2]
8
9 df1[df1['Three'] > 5]
10
11 df1 > 5
12
13 df1[df1 < 8] = 30
14 df1

```

DataFrame 的行上进行标签索引，可以通过 **NumPy** 的标记法以及轴标签选取行和列的子集。

```

1 df1.ix['d', ['Three', 'Two']]
2
3 df1.ix[['c', 'd'], [3, 0, 1]]
4
5 df1.ix[2]
6
7 df1.ix[df1.Three > 20, :2]

```

4. 算术运算和数据对齐

```

1 s1 = Series([10.3, -3.6, 4.3, 2.7], index=['a', 'b', 'd', 'e'])
2 s2 = Series([-3.9, 7.4, -2.2, 1.2, 5.3], index=['a', 'd', 'e', 'f'
  ↳ , 'g'])
3 s1 + s2
4

```

```

5 d1 = DataFrame(np.arange(1,10).reshape(3,3),columns=list('
    ↪ abc'))
6 d2 = DataFrame(np.arange(1,13).reshape(4,3),columns=list('
    ↪ bcd'))
7 d1 + d2

```

5. 在算术方法中充值

```

1 d3 = DataFrame(np.arange(1,10).reshape(3,3),columns=list('
    ↪ abc'))
2 d4 = DataFrame(np.arange(1,13).reshape(3,4),columns=list('
    ↪ abcd'))
3 d3.add(d4,fill_value=3)

```

6. 排序和排名

排序相关操作

```

1 sort_ser1 = Series(range(5),index=['d','a','e','c','b'])
2 sort_ser1.sort_index()
3
4 sort_df1 = DataFrame(np.arange(1,11).reshape(2,5),index=['
    ↪ Two','One'], columns=['d','a','e','c','b'])
5 sort_df1.sort_index()
6
7 sort_df1.sort_index(axis=1)
8
9 sort_df1.sort_index(axis=1,ascending=False)
10
11 sort_ser2 = Series([3,5,np.nan,-4,np.nan,1,-2])

```

```

12 sort_ser2.sort_values()
13
14 sort_df2 = DataFrame({'b':[4,7,-3,2], 'a':[0,1,0,1]})
15 sort_df2
16
17 sort_df2.sort_values(by='b')
18
19 sort_df2.sort_values(['a','b'])

```

排名相关操作

```

1 sort_ser3 = Series([7,-3,7,3,2,0,3])
2 sort_ser3.rank()
3
4 sort_ser3.rank(method='first')
5
6 sort_ser3.rank(ascending=False,method='max')
7
8 sort_df3 = DataFrame({'b':[4.3,7,-3,2],
9                       'a':[0,1,0,1],
10                      'c':[-2,5,8,-2.5]})
11 sort_df3
12
13 sort_df3.rank(axis=1)

```

7. 带有重复值的轴索引

```

1 rep_ser = Series(range(1,6),index=['b','b','a','a','c'])
2 rep_ser
3

```

```
4 rep_ser.index.is_unique
5
6 rep_ser['b']
7
8 rep_ser['c']
9
10 rep_df = DataFrame(np.random.randn(4,3),index=['a','a','b'
    ↪ , 'b'])
11 rep_df
12
13 rep_df.ix['b']
```

3.3 Pandas 的约简与汇总统计

1. 汇总和计算描述统计

简约方法

```
1 df1 = DataFrame([[2.4,np.nan],[6.3,-3.4],
2                 [np.nan,np.nan],[0.56,-0.7]],
3                 index=['a','b','c','d'],
4                 columns=['One','Two'])
5 df1
6
7 df1.sum()
8
9 df1.sum(axis=1)
10
```

```
11 df1.sum(axis=1, skipna=False)
```

描述和汇总统计方法

```
1 df1.idxmax()  
2  
3 df1.cumsum()  
4  
5 df1.describe()  
6  
7 ser1 = Series(['a', 'b', 'b', 'c']*4)  
8 ser1.describe()
```

2. 唯一值、值计数以及成员资格

```
1 ser2 = Series(['a', 'c', 'd', 'a', 'a', 'b', 'b', 'c', 'c'])  
2 ser2  
3  
4 ser2.unique()  
5  
6 ser2.value_counts()  
7  
8 mask = ser2.isin(['b', 'c'])  
9 mask  
10  
11 ser2[mask]
```

3.4 Pandas 缺失数据处理

1. 缺失数据介绍

```
1 str_data = Series(['电脑','手机',np.nan,'电视'])
2 str_data
3
4 str_data.isnull()
```

2. 滤除缺失数据

```
1 str_data.dropna()
2
3 str_data[str_data.notnull()]
4
5 df1 = DataFrame([[10,5.3,7.2],[1.5,np.nan,np.nan],
6                  [np.nan,np.nan,np.nan],[np.nan,4.7,3.5]])
7 df1
8
9 df1.dropna()
10
11 df1.dropna(how='all')
```

3. 填充缺失数据

```
1 df1.fillna(value=0)
2
3 df1.fillna(value={0:0.3,2:0.4})
```

3.5 Pandas 层次化索引功能

1. 层次化索引介绍

```
1 list1 = list('aaabbbccdd')
2 list2 = [1,2,3]*3
3 list2.insert(-2,2)
4 ser = Series(np.random.randn(10),
5               index=[list1,list2
6                       ])
7 ser
8
9 ser.index
10
11 ser['b']
12
13 ser['a':'c']
14
15 ser[:,2]
16
17 ser.unstack()
18
19 ser.unstack().stack()
20
21 df = DataFrame(np.arange(12).reshape(4,3),
22                 index=[['a','a','b','b'],[1,2,1,2]],
23                 columns=[['One','One','Two'],
24                           ['red','blue','red']])
```

```
25 df
26
27 df.index.names = ['key1','key2']
28 df.columns.names = ['state','color']
29 df
30
31 df['One']
```

2. 重排分级顺序

```
1 df.swaplevel('key1','key2')
2
3 df.sort_index(level=1)
4
5 df.swaplevel('key1','key2').sort_index(level=0)
```

3. 根据级别汇总统计

```
1 df.sum(level='key2')
2
3 df.sum(level='color',axis=1)
```


第 4 章

实践练习：高级数据结构和操作类库 *Pandas* 进阶

4.1 Pandas 读写文本格式的数据

1. 读写文本格式数据

```
1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 pd.read_csv('ex1.csv', encoding='GBK')
6
7 pd.read_table('ex1.csv', encoding='GBK', sep=',')
8
9 pd.read_csv('ex2.csv', encoding='GBK', header=None)
```

```
10
11 colnames = ['商品编号','进货价格','销售价格','销售数量','
    ↳ 商品名称']
12 pd.read_csv('ex2.csv',encoding='GBK',names=colnames)
13
14 pd.read_csv('ex2.csv',encoding='GBK',names=colnames,
    ↳ index_col='商品编号')
15
16 pd.read_csv('csv_mindex.csv',encoding='GBK',index_col=['生
    ↳ 产厂商', '产品类别'])
17
18 list(open('ex3.txt'))
19
20 pd.read_table('ex3.txt',sep='\s+')
21
22 pd.read_csv('ex4.csv',encoding='GBK')
23
24 pd.read_csv('ex4.csv',encoding='GBK',skiprows=[0,2,3])
```

2. 逐块读取文本文件

```
1 pd.read_csv('ex5.csv')
2
3 pd.read_csv('ex5.csv',nrows=20)
4
5 pd.read_csv('ex5.csv',chunksize=500)
6
7 chunker = pd.read_csv('ex5.csv',chunksize=500)
8 keyCount = Series([])
```

```

9 for i in chunker:
10     keyCount = keyCount.add(i['key'].value_counts(),
        ↪ fill_value=0)
11
12 keyCount = keyCount.sort_index(ascending=False)
13 keyCount[10:20]

```

3. 将数据写出到文本格式

```

1 data = pd.read_csv('ex6.csv',encoding='GBK')
2 data
3
4 data.to_csv('out1.csv')
5
6 data.to_csv('out2.csv',na_rep='NULL')
7
8 data.to_csv('out3.csv',na_rep='NULL',index=False,header=
    ↪ False)
9
10 data.to_csv('out4.csv',na_rep='NULL',index=False,columns=[
    ↪ '商品编号','销售数量','销售价格'])
11
12 dates = pd.date_range('8/1/2017',periods=7)
13 dates
14
15 ts = Series(np.arange(1,8),index=dates)
16 ts
17
18 ts.to_csv('tsout1.csv')

```

```

19
20 Series.from_csv('tsout1.csv')

```

4. 手工处理分隔符格式

```

1 import csv
2 f = open('ex7.csv')
3 reader = csv.reader(f)
4 for line in reader:
5     print(line)
6
7 lines = list(csv.reader(open('ex7.csv')))
8 header, values = lines[0], lines[1:]
9 data_dict = {h: v for h, v in zip(header, zip(*values))}
10 data_dict
11
12 class my_dialect(csv.Dialect):
13     lineterminator = '\n'
14     delimiter = ';'
15     quotechar = '"'
16     quoting = csv.QUOTE_MINIMAL
17
18 with open('mydata.csv', 'w') as f:
19     writer = csv.writer(f, dialect=my_dialect)
20     writer.writerow(('one', 'two', 'three'))
21     writer.writerow(('1', '2', '3'))
22     writer.writerow(('4', '5', '6'))
23     writer.writerow(('7', '8', '9'))
24 pd.read_table('mydata.csv', sep=';')

```

5. JSON 数据

```
1 obj = """
2 {"name": "Wes",
3  "places_lived": ["United States", "Spain", "Germany"],
4  "pet": null,
5  "siblings": [{"name": "Scott", "age": 25, "pet": "Zuko"},
6               {"name": "Katie", "age": 33, "pet": "Cisco"}]
7 }
8 """
9
10 import json
11 result = json.loads(obj)
12 result
13
14 asjson = json.dumps(result)
15 asjson
16
17 siblings = DataFrame(result['siblings'], columns=['name',
18           ↪ 'age'])
19 siblings
```

4.2 Pandas 读写二进制数据格式

1. pickle 序列化

```
1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 frame = pd.read_csv('ex1.csv', encoding='GBK')
6 frame
7
8 frame.to_pickle('frame_pickle')
9
10 pd.read_pickle('frame_pickle')
```

2. 读取 Microsoft Excel 文件

```
1 import xlrd, xlwt
2
3 myWork = xlwt.Workbook(encoding='utf-8')
4
5 sheet1 = myWork.add_sheet('工作表数据1')
6 sheet2 = myWork.add_sheet('工作表数据2')
7
8 data = np.arange(2, 26).reshape((6, 4))
9
10 for r in range(data.shape[0]):
11     for c in range(data.shape[1]):
12         sheet1.write(r, c, data[r, c].astype(float))
13         sheet2.write(r, c, (data[r, c]**2).astype(float))
14
15 myWork.save('D:/WorkBook.xls')
```

```
16
17
18 myBook = xlrd.open_workbook('D:/WorkBook.xls')
19
20 myBook.sheet_names()
21
22 sheet_1 = myBook.sheet_by_name('工作表数据1')
23 sheet_1.name
24
25 sheet_2 = myBook.sheet_by_index(1)
26 sheet_2.name
27
28 sheet_1.nrows, sheet_1.ncols
29
30 sheet_1.row_values(0)
31
32 sheet_1.col_values(1, start_rowx=1, end_rowx=5)
33
34 sheet_1.cell(0,2).value
35
36 xlsFile = pd.ExcelFile('D:/WorkBook.xls')
37
38 xlsFile.parse('工作表数据2')
```

4.3 Pandas 使用数据库

1. 加载 MySQL

```
1 import pandas as pd
2 import MySQLdb
3
4 conn = MySQLdb.connect(host="localhost",user="root",db="
    ↳ employees",charset="utf8")
5 sql = 'select * from departments'
6 df = pd.read_sql(sql,conn)
7 df
```

4.4 Pandas 合并数据集

1. 数据库风格的 DataFrame 合并

```
1 import numpy as np
2 from pandas import Series,DataFrame
3 import pandas as pd
4
5 df1 = DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'a', 'b']
    ↳ }, 'data1': range(7))
6 df2 = DataFrame({'key': ['a', 'b', 'd'], 'data2': range(3)
    ↳ })
7 df1
8
```



```

9 df2
10
11 pd.merge(df1, df2)
12
13 pd.merge(df1, df2, on='key')
14
15 df3 = DataFrame({'lkey': ['b', 'b', 'a', 'c', 'a', 'a', 'b',
    ↪ ''], data1': range(7)})
16 df4 = DataFrame({'rkey': ['a', 'b', 'd'], 'data2': range
    ↪ (3)})
17 pd.merge(df3, df4, left_on='lkey', right_on='rkey')
18
19 df1 = DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'b'],
20                  'data1': range(6)})
21 df2 = DataFrame({'key': ['a', 'b', 'a', 'b', 'd'],
22                  'data2': range(5)})
23 pd.merge(df1, df2, on='key', how='left')
24
25 pd.merge(df1, df2, how='inner')
26
27 left = DataFrame({'key1': ['foo', 'foo', 'bar'],
28                  'key2': ['one', 'two', 'one'],
29                  'lval': [1, 2, 3]})
30 right = DataFrame({'key1': ['foo', 'foo', 'bar', 'bar'],
31                   'key2': ['one', 'one', 'one', 'two'],
32                   'rval': [4, 5, 6, 7]})
33 pd.merge(left, right, on=['key1', 'key2'], how='outer')
34

```

```

35 pd.merge(left, right, on='key1')
36
37 pd.merge(left, right, on='key1', suffixes=('_left', '
    ↪ _right'))

```

2. 索引上的合并

```

1  import numpy as np
2  from pandas import Series, DataFrame
3  import pandas as pd
4
5  left1 = DataFrame({'key': ['a', 'b', 'a', 'a', 'b', 'c'], '
    ↪ value': range(6)})
6  right1 = DataFrame({'group_val': [3.5, 7]}, index=['a', 'b
    ↪ '])
7  left1
8
9  right1
10
11 pd.merge(left1, right1, left_on='key', right_index=True)
12
13 pd.merge(left1, right1, left_on='key', right_index=True,
    ↪ how='outer')
14
15 lefth = DataFrame({'key1': ['Ohio', 'Ohio', 'Ohio', '
    ↪ Nevada', 'Nevada'],
16                    'key2': [2000, 2001, 2002, 2001, 2002],
17                    'data': np.arange(5.)})
18 righth = DataFrame(np.arange(12).reshape((6, 2)), index=[

```

```

    ↪ 'Nevada', 'Nevada', 'Ohio', 'Ohio', 'Ohio', 'Ohio'],
    ↪ [2001, 2000, 2000, 2000, 2001, 2002]], columns=['
    ↪ event1', 'event2'])
19 lefth
20
21 righth
22
23 pd.merge(lefth, righth, left_on=['key1', 'key2'],
    ↪ right_index=True)
24
25 pd.merge(lefth, righth, left_on=['key1', 'key2'],
    ↪ right_index=True, how='outer')
26
27 left2 = DataFrame([[1., 2.], [3., 4.], [5., 6.]], index=['
    ↪ a', 'c', 'e'], columns=['Ohio', 'Nevada'])
28 right2 = DataFrame([[7., 8.], [9., 10.], [11., 12.], [13,
    ↪ 14]], index=['b', 'c', 'd', 'e'], columns=['Missouri
    ↪ ', 'Alabama'])
29 left2
30
31 right2
32
33 pd.merge(left2, right2, how='outer', left_index=True,
    ↪ right_index=True)
34
35 left2.join(right2, how='outer')
36
37 left1.join(right1, on='key')

```

```

38
39 another = DataFrame([[7., 8.], [9., 10.], [11., 12.],
    ↪ [16., 17.]], index=['a', 'c', 'e', 'f'], columns=['
    ↪ New York', 'Oregon'])
40 left2.join([right2, another])
41
42 left2.join([right2, another], how='outer')

```

3. 轴向连接

```

1 arr = np.arange(12).reshape((3, 4))
2 arr
3
4 np.concatenate([arr, arr], axis=1)
5
6
7 ser1 = Series([0, 1], index=['a', 'b'])
8 ser2 = Series([2, 3, 4], index=['c', 'd', 'e'])
9 ser3 = Series([5, 6], index=['f', 'g'])
10 pd.concat([ser1, ser2, ser3])
11
12 pd.concat([ser1, ser2, ser3], axis=1)
13
14 ser4 = pd.concat([ser1 * 5, ser3])
15 pd.concat([ser1, ser4], axis=1)
16
17 pd.concat([ser1, ser4], axis=1, join='inner')
18
19 pd.concat([ser1, ser4], axis=1, join_axes=[['a', 'c', 'b',

```

```

    ↪ 'e']]
20
21 result = pd.concat([ser1, ser1, ser3], keys=['one', 'two',
    ↪ 'three'])
22 result
23
24 pd.concat([ser1, ser2, ser3], axis=1, keys=['one', 'two',
    ↪ 'three'])
25
26 df1 = DataFrame(np.arange(6).reshape(3, 2), index=['a', 'b
    ↪ ', 'c'], columns=['one', 'two'])
27 df2 = DataFrame(5 + np.arange(4).reshape(2, 2), index=['a'
    ↪ , 'c'], columns=['three', 'four'])
28
29 pd.concat([df1, df2], axis=1, keys=['level1', 'level2'])
30
31 pd.concat({'level1': df1, 'level2': df2}, axis=1)
32
33 pd.concat([df1, df2], axis=1, keys=['level1', 'level2'],
    ↪ names=['upper', 'lower'])

```

4. 合并重叠数据

```

1 a = Series([np.nan, 2.5, np.nan, 3.5, 4.5, np.nan], index
    ↪ =['f', 'e', 'd', 'c', 'b', 'a'])
2 b = Series(np.arange(len(a), dtype=np.float64), index=['f'
    ↪ , 'e', 'd', 'c', 'b', 'a'])
3 b[-1] = np.nan
4 a

```

```

5
6 b
7
8 np.where(pd.isnull(a), b, a)
9
10 b[:-2].combine_first(a[2:])
11
12 df1 = DataFrame({'a': [1., np.nan, 5., np.nan], 'b': [np.
    ↳ nan, 2., np.nan, 6.], 'c': range(2, 18, 4)})
13 df2 = DataFrame({'a': [5., 4., np.nan, 3., 7.], 'b': [np.
    ↳ nan, 3., 4., 6., 8.]})
14 df1.combine_first(df2)

```

4.5 Pandas 重塑和轴向旋转

1. 重塑层次化索引

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 data = DataFrame(np.arange(6).reshape((2, 3)), index=pd.
    ↳ Index(['Ohio', 'Colorado'], name='state'), columns=
    ↳ pd.Index(['one', 'two', 'three'], name='number'))
6 data
7
8 result = data.stack()

```

```

9 result
10
11 result.unstack()
12
13 result.unstack(0)
14
15 result.unstack('state')
16
17 s1 = Series([0, 1, 2, 3], index=['a', 'b', 'c', 'd'])
18 s2 = Series([4, 5, 6], index=['c', 'd', 'e'])
19 data2 = pd.concat([s1, s2], keys=['one', 'two'])
20 data2.unstack()
21
22 data2.unstack().stack()
23
24 data2.unstack().stack(dropna=False)
25
26 df = DataFrame({'left': result, 'right': result + 5},
    ↪ columns=pd.Index(['left', 'right'], name='side'))
27 df
28
29 df.unstack('state')
30
31 df.unstack('state').stack('side')

```

2. 将“长格式”旋转为“宽格式”

```

1 data = pd.read_csv('macrodata.csv')
2 periods = pd.PeriodIndex(year=data.year, quarter=data.

```

```

    ↪ quarter, name='date')
3 data = DataFrame(data.to_records(),
4                   columns=pd.Index(['realgdp', 'infl', '
    ↪ unemp'], name='item'),
5                   index=periods.to_timestamp('D', 'end'))
6
7 ldata = data.stack().reset_index().rename(columns={0: '
    ↪ value'})
8 wdata = ldata.pivot('date', 'item', 'value')
9 ldata[:10]
10
11 pivoted = ldata.pivot('date', 'item', 'value')
12 pivoted.head()
13
14 ldata['value2'] = np.random.randn(len(ldata))
15 ldata[:10]
16
17 pivoted = ldata.pivot('date', 'item')
18 pivoted[:5]
19
20 pivoted['value'][:5]
21
22 unstacked = ldata.set_index(['date', 'item']).unstack('
    ↪ item')
23 unstacked[:7]

```


4.6 Pandas 数据转换

1. 移除重复数据

```
1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 data = DataFrame({'k1': ['one'] * 3 + ['two'] * 4,
6                   'k2': [1, 1, 2, 3, 3, 4, 4]})
7 data
8
9 data.duplicated()
10
11 data.drop_duplicates()
12
13 data['v1'] = range(7)
14 data.drop_duplicates(['k1'])
```

2. 用函数或映射进行数据转换

```
1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 data = DataFrame({'food': ['bacon', 'pulled pork', 'bacon'
6     → , 'Pastrami',          'corned beef', 'Bacon', 'pastrami'
7     → , 'honey ham',      'nova lox'], 'ounces': [4, 3, 12,
8     → 6, 7.5, 8, 3, 5, 6]})
```

```

6 data
7
8 meat_to_animal = {
9     'bacon': 'pig',
10    'pulled pork': 'pig',
11    'pastrami': 'cow',
12    'corned beef': 'cow',
13    'honey ham': 'pig',
14    'nova lox': 'salmon'
15 }
16 data['animal'] = data['food'].map(meat_to_animal)
17 data
18
19 data['animal'] = data['food'].map(str.lower).map(
20     ↪ meat_to_animal)
21 data
22 data['animal']=data['food'].map(lambda x: meat_to_animal[x
23     ↪ .lower()])
24 data

```

3. 替换值

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 data = Series([77., -1., 31., -1., -2., 5.])
6 data

```

```

7
8 data.replace(-1, np.nan)
9
10 data.replace([-1, -2], 0)
11
12 data.replace([-1, -2], [np.nan, 0])
13
14 data.replace({-1: np.nan, -2: 0})

```

4. 重命名轴索引

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 data = DataFrame(np.arange(12).reshape((3, 4)), index=[
6     ↳ Ohio', 'Colorado', 'New York'], columns=['one', 'two
7     ↳ ', 'three', 'four'])
8
9 data
10
11 data.index = data.index.map(str.upper)
12
13 data
14
15 data.rename(index=str.title, columns=str.upper)
16
17 data.rename(index={'OHIO': 'INDIANA'},
18 columns={'three': 'peekaboo'})

```

5. 离散化和面元划分

```
1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 ages = [20, 22, 25, 27, 21, 23, 37, 31, 61, 45, 41, 32]
6 bins = [18, 25, 35, 60, 100]
7 cats = pd.cut(ages, bins)
8 cats
9
10 pd.value_counts(cats)
11
12 group_names = ['青年', '成年', '中年', '老年']
13 pd.cut(ages, bins, labels=group_names)
14
15 data = np.random.rand(20)
16 pd.cut(data, 4, precision=2)
17
18 data = np.random.randn(500)
19 cats = pd.qcut(data, 4)
20 cats
21
22 pd.value_counts(cats)
23
24 pd.qcut(data, [0, 0.1, 0.5, 0.9, 1.]
```

6. 检测和过滤异常值

```
1 import numpy as np
```

```
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 np.random.seed(12345)
6 data = DataFrame(np.random.randn(500, 4))
7 data.describe()
8
9 col = data[2]
10 col[np.abs(col) > 3]
11
12 data[(np.abs(data) > 3).any(1)]
13
14 data[np.abs(data) > 3] = np.sign(data) * 3
15 data.describe()
```

7. 排列和随机采样

```
1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 sampler = np.random.permutation(5)
6 sampler
7
8 df = DataFrame(np.arange(20).reshape((5, 4)))
9 df
10
11 df.take(sampler)
12
```

```

13 df.take(np.random.permutation(len(df))[:3])
14
15 bag = np.array([5, 7, -1, 6, 4])
16 sampler = np.random.randint(0, len(bag), size=10)
17 sampler
18
19 draws = bag.take(sampler)
20 draws

```

8. 计算指标/哑变量

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4
5 df = DataFrame({'key': ['b', 'b', 'a', 'c', 'a', 'b'], '
    ↪ data1': range(6)})
6 pd.get_dummies(df['key'])
7
8 dummies = pd.get_dummies(df['key'], prefix='key')
9 df_with_dummy = df[['data1']].join(dummies)
10 df_with_dummy

```

第 5 章

实践练习：可视化图表类库 *Matplotlib*

5.1 Matplotlib 类库快速绘图

1. 绘制 2D 等高线图

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 fig = plt.figure(figsize=(10,7))
5 ax = fig.add_subplot(1,1,1)
6 u = np.linspace(-1, 1, 100)
7 x, y = np.meshgrid(u, u)
8 z = x ** 2 + y ** 2
9 ax.contourf(x, y, z)
10 plt.title('2D等高线图',size=22)
```

```
11 plt.show()
```

2. 绘制 3D 曲面“瓦片”图

```
1 from mpl_toolkits.mplot3d import Axes3D
2 import matplotlib.pyplot as plt
3 import numpy as np
4 from matplotlib import cm
5
6 fig = plt.figure(figsize=(10,7))
7 ax = fig.add_subplot(111,projection='3d')
8 u = np.linspace(-1, 1, 100)
9 x, y = np.meshgrid(u, u)
10 z = x ** 2 + y ** 2
11 ax.plot_surface(x, y, z, rstride=4, cstride=4, cmap=cm.
    ↪ RdYlBu_r)
12 plt.title('3D曲面“瓦片”图',size=22)
13 plt.show()
```

5.2 Figure 和 Subplot

1. Figure 和 Subplot 的作用

Figure 对象包含多个子图 (Axes)

```
1 import matplotlib.pyplot as plt
2 fig = plt.figure(figsize=(10,7))
3 ax1 = fig.add_subplot(2,2,1)
```



```

4 ax2 = fig.add_subplot(2,2,2)
5 ax3 = fig.add_subplot(2,2,3)
6 ax4 = fig.add_subplot(2,2,4)
7 plt.show()

```

示例 1：在第一个和最后一个进行绘制虚线和实线走势图

```

1 import matplotlib.pyplot as plt
2 from numpy.random import randn
3
4 fig = plt.figure(figsize=(10,7))
5 ax1 = fig.add_subplot(2,2,1)
6 plt.plot(randn(50).cumsum(), 'k--')
7 ax2 = fig.add_subplot(2,2,2)
8 ax3 = fig.add_subplot(2,2,3)
9 ax4 = fig.add_subplot(2,2,4)
10 plt.plot(randn(50).cumsum(), 'r')
11 plt.show()

```

示例 2：在第二个和第三个进行绘柱形和散点图

```

1 import matplotlib.pyplot as plt
2 from numpy.random import randn
3 import numpy as np
4
5 fig = plt.figure(figsize=(10,7))
6 ax1 = fig.add_subplot(2,2,1)
7 plt.plot(randn(50).cumsum(), 'k--')
8 ax2 = fig.add_subplot(2,2,2)
9 ax2.hist(randn(100), bins=20, color='b', alpha=0.3)

```

```

10 ax3 = fig.add_subplot(2,2,3)
11 ax3.scatter(np.arange(60), np.arange(60) + 3 * randn(60))
12 ax4 = fig.add_subplot(2,2,4)
13 plt.plot(randn(50).cumsum(), 'r')
14 plt.show()

```

pyplot.subplots

```

1 fig, axs = plt.subplots(3, 3)
2 axs

```

2. 调整多个 Subplot 之间的间距

示例 3：间距的调整

```

1 import matplotlib.pyplot as plt
2 from numpy.random import randn
3
4 rowNum = 2
5 colNum = 3
6 fig, axs = plt.subplots(rowNum, colNum, sharex=True,
7     ↳ sharey=True, figsize=(10,7))
8 for i in range(rowNum):
9     for j in range(colNum):
10         axs[i, j].hist(randn(800), bins=80, color='b',
11             ↳ alpha=0.4)
12 plt.subplots_adjust(wspace=0, hspace=0)
13 plt.show()

```

5.3 Matplotlib 类库基本功能

1. 颜色、线型和标记

示例 4：简单颜色和线型设置

```
1 import matplotlib.pyplot as plt
2 plt.figure(figsize=(10,7))
3 plt.plot(x,y,'b-')
4 plt.show()
```

示例 5：更为明确的颜色和线型设置

```
1 plt.figure(figsize=(10,7))
2 plt.plot(x, y, linestyle='--', color='#0000ff')
3 plt.show()
```

示例 6：简单标记设置

```
1 plt.figure(figsize=(10,7))
2 plt.plot(randn(50).cumsum(), 'bo--')
3 plt.show()
```

示例 7：更为明确的标记设置

```
1 plt.figure(figsize=(10,7))
2 plt.plot(randn(50).cumsum(), color='b', linestyle='--',
3         ↪ marker='o')
3 plt.show()
```

示例 8：线型强调点连接

```

1 import matplotlib.pyplot as plt
2 from numpy.random import randn
3
4 plt.figure(figsize=(10,7))
5 data = randn(50).cumsum()
6 plt.plot(data, 'r--')
7 plt.plot(data, 'b-', marker='o', drawstyle='steps-post')
8 plt.show()

```

2. 图例、刻度和轴标签

示例 9：为图表添加图例

```

1 import matplotlib.pyplot as plt
2 from numpy.random import randn
3
4 plt.figure(figsize=(10,7))
5 data = randn(50).cumsum()
6 plt.plot(data, 'r--', label='虚线走势')
7 plt.plot(data, 'b-', marker='o', drawstyle='steps-post',
8     ↪ label='实线走势')
9 plt.legend(loc='best', fontsize=15)
10 plt.show()

```

示例 10：设置图表 X 和 Y 轴的范围

```

1 plt.figure(figsize=(10,7))
2 data = randn(50).cumsum()
3 plt.plot(data, 'r--', label='虚线走势')

```

```

4 plt.plot(data, 'b-', marker='o', drawstyle='steps-post',
    ↪ label='实线')
5 plt.xlim(0,60)
6 plt.ylim(-20,20)
7 plt.legend(loc='best',fontsize=15)
8 plt.show()

```

示例 11：设置图表的刻度位置

```

1 plt.figure(figsize=(10,7))
2 data = randn(50).cumsum()
3 plt.plot(data, 'r--', label='虚线走势')
4 plt.plot(data, 'b-', marker='o', drawstyle='steps-post',
    ↪ label='实线')
5 plt.xticks(range(0,62,2))
6 plt.yticks(range(-10,11,1))
7 plt.legend(loc='best',fontsize=15)
8 plt.show()

```

示例 12：设置图表 X 和 Y 轴的轴标签

```

1 plt.figure(figsize=(10,7))
2 data = randn(50).cumsum()
3 plt.plot(data, 'r--', label='虚线走势')
4 plt.plot(data, 'b-', marker='o', drawstyle='steps-post',
    ↪ label='实线走势')
5 plt.xlabel('X轴的标签',fontsize=18)
6 plt.ylabel('Y轴的标签',fontsize=18)
7 plt.legend(loc='best',fontsize=15)
8 plt.show()

```

3. 设置图表标题和刻度、刻度标签、轴标签的其它方式

示例 13：设置图表标题及其它刻度和轴标签

```
1 import matplotlib.pyplot as plt
2 from numpy.random import randn
3
4 fig = plt.figure(figsize=(10,7))
5 ax = fig.add_subplot(1, 1, 1)
6 ax.plot(randn(1000).cumsum(),color='r',alpha=0.5)
7 ax.set_title('随机漫步图表',fontsize=20)
8 ticks = ax.set_xticks(range(0,1001,250))
9 ax.set_xlabel('X轴 标签',fontsize=16)
10 plt.show()
```

示例 14：将刻度设置为其它值

```
1 import matplotlib.pyplot as plt
2 from numpy.random import randn
3
4 fig = plt.figure(figsize=(10,7))
5 ax = fig.add_subplot(1, 1, 1)
6 ax.plot(randn(1000).cumsum(),color='r',alpha=0.5)
7 ax.set_title('随机漫步图表',fontsize=20)
8 ticks = ax.set_xticks(range(0,1001,250))
9 labels = ax.set_xticklabels(['星期一', '星期二', '星期三',
    ↳ '星期四', '星期五'], rotation=-25, fontsize='small'
    ↳ )
```

```

10 ax.set_xlabel('X轴 标签', fontsize=16)
11 plt.show()

```

4. 绘制注解

示例 15：绘制简单的注解

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from pylab import mpl
4 mpl.rcParams['font.sans-serif'] = ['SimHei']
5 mpl.rcParams['axes.unicode_minus'] = False
6
7 fig = plt.figure(figsize=(10,7))
8 ax = fig.add_subplot(111)
9 t = np.arange(0.0, 5.0, 0.01)
10 s = np.cos(2*np.pi*t)
11 ax.plot(t, s, color='b', alpha=0.5)
12
13 ax.annotate('标注文本', xy=(2, 1), xytext=(3, 1.5),
14             arrowprops=dict(color='r',), fontsize=20)
15
16 ax.set_ylim(-2,2)
17 plt.show()

```

示例 16：绘制世界金融危机重要日期注解

```

1 import matplotlib.pyplot as plt
2 from datetime import datetime
3 import pandas as pd

```

```

4 from pylab import mpl
5 mpl.rcParams['font.sans-serif'] = ['SimHei']
6 mpl.rcParams['axes.unicode_minus'] = False
7
8 fig = plt.figure(figsize=(14,10))
9 ax = fig.add_subplot(111)
10
11 data = pd.read_csv('StockData.csv', index_col=0,
    ↳ parse_dates=True, encoding='gbk')
12 stock = data['收盘价']
13
14 stock.plot(ax=ax, color='b', alpha=0.5)
15
16 key_data = [
17     (datetime(2007, 10, 11), '股市峰值'),
18     (datetime(2008, 3, 12), '贝尔斯登银行衰退'),
19     (datetime(2008, 9, 15), '雷曼兄弟破产')
20 ]
21
22 for date, label in key_data:
23     ax.annotate(label, xy=(date, stock.asof(date) + 70),
24                 xytext=(date, stock.asof(date) + 240),
25                 arrowprops=dict(color='r'), fontsize=16)
26
27 ax.set_title('2008-2009年世界金融危机的重要日期', fontsize
    ↳ =20)
28 ax.set_xlim(['5/1/2007', '1/1/2010'])
29 ax.set_ylim([600,2000])

```



```
30 plt.show()
```

5. 将图表保存为图片

示例 17：绘制世界金融危机重要日期注解

```
1 import matplotlib.pyplot as plt
2 from datetime import datetime
3 import pandas as pd
4 from pylab import mpl
5 mpl.rcParams['font.sans-serif'] = ['SimHei']
6 mpl.rcParams['axes.unicode_minus'] = False
7
8 fig = plt.figure(figsize=(14,10))
9 ax = fig.add_subplot(111)
10
11 data = pd.read_csv('StockData.csv', index_col=0,
12     ↳ parse_dates=True, encoding='gbk')
13
14 stock = data['收盘价']
15
16 stock.plot(ax=ax, color='b', alpha=0.5)
17
18 key_data = [
19     (datetime(2007, 10, 11), '股市峰值'),
20     (datetime(2008, 3, 12), '贝尔斯登银行衰退'),
21     (datetime(2008, 9, 15), '雷曼兄弟破产')
22 ]
23
24 for date, label in key_data:
```

```

23     ax.annotate(label, xy=(date, stock.asof(date) + 70),
24                  xytext=(date, stock.asof(date) + 240),
25                  arrowprops=dict(color='r'), fontsize=16)
26
27 ax.set_title('2008-2009年世界金融危机的重要日期', fontsize
    ↪ =20)
28 ax.set_xlim(['5/1/2007', '1/1/2010'])
29 ax.set_ylim([600,2000])
30 plt.savefig('D:/StockData.png', dpi=400, bbox_inches='
    ↪ tight')

```

5.4 Pandas 绘图函数

1. Series 和 DataFrame 对象的绘图方法

示例 18: Series 对象绘制简单的线形图

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import matplotlib.pyplot as plt
4
5 ser = Series(np.random.randn(20).cumsum(), index=np.arange
    ↪ (0, 300, 15))
6 ser.plot(figsize=(8,5))
7 plt.show()

```

示例 19: DataFrame 对象绘制简单的线形图

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import matplotlib.pyplot as plt
4 from pylab import mpl
5 mpl.rcParams['font.sans-serif'] = ['SimHei']
6 mpl.rcParams['axes.unicode_minus'] = False
7
8 df = DataFrame(np.random.randn(50, 8).cumsum(0),
9                columns=['小米', '华为', '阿里', '腾讯', '网易',
10                        ↪ , '百度', '搜狐', '盛大'],
11                index=np.arange(0, 1000, 20))
12 df.plot(figsize=(10,7))
13 plt.show()

```

示例 20: Series 对象绘制简单的垂直和水平柱状图

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import matplotlib.pyplot as plt
4
5 fig, ax = plt.subplots(2, 1, figsize=(16,12))
6 data = Series(np.random.rand(10), index=list('abcdefghij')
7 ↪ )
8 data.plot(kind='bar', ax=ax[0], color='r', alpha=0.5, rot
9 ↪ =360, fontsize=16)
10 data.plot(kind='barh', ax=ax[1], color='b', alpha=0.7,
11 ↪ fontsize=16)
12 plt.show()

```

示例 21: DataFrame 对象绘制简单的垂直柱状图

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4 import matplotlib.pyplot as plt
5
6 df = DataFrame(np.random.rand(7, 4),
7                 index=['周一', '周二', '周三', '周四', '周五',
8                       ↪ '周六', '周日'],
9                 columns=pd.Index(['手机', '电脑', '家电', '服
10                                ↪ 装'], name='XX 电商指数'))
11 df.plot(kind='bar', figsize=(10, 7), rot=360, fontsize=16)
12 plt.show()

```

示例 22: 餐厅聚餐数据堆积柱状图

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4 import matplotlib.pyplot as plt
5
6 restaurant = pd.read_csv('RestaurantData.csv', encoding='
7                       ↪ gbk')
8 num_counts = pd.crosstab(restaurant['周期'], restaurant['
9                       ↪ 人数'])
10 num_counts = num_counts.loc[:, 2:5]
11 num_pcts = num_counts.div(num_counts.sum(1).astype(float),
12                           ↪ axis=0)
13 num_pcts.plot(kind='bar', figsize=(10, 7), fontsize=16, rot

```

```

    ↪ =360, stacked=True)
11 plt.show()

```

示例 23：餐厅小费与消费总金额占比数据直方图

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4 import matplotlib.pyplot as plt
5
6 restaurant = pd.read_csv('RestaurantData.csv', encoding='
    ↪ gbk')
7 restaurant['百分比'] = restaurant['小费'] / restaurant['消
    ↪ 费总金额']
8 restaurant['百分比'].hist(bins=100, figsize=(10,7), color=
    ↪ 'r', alpha=0.7)
9 plt.show()

```

示例 24：餐厅小费与消费总金额占比数据密度图

```

1 import numpy as np
2 from pandas import Series, DataFrame
3 import pandas as pd
4 import matplotlib.pyplot as plt
5
6 restaurant = pd.read_csv('RestaurantData.csv', encoding='
    ↪ gbk')
7 restaurant['百分比'] = restaurant['小费'] / restaurant['消
    ↪ 费总金额']

```

```

8 restaurant['百分比'].plot(kind='kde', figsize=(10,7),
    ↪ color='r',alpha=0.7, fontsize=16)
9 plt.show()

```

示例 25：直方图和密度图组成双峰分布图

```

1 import numpy as np
2 from pandas import Series,DataFrame
3 import pandas as pd
4 import matplotlib.pyplot as plt
5
6 plt.figure(figsize=(10,7))
7 data1 = np.random.normal(0, 1, size=200)
8 data2 = np.random.normal(10, 2, size=200)
9 values = Series(np.concatenate([data1, data2]))
10 values.hist(bins=100, alpha=0.6, color='b', normed=True)
11 values.plot(kind='kde', style='r--')
12 plt.show()

```

示例 26：绘制散布图

```

1 import numpy as np
2 from pandas import Series,DataFrame
3 import pandas as pd
4 import matplotlib.pyplot as plt
5
6 md = pd.read_csv('MacroData.csv')
7 data = md[['cpi', 'm1', 'tbilrate', 'unemp']]
8 trans_data = np.log(data).diff().dropna()
9 trans_data[-5:]

```

```

10 plt.figure(figsize=(10,7))
11 plt.scatter(trans_data['m1'], trans_data['unemp'])
12 plt.show()

```

5.5 Matplotlib 类库绘图

1. 使用 Matplotlib 进行作图

示例 27：绘制误差条形图

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from pylab import mpl
4 mpl.rcParams['font.sans-serif'] = ['SimHei']
5 mpl.rcParams['axes.unicode_minus'] = False
6
7
8 x = np.arange(1, 17, 2)
9 y = x ** 0.99
10 xe = 0.1 * np.abs(np.random.randn(len(y)))
11
12 plt.figure(figsize=(10,7))
13 plt.bar(x, y, yerr=xe, width=0.4, align='center', ecolor='r
    ↪ ', color='b', label='图例标签');
14 plt.xlabel('X轴标签')
15 plt.ylabel('Y轴标签')
16 plt.title('标题')
17 plt.legend(loc='upper left')

```

```
18 plt.show()
```

示例 28：绘制饼图

```
1 import matplotlib.pyplot as plt
2 from pylab import mpl
3 mpl.rcParams['font.sans-serif'] = ['SimHei']
4 mpl.rcParams['axes.unicode_minus'] = False
5
6 plt.figure(figsize=(10, 10))
7 ax = plt.axes([0.2, 0.2, 0.7, 0.7])
8
9 labels = '小米', '华为', '中兴', '格力'
10 values = [15, 16, 16, 28]
11 explode = [0, 0, 0, 0.2]
12
13 plt.pie(values, explode=explode, labels=labels, autopct='
    ↪ %1.1f%%', startangle=67)
14
15 plt.title('商家销售占比饼图')
16 plt.show()
```

示例 29：绘制等高线图

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3 import matplotlib as mp
4 from pylab import mpl
5 mpl.rcParams['font.sans-serif'] = ['SimHei']
6 mpl.rcParams['axes.unicode_minus'] = False
```



```

7
8 def signals(x, y):
9     return (1 - (x ** 2 + y ** 2)) * np.exp(-y ** 3 / 3)
10
11 x = np.arange(-1.5, 1.5, 0.1)
12 y = np.arange(-1.5, 1.5, 0.1)
13
14 X, Y = np.meshgrid(x, y)
15
16 Z = signals(X, Y)
17
18 N = np.arange(-1, 1.5, 0.3)
19
20 CS = plt.contour(Z, N, linewidths=2, cmap=mp.cm.jet)
21 plt.clabel(CS, inline=True, fmt='%1.1f', fontsize=10)
22 plt.colorbar(CS)
23
24 plt.title('我的高线图')
25 plt.show()

```

示例 30：绘制 3D 直方图

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 from mpl_toolkits.mplot3d import Axes3D
4 from pylab import mpl
5 mpl.rcParams['font.sans-serif'] = ['SimHei']
6 mpl.rcParams['axes.unicode_minus'] = False
7 mpl.rcParams['font.size'] = 14

```

```
8
9 samples = 25
10
11 x = np.random.normal(5, 1, samples)
12 y = np.random.normal(3, .5, samples)
13
14 fig = plt.figure(figsize=(10,7))
15 ax = fig.add_subplot(111, projection='3d')
16
17 hist, xedges, yedges = np.histogram2d(x, y, bins=10)
18
19 elements = (len(xedges) - 1) * (len(yedges) - 1)
20 xpos, ypos = np.meshgrid(xedges[:-1]+.25, yedges[:-1]+.25)
21
22 xpos = xpos.flatten()
23 ypos = ypos.flatten()
24 zpos = np.zeros(elements)
25
26 ax.bar3d(xpos, ypos, zpos, dx, dy, dz, color='b', alpha
    ↪ =0.4)
27 ax.set_xlabel('X轴')
28 ax.set_ylabel('Y轴')
29 ax.set_zlabel('Z轴')
30
31 plt.show()
```