

量化投资与Python

本节内容

- ▶ 怎样用Python做量化投资
- ▶ IPython命令行介绍
- ▶ Numpy模块介绍
- ▶ pandas模块介绍
- ▶ Matplotlib模块介绍

为什么选择Python?

- ▶ 其他选择：Excel、SAS/SPSS、R
- ▶ 量化投资实际上就是分析数据从而作出决策的过程。
- ▶ Python数据处理相关模块
 - ▶ NumPy：数组批量计算
 - ▶ pandas：灵活的表计算
 - ▶ Matplotlib：数据可视化

如何使用Python进行量化投资

- ▶ 自己编写：NumPy+pandas+Matplotlib+.....
- ▶ 在线平台：聚宽、优矿、米筐、Quantopian、.....
- ▶ 开源框架：RQAlpha、QUANTAXIS、.....

IPython

交互式的Python命令行

IPython的安装与使用

- ▶ 安装: `pip install ipython`
- ▶ 使用: `ipython`
- ▶ 与Python解释器的使用方法一致

```
1. IPython: Users/macbook (Python)
macbook@bogon ~$ ipython
Python 3.6.0 (v3.6.0:41df79263a11, Dec 22 2016, 17:23:13)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import numpy as np

In [2]: import pandas as pd

In [3]: import matplotlib.pyplot as plt

In [4]: df = pd.DataFrame(np.random.randn(50, 4), columns=list('ABCD'))

In [5]: df.head()
Out[5]:
```

	A	B	C	D
0	-0.595309	-1.992211	0.400070	-1.617518
1	-1.387511	0.742664	-0.397362	0.487857
2	-0.409108	-1.232504	1.764710	-0.472531
3	0.708190	-0.194640	-1.234298	0.657188
4	-1.126034	0.788288	0.049270	1.027379

```
In [6]:
```

IPython高级功能

- ▶ TAB键自动完成
- ▶ ?: 内省、命名空间搜索
- ▶ !: 执行系统命令
- ▶ 丰富的快捷键

```
1. IPython: Users/macbook (Python)

In [7]: li = [1,2,3,4,5]

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

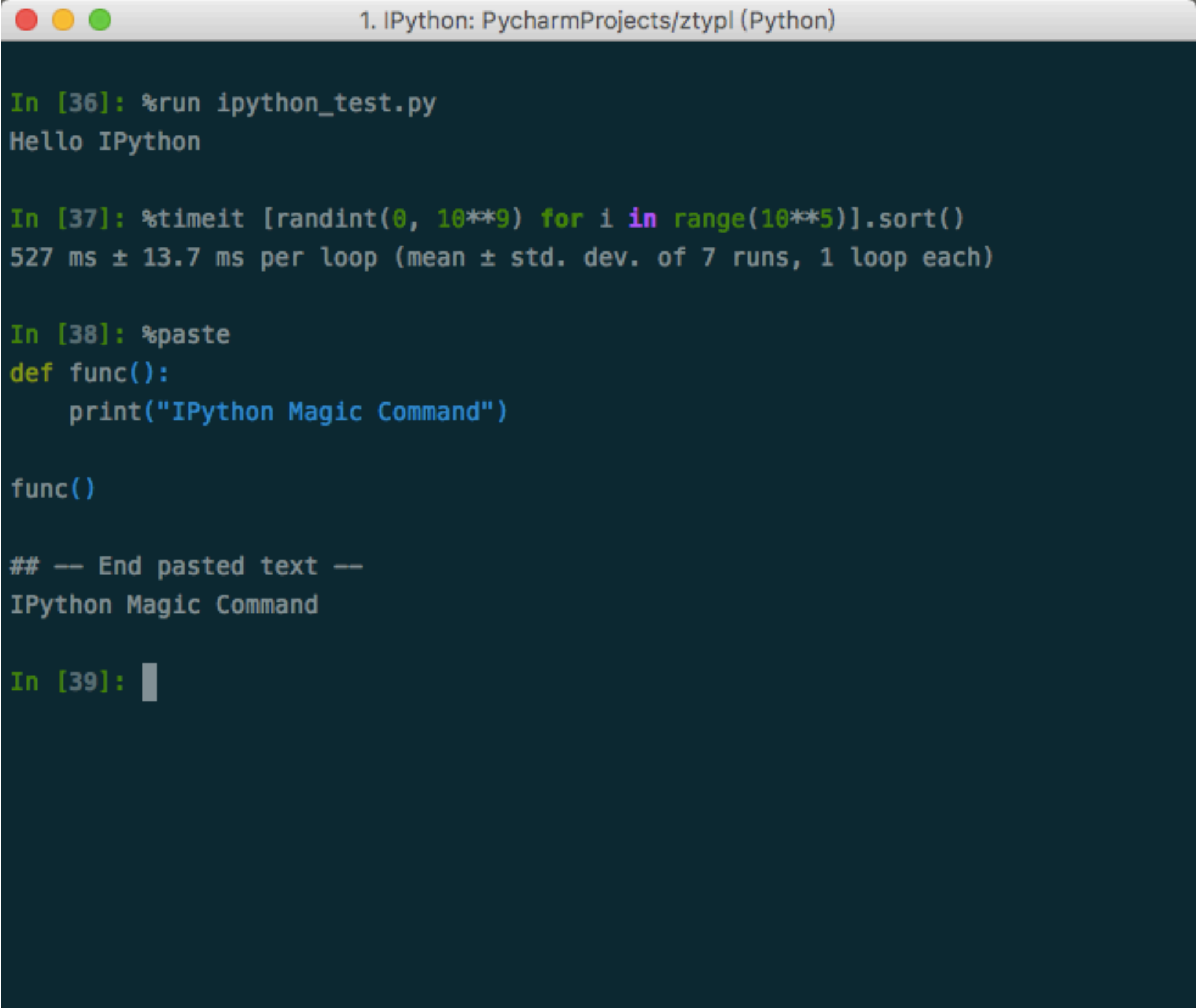
In [9]: !pip --version
pip 9.0.1 from /usr/local/lib/python2.7/site-packages (python 2.7)

In [10]: li.
append()  count()  insert()  reverse()
clear()   extend()  pop()     sort()
copy()    index()   remove()
```

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
Ctrl-K	删除从光标开始至行尾的文本
Ctrl-U	清除当前行的所有文本 ^{译注12}
Ctrl-F	将光标向前移动一个字符
Ctrl-b	将光标向后移动一个字符
Ctrl-L	清屏

IPython高级功能

- ▶ 魔术命令：以%开始的命令
 - ▶ %run：执行文件代码
 - ▶ %paste：执行剪贴板代码
 - ▶ %timeit：评估运行时间
 - ▶ %pdb：自动调试



```
1. IPython: PycharmProjects/ztypl (Python)

In [36]: %run ipython_test.py
Hello IPython

In [37]: %timeit [randint(0, 10**9) for i in range(10**5)].sort()
527 ms ± 13.7 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)

In [38]: %paste
def func():
    print("IPython Magic Command")

func()

## -- End pasted text --
IPython Magic Command

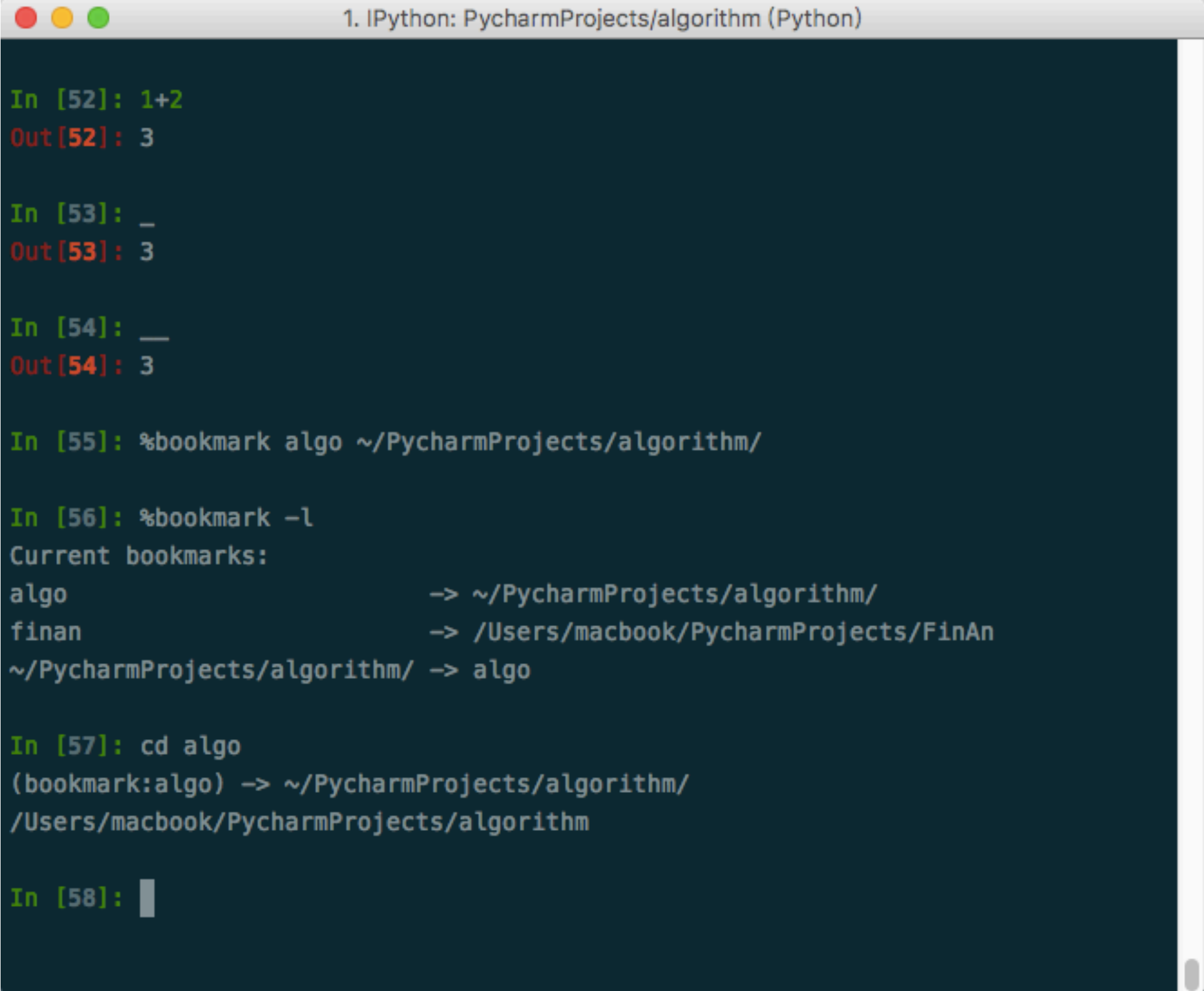
In [39]:
```

命令	说明
%quickref	显示IPython的快速参考
%magic	显示所有魔术命令的详细文档
%debug	从最新的异常跟踪的底部进入交互式调试器
%hist	打印命令的输入（可选输出）历史
%pdb	在异常发生后自动进入调试器
%paste	执行剪贴板中的Python代码
%cpaste	打开一个特殊提示符以便手工粘贴待执行的Python代码
%reset	删除interactive命名空间中的全部变量/名称
%page <i>OBJECT</i>	通过分页器打印输出OBJECT
%run <i>script.py</i>	在IPython中执行一个Python脚本文件
%prun <i>statement</i>	通过cProfile执行statement，并打印分析器的输出结果
%time <i>statement</i>	报告statement的执行时间
%timeit <i>statement</i>	多次执行statement以计算系统平均执行时间。对那些执行时间非常小的代码很有用
%who、%who_ls、%whos	显示interactive命名空间中定义的变量，信息级别/冗余度可变
%xdel <i>variable</i>	删除variable，并尝试清除其在IPython中的对象上的一切引用

命令	功能
h(elp)	显示命令列表
help <i>command</i>	显示 <i>command</i> 的文档
c(ontinue)	恢复程序的执行
q(uit)	退出调试器，不再执行任何代码
b(reak) <i>number</i>	在当前文件的第 <i>number</i> 行设置一个断点
b <i>path/to/file.py:number</i>	在指定文件的第 <i>number</i> 行设置一个断点
s(tep)	单步进入函数调用
n(ext)	执行当前行，并前进到当前级别的下一行
u(p)/d(own)	在函数调用栈中向上或向下移动
a(rgs)	显示当前函数的参数
debug <i>statement</i>	在新的（递归）调试器中调用语句 <i>statement</i>
l(ist) <i>statement</i>	显示当前行，以及当前栈级别上的上下文参考代码
w(here)	打印当前位置的完整栈跟踪（包括上下文参考代码）

IPython高级功能

- ▶ 使用命令历史
- ▶ 获取输入输出结果
- ▶ 目录标签系统
- ▶ IPython Notebook



```
1. IPython: PycharmProjects/algorithm (Python)

In [52]: 1+2
Out[52]: 3

In [53]: _
Out[53]: 3

In [54]: __
Out[54]: 3

In [55]: %bookmark algo ~/PycharmProjects/algorithm/

In [56]: %bookmark -l
Current bookmarks:
algo                -> ~/PycharmProjects/algorithm/
finan               -> /Users/macbook/PycharmProjects/FinAn
~/PycharmProjects/algorithm/ -> algo

In [57]: cd algo
(bookmark:algo) -> ~/PycharmProjects/algorithm/
/Users/macbook/PycharmProjects/algorithm

In [58]:
```

NumPy

数据分析基础包

NumPy简介

- ▶ NumPy是高性能科学计算和数据分析的基础包。它是pandas等其他各种工具的基础。
- ▶ NumPy的主要功能：
 - ▶ ndarray，一个多维数组结构，高效且节省空间
 - ▶ 无需循环对整组数据进行快速运算的数学函数
 - ▶ 线性代数、随机数生成和傅里叶变换功能
- ▶ 安装方法： `pip install numpy`
- ▶ 引用方式： `import numpy as np`

为什么要用NumPy?

- ▶ 例1：已知若干家跨国公司的市值（美元），将其换算为人民币
- ▶ 例2：已知购物车中每件商品的价格与商品件数，求总金额。

ndarray-多维数组对象

- ▶ 创建ndarray: `np.array(array_like)`
- ▶ 数组与列表的区别:
 - ▶ 数组对象内的元素类型必须相同
 - ▶ 数组大小不可修改

ndarray-常用属性

- ▶ T 数组的转置（对高维数组而言） $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$
- ▶ size 数组元素的个数
- ▶ ndim 数组的维数
- ▶ shape 数组的维度大小（元组形式）
- ▶ dtype 数组元素的数据类型

ndarray-数据类型

- ▶ 布尔型: bool_
- ▶ 整型: int_ int8 int16 int32 int64
- ▶ 无符号整型: uint8 uint16 uint32 uint64
- ▶ 浮点型: float_ float16 float32 float64
- ▶ 复数型: complex_ complex64 complex 128

ndarray-创建

- ▶ `array()` 将列表转换为数组，可选择显式指定dtype
- ▶ `arange()` `range`的numpy版，支持浮点数
- ▶ `linspace()` 类似`arange()`，第三个参数为数组长度
- ▶ `zeros()` 根据指定形状和dtype创建全0数组
- ▶ `ones()` 根据指定形状和dtype创建全1数组
- ▶ `empty()` 根据指定形状和dtype创建空数组（随机值）
- ▶ `eye()` 根据指定边长和dtype创建单位矩阵

ndarray-批量运算

- ▶ 数组和标量之间的运算

- ▶ $a+1$ $a*3$ $1//a$ $a**0.5$ $a>5$

- ▶ 同样大小数组之间的运算

- ▶ $a+b$ a/b $a**b$ $a\%b$ $a==b$

ndarray-索引

- ▶ 一维数组的索引: `a[5]`
- ▶ 多维数组的索引:
 - ▶ 列表式写法: `a[2][3]`
 - ▶ 新式写法: `a[2,3]`

ndarray-切片

- ▶ 一维数组的切片： `a[5:8]` `a[4:]` `a[2:10] = 1`
- ▶ 多维数组的切片： `a[1:2, 3:4]` `a[:,3:5]` `a[:,1]`
- ▶ 数组切片与列表切片的不同：数组切片时并不会自动复制（而是创建一个视图），在切片数组上的修改会影响原数组。
- ▶ `copy()`方法可以创建数组的深拷贝

ndarray-布尔型索引

- ▶ 问题：给一个数组，选出数组中所有大于5的数。
 - ▶ 答案： `a[a>5]`
- ▶ 原理：
 - ▶ **数组与标量的运算**： `a>5`会对a中的每一个元素进行判断，返回一个布尔数组
 - ▶ **布尔型索引**： 将同样大小的布尔数组传进索引，会返回一个由所有True对应位置的元素的数组

ndarray-布尔型索引

- ▶ 问题2：给一个数组，选出数组中所有大于5的偶数。
 - ▶ 答案： `a[(a>5) & (a%2==0)]`
- ▶ 问题3：给一个数组，选出数组中所有大于5的数和偶数。
 - ▶ 答案： `a[(a>5) | (a%2==0)]`

ndarray-花式索引

- ▶ 问题1：对于一个数组，选出其第1， 3， 4， 6， 7个元素，组成新的二维数组。
 - ▶ 答案： `a[[1,3,4,6,7]]`
- ▶ 问题2：对一个二维数组，选出其第一列和第三列，组成新的二维数组。
 - ▶ 答案： `a[:,[1,3]]`

NumPy-通用函数

- ▶ 通用函数：能同时对数组中所有元素进行运算的函数
- ▶ 常见通用函数：
 - ▶ 一元函数：**abs**, **sqrt**, exp, log, **ceil**, **floor**, rint, trunc, modf, isnan, isinf, cos, sin, tan
 - ▶ 二元函数：add, subtract, multiply, divide, power, mod, **maximum**, **mininum**,

补充-浮点数特殊值

- ▶ `nan`(Not a Number): 不等于任何浮点数 (`nan != nan`)
- ▶ `inf`(infinity): 比任何浮点数都大
- ▶ NumPy中创建特殊值: `np.nan` `np.inf`
- ▶ 在数据分析中, `nan`常被用作表示数据缺失值

NumPy-数学和统计方法

- | | | | |
|--------|------|----------|--------|
| ▶ sum | 求和 | ▶ min | 求最小值 |
| ▶ mean | 求平均数 | ▶ max | 求最大值 |
| ▶ std | 求标准差 | ▶ argmin | 求最小值索引 |
| ▶ var | 求方差 | ▶ argmax | 求最大值索引 |

NumPy-随机数生成

- ▶ 随机数函数在np.random子包内
 - ▶ rand 给定形状产生随机数组 (0到1之间的数)
 - ▶ randint 给定形状产生随机整数
 - ▶ choice 给定形状产生随机选择
 - ▶ shuffle 与random.shuffle相同
 - ▶ uniform 给定形状产生随机数组

pandas

数据分析核心工具包

pandas简介

- ▶ pandas是一个强大的Python数据分析的工具包，是基于NumPy构建的。
- ▶ pandas的主要功能
 - ▶ 具备对其功能的数据结构DataFrame、Series
 - ▶ 集成时间序列功能
 - ▶ 提供丰富的数学运算和操作
 - ▶ 灵活处理缺失数据
- ▶ 安装方法： `pip install pandas`
- ▶ 引用方法： `import pandas as pd`

Series——维数据对象

- ▶ Series是一种类似于一位数组的对象，由一组数据和一组与之相关的数据标签（索引）组成。
- ▶ 创建方式：
`pd.Series([4,7,-5,3])`
`pd.Series([4,7,-5,3],index=['a','b','c','d'])`
`pd.Series({'a':1, 'b':2})`
`pd.Series(0, index=['a','b','c','d'])`
- ▶ 获取值数组和索引数组：`values`属性和`index`属性
- ▶ Series比较像列表（数组）和字典的结合体

```
In [81]: pd.Series([4,7,-5,3],index=['a','b','c','d'])
Out[81]:
a      4
b      7
c     -5
d      3
dtype: int64
```


Series-使用特性

► Series支持array的特性（下标）：

- 从ndarray创建Series: `Series(arr)`
- 与标量运算: `sr*2`
- 两个Series运算: `sr1+sr2`
- 索引: `sr[0]`, `sr[[1,2,4]]`
- 切片: `sr[0:2]`
- 通用函数: `np.abs(sr)`
- 布尔值过滤: `sr[sr>0]`

► Series支持字典的特性（标签）：

- 从字典创建Series: `Series(dic)`,
- in运算: `'a' in sr`
- 键索引: `sr['a']`, `sr[['a', 'b', 'd']]`

Series-整数索引

- ▶ 整数索引的pandas对象往往会使新手抓狂。
- ▶ 例：
 - ▶ `sr = pd.Series(np.arange(4.))`
 - ▶ `sr[-1]`
- ▶ 如果索引是整数类型，则根据整数进行下标获取值时总是面向标签的。
- ▶ 解决方法：**loc**属性（将索引解释为标签）和**iloc**属性（将索引解释为下标）

Series-数据对齐

► 例：

► `sr1 = pd.Series([12,23,34], index=['c','a','d'])`

► `sr2 = pd.Series([11,20,10], index=['d','c','a'])`

► `sr1+sr2`

► pandas在进行两个Series对象的运算时，会按索引进行对齐然后计算。

Series-数据对齐

► 例：

► `sr1 = pd.Series([12,23,34], index=['c','a','d'])`

► `sr2 = pd.Series([11,20,10], index=['b','c','a'])`

► `sr1+sr2`

► 如果两个Series对象的索引不完全相同，则结果的索引是两个操作数索引的并集。
如果只有一个对象在某索引下有值，则结果中该索引的值为nan（缺失值）。

Series-数据对齐

► 例：

► `sr1 = pd.Series([12,23,34], index=['c','a','d'])`

► `sr2 = pd.Series([11,20,10], index=['b','c','a'])`

► 如何使结果在索引'b'处的值为11，在索引'd'处的值为34？

► 灵活的算术方法： `add`, `sub`, `div`, `mul`

► `sr1.add(sr2, fill_value=0)`

Series-缺失数据

- ▶ 缺失数据：使用NaN（Not a Number）来表示缺失数据。其值等于np.nan。内置的None值也会被当做NaN处理。
- ▶ 处理缺失数据的相关方法：
 - ▶ dropna() 过滤掉值为NaN的行
 - ▶ fillna() 填充缺失数据
 - ▶ isnull() 返回布尔数组，缺失值对应为True
 - ▶ notnull() 返回布尔数组，缺失值对应为False
- ▶ 过滤缺失数据：sr.dropna() 或 sr[~sr.isnull()]
- ▶ 填充缺失数据：fillna(0)

DataFrame-二维数据对象

- ▶ DataFrame是一个表格型的数据结构，含有一组有序的列。DataFrame可以被看做是由Series组成的字典，并且共用一个索引。
- ▶ 创建方式：
 - ▶ `pd.DataFrame({'one':[1,2,3,4],'two':[4,3,2,1]})`
 - ▶ `pd.DataFrame({'one':pd.Series([1,2,3],index=['a','b','c']), 'two':pd.Series([1,2,3,4],index=['b','a','c','d'])})`
 - ▶
- ▶ csv文件读取与写入：
 - ▶ `df.read_csv('filename.csv')`
 - ▶ `df.to_csv()`

```
In [107]: pd.DataFrame({'one':pd.Series([1,2,3],index=['a','b','c']), 'two':pd.S
...:   eries([1,2,3,4],index=['b','a','c','d'])})
Out[107]:
```

	one	two
a	1.0	2
b	2.0	1
c	3.0	3
d	NaN	4

DataFrame-常用属性

- ▶ index 获取索引
- ▶ T 转置
- ▶ columns 获取列索引
- ▶ values 获取值数组
- ▶ describe() 获取快速统计

DataFrame-索引和切片

- ▶ DataFrame是一个二维数据类型，所以有行索引和列索引。
- ▶ DataFrame同样可以通过标签和位置两种方法进行索引和切片
- ▶ loc属性和iloc属性
 - ▶ 使用方法：逗号隔开，前面是行索引，后面是列索引
 - ▶ 行/列索引部分可以是常规索引、切片、布尔值索引、花式索引任意搭配

DataFrame-数据对齐与缺失数据

- ▶ DataFrame对象在运算时，同样会进行数据对齐，其行索引和列索引分别对齐。
- ▶ DataFrame处理缺失数据的相关方法：
 - ▶ `dropna(axis=0,where='any',...)`
 - ▶ `fillna()`
 - ▶ `isnull()`
 - ▶ `notnull()`

pandas-其他常用方法

- ▶ `mean(axis=0, skipna=False)` 对列（行）求平均值
- ▶ `sum(axis=1)` 对列（行）求和
- ▶ `sort_index(axis, ..., ascending)` 对列（行）索引排序
- ▶ `sort_values(by, axis, ascending)` 按某一系列（行）的值排序
- ▶ NumPy的通用函数同样适用于pandas

pandas-其他常用方法

- ▶ `apply(func, axis=0)` 将自定义函数应用在各行或者各列上, func可返回标量或者Series
- ▶ `applymap(func)` 将函数应用在DataFrame各个元素上
- ▶ `map(func)` 将函数应用在Series各个元素上

pandas-时间对象处理

- ▶ 时间序列类型：
 - ▶ 时间戳：特定时刻
 - ▶ 固定时期：如2017年7月
 - ▶ 时间间隔：起始时间-结束时间
- ▶ Python标准库处理时间对象：datetime
 - ▶ 灵活处理时间对象：dateutil
 - ▶ `dateutil.parser.parse()`
 - ▶ 成组处理时间对象：pandas
 - ▶ `pd.to_datetime()`

pandas-时间对象处理

- ▶ 产生时间对象数组: `date_range`
 - ▶ `start` 开始时间
 - ▶ `end` 结束时间
 - ▶ `periods` 时间长度
 - ▶ `freq` 时间频率, 默认为'D', 可选H(our),W(eek),B(usiness),S(emi-)M(onth), (min)T(es), S(econd), A(year),...

pandas-时间序列

- ▶ 时间序列就是以时间对象为索引的Series或DataFrame。
- ▶ datetime对象作为索引时是存储在DatetimeIndex对象中的。
- ▶ 时间序列特殊功能：
 - ▶ 传入“年”或“年月”作为切片方式
 - ▶ 传入日期范围作为切片方式
 - ▶ 丰富的函数支持：resample(), strftime(),

pandas-文件处理

- ▶ 数据文件常用格式：csv（以某间隔符分割数据）
- ▶ pandas读取文件：从文件名、URL、文件对象中加载数据
 - ▶ read_csv 默认分隔符为逗号
 - ▶ read_table 默认分隔符为制表符

pandas-文件处理

▶ read_csv、read_table函数主要参数：

- ▶ sep 指定分隔符，可用正则表达式如'\s+'
- ▶ header=None 指定文件无列名
- ▶ name 指定列名
- ▶ index_col 指定某列作为索引
- ▶ skip_row 指定跳过某些行
- ▶ na_values 指定某些字符串表示缺失值
- ▶ parse_dates 指定某些列是否被解析为日期，类型为布尔值或列表

pandas-文件处理

- ▶ 写入到csv文件：to_csv函数
- ▶ 写入文件函数的主要参数：
 - ▶ sep 指定文件分隔符
 - ▶ na_rep 指定缺失值转换的字符串，默认为空字符串
 - ▶ header=False 不输出列名一行
 - ▶ index=False 不输出行索引一列
 - ▶ cols 指定输出的列，传入列表

pandas-文件处理

- ▶ pandas支持的其他文件类型：
 - ▶ json, XML, HTML, 数据库, pickle, excel...

Matplotlib

数据可视化工具包

Matplotlib-介绍

- ▶ Matplotlib是一个强大的Python绘图和数据可视化的工具包。
- ▶ 安装方法: `pip install matplotlib`
- ▶ 引用方法: `import matplotlib.pyplot as plt`
- ▶ 绘图函数: `plt.plot()`
- ▶ 显示图像: `plt.show()`

Matplotlib-plot函数

- ▶ plot函数：绘制点图或线图
 - ▶ 线型linestyle (-,-.,--,...)
 - ▶ 点型marker (v,^,s,*,H,+,x,D,o,...)
 - ▶ 颜色color (b,g,r,y,k,w,...)
- ▶ plot函数绘制多条曲线

Matplotlib-图像标注

- ▶ 设置图像标题: `plt.title()`
- ▶ 设置x轴名称: `plt.xlabel()`
- ▶ 设置y轴名称: `plt.ylabel()`
- ▶ 设置x轴范围: `plt.xlim()`
- ▶ 设置y轴范围: `plt.ylim()`
- ▶ 设置x轴刻度: `plt.xticks()`
- ▶ 设置y轴刻度: `plt.yticks()`
- ▶ 设置曲线图例: `plt.legend()`

Matplotlib-画布与图

- ▶ 画布: figure

- ▶ `fig = plt.figure()`

- ▶ 图: subplot

- ▶ `ax1 = fig.add_subplot(2,2,1)`

- ▶ 调节子图间距:

- ▶ `subplots_adjust(left, bottom, right, top, wspace, hspace)`