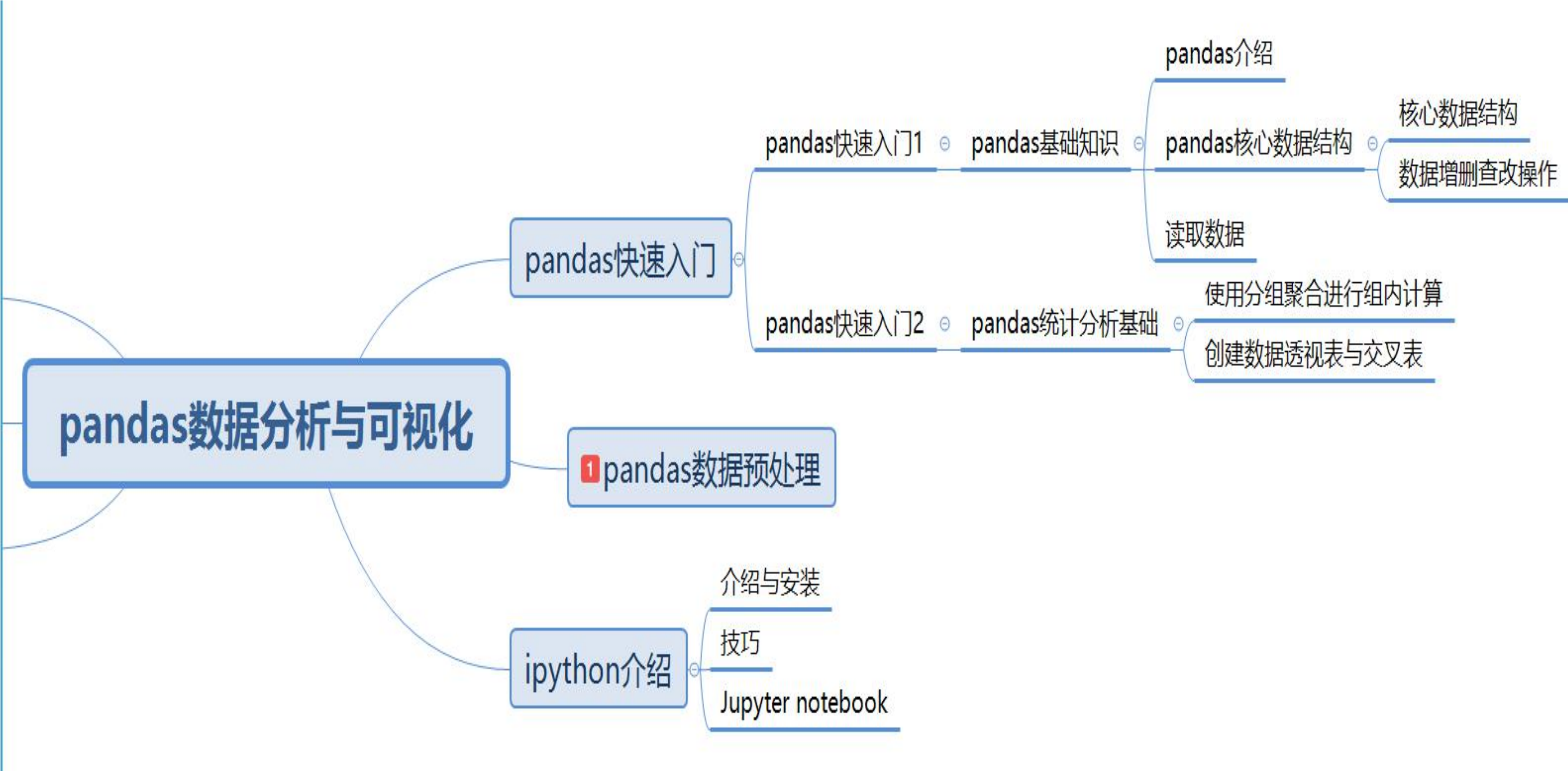


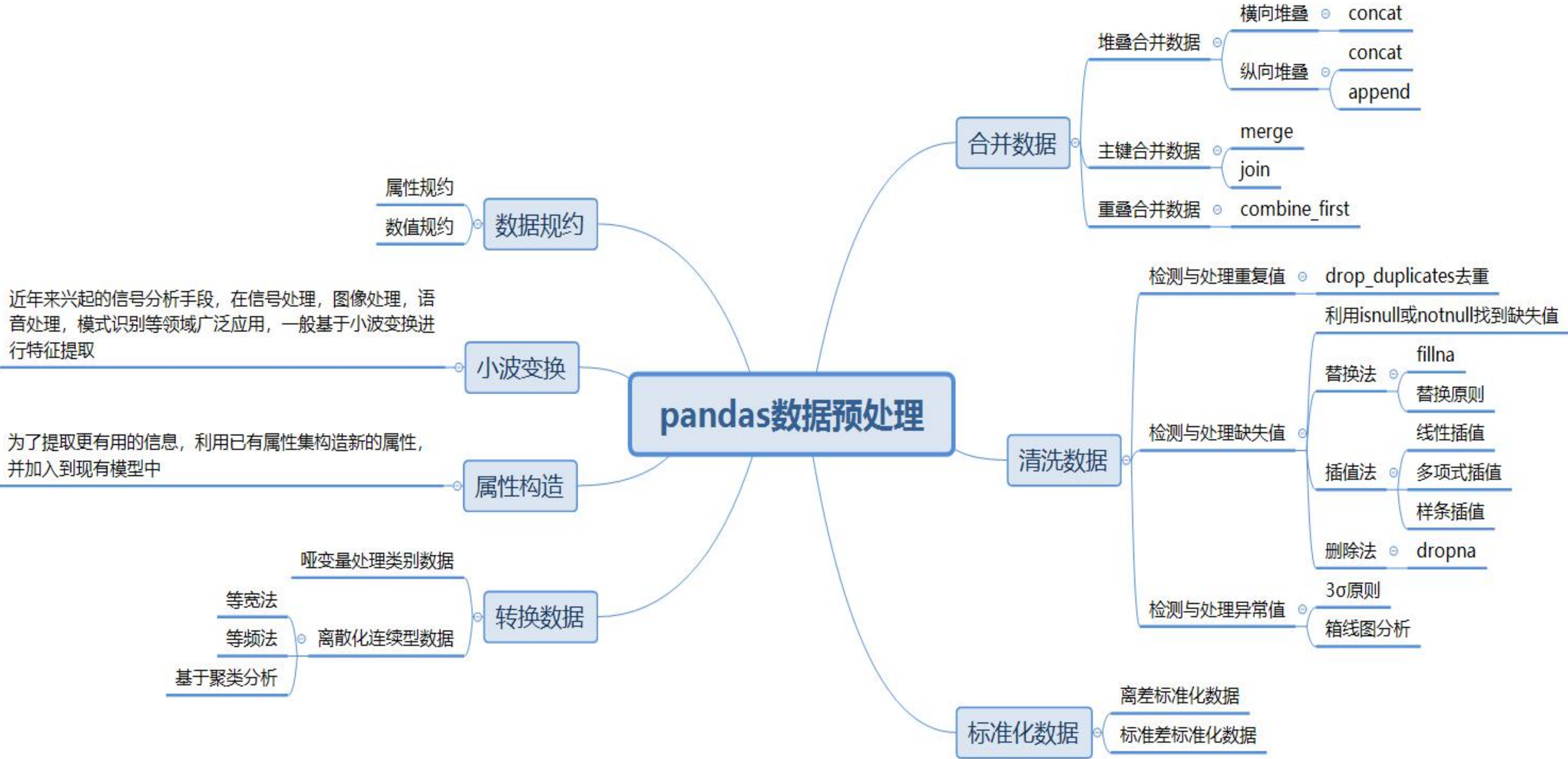
# 数据分析与可视化

python数据分析与可视化



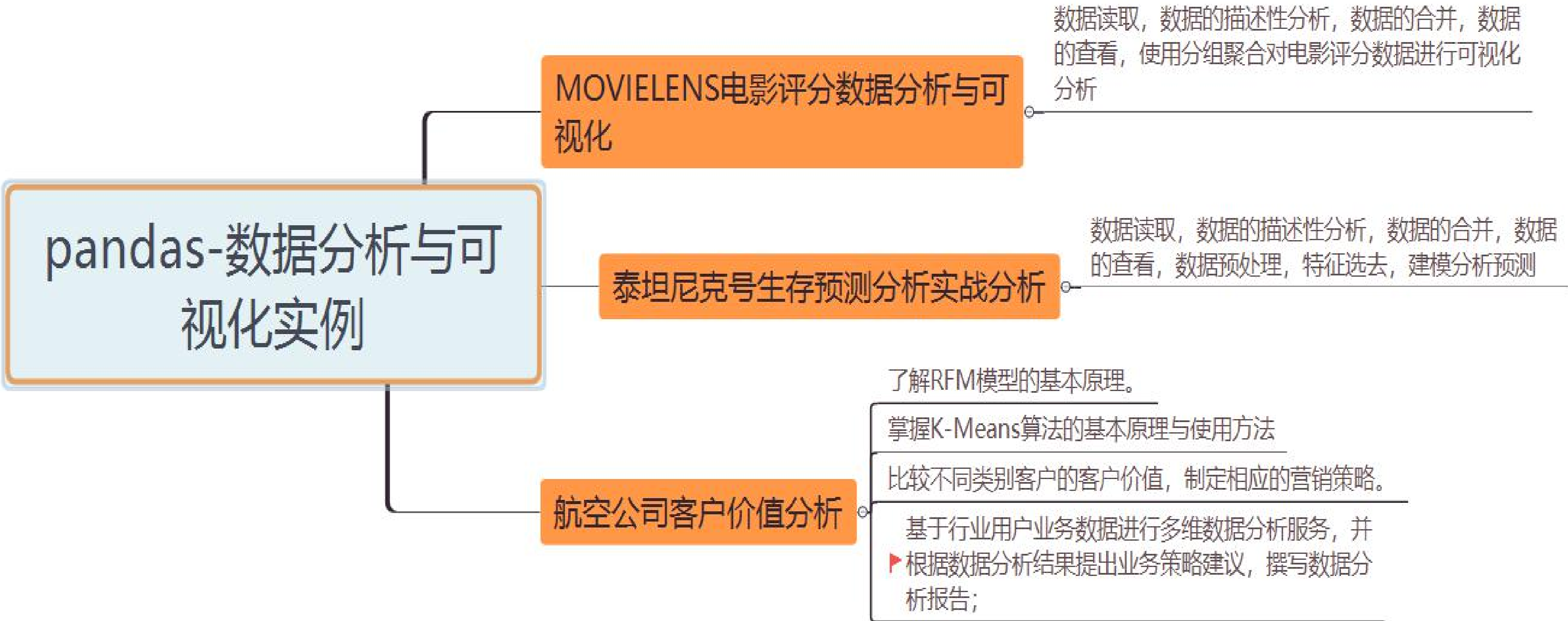
以理服人







以理服人



# 1

## pandas数据分析

基础知识

统计分析基础

Jupyter notebook介绍

数据预处理

---

达内教育研究院

1. pandas介绍

2. pandas核心数据结构及获取数据

名称：Python Data Analysis Library 或 pandas

介绍：是基于NumPy 的一种工具，该工具是为了解决数据分析任务而创建的。Pandas 纳入了大量库和一些标准的数据模型，提供了高效地操作大型数据集所需的工具。

定义1：pandas提供了大量能使我们快速便捷地处理数据的函数和方法。Python长期以来一直非常适合数据整理和准备，你很快就会发现，它是使Python成为强大而高效的数据分析环境的重要因素之一。

定义2：pandas是python里面分析结构化数据的工具集，基础是numpy，图像库是matplotlib

**结构化数据：**是数据的数据库(即行数据,存储在数据库里,可以用二维表结构来逻辑表达实现的数据)

## Structured Data

Size	#bedrooms	...	Price (1000\$s)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540



**非结构化数据：**包括所有格式的办公文档、文本、图片、HTML、各类报表、图像和音频/视频信息等等

## Unstructured Data



Audio



Image

Four scores and seven  
years ago...

Text

## 半结构化数据:

所谓半结构化数据，就是介于完全结构化数据（如关系型数据库、面向对象数据库中的数据）和完全无结构的数据（如声音、图像文件等）之间的数据，XML、json就属于半结构化数据。

它一般是自描述的，数据的结构和内容混在一起，没有明显的区分。

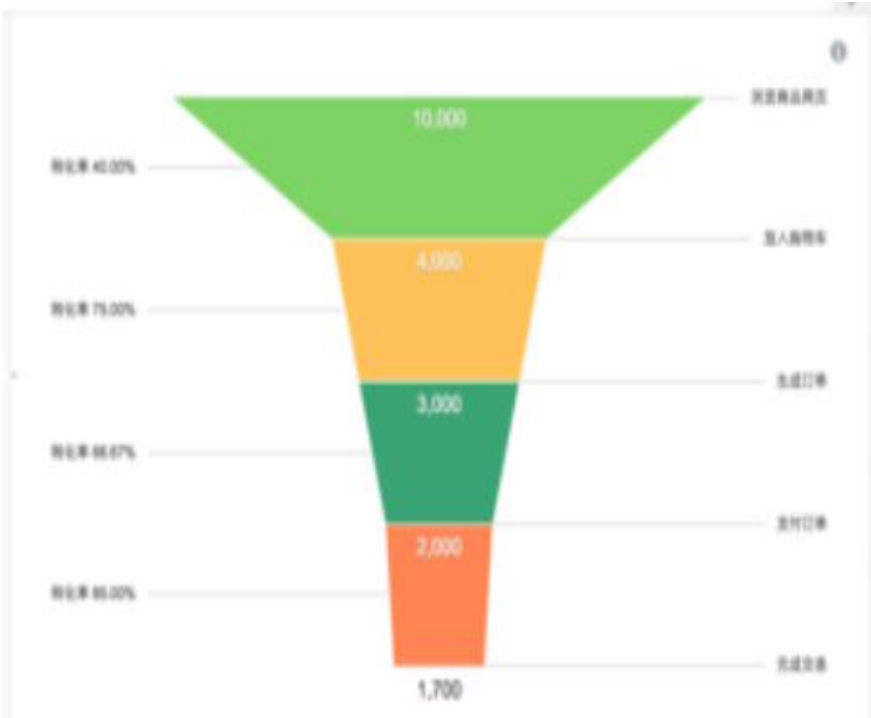
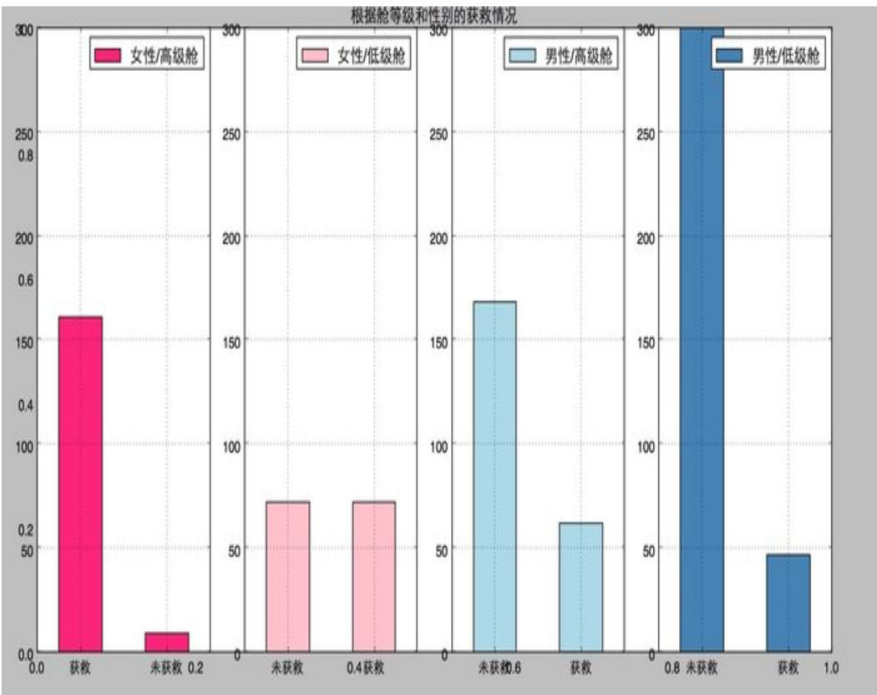
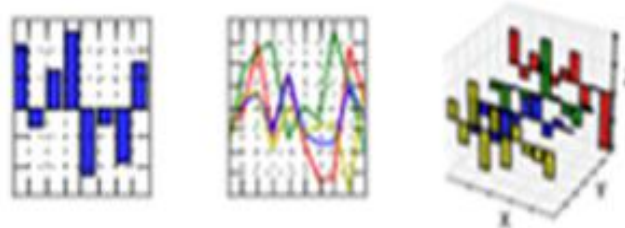


做数据分析的相关工作，我们会见到各个方面的数据，可以是产品的用户体验数据，可以是产品的质量监测数据，也可以是证券市场数据，也可以是资本市场数据等等。这些数据杂乱无章，而且随着时代发展，数据量越来越庞大，单纯的Excel操作渐渐开始捉襟见肘。

<http://pandas.pydata.org/pandas-docs/version/0.23/>

以理服人

pandas



先导：什么是数据结构？？

入门：pandas数据结构介绍

- 1.查改增删DataFrame数据
- 2.描述分析DataFrame数据
- 3.pandas获取数据



什么是数据结构？？？

最简单的答案是，当你有几千几万个数据点时，每一个存放数据点的位置之间的排列关系就是数据结构。

数据结构是计算机存储、组织数据的方式。

数据结构是指相互之间存在一种或多种特定关系的数据元素的集合。通常情况下，精心选择的数据结构可以带来更高的运行或者存储效率。数据结构往往同高效的检索算法和索引技术有关。

## pandas核心数据结构--- Series

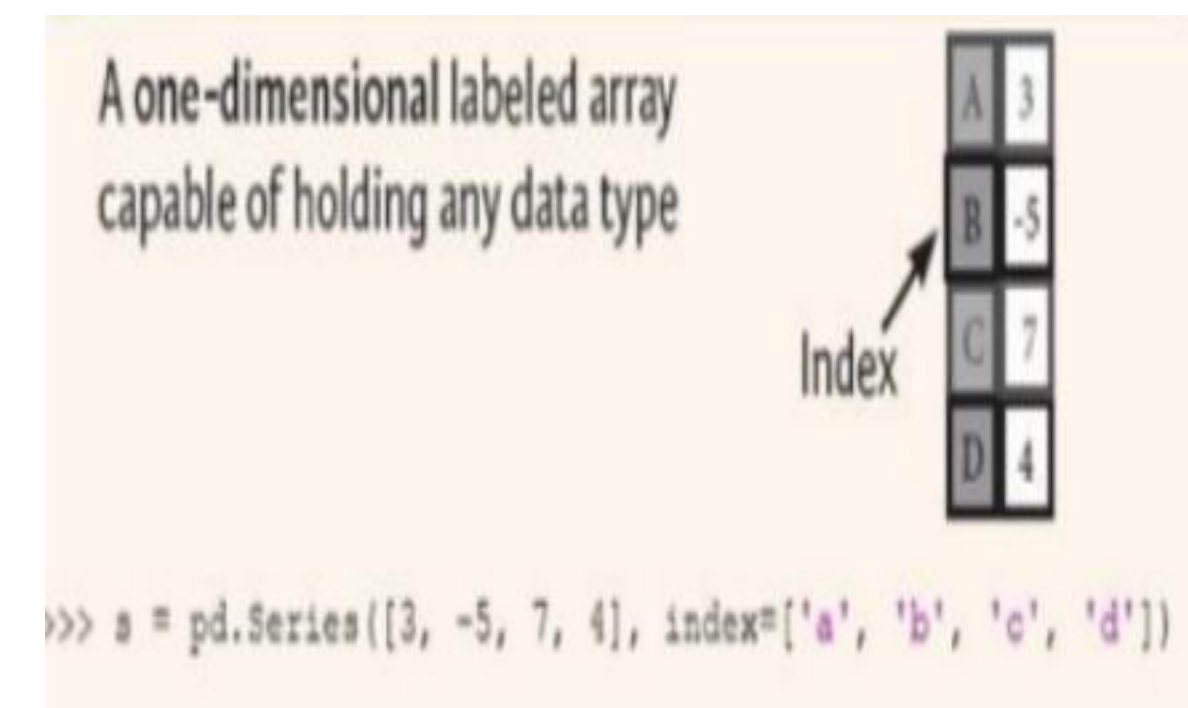
Series可以理解为一个一维的数组，只是index可以自己改动。类似于定长的有序字典，有Index和value。

创建的方法统一为pd.Series(data,index=)。打印的时候按照index赋值的顺序，

index参数默认从0开始的整数，也是Series的绝对位置，即使index被赋值之后，绝对位置不会被覆盖。

Series可以通过以下形式创建：

python的dict、  
numpy当中的ndarray（numpy中的基本数据结构）、  
具体某个数值。index赋值必须是list类型。





## pandas数据结构--- DataFrame

DataFrame是一个类似于表格的数据类型，如图：

有这样一些参数：data（方框内的数据）：numpy ndarray (structured or homogeneous), dict, or DataFrame

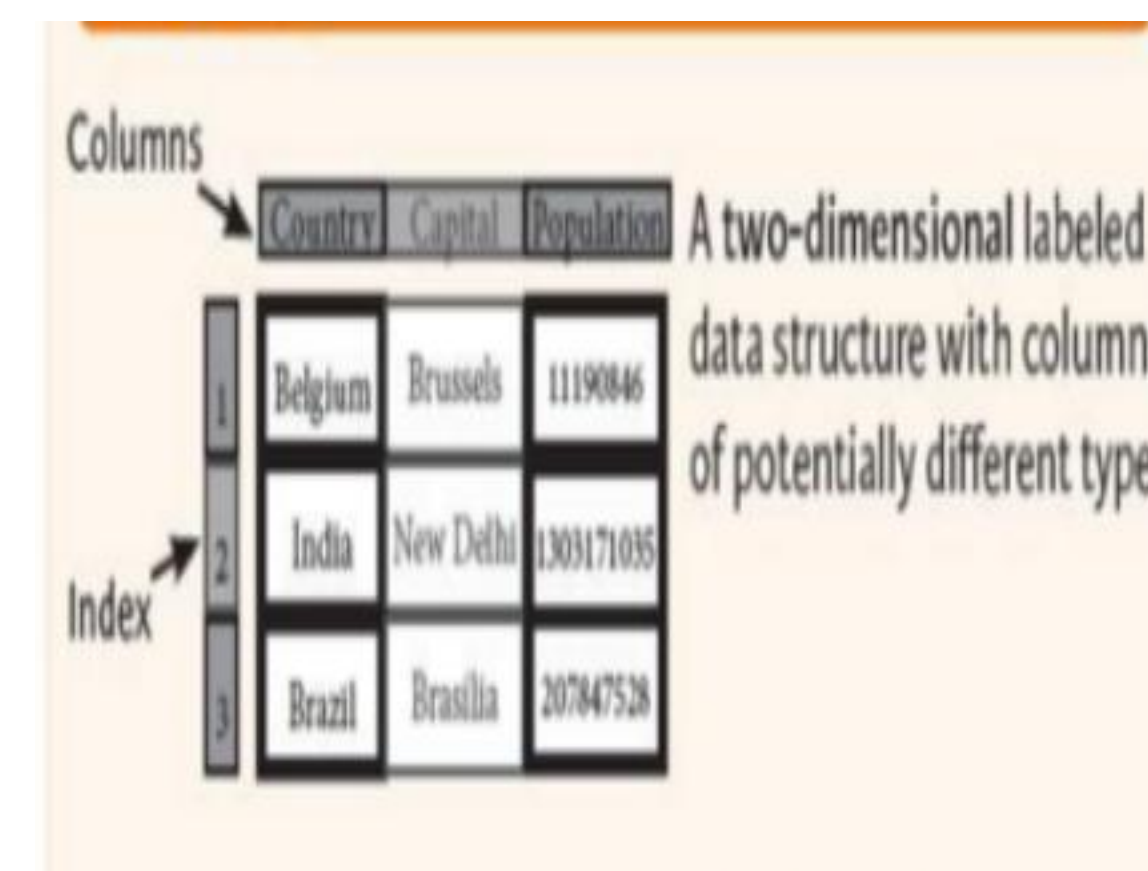
index（行索引索引）：Index or array-like

columns（列索引）：Index or array-like

dtype（data的数据类型）：dtype, default None

DataFrame可以理解为一个二维数组，index有两个维度，可更改。

DataFrame统一的创建形式为：pd.DataFrame(data,columns=,index=)其中columns为列的索引，index为行的索引。index或者columns如果不进行设置则默认为0开始的整数



A two-dimensional labeled data structure with columns of potentially different types

	Country	Capital	Population
1	Belgium	Brussels	11190846
2	India	New Delhi	1303171035
3	Brazil	Brasilia	207847528

完成以下练习：

使用pandas创建以下数据，要求采取合适的数据结构，打开pandas\_data\_structure.py文件完成

pd.Series(data,index=)

pd.DataFrame(data,columns=,index=)

以理服人

leo	90
kate	86
john	70

A	-0.192008
B	-0.617690
C	0.650799
D	-1.515807
E	-0.194408

0	6
---	---

	a	b	c	d
2010-01-01	-1.280865	0.646598	1.291709	1.152734
2010-01-02	0.054346	-0.776157	-0.408969	0.465922
2010-01-03	-0.814692	0.743723	-1.055881	-1.617472
2010-01-04	-0.072644	0.629756	0.277622	1.243546
2010-01-05	-0.373623	0.321123	-1.900529	-0.607704
2010-01-06	-0.198908	-0.302633	-1.782796	1.367930



### 1、对Series操作

简单来说就是通过索引查看：(1) 通过index对应的标签；(2)通过绝对位置查看。

如果通过绝对位置查看，会使用s[XXX]，XXX可以是绝对位置的数字，列表，或者表达式等

基础属性

函数	返回值
values	元素
index	索引
columns	列名
dtypes	类型
size	元素个数
ndim	维度数
shape	数据形状（行列数目）

以理服人



## 2.查看访问DataFrame中的数据

### (1).数据基本查看方式

对单列数据的访问：DataFrame的单列数据为一个Series。根据DataFrame的定义可以知晓DataFrame是一个带有标签的二维数组，每个标签相当每一列的列名。`df.a` `df['a']`

对多列数据访问：访问DataFrame多列数据可以将多个列索引名称视为一个列表，`df[['a','b']]`

## 2.查看访问DataFrame中的数据

### (1).数据基本查看方式

#### ➤ 对某几行访问：

- 如果只是需要访问DataFrame某几行数据的实现方式则采用数组的选取方式，使用 “:” 。
- head和tail也可以得到多行数据，但是用这两种方法得到的数据都是从开始或者末尾获取的连续数据。默认参数为访问5行，只要在方法后方的 “()” 中填入访问行数即可实现目标行数的查看。

## 2.查看访问DataFrame中的数据

### (1).查看访问DataFrame中的数据——loc,iloc方法介绍

- loc方法是针对DataFrame索引名称的切片方法，如果传入的不是索引名称，那么切片操作将无法执行。利用loc方法，能够实现所有单层索引切片操作。loc方法使用方法如下。

*DataFrame.loc[行索引/名称或条件, 列索引/名称]*

- iloc和loc区别是iloc接收的必须是行索引和列索引的位置。iloc方法的使用方法如下。

*DataFrame.iloc[行索引/位置, 列索引/位置]*



### 2.查看访问DataFrame中的数据

#### (1).查看访问DataFrame中的数据——loc,iloc方法介绍

使用loc方法和iloc实现多列切片，其原理的通俗解释就是将多列的列名或者位置作为一个列表或者数据传入。

使用loc，iloc方法可以取出DataFrame中的任意数据。

loc内部还可以传入表达式，结果会返回满足表达式的所有值。

### 2.查看访问DataFrame中的数据

#### (1).查看访问DataFrame中的数据——loc,iloc方法介绍

loc更加灵活多变，代码的可读性更高，iloc的代码简洁，但可读性不高。具体在数据分析工作中使用哪一种方法，根据情况而定，大多数时候建议使用loc方法。

在loc使用的时候内部传入的行索引名称如果为一个区间，则前后均为闭区间；iloc方法使用时内部传入的行索引位置或列索引位置为区间时，则为前闭后开区间。

### 2.查看访问DataFrame中的数据

#### (1).查看访问DataFrame中的数据——切片方法之ix

- ix方法更像是loc和iloc两种切片方法的融合。ix方法在使用时既可以接收索引名称也可以接收索引位置。其使用方法如下。

*DataFrame.ix[行索引的名称或位置或者条件, 列索引名称或位置]*

- 使用ix方法时有个注意事项，第一条，当索引名称和位置存在部分重叠时，ix默认优先识别名称。



### 2.查看访问DataFrame中的数据

#### (1).查看访问DataFrame中的数据——切片方法之ix

控制ix方法需要注意以下几点。

使用列索引名称，而非列索引位置。主要用来保证代码可读性。

使用列索引位置时，需要注解。同样保证代码可读性。

除此之外ix方法还有一个缺点，就是在面对数据量巨大的任务的时候，其效率会低于loc和iloc方法，所以在日常的数据分析工作中建议使用loc和iloc方法来执行切片操作。

1.select by label : loc, 标签参数查找DataFrame.loc[index:index,[ 'columns' ]], loc方法当中的columns可以选择多列, 如果表示只按列选择的话index可以不填但是冒号 ( : ) 和逗号 ( , ) 一定要写

eg: df.loc['2010-01-01':'2010-01-04', ['a','b']]

2.select by position:iloc: 索引查找, 把标签换成绝对位置

eg: df.iloc[:4,[0,1]]

3.mixed selection:ix

eg:df.ix[:4,['a','b']]

	a	b	c	d
2010-01-01	0.537276	1.600419	1.037267	-0.416919
2010-01-02	1.466693	1.418624	-1.792067	1.595889
2010-01-03	0.694957	0.326015	0.463335	0.568509
2010-01-04	0.091378	0.223707	0.525408	-0.162677
2010-01-05	-0.685320	-1.928679	-0.541296	-0.401515
2010-01-06	0.428047	-1.245919	0.781141	0.469301

完成以下练习：

使用pandas创建以下数据，要求采取合适的数据结构构建下图所示 打开practice.py文件练习

进行对于数据的选取，切片练习

	name	gender	age
0	Snow	M	22
1	Tyrion	M	32
2	Sansa	F	18
3	Arya	F	14

1. 选取多列， gender和age列
2. 读取第1行到第2行的数据
2. 读取第1行和第3行，从第0列到第2列（不包含第2列）的数据
3. 读取倒数第3行到倒数第1行的数据，不包括最后一行



### (3).更新修改DataFrame中的数据

更改DataFrame中的数据，原理是将这部分数据提取出来，重新赋值为新的数据。

需要注意的是，数据更改直接针对DataFrame原数据更改，操作无法撤销，如果做出更改，需要对更改条件做确认或对数据进行备份。

eg: 请把上个例子中Snow年龄改为25 打开practice.py文件

#iat取某个单值,只能数字索引 df.iat[1,1]#第1行, 1列

#at取某个单值,只能index和columns索引名称 df.at['one','a']#one行, a列

(4).为DataFrame增添数据

DataFrame添加一列的方法非常简单，只需要新建一个列索引。并对该索引下的数据进行赋值操作即可。

新增的一列值是相同的则直接赋值一个常量即可。

eg：给上个例子学生加上score一列。分别为：80，98，67，90

给上个例子gender后面加上city一列。

给上个例子学生加上一行信息 信息分别为：lisa F 北京 19 100

	name	gender	age
0	Snow	M	22
1	Tyrion	M	32
2	Sansa	F	18
3	Arya	F	14

(5). 删除某列或某行数据

删除某列或某行数据需要用到pandas提供的方法drop，drop方法的用法如下。

axis为0时表示删除行，axis为1时表示删除列。

drop(labels, axis=0, level=None, inplace=False, errors='raise')

常用参数如下所示。

参数名称	说明
labels	接收string或array。代表删除的行或列的标签。无默认。
axis	接收0或1。代表操作的轴向。默认为0。
levels	接收int或者索引名。代表标签所在级别。默认为None。
inplace	接收boolean。代表操作是否对原数据生效。默认为False。



### (6). 对数据进行排序

`df.sort_index(axis=,ascending=)` axis为0/1的参数, 表示按行/按列排序;

ascending为boolean参数, False表示降序, True表示升序。

`df.sort_values(by=, ascending=)` by表示按哪一个columns参数排序。

描述分析DataFrame数据

(1)数值型特征的描述性统计——NumPy中的描述性统计函数

数值型数据的描述性统计主要包括了计算数值型数据的完整情况、最小值、均值、中位数、最大值、四分位数、极差、标准差、方差、协方差和变异系数等。在NumPy库中一些常用的统计学函数如下表所示。

pandas库基于NumPy，自然也可以用这些函数对数据框进行描述性统计。

函数名称	说明	函数名称	说明
np.min	最小值	np.max	最大值
np.mean	均值	np.ptp	极差
np.median	中位数	np.std	标准差
np.var	方差	np.cov	协方差

以理服人

(1).数值型特征的描述性统计—— pandas描述性统计方法

pandas还提供了更加便利的方法来计算均值，如detail['amounts'].mean()。

pandas还提供了方法叫作describe，能够一次性得出数据框所有数值型特征的非空值数目、均值、四分位数、标准差。

方法名称	说明	方法名称	说明
min	最小值	max	最大值
mean	均值	ptp	极差
median	中位数	std	标准差
var	方差	cov	协方差
sem	标准误差	mode	众数
skew	样本偏度	kurt	样本峰度
quantile	四分位数	count	非空值数目
describe	描述统计	mad	平均绝对离差

以理服人



以理服人

方法名称	说明	方法名称	说明
min	最小值	max	最大值
mean	均值	ptp	极差
median	中位数	std	标准差
var	方差	cov	协方差
sem	标准误差	mode	众数
skew	样本偏度	kurt	样本峰度
quantile	四分位数	count	非空值数目
describe	描述统计	mad	平均绝对离差

平均绝对离差指的是各观察值与平均值的距离总和，然后取其平均数

偏度含义是统计数据分布偏斜方向和程度的度量，是统计数据分布非对称程度的数字特征。

峰度，表征概率密度分布曲线在平均值处峰值高低的特征数，反映了峰部的尖度。

### (2).类别型特征的描述性统计

描述类别型特征的分布状况，可以使用频数统计表。pandas库中实现频数统计的方法为value\_counts。

describe方法除了支持传统数值型以外，还能够支持对数据进行描述性统计，四个统计量分别为列非空元素的数目，类别的数目，数目最多的类别，数目最多类别的数目。

我们平时接触最多的轻量级数据，一般是录入在Excel中，如果想要用Python处理这些数据，就需要将数据导出来，这一步是初始工作，但也是操作频繁必不可少的一步。

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet
binary	Msgpack	read_msgpack	to_msgpack
binary	Stata	read_stata	to_stata
binary	SAS	read_sas	
binary	Python Pickle Format	read_pickle	to_pickle
SQL	SQL	read_sql	to_sql
SQL	Google Big Query	read_gbq	to_gbq



## 文本文件读取

- 文本文件是一种由若干行字符构成的计算机文件，它是一种典型的顺序文件。
- csv是一种逗号分隔的文件格式，因为其分隔符不一定是逗号，又被称为字符分隔文件，文件以纯文本形式存储表格数据（数字和文本）。

```
1.4, 0.2, 0  
1.4, 0.2, 0  
1.3, 0.2, 0  
1.5, 0.2, 0  
1.4, 0.2, 0
```

iris.csv 鸢尾花数据集

## 文本文件读取

- 使用read\_table来读取文本文件。

```
pandas.read_table(filepath_or_buffer, sep='\t', header='infer', names=None, index_col=None, dtype=None, engine=None, nrows=None)
```

- 使用read\_csv函数来读取csv文件。

```
pandas.read_csv(filepath_or_buffer, sep=',', header='infer', names=None, index_col=None, dtype=None, engine=None, nrows=None)
```

## 文本文件读取

➤ read\_table和read\_csv常用参数及其说明。

参数名称	说明
filepath	接收string。代表文件路径。无默认。该字符串可以是一个URL。有效的URL方案包括http, ftp s3和file
sep	接收string。代表分隔符。read_csv默认为 “,” , read_table默认为制表符 “[Tab]” 。
header	接收int或sequence。表示将某行数据作为列名。默认为infer, 表示自动识别。
names	接收array。表示列名。默认为None。
index_col	接收int、sequence或False。表示索引列的位置, 取值为sequence则代表多重索引。默认为None。
dtype	接收dict。代表写入的数据类型 (列名为key, 数据格式为values) 。默认为None。
engine	接收c或者python。代表数据解析引擎。默认为c。
nrows	接收int。表示读取前n行。默认为None。



## 文本文件读取

`read_table`和`read_csv`函数中的`sep`参数是指定文本的分隔符的，如果分隔符指定错误，在读取数据的时候，每一行数据将连成一片。

`header`参数是用来指定列名的，如果是`None`则会添加一个默认的列名。

`encoding`代表文件的编码格式，常用的编码有`utf-8`、`utf-16`、`gbk`、`gb2312`、`gb18030`等。如果编码指定错误数据将无法读取，IPython解释器会报解析错误。

文本文件储存 涉及xlrd openpyxl库的安装

文本文件的存储和读取类似，结构化数据可以通过pandas中的to\_csv函数实现以csv文件格式存储文件。

```
DataFrame.to_csv(path_or_buf=None, sep=',', na_rep='', columns=None, header=True, index=True, index_label=None, mode='w', encoding=None)
```

参数名称	说明	参数名称	说明
path_or_buf	接收string。代表文件路径。无默认。	index	接收boolean，代表是否将行名（索引）写出。默认为True。
sep	接收string。代表分隔符。默认为 “,” 。	index_labels	接收sequence。表示索引名。默认为None。
na_rep	接收string。代表缺失值。默认为 "" 。	mode	接收特定string。代表数据写入模式。默认为w。
columns	接收list。代表写出的列名。默认为None。	encoding	接收特定string。代表存储文件的编码格式。默认为None。
header	接收boolean，代表是否将列名写出。默认为True。		

以理服人

Excel文件读取

pandas提供了read\_excel函数来读取 “xls” “xlsx” 两种Excel文件。

```
pandas.read_excel(io, sheetname=0, header=0, index_col=None, names=None, dtype=None)
```

参数名称	说明
io	接收string。表示文件路径。无默认。
sheetname	接收string、int。代表excel表内数据的分表位置。默认为0。
header	接收int或sequence。表示将某行数据作为列名。默认为infer，表示自动识别。
names	接收int、sequence或者False。表示索引列的位置，取值为sequence则代表多重索引。默认为None。
index_col	接收int、sequence或者False。表示索引列的位置，取值为sequence则代表多重索引。默认为None。
dtype	接收dict。代表写入的数据类型（列名为key，数据格式为values）。默认为None。

以理服人



## Excel文件储存

- 将文件存储为Excel文件，可以使用to\_excel方法。其语法格式如下。

*DataFrame.to\_excel(excel\_writer=None, sheetname=None'', na\_rep='', header=True, index=True, index\_label=None, mode='w', encoding=None)*

- to\_csv方法的常用参数基本一致，区别之处在于指定存储文件的文件路径参数名称为excel\_writer，并且没有sep参数，增加了一个sheetnames参数用来指定存储的Excel sheet的名称，默认为sheet1。

## 1.获取电影用户数据表格

```
1::F::1::10::48067
2::M::56::16::70072
3::M::25::15::55117
4::M::45::7::02460
5::M::25::20::55455
```

## 2.获取iris.csv文件并储存

```
1. 4, 0. 2, 0
1. 4, 0. 2, 0
1. 3, 0. 2, 0
1. 5, 0. 2, 0
```

## 3.获取data.xlsx表格文件并储存

age	height	gender
21	165	M
22	145	M
23	164	M
24	165	M
25	166	F
26	167	F

以理服人

完成以下练习：

要求使用pandas合适的方法读取文件，文件在pandas\_student\_02里面get\_data\_practice.py

test.csv, ---泰坦尼克号生存预测分析的测试数据

movies.dat, ratings.dat ---movielens电影评分数据分析的电影详情表，评分表

movies.dat 中需要自定义列名为：['MovieID', 'Title', 'Genres'] 表格中数据是 “::” 隔开

ratings.dat 中需要自定义列名为：['UserID', 'MovieID', 'Rating', 'Timestamp'] 表格中数据是 “::” 隔开

并要求查看共有多少条数据，且打印数据前五行进行查看

## 扩展：数据库数据读取

- pandas提供了读取与存储关系型数据库数据的函数与方法。除了pandas库外，还需要使用SQLAlchemy库建立对应的数据库连接。SQLAlchemy配合相应数据库的Python连接工具（例如MySQL数据库需要安装mysqlclient或者pymysql库），使用create\_engine函数，建立一个数据库连接。
- create\_engine中填入的是一个连接字符串。在使用Python的SQLAlchemy时，MySQL和Oracle数据库连接字符串的格式如下：

数据库产品名+连接工具名：//用户名:密码@数据库IP地址:数据库端口号/数据库名称? charset = 数据库数据编码

```
from sqlalchemy import create_engine
```

```
engine = create_engine('mysql+mysqldb://root@localhost:3306/shop')
```



## 扩展：数据库数据读取

- read\_sql\_table只能够读取数据库的某一个表格，不能实现查询的操作。

```
pandas.read_sql_table(table_name, con, schema=None, index_col=None, coerce_float=True, columns=None)
```

- read\_sql\_query则只能实现查询操作，不能直接读取数据库中的某个表。

```
pandas.read_sql_query(sql, con, index_col=None, coerce_float=True)
```

- read\_sql是两者的综合，既能够读取数据库中的某一个表，也能够实现查询操作。

```
pandas.read_sql(sql, con, index_col=None, coerce_float=True, columns=None)
```

pandas三个数据库数据读取函数的参数几乎完全一致，唯一的区别在于传入的是语句还是表名。

参数名称	说明
sql or table_name	接收string。表示读取的数据的表名或者sql语句。无默认。
con	接收数据库连接。表示数据库连接信息。无默认
index_col	接收int， sequence或者False。表示设定的列作为列名， 如果是一个数列则是多重索引。 默认为None。
coerce_float	接收boolean。将数据库中的decimal类型的数据转换为pandas中的float64类型的数据。 默认为True。
columns	接收list。表示读取数据的列名。默认为None。

数据库数据存储

数据库数据读取有三个函数，但数据存储则只有一个to\_sql方法。

```
DataFrame.to_sql(name, con, schema=None, if_exists='fail', index=True, index_label=None, dtype=None)
```

参数名称	说明
name	接收string。代表数据库表名。无默认。
con	接收数据库连接。无默认。
if_exists	接收fail, replace, append。fail表示如果表名存在则不执行写入操作；replace表示如果存在，将原数据库表删除，再重新创建；append则表示在原数据库表的基础上追加数据。默认为fail。
index	接收boolean。表示是否将行索引作为数据传入数据库。默认True。
index_label	接收string或者sequence。代表是否引用索引名称，如果index参数为True此参数为None则使用默认名称。如果为多重索引必须使用sequence形式。默认为None。
dtype	接收dict。代表写入的数据类型（列名为key，数据格式为values）。默认为None。

以理服人

## 扩展：其他格式

**HDF5**: HDF5 是一种层次化的格式（**hierarchical format**），经常用于存储复杂的科学数据。例如 **MATLAB** 就是用这个格式来存储数据。在存储带有关联的元数据（**metadata**）的复杂层次化数据的时候，这个格式非常有用，例如计算机模拟实验的运算结果等等，

**pandas** 还提供一个直接读取 **h5** 文件的函数： `pd.HDFStore`

**Json**: 通过 **json** 模块转换为字典，再转换为 **DataFrame**， `pd.read_json`

**MongoDB** 数据库： 需要结合相应的数据库模块，如： **pymongo**，再通过游标把数据读出来，转换为 **DataFrame**



谢谢