Python 数据科学 *速查表*
Pandas 基础

呆鸟译

Pandas

Pandas 是基于 Numpy 创建的 Python 库，为 Python 提供了易于使用的**数据结构**和**数据分析**工具。

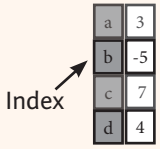
使用以下语句导入 Pandas 库：

```
>>> import pandas as pd
```

Pandas 数据结构

Series - 序列

存储任意类型数据的**一维**数组




```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame - 数据框

列 → 存储不同类型数据的**二维**数组

索引

	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasília	207847528

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasília'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

调用帮助

```
>>>help(pd.Series.loc)
```

选择

参阅 NumPy Arrays

取值

```
>>> s['b']
 -5
```
取序列的值

```
>>> df[1:]
   Country     Capital   Population
1   India   New Delhi   1303171035
2  Brazil    Brasília    207847528
```
取数据框的子集

选取、布尔索引及设置值

按位置

```
>>> df.iloc[[0],[0]]
 'Belgium'
>>> df.iat([0],[0])
 'Belgium'
```
按行与列的位置选择某值

按标签

```
>>> df.loc[[0], ['Country']]
 'Belgium'
>>> df.at([0], ['Country'])
 'Belgium'
```
按行与列的名称选择某值

按标签/位置

```
>>> df.ix[2]
Country      Brazil
Capital    Brasília
Population  207847528
```
选择某行

```
>>> df.ix[:,'Capital']
0     Brussels
1    New Delhi
2     Brasília
```
选择某列

```
>>> df.ix[1,'Capital']
 'New Delhi'
```

布尔索引

```
>>> s[~(s > 1)]
```
序列 S 中没有大于1的值

```
>>> s[(s < -1)  |  (s > 2)]
```
序列 S 中小于-1或大于2的值

```
>>> df[df['Population']>1200000000]
```
使用筛选器调整数据框

设置值

```
>>> s['a'] = 6
```
将序列 S 中索引为 a 的值设为6

删除数据

```
>>> s.drop(['a', 'c'])
```
按索引删除序列的值 (axis=0)

```
>>> df.drop('Country', axis=1)
```
按列名删除数据框的列(axis=1)

排序和排名

```
>>> df.sort_index()
```
按索引排序

```
>>> df.sort_values(by='Country')
```
按某列的值排序

```
>>> df.rank()
```
数据框排名

查询序列与数据框的信息

基本信息

```
>>> df.shape
```
(行,列))

```
>>> df.index
```
获取索引

```
>>> df.columns
```
获取列名

```
>>> df.info()
```
获取数据框基本信息

```
>>> df.count()
```
非Na值的数量

汇总

```
>>> df.sum()
```
合计

```
>>> df.cumsum()
```
累计

```
>>> df.min()/df.max()
```
最小值除以最大值

```
>>> df.idxmin()/df.idxmax()
```
索引最小值除以索引最大值

```
>>> df.describe()
```
基础统计数据

```
>>> df.mean()
```
平均值

```
>>> df.median()
```
中位数

应用函数

```
>>> f = lambda x: x*2
```
应用匿名函数lambda

```
>>> df.apply(f)
```
应用函数

```
>>> df.applymap(f)
```
对每个单元格应用函数

数据对齐

内部数据对齐

如有不一致的索引，则使用NA值：

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
 a     10.0
 b      NaN
 c      5.0
 d      7.0
```

使用 Fill 方法运算

还可以使用 Fill 方法进行内部对齐运算：

```
>>> s.add(s3, fill_value=0)
 a    10.0
 b    -5.0
 c    5.0
 d    7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

输入/输出

读取/写入CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

读取/写入Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

读取内含多个表的Excel

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

读取和写入 SQL 查询及数据库表

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

read_sql()是 read_sql_table() 与 read_sql_query()的便捷打包器

```
>>> pd.to_sql('myDf', engine)
```