导入 Pandas 与 NumPy

```
import numpy as np
import pandas as pd
```

生成对象

用值列表生成 Series 时，Pandas 默认自动生成整数索引：

```
s = pd.Series([1, 3, 5, np.nan, 6, 8])
```

用含日期时间索引与标签的 NumPy 数组生成 DataFrame：

```
dates = pd.date_range('20130101', periods=6)
df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))
```

用 Series 字典对象生成 DataFrame:

```
df1 = pd.DataFrame({'A': 1.,
                    'B': pd.Timestamp('20130102'),
                    'C': pd.Series(1, index=list(range(4)), dtype='float32'),
                    'D': np.array([3] * 4, dtype='int32'),
                    'E': pd.Categorical(["test", "train", "test", "train"]),
                    'F': 'foo'})
```

查看数据

```
print(df.head()) # 查看 DataFrame 头部数据
print(df.tail(3)) # 查看 DataFrame 尾部数据
print(df.index) # 显示索引
print(df.columns) # 显示列名
print(df.to_numpy()) # 输出底层数据的 NumPy 对象
print(df.describe()) # 查看数据的统计摘要
print(df.T) # 转置数据
print(df.sort_index(axis=1, ascending=False)) # 按轴排序
print(df.sort_values(by='B')) # 按值排序
```

获取数据

```
print(df['A']) # 选择A列，产生 Series，与 df.A 等效
print(df[0:3]) # 切片输出前三行
print(df['20130102':'20130104']) # 切片输出'20130102'到'20130104'行
```

按标签选择

```
print(df.loc[dates[0]]) # 提取一行数据
print(df.loc[:, ['A', 'B']]) # 选择多列数据
print(df.loc['20130102':'20130104', ['A', 'B']]) # 标签切片，包含行与列结束点
print(df.loc['20130102', ['A', 'B']]) # 返回对象降维
print(df.loc[dates[0], 'A']) # 提取标量值
print(df.at[dates[0], 'A']) # 快速访问标量
```

按位置选择

```
print(df.iloc[3]) # 用整数位置选择行
print(df.iloc[3:5, 0:2]) # 用整数切片
print(df.iloc[[1, 2, 4], [0, 2]]) # 用整数列表按位置切片
print(df.iloc[1:3, :]) # 显式整行切片
print(df.iloc[:, 1:3]) # 显式整列切片
print(df.iloc[1, 1]) # 显式提取值
print(df.iat[1, 1]) # 快速访问标量
```

布尔索引

```
print(df[df.A > 0]) # 用单列的值选择数据,选择A列数据大于0的行
print(df[df > 0]) # 输出满足条件的值
df2 = df.copy()
df2['E'] = ['one', 'one', 'two', 'three', 'four', 'three']
print(df2[df2['E'].isin(['two', 'four'])]) # 筛选符合条件的行
```

赋值

```
s1 = pd.Series([1, 2, 3, 4, 5, 6], index=pd.date_range('20130102', periods=6)) # 索引自动对齐新增列的数据
df['F'] = s1
print(df)
df.at[dates[0], 'A'] = 0 # 按标签赋值
df.iat[0, 1] = 0 # 按位置赋值
df.loc[:, 'D'] = np.array([5] * len(df)) # 按 NumPy 数组赋值
```

缺失值

Pandas 主要用 np.nan 表示缺失数据。 计算时，默认不包含空值。
重建索引（reindex）可以更改、添加、删除指定轴的索引，并返回数据副本，即不更改原数据。

```
df1 = df.reindex(index=dates[0:4], columns=list(df.columns) + ['E'])
df1.loc[dates[0]:dates[1], 'E'] = 1
print(df1.dropna(how='any')) # 删除所有含缺失值的行
print(df1.fillna(value=5)) # 填充缺失值
print(pd.isna(df1)) # 提取 nan 值的布尔掩码
```

统计

一般情况下，运算时排除缺失值。

```
print(df.mean()) # 取均值
print(df.mean(1)) # 在另一个轴(即，行)上执行同样的操作
```

Apply 函数

```
df.apply(np.cumsum) # Apply 函数处理数据
df.apply(lambda x: x.max() - x.min())
```

字符串方法

```
s = pd.Series(['A', 'B', 'C', 'Aaba', 'Baca', np.nan, 'CABA', 'dog', 'cat'])
s.str.lower()
```

结合（Concat）

```
df = pd.DataFrame(np.random.randn(10, 4))
pieces = [df[:3], df[3:7], df[7:]]
print(pd.concat(pieces))
```

连接（join）

```
left = pd.DataFrame({'key': ['foo', 'foo'], 'lval': [1, 2]})
right = pd.DataFrame({'key': ['foo', 'foo'], 'rval': [4, 5]})
pd.merge(left, right, on='key')
```

追加（Append）

```
df = pd.DataFrame(np.random.randn(8, 4), columns=['A', 'B', 'C', 'D'])
s = df.iloc[3]
print(df.append(s, ignore_index=True))
```

数据输入 / 输出

```
df.to_csv('foo.csv') # 写入 CSV 文件
print(pd.read_csv('foo.csv')) # 读取 CSV 文件数据
Excel
df.to_excel('foo.xlsx', sheet_name='Sheet1') # 写入 Excel 文件
pd.read_excel('foo.xlsx', 'Sheet1', index_col=None, na_values=['NA']) # 读取 Excel 文件
```