Pandas学习

pandas基础

- 一维数据Series
- 二维数据DataFrame
- 三维数据Panel

pip install pandas

`pandas` 中有三种基本结构：

- `Series`
  - 1D labeled homogeneously-typed array
- `DataFrame`
  - General 2D labeled, size-mutable tabular structure with potentially heterogeneously-typed columns
- `Panel`
  - General 3D labeled, also size-mutable array

清華大學 Tsinghua University
清華 Tsinghua iCenter

pandas一维 数据Series

Pandas一维数据结构：Series

`Series` 是一维带标记的数组结构，可以存储任意类型的数据（整数，浮点数，字符串，`Python` 对象等等）。

作为一维结构，它的索引叫做 `index`，基本调用方法为

```
s = pd.Series(data, index=index)
```

其中，`data` 可以是以下结构：

- 字典
- `ndarray`
- 标量，例如 `5`

`index` 是一维坐标轴的索引列表。

如果 `data` 是个 `ndarray`，那么 `index` 的长度必须跟 `data` 一致：

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

```
s = pd.Series(np.random.randn(5), index=["a", "b", "c", "d", "e"])

s
```

```
a    0.039395
b    1.358265
c   -0.631936
d    0.118403
e   -0.603954
dtype: float64
```

```
pd.Series(np.random.randn(5))
```

```
0    0.592287
1   -0.324203
2   -0.392452
3    0.099054
4   -0.238215
dtype: float64
```

向量化操作

简单的向量操作与 `ndarray` 的表现一致：

```
s + s
```

```
a    0.465335
b   -1.036567
c   -1.275411
d   -1.524308
e   -1.196148
dtype: float64
```

```
s * 2
```

```
a    0.465335
b   -1.036567
c   -1.275411
d   -1.524308
e   -1.196148
dtype: float64
```

但 `Series` 和 `ndarray` 不同的地方在于，`Series` 的操作默认是使用 `index` 的值进行对齐的，而不是相对位置：

```
s[1:] + s[:-1]
```

```
a         NaN
b   -1.036567
c   -1.275411
d   -1.524308
e         NaN
dtype: float64
```

对于上面两个不能完全对齐的 `Series`，结果的 `index` 是两者 `index` 的并集，同时不能对齐的部分当作缺失值处理。

pandas二维 数据DataFrame

Pandas二维数据结构：DataFrame

`DataFrame` 是 `pandas` 中的二维数据结构，可以看成一个 `Excel` 中的工作表，或者一个 `SQL` 表，或者一个存储 `Series` 对象的字典。

`DataFrame(data, index, columns)` 中的 `data` 可以接受很多数据类型：

- 一个存储一维数组，字典，列表或者 `Series` 的字典
- 2-D 数组
- 结构或者记录数组
- 一个 `Series`
- 另一个 `DataFrame`

`index` 用于指定行的 `label`， `columns` 用于指定列的 `label`，如果参数不传入，那么会按照传入的内容进行设定。

从 Series 字典中构造

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

如果没有传入 `columns` 的值，那么 `columns` 的值默认为字典 `key`， `index` 默认为所有 `value` 中 `index` 的并集。

```
df = pd.DataFrame(d)

df
```

	one	two
a	1.0	1.0
b	2.0	2.0
c	3.0	3.0
d	NaN	4.0

```
df.index
```

```
Index(['a', 'b', 'c', 'd'], dtype='object')
```

```
df.columns
```

```
Index(['one', 'two'], dtype='object')
```

如果指定了 `columns` 值，会去字典中寻找，找不到的值为 `NaN`：

```
pd.DataFrame(d, index=['d', 'b', 'a'], columns=['two', 'three'])
```

	two	three
d	4.0	NaN
b	2.0	NaN
a	1.0	NaN

列操作

`DataFrame` 可以类似于字典一样对列进行操作:

```
df["one"]
```

```
a    1.0
b    2.0
c    3.0
d    NaN
Name: one, dtype: float64
```

```
df['three'] = df['one'] * df['two']

df['flag'] = df['one'] > 2

df
```

	one	two	three	flag
a	1.0	1.0	1.0	False
b	2.0	2.0	4.0	False
c	3.0	3.0	9.0	True
d	NaN	4.0	NaN	False

操作csv文件

从csv文件中读取：

```
pd.read_csv('foo.csv')
```

保存写入csv文件：

```
df.to_csv('foo.csv')
```

总结索引和选择

Operation	Syntax	Result
Select column	df[col]	Series
Select row by label	df.loc[label]	Series
Select row by integer location	df.iloc[loc]	Series
Slice rows	df[5:10]	DataFrame
Select rows by boolean vector	df[bool_vec]	DataFrame

`DataFrame` 则是个二维结构，这里首先构造一组时间序列，作为我们第一维的下标：

```
dates = pd.date_range('20200101', periods=6)

print(dates)
```

```
DatetimeIndex(['2020-01-01', '2020-01-02', '2020-01-03', '2020-01-04',
               '2020-01-05', '2020-01-06'],
              dtype='datetime64[ns]', freq='D')
```

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

df
```

	A	B	C	D
2020-01-01	0.716750	-0.289619	1.239022	0.450197
2020-01-02	0.142085	-0.861210	-0.664251	-0.058763
2020-01-03	1.293630	0.328164	0.895875	-0.943286
2020-01-04	2.137313	-0.540347	0.698914	1.622408
2020-01-05	1.170859	0.137662	-0.984217	0.701953
2020-01-06	-0.137379	-0.048214	0.731484	0.830699

本地数据访问-Pandas-SQLite

```
import pandas as pd
from sqlalchemy import create_engine
engine = create_engine('sqlite:///:memory:')
with engine.connect() as conn, conn.begin():
    data = pd.read_sql_table('data', conn)
data.to_sql('data', engine)
```

- https://pandas.pydata.org/docs/user_guide/io.html#sql-queries

THANKS