【华子】什么是Pandas?

Pandas是一个强大的分析结构化数据的工具集，基于NumPy构建，提供了 **高级数据结构** 和 **数据操作工具**，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。

- 基础是NumPy，提供了高性能矩阵的运算
- 提供了大量能够快速便捷地处理数据的函数和方法
- 应用于数据挖掘，数据分析
- 提供数据清洗功能

数据类型：Series（一维）

```
# 通过list构建Series
ser_obj = pd.Series(range(10, 20))
print(ser_obj.head(3))
```

数据类型：DataFrame（二维）

```
import numpy as np
import pandas as pd
np.random.seed(10)
array = np.random.randn(3,5)
print(array)
df_data = pd.DataFrame(array)
df_data.columns=['a','b','c','d','e']
print(df_data['a'])
```

```
[[ 1.3315865   0.71527897 -1.54540029 -0.00838385  0.62133597]
 [-0.72008556  0.26551159  0.10854853  0.00429143 -0.17460021]
 [ 0.43302619  1.20303737 -0.96506567  1.02827408  0.22863013]]
0    1.331587
1   -0.720086
2    0.433026
Name: a, dtype: float64
```

删除行/列：

```
array = np.random.randn(3,5)
df_data = pd.DataFrame(array)
df_data.columns=['a','b','c','d','e']
df_data.drop([0],inplace=True)
print(df_data)
print(df_data.iloc[0])
```

```
          a         b         c         d        e
1 -0.720086  0.265512  0.108549  0.004291 -0.17460
2  0.433026  1.203037 -0.965066  1.028274  0.22863
a   -0.720086
b    0.265512
c    0.108549
d    0.004291
e   -0.174600
Name: 1, dtype: float64
```

loc 和 iloc：

loc是基于标签名的索引，也就是我们自定义的索引名；

作用和loc一样，不过是基于索引编号来索引；

```
array = np.random.randn(3,5)
df_data = pd.DataFrame(array)
df_data.columns=['a','b','c','d','e']
df_data.drop([0],inplace=True)
print(df_data)
print(df_data.iloc[0])
print(df_data.loc[1])
```

```
          a         b         c         d         e
1 -0.720086  0.265512  0.108549  0.004291 -0.17460
2  0.433026  1.203037 -0.965066  1.028274  0.22863
a   -0.720086
b    0.265512
c    0.108549
d    0.004291
e   -0.174600
Name: 1, dtype: float64
a   -0.720086
b    0.265512
c    0.108549
d    0.004291
e   -0.174600
Name: 1, dtype: float64
```

函数的使用：

1.可直接使用NumPy的函数：

```
print(np.abs(df_data))
print(np.mean(df_data))
print(np.sum(df_data))
```

```
          a         b         c         d         e
0  1.331587  0.715279 -1.545400 -0.008384  0.621336
1 -0.720086  0.265512  0.108549  0.004291 -0.174600
2  0.433026  1.203037 -0.965066  1.028274  0.228630
          a         b         c         d         e
0  1.331587  0.715279  1.545400  0.008384  0.621336
1  0.720086  0.265512  0.108549  0.004291  0.174600
2  0.433026  1.203037  0.965066  1.028274  0.228630
a    0.348176
b    0.727943
c   -0.800639
d    0.341394
e    0.225122
dtype: float64
a    1.044527
b    2.183828
c   -2.401917
d    1.024182
e    0.675366
dtype: float64
```

2.排序的使用：

data.sort_index() :按照索引排序，默认升序

data.sort_values():按照‘列名’排序（多列可用 ['a','b'....]），默认升序

```
print(df_data.sort_index(axis=0,ascending=False))
print(df_data.sort_values(by='a',ascending=False))
```

原数据：

```
          a         b         c         d         e
0  1.331587  0.715279 -1.545400 -0.008384  0.621336
1 -0.720086  0.265512  0.108549  0.004291 -0.174600
2  0.433026  1.203037 -0.965066  1.028274  0.228630
          a         b         c         d         e
2  0.433026  1.203037 -0.965066  1.028274  0.228630
1 -0.720086  0.265512  0.108549  0.004291 -0.174600
0  1.331587  0.715279 -1.545400 -0.008384  0.621336
          a         b         c         d         e
0  1.331587  0.715279 -1.545400 -0.008384  0.621336
2  0.433026  1.203037 -0.965066  1.028274  0.228630
1 -0.720086  0.265512  0.108549  0.004291 -0.174600
```

处理缺失数据：

```
#构造数据
df_data = pd.DataFrame([np.random.randn(3), [1., 2., np.nan],
                        [np.nan, 4., np.nan], [1., 2., 3.]])
print(df_data.head())
```

```
          0        1        2
0 -0.212698 -0.33914  0.31217
1  1.000000  2.00000      NaN
2       NaN  4.00000      NaN
3  1.000000  2.00000  3.00000
```

注意：如需修改数据，可在参数中适当加入inplace=True

```
print(df_data)
#判断缺失值
print(df_data.isnull())
#删除缺失值
print(df_data.dropna())
#填充缺失值
print(df_data.fillna(100))
```

```
          0         1         2
0 -0.21579  0.989072  0.314754
1  1.00000  2.000000       NaN
2      NaN  4.000000       NaN
3  1.00000  2.000000  3.000000
       0      1      2
0  False  False  False
1  False  False   True
2   True  False   True
3  False  False  False
         0         1         2
0 -0.21579  0.989072  0.314754
3  1.00000  2.000000  3.000000
           0         1           2
0   -0.21579  0.989072    0.314754
1    1.00000  2.000000  100.000000
2  100.00000  4.000000  100.000000
3    1.00000  2.000000    3.000000
```

常用统计描述方法：

方法	说明
count	非NA值的数量
describe	针对Series或各DataFrame列计算汇总统计
min、max	计算最小值和最大值
argmin、argmax	计算能够获取到最小值和最大值的索引位置（整数）
idxmin、idxmax	计算能够获取到最小值和最大值的索引值
quantile	计算样本的分位数（0到1）
sum	值的总和
mean	值的平均数
median	值的算术中位数（50%分位数）
mad	根据平均值计算平均绝对离差
var	样本值的方差
std	样本值的标准差

替换：

```python
# 单个值替换单个值
print(df_data.replace(1, 111))
# 多个值替换一个值
print(df_data.replace([2,4], 2424))
# 多个值替换多个值
print(df_data.replace([2, 4], [222, 444]))
```

```
          0         1         2
0 -0.922909  0.469751 -0.144367
1  1.000000  2.000000       NaN
2       NaN  4.000000       NaN
3  1.000000  2.000000  3.000000
------------
            0         1         2
0   -0.922909  0.469751 -0.144367
1  111.000000  2.000000       NaN
2         NaN  4.000000       NaN
3  111.000000  2.000000  3.000000
          0            1         2
0 -0.922909     0.469751 -0.144367
1  1.000000  2424.000000       NaN
2       NaN  2424.000000       NaN
3  1.000000  2424.000000  3.000000
          0           1         2
0 -0.922909    0.469751 -0.144367
1  1.000000  222.000000       NaN
2       NaN  444.000000       NaN
3  1.000000  222.000000  3.000000
```

读取网络

```python
lst = pd.read_html('http://quote.stockstar.com/')
a=lst2       #获取多个行
a
```

读取数据库

```python
import pymysql
import pandas as pd
con = pymysql.connect(host="127.0.0.1",user="root",password="123456",db="sms")
data_sql=pd.read_sql("select * from school",con)
data_sql.to_csv("test.csv")
```

透视表

```
df = DataFrame({'水果种类':['苹果','苹果','梨','梨','草莓','草莓'],
                '信息':['价格','数量','价格','数量','价格','数量'],
                '值':[4,3,5,4,6,5]})
df.pivot(index='水果种类',columns='信息',values='值')#将水果种类作为行索引，将信息作为列索引
```