Python数据分析实战

第十三课 pandas数据结构

本节课程目标

- Series的创建以及常见操作
- DataFrame的创建以及常见操作

pandas概述

pandas的介绍

- Python Data Analysis Library 或 pandas 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
- 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
- 它使Python成为强大而高效的数据分析环境的重要因素之一

```
import numpy as np

import pandas as pd
from pandas import Series,DataFrame
```

1、Series

Series是一种类似与一维数组的对象，由下面两个部分组成：

- values：一组数据（ndarray类型）
- index：相关的数据索引标签

```
#比如，我们使用numpy先来创建一个array
nd = np.random.randint(0,10,size = 5)
nd
```

```
array([8, 5, 8, 3, 4])
```

1）Series的创建

两种创建方式：

- 列表或者numpy数组
- 字典

```
# 先使用numpy对象
s = Series(nd,index = ['a','b','c','d','e'])
s
```

```
a    8
b    5
c    8
d    3
e    4
dtype: int64
```

```
s2 = Series(nd)
s2
```

```
0    8
1    5
2    8
3    3
4    4
dtype: int64
```

```
#第二种使用列表来进行创建
s3 = Series([2,4,6,8,10])
s3
```

```
0     2
1     4
2     6
3     8
4    10
dtype: int64
```

```
s4 = Series([2,4,6,8],index=['a','b','c','d'])
s4
```

```
a    2
b    4
c    6
d    8
dtype: int64
```

```
nd
```

```
array([8, 5, 8, 3, 4])
```

```
s
```

```
a    8
b    5
c    8
d    3
e    4
dtype: int64
```

```
s['d'] = 100
s
```

```
a      8
b      5
c      8
d    100
e      4
dtype: int64
```

```
nd
```

```
array([  8,   5,   8, 100,   4])
```

```
s4 = Series({'老王':120,'老宋':128,'蓉妹':99},name = 'python')
s4
```

```
老王    120
老宋    128
蓉妹     99
Name: python, dtype: int64
```

```

```

2、Series的索引和切片

Series的索引和切片和之前将的numpy是非常相似的

(1) 显式索引：

```
- 使用index中的元素作为索引值
- 使用.loc[]（推荐）
```

注意，此时是闭区间

```
s
```

```
a      8
b      5
c      8
d    100
e      4
dtype: int64
```

```
#通过键取值
s['b']
```

```
5
```

```
#取两个值
s.loc[['a','d']]
```

```
a      8
d    100
dtype: int64
```

```
s.loc[['a']]
```

```
a    8
dtype: int64
```

隐式索引：

- 使用整数作为索引值
- 使用.iloc[]推荐

注意，此时是半开区间

```
s[0]
```

```
8
```

```
#同样换一个iloc[]，获取一个索引的元素
s.iloc[2]
```

```
8
```

```
#获取多个索引对应的元素
s.iloc[[3,1]]
```

```
d    100
b      5
dtype: int64
```

```
#切片
s['a':'d']#左闭右闭
```

```
a      8
b      5
c      8
d    100
dtype: int64
```

```
#每隔两个元素切一下
s[::2]
```

```
a    8
c    8
e    4
dtype: int64
```

```
s.loc['a':'d']
```

```
a      8
b      5
c      8
d    100
dtype: int64
```

```
s.iloc[0:3]#左闭右开
```

```
a    8
b    5
c    8
dtype: int64
```

3）Series的基本概念

```
可以把Series看成一个定长的有序字典
可以通过shape，size，index,values等得到series的属性
```

```
s
```

```
a      8
b      5
c      8
d    100
e      4
dtype: int64
```

```
display(s.shape,s.size,s.index,s.values)
```

```
(5,)
```

```
5
```

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

```
array([  8,   5,   8, 100,   4])
```

可以通过head(),tail()快速查看Series对象的样式

```
s = Series(np.random.randint(0,100000,size=1500))
s
```

```
0       87099
1       92833
2        6923
3       42750
4       27568
5       29611
6       82535
7       16366
8       43562
9        5371
10      17956
11      50145
12      34557
13      48295
14      96050
15      25509
16      46655
17      24022
18      73287
19      69341
20      57382
21      18099
22      39225
23      60203
24      75504
25      10747
26       3560
27      82768
```

```
28      55624
29      64946
        ...
1470     7078
1471    75876
1472     1036
1473    22530
1474    45767
1475    33219
1476    26323
1477    85634
1478    43802
1479    28685
1480    58820
1481    84782
1482    19172
1483    52925
1484    86581
1485     3609
1486    14519
1487     5014
1488    24574
1489    63662
1490    74045
1491    59705
1492    24679
1493    89147
1494     6445
1495    62556
1496    31461
1497    47815
1498    53316
1499    32314
Length: 1500, dtype: int64
```

```
s.head(2)
```

```
0    87099
1    92833
dtype: int64
```

```
s.tail(2)
```

```
1498    53316
1499    32314
dtype: int64
```

```
s2 = Series([2,3,4,None])
s2
```

```
0    2.0
1    3.0
2    4.0
3    NaN
dtype: float64
```

```
nd = np.array([2,3,4,None])
nd
```

```
array([2, 3, 4, None], dtype=object)
```

```
nd.sum()
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-35-41468127c3ff> in <module>
----> 1 nd.sum()
```

```
/anaconda3/lib/python3.7/site-packages/numpy/core/_methods.py in _sum(a, axis,
dtype, out, keepdims, initial)
     34 def _sum(a, axis=None, dtype=None, out=None, keepdims=False,
     35          initial=_NoValue):
---> 36     return umr_sum(a, axis, dtype, out, keepdims, initial)
     37
     38 def _prod(a, axis=None, dtype=None, out=None, keepdims=False,
```

```
TypeError: unsupported operand type(s) for +: 'int' and 'NoneType'
```

```
s2.sum()
```

```
9.0
```

```
```

可以使用pd.isnull()，pd.notnull()，或自带isnull(),notnull()函数检测缺失数据

4）Series的运算

(1) 适用于numpy的数组运算也适用于Series

(2) Series之间的运算

- 在运算中自动对齐不同索引的数据
- 如果索引不对应，则补NaN
- 注意：要想保留所有的index，则需要使用.add()函数

```
# 通过Series创建两组数据，进行运算
s1 = Series(np.random.randint(0,10,size = 5),index = list('abcde'))


s2 = Series(np.random.randint(0,10,size = 5),index = list('abcgh'))

display(s1,s2)
```

```
a    3
b    7
c    8
d    2
e    7
dtype: int64
```

```
a    3
b    1
c    6
g    8
h    0
dtype: int64
```

```
s1 + s2
```

```
a     6.0
b     8.0
c    14.0
d     NaN
e     NaN
g     NaN
h     NaN
dtype: float64
```

```
s1.add(s2,fill_value=0)
```

```
a     6.0
b     8.0
c    14.0
d     2.0
e     7.0
g     8.0
h     0.0
dtype: float64
```

2、DataFrame

DataFrame是一个【表格型】的数据结构

DataFrame既有行索引，也有列索引。

- 行索引：index
- 列索引：columns
- 值：values（numpy的二维数组）

1）DataFrame的创建

```
#通过numpy来构建一个DataFrame
df = DataFrame(np.random.randint(0,150,size = (5,3)),index =
list('abcde'),columns=['Python','Math','En'])
df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	84	5	69
b	145	119	87
c	61	5	46
d	44	115	123
e	44	48	60

```
#使用字典的方式来创建一个DataFrame
df2 = DataFrame({'Python':[120,137,149],'Math':[119,128,99]},
                index = ['老王','老宋','蓉妹'])
df2
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math
老王	120	119
老宋	137	128
蓉妹	149	99

```
df2.shape
```

```
(3, 2)
```

```
df2.values
```

```
array([[120, 119],
       [137, 128],
       [149,  99]])
```

```
df2.index
```

```
Index(['老王', '老宋', '蓉妹'], dtype='object')
```

```
df2.columns
```

```
Index(['Python', 'Math'], dtype='object')
```

```
df2.dtypes
```

```
Python    int64
Math      int64
dtype: object
```

```

```

```

```

2）DataFrame的索引

(1) 对列进行索引

```
- 通过类似字典的方式
- 通过属性的方式
```

```
df2
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math
老王	120	119
老宋	137	128
蓉妹	149	99

```
df2['Math']
```

```
老王    119
老宋    128
蓉妹     99
Name: Math, dtype: int64
```

```
s = df2.Math
s
```

```
老王    119
老宋    128
蓉妹     99
Name: Math, dtype: int64
```

```
df2[['Math','Python']]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Math	Python
老王	119	120
老宋	128	137
蓉妹	99	149

```
df_math = DataFrame(s)
df_math
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Math
老王	119
老宋	128
蓉妹	99

```

```

(2) 对行进行索引

```
- 使用.ix[]来进行行索引
- 使用.loc[]加index来进行行索引
- 使用.iloc[]加整数来进行行索引
```

```
df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	84	5	69
b	145	119	87
c	61	5	46
d	44	115	123
e	44	48	60

```
df.loc['a']
```

```
Python    84
Math       5
En        69
Name: a, dtype: int64
```

```
df.loc[['a','d']]
```


```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	84	5	69
d	44	115	123

```
df.iloc[[0,1]]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	84	5	69
b	145	119	87

(3) 对元素索引的方法 - 使用列索引 - 使用行索引(iloc[3,1]相当于两个参数;iloc[[3,3]] 里面的[3,3]看做一个参数) - 使用values属性（二维numpy数组）

```
df
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	84	5	69
b	145	119	87
c	61	5	46
d	44	115	123
e	44	48	60

```
df['Python']['a']
```

```
84
```

```

```

```

```

3）DataFrame的运算

下面是Python 操作符与pandas操作函数的对应表：

Python Operator	Pandas Method(s)
`+`	`add()`
`-`	`sub()`, `subtract()`
`*`	`mul()`, `multiply()`
`/`	`truediv()`, `div()`, `divide()`
`//`	`floordiv()`
`%`	`mod()`
`**`	`pow()`

（1） DataFrame之间的运算

同Series一样：

- 在运算中自动对齐不同索引的数据
- 如果索引不对应，则补NaN

```
#创建DataFrame df1 不同人员的各科目成绩，月考一
df1 = DataFrame(np.random.randint(0,150,size = (5,3)),index = list('abcde'),columns=
['Python','Math','En'])
df1
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	4	105	0
b	56	41	14
c	47	52	46
d	76	103	102
e	126	20	3

```
# 创建DataFrame df2 不同人员的各科目成绩，月考二
# 考试的科目不变，但是有新学生转入
df2 = DataFrame(np.random.randint(0,150,size = (6,3)),index = list('abcedf'),columns=
['Python','Math','En'])
df2
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	69	40	19
b	11	15	122
c	18	145	53
e	105	82	8
d	94	102	5
f	133	91	42

```
df1 + df2
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	73.0	145.0	19.0
b	67.0	56.0	136.0
c	65.0	197.0	99.0
d	170.0	205.0	107.0
e	231.0	102.0	11.0
f	NaN	NaN	NaN

```
df3 = df1.add(df2,fill_value=0)
df3
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	73.0	145.0	19.0
b	67.0	56.0	136.0
c	65.0	197.0	99.0
d	170.0	205.0	107.0
e	231.0	102.0	11.0
f	133.0	91.0	42.0

```
df3['a':'e']/2
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Python	Math	En
a	36.5	72.5	9.5
b	33.5	28.0	68.0
c	32.5	98.5	49.5
d	85.0	102.5	53.5
e	115.5	51.0	5.5