pandas基础操作

1 新的数据格式 csv

- 纯文本，使用某个字符集，比如ASCII，Unicode，EBCDIC或GB2312（简体中文环境）
- 有记录组成（典型的是每一条记录）
- 每条记录被分隔符（Delimiter）分隔为字段（Field（computer science））（典型分隔符有逗号，分号，制表符；有时分隔符可以包括可选空格）；
- 每条记录都有相同的字段序列

In [44]:

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

In [103]:

```
abs = 'C:\\Users\\泽娃娃\\Desktop\\text.csv'
df = pd.read_csv(abs, encoding='gbk')
```

In [40]:

```
df.head(6)
```

Out[40]:

	序号	姓名	性别	班级	语文	数学	英语	总分
0	1	wong	男	高一一班	89	100	99	288
1	2	li	女	高一一班	72	98	75	245
2	3	xi	女	高一一班	73	89	53	215
3	4	xiong	男	高一一班	84	66	75	225
4	5	li	男	高一一班	99	77	87	263
5	6	xian	女	高一一班	77	65	88	230

In [41]:

```
type(df)
```

Out[41]:

```
pandas.core.frame.DataFrame
```

2 DataFrame

In [42]:

```
# 列名
print(df.columns)

# 索引
print(df.index)
```

```
Index(['序号', '姓名', '性别', '班级', '语文', '数学', '英语', '总分'], dtype='objec
t')
RangeIndex(start=0, stop=42, step=1)
```

In [43]:

```
df.loc[0]
```

Out[43]:

```
序号       1
姓名    wong
性别       男
班级    高一一班
语文      89
数学     100
英语      99
总分     288
Name: 0, dtype: object
```

In [48]:

```
a = np.array(range(10))
a > 4
```

Out[48]:

```
array([False, False, False, False, False,  True,  True,  True,  True,
        True])
```

In [54]:

```
# 筛选数学成绩小于60
df[df.数学 < 60]
```

Out[54]:

	序号	姓名	性别	班级	语文	数学	英语	总分
10	11	chen	男	高一一班	99	55	84	238
12	13	zhneg	女	高一一班	76	44	76	196
27	28	liang	男	高一一班	80	53	99	232
40	41	lai	女	高一一班	77	54	77	208

In [56]:

```
# 复杂筛选（语数外同时大于85）
df[(df.语文 > 85) & (df.数学 > 85) & (df.英语 > 85)]
```

Out[56]:

	序号	姓名	性别	班级	语文	数学	英语	总分
0	1	wong	男	高一一班	89	100	99	288
38	39	kong	男	高一一班	87	89	88	264

3 排序

In [68]:

```
df.sort_values(['数学', '语文']).head(5)
```

Out[68]:

	序号	姓名	性别	班级	语文	数学	英语	总分
12	13	zhneg	女	高一一班	76	44	76	196
27	28	liang	男	高一一班	80	53	99	232
40	41	lai	女	高一一班	77	54	77	208
10	11	chen	男	高一一班	99	55	84	238
17	18	sun	男	高一一班	65	60	98	223

4 访问

In [71]:

```
# 按照索引定位
df.loc[1]
```

Out[71]:

```
序号       2
姓名      li
性别       女
班级    高一一班
语文      72
数学      98
英语      75
总分     245
Name: 1, dtype: object
```

5 索引

In [94]:

```
score = {
    '姓名': ['wang', 'sun', 'li'],
    '数学': [64, 78, 84],
    '英语': [90, 67, 89],
}
f = pd.DataFrame(score, index = ['one', 'two', 'three'])
f
```

Out[94]:

	姓名	数学	英语
one	wang	64	90
two	sun	78	67
three	li	84	89

In [95]:

```
f.index
```

Out[95]:

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

In [96]:

```
1 # 此时不存在数字索引，所以不能通过数字索引访问
2 f.loc[1]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-96-8350517ee18c> in <module>
      1 # 此时不存在数字索引，所以不能通过数字索引访问
----> 2 f.loc[1]

D:\anacanda\anaconda\lib\site-packages\pandas\core\indexing.py in __getitem__(self, key)
   1422
   1423             maybe_callable = com.apply_if_callable(key, self.obj)
-> 1424             return self._getitem_axis(maybe_callable, axis=axis)
   1425
   1426     def _is_scalar_access(self, key: Tuple):

D:\anacanda\anaconda\lib\site-packages\pandas\core\indexing.py in _getitem_axis(self, key, axis)
   1847
   1848         # fall thru to straight lookup
-> 1849         self._validate_key(key, axis)
   1850         return self._get_label(key, axis=axis)
   1851

D:\anacanda\anaconda\lib\site-packages\pandas\core\indexing.py in _validate_key(self, key, axis)
   1723
   1724         if not is_list_like_indexer(key):
-> 1725             self._convert_scalar_indexer(key, axis)
   1726
   1727     def _is_scalar_access(self, key: Tuple):

D:\anacanda\anaconda\lib\site-packages\pandas\core\indexing.py in _convert_scalar_indexer(self, key, axis)
    272         ax = self.obj._get_axis(min(axis, self.ndim - 1))
    273         # a scalar
--> 274         return ax._convert_scalar_indexer(key, kind=self.name)
    275
    276     def _convert_slice_indexer(self, key, axis: int):

D:\anacanda\anaconda\lib\site-packages\pandas\core\indexes\base.py in _convert_scalar_indexer(self, key, kind)
   3136             elif kind in ["loc"] and is_integer(key):
   3137                 if not self.holds_integer():
-> 3138                     return self._invalid_indexer("label", key)
   3139
   3140         return key

D:\anacanda\anaconda\lib\site-packages\pandas\core\indexes\base.py in _invalid_indexer(self, form, key)
   3338             "cannot do {form} indexing on {klass} with these "
   3339             "indexers [{key}] of {kind}".format(
-> 3340                 form=form, klass=type(self), key=key, kind=type(key)
   3341             )
   3342         )
```

```
TypeError: cannot do label indexing on <class 'pandas.core.indexes.base.Index'>
with these indexers [1] of <class 'int'>
```

In [97]:

```
f.loc['one']
```

Out[97]:

```
姓名    wang
数学      64
英语      90
Name: one, dtype: object
```

In [98]:

```
# 不知道索引形式是什么，iloc为当前行数
f.iloc[0]
```

Out[98]:

```
姓名    wang
数学      64
英语      90
Name: one, dtype: object
```

6 ix 索引器已被弃用

In [99]:

```
f.ix[0]
```

```
D:\anacanda\anaconda\lib\site-packages\ipykernel_launcher.py:1: FutureWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ix-indexer-is-deprecated (http://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ix-indexer-is-deprecated)
  """Entry point for launching an IPython kernel.
```

Out[99]:

```
姓名    wang
数学      64
英语      90
Name: one, dtype: object
```

In [100]:

```
f.iloc[:2]
```

Out[100]:

	姓名	数学	英语
one	wang	64	90
two	sun	78	67

In [101]:

```
# 访问某一行时，是错误的
# df.[0]

# 访问多行数据时，时可以使用切片的
f[:2]
```

Out[101]:

	姓名	数学	英语
one	wang	64	90
two	sun	78	67

In [104]:

```
df[:2]
```

Out[104]:

	序号	姓名	性别	班级	语文	数学	英语	总分
0	1	wong	男	高一一班	89	100	99	288
1	2	li	女	高一一班	72	98	75	245

In [108]:

```
# dataframe中的数组
df.数学.values
```

Out[108]:

```
array([100,  98,  89,  66,  77,  65,  87,  78,  98,  88,  55,  77,  44,
        87,  84,  80,  90,  60,  78,  93,  72,  73,  84,  99,  98,  76,
        75,  53,  75,  87,  98,  84,  73,  63,  62,  73,  84,  94,  89,
        77,  54,  65], dtype=int64)
```

In [111]:

```
# 统计每个分数有几人获得
df.数学.value_counts()
```

Out[111]:

```
98     4
84     4
73     3
77     3
87     3
89     2
75     2
78     2
65     2
99     1
53     1
66     1
72     1
60     1
55     1
76     1
54     1
80     1
94     1
44     1
100    1
88     1
62     1
90     1
93     1
63     1
Name: 数学, dtype: int64
```

In [116]:

```
#提取多列
n = df[['语文', '数学']].head()
n
```

Out[116]:

	语文	数学
0	89	100
1	72	98
2	73	89
3	84	66
4	99	77

In [118]:

```
n * 2
```

Out[118]:

	语文	数学
0	178	200
1	144	196
2	146	178
3	168	132
4	198	154

7 重新生成一列表格

In [122]:

```
# map函数
def func(score):
    if score >= 85:
        return "优秀"
    elif score >= 75:
        return "良好"
    elif score >= 60:
        return "及格"
    else:
        return "不及格"


df['数学分类'] = df.数学.map(func)
df.head()
```

Out[122]:

	序号	姓名	性别	班级	语文	数学	英语	总分	数学分类
0	1	wong	男	高一一班	89	100	99	288	优秀
1	2	li	女	高一一班	72	98	75	245	优秀
2	3	xi	女	高一一班	73	89	53	215	优秀
3	4	xiong	男	高一一班	84	66	75	225	及格
4	5	li	男	高一一班	99	77	87	263	良好

In [132]:

```
# applymap 对所有数据进行操作的函数
def func(num):
    return num + 10
# 等价
func = lambda num: num + 10

df.applymap(lambda x: str(x) + ' -').head()
```

Out[132]:

	序号	姓名	性别	班级	语文	数学	英语	总分	数学分类
0	1 -	wong -	男 -	高一一班 -	89 -	100 -	99 -	288 -	优秀 -
1	2 -	li -	女 -	高一一班 -	72 -	98 -	75 -	245 -	优秀 -
2	3 -	xi -	女 -	高一一班 -	73 -	89 -	53 -	215 -	优秀 -
3	4 -	xiong -	男 -	高一一班 -	84 -	66 -	75 -	225 -	及格 -
4	5 -	li -	男 -	高一一班 -	99 -	77 -	87 -	263 -	良好 -

8 匿名函数

In [126]:

```
[i + 100 for i in range(10)]
```

Out[126]:

```
[100, 101, 102, 103, 104, 105, 106, 107, 108, 109]
```

In [127]:

```
def func(x):
    return x + 100
```

In [131]:

```
#list(map(func, range(10)))  与下面等价
list(map(lambda x: x + 100, range(10)))
```

Out[131]:

```
[100, 101, 102, 103, 104, 105, 106, 107, 108, 109]
```

In [152]:

```
# apply函数时根据多列生成新的一列的操作
df['new_score'] = df.apply(lambda x: x.语文 + x.数学, axis = 1)
```

In [154]:

```
# 前面几行
df.head(3)
# 后面几行
df.tail(3)
```

Out[154]:

	序号	姓名	性别	班级	语文	数学	英语	总分	数学分类	new_score
39	40	liang	男	高一一班	66	77	55	198	良好	143
40	41	lai	女	高一一班	77	54	77	208	不及格	131
41	42	luo	男	高一一班	65	65	68	198	及格	130

9 pandas中的dataframe 的操作，跟numpy很二维操作是相似的