

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]:

```
1 import pandas as pd
2 import numpy as np
```

In [103]:

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

In [40]:

```
1 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—班	89	100	99	288
1	2	li	女	高一—班	72	98	75	245
2	3	xi	女	高一—班	73	89	53	215
3	4	xiong	男	高一—班	84	66	75	225
4	5	li	男	高一—班	99	77	87	263
5	6	xian	女	高一—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='object')
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—班	89	100	99	288
38	39	kong	男	高一—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——班	76	44	76	196
	27	28	liang	男	高一——班	80	53	99	232
	40	41	lai	女	高一——班	77	54	77	208
	10	11	chen	男	高一——班	99	55	84	238
	17	18	sun	男	高一——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—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__
    tem__(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
    em_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
    ate_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
    rt_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
    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
    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

Home Page - Select or c x pandas - Jupyter Noteb x 索引和选择数据—Panda x TypeError: unhashable x (1条消息)解决 TypeError x (1条消息)Python初学者之 x + - X

pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ix-indexer-is-deprecated

应用 Gmail 网易邮箱 翻译 GitHub 文章管理-CSDN博客 社交 学习 图书馆远程访问 examples.pdf Python基础课程 (... 万 万门| 我的课程 Running a notebo...

pandas

家 1.0.0的新功能 入门 用户指南 API参考 发展历程 发行说明

Search the docs ...

IO工具 (文本, CSV, HDF5等)

索引和选择数据

多索引/高级索引

合并, 加入和连接

重望和透视表

处理文本数据

处理丢失的数据

分类数据

可为空的整数数据类型

可空布尔数据类型

可视化

计算工具

分组依据: split-apply-combine

时间序列/日期功能

时间增量

造型

选项和设置

提升表现

扩展到大型数据集

稀疏的数据结构

常见问题解答 (FAQ)

菜谱

IX索引器已弃用

警告

在0.20.0开始, .ix索引器已被弃用, 赞成更加严格.iloc 和.loc索引。

.ix在推断用户想要做什么方面提供了很多魔力。也就是说, .ix可以根据索引的数据类型决定是否通过 标签对位置进行索引。多年来, 这引起了相当多的用户混乱。

推荐的索引编制方法是:

- .loc如果要 标记索引。
- .iloc如果要 位置索引。

In [93]: dfd = pd.DataFrame({'A': [1, 2, 3],
.....: 'B': [4, 5, 6]},
.....: index=list('abc'))
.....:

In [94]: dfd
Out[94]:

A B
a 1 4
b 2 5
c 3 6

先前的行为, 您希望从 “A” 列的索引中获取第0个元素和第2个元素。

In [3]: dfd.ix[[0, 2], 'A']
Out[3]:
a 1
c 3
Name: A, dtype: int64

使用 .loc。在这里, 我们将从索引中选择适当的索引, 然后使用 标签索引。

在此页

索引的不同选择

基本

属性访问

切片范围

按标签选择

按位置选择

通过可调用选择

IX索引器已弃用

不建议使用缺少标签的列表建立索引

选择随机样本

放大设置

快速获取和设置标量值

布尔索引

用isin编制索引

where () 方法和遮罩

query () 方法

资料重复

类似于字典的get () 方法

lookup () 方法

索引对象

设置/重置索引

返回视图与副本

localhost:8888/notebooks/pandas.ipynb#

6/11

In [99]:

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

D:\anaconda\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，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—班	89	100	99	288
1	2	li	女	高一—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—班	89	100	99	288	优秀
1	2	li	女	高一—班	72	98	75	245	优秀
2	3	xi	女	高一—班	73	89	53	215	优秀
3	4	xiong	男	高一—班	84	66	75	225	及格
4	5	li	男	高一—班	99	77	87	263	良好

In [132]:

```
# applymap 对所有数据进行操作函数
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—班 -	89 -	100 -	99 -	288 -	优秀 -
1	2 -	li -	女 -	高一—班 -	72 -	98 -	75 -	245 -	优秀 -
2	3 -	xi -	女 -	高一—班 -	73 -	89 -	53 -	215 -	优秀 -
3	4 -	xiong -	男 -	高一—班 -	84 -	66 -	75 -	225 -	及格 -
4	5 -	li -	男 -	高一—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——班	66	77	55	198	良好	143
40	41	lai	女	高一——班	77	54	77	208	不及格	131
41	42	luo	男	高一——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