

## 03Pandas索引

**Pandas** 数据的索引就像一本书的目录，让我们很快地找到想要看的章节，作为大量数据，创建合理的具有业务意义的索引对我们分析数据至关重要。

### 一、认识索引

下图是一个简单的 **DataFrame** 中索引的示例：

**DataFrame object**

	Country	Popu	Percent
IT	Italy	61	0.83
ES	Spain	46	0.63
GR	Greece	11	0.15
FR	France	65	0.88
PO	Portugal	10	0.14

其中：

- 行索引是数据的索引，列索引指向的是一个 **Series**
- **DataFrame** 的索引也是系列形成的 **Series** 的索引
- 建立索引让数据更加直观明确
- 建立索引方便数据处理
- 索引允许重复，但业务上一般不会让它重复

有时一个行和列层级较多的数据会出现多层索引的情况。

### 二、设置索引

之前我们学习了加载数据生成 **DataFrame** 时可以指定索引

```

1 data = 'res/team.xlsx'
2 df = pd.read_excel(data, index_col='name') # 设置索引为 name
3 df
4 '''
5         team  Q1  Q2  Q3  Q4
6 name
7 Liver      E  89  21  24  64
8 Arry       C  36  37  37  57
9 Ack        A  57  60  18  84
10 Eorge      C  93  96  71  78
11 Oah        D  65  49  61  86
12 '''

```

如果加载时没有指定索引，我们可以使用 `df.set_index()` 指定：

### 三、数据查询

#### 1. Series 数据查询

类似Python的字典dict

```

1 s1 = pd.Series([1, 'a', 5.2, 7], index=
    ['a', 'b', 'c', 'd'])
2 s1
3 s1['a']
4 type(s1['a'])
5 s1[['b', 'a']]
6 type(s1[['b', 'a']])

```

#### 2. 从DataFrame中查询Series

- 如果只查询一行、一列，返回的是 `pd.Series`
- 如果查询多行、多列，返回的是 `pd.DataFrame`

```

1 data = 'res/team.xlsx'
2 df = pd.read_excel(data)
3 df
4 #查询一列，得到一个Series
5 df['Q1']
6 type(df['Q1'])
7 #查询多列，得到的是DataFrame
8 df['Q1', 'Q2']
9 type(df['Q1', 'Q2'])
10 #查询一行，得到的是Series
11 df.loc[1]
12 type(df.loc[1])
13 df.loc['Arry']
14 #查询多行，得到的是DataFrame
15 df.loc[1:3]
16 type(df.loc[1:3])

```

### 三、查询方法

Pandas 查询数据的几种方法

1. `df.loc()` 方法, 根据标签值查询
2. `df.iloc()` 方法, 根据数字位置查询
3. `df.where()` 方法
4. `df.query()` 方法

注意:

- 以上查询方法, 既适用于行, 也适用于列
- 注意观察降维 `dataFrame>Series>值`
- `df.loc()` 既能查询, 又能覆盖写入, 强烈推荐!

```
1 # 0.读取数据
2 df = pd.read_csv('res/beijing_tianqi_2018.csv')
3 df.head() #读取前几行, 默认是前5行
4 df.set_index('ymd', inplace=True) # 设定索引为日期, 方便按日期筛选
5 df.index #后续会学习时间序列, 先按字符串处理
6 df.head()
7
8 # 1.使用单个标签查询
9 df.loc['2018-01-03']
10
11 # 2.使用值列表查询
12 df.loc[['2018-01-03', '2018-01-04', '2018-01-05'], 'bwendu'] #得到
    Series
13 df.loc[['2018-01-03', '2018-01-04', '2018-01-05'],
    ['bwendu', 'ywendu']] #得到DataFrame
14
15 # 3.使用数值区间进行范围查询 (注意: 区间左右都包含)
16 df.loc['2018-01-03': '2018-01-08', 'bwendu'] # 行index按区间
17 df.loc['2018-01-03', 'bwendu': 'fengxiang'] # 列index按区间
18 df.loc['2018-01-03': '2018-01-08', 'bwendu': 'fengxiang'] # 行和列
    都按区间查询
19
20 # 4.使用布尔索引
21 df.loc[df['aqi'] < 30] # 简单条件查询, aqi < 30
22 df['aqi'] < 30 # 查看这里的布尔条件
23 # 复杂条件查询, 最高气温小于30度, 最低气温大于15度, 晴天, 空气质量等级一级
24 # 替换掉温度的后缀℃
25 df.loc[:, "bwendu"] = df["bwendu"].str.replace("℃",
    "").astype('int32')
26 df.loc[:, "ywendu"] = df["ywendu"].str.replace("℃",
    "").astype('int32')
27 df.loc[(df["bwendu"] <= 30) & (df["ywendu"] >= 15) &
    (df["tianqi"] == '晴') & (df["aqiLevel"] == 1)]
28
29 # 5.调用函数查询
```

```
30 df.loc[lambda df : (df["bwendu"]<=30) & (df["ywendu"]>=15)] #  
    直接写lambda表达式  
31
```