Pandas

Pandas是数据分析和科学领域最常用的包，以table表格为主要数据类型

- 官方文档： https://pandas.pydata.org/docs/user_guide/index.html#user-guide
- 非官方中文文档：https://www.pypandas.cn/docs/user_guide/

Pandas两种主要的数据类型：

- series：Numpy array + index组成的字典
- dataframe：series 组成的字典

引入pandas 及 numpy 包

请注意下面第一行代码是引入pandas包，并且后续都用pd来做缩写。同样，第二行是引入numpy，然后用np做缩写。

这两个缩写都是相对通用的缩写，尽量不要修改，同时在网上看到pd和np也可以更快的明白是代表着pandas 和 numpy

因为Python及其相关的ecosystem都是在不断发展的，因此有时会出现新的版本更新旧的代码无法正常运行的情况，或者新的版本给出了更简化的代码。所以在网上看到代码时，要注意是否跟你目前的python以及相关包的版本一致。

本教材所讲内容都是基于numpy1.1.3以及pandas1.19.2的版本

```
import pandas as pd
import numpy as np

# 下面两行代码的意思是告诉你，目前所安装的
print(pd.__version__)
print(np.__version__)
```

```
1.1.3
1.19.2
```

用pandas读入文件

pandas可以读取不同'表格'类型的数据，比如，如果需要读入csv文件，那么：

```
pd.read_csv('文件路径及名称包括csv后缀名')
```

其中第一个参数需要用字符串的类型输入文件的路径及名称，包括文件后缀名，在这里也就是 ".csv"

如果需要读入其他数据，在输入，pd.read_，后按键可以看到各种不同的文件的读入，常用的有table, excel等。

读入后，我们可以把读入的内容赋值给一个变量，这个变量的类型就是dataframe

```
# 读入保存在data文件夹里的Body_temp.csv文件
df = pd.read_csv('data/Body_temp.csv')
# 显示df的内容。当dataframe内容过多，也就是栏或者行过多时，往往中间的部分不会显示
print(df)
```

```
    日期    体温
0    1  36.0
1    2  36.2
2    3  36.3
3    4  36.8
4    5  36.2
5    6  36.1
6    7  36.0
7    8  36.2
8    9  37.1
9   10  36.4
10  11  36.6
11  12  36.1
12  13  36.3
13  14  36.2
```

pandas里的数据显示问题

1. 最左边一列，从0-13的数字是在读取时默认添加的index，相当于行的序号
2. 最上面一行，‘日期 体温’是原csv数据中的第一行，被默认作为每一列的名字，也就是 列名。
3. 需要注意的是，如果我们的csv没有标题行，也就是说，第一行就是数据，"1 36.0"那么，1和36.0就会被认为是 列名。相当于数据少了一行
4. 如果需要额外添加列名，需要在读取时加入参数，names

建议仔细对比添加**names**参数前后，同样数据的差异

```
# 在读取时加入参数，names，额外赋予列名
df1 = pd.read_csv('data/Body_temp.csv', names=['1', '2'])
# 请注意，这时，csv原有的第一行就变成了数据的一部分
print(df1)
```

```
     1     2
0   日期    体温
1    1    36
2    2  36.2
3    3  36.3
4    4  36.8
5    5  36.2
6    6  36.1
7    7    36
8    8  36.2
9    9  37.1
10  10  36.4
11  11  36.6
12  12  36.1
13  13  36.3
14  14  36.2
```

5. 当dataframe内容过多，也就是栏或者行过多时，往往中间的部分不会显示
6. 如果想修改pandas默认显示的行数，可以用options修改.比如下面代码就是修改为显示999行

```
pd.options.display.max_rows = 999
```

选取**dataframe**里面的一列

选取一列最简单的方法是：

```
df[列名]
```

其中，列名需要用字符串的类型传递

```
print(df['日期'])
```

```
0      1
1      2
2      3
3      4
4      5
5      6
6      7
7      8
8      9
9     10
10    11
11    12
12    13
13    14
Name: 日期, dtype: int64
```

用**df.loc & df.iloc**取多行、多列、或某个单元格

两个方法用法相似，不同的是，.loc后面需要添加具体的行或者列的名字（label）,而.iloc后面需要添加的是行或者列的位置（index），也就是0，1，2，3，。。。。这些数字

```
df.loc[,]
```

需要注意的是，因为两个方法的目的是取值，不是调用方法的概念，所以，后面跟着中括号，而不是小括号。中括号钟以逗号“，”分隔,前面为指定的行，后面为指定的列。只取一行或一列的时候，直接输入一个值；需取多行或多列时，需要用list的类型传递。如果需要取全部的行或列，用 冒号表示“:”

具体用法：

```
df.loc[此处为行的名字label（具体类型以label自身的类型为准）,此处为列的名字label（具体类型以label自身的类型为准）]
df.iloc[,]
```

请注意，在我们的例子中，行的label刚好跟其位置一致，都是0，1，2，3...因此在loc和iloc使用无差别。但是列的名字是 日期和体温，但位置则为0，1

```
df.loc[0,'体温']
```

```
36.0
```

```
df.loc[:,'体温']
```

```
0     36.0
1     36.2
2     36.3
3     36.8
4     36.2
5     36.1
6     36.0
7     36.2
8     37.1
9     36.4
10    36.6
11    36.1
12    36.3
13    36.2
Name: 体温, dtype: float64
```

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

```
日期      1.0
体温     36.0
Name: 0, dtype: float64
```

获取到的值的数据类型

1. 当获取的是整行或整列，通常数据类型为series
2. 获取的是多行多列，数据类型往往为dataframe
3. 只获取某一个单元格的值，数据类型往往为该单元格所对应的值的实际类型
4. 强烈建议取值之后用type看一下

```
type(df.iloc[0,:])
```

```
pandas.core.series.Series
```

该如何获得具体的数值？

如上所述，取值后通常我们得到的数据类型是pandas里面的df或者series数据类型。但往往我们还需要针对里面的具体数值进行计算或进一步处理。比如，我们想选取所有的体温，求平均值该如何做？

其中一个非常重要的获取所对应的“数值”的方式是在取值后面加 “.values”， 注意因为这个values是获取属性，不是方法，因此后面没有括号

```
print(df['体温'].values)
print(type(df['体温'].values))
```

```
[36.  36.2 36.3 36.8 36.2 36.1 36.  36.2 37.1 36.4 36.6 36.1 36.3 36.2]
<class 'numpy.ndarray'>
```

通过上面代码可以看到，获得是一组数值，也就是体温这列里面所有的数值，这一组数值的数据类型是numpy里面的arrary。

numpy array

这个array跟list几乎一样，比如index,slice的方式都一样。

但是array功能更为强大。最方便的地方在于，支持大量的数学运算，比如，对全部的值乘二, 及求平均值

更多的就请用键自己探索吧

```
temps = df['体温'].values
print(temps*2)
print(temps)
```

```
[72.  72.4 72.6 73.6 72.4 72.2 72.  72.4 74.2 72.8 73.2 72.2 72.6 72.4]
[36.  36.2 36.3 36.8 36.2 36.1 36.  36.2 37.1 36.4 36.6 36.1 36.3 36.2]
```

```
temps[0]
```

```
36.0
```

```
temps.mean()
```

```
36.32142857142858
```

```
```