

Pandas 复习

【导入】

```
import pandas as pd
```

【两种数据结构】



一、Series

【概念】

一个一维数据结构，由两列数据组成，一列为索引，一列于索引关联的值

【创建】

```
s1=pd.Series([80,90])
s2=pd.Series({"张三":80,"李四":90})
```

0	80
1	90

s1

张三	80
李四	90

s2

【常用属性】

index: Series 的下标索引，其值默认是从 0 起递增的整数（未指定索引时）

values: 存储 Series 值的一个数组

```
s1.index: [0,1]
```

```
s2.index: ["张三","李四"]
```

```
s1.values: [80,90]
```

```
s2.values: [80,90]
```

【操作】

```
a=s1[0] #读取对象 s1 索引为 1 的元素值 80 赋值给变量 a,即 a=80
s1[0]=100 #将对象 s1 索引为 0 的元素值修改为 100
s2["李四"]=95 #将对象 s2 索引为"李四"的元素值修改为 95
s2[1]=95 #将对象 s2 索引为 1 ("李四") 的元素值修改为 95
s2["王五"]=80 #在对象 s2 增加元素，索引为"王五"，元素值为 95，即索引不存在会增加
```

二、DataFrame

【概念】

是一个二维数据结构，由多列数据组成，一列为索引，若干列于索引关联的值

【创建】

字典数据

列表数据

```
data={"姓名":["张三","李四","王五"],
      "班级":["1班","1班","2班"],
      "信息":[45,38,47],
      "通用":[39,42,39]}
df=pd.DataFrame(data)
```

```
data=[["张三","1班",45,39],
      ["李四","1班",38,42],
      ["王五","2班",47,39]]
df=pd.DataFrame(data,columns=["姓名","班级","信息","通用"])
```

```
df=pd.read_excel("student.xlsx")
```

```
df=pd.read_csv("student.csv")
```

	A	B	C	D
1	姓名	班级	信息	通用
2	张三	1班	45	39
3	李四	1班	38	42
4	王五	2班	47	39

文件(F)	编辑(E)	格式(O)	查看(V)
姓名,班级,信息,通用			
张三,1班,45,39			
李四,1班,38,42			
王五,2班,47,39			

以上 4 种方式创建 DataFrame 对象 df 如右图 a:

【常用属性】

index: DataFrame 的下标索引, 即行索引, 其值默认是从 0 起递增 (+1) 的整数

values: 存放值的二维数据

columns: 存放各列的列标题的一维列表

T: 行列倒置

df.index: [0,1,2] #行索引

df.values: [['张三', '1 班', 45, 39],['李四', '1 班', 38, 42],['王五', '2 班', 47, 39]] #二维列表

df.columns: ['姓名', '班级', '信息', '通用'] #各列的列标题一维列表

df.T: 如右图 b 所示,此时图 a 中索引变为图 b 的列标题, 图 a 中列标题变为图 b 中的索引

	姓名	班级	信息	通用
0	张三	1班	45	39
1	李四	1班	38	42
2	王五	2班	47	39

图 a

	0	1	2
姓名	张三	李四	王五
班级	1班	1班	2班
信息	45	38	47
通用	39	42	39

图 b

【常见操作】

❖ 修改

如将图 a 中李四的信息修改为 35: `df.at[1,'信息']=35`

如将图 a 中所有人的信息成绩加 2 分: `df['信息']=df['信息']+2`

`rename()` 修改列名或者索引

`df=df.rename({0:100})` #将索引 0 改为 100

`df=df.rename(columns={'姓名':'学生名字'})` #原列名为“姓名”更改为“学生名字”

❖ 增加

在最后增加 1 列, 列标题为“技术”, 值为信息加通用成绩: `df['技术']=df['信息']+df['通用']`

`insert()` 在指定的位置插入位置

`df.insert(0,"性别",['男','女','男'])` #在最前面增加一列, 列名为性别

`append()` 在指定的元素的结尾插入内容, 可实现追加数据行的功能

例如: `df.append(df1)` #将两个 DataFrame 对象合并

❖ 删除

使用 `drop` 方法删除, 可以删除列和行数据, `drop` 不改变原对象的数据。

删除班级这一列: `df=df.drop('班级',axis=1)` #将改变的数据覆盖原对象的数据

删除王五所在的行数据: `df=df.drop(2)` 或 `df1=df.drop(2,axis=0)` #不改变原对象的数据, 删除后的数据存放在 `df1` 对象中。

使用 `del` 方法删除, 直接在原对象中删除, 如: `del df['班级']`

❖ 读取

读取一列: `df['列名']` 如读取姓名列: `df['姓名']` #结果是 Series 数据结构, 如图 c

读取某一具体值: `df.at[行索引,'列名']` 如读取李四的信息成绩: `df.at[1,'信息']`

读取多行

`df.head(2)` #读取前 2 行, 注 `df.head()` 表示读取前 5 行

`df.tail(2)` #读取后 2 行, 注 `df.tail()` 表示读取后 5 行

`df[0:2]` #读取前 2 行, 教材中未出现, 简单了解即可

`df.loc[0:1]` #读取前 2 行, 教材中未出现, 简单了解即可

上面 4 行代码得到的结果都是 DataFrame 对象, 如图 d

0	张三
1	李四
2	王五

图 c

	姓名	班级	信息	通用
0	张三	1班	45	39
1	李四	1班	38	42

图 d

❖ 筛选

对象名[**条件表达式**], 这里的条件表达式一般是针对某一列的值做比较关系

例如: 显示图 a 中 1 班的数据

条件表达式: `df['班级']=='1 班'`

完整的语句: `df[df['班级']=='1 班']` #得到 1 班同学的数据

例如：显示图 a 中信息大于等于 40 的学生：df[df['信息']>=40]

❖ 排序

对象名.sort_values('列名',ascending=True/False),参数 ascending 值：True 表示升序，False 表示降序

df.sort_values('信息',ascending=False) #按信息列的值由低到高进行降序排列

df.sort_values('信息',ascending=True) #按信息列的值由低到高进行升序排列

df.sort_values('信息') #按信息列的值由低到高进行升序排列

注：一般会与 head 或 tail 一起使用，sort_values 不改变原对象中的数据，所以一般要存为新对象

df_sort=df.sort_values('信息',ascending=False).head(2) #df_sort 对象存储信息前 2 名的数据

❖ 统计

求和 sum()

df.sum() 或 df.sum(axis=0) #对 df 所有列进行按列求和，如图 e

df['信息'].sum() 或 df['信息'].sum(axis=0) #只对信息列进行按列求和，如图 f

df.sum(axis=1) #对 df 所有列进行按行求和，如图 g

求平均值 mean() 用法可参考 sum

df.mean() 或 df.mean(axis=0) #对 df 所有列进行按列求平均值

df['信息'].mean() 或 df['信息'].mean(axis=0) #只对信息列进行按列求平均值

最大值 max() 用法可参考 sum

df.max() 或 df.max(axis=0) #对 df 所有列进行按列求最大值

df['信息'].max() 或 df['信息'].max(axis=0) #只对信息列进行按列求最大值

最小值 min() 用法可参考 sum

df.min() 或 df.min(axis=0) #对 df 所有列进行按列求最小值

df['信息'].min() 或 df['信息'].min(axis=0) #只对信息列进行按列求最小值

统计非空数据项数 count()

df.count() 或 df.count(axis=0) #对 df 所有列进行按列统计非空数据项个数如图 h

df['信息'].count() 或 df['信息'].count(axis=0) #只对信息列进行按列统计非空数据项个数

姓名	张三李四王五
班级	1班1班2班
信息	130
通用	120

图 e

130

图 f

0	84
1	80
2	86

图 g

姓名	3
班级	3
信息	3
通用	3

图 h

分组 groupby()

对各列或各行中的数据进行分组，然后可对其中一组数据进行不同的操作，一般需与 sum、mean、max、min、count 配合使用

使用方法：对象名.groupby("分组列名",as_index=True/False) #此语句仅仅是进行分组，并未统计或计算，参数 as_index 的值 True:分组列做为结果的索引列 False:分组列不做为索引列，索引列自动生成，由 0 开始递增

对所有数值列进行分组求平均值

df.groupby("班级").mean() #对图 a 数据计算信息与通用平均分，如图 I

df.groupby("班级",as_index=True).mean() #对图 a 数据计算信息与通用平均分，如图 I

df.groupby("班级",as_index=False).mean() #对图 a 数据计算信息与通用平均分，如图 J

df.groupby("班级").count() #统计图 a 数据各个班级非空数据项个数，如图 M

对指定列进行分组统计、计算

df.groupby("班级")["信息"].mean() #对图 a 数据计算每个班级信息平均分，如图 L

df.groupby("班级",as_index=True)["信息"].mean() #对图 a 数据计算每个班级信息平均分，如图 L

df.groupby("班级",as_index=False)["信息"].mean() #对图 a 数据计算每个班级信息平均分,如图 K
 注: **groupby** 与 **sort_values** 一样不改变原对象中的数据,所以一般要存为新对象,另 **groupby** 与 **sort_values** 可以分开写。

例: df.groupby("班级",as_index=False)["信息"].mean() 可以写成以下 2 种代码

<pre>g=df.groupby("班级",as_index=False) df1=g["信息"].mean()</pre>	<pre>g=df.groupby("班级",as_index=False)["信息"] df1=g.mean()</pre>
---	---

信息 通用		
1班	41.5	40.5
2班	47.0	39.0

班级	信息	通用
0	1班	41.5 40.5
1	2班	47.0 39.0

班级	信息
0	1班 41.5
1	2班 47.0

班级	信息
1班	41.5
2班	47.0

姓名	信息	通用
1班	2	2 2
2班	1	1 1

图 I 图 J 图 K 图 L 图 M

三、其它

【describe】

返回各列的基本描述统计值,包含计数、平均数、最大值、最小值、标准差及 4 分位差

【不改变原来对象】

drop、append、sort_values、groupby、rename、insert、sum、mean、max、min、count 等以上操作不改变原来对象,而是通过返回另一个对象来存放改变后的数据,所以一般书写的语句为:

df_new=df.drop(0,axis) 或 df_new=df.sum()

如需覆盖原对象,应书写为: df=df.drop(0,axis)

【括号】

什么情况需要使用[]和()

[]使用情况:

- 1、df.at[行索引,'列名'] #读取或修改某一具体值时
- 2、df['列名'] #读取某一列的值时,包含分组中对某一列进行统计 df.groupby("班级")["信息"].mean()
- 3、采用[]读取列数据时,不使用“.”,df["班级"]是**错误**书写,正确书写应为 df["班级"]
- 4、df[0:5] 或 df.loc[0:4] #读取行数据时,这个教材中未出现
- 5、df[df['班级']=='1班'] #筛选,此时可看来是读取某列的值,参考第 2 点
- 6、Series 与 DataFrame 创建时,所用列表数据与列名,参考前面的创建
- 7、除上述情况外,均使用()
- 8、在()里的列名除需要多列时使用列表[],其余均使用'列名',注意列名需加单引号或双引号

df.groupby("班级",as_index=False)

df.sort_values('信息')

df.drop('班级',axis=1)

...

【sum 等函数使用】

sum、mean、count、min、max 函数使用方法相同,在有 3 种使用方式: ()、(axis=0)、(axis=1)

例 1: 统计 df 对象(图 a)中所有人的信息平均分

df["信息"].mean() 或 df["信息"].mean(axis=0) #这里 axis=0 表示按列进行计算

错误书写: df.mean("信息")、df.mean("信息",axis=0)

例 2: 统计 df 对象(图 a)中所有人的信息+通用的总分

df.sum(axis=1) #这里 axis=1 表示按行进行计算

注: 求平均值,在 pandas 使用 mean 函数,有 Excel 使用 average