第四章 结构化数据处理

第一部分：Excel 基本操作

一、公式的编辑

1.**公式必须等号“=”开头**，乘号为“*”，除号为“/”，括号为“()”。乘号不能省略，不能用中括号[]。比如公式“=((A2+B2)/2)-5”，不能写成“=[(A2-B2)/2]-5”。**如果单元格格式被设置成“文本”，输入公式会无效（被当做文本字符）。**

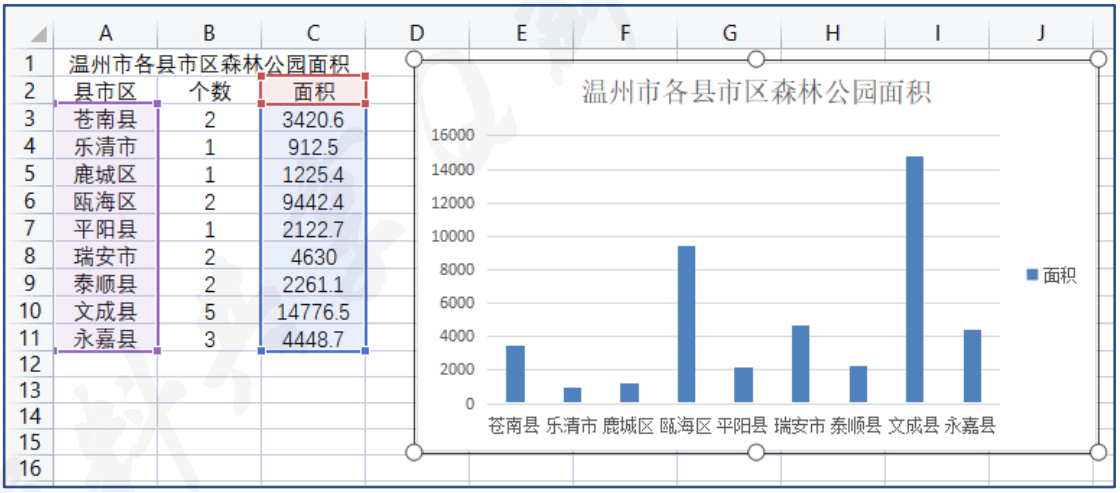
2.Excel 常用的函数有 Sum(数据区域)、Average(数据区域)、Max(数据区域)、Min(数据区域)等。**多个不连续区域中间用逗号分隔**，比如“=Sum(B2:D2,F2)”。Excel 中函数不区分大小写。

3.**编辑的公式引用的空白单元格，空白单元格当作 0 处理**。比如公式“=A2/B2”，如果 B2 为空白单元格，此时除数为 0，会出现“#DIV/0!”错误提示。而函数计算时会自动忽略非数字类型单元格和空白单元格。

4.**公式相对引用**：在单元格 C2 中输入公式“=A2*5+B2”，将 C2 中内容复制粘贴到 E3 单元格后，公式会自动变成“=C3*5+D3”。因为单元格 E3 相对 C2，增加 2 列，增加 1 行。公式中的行号和列号会分别自动增 2 和增 1,即“=A2*5+B2”变成“=C3*5+D3”。

5.**行绝对引用和列绝对引用**。单元格中输入公式“=B2/F2"，自动填充到下一个单元格，下一个单元格中公式为“=B3/F3”(行号自动加 1)：如果不想让某个行号发生变化，则需要在行号前加绝对引用符号$，如“=B2/F$2”、下个单元格公式则为“=B3/F$2”。同理自动填充到右边单元格，列号会自动递增公式由“=B2/F2”变为“=C2/G2”(列号递增)。如果不想让某个列号发生变化，则需要在列号前加绝对引用符号$，比如“=B2/$F2”，右单元格公式则为“=C2/$F2”。

二、图表的绘制



1.图表数据区域的解题方法：

(1)根据 X 轴，确定数据区域 A3:A11

(2)根据 Y 轴，确定数据区域 C3:C11

(3)根据图例，确定标题区域 C2

(4)最后根据高平齐长对正，确定 A2

最终确定数据区域为 A2:A11,C2:C11，**图表标题与选区无关**

2.图表和数据区域中内容相互关联：改变区域 A2:A11,C2:C11 中的某个单元格内的内容，图表内容也会自动更新。图表数据区域内的数据排序后，图表也会发生变化，筛选后，不符合筛选结果的内容不会在图表上显示。在图表数据区域外操作，图表不会发生变化。

第二部分：pandas 中的 Series 和 DataFrame

一、一维结构 Series

Series 是一种一维的数据结构，包含一个数组的数据和一个与数据关联的索引（index）,索引默认是从 0 递增的整数。列表、字段等可以来创建 Serise 数据结构，与列表不同的是，Series 的索引可以指定，类型可以为字符串类型。

1.创建一个 Series 对象

```
import pandas as pd
s1 = pd.Series([166,178,180])
print(s1)
```

```
0    166
1    178
2    180
dtype: int64
```

```
s2 = pd.Series([166,178,180],index=["s1","s2","s3"])
print(s2)
```

```
s2 = pd.Series({"s1":166,"s2":178,"s3":180})
print(s2)
```

```
s01    166
s02    178
s03    180
dtype: int64
```

2.Series 对象常用属性

`print(s2.index)`	`Index(['s01','s02','s03'], dtype='object')`
`print(s2.values)`	`[166 178 180]`

【注意】通过循环遍历查看 s2 时，默认输出 s2.values

3.取值或修改 Series 对象

`print(s2["s01"])`	`166`
`print(s2[0])`	

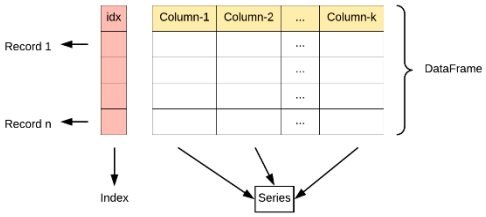
【注意】s2 的索引值创建时被指定为字符串，所以可以使用指定索引访问，但此时默认索引依然可以使用。

```
s2[1] = 174
#s2["s02"]=174 等价
print(s1)
```

```
s01    166
s02    174
s03    180
dtype: int64
```

二、二维结构 DataFrame

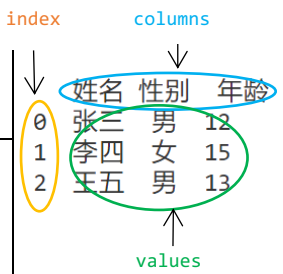
DataFrame 是一种二维结构，由一个索引列（index）和若干个数据列组成，每个数据列可以是不同类型。DataFrame 可以看作是共享同一个 index 的 Series 的集合。



1.创建 DataFrame 对象

```
import pandas as pd
list1 = [["张三","男",12],
         ["李四","女",15],
         ["王五","男",13]]
df1 = pd.DataFrame(list1,columns=["姓名","性别","年龄"])
```

```
dict1 = {"姓名":["张三","李四","王五"],
         "年龄":[12,15,13],
         "性别":["男","女","男"]}
df1 = pd.DataFrame(dict1,columns=["姓名","性别","年龄"])
print(df1)
```



【注意】上面的案例中使用了两种方式创建 DataFrame 对象。用列表创建时，二维列表中的每一个一维列表会成为 DataFrame 中的一列；用字典创建时，“键”会作为 columns。DataFrame 函数的 columns 参数可以设置列名称或设定列顺序。由于没有设定 index，df1 使用默认索引。

2.DataFrame 对象常用属性

`for i in df1.index:`
`  print(i)`
运行结果：
`0`
`1`
`2`
	`for i in df1.columns:`
	`  print(i)`
	运行结果：
	`姓名`
	`性别`
	`年龄`
		`for i in df1.values:`
		`  print(i)`
		运行结果：
		`['张三' '男' 12]`
		`['李四' '女' 15]`
		`['王五' '男' 13]`
			`print(df1.T)`
			`#转置行、列`
			运行结果：
			`  0 1 2`
			`姓名 张三 李四 王五`
			`性别 男 女 男`
			`年龄 12 15 13`

3.DataFrame 对象的取值和修改

(1)检索“姓名”列数据

`print(df1.姓名)  #通过属性检索`
`print(df1["姓名"])  #通过字典记法检索`
	`0 张三`
	`1 李四`
	`2 王五`
	`Name: 姓名, dtype: object`

DataFrame 可以通过两种方式检索单列，结果相同。需要注意的是，对二维 DataFrame 对象取单列后，得到的是一个一维 Series 结构。其次，当列名为数值类型，属性检索不可用。

(2)同时检索多列

`print(df1[["年龄","姓名"]])`	`  年龄 姓名`
	`0 12 张三`
	`1 15 李四`
	`2 13 王五`

(3)修改“年龄”列数据

`df1.年龄 = [18,22,20]`
`print(df1)`
	`  姓名 性别 年龄`
	`0 张三 男 18`
	`1 李四 女 22`
	`2 王五 男 20`

(4)检索单行或多行数据

`print(df1[0:1])`	`  姓名 性别 年龄` `0 张三 男 12`

`print(df1[1:3])`	`  姓名 性别 年龄`
	`1 李四 女 15`
	`2 王五 男 13`

【可通过这种方式修改单行数据，略】

(5)通过条件筛选满足条件的行

`print(df1[df1["年龄"]>13])` `print(df1[df1.年龄>13])`	`  姓名 性别 年龄` `1 李四 女 15`

(6)获取指定行列的值

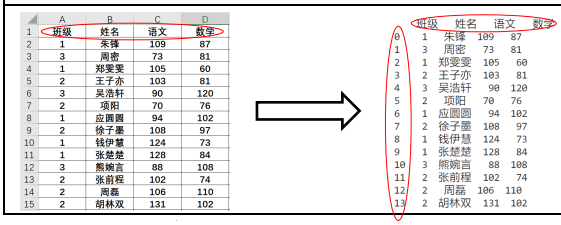
`print(df1.at[0,"姓名"])  #at[]方法获取` `print(df1["姓名"][0])  #转一维后通过索引值获取`	张三

【可通过这种方式修改单个数据，略】

4.DataFrame 常用函数

从 Excel 或 csv 文件中读取数据

```
df2 = pd.read_excel("成绩单.xlsx")
print(df2)
```



	班级	姓名	语文	数学
1	1	朱锋	109	87
2	3	周密	73	81
3	1	郑雯雯	105	60
4	2	王子亦	103	81
5	3	吴浩轩	90	120
6	2	项阳	70	76
7	1	应圆圆	94	102
8	2	徐子墨	108	97
9	1	钱伊慧	124	73
10	1	张楚楚	128	84
11	3	熊婉言	88	108
12	2	张前程	102	74
13	2	周磊	106	110
14	2	胡林双	131	102

```
   班级   姓名   语文  数学
0   1   朱锋  109   87
1   3   周密   73   81
2   1  郑雯雯  105   60
3   2  王子亦  103   81
4   3  吴浩轩   90  120
5   2   项阳   70   76
6   1  应圆圆   94  102
7   2  徐子墨  108   97
8   1  钱伊慧  124   73
9   1  张楚楚  128   84
10  3  熊婉言   88  108
11  2  张前程  102   74
12  2   周磊  106  110
13  2  胡林双  131  102
```

Excel 转 DataFrame 时，第一行成为 columns，其他行成为 values。index 是转换时自动生成的。

(1)数据统计

函数	功能
sum(axis=0/1)	求和
count(axis=0/1)	统计非空(NaN)数据项个数
mean(axis=0/1)	求平均值
max(axis=0/1)、min(axis=0/1)	求最大值，最小值
describe()	返回各列基本描述统计值

axis 参数用于确定行/列，axis=0（默认）对行统计，axis=1 对列统计。

```
print(df2.姓名.count())          #求总人数
print(df2[0:1].sum(axis=1))      #求朱峰总分
```

```
14
0    197
dtype: int64
```

(2)head(n)、tail(n)返回前 n 行、后 n 行数据，n 默认为 5

```
print(df2.tail())       #返回最后 5 行
```

```
    班级   姓名   语文   数学
9    1  张楚楚  128   84
10   3  熊婉言   88  108
11   2  张前程  102   74
12   2   周磊  106  110
13   2  胡林双  131  102
```

(3)groupby(as_index)分类汇总

```
#统计各个班级各科平均分
print(df2.groupby("班级").sum())
```

```
          语文      数学
班级
1   112.000000   81.2
2   103.333333   90.0
3    83.666667  103.0
```

```
#设置 as_index=False，使“班级“不成为索引
print(df2.groupby("班级",as_index=False).mean())
```

```
   班级          语文      数学
0   1  112.000000   81.2
1   2  103.333333   90.0
2   3   83.666667  103.0
```

(4)sort_values(ascending,inplace)排序

```
#根据班级降序排序，ascending 默认为 True,升序
print(df2.sort_values("班级",ascending=False))

#1.排序时索引会跟随数据排序
#2.对 Series 对象排序时不需要写列名
```

```
    班级   姓名   语文   数学
1    3   周密   73   81
4    3  吴浩轩   90  120
10   3  熊婉言   88  108
3    2  王子亦  103   81
5    2   项阳   70   76
7    2  徐子墨  108   97
11   2  张前程  102   74
12   2   周磊  106  110
13   2  胡林双  131  102
0    1   朱锋  109   87
2    1  郑雯雯  105   60
6    1  应圆圆   94  102
8    1  钱伊慧  124   73
9    1  张楚楚  128   84
```

```
#根据语文成绩升序排序，并用结果替换原对象
df2.sort_values("语文",inplace=True)
print(df2)
```

```
    班级   姓名   语文   数学
5    2   项阳   70   76
1    3   周密   73   81
10   3  熊婉言   88  108
4    3  吴浩轩   90  120
6    1  应圆圆   94  102
11   2  张前程  102   74
3    2  王子亦  103   81
2    1  郑雯雯  105   60
12   2   周磊  106  110
7    2  徐子墨  108   97
0    1   朱锋  109   87
8    1  钱伊慧  124   73
9    1  张楚楚  128   84
13   2  胡林双  131  102
```

(5)drop(axis)删除

```
print(df2.drop("班级",axis=1))       #删除班级列
print(df2.drop(columns="班级"))
```

```
    姓名   语文   数学
0   朱锋  109   87
1   周密   73   81
2  郑雯雯  105   60
3  王子亦  103   81
```

```
print(df2.drop(0))
print(df2.drop(index=0))
```

```
    班级   姓名   语文   数学
1    3   周密   73   81
2    1  郑雯雯  105   60
3    2  王子亦  103   81
4    3  吴浩轩   90  120
```

【注意】

1.drop()的 axis 与前面数据统计行列使用方式相反，当要删除列时，axis 需要特别设置为 1，否则默认 0 删除的是行。

2.drop()删除行时，索引也会一起删除，不会向前自动补全。

3.除了 drop()外，还可以使用 del 保留字执行删除，`del df2.班级` 可永久删除“班级”列。

(6)append(ignore_index)追加新数据

```
#追加新数据行，并保持索引连续
dict2 = {"班级":4,"姓名":"李四"}
print(df2.append(dict2,ignore_index=True))
```

```
9    1  张定定  128.0   84.0
10   3  熊婉言   88.0  108.0
11   2  张前程  102.0   74.0
12   2   周磊  106.0  110.0
13   2  胡林双  131.0  102.0
14   4   李四    NaN    NaN
```

```
#追加新表，ignore_index 默认为 False
dict2 = {"班级":4,"姓名":"李四"}
df3 = pd.DataFrame(dict2,index=[0])
print(df2.append(df3))
```

```
6    1  应圆圆   94.0  102.0
7    2  徐子墨  108.0   97.0
8    1  钱伊慧  124.0   73.0
9    1  张楚楚  128.0   84.0
10   3  熊婉言   88.0  108.0
11   2  张前程  102.0   74.0
12   2   周磊  106.0  110.0
13   2  胡林双  131.0  102.0
0    4   李四    NaN    NaN
```

(7)insert(loc,column,value)插入新列

```
#在第二列插入“年级”，并统一设置为 2
df2.insert(2,"年级",2)
print(df2)
```

```
   班级   姓名  年级   语文   数学
0   1   朱锋   2  109   87
1   3   周密   2   73   81
2   1  郑雯雯   2  105   60
3   2  王子亦   2  103   81
4   3  吴浩轩   2   90  120
5   2   项阳   2   70   76
```

(8) rename(index,columns)修改行、列名称

```
print(df2.rename(index={0:"s01",1:"s02"},
                 columns={"姓名":"xingming"}))
```

```
     班级 xingming   语文   数学
s01   1       朱锋  109   87
s02   3       周密   73   81
2     1      郑雯雯  105   60
```

(9)concat([合并的表],axis,ignore_index)多张表合并

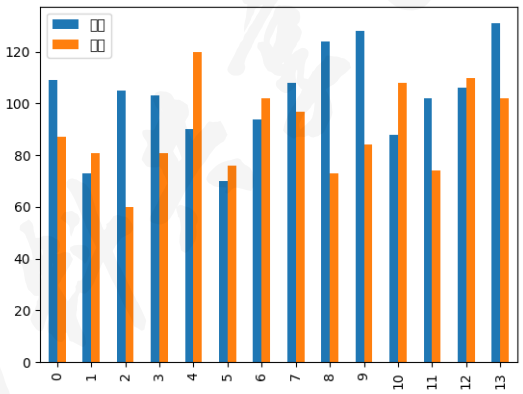
```
df4 = pd.concat([df2,df3],ignore_index=True)
print(df4)
```

```
12   2   周磊  106.0  110.0
13   2  胡林双  131.0  102.0
14   4   李四    NaN    NaN
```

【注意】这里演示的结果与 append()类似，但 concat(axis=1)还可以实现两表横向连接，ignore_index 与 append()中的定义相同。

(10)plot()绘图

```
import matplotlib.pyplot as plt
df2.drop(["班级","姓名"],axis=1).plot(kind="bar")     #用语文、数学成绩绘图
plt.show()
```



通过观察结果可以发现 plot 方法生成图表的逻辑：x 轴为 index 的值，生成图的只有数值列的数据。

这里需要指出 plot()方法默认生成的是折线图，代码中通过 kind=bar 改为柱状图。其他类型图表：line：折线图；barh：横向柱状图；scatter：散点图

例题：以下是某学校一次测试成绩，现需要对表格内的数据进行整理，请根据注释补全程序。

班级	姓名	语文	数学	英语	日语	悟理	化学	生物	政治	历史	地理	信息	通用
4	朱锋	109	87	128					48	100	48		
3	周密	73	81	91		86		97			98		
1	郑雯雯	105	60		125	54	58	47					
5	王子亦	103	81	122			48	74				17	20
10	吴浩轩	90	120	134					64		79	32	39
5	项阳	70	76	123			43	85				40	31
4	应圆圆	94	102	131					99	48	54		
8	徐子墨	108	97	112		46					73	26	17
3	钱伊慧	124	73		97	71		43			72		

```
import pandas as pd
import matplotlib.pyplot as plt

#整理数据
df = pd.read_excel("成绩.xlsx")                    #打开 excel 文件
df.fillna(0,inplace=True)                          #空单元格(NaN)填充 0
df = df.rename(columns={"悟理":"物理"})             #修改"悟理"为"物理"
df.at[1,"数学"] = 123                              #修改“周密”的数学成绩为 123
#插入外语列，数据为英语和日语列的合并
df.insert(4,"外语",df["英语"] + df["日语"])
df["技术"] = df[["信息","通用"]].sum(axis=1)    #创建技术列，数据为信息+通用
df = df.drop(["信息","通用","英语","日语"],axis=1)      #删除多余列
df["总分"] = df.drop(["班级","姓名"],axis=1).sum(axis=1)  #计算总分

#计算各班总分平均分
df_g = df.groupby("班级",as_index=False)["总分"].mean()
df_sort = df_g.sort_values("总分",ascending=False)     #对总分进行降序排序
print(df_sort.head(3))    #输出总分前三名

#统计优秀学生个数
df_1 = df[df.总分>600]    #筛选总分大于 600 的同学
print(df_1.姓名.count())  #统计大于 600 分的学生人数

#绘图
df_g.plot(x="班级",y="总分",kind="bar")    #用分类汇总结果绘制柱状图
plt.show()    #显示图像
```