Pandas 编程处理数据

	A	B	C	D	E	F	G	H
1	班级	学号	姓名	性别	团员	语文	数学	英语
2	1班	104	阿地力	男	是	132	124	140
3	1班	107	伊力亚斯	男	否	128	86	112
4	1班	115	卢静	女	是	139	88	103
5	1班	119	古丽米热	女	否	94	139	116
6	1班	140	阿依古再丽	女	是	114	86	117
7	2班	204	张丞国	男	否	147	143	149
8	2班	205	萨迪克江	男	是	123	124	149
9	2班	216	阿比旦	女	否	99	80	94
10	2班	220	阿依祖克热	女	是	120	106	138
11	2班	229	穆馨春	女	否	127	124	145
12	3班	311	也斯哈提	男	否	100	89	142
13	3班	316	米浩然	男	是	99	133	127
14	3班	320	依力哈木江	男	是	139	106	109
15	3班	326	热伊麦	女	否	134	108	134
16	3班	334	艾力皮也	女	是	149	112	135
17	4班	404	伊合巴力	男	否	100	127	117
18	4班	406	田坤	男	否	147	137	146
19	4班	407	依力格尔	男	是	106	128	104
20	4班	411	马硕宇	男	否	125	132	103
21	4班	436	吴灿	女	是	148	83	96

登分表.xlsx 的数据内容截图

1.导入 pandas 并设置其别名

```
import pandas as pd # pd 是 pandas 的别名，在程序中可以用 pd 代替 pandas
```

2.读取 Excel 文件

```
# 通过 pd 上的 read_excel 函数读取 excel 文件
# read_excel 函数执行的结果是一个表格，这个表格赋值给了 df
df = pd.read_excel("登分表.xlsx")
```

3.行、列、单元格的访问

(1)行的访问

①切片方式访问

在交互窗口中直接输入：

df[0:3] # 取位置 0、1、2，不包括 3

运行得到结果：

	班级	学号	姓名	性别	团员	语文	数学	英语
0	1 班	104	阿地力	男	是	132	124	140
1	1 班	107	伊力亚斯	男	否	128	86	112
2	1 班	115	卢静	女	是	139	88	103

②head、tail 函数访问

I.head 函数 - head(x) 取表格的前 x 个

交互窗口运行以下代码：

df.head(5)

运行得到的结果：

	班级	学号	姓名	性别	团员	语文	数学	英语
0	1 班	104	阿地力	男	是	132	124	140
1	1 班	107	伊力亚斯	男	否	128	86	112
2	1 班	115	卢静	女	是	139	88	103
3	1 班	119	古丽米热	女	否	94	139	116
4	1 班	140	阿依古再丽	女	是	114	86	117

II.tail 函数 - tail(x) 取表格的后 x 个

交互窗口运行以下代码：

df.tail(5)

运行得到的结果：

略

(2)列的访问

①访问单列

交互窗口运行以下代码：

df["姓名"] 或者 df.姓名

运行得到的结果：

```
0 阿地力·艾麦提
1 伊力亚斯·托合苏
……（数据太多，用省略号省略中间部分数据）
19 吴灿
Name: 姓名, dtype: object
```

②访问多列

交互窗口运行以下代码：

```
df[["姓名", "英语", "语文"]]
```

运行得到的结果：

	姓名	英语	语文
0	阿地力	140	132
1	伊力亚斯	112	128

……

19 吴灿 96 148

注意此处用法：是“[]”中再次包裹了“["姓名", "英语", "语文"]”！！！

(3)单元格的访问与修改

通过“df.at[index, 列名称]”来访问单元格的具体数值。比如：“df.at[8, "姓名"]”的结果是“'阿依祖克热·吐尔洪'”；“df.at[8, "语文"]”的结果是“120”。要修改单元格值需要再加一个赋值操作：df.at[8, "姓名"] = "yuyingjian"即可。

4.计算

(1)列与列之间可进行四则混合运算

在 pandas 中，列与列能够直接进行+、-、*、/等四则混合运算，得到的结果也是列。

例 1

交互窗口运行以下代码：

```
df["语文"] + df["数学"] + df["英语"]
```

运行得到的结果：

```
0     396
1     326
……
19    327
dtype: int64
```

要把计算得到的结果放到原来的表格中，可以运行下面的代码：

```
df["总分"] = df["语文"] + df["数学"] + df["英语"]
```

例 2

交互窗口运行以下代码：

```
df["语文"]  / 150 * 100
```

运行得到的结果：

```
0     88.000000
1     85.333333
……
19    98.666667
Name: 语文, dtype: float64
```

(2)使用统计函数进行计算

在 pandas 中内置了对表格、行、列进行统计计算的函数，进一步的简化统计分析工作。

①describe()

describe 函数返回表格中数值列（列中存储的数据是 int、float 等数值）的基本描述统计值。

交互窗口运行以下代码：

```
df.describe()
```

运行得到的结果：

	学号	语文	数学	英语
count	20.000000	20.000000	20.000000	20.000000
mean	266.500000	123.500000	112.750000	123.800000
std	114.586442	18.689992	21.110923	18.746508
min	104.000000	94.000000	80.000000	94.000000
25%	188.000000	104.500000	88.750000	107.750000
50%	270.000000	126.000000	118.000000	122.000000
75%	351.500000	139.000000	129.000000	140.500000
max	436.000000	149.000000	143.000000	149.000000

其中的 count 是列上有多少**非空数据项的计数**，mean 是列上**所有非空数据项的平均值**，min 是列上**所有非空数据项的最小值**，max 是列上**所有非空数据项的最大值**。describe 也可以应用在某一列或者某几列上。

②count()

count 返回表格或者列上**非空数据项的数量**。

交互窗口运行以下代码： df.count()	交互窗口运行以下代码： df["班级"].count()	交互窗口运行以下代码： df[["班级", "学号"]].count()
运行得到的结果： 班级 20 学号 20 姓名 20 性别 20 团员 20 语文 20 数学 20 英语 20 dtype: int64	运行得到的结果： 20	运行得到的结果： 班级 20 学号 20 dtype: int64

③sum()

对行、列上的**非空数据项进行求和**。

I.按列进行计算

要对表格的每一列进行求和计算，可以将 sum 函数的参数 axis 设置为 0。

交互窗口运行以下代码：

```
df.sum() #  省略 axis 就是 axis=0
df.sum(axis=0)
```

运行得到的结果：

班级 1 班 1 班 1 班 1 班 1 班 2 班 2 班 2 班 2 班 2 班 3 班 3 班 3 班 3 班 3 班……

学号 5330

姓名 阿地力伊力亚斯卢静古丽米热阿依古再丽张丞国萨迪克江阿比旦阿……

性别 男男女女女男男女女女男男男女女男男男男女

团员 是否是否是否是否是否否是是否是否否是否是

语文 2470
数学 2255
英语 2476
总分 7201
dtype: object

注意：sum 函数并不是只计算数值列，实际上列上数据项是字符串，它会做简单的连接。

交互窗口运行以下代码： df["语文"].sum() df["语文"].sum(axis=0)	交互窗口运行以下代码： df[["语文", "数学"]].sum() df[["语文", "数学"]].sum(axis=0)
运行得到的结果： 2470	运行得到的结果： 语文 2470 数学 2225 dtype: int64

II.按行进行计算

要对表格的每一行进行求和计算，可以将 sum 函数的参数 axis 设置为 1。

交互窗口运行以下代码： df.sum(axis=1) # axis=1 不能省略	交互窗口运行以下代码： df[["语文", "数学", "英语"]].sum(axis=1)
运行得到的结果： 0 500 …… 19 763 dtype: int64	运行得到的结果： 0 377 …… 19 327 dtype: int64

df.sum(axis=1)将每一行的数值项都累加起来，包括学号、语文、数学、英语，所以 index 为 0 的结果是 104+132+124+140=500。注意，对每一行进行求和的时候，只对非空的数值项进行求和，字符串的数据项会被忽略。df["语文"].sum(axis=1)是没有意义的，因为一列的每一行只有一个数据项，总和是其本身。将计算的语数外总分存到原表格的“总分”列可以使用如下代码：

df["总分"] = df[["语文", "数学", "英语"]].sum(axis=1)

③mean()

对行、列上的**非空数值项进行求平均**。

I.按列进行计算

要对表格的每一列进行求平均计算，可以将 mean 函数的参数 axis 设置为 0。

交互窗口运行以下代码： df.mean() df.mean(axis=0)	交互窗口运行以下代码： df["语文"].mean() df["语文"].mean(axis=0)	交互窗口运行以下代码： df[["语文", "数学"]].mean() df[["语文", "数学"]].mean(axis=0)
运行得到的结果： 学号 266.50 语文 123.15 数学 112.75	运行得到的结果： 123.15	运行得到的结果： 语文 123.15 数学 112.75 dtype: float64

英语 123.80 dtype: float64		

II.按行进行计算

要对表格的每一行进行求平均计算，可以将 mean 函数的参数 axis 设置为 1。

交互窗口运行以下代码： df.mean(axis=1)	交互窗口运行以下代码： df[["语文", "数学", "英语"]].mean(axis=1)
运行得到的结果： 0 125.00 …… 19 190.75 dtype: float64	运行得到的结果： 0 132.000000 …… 19 109.000000 dtype: float64

df.mean(axis=1)将每一行的数值项求平均，包括学号、语文、数学、英语，所以 index 为 0 的结果是(104+132+111+134)/4=125.00。注意，对每一行进行求平均的时候，只对非空的数值项进行求平均，字符串的数据项会被忽略。df["语文"].mean(axis=1)是没有意义的，因为一列的每一行只有一个数据项，平均值是其本身。

③max()与 min()

对行、列上的**非空数据项求最大值、最小值**。

I.按列进行计算

要对表格的每一列进行求最大值、最小值，可以将函数的参数 axis 设置为 0。

交互窗口运行以下代码： df.max() df.max(axis=0)	交互窗口运行以下代码： df["语文"].min() df["语文"].min(axis=0)	交互窗口运行以下代码： df[["语文", "数学"]].max() df[["语文", "数学"]].max(axis=0)
班级 4 班 学号 436 姓名 马硕宇 性别 男 团员 是 语文 149 数学 143 英语 149 dtype: object	运行得到的结果： 94	运行得到的结果： 语文 149 数学 143 dtype: int64

II.按行进行计算

要对表格的每一行进行求最大值、最小值，可以将函数的参数 axis 设置为 1。

交互窗口运行以下代码： df.max(axis=1)	交互窗口运行以下代码： df[["语文", "数学", "英语"]].min(axis=1)
运行得到的结果： 0 140 …… 19 436	运行得到的结果： 0 124 …… 19 83

dtype: int64	dtype: int64

df.max(axis=1)对每一行的数值项求最大值，包括学号、语文、数学、英语，所以 index 为 19 的结果是 436、148、83、96 的最大值 436。注意，对每一行进行求最大值、最小值的时候，只对非空的数值项进行求值，字符串的数据项会被忽略。

5.将 pandas 的表格导出为 excel

```
# 通过 pandas 表格上的 to_excel 函数导出 excel 文件，其参数是导出的 excel 文件名
# to_excel 函数执行的结果把表格 df 中的内容导出到了“导出.xlsx”文件
df.to_excel("导出.xlsx")
```

6.排序

pandas 通过 sort_values 函数完成排序，sort_values 的函数签名如下：

DataFrame 对象.sort_values(列名称, ascending=True/False)

第 1 个参数指定要排序的列，第 2 个参数 ascending 指定是升序还是降序，ascending 为 True 的时候为升序排序，ascending 为 False 的时候是降序排序，省略第 2 个参数等同于 ascending 为 True。

例 1 将 df 对象的数据根据总分降序排序

```
df2 = df.sort_values("总分", ascending=False)
```

索引	班级	学号	姓名	性别	团员	语文	数学	英语	总分
5	2 班	204	张丞国	男	否	147	143	149	439
16	4 班	406	田坤	男	否	147	137	146	430
……									
7	2 班	216	阿比旦	女	否	99	80	94	273

例 2 将 df 对象的数据根据总分升序排序

```
df3 = df.sort_values("总分", ascending=True)
```

索引	班级	学号	姓名	性别	团员	语文	数学	英语	总分
7	2 班	216	阿比旦	女	否	99	80	94	273
4	1 班	140	阿依古再丽	女	是	114	86	117	317
……									
5	2 班	204	张丞国	男	否	147	143	149	439

注意：sort_values 函数并不会改变原有的 DataFrame 对象，sort_values 函数执行完毕得到一个新的排序完成的 DataFrame 对象，为了保存这个 DataFrame 对象，要赋值给新的变量。

7.筛选

pandas 支持对 DataFrame 对象进行筛选，语法如下：

DataFrame 对象[筛选条件]

筛选条件的最小单元是对列的关系运算。

例 1 将 df 对象中的所有女同学筛选出来

```
df4 = df[df["性别"]=="女"]
```

索引	班级	学号	姓名	性别	团员	语文	数学	英语	总分
2	1 班	115	卢静	女	是	139	88	103	330
3	1 班	119	古丽米热	女	否	94	139	116	349
……				女	……				
19	4 班	436	吴灿	女	是	148	83	96	327

例 2 将 df 对象中总分在 380 分以上的同学筛选出来

```
df5 = df[df["总分"]>=380]
```

索引	班级	学号	姓名	性别	团员	语文	数学	英语	总分
0	1 班	104	阿地力	男	是	132	124	140	396
5	2 班	204	张丞国	男	否	147	143	149	439
……									
16	4 班	406	田坤	男	否	147	137	146	430

注意：筛选得到新的 DataFrame 对象，需要通过赋值给新变量来保存新 DataFrame 对象。

8.分组

DataFame 对象上可以使用 groupby 函数，可以对 DataFrame 对象上的行进行分组（对列分组也可以，但是考试不要求），并对每一个分组进行统计计算。groupby 函数的签名如下：

grp = DataFrame 对象.groupby(列名称, as_index=True/False)

第 1 个参数列名称指定依据哪一列的数据来进行分组，第 2 个参数 as_index 控制分组新生成的 DataFrame 对象的 index。未指定 as_index 参数默认为 True。得到分组对象 grp 后，可以对分组对象 grp 上的列进行统计计算。

例 1 将 df 对象中的数据按“班级”分组，并计算分组后各组数据的平均值。

代码（1）

```
grp1 = df.groupby("班级", as_index=False)
dfr1 = grp1.mean()
```

索引	班级	学号	语文	数学	英语	总分
0	1 班	117.0	121.4	104.6	117.6	343.6
1	2 班	214.8	123.2	115.4	135.0	373.6
2	3 班	321.4	124.2	109.6	129.4	363.2
3	4 班	412.8	125.2	121.4	113.2	359.8

as_index=False，索引是自动生成的 0、1、2……的数字序列。分组依据的班级列保留。

代码（2）

```
grp2 = df.groupby("班级", as_index=True)
dfr2 = grp2.mean()
```

索引/班级	学号	语文	数学	英语	总分
1 班	117.0	121.4	104.6	117.6	343.6
2 班	214.8	123.2	115.4	135.0	373.6
3 班	321.4	124.2	109.6	129.4	363.2
4 班	412.8	125.2	121.4	113.2	359.8

as_index=True，分组数据直接转化成了索引，注意！班级列没有了。

上面两段代码有个共同点，学号列是数值列，所以也会对学号列进行求平均的操作。要修正这点，我们可以指定统计计算的列，代码如下所示：

代码（1）

```
grp1 = df.groupby("班级", as_index=False)
dfr1 =
    grp1[["语文", "数学", "英语"]].mean()
```

索引	班级	语文	数学	英语
0	1 班	121.4	104.6	117.6
1	2 班	123.2	115.4	135.0
2	3 班	124.2	109.6	129.4
3	4 班	125.2	121.4	113.2

代码（2）

```
grp2 = df.groupby("班级", as_index=True)
dfr2 =
    grp2[["语文", "数学", "英语"]].mean()
```

索引/班级	语文	数学	英语
1 班	121.4	104.6	117.6
2 班	123.2	115.4	135.0
3 班	124.2	109.6	129.4
4 班	125.2	121.4	113.2

在指定统计的列的时候，也可以只指定一列，“grp1["语文"].mean()”或者“grp2["英语"].mean()”也能够正确的运行并返回结果。

例 2 将 df 对象中的数据按“团员”分组，并计算分组后各组数据的数量。

```
grp3 = df.groupby("团员", as_index=True)
dfr3 = grp3["姓名"].count()
```

索引/团员	姓名
否	10
是	10

在上面的代码中，“"姓名"”可以更换为任意列的列名，因为 count 函数只是统计该列非空数据的数量。

有一点需要注意，在进行统计计算的时候，不管是 mean、count 还是 sum，都会忽略空数据！运行下面的代码：df.at[0, "姓名"] = None; df.at[0, "语文"] = None 将 df 中 index 为 0 的同学的姓名和语文设置为空数据，再执行例 1、例 2 的代码，得到的结果如图：

例 1 代码

索引	班级	语文	数学	英语
0	1 班	**118.75**	104.6	117.6
1	2 班	123.2	115.4	135.0
2	3 班	124.2	109.6	129.4
3	4 班	125.2	121.4	113.2

例 2 代码

索引/团员	姓名
否	10
是	**9**

之所以结果不同，因为 index 为 0 的同学语文成绩为 nan（空数据），在计算 1 班语文平均分的时候 mean 统计函数会忽略它，用其它有数值的语文成绩求平均，所以 1 班的语文平均分发生了变化。index 为 0 的同学姓名为 None（空数据），count 统计的时候会略过他，他是团员，略过之后，团员就少了一位，所以统计的团员数量变成了 9。

8.DataFrame 对象的结构

一个 DataFrame 对象由若干个 Series 对象组成，如图 8.1、8.2 所示。如果把 DataFrame 对象看成一个表格，那么 Series 对象就是列，一个表格由多列组成。

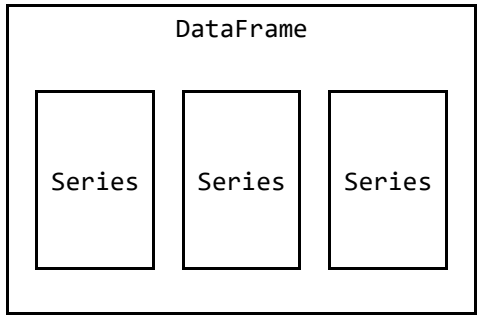
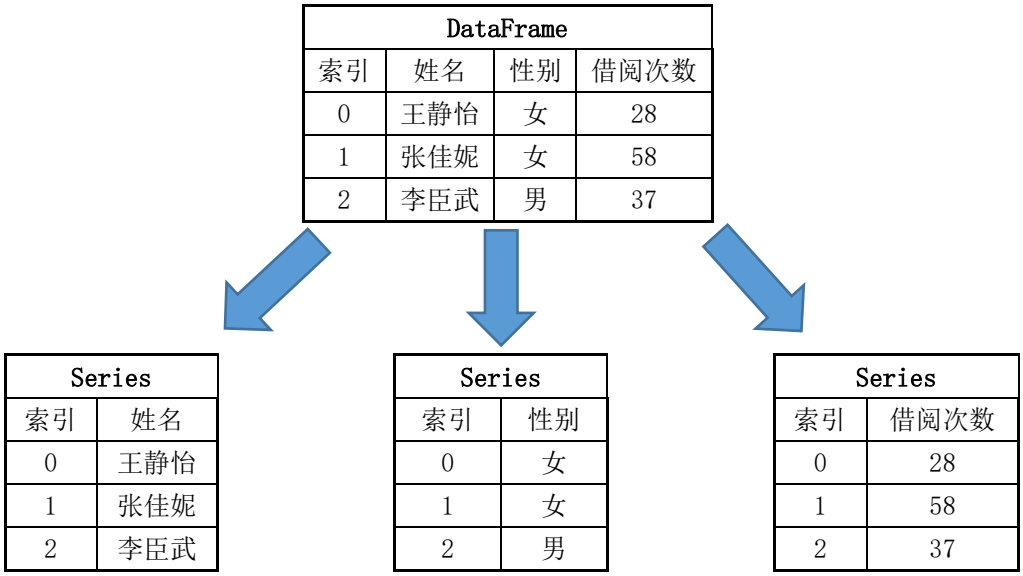


图 8.1 DataFrame 的结构



DataFrame			
索引	姓名	性别	借阅次数
0	王静怡	女	28
1	张佳妮	女	58
2	李臣武	男	37

Series	
索引	姓名
0	王静怡
1	张佳妮
2	李臣武

Series	
索引	性别
0	女
1	女
2	男

Series	
索引	借阅次数
0	28
1	58
2	37

图 8.2 DataFrame 的结构实例

(1)Series 对象

Series 对象表示 DataFrame 中的列，Series 除了包含列中的每一个数据项，还包含了数据项与之对应的索引。

例 1 创建 1 个 Series 对象，存储 3 名同学的姓名

s1 的结构	s1_2 的结构
0 王静怡	s01 王静怡
1 张佳妮	s02 张佳妮
2 李臣武	s03 李臣武
dtype: object	dtype: object

```
s1 = pd.Series(["王静怡", "张佳妮", "李臣武"])
s1_2 = pd.Series(["王静怡","张佳妮","李臣武"], index=["s01", "s02", "s03"])
```

通过索引可以访问 Series 对象的值，s1[0]的值是“王静怡”。通过赋值可以修改 Series 对象的值，s1_2["s01"] = "虞颖健"。通过 Series 对象上的属性 index 可以访问所有的索引，如：s1.index 的值等价于[1, 2, 3]。通过 Series 对象上的属性 values 可以访问所有的数据项，如：s1.values 的值等价于["王静怡", "张佳妮", "李臣武"]。

例 2 通过合并 Series 对象来构建 DataFrame 对象

```
s1 = pd.Series(["王静怡", "张佳妮", "李臣武"], index=[0, 1, 2])
s2 = pd.Series(["女", "女", "男"], index=[1, 2, 0])
s3 = pd.Series([28, 58, 37], index=[2, 0, 1])
df = pd.DataFrame() # 创建一个空的 DataFrame 对象，代码不需要掌握
df["姓名"] = s1
df["性别"] = s2
df.借阅次数 = s3
```

df 的结构：

索引	姓名	性别	借阅次数
0	王静怡	男	58
1	张佳妮	女	37
2	李臣武	女	28

例 2 的程序演示了如何将 Series 对象组合成 DataFrame 对象。**注意！在组合的过程中，Series 索引相同的项会处在 DataFrame 对象的同一行。**s1 索引 1 的“张佳妮”、s2 索引 1 的“女”和 s3 索引 1 的“37”都在 DataFrame 对象的第 2 行。

(2)DataFrame 对象

DataFrame 对象是 Series 对象的合集。

①DataFrame 的构建

例 使用字典构建 DataFrame 对象

```
data = {"姓名": ["王静怡", "张佳妮", "李臣武"],
        "性别": ["女", "女", "男"],
        "借阅次数": [28, 58, 37]}
df = pd.DataFrame(data, columns=["姓名", "性别", "借阅次数"])
df = pd.DataFrame(data) # columns 省略，默认使用字典的键作为列名
```

df 的结构：

索引	姓名	性别	借阅次数
0	王静怡	女	28
1	张佳妮	女	58
2	李臣武	男	37

②DataFrame 上的属性

使用 DataFrame 对象的 index 属性访问索引。df.index 的值等价于“[0, 1, 2]”。

使用 DataFrame 对象的 columns 属性访问索引。df.columns 的值等价于“["姓名

", "性别", "借阅次数"]”。

使用 DataFrame 对象的 values 属性访问每一行。df.values 的值等价于“[['王静怡', '女', 28],['张佳妮', '女', 58],['李臣武', '男', 37]]”，每一行的数据都相当于是一个列表。

使用 DataFrame 对象的 T 属性可以进行行列转置。转置就是第 1 列变第 1 行，第 2 列变第 2 行……，依此类推，df.T 的结构如下：

姓名	王静怡	张佳妮	李臣武
性别	女	女	男
借阅次数	28	58	37

例 修改借阅次数列

```
df.借阅次数 = [30, 52, 68]
```

注意！如果“借阅次数”列存在的情况下修改可以，如果“借阅次数”列不存在，要添加名为“借阅次数”的列，该方式无法正常工作。

③DataFrame 上的常用函数

I.drop 函数

drop 函数删除行（axis=0）或列(axis=1)。drop 函数不改变原有的 DataFrame 对象，通过返回一个新的 DataFrame 对象保存改变后的数据。

```
df1 = df.drop("性别", axis=1)
```

运行后 df1 的结构：

索引	姓名	借阅次数
0	王静怡	28
1	张佳妮	58
2	李臣武	37

```
df2 = df.drop(0)
df2 = df.drop(0, axis=0)
```

运行后 df2 的结构：

索引	姓名	性别	借阅次数
1	张佳妮	女	58
2	李臣武	男	37

II.append 函数　　III.insert 函数　　IV.rename 函数　　V.concat 函数

以上 4 个函数书上没演示代码，暂略。