2.6 Pandas函数使用

In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
plt.rcParams["font.sans-serif"] = ["SimHei"]
plt.rcParams["axes.unicode_minus"] = False  # 显示中文属性设置

# from IPython.core.interactiveshell import InteractiveShell  # 单元格cell输出设置
# InteractiveShell.ast_node_interactivity = "all"  # 整个打印输出cell中所有结果

# 改变当前工作目录到指定的路径（这里改为存放数据的路径，设置当前文档运行环境）
import os
# os.chdir("DA\\data")
# os.getcwd()查看当前路径
```

排序和排名

1. 索引排序

sort_index()：排序默认使用升序排序，ascending=False 为降序排序

2. 按值排序

DataFrame.sort_values('排序列名',ascending=True,na_position)
根据某个唯一的列名进行排序，如果有其他相同列名则报错。
ascending:是否升序排列，默认True，降序则为False
na_position：空值的位置，‘first’或‘last’, default ‘last'.

3. 排名

data['排名列名'].rank(ascending, method = ‘max’) →可计算排名的位次
ascending:是需排名的数据否升序排列，默认True，降序则为False;
method：'max','min','first'排名相同时的取值，默认取均值

- max：相同的值取较大的排名。
- min：相同的值取较小的排名。
- first：按顺序排列，不允许并列。

数据源：一个2016年电子游戏的销量排名数据包，数据包含2016年截至的游戏销量统计排名、数据名称、分地区销量、全球总销量等

- Rank: 排名
- Name：游戏名称
- Platform：游戏运行平台
- Year：游戏发行年份
- Genre：游戏类型
- Publisher：游戏发行人

- NA_Sales：北美销售量（百万套）
- EU_Sales：欧盟销售量（百万套）
- JP_Sales：日本销售量（百万套）
- Other_sales：其余国家销售量（百万套）
- Global_Sales：全球总销售量（百万套）

In [2]:

```
vgsales = pd.read_csv("../data/vgsales.csv", encoding="utf-8", dtype={"Year": float})
vgsales.head(5)
```

Out[2]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sale
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3.7
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.8
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3.7
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01	3.2
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.2

In [3]:

```
vgsales.sort_values(
    # 排序的列名  # 升序  # (缺失值排最后面)
    "Year", ascending=True, na_position="last", inplace=True
)  # inplace =True表示是否作用在原数据上
```

In [4]:

```
vgsales
```

Out[4]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JI
6896	6898	Checkers	2600	1980.0	Misc	Atari	0.22	0.01	
2669	2671	Boxing	2600	1980.0	Fighting	Activision	0.72	0.04	
5366	5368	Freeway	2600	1980.0	Action	Activision	0.32	0.02	
1969	1971	Defender	2600	1980.0	Misc	Atari	0.99	0.05	
1766	1768	Kaboom!	2600	1980.0	Misc	Activision	1.07	0.07	
...	...	...	...	...	...	...	...	...	
16307	16310	Freaky Flyers	GC	NaN	Racing	Unknown	0.01	0.00	
16327	16330	Inversion	PC	NaN	Shooter	Namco Bandai Games	0.01	0.00	
16366	16369	Hakuouki: Shinsengumi Kitan	PS3	NaN	Adventure	Unknown	0.01	0.00	
16427	16430	Virtua Quest	GC	NaN	Role-Playing	Unknown	0.01	0.00	
16493	16496	The Smurfs	3DS	NaN	Action	Unknown	0.00	0.01	

16598 rows × 11 columns

In [5]:

```
# 注意没有加inplace = True表示返回的是视图，没有作用在原数据上
vgsales.sort_values(["NA_Sales", "EU_Sales"])
```

Out[5]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sa
1857	1859	Lode Runner	NES	1984.0	Puzzle	Hudson Soft	0.00	0.00	1
1233	1235	F1 Race	NES	1984.0	Racing	Nintendo	0.00	0.00	1
2955	2957	Mappy	NES	1984.0	Platform	Namco Bandai Games	0.00	0.00	0
1324	1326	4 Nin uchi Mahjong	NES	1984.0	Misc	Nintendo	0.00	0.00	1
1973	1975	Tag Team Match M.U.S.C.L.E.	NES	1985.0	Fighting	Namco Bandai Games	0.00	0.00	1
...	...	...	...	...	...	...	...	...	
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88	3
5	6	Tetris	GB	1989.0	Puzzle	Nintendo	23.20	2.26	4
9	10	Duck Hunt	NES	1984.0	Shooter	Nintendo	26.93	0.63	0
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02	3

16598 rows × 11 columns

In [6]:

```
# 按照索引排序
vgsales.sort_index(ascending=False)  # 按照索引来进行排序,只要不加inplace = True,返回是视图
```

Out[6]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales
16597	16600	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00
16596	16599	Know How 2	DS	2010.0	Puzzle	7G//AMES	0.00	0.01
16595	16598	SCORE International Baja 1000: The Official Game	PS2	2008.0	Racing	Activision	0.00	0.00
16594	16597	Men in Black II: Alien Escape	GC	2003.0	Shooter	Infogrames	0.01	0.00
16593	16596	Woody Woodpecker in Crazy Castle 5	GBA	2002.0	Platform	Kemco	0.01	0.00
...	...	...	...	...	...	...	...	...
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.01
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.88
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.02

16598 rows × 11 columns

排名： data.rank(ascending, method)
是指对数组从1到有效数据点总数分配名次的操作,
默认情况下，rank通过将平均排名分配到每个组来打破平级关系。
使用rank()会对排名求平均值，也就是说有N个相同的元素，排名会相加并除以N

In [7]:

```
vgsales.head(3)
```

Out[7]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales
6896	6898	Checkers	2600	1980.0	Misc	Atari	0.22	0.01	0.0
2669	2671	Boxing	2600	1980.0	Fighting	Activision	0.72	0.04	0.0
5366	5368	Freeway	2600	1980.0	Action	Activision	0.32	0.02	0.0

In [8]:

```
vgsales["全球销售额排名"] = vgsales["Global_Sales"].rank()
```

In [9]:

```
vgsales
```

Out[9]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JI
6896	6898	Checkers	2600	1980.0	Misc	Atari	0.22	0.01	
2669	2671	Boxing	2600	1980.0	Fighting	Activision	0.72	0.04	
5366	5368	Freeway	2600	1980.0	Action	Activision	0.32	0.02	
1969	1971	Defender	2600	1980.0	Misc	Atari	0.99	0.05	
1766	1768	Kaboom!	2600	1980.0	Misc	Activision	1.07	0.07	
...	...	...	...	...	...	...	...	...	...
16307	16310	Freaky Flyers	GC	NaN	Racing	Unknown	0.01	0.00	
16327	16330	Inversion	PC	NaN	Shooter	Namco Bandai Games	0.01	0.00	
16366	16369	Hakuouki: Shinsengumi Kitan	PS3	NaN	Adventure	Unknown	0.01	0.00	
16427	16430	Virtua Quest	GC	NaN	Role-Playing	Unknown	0.01	0.00	
16493	16496	The Smurfs	3DS	NaN	Action	Unknown	0.00	0.01	

16598 rows × 12 columns

In [10]:

```
vgsales["全球销售额降序排名"]=vgsales["Global_Sales"].rank(ascending=False,method='min')
```

In [11]:

```
vgsales
```

Out[11]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sales	JI
6896	6898	Checkers	2600	1980.0	Misc	Atari	0.22	0.01	
2669	2671	Boxing	2600	1980.0	Fighting	Activision	0.72	0.04	
5366	5368	Freeway	2600	1980.0	Action	Activision	0.32	0.02	
1969	1971	Defender	2600	1980.0	Misc	Atari	0.99	0.05	
1766	1768	Kaboom!	2600	1980.0	Misc	Activision	1.07	0.07	
...	...	...	...	...	...	...	...	...	
16307	16310	Freaky Flyers	GC	NaN	Racing	Unknown	0.01	0.00	
16327	16330	Inversion	PC	NaN	Shooter	Namco Bandai Games	0.01	0.00	
16366	16369	Hakuouki: Shinsengumi Kitan	PS3	NaN	Adventure	Unknown	0.01	0.00	
16427	16430	Virtua Quest	GC	NaN	Role-Playing	Unknown	0.01	0.00	
16493	16496	The Smurfs	3DS	NaN	Action	Unknown	0.00	0.01	

16598 rows × 13 columns

In [12]:

```
vgsales = vgsales.sort_values(["全球销售额排名"])
```

In [13]:

```
vgsales
```

Out[13]:

	Rank	Name	Platform	Year	Genre	Publisher	NA_Sales	EU_Sal
16493	16496	The Smurfs	3DS	NaN	Action	Unknown	0.00	0.(
16371	16374	S.Y.K: Shinsetsu Saiyuuki Portable	PSP	2010.0	Adventure	Idea Factory	0.00	0.(
16328	16331	Wedding Planner: Dream Weddings Guaranteed	DS	2010.0	Simulation	DTP Entertainment	0.00	0.(
16074	16077	Umihara Kawase Jun: Second Edition Kanzenban	DS	2009.0	Puzzle	Genterprise	0.00	0.(
16100	16103	Konami Classics Vol. 2	X360	2009.0	Misc	Konami Digital Entertainment	0.01	0.(
...	...	...	...	...	...	...	...	
4	5	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.8
3	4	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.75	11.(
2	3	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.85	12.8
1	2	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.5
0	1	Wii Sports	Wii	2006.0	Sports	Nintendo	41.49	29.(

16598 rows × 13 columns

常用函数小结:

1. dataFrame.sort_index()： 按索引值升序排序,ascending=False 为降序排序
2. dataFrame.sort_values('排序列名',ascending=True,na_position='last')： 根据某个唯一的列名进行排序（默认为True升序排列），如果有其他相同列名则报错。
3. dataFrame.reset_index(inplace=True) :修改原数据的索引(重置索引)
4. data['排名列名'].rank(ascending, method = 'min')： ascending=false,表示先对值（降序）按从高到低排序，再计算排名的位次，排名相同时的取最小值，不填默认取均值（可取：'max','min','first'）

数据合并

在数据采集时，往往会将数据分散存储于不同的数据集中。
而在数据分析时，常常又需要通过一个或多个键将两个数据集的行连接起来，
或者沿着一条轴将多个数据堆叠到一起，以实现数据合并操作。
数据合并操作类似于数据库中运用SQL语句的JOIN连接来实现多表查询。
通过数据合并，可以将多个数据集整合到一个数据集中，在pandas中，常用的数据合并函数有:

1. merge():

- inner:对两张表都有的键的交集进行联合(默认)；全连接 outer：对两者表的都有的键的并集进行联合；
- 左连接 left：对所有左表的键进行联合；右连接 right：对所有右表的键进行联合

2. join():应用于两个数据框的列名没有重复的表，按照索引进行合并（并集，新增列）,列名不重复的表进行左右拼接
3. concat()：axis=0,以列名为主键，进行上下拼接；axis=1,以索引名为主键，进行左右拼接

pd.merge:(left, right, how='inner',on=None,left_index=True,right_index=True)

- left:合并时左边的DataFrame
- right:合并时右边的DataFrame
- how:合并的方式,默认'inner', 'outer', 'left', 'right'
- on:需要合并的列名,必须两边都有的列名，并以 left 和 right 中的列名的交集作为连接键
- 内连接 inner:对两张表都有的键的交集进行联合

In [14]:

```
# 创建产品信息（productinfo）和产品销售（productsales）数据集
info = pd.DataFrame(
    {
        "产品编号": list("ABCD"),
        "类型名称": ["电视机", "手机", "电脑", "空调"],
        "品牌": ["格力", "康佳", "海信", "TCL"],
    },
    index=range(1001, 1005),
)
info
```

Out[14]:

	产品编号	类型名称	品牌
1001	A	电视机	格力
1002	B	手机	康佳
1003	C	电脑	海信
1004	D	空调	TCL

In [15]:

```
sale = pd.DataFrame(
    {
        "产品编号": list("ABECDF"),
        "品牌": ["格力", "康佳", "海信", "TCL", "康佳", "格力"],
        "价格": [3600, 1500, 4500, 2000, 2300, 3500],
    },
    index=range(1001, 1007),
)
sale
```

Out[15]:

	产品编号	品牌	价格
1001	A	格力	3600
1002	B	康佳	1500
1003	E	海信	4500
1004	C	TCL	2000
1005	D	康佳	2300
1006	F	格力	3500

1. merge()

In [16]:

```
# 对重复的列名处理。
pd.merge(info, sale, how="outer", on="产品编号", suffixes=("_info", "_sale"))
```

Out[16]:

	产品编号	类型名称	品牌_info	品牌_sale	价格
0	A	电视机	格力	格力	3600
1	B	手机	康佳	康佳	1500
2	C	电脑	海信	TCL	2000
3	D	空调	TCL	康佳	2300
4	E	NaN	NaN	海信	4500
5	F	NaN	NaN	格力	3500

In [17]:

```
pd.merge(info, sale, on=["产品编号", "品牌"])
```

Out[17]:

	产品编号	类型名称	品牌	价格
0	A	电视机	格力	3600
1	B	手机	康佳	1500

2. join()

In [18]:

```
# 创建产品信息（productinfo）和产品销售（productsales）数据集
info = pd.DataFrame(
    {
        "产品编号1": list("ABCD"),
        "类型名称": ["电视机", "手机", "电脑", "空调"],
        "品牌1": ["格力", "康佳", "海信", "TCL"],
    },
    index=range(1001, 1005),
)
info
```

Out[18]:

	产品编号1	类型名称	品牌1
1001	A	电视机	格力
1002	B	手机	康佳
1003	C	电脑	海信
1004	D	空调	TCL

In [19]:

```
sale = pd.DataFrame(
    {
        "产品编号": list("ABECDF"),
        "品牌": ["格力", "康佳", "海信", "TCL", "康佳", "格力"],
        "价格": [3600, 1500, 4500, 2000, 2300, 3500],
    },
    index=range(1001, 1007),
)
sale
```

Out[19]:

	产品编号	品牌	价格
1001	A	格力	3600
1002	B	康佳	1500
1003	E	海信	4500
1004	C	TCL	2000
1005	D	康佳	2300
1006	F	格力	3500

In [20]:

```
# join是按照索引进行合并，并两个数据框的列名没有重复,新增列(左右拼接)
info.join(sale)# 默认以前面的对象的索引为主
```

Out[20]:

	产品编号1	类型名称	品牌1	产品编号	品牌	价格
1001	A	电视机	格力	A	格力	3600
1002	B	手机	康佳	B	康佳	1500
1003	C	电脑	海信	E	海信	4500
1004	D	空调	TCL	C	TCL	2000

In [21]:

```
info.join(sale,how="outer")#也可以通过how来扩展或减少行索引
```

Out[21]:

	产品编号1	类型名称	品牌1	产品编号	品牌	价格
1001	A	电视机	格力	A	格力	3600
1002	B	手机	康佳	B	康佳	1500
1003	C	电脑	海信	E	海信	4500
1004	D	空调	TCL	C	TCL	2000
1005	NaN	NaN	NaN	D	康佳	2300
1006	NaN	NaN	NaN	F	格力	3500

In [22]:

```
sale.join(info,how="inner")#也可以通过how来扩展或减少行索引
```

Out[22]:

	产品编号	品牌	价格	产品编号1	类型名称	品牌1
1001	A	格力	3600	A	电视机	格力
1002	B	康佳	1500	B	手机	康佳
1003	E	海信	4500	C	电脑	海信
1004	C	TCL	2000	D	空调	TCL

In [23]:

```
sale.join(info)# 默认以前面的对象的索引为主
```

Out[23]:

	产品编号	品牌	价格	产品编号1	类型名称	品牌1
1001	A	格力	3600	A	电视机	格力
1002	B	康佳	1500	B	手机	康佳
1003	E	海信	4500	C	电脑	海信
1004	C	TCL	2000	D	空调	TCL
1005	D	康佳	2300	NaN	NaN	NaN
1006	F	格力	3500	NaN	NaN	NaN

In [24]:

```
sale.join(info,how="outer")#也可以通过how来扩展或减少行索引
```

Out[24]:

	产品编号	品牌	价格	产品编号1	类型名称	品牌1
1001	A	格力	3600	A	电视机	格力
1002	B	康佳	1500	B	手机	康佳
1003	E	海信	4500	C	电脑	海信
1004	C	TCL	2000	D	空调	TCL
1005	D	康佳	2300	NaN	NaN	NaN
1006	F	格力	3500	NaN	NaN	NaN

3. concat()

In [25]:

```
info.columns = ["产品编号", "类型", "品牌"]
info
```

Out[25]:

	产品编号	类型	品牌
1001	A	电视机	格力
1002	B	手机	康佳
1003	C	电脑	海信
1004	D	空调	TCL

In [26]:

```
pd.concat([info, sale])  # 默认外连接，axis=0,当列名一致时，以列名为主键，进行上下拼接
```

Out[26]:

	产品编号	类型	品牌	价格
1001	A	电视机	格力	NaN
1002	B	手机	康佳	NaN
1003	C	电脑	海信	NaN
1004	D	空调	TCL	NaN
1001	A	NaN	格力	3600.0
1002	B	NaN	康佳	1500.0
1003	E	NaN	海信	4500.0
1004	C	NaN	TCL	2000.0
1005	D	NaN	康佳	2300.0
1006	F	NaN	格力	3500.0

In [27]:

```
pd.concat([info, sale], axis=1)  # 当索引一致时，以索引名为主键，进行左右关联连接
```

Out[27]:

	产品编号	类型	品牌	产品编号	品牌	价格
1001	A	电视机	格力	A	格力	3600
1002	B	手机	康佳	B	康佳	1500
1003	C	电脑	海信	E	海信	4500
1004	D	空调	TCL	C	TCL	2000
1005	NaN	NaN	NaN	D	康佳	2300
1006	NaN	NaN	NaN	F	格力	3500

数据合并小结：

1. data1.join(data2):应用于两个数据框的列名没有重复的表，按照索引进行合并（并集，新增列）
2. merge(data1,data2,how,on):对两张表都按指定方式进行联合
3. **pd.concat([data1,data2], axis)**： axis=0,以列名为主键，进行上下拼接；axis=1,以索引名为主键，进行左右拼接

In [ ]:

```

```

In [ ]:

```
```