

1 Pandas

学习目的:

- 掌握pandas中series与dataframe
- pandas数据清洗
- 使用pandas进行数据基本统计
- 时间序列分析
- 数据分析常用方法

2 pandas基础

- pandas简介
- pandas安装
- series数据类型
- dataframe数据类型
- 数据导入

2.1 pandas简介:

- pandas是python中数据分析核心库,能够快速,灵活的对大量数据进行分析,是Python进行数据分析的必要利器;
- pandas支持多种数据导入,支持数据合并,拆分,基本统计,时间序列分析,透视表等多种操作;

2.2 pandas安装

- 1) 如果是anaconda环境,不需要安装,直接使用
- 2) 其他环境可以使用pip安装: pip install Pandas

2.3 Series数据结构

Series: 一维的带索引数据结构(单列)

Series类:

pd.Series(data=None,index=None,dtype=None,name=None,copy=False,fastpath=False)

- 创建Series对象

```
import pandas as pd
import numpy as np
sdata = pd.Series(np.arange(1, 4))
sdata
```

结果:

```
0    1
1    2
2    3
dtype: int32
```

第一列为索引,第二列为Series数据

- 添加索引,并设置数据类型

```
import pandas as pd
import numpy as np
sdata = pd.Series(np.arange(1, 4), index=list('abc'), dtype='U')
sdata
```

结果:

```
a    1
b    2
c    3
dtype: object
```

- Series对象访问

```
In [37]: 1 import pandas as pd
           2 import numpy as np
           3 sdata = pd.Series(np.arange(1, 4), index=list('abc'))
           4 #默认数字索引
           5 print(sdata[0])
           6 #使用标签[a,b,c]
           7 print(sdata['b'])
           8 #使用loc方式,只能使用标签
           9 print(sdata.loc['c'])
```

2.4 获取index与value

```
In [105]: 1 import pandas as pd
           2 import numpy as np
           3 sdata = pd.Series(np.arange(1, 4), index=list('abc'))
```

```
In [109]: 1 #获取索引:
           2 sdata.index.values
```

Out[109]: array(['a', 'b', 'c'], dtype=object)

```
In [110]: 1 #获取索引
           2 sdata.values
```

Out[110]: array([1, 2, 3])

2.5 将index与value转成列表

```
In [115]: 1 #将索引转成列表
           2 sdata.index.values.tolist()
```

Out[115]: ['a', 'b', 'c']

数据分析之_pandas

1 Pandas
2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter过滤
12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [117]: 1 #将数据转成列表  
          2 sdata.values.tolist()
```

```
Out[117]: [1, 2, 3]
```

2.6 dataframe

DataFrame: 多种类型的列构成的二维标签数据结构(多列);

DataFrame类:

```
pd.DataFrame(data=None, index=None, columns=None, dtype=None, copy=False)
```

- data:一维数据, 二维数据
- index:行标签
- columns:列标签

2.6.1 一行一列

```
In [43]: 1 pd.DataFrame(data=np.arange(1, 4))
```

```
Out[43]: 0
```

```
0 1
```

```
1 2
```

```
2 3
```

第一列为行索引, 0为列索引;

2.6.2 多列

```
In [45]: 1 #data为4X4  
          2 data = np.arange(16).reshape(4, 4)  
          3 pd.DataFrame(data=data)
```

```
Out[45]: 0 1 2 3
```

```
0 0 1 2 3
```

```
1 4 5 6 7
```

```
2 8 9 10 11
```

```
3 12 13 14 15
```

2.6.3 设置index与columns

```
In [47]: 1 data = np.arange(16).reshape(4, 4)  
          2 pdata = pd.DataFrame(data=data, index=list('abcd'), columns=['c1', 'c2', 'c3', 'c4'])  
          3 pdata
```

```
Out[47]: c1 c2 c3 c4
```

```
a 0 1 2 3
```

```
b 4 5 6 7
```

```
c 8 9 10 11
```

```
d 12 13 14 15
```

2.6.4 使用字典创建DataFrame

```
In [48]: 1 data = {'c1': [1, 2, 3], 'c2': [4, 5, 6]}  
          2 pdata = pd.DataFrame(data=data)  
          3 pdata
```

```
Out[48]: c1 c2
```

```
0 1 4
```

```
1 2 5
```

```
2 3 6
```

2.6.5 设置列标签

```
In [50]: 1 pdata.columns = ['t1', 't2']  
          2 pdata
```

```
Out[50]: t1 t2
```

```
0 1 4
```

```
1 2 5
```

```
2 3 6
```

2.7 DataFrame对象访问

```
In [74]: 1 data = {'c1': [1, 2, 3], 'c2': [4, 5, 6], 'c3': [7, 8, 9]}  
          2 pdata = pd.DataFrame(data=data)  
          3 pdata
```

```
Out[74]: c1 c2 c3
```

```
0 1 4 7
```

```
1 2 5 8
```

```
2 3 6 9
```

2.7.1 获取指定列

```
In [75]: 1 #获取一列数据, 返回Series对象  
          2 pdata['c1']
```

```
Out[75]: 0 1  
          1 2  
          2 3
```

Name: c1, dtype: int64

数据分析之_pandas

- 1 Pandas
- 2 pandas基础
 - 2.1 pandas简介
 - 2.2 pandas安装
 - 2.3 Series数据结构
 - 2.4 获取index与value
 - 2.5 将index与value转成列表
- 2.6 dataframe
 - 2.6.1 一行一列
 - 2.6.2 多列
 - 2.6.3 设置index与columns
 - 2.6.4 使用字典创建DataFrame
 - 2.6.5 设置列标签
- 2.7 DataFrame对象访问
 - 2.7.1 获取指定列
 - 2.7.2 loc操作
 - 2.7.3 获取指定行
 - 2.7.4 遍历DataFrame对象
- 2.8 DataFrame修改
 - 2.8.1 修改元素
 - 2.8.2 DataFrame插入列
 - 2.8.3 DataFrame插入行
- 3 pandas数据导入与保存
 - 3.1 数据导入
 - 3.1.1 读取excel文件
 - 3.1.2 读取csv文件
 - 3.2 保存文件
- 4 缺失数据处理
 - 4.1 缺失值与空值
 - 4.2 缺失值判断
 - 4.3 判断是否有缺失值
 - 4.4 缺省值处理方式
 - 4.5 缺省值过滤
 - 4.6 删除缺省值
 - 4.7 缺失值填充
 - 4.8 插入均值, 中位数, 最大值, ...
- 5 数据清洗
 - 5.1 准备数据
 - 5.2 获取指定列
 - 5.3 获取指定多列
 - 5.4 根据指定条件获取数据
 - 5.5 根据指定多个条件获取数据
 - 5.6 根据集合获取数据
 - 5.7 根据数据排序
- 6 pandas汇总与描述性统计
 - 6.1 产生数据:
 - 6.2 基本描述与计算
 - 6.3 分位数
- 7 索引/多级索引
 - 7.1 准备数据
 - 7.2 设置索引
 - 7.3 多级索引
 - 7.4 通过多级索引取值
 - 7.5 索引交换
 - 7.6 行列变换
- 8 时间与时间序列
 - 8.1 时间戳
 - 8.2 时间索引
 - 8.3 周期
 - 8.4 时间索引
 - 8.5 时间差值
 - 8.6 重采样
 - 8.7 时间迁移
- 9 数据清洗
 - 9.1 产生数据:
 - 9.2 唯一值与数值出现次数
 - 9.3 删除指定行列
 - 9.4 去重
- 10 数据合并
 - 10.1 merge
 - 10.2 准备数据
 - 10.3 数据合并
 - 10.4 join方法:
 - 10.5 concat
- 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace替换
 - 11.5 agg聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter: 过滤
- 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
 - 12.8 str相关方法:
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
- 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法:

```
In [76]: 1 #取多列数据
          2 pdata[['c1', 'c2']]
```

Out[76]:

	c1	c2
0	1	4
1	2	5
2	3	6

2.7.2 loc操作

loc操作: 使用类似列表方式去对数据进行访问, 支持bool索引;
详细才做可以参考loc的说明

```
In [1]: 1 import pandas as pd
          2 data = {'c1': [1, 2, 3], 'c2': [4, 5, 6], 'c3': [7, 8, 9]}
          3 pdata = pd.DataFrame(data=data)
```

2.7.3 获取指定行

```
In [82]: 1 #获取第一行
          2 pdata.loc[0]
```

Out[82]:

	c1	c2	c3
0	1	4	7

```
In [92]: 1 #获取第一行指定c1,c2列
          2 pdata.loc[0, ['c1', 'c2']]
```

Out[92]:

	c1	c2
0	1	4

2.7.4 遍历DataFrame对象

```
In [3]: 1 pdata
```

Out[3]:

	c1	c2	c3
0	1	4	7
1	2	5	8
2	3	6	9

```
In [2]: 1 #获取列索引
          2 for item in pdata:
          3     print(item)
```

c1

c2

c3

```
In [5]: 1 #按列遍历
          2 for item in pdata.items():
          3     print(item)
```

('c1', 0 1)

1 2

2 3

Name: c1, dtype: int64)

('c2', 0 4)

1 5

2 6

Name: c2, dtype: int64)

('c3', 0 7)

1 8

2 9

Name: c3, dtype: int64)

```
In [7]: 1 #按行遍历
          2 for item in pdata.iterrows():
          3     print(item)
```

(0, c1 1)

c2 4

c3 7

Name: 0, dtype: int64)

(1, c1 2)

c2 5

c3 8

Name: 1, dtype: int64)

(2, c1 3)

c2 6

c3 9

Name: 2, dtype: int64)

数据分析之_pandas

- 1 Pandas
- 2 pandas基础
 - 2.1 pandas简介:
 - 2.2 pandas安装
 - 2.3 Series数据结构
 - 2.4 获取index与value
 - 2.5 将index与value转成列表
- 2.6 dataframe
 - 2.6.1 一行一列
 - 2.6.2 多列
 - 2.6.3 设置index与columns
 - 2.6.4 使用字典创建DataFrame
 - 2.6.5 设置列标签
- 2.7 DataFrame对象访问
 - 2.7.1 获取指定列
 - 2.7.2 loc操作
 - 2.7.3 获取指定行
 - 2.7.4 遍历DataFrame对象
- 2.8 DataFrame修改
 - 2.8.1 修改元素
 - 2.8.2 DataFrame插入列
 - 2.8.3 DataFrame插入行
- 3 pandas数据导入与保存
 - 3.1 数据导入
 - 3.1.1 读取excel文件
 - 3.1.2 读取csv文件:
 - 3.2 保存文件
- 4 缺失数据处理
 - 4.1 缺失值与空值
 - 4.2 缺失值判断
 - 4.3 判断是否有缺失值
 - 4.4 缺省值处理方式
 - 4.5 缺省值过滤
 - 4.6 删除缺省值
 - 4.7 缺失值填充
 - 4.8 插入均值, 中位数, 最大值, :
- 5 数据清洗
 - 5.1 准备数据
 - 5.2 获取指定列
 - 5.3 获取指定多列
 - 5.4 根据指定条件获取数据
 - 5.5 根据指定多个条件获取数据
 - 5.6 根据集合获取数据
 - 5.7 根据数据排序
- 6 pandas汇总与描述性统计
 - 6.1 产生数据:
 - 6.2 基本描述与计算
 - 6.3 分位数
- 7 索引/多级索引
 - 7.1 准备数据
 - 7.2 设置索引
 - 7.3 多级索引
 - 7.4 通过多级索引取值
 - 7.5 索引交换
 - 7.6 行列变换
- 8 时间与时间序列
 - 8.1 时间戳
 - 8.2 时间索引
 - 8.3 周期
 - 8.4 时间索引
 - 8.5 时间差值
 - 8.6 重采样
 - 8.7 时间迁移
- 9 数据清洗
 - 9.1 产生数据:
 - 9.2 唯一值与数值出现次数
 - 9.3 删除指定行列
 - 9.4 去重
- 10 数据合并
 - 10.1 merge
 - 10.2 准备数据
 - 10.3 数据合并
 - 10.4 join方法:
 - 10.5 concat
- 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace:替换
 - 11.5 agg:聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter:过滤
- 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
 - 12.8 str相关方法:
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
- 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法:

```
In [43]: 1 import pandas as pd  
2 data = {'c1': [1, 2, 3], 'c2': [4, 5, 6], 'c3': [7, 8, 9]}  
3 pdata = pd.DataFrame(data=data)  
4 #修改c1列值  
5 pdata['c4'] = [-1, -1, -1]  
6 pdata
```

```
Out[43]:   c1  c2  c3  c4  
0    1    4    7   -1  
1    2    5    8   -1  
2    3    6    9   -1
```

2.8.3 DataFrame插入行

```
In [45]: 1 import pandas as pd  
2 data = {'c1': [1, 2, 3], 'c2': [4, 5, 6], 'c3': [7, 8, 9]}  
3 pdata = pd.DataFrame(data=data)  
4 #修改c1列值  
5 pdata.loc[3] = [-1, -1, -1]  
6 pdata
```

```
Out[45]:   c1  c2  c3  
0    1    4    7  
1    2    5    8  
2    3    6    9  
3   -1   -1   -1
```

3 pandas数据导入与保存

目的:

- 数据导入: excel, csv文件
- 数据导出
- 基本统计
- 缺省数据处理

3.1 数据导入

数据是分析基础, 实际工作中, 数据来自于企业内部数据, 网络数据, 开源数据集;

pandas支持导入方式:

	方法	说明
	pd.read_csv(filepath_or_buffer, sep=',', delimiter=None, header='infer', names=None, index_col=None...)	读取csv文件
	pd.read_excel(io, sheet_name=0, names=None, index_col=None, usecols=None,...)	读取excel文件
	pd.read_json(path_or_buf=None, orient=None, typ='frame', dtype=None,...)	读取json文件

3.1.1 读取excel文件

- 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace:替换
 - 11.5 agg:聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter:过滤
- 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
 - 12.8 str相关方法:
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
- 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法:

数据分析之_pandas

- 1 Pandas
- 2 pandas基础
 - 2.1 pandas简介:
 - 2.2 pandas安装
 - 2.3 Series数据结构
 - 2.4 获取index与value
 - 2.5 将index与value转成列表
- 2.6 dataframe
 - 2.6.1 一行一列
 - 2.6.2 多列
 - 2.6.3 设置index与columns
 - 2.6.4 使用字典创建DataFrame
 - 2.6.5 设置列标签
- 2.7 DataFrame对象访问
 - 2.7.1 获取指定列
 - 2.7.2 loc操作
 - 2.7.3 获取指定行
 - 2.7.4 遍历DataFrame对象
- 2.8 DataFrame修改
 - 2.8.1 修改元素
 - 2.8.2 DataFrame插入列
 - 2.8.3 DataFrame插入行
- 3 pandas数据导入与保存
 - 3.1 数据导入
 - 3.1.1 读取excel文件
 - 3.1.2 读取csv文件:
 - 3.2 保存文件
- 4 缺失数据处理
 - 4.1 缺失值与空值
 - 4.2 缺失值判断
 - 4.3 判断是否有缺失值
 - 4.4 缺省值处理方式
 - 4.5 缺省值过滤
 - 4.6 删除缺省值
 - 4.7 缺失值填充
 - 4.8 插入均值, 中位数, 最大值, :
- 5 数据清洗
 - 5.1 准备数据
 - 5.2 获取指定列
 - 5.3 获取指定多列
 - 5.4 根据指定条件获取数据
 - 5.5 根据指定多个条件获取数据
 - 5.6 根据集合获取数据
 - 5.7 根据数据排序
- 6 pandas汇总与描述性统计
 - 6.1 产生数据:
 - 6.2 基本描述与计算
 - 6.3 分位数
- 7 索引/多级索引
 - 7.1 准备数据
 - 7.2 设置索引
 - 7.3 多级索引
 - 7.4 通过多级索引取值
 - 7.5 索引交换
 - 7.6 行列变换
- 8 时间与时间序列
 - 8.1 时间戳
 - 8.2 时间索引
 - 8.3 周期
 - 8.4 时间索引
 - 8.5 时间差值
 - 8.6 重采样
 - 8.7 时间迁移
- 9 数据清洗
 - 9.1 产生数据:
 - 9.2 唯一值与数值出现次数
 - 9.3 删除指定行列
 - 9.4 去重
- 10 数据合并
 - 10.1 merge
 - 10.2 准备数据
 - 10.3 数据合并
 - 10.4 join方法:
 - 10.5 concat
- 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace:替换
 - 11.5 agg:聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter:过滤
- 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
 - 12.8 str相关方法:
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
- 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法:

In [2]:

```
1 import pandas as pd
2 import numpy as np
3 #读取excel文件
4 fpath = r'F:\database\pandas_dir\cuscap1.xls'
5 pdata = pd.read_excel(fpath)
6 pdata
```

Out[2]:

	user_id	order_dt	order_products	order_amount
0	vs30033073	2020-01-17 00:00:00	1	20
1	vs30026748	2019-12-04 00:00:00	1	20
2	vs10000716	2019-07-05 00:00:00	1	20
3	vs30032785	2019-08-21 00:00:00	2	0
4	vs10000716	2019-10-24 00:00:00	1	20
5	vs30033073	2019-11-29 00:00:00	2	20
6	vs10000621	2019-07-19 00:00:00	2	20
7	vs30029475	2019-05-17 00:00:00	1	20
8	vs30030664	2019-11-11 00:00:00	1	20
9	vs10000773	2019-11-25 00:00:00	1	20
10	vs30032164	2019-07-26 00:00:00	5	131
11	vs30031921	2019-10-24 00:00:00	1	20
12	vs30026748	2019-12-19 00:00:00	1	20
13	vs30032570	2019-07-19 00:00:00	2	40
14	vs30033073	2019-11-13 00:00:00	2	20
15	vs10001043	2019-12-06 00:00:00	1	21
16	vs30029475	2019-10-23 00:00:00	1	20
17	vs30030265	2019-07-11 00:00:00	2	20
18	vs30030664	2019-10-22 00:00:00	1	20
19	vs30032696	2019-08-01 00:00:00	1	0
20	vs30031933	2019-10-24 00:00:00	1	20
21	vs30030265	2019-07-12 00:00:00	2	20
22	vs10000866	2019-09-02 00:00:00	4	0
23	vs10000621	2019-06-13 00:00:00	2	20
24	vs30028628	2019-07-03 00:00:00	1	20
25	vs30030664	2019-11-12 00:00:00	1	20
26	vs30030664	2019-04-26 00:00:00	2	20
27	vs10000621	2019-07-15 00:00:00	2	20
28	vs10000716	2019-08-07 00:00:00	1	20
29	vs30029475	2019-04-01 00:00:00	1	20
...
1983	vs10000925	2020-02-25 13:44:21	1	20
1984	vs30030664	2019-05-29 14:38:27	2	20
1985	vs10001220	2019-09-09 13:09:15	1	0
1986	vs30032940	2019-09-05 09:26:05	1	0
1987	vs30029475	2019-12-10 14:02:18	1	20
1988	vs10000621	2019-04-25 13:19:54	1	20
1989	vs30029475	2019-11-19 15:44:06	1	20
1990	vs10000621	2020-01-14 09:20:03	1	20
1991	vs30026748	2019-07-04 12:12:18	2	20
1992	vs30030664	2019-05-28 16:06:53	2	20
1993	vs10001036	2019-07-29 09:28:23	3	39
1994	vs30032694	2019-08-08 07:59:22	1	0
1995	vs30032722	2019-08-05 10:23:24	1	0
1996	vs10000621	2019-07-23 16:42:32	2	20
1997	vs10000716	2019-10-21 18:27:56	1	20
1998	vs30032612	2020-01-03 15:51:18	1	20
1999	vs10000621	2019-07-16 14:05:47	1	20
2000	vs30030265	2019-07-08 16:14:54	1	20
2001	vs10001399	2020-01-10 09:07:06	1	20
2002	vs30029475	2019-04-08 17:12:07	1	20
2003	vs30033633	2020-01-17 08:51:43	1	0
2004	vs10001220	2019-09-09 13:07:46	1	0
2005	vs30032675	2019-07-31 12:00:10	2	33
2006	vs10000621	2019-06-28 09:44:35	2	20
2007	vs30033467	2019-12-05 15:16:38	2	20
2008	vs30029475	2019-07-03 19:43:09	1	20
2009	vs30029475	2019-03-18 17:37:25	2	20
2010	vs10000621	2020-01-17 09:26:05	1	20
2011	vs30030664	2019-05-24 10:22:16	2	20
2012	vs10000621	2020-01-19 09:15:16	1	20

2013 rows × 4 columns

3.1.2 读取csv文件:

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
▼ 3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [1]: 1 import pandas as pd
2 import numpy as np
3 #读取excel文件
4 fpath = r'F:\database\pandas_dir\GDP.csv'
5 pdata = pd.read_csv(fpath)
6 pdata
```

Out[1]:

	Country	Country Code	1990	1991	1992	1993	1994	1995	1996
0	Aruba	ABW	24101.109430	25870.755940	26533.343900	27430.752400	28656.520210	28648.990020	28499.089430
1	Afghanistan	AFG	NaN						
2	Angola	AGO	3089.683369	3120.356148	2908.160798	2190.768160	2195.532289	2496.199493	2794.896906
3	Albania	ALB	2549.473022	1909.114038	1823.307673	2057.449657	2289.873135	2665.764906	2980.066288
4	Arab World	ARB	6808.206995	6872.273195	7255.328362	7458.647059	7645.682856	7774.207360	8094.149842
5	United Arab Emirates	ARE	72906.520120	71753.729560	71567.827520	70082.389330	72471.687290	74994.380620	76848.792240
6	Argentina	ARG	7380.115031	8210.643432	8942.569853	9777.214005	10435.910770	10225.118710	10857.429670
7	Armenia	ARM	2428.558960	2237.752728	1356.210786	1296.178498	1429.102386	1591.894846	1742.734114
8	Antigua and Barbuda	ATG	11326.702620	11806.234410	12005.986790	12682.475350	13524.670290	12924.355370	13729.757550

3.2 保存文件

方法	说明
pdata.to_csv(path_or_buf=None, sep=',', ...)	保存成csv文件
pdata.to_excel(excel_writer, sheet_name='Sheet1', na_rep='', ...)	保存成excel文件
pdata.to_json(path_or_buf=None, orient=None, ...)	保存成json格式

```
In [4]: 1 import pandas as pd
2 import numpy as np
3 #读取excel文件
4 fpath = r'F:\database\pandas_dir\GDP.csv'
5 csv_path1 = r'F:\database\pandas_dir\new_GDP_1.csv'
6 csv_path2 = r'F:\database\pandas_dir\new_GDP_2.csv'
7 csv_path3 = r'F:\database\pandas_dir\new_GDP_3.csv'
8 pdata = pd.read_csv(fpath)
9 #保存格式带索引
10 pdata.to_csv(csv_path1)
11 #保存格式不带索引
12 pdata.to_csv(csv_path2, index=False)
13 #保存格式不带索引, 保存指定列
14 pdata.to_csv(csv_path3, index=False, columns=['1990', '1991'])
```

4 缺失数据处理

4.1 缺失值与空值

缺省值: 数据集中数值为空的值, pandas使用NaN/NaT表示

空值:""

一个例子 某两只股票数据5天数据, 停牌一天数据为None:

```
In [29]: 1 s1 = [10, 10.5, None, 11]
2 s2 = [7, 6.9, 7.5, None]
3 pdata = pd.DataFrame({'s1':s1, 's2':s2})
4 pdata
```

Out[29]:

	s1	s2
0	10.0	7.0
1	10.5	6.9
2	NaN	7.5
3	11.0	NaN

4.2 缺失值判断

判断方法:

- pd.isnull(): 缺省值对应的值为True, 返回值为Boolean的Series或者DataFrame对象
- pd.notnull(): 缺省值对应的值为False, 返回值为Boolean的Series或者DataFrame对象
- pdata.isnull() / pdata.notnull(): 同上

```
In [35]: 1 sdata = pd.Series([1, 2, 3, np.nan])
2 pd.isnull(sdata)
```

Out[35]:

0	False
1	False
2	False
3	True

```
In [36]: 1 s1 = [10, 10.5, None, 11]
2 s2 = [7, 6.9, 7.5, None]
3 pdata = pd.DataFrame({'s1':s1, 's2':s2})
4 pd.isnull(pdata)
```

Out[36]:

	s1	s2
0	False	False
1	False	False
2	True	False
3	False	True

4.3 判断是否有缺失值

方式: np.all与pd.notnull结合

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺失值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [48]: 1 s1 = [10, 10.5, None, 11]
          2 s2 = [7, 6.9, 7.5, None]
          3 pdata = pd.DataFrame({'s1':s1, 's2':s2})
          4 #pd.notnull, 若包含缺省值, 缺省值对应值为False
          5 #np.all: 若对象中包含假, 返回False, 否则返回真
          6 np.all(pd.notnull(pdata))
          7 #返回False, 说明包含缺省值, 否则不包含缺省值
```

Out[48]: False

```
In [43]: 1 s1 = [10, 10.5, 11]
          2 s2 = [7, 6.9, 7.5]
          3 pdata = pd.DataFrame({'s1':s1, 's2':s2})
          4 np.all(pd.notnull(pdata))
```

Out[43]: True

np.any与pd.isnull()

```
In [49]: 1 s1 = [10, 10.5, 11]
          2 s2 = [7, 6.9, 7.5]
          3 pdata = pd.DataFrame({'s1':s1, 's2':s2})
          4 #isnull: 缺省值对应值为True
          5 #any: 对象中包含真, 返回True
          6 np.any(pd.isnull(pdata))
          7 #返回False, 说明不含缺省值, 返回True说明包括缺省值
```

Out[49]: False

4.4 缺省值处理方式

缺省值处理:

- 过滤缺省值(按行列)
- 删除缺省值(按行列)
- 填充值, 填充值方式:
- 插入均值, 中位数, 最大值, 最小值等
- 插入特殊值
- 插入前(后)值

4.5 缺省值过滤

数据: 某两只股票1周收盘值, None表示当前停牌

```
In [52]: 1 s1 = [10, 10.5, None, 11]
          2 s2 = [7, 6.9, 7.5, None]
          3 pdata = pd.DataFrame({'s1':s1, 's2':s2})
          4 pdata
```

Out[52]:

	s1	s2
0	10.0	7.0
1	10.5	6.9
2	NaN	7.5
3	11.0	NaN

需求: 获取两只股票都没有停牌的数据

```
In [57]: 1 #获取boolean索引
          2 bindex = np.all(pdata.notnull(), axis=1)
          3 bindex
```

Out[57]: array([True, True, False, False], dtype=bool)

```
In [58]: 1 #获取没有停牌数据
          2 pdata[bindex]
```

Out[58]:

	s1	s2
0	10.0	7.0
1	10.5	6.9

4.6 删除缺省值

pdata.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)

参数	说明
axis	0/index:按行, 1/columns:按列
how	根据axis, any:一个为Na删除, all:全部为Na删除
thresh	指定非Na数量(非Na数量>=thresh, 不删除)
subset	指定列子集
inplace	True:在原始数据中修改

准备数据

```
In [68]: 1 s1 = [10, 10.5, None, 11]
          2 s2 = [7, 6.9, 7.5, None]
          3 s3 = [7, 6.9, 7.5, 7]
          4 s4 = [None, 6.9, None, 7.2]
          5 pdata = pd.DataFrame({'s1':s1, 's2':s2, 's3':s3, 's4':s4})
          6 pdata
```

Out[68]:

	s1	s2	s3	s4
0	10.0	7.0	7.0	NaN
1	10.5	6.9	6.9	6.9
2	NaN	7.5	7.5	NaN
3	11.0	NaN	7.0	7.2

需求:

- 1: 删除包含缺省值的行
- 2: 删除包含2个缺省值行
- 3: 删除指定列包含缺省值
- 4: 删除包含缺省值的列

数据分析之_pandas

1 Pandas
2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺失值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [74]: 1 #删除包含缺省值行
          2 pdata.dropna()
```

Out[74]:

	s1	s2	s3	s4
1	10.5	6.9	6.9	6.9

```
In [75]: 1 #缺省值数量大于1, thresh设置为3
          2 pdata.dropna(thresh=3)
```

Out[75]:

	s1	s2	s3	s4
0	10.0	7.0	7.0	NaN
1	10.5	6.9	6.9	6.9
3	11.0	NaN	7.0	7.2

```
In [77]: 1 #指定列: ['s1', 's4']
          2 pdata.dropna(subset=['s1', 's4'])
```

Out[77]:

	s1	s2	s3	s4
1	10.5	6.9	6.9	6.9
3	11.0	NaN	7.0	7.2

```
In [79]: 1 #删除包含缺省值列
          2 pdata.dropna(axis=1)
```

Out[79]:

	s3
0	7.0
1	6.9
2	7.5
3	7.0

注意:

- 以上数据删除都不对原始数据进行修改
- 指定inplace为True, 在原始数据中进行修改

4.7 缺失值填充

填充方法: pdata.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None, **kwargs) 主要参数:

参数	说明
value	填充值
method	填充方式: {'backfill', 'bfill', 'pad', 'ffill', None},
axis	指定行列: 0 or 'index', 1 or 'columns'
limit	插入数量限制

```
In [85]: 1 pdata
```

Out[85]:

	s1	s2	s3	s4
0	10.0	7.0	7.0	NaN
1	10.5	6.9	6.9	6.9
2	NaN	7.5	7.5	NaN
3	11.0	NaN	7.0	7.2

需求:

- 缺省值填充固定值0
- 使用前/后面数据填充
- 使用均值填充
- 插入均值

```
In [87]: 1 #固定值0
          2 pdata.fillna(0)
```

Out[87]:

	s1	s2	s3	s4
0	10.0	7.0	7.0	0.0
1	10.5	6.9	6.9	6.9
2	0.0	7.5	7.5	0.0
3	11.0	0.0	7.0	7.2

```
In [88]: 1 #固定值1
          2 pdata.fillna(1)
```

Out[88]:

	s1	s2	s3	s4
0	10.0	7.0	7.0	1.0
1	10.5	6.9	6.9	6.9
2	1.0	7.5	7.5	1.0
3	11.0	1.0	7.0	7.2

```
In [92]: 1 #使用前一行数据填充
          2 pdata.fillna(method='ffill')
```

Out[92]:

	s1	s2	s3	s4
0	10.0	7.0	7.0	NaN
1	10.5	6.9	6.9	6.9
2	10.5	7.5	7.5	6.9
3	11.0	7.5	7.0	7.2

数据分析之_pandas

- 1 Pandas
- 2 pandas基础
 - 2.1 pandas简介:
 - 2.2 pandas安装
 - 2.3 Series数据结构
 - 2.4 获取index与value
 - 2.5 将index与value转成列表
- 2.6 dataframe
 - 2.6.1 一行一列
 - 2.6.2 多列
 - 2.6.3 设置index与columns
 - 2.6.4 使用字典创建DataFrame
 - 2.6.5 设置列标签
- 2.7 DataFrame对象访问
 - 2.7.1 获取指定列
 - 2.7.2 loc操作
 - 2.7.3 获取指定行
 - 2.7.4 遍历DataFrame对象
- 2.8 DataFrame修改
 - 2.8.1 修改元素
 - 2.8.2 DataFrame插入列
 - 2.8.3 DataFrame插入行
- 3 pandas数据导入与保存
 - 3.1 数据导入
 - 3.1.1 读取excel文件
 - 3.1.2 读取csv文件:
 - 3.2 保存文件
- 4 缺失数据处理
 - 4.1 缺失值与空值
 - 4.2 缺失值判断
 - 4.3 判断是否有缺失值
 - 4.4 缺省值处理方式
 - 4.5 缺省值过滤
 - 4.6 删除缺省值
 - 4.7 缺失值填充
 - 4.8 插入均值, 中位数, 最大值, 最小值
- 5 数据清洗
 - 5.1 准备数据
 - 5.2 获取指定列
 - 5.3 获取指定多列
 - 5.4 根据指定条件获取数据
 - 5.5 根据指定多个条件获取数据
 - 5.6 根据集合获取数据
 - 5.7 根据数据排序
- 6 pandas汇总与描述性统计
 - 6.1 产生数据:
 - 6.2 基本描述与计算
 - 6.3 分位数
- 7 索引/多级索引
 - 7.1 准备数据
 - 7.2 设置索引
 - 7.3 多级索引
 - 7.4 通过多级索引取值
 - 7.5 索引交换
 - 7.6 行列变换
- 8 时间与时间序列
 - 8.1 时间戳
 - 8.2 时间索引
 - 8.3 周期
 - 8.4 时间索引
 - 8.5 时间差值
 - 8.6 重采样
 - 8.7 时间迁移
- 9 数据清洗
 - 9.1 产生数据:
 - 9.2 唯一值与数值出现次数
 - 9.3 删除指定行列
 - 9.4 去重
- 10 数据合并
 - 10.1 merge
 - 10.2 准备数据
 - 10.3 数据合并
 - 10.4 join方法:
 - 10.5 concat
- 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace替换
 - 11.5 agg聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter过滤
- 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
 - 12.8 str相关方法
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
- 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法:

```
In [93]: 1 #使用后一行数据填充
           2 pdata.fillna(method='bfill')
```

```
Out[93]:   s1    s2    s3    s4
0  10.0   7.0   7.0  6.9
1  10.5   6.9   6.9  6.9
2  11.0   7.5   7.5  7.2
3  11.0   NaN   7.0  7.2
```

4.8 插入均值, 中位数, 最大值, 最小值

pdata.mean/max/min/median(axis=None, skipna=None, level=None, numeric_only=None, **kwargs) 主要参数:

参数	说明
axis	方向,0:列, 1: 行
skipna	不计算na, 默认True

```
In [114]: 1 pdata
```

```
Out[114]:   s1    s2    s3    s4
0  10.0   7.0   7.0  NaN
1  10.5   6.9   6.9  6.9
2  NaN    7.5   7.5  NaN
3  11.0   NaN   7.0  7.2
```

```
In [115]: 1 #插入均值
           2 pdata.fillna(pdata.mean())
```

```
Out[115]:   s1    s2    s3    s4
0  10.0  7.000000  7.0  7.05
1  10.5  6.900000  6.9  6.90
2  10.5  7.500000  7.5  7.05
3  11.0  7.133333  7.0  7.20
```

```
In [117]: 1 #插入中位数
           2 pdata.fillna(pdata.median())
```

```
Out[117]:   s1    s2    s3    s4
0  10.0   7.0   7.0  7.05
1  10.5   6.9   6.9  6.90
2  10.5   7.5   7.5  7.05
3  11.0   7.0   7.0  7.20
```

对于股票缺省值, 我们倾向于, 使用前一天数据填充缺失值

5 数据清洗

5.1 准备数据

某次考试成绩

```
In [64]: 1 import pandas as pd
           2 import numpy as np
           3 #
           4 names = list('ABCD')
           5 math = [90, 100, 50, 80]
           6 chinese = [89, 96, 58, 77]
           7 pdata = pd.DataFrame({'name':names, 'math':math, 'chinese':chinese})
           8 pdata
```

```
Out[64]:    name  math  chinese
0      A     90      89
1      B    100      96
2      C     50      58
3      D     80      77
```

5.2 获取指定列

通过列名, 直接获取值

```
In [18]: 1 pdata['name']
```

```
Out[18]: 0      A
          1      B
          2      C
          3      D
Name: name, dtype: object
```

5.3 获取指定多列

基本方法: pdata[[col1, col2]]
获取姓名与数学成绩

```
In [19]: 1 pdata[['name', 'math']]
```

```
Out[19]:    name  math
0      A     90
1      B    100
2      C     50
3      D     80
```

5.4 根据指定条件获取数据

需求1: 数学成绩大于80的所有成绩;
实现思路:

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

- 根据条件生成boolean索引
- 通过boolean索引获取数据

需求2: 获取同学A的成绩;

实现思路:

- 根据条件生成boolean索引
- 通过boolean索引获取数据

```
In [55]: 1 #需求1
           2 #方式1:
           3 bindex = pdata['math'] > 80
           4 pdata[bindex]
           5 #方式2:
           6 pdata[pdata['math'] > 80]
```

```
Out[55]:   name  math  chinese
           0     A    90      89
           1     B   100      96
```

```
In [37]: 1 #需求2
           2 pdata[pdata['name']=='A']
```

```
Out[37]:   name  math  chinese
           0     A    90      89
```

```
In [ ]: 1
```

5.5 根据指定多个条件获取数据

需求1: 获取数学语文都及格成绩 思路:

- 条件1: 数学成绩大于59,
- 条件2: 语文成绩大于59
- 条件3: 两个条件与操作:&
- 基本语法: pdata[condition1&condition2]

需求1: 获取数学语文有一门大于等于80分 思路:

- 条件1: 数学成绩大于等于80,
- 条件2: 语文成绩大于等于80
- 条件3: 两个条件与操作:|
- 基本语法: pdata[condition1|condition2]

```
In [38]: 1 #注意: 两个条件要加括号
           2 pdata[(pdata['math']>59) & (pdata['chinese']>59)]
```

```
Out[38]:   name  math  chinese
           0     A    90      89
           1     B   100      96
           3     D    80      77
```

```
In [39]: 1 #注意: 两个条件要加括号
           2 pdata[(pdata['math']>=80) | (pdata['chinese']>=80)]
```

```
Out[39]:   name  math  chinese
           0     A    90      89
           1     B   100      96
           3     D    80      77
```

5.6 根据集合获取数据

获取数学成绩为100或者90的学生成绩

- 多个值判断: pdata.isin(values), 返回boolean索引

```
bindex = pdata['math'].isin([100, 90])
pdata[bindex]
```

	name	math	chinese
0	A	90	89
1	B	100	96

5.7 根据数据排序

- 排序方式1: 根据索引排序

```
pdata.sort_index(axis=0, level=None, ascending=True, ...)
```

- 排序方式2: 根据指定列内容排序

```
pdata.sort_values(by, axis=0, ascending=True, ...)
```

```
In [76]: 1 import pandas as pd
           2 import numpy as np
           3 #
           4 names = list('ABCD')
           5 math = [90, 100, 80, 80]
           6 chinese = [89, 96, 58, 77]
           7 pdata = pd.DataFrame({'name':names, 'math':math, 'chinese':chinese})
           8 pdata
```

```
Out[76]:   name  math  chinese
           0     A    90      89
           1     B   100      96
           2     C    80      58
           3     D    80      77
```

数据分析之_pandas

- 1 Pandas
- 2 pandas基础
 - 2.1 pandas简介:
 - 2.2 pandas安装
 - 2.3 Series数据结构
 - 2.4 获取index与value
 - 2.5 将index与value转成列表
- 2.6 dataframe
 - 2.6.1 一行一列
 - 2.6.2 多列
 - 2.6.3 设置index与columns
 - 2.6.4 使用字典创建DataFrame
 - 2.6.5 设置列标签
- 2.7 DataFrame对象访问
 - 2.7.1 获取指定列
 - 2.7.2 loc操作
 - 2.7.3 获取指定行
 - 2.7.4 遍历DataFrame对象
- 2.8 DataFrame修改
 - 2.8.1 修改元素
 - 2.8.2 DataFrame插入列
 - 2.8.3 DataFrame插入行
- 3 pandas数据导入与保存
 - 3.1 数据导入
 - 3.1.1 读取excel文件
 - 3.1.2 读取csv文件:
 - 3.2 保存文件
- 4 缺失数据处理
 - 4.1 缺失值与空值
 - 4.2 缺失值判断
 - 4.3 判断是否有缺失值
 - 4.4 缺省值处理方式
 - 4.5 缺省值过滤
 - 4.6 删除缺省值
 - 4.7 缺失值填充
 - 4.8 插入均值, 中位数, 最大值, :
- 5 数据清洗
 - 5.1 准备数据
 - 5.2 获取指定列
 - 5.3 获取指定多列
 - 5.4 根据指定条件获取数据
 - 5.5 根据指定多个条件获取数据
 - 5.6 根据集合获取数据
 - 5.7 根据数据排序
- 6 pandas汇总与描述性统计
 - 6.1 产生数据:
 - 6.2 基本描述与计算
 - 6.3 分位数
- 7 索引/多级索引
 - 7.1 准备数据
 - 7.2 设置索引
 - 7.3 多级索引
 - 7.4 通过多级索引取值
 - 7.5 索引交换
 - 7.6 行列变换
- 8 时间与时间序列
 - 8.1 时间戳
 - 8.2 时间索引
 - 8.3 周期
 - 8.4 时间索引
 - 8.5 时间差值
 - 8.6 重采样
 - 8.7 时间迁移
- 9 数据清洗
 - 9.1 产生数据:
 - 9.2 唯一值与数值出现次数
 - 9.3 删除指定行列
 - 9.4 去重
- 10 数据合并
 - 10.1 merge
 - 10.2 准备数据
 - 10.3 数据合并
 - 10.4 join方法:
 - 10.5 concat
- 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace替换
 - 11.5 agg聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter: 过滤
- 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
 - 12.8 str相关方法
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
- 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法:

```
In [64]: 1 #根据索引排序, 降序, ascending=False
          2 pdata.sort_index(ascending=False)
```

```
Out[64]:   name  math  chinese
            3     D    80      77
            2     C    80      58
            1     B   100      96
            0     A    90      89
```

```
In [65]: 1 #根据数学成绩排序, 降序, ascending=False
          2 pdata.sort_values(['math'], ascending=False)
```

```
Out[65]:   name  math  chinese
            1     B   100      96
            0     A    90      89
            2     C    80      58
            3     D    80      77
```

```
In [67]: 1 #根据数学与语文成绩排序, 降序, ascending=False
          2 #sort_values中加入两列数据
          3 pdata.sort_values(['math', 'chinese'], ascending=False)
```

```
Out[67]:   name  math  chinese
            1     B   100      96
            0     A    90      89
            3     D    80      77
            2     C    80      58
```

6 pandas汇总与描述性统计

pandas计算与统计相关方法:

最大值: pdata.max(axis=None, skipna=None, level=None, numeric_only=None, **kwargs)
最小值: pdata.min(axis=None, skipna=None, level=None, numeric_only=None, **kwargs)
均值: pdata.mean(axis=None, skipna=None, level=None, numeric_only=None, **kwargs)
中位数: pdata.median(axis=None, skipna=None, level=None, numeric_only=None, **kwargs)
求和: pdata.sum(axis=None, skipna=None, level=None, numeric_only=None, min_count=0, **kwargs)
方差: pdata.var(axis=None, skipna=None, level=None, ddof=1, numeric_only=None, **kwargs)
标准差: pdata.std(axis=None, skipna=None, level=None, ddof=1, numeric_only=None, **kwargs)
累加和: pdata.cumsum(axis=None, skipna=True, *args, **kwargs)
分位数: pdata.quantile(q=0.5, axis=0, numeric_only=True, interpolation='linear')
每个数值到均值的平均差: pdata.mad(axis=None, skipna=None, level=None)
元素与先前元素的相差百分比: pdata.pct_change(periods=1, fill_method='pad', limit=None, freq=None, **kwargs)
偏度: pdata.skew(axis=None, skipna=None, level=None, numeric_only=None, **kwargs)
峰度: pdata.kurt(axis=None, skipna=None, level=None, numeric_only=None, **kwargs)
数据描述: pdata.describe(percentiles=None, include=None, exclude=None)

通用参数说明:

参数	说明
axis	index (0)/columns (1)
skipna	是否跳过Na/Null, 默认True
level	多级索引level

```
In [ ]:
```

```
1
```

6.1 产生数据:

```
In [74]: 1 import pandas as pd
          2 import numpy as np
          3 #
          4 names = list('ABCD')
          5 math = [90, 100, 80, 80]
          6 chinese = [89, 96, 58, 77]
          7 pdata = pd.DataFrame({'name':names, 'math':math, 'chinese':chinese})
          8 pdata
```

```
Out[74]:   name  math  chinese
            0     A    90      89
            1     B   100      96
            2     C    80      58
            3     D    80      77
```

6.2 基本描述与计算

```
In [129]: 1 #最大值, 最小值, 四分位数, 均值, 数量, 标准差
          2 pdata.describe()
```

```
Out[129]:   math  chinese
            count  4.000000  4.0000
            mean   87.500000  80.0000
            std    9.574271  16.6333
            min   80.000000  58.0000
            25%  80.000000  72.2500
            50%  85.000000  83.0000
            75%  92.500000  90.7500
            max 100.000000  96.0000
```

数据分析之_pandas	1 Pandas
	2 pandas基础
	2.1 pandas简介:
	2.2 pandas安装
	2.3 Series数据结构
	2.4 获取index与value
	2.5 将index与value转成列表
	2.6 dataframe
	2.6.1 一行一列
	2.6.2 多列
	2.6.3 设置index与columns
	2.6.4 使用字典创建DataFrame
	2.6.5 设置列标签
	2.7 DataFrame对象访问
	2.7.1 获取指定列
	2.7.2 loc操作
	2.7.3 获取指定行
	2.7.4 遍历DataFrame对象
	2.8 DataFrame修改
	2.8.1 修改元素
	2.8.2 DataFrame插入列
	2.8.3 DataFrame插入行
	3 pandas数据导入与保存
	3.1 数据导入
	3.1.1 读取excel文件
	3.1.2 读取csv文件:
	3.2 保存文件
	4 缺失数据处理
	4.1 缺失值与空值
	4.2 缺失值判断
	4.3 判断是否有缺失值
	4.4 缺省值处理方式
	4.5 缺省值过滤
	4.6 删除缺省值
	4.7 缺失值填充
	4.8 插入均值, 中位数, 最大值, :
	5 数据清洗
	5.1 准备数据
	5.2 获取指定列
	5.3 获取指定多列
	5.4 根据指定条件获取数据
	5.5 根据指定多个条件获取数据
	5.6 根据集合获取数据
	5.7 根据数据排序
	6 pandas汇总与描述性统计
	6.1 产生数据:
	6.2 基本描述与计算
	6.3 分位数
	7 索引/多级索引
	7.1 准备数据
	7.2 设置索引
	7.3 多级索引
	8 时间与时间序列
	8.1 时间戳
	8.2 时间索引
	8.3 周期
	8.4 时间索引
	8.5 时间差值
	8.6 重采样
	8.7 时间迁移
	9 数据清洗
	9.1 产生数据:
	9.2 唯一值与数值出现次数
	9.3 删除指定行列
	9.4 去重
	10 数据合并
	10.1 merge
	10.2 准备数据
	10.3 数据合并
	10.4 join方法:
	10.5 concat
	11 pandas数据处理常用函数
	11.1 apply函数
	11.2 func处理对象
	11.3 map
	11.4 replace替换
	11.5 agg聚合操作
	11.6 transform: 处理数据
	11.7 filter: 过滤
	12 分组处理
	12.1 groupby分组
	12.2 分组基本操作
	12.3 聚合操作(Aggregations)
	12.4 transform
	12.5 filter
	12.6 cut分组
	12.7 透视表
	12.8 str相关方法:
	12.8.1 字符串类似方法:
	12.8.2 正则类似方法:
	13 pandas可视化:
	13.1 基本使用
	13.2 plot中可视化方法:

```
In [132]: 1 #计算每个学生总分, 平均分
2 print(pdata.sum(axis=1))
3 print(pdata.mean(axis=1))
```

```
0 179
1 196
2 138
3 157
dtype: int64
0 89.5
1 98.0
2 69.0
3 78.5
dtype: float64
```

6.3 分位数

计算: 下四分位数, 中位数, 上四分位数

```
In [136]: 1 #下四分位数
2 print(pdata.quantile(q=0.25))
3 #中位数
4 print(pdata.quantile(q=0.5))
5 #上位数
6 print(pdata.quantile(q=0.75))
```

```
math      80.00
chinese   72.25
Name: 0.25, dtype: float64
math      85.0
chinese   83.0
Name: 0.5, dtype: float64
math      92.50
chinese   90.75
Name: 0.75, dtype: float64
```

7 索引/多级索引

主要内容

- 重置索引
- 索引变换
- 多级索引

7.1 准备数据

```
In [226]: 1 import pandas as pd
2 import numpy as np
3 names = list('ABCDABCD')
4 pdata = pd.DataFrame(np.random.randint(30,100, size=(8,2)),columns=['math', 'chinese'])
5 pdata['team'] = [1]*4+[2]*4
6 pdata['name'] = names
7 pdata
```

```
Out[226]:   math  chinese  team  name
0      92       41     1    A
1      36       48     1    B
2      31       41     1    C
3      97       32     1    D
4      40       55     2    A
5      60       51     2    B
6      99       61     2    C
7      71       78     2    D
```

7.2 设置索引

- name设置为索引
- 重置索引方法: pdata.set_index(keys, drop=True, append=False, inplace=False) 主要参数:

参数	说明
keys	指定索引名称, 可以多列
drop	True:删除列数据, False保留列数据
append	True: 在原有列基础上追加
inplace	True: 在原数据中修改

```
In [227]: 1 #set_index(inplace返回副本, 新数据
2 ndata = pdata.set_index('name')
3 ndata
```

```
Out[227]:   math  chinese  team
name
A      92       41     1
B      36       48     1
C      31       41     1
D      97       32     1
A      40       55     2
B      60       51     2
C      99       61     2
D      71       78     2
```

7.3 多级索引

- 需求: 通过索引获取指定学期数据
- 知识点: 设置多级索引:

```
#根据array生成索引
pd.MultiIndex.from_arrays(arrays, sortorder=None, names=None)
```

数据:

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
▼ 3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, ...
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法:
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [238]: 1 import pandas as pd
2 import numpy as np
3 names = list('ABCDABCD')
4 pdata = pd.DataFrame(np.random.randint(30,100, size=(8,2)), columns=['math', 'chinese'])
5 lv1 = [1]*4+[2]*4
6 lv2 = list('ABCDABCD')
7 #创建MultiIndex对象
8 mindex = pd.MultiIndex.from_arrays([lv1, lv2])
9 data = pdata.set_index(mindex)
10 data
```

```
Out[238]:
```

	math	chinese
1	A 91	96
	B 79	49
1	C 34	46
	D 47	72
2	A 71	98
	B 91	65
2	C 93	98
	D 93	31

7.4 通过多级索引取值

需求:

- 获取第一学期数据
- 获取第一学期A同学数据
- 获取A同学所有数据

```
In [239]: 1 #第一学期数据
2 data.loc[1]
```

```
Out[239]:
```

	math	chinese
A	91	96
B	79	49
C	34	46
D	47	72

```
In [240]: 1 #第一学期A同学数据
2 data.loc[(1, 'A')]
```

```
Out[240]:
```

math 91
chinese 96
Name: (1, A), dtype: int32

问题: 如何获取A同学所有数据?

7.5 索引交换

方法: data.swaplevel(i=-2, j=-1, axis=0)

```
In [241]: 1 data = data.swaplevel(0, 1)
2 data
```

```
Out[241]:
```

	math	chinese
A 1	91	96
B 1	79	49
C 1	34	46
D 1	47	72
A 2	71	98
B 2	91	65
C 2	93	98
D 2	93	31

```
In [224]: 1 data.loc['A']
```

```
Out[224]:
```

	math	chinese
1	61	89
2	54	66

7.6 行列变换

- pdata.stack(level=-1, dropna=True):将列“旋转”为行
- pdata.unstack(level=-1, fill_value=None):将行“旋转”为列

准备数据

数据分析之_pandas

1 Pandas
2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [263]: 1 import pandas as pd
2 import numpy as np
3 names = list('ABCDABCD')
4 pdata = pd.DataFrame(np.random.randint(30,100, size=(8,2)), columns=['math', 'chinese'])
5 lv1 = [1]*4+[2]*4
6 lv2 = list('ABCDABCD')
7 mindex = pd.MultiIndex.from_arrays([lv1, lv2])
8 pdata = pdata.set_index(mindx)
9 pdata
```

```
Out[263]:
```

	math	chinese
1	A 39	35
1	B 73	30
1	C 94	77
1	D 37	54
2	A 99	95
2	B 77	87
2	C 59	86
2	D 63	80

需求:

- 将学期转到列
- 获取所有学生数学成绩
- 获取第一学期数据

```
In [274]: 1 tmp = pdata.unstack(level=0)
2 tmp
```

```
Out[274]:
```

	math	chinese
1	1 2	1 2
A	39 99	35 95
B	73 77	30 87
C	94 59	77 86
D	37 63	54 80

```
In [275]: 1 tmp['math']
```

```
Out[275]:
```

	1	2
A	39	99
B	73	77
C	94	59
D	37	63

```
In [279]: 1 #列索引层级交换, 获取第一学期数据
2 tmp.swaplevel(axis=1)[1]
```

```
Out[279]:
```

	math	chinese
A	39	35
B	73	30
C	94	77
D	37	54

```
In [289]: 1 tmp
```

```
Out[289]:
```

	math	chinese
1	1 2	1 2
A	39 99	35 95
B	73 77	30 87
C	94 59	77 86
D	37 63	54 80

- 将列中的math, chinese转成索引

```
In [301]: 1 t = tmp.stack(level=0)
2 t
```

```
Out[301]:
```

	1	2
A	chinese 35 95	
	math 39 99	
B	chinese 30 87	
	math 73 77	
C	chinese 77 86	
	math 94 59	
D	chinese 54 80	
	math 37 63	

```
In [306]: 1 #获取A同学第一学期成绩
2 t.loc['A'][1]
```

```
Out[306]:
```

chinese 35
math 39
Name: 1, dtype: int32

8 时间与时间序列

时间是数据分析重要维度, pandas中时间主要知识点:

- 时间戳
- 周期
- 时间间隔
- 时间索引
- 时间滑动窗口

8.1 时间戳

时间处理中常见的对象;

时间戳方法:

- pd.Timestamp(ts_input=<object object at 0x00000258D4E907F0>, freq=None, tz=None,...), 详情见说明案例
- pd.to_datetime(arg, errors='raise', dayfirst=False,...)

相关操作:

In [342]:

```
1 #年
2 print(pd.Timestamp(2020))
3 #年月日
4 print(pd.Timestamp(2020, 6, 2))
5 #字符串
6 print(pd.Timestamp('2020-05-07'))
7 #字符串时间
8 print(pd.Timestamp('2020-05-07 04:02:01'))
9 #字符串时间
10 print(pd.Timestamp('2017-01-01T12'))
11 #时间戳
12 print(pd.Timestamp(1513393355.5, unit='s'))
13 #2017-03-01与format对应
14 print(pd.to_datetime('2017-02-01', format="%Y-%m-%d"))
15 #20170301与年月日对应
16 print(pd.to_datetime('20170301', format="%Y%m%d"))
```

```
1970-01-01 00:00:00.0000002020
2020-06-02 00:00:00
2020-05-07 00:00:00
2020-05-07 04:02:01
2017-01-01 12:00:00
2017-12-16 03:02:35.500000
2017-02-01 00:00:00
2017-03-01 00:00:00
```

In [402]:

```
1 print(pd.to_datetime('2017-02-01', format="%Y-%m-%d"))
2 #生成时间索引
3 print(pd.to_datetime(['2017-02-01'], format="%Y-%m-%d"))
```

```
2017-02-01 00:00:00
DatetimeIndex(['2017-02-01'], dtype='datetime64[ns]', freq=None)
```

8.2 时间索引

- pd.date_range(start=None, end=None, periods=None, freq=None, tz=None, ... **kwargs): 生成DatetimeIndex, 官方文档: [\(https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.date_range.html\)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.date_range.html)

主要参数:

参数	说明
start	开始时间
end	结束时间
periods	产生周期数量
freq	间隔, 默认为D(天)
closed	闭区间: {None, 'left', 'right'}

freq的主要参数: [\(https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#timeseries-offset-aliases\)](https://pandas.pydata.org/pandas-docs/stable/user_guide/timeseries.html#timeseries-offset-aliases)

参数	说明
M	每月最后一天
MS	每月第一天
D	天
H	小时
T, min	分
S	秒
Q	季度

相关操作:

In [398]:

```
1 #周期单位为Day
2 print(pd.date_range('2017-01-01', periods=2))
3 #周期单位为hour
4 print(pd.date_range('2017-01-01 02', periods=2, freq='H'))
5 #每个月01
6 print(pd.date_range('2017-01', periods=3, freq='MS'))
7 #每个月月底
8 print(pd.date_range('2017-01', periods=3, freq='M'))
```

```
DatetimeIndex(['2017-01-01', '2017-01-02'], dtype='datetime64[ns]', freq='D')
DatetimeIndex(['2017-01-01 02:00:00', '2017-01-01 03:00:00'], dtype='datetime64[ns]', freq='H')
DatetimeIndex(['2017-01-01', '2017-02-01', '2017-03-01'], dtype='datetime64[ns]', freq='MS')
DatetimeIndex(['2017-01-31', '2017-02-28', '2017-03-31'], dtype='datetime64[ns]', freq='M')
```

8.3 周期

通过时间段与固定时间间隔一系列时间;

周期作用: 可以获取指定年, 指定月, 指定日等的数据

- 周期: pd.Period(value=None, freq=None, ordinal=None, year=None, ...)
- 周期序列: pd.period_range(start=None, end=None, periods=None, freq=None, name=None): 生成PeriodIndex
- pd.date.to_period(freq=None, axis=0, copy=True)

In [399]:

```
1 print(pd.Period('2017-01-02'))
2 print(pd.Period('2017-01'))
3 print(pd.Period('2017'))
```

```
2017-01-02
2017-01
2017
```

数据分析之_pandas

1 Pandas
2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺失值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [400]: 1 #周期单位为Day
2 print(pd.period_range('2017-01-01', periods=2))
3 #周期单位为月
4 print(pd.period_range('2017-01', periods=2, freq='M'))
5 #周期单位为小时
6 print(pd.period_range('2017-01-02', periods=2, freq='H'))
```

```
PeriodIndex(['2017-01-01', '2017-01-02'], dtype='period[D]', freq='D')
PeriodIndex(['2017-01', '2017-02'], dtype='period[M]', freq='M')
PeriodIndex(['2017-01-02 00:00', '2017-01-02 01:00'], dtype='period[H]', freq='H')
```

```
In [411]: 1 tmp = pd.to_datetime(['2017-01-02', '2017-03-04'])
2 tmp.to_period('M')
```

```
Out[411]: PeriodIndex(['2017-01', '2017-03'], dtype='period[M]', freq='M')
```

8.4 时间索引

很多数据及数据集中都会有时序维度，可以将其设置为时间索引；
内容：

- 获取指定时间数据
- 获取时间段数据
- 获取某个时期数据

```
In [60]: 1 #准备数据
2 import pandas as pd
3 import numpy as np
4 #注意这里是字符串
5 ts = ['2019-03-25', '2019-03-26', '2019-03-27', '2019-03-28', '2019-03-29', '2019-03-30', '2019-03-31',
6       '2019-04-01', '2019-04-02', '2019-04-03', '2019-04-04', '2019-04-05', '2019-04-06']
7 values = np.arange(len(ts))
8 pdata = pd.DataFrame({'ts':ts, 'values':values})
9 pdata
```

```
Out[60]:      ts  values
0  2019-03-25      0
1  2019-03-26      1
2  2019-03-27      2
3  2019-03-28      3
4  2019-03-29      4
5  2019-03-30      5
6  2019-03-31      6
7  2019-04-01      7
8  2019-04-02      8
9  2019-04-03      9
10 2019-04-04     10
11 2019-05-01     11
12 2019-04-05     12
```

需求：

- 获取三月份数据
- 获取4月1号到4号数据
- 获取第2季度数据

```
In [61]: 1 #生成时间索引
2 #方式1:
3 tindex = pd.to_datetime(pdata['ts'])
4 #方式2
5 tindex = pd.DatetimeIndex(pdata['ts'])
6 #创建时间索引
7 tmp = pdata.set_index(tindex)
8
```

```
In [46]: 1 tmp.index
```

```
Out[46]: DatetimeIndex(['2019-03-25', '2019-03-26', '2019-03-27', '2019-03-28',
 '2019-03-29', '2019-03-30', '2019-03-31', '2019-04-01',
 '2019-04-02', '2019-04-03', '2019-04-04', '2019-04-05'],
 dtype='datetime64[ns]', name='ts', freq=None)
```

```
In [62]: 1 #三月数据
2 tmp['2019-03']
```

```
Out[62]:      ts  values
ts
2019-03-25  2019-03-25      0
2019-03-26  2019-03-26      1
2019-03-27  2019-03-27      2
2019-03-28  2019-03-28      3
2019-03-29  2019-03-29      4
2019-03-30  2019-03-30      5
2019-03-31  2019-03-31      6
```

```
In [65]: 1 #4月1号到4号数据: 切片操作
2 tmp.loc['2019-04-01':'2019-04-05']
```

```
Out[65]:      ts  values
ts
2019-04-01  2019-04-01      7
2019-04-02  2019-04-02      8
2019-04-03  2019-04-03      9
2019-04-04  2019-04-04     10
2019-04-05  2019-04-05     12
```

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
▼ 3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [64]: 1 # 获取第2季度数据
          2 #按季度生成索引
          3 qindex = tmp.index.to_period('Q')
          4 qdata = tmp.set_index(qindex)
          5 qdata['2019Q2']
```

```
Out[64]:      ts  values
              ts
2019Q2 2019-04-01    7
2019Q2 2019-04-02    8
2019Q2 2019-04-03    9
2019Q2 2019-04-04   10
2019Q2 2019-05-01   11
2019Q2 2019-04-05   12
```

8.5 时间差值

- timedelta: 两个datetime值之间的差(如日,秒和微妙)的类型
- pd.Timedelta(value=<object object at 0x0000023DD1424CC0>, unit=None, **kwargs)

```
In [51]: 1 #Timestamp相减
          2 pd.Timestamp('2019-01-02')-pd.Timestamp('2019-01-01')
```

```
Out[51]: Timedelta('1 days 00:00:00')
```

```
In [52]: 1 pd.Timedelta(5, 'T')
```

```
Out[52]: Timedelta('0 days 00:05:00')
```

8.6 重采样

重采样作用:

- 降低采样率
- 提升采样率
- 方法: pdata.resample(rule,how=None, axis=0, fill_method=None, closed=None, label=None, convention='start'...)

参数说明:

参数	说明
rule	规则,'T', 'M'
fill_method	提升采样率填充方式, 'ffill'、'bfill'
closed	降低采样率, 闭合方式: 'right'或'left', 默认'right'
label	降低采样率, 聚合值标签, {'right', 'left'}
loffset	时间偏差, timedelta
kind	聚合方式: 'period'或'timestamp',默认聚合到时间索引

应用场景: 股票分析, 金融等;

产生数据:

```
In [156]: 1 import pandas as pd
          2 import numpy as np
          3 index = pd.date_range('1/1/2000', periods=9, freq='2T')
          4 series = pd.Series(range(9), index=index)
          5 series
```

```
Out[156]: 2000-01-01 00:00:00    0
          2000-01-01 00:02:00    1
          2000-01-01 00:04:00    2
          2000-01-01 00:06:00    3
          2000-01-01 00:08:00    4
          2000-01-01 00:10:00    5
          2000-01-01 00:12:00    6
          2000-01-01 00:14:00    7
          2000-01-01 00:16:00    8
          Freq: 2T, dtype: int64
```

```
In [157]: 1 #降低采样率, 将时间间隔改成4S, 每个时间对应值为均值
          2 s = series.resample('4T')
```

Resampler对象相关方法:

方法	说明
s.groups	Resampler对象, 字典
s.max()	降频分组最大值
s.min()	降频分组后最小值
s.first()	降频分组第一个值
s.last()	降频分组最后一个值
s.mean()	降频分组均值
s.median()	降频分组后中位数

```
In [158]: 1 s.first()
```

```
Out[158]: 2000-01-01 00:00:00    0
          2000-01-01 00:04:00    2
          2000-01-01 00:08:00    4
          2000-01-01 00:12:00    6
          2000-01-01 00:16:00    8
          Freq: 4T, dtype: int64
```

```
In [159]: 1 s.mean()
```

```
Out[159]: 2000-01-01 00:00:00    0.5
          2000-01-01 00:04:00    2.5
          2000-01-01 00:08:00    4.5
          2000-01-01 00:12:00    6.5
          2000-01-01 00:16:00    8.0
          Freq: 4T, dtype: float64
```

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [160]: 1 s.median()
Out[160]: 2000-01-01 00:00:00    0.5
2000-01-01 00:04:00    2.5
2000-01-01 00:08:00    4.5
2000-01-01 00:12:00    6.5
2000-01-01 00:16:00    8.0
Freq: 4T, dtype: float64
```

```
In [161]: 1 #提高采样率, 将时间间隔改成S, 每个时间对应值为均值
2 rd = series.resample('T')
3 rd.bfill()
```

```
Out[161]: 2000-01-01 00:00:00    0
2000-01-01 00:01:00    1
2000-01-01 00:02:00    1
2000-01-01 00:03:00    2
2000-01-01 00:04:00    2
2000-01-01 00:05:00    3
2000-01-01 00:06:00    3
2000-01-01 00:07:00    4
2000-01-01 00:08:00    4
2000-01-01 00:09:00    5
2000-01-01 00:10:00    5
2000-01-01 00:11:00    6
2000-01-01 00:12:00    6
2000-01-01 00:13:00    7
2000-01-01 00:14:00    7
2000-01-01 00:15:00    8
2000-01-01 00:16:00    8
Freq: T, dtype: int64
```

8.7 时间迁移

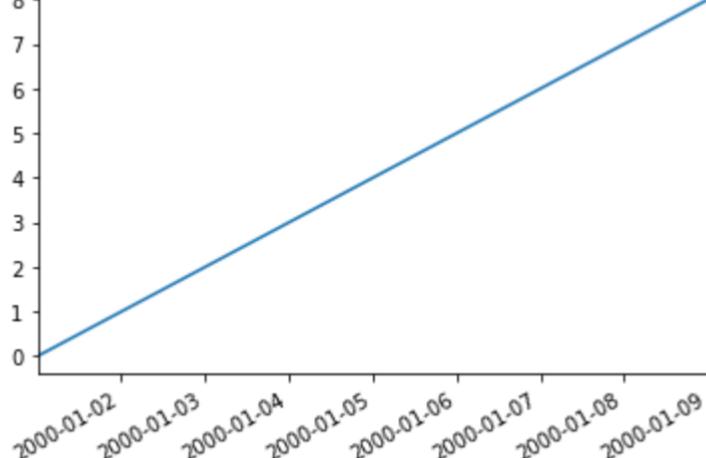
- 时间序列常用操作: 对数据按照时间进行迁移
- 迁移数据: df.shift(periods=1, freq=None, axis=0, fill_value=None)
- 迁移索引: df.tshift(periods=1, freq=None, axis=0)

```
In [194]: 1 index = pd.date_range('1/1/2000', periods=9, freq='D')
2 series = pd.Series(range(9), index=index)
3 series
```

```
Out[194]: 2000-01-01    0
2000-01-02    1
2000-01-03    2
2000-01-04    3
2000-01-05    4
2000-01-06    5
2000-01-07    6
2000-01-08    7
2000-01-09    8
Freq: D, dtype: int64
```

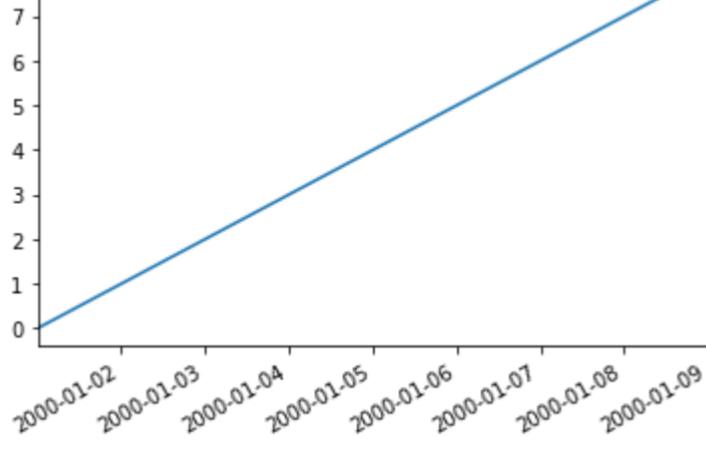
```
In [195]: 1 series.shift(1, freq='T').plot()
```

```
Out[195]: <matplotlib.axes._subplots.AxesSubplot at 0x1e0d61ef88>
```



```
In [196]: 1 series.tshift(1, freq='T').plot()
```

```
Out[196]: <matplotlib.axes._subplots.AxesSubplot at 0x1e0d621f788>
```



9 数据清洗

数据清洗方式:

- 获取某列的唯一值: Series.unique()
- 每个值出现次数: Series.value_counts()
- 删除指定行列: pdata.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')
- 去重: pdata.drop_duplicates(subset=None, keep='first', inplace=False)

9.1 产生数据:

```
In [120]: 1 #准备数据
2 import pandas as pd
3 import numpy as np
4 #注意这里是字符串
5 ts = ['2019-03-25', '2019-03-26', '2019-03-26', '2019-03-26', '2019-03-29', '2019-03-30', '2019-03-31',
6       '2019-04-01', '2019-04-02', '2019-04-03', '2019-04-04', '2019-05-01', '2019-04-05']
7 values = np.arange(len(ts))
8 pdata = pd.DataFrame({'ts':ts, 'values':values})
```

9.2 唯一值与数值出现次数

数据分析之_pandas

1 Pandas
2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法:
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [121]: 1 pdata.ts.unique()
Out[121]: array(['2019-03-25', '2019-03-26', '2019-03-29', '2019-03-30',
       '2019-03-31', '2019-04-01', '2019-04-02', '2019-04-03',
       '2019-04-04', '2019-05-01', '2019-04-05'], dtype=object)

In [122]: 1 pdata.ts.value_counts()
Out[122]: 2019-03-26    3
2019-04-01    1
2019-05-01    1
2019-04-02    1
2019-03-29    1
2019-03-25    1
2019-04-04    1
2019-03-31    1
2019-03-30    1
2019-04-05    1
2019-04-03    1
Name: ts, dtype: int64
```

9.3 删除指定行列

```
In [ ]: 1 #删除单行
2 print(pdata.drop(0))
3 #删除多行
4 print(pdata.drop(index=[1, 2, 3]))
5 #删除列
6 print(pdata.drop(columns='ts'))
7 #删除index为0值, 删除列为ts的值
8 pdata.drop(index=0, columns='ts')
9
```

9.4 去重

pdata的数据中, 有重复日期: 2019-03-26, 问题

- 保留第一次出现记录
- 保留最后一次出现记录

```
In [ ]: 1 #保留第一次
2 pdata.drop_duplicates(subset='ts')
3 #保留最后一次
4 pdata.drop_duplicates(subset='ts', keep='last')
```

```
In [129]: 1 #保留第一次数据, 生成boolean索引, 为True的为需要删除数据
2 bindex = pdata.duplicated(subset='ts')
```

```
In [130]: 1 pdata[bindex==False]
```

```
Out[130]:
ts  values
0  2019-03-25  0
1  2019-03-26  1
4  2019-03-29  4
5  2019-03-30  5
6  2019-03-31  6
7  2019-04-01  7
8  2019-04-02  8
9  2019-04-03  9
10 2019-04-04  10
11 2019-05-01  11
12 2019-04-05  12
```

10 数据合并

目的: 根据需求, 合并多个数据集

- merge:
- join:
- concat:

10.1 merge

将不同数据集根据指定字段进行合并得到新的数据集。

- pd.merge(left,right,how='inner',on=None,left_on=None,right_on=None,left_index=False,right_index=False,sort=False,suffixes=('_x', '_y'),copy=True,indicator=False,validate=None)

主要参数:

参数	说明
left/right	数据集
how	合并方式
on	索引或者column
left_on/right_on	数据连接, 左右列名称
left_index/right_index	使用左右索引合并
sort	是否根据连接键排序
suffixes	合并后, 相同列名后缀

合并方式:

参数	说明
left	左连接
right	右连接
outer	外链接
inner	内连接

10.2 准备数据

数据分析之_pandas

- 1 Pandas
- ▼ 2 pandas基础
 - 2.1 pandas简介:
 - 2.2 pandas安装
 - 2.3 Series数据结构
 - 2.4 获取index与value
 - 2.5 将index与value转成列表
 - ▼ 2.6 dataframe
 - 2.6.1 一行一列
 - 2.6.2 多列
 - 2.6.3 设置index与columns
 - 2.6.4 使用字典创建DataFrame
 - 2.6.5 设置列标签
 - ▼ 2.7 DataFrame对象访问
 - 2.7.1 获取指定列
 - 2.7.2 loc操作
 - 2.7.3 获取指定行
 - 2.7.4 遍历DataFrame对象
 - ▼ 2.8 DataFrame修改
 - 2.8.1 修改元素
 - 2.8.2 DataFrame插入列
 - 2.8.3 DataFrame插入行
 - ▼ 3 pandas数据导入与保存
 - 3.1 数据导入
 - 3.1.1 读取excel文件
 - 3.1.2 读取csv文件:
 - 3.2 保存文件
 - ▼ 4 缺失数据处理
 - 4.1 缺失值与空值
 - 4.2 缺失值判断
 - 4.3 判断是否有缺失值
 - 4.4 缺省值处理方式
 - 4.5 缺省值过滤
 - 4.6 删除缺省值
 - 4.7 缺失值填充
 - 4.8 插入均值, 中位数, 最大值, :
 - ▼ 5 数据清洗
 - 5.1 准备数据
 - 5.2 获取指定列
 - 5.3 获取指定多列
 - 5.4 根据指定条件获取数据
 - 5.5 根据指定多个条件获取数据
 - 5.6 根据集合获取数据
 - 5.7 根据数据排序
 - ▼ 6 pandas汇总与描述性统计
 - 6.1 产生数据:
 - 6.2 基本描述与计算
 - 6.3 分位数
 - ▼ 7 索引/多级索引
 - 7.1 准备数据
 - 7.2 设置索引
 - 7.3 多级索引
 - 7.4 通过多级索引取值
 - 7.5 索引交换
 - 7.6 行列变换
 - ▼ 8 时间与时间序列
 - 8.1 时间戳
 - 8.2 时间索引
 - 8.3 周期
 - 8.4 时间索引
 - 8.5 时间差值
 - 8.6 重采样
 - 8.7 时间迁移
 - ▼ 9 数据清洗
 - 9.1 产生数据:
 - 9.2 唯一值与数值出现次数
 - 9.3 删除指定行列
 - 9.4 去重
 - ▼ 10 数据合并
 - 10.1 merge
 - 10.2 准备数据
 - 10.3 数据合并
 - 10.4 join方法:
 - 10.5 concat
 - ▼ 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace替换
 - 11.5 agg聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter过滤
 - ▼ 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
 - ▼ 12.8 str相关方法
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
 - ▼ 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法:

```
In [1]: 1 import pandas as pd
          2 import numpy as np
          3 n1 = list('ABCE')
          4 n2 = list('ABCD')
          5 d1 = [90, 80, 100, 69]
          6 d2 = [95, 78, 96, 72]
          7 c1 = ['001', '001', '002', '002']
          8 c2 = ['001', '001', '002', '002']
          9
         10 df1 = pd.DataFrame({'name': n1, 'math': d1, 'class': c1})
         11 df2 = pd.DataFrame({'name': n2, 'chinese': d2, 'class': c2, 'pname': n1})
```

```
In [2]: 1 df1
```

```
Out[2]:   name  math  class
```

0	A	90	001
1	B	80	001
2	C	100	002
3	E	69	002

```
In [3]: 1 df2
```

```
Out[3]:   name  chinese  class  pname
```

0	A	95	001	A
1	B	78	001	B
2	C	96	002	C
3	D	72	002	E

10.3 数据合并

```
In [4]: 1 #默认按索引合并
          2 pd.merge(df1, df2)
```

```
Out[4]:   name  math  class  chinese  pname
```

0	A	90	001	95	A
1	B	80	001	78	B
2	C	100	002	96	C

```
In [5]: 1 #按class合并, 根据class, 两两组合, 但是对于当前成绩来说, 我们希望以名称进行合并
          2 pd.merge(df1, df2, on = 'class')
          3 #
```

```
Out[5]:   name_x  math  class  name_y  chinese  pname
```

0	A	90	001	A	95	A
1	A	90	001	B	78	B
2	B	80	001	A	95	A
3	B	80	001	B	78	B
4	C	100	002	C	96	C
5	C	100	002	D	72	E
6	E	69	002	C	96	C
7	E	69	002	D	72	E

```
In [6]: 1 #按照name进行合并, 注意class_x, class_y, 默认方式, 按照name取交集, 然后合并
          2 pd.merge(df1, df2, on = 'name')
```

```
Out[6]:   name  math  class_x  chinese  class_y  pname
```

0	A	90	001	95	001	A
1	B	80	001	78	001	B
2	C	100	002	96	002	C

```
In [7]: 1 #修改合并方式, 取并集, 会产生缺失数据
          2 pd.merge(df1, df2, on = 'name', how='outer')
```

```
Out[7]:   name  math  class_x  chinese  class_y  pname
```

0	A	90.0	001	95.0	001	A
1	B	80.0	001	78.0	001	B
2	C	100.0	002	96.0	002	C
3	E	69.0	002	NaN	NaN	NaN
4	D	NaN	NaN	72.0	002	E

```
In [8]: 1 #指定left, 以left为主
          2 pd.merge(df1, df2, on = 'name', how='left')
```

```
Out[8]:   name  math  class_x  chinese  class_y  pname
```

0	A	90	001	95.0	001	A
1	B	80	001	78.0	001	B
2	C	100	002	96.0	002	C
3	E	69	002	NaN	NaN	NaN

```
In [9]: 1 #指定right, 以right为主
          2 pd.merge(df1, df2, on = 'name', how='right')
```

```
Out[9]:   name  math  class_x  chinese  class_y  pname
```

0	A	90.0	001	95	001	A
1	B	80.0	001	78	001	B
2	C	100.0	002	96	002	C
3	D	NaN	NaN	72	002	E

数据分析之_pandas

- 1 Pandas
- 2 pandas基础
 - 2.1 pandas简介:
 - 2.2 pandas安装
 - 2.3 Series数据结构
 - 2.4 获取index与value
 - 2.5 将index与value转成列表
- 2.6 dataframe
 - 2.6.1 一行一列
 - 2.6.2 多列
 - 2.6.3 设置index与columns
 - 2.6.4 使用字典创建DataFrame
 - 2.6.5 设置列标签
- 2.7 DataFrame对象访问
 - 2.7.1 获取指定列
 - 2.7.2 loc操作
 - 2.7.3 获取指定行
 - 2.7.4 遍历DataFrame对象
- 2.8 DataFrame修改
 - 2.8.1 修改元素
 - 2.8.2 DataFrame插入列
 - 2.8.3 DataFrame插入行
- 3 pandas数据导入与保存
 - 3.1 数据导入
 - 3.1.1 读取excel文件
 - 3.1.2 读取csv文件:
 - 3.2 保存文件
- 4 缺失数据处理
 - 4.1 缺失值与空值
 - 4.2 缺失值判断
 - 4.3 判断是否有缺失值
 - 4.4 缺省值处理方式
 - 4.5 缺省值过滤
 - 4.6 删除缺省值
 - 4.7 缺失值填充
 - 4.8 插入均值, 中位数, 最大值, :
- 5 数据清洗
 - 5.1 准备数据
 - 5.2 获取指定列
 - 5.3 获取指定多列
 - 5.4 根据指定条件获取数据
 - 5.5 根据指定多个条件获取数据
 - 5.6 根据集合获取数据
 - 5.7 根据数据排序
- 6 pandas汇总与描述性统计
 - 6.1 产生数据:
 - 6.2 基本描述与计算
 - 6.3 分位数
- 7 索引/多级索引
 - 7.1 准备数据
 - 7.2 设置索引
 - 7.3 多级索引
 - 7.4 通过多级索引取值
 - 7.5 索引交换
 - 7.6 行列变换
- 8 时间与时间序列
 - 8.1 时间戳
 - 8.2 时间索引
 - 8.3 周期
 - 8.4 时间索引
 - 8.5 时间差值
 - 8.6 重采样
 - 8.7 时间迁移
- 9 数据清洗
 - 9.1 产生数据:
 - 9.2 唯一值与数值出现次数
 - 9.3 删除指定行列
 - 9.4 去重
- 10 数据合并
 - 10.1 megre
 - 10.2 准备数据
 - 10.3 数据合并
 - 10.4 join方法:
 - 10.5 concat
- 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace替换
 - 11.5 agg聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter: 过滤
- 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
 - 12.8 str相关方法:
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
- 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法:

```
In [10]: 1 #实际工作中, 数据集列名称可能不能, 需要制定不同列名进行合并
          2 pd.merge(df1, df2, left_on = 'name', right_on='pname')
```

```
Out[10]:   name_x  math  class_x  name_y  chinese  class_y  pname
0         A    90     001      A      95     001      A
1         B    80     001      B      78     001      B
2         C   100     002      C      96     002      C
3         E    69     002      D      72     002      E
```

10.4 join方法:

join方法: DataFrame对象方法, 与megre方法类似, how的方式默认为left

- df1.join(other, on=None, how='left', lsuffix='', rsuffix='', sort=False)

主要参数:

参数	说明
on	指定列
how	拼接方式
lsuffix	left后缀
rsuffix	right后缀

注意: 合并时候如果有相同列名, 需要指定lsuffix, rsuffix, 用于区分合并后列;

```
In [18]: 1
          2 df1.join(df2, lsuffix='_x', rsuffix='_y')
```

```
Out[18]:   name_x  math  class_x  name_y  chinese  class_y  pname
0         A    90     001      A      95     001      A
1         B    80     001      B      78     001      B
2         C   100     002      C      96     002      C
3         E    69     002      D      72     002      E
```

10.5 concat

concat: 根据设置轴与条件, 将两个数据进行拼接

- pd.concat(objs, axis=0, join='outer', join_axes=None, ignore_index=False, keys=None, levels=None, names=None...)

参数说明:

参数	说明
objs	Series列表, DataFrame列表, 字典列表
axis	0:index, 1:columns
join	inner:交集, outer: 并集
ignore_index	True:不使用并置轴上的索引值
join_axes	Index对象列表
keys	序列
levels	多级索引的特定值
names	list,层级索引名称

```
In [27]: 1 #按axis=0, 按列进行拼接
          2 pd.concat([df1, df2], sort=True)
```

```
Out[27]:   chinese  class  math  name  pname
0       NaN    001  90.0     A    NaN
1       NaN    001  80.0     B    NaN
2       NaN    002 100.0     C    NaN
3       NaN    002  69.0     E    NaN
0      95.0    001  NaN     A     A
1      78.0    001  NaN     B     B
2      96.0    002  NaN     C     C
3      72.0    002  NaN     D     E
```

```
In [37]: 1 #按axis=0, 按列进行拼接, 并设置key, 结果: 多级索引
          2 pd.concat([df1, df2], sort=True, keys=['p1', 'p2'])
```

```
Out[37]:   chinese  class  math  name  pname
0       NaN    001  90.0     A    NaN
p1    1       NaN    001  80.0     B    NaN
      2       NaN    002 100.0     C    NaN
      3       NaN    002  69.0     E    NaN
0      95.0    001  NaN     A     A
p2    1      78.0    001  NaN     B     B
      2      96.0    002  NaN     C     C
      3      72.0    002  NaN     D     E
```

```
In [38]: 1 #axis=1, 按索引进行拼接
          2 pd.concat([df1, df2], axis=1)
```

```
Out[38]:   name  math  class  name  chinese  class  pname
0       A    90     001      A      95     001      A
1       B    80     001      B      78     001      B
2       C   100     002      C      96     002      C
3       E    69     002      D      72     002      E
```

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [40]: 1 #axis=1, 按索引进行拼接, 列中增加多级索引
          2 pd.concat([df1, df2], axis=1, keys=['s1', 's2'])
```

```
Out[40]:
```

	s1			s2			
	name	math	class	name	chinese	class	pname
0	A	90	001	A	95	001	A
1	B	80	001	B	78	001	B
2	C	100	002	C	96	002	C
3	E	69	002	D	72	002	E

```
In [48]: 1 #拼接数据, 根据name进行拼接
          2 t1 = df1.set_index('name')
          3 t2 = df2.set_index('name')
          4 pd.concat([t1, t2], axis=1, sort=True)
```

```
Out[48]:
```

	math	class	chinese	class	pname
A	90.0	001	95.0	001	A
B	80.0	001	78.0	001	B
C	100.0	002	96.0	002	C
D	NaN	NaN	72.0	002	E
E	69.0	002	NaN	NaN	NaN

```
In [49]: 1 #拼接数据, 根据name进行拼接, join设置为inner
          2 t1 = df1.set_index('name')
          3 t2 = df2.set_index('name')
          4 pd.concat([t1, t2], axis=1, sort=True, join='inner')
```

```
Out[49]:
```

	math	class	chinese	class	pname
	name				
A	90	001	95	001	A
B	80	001	78	001	B
C	100	002	96	002	C

11 pandas数据处理常用函数

目的: 掌握apply, agg, str等函数, 对数据灵活处理

11.1 apply函数

- 对pandas中DataFrame或者Series中每个数据进行处理
- apply方法: df.apply(func,axis=0,broadcast=None, raw=False, reduce=None, result_type=None, args=(), **kwargs)

主要参数:

参数	说明
func	处理函数, 处理一系列值
axis	轴设置

```
In [65]: 1 import pandas as pd
          2 import numpy as np
          3 n = list('ABCD')
          4 math = [90, 80, 47, 69]
          5 chinese = [95, 78, 96, 59]
          6 nclass = ['001', '001', '002', '002']
          7
          8 df = pd.DataFrame({'name':n, 'math':math, 'chinese':chinese, 'class':nclass})
          9 df
```

```
Out[65]:
```

	name	math	chinese	class
0	A	90	95	001
1	B	80	78	001
2	C	47	96	002
3	D	69	59	002

- 将成绩转成True或者False

```
In [109]: 1 df[['math', 'chinese']].apply(lambda x : x>59)
```

```
Out[109]:
```

	math	chinese
0	True	True
1	True	True
2	False	True
3	True	False

11.2 func处理对象

- func处理对象是什么?
- 是否及格对应: 'Pass'/'Failed'

- 揭秘func处理对象

```
In [116]: 1 def func(value):
          2     print(type(value))
          3     return np.mean(value)
          4 df[['math', 'chinese']].apply(func)
```

<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>

```
Out[116]: math      71.5
          chinese   82.0
          dtype: float64
```

11.3 map

- 适用于Series对象
- map方法: Series.map(arg, na_action=None)
- 对Series中的每个数值进行处理

数据分析之_pandas

1 Pandas
2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获得指定行
2.7.4 遍历DataFrame对象
2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
5 数据清洗
5.1 准备数据
5.2 获得指定列
5.3 获得指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace:替换
11.5 agg:聚合操作
11.6 transform: 处理数据
11.7 filter:过滤
12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [128]: 1 df['math'].map(lambda x: 'pass' if x > 59 else 'failed')
```

```
Out[128]: 0    pass
1    pass
2    failed
3    pass
Name: math, dtype: object
```

- 需求: 将根据成绩单生成'pass','failed'
- apply与map函数结合使用

```
In [130]: 1 func = lambda x: 'pass' if x > 59 else 'failed'
2 df[['math','chinese']].apply(lambda x : x.map(func))
```

```
Out[130]:   math  chinese
```

0	pass	pass
1	pass	pass
2	failed	pass
3	pass	failed

11.4 replace:替换

- 对当前数据集中指定数据进行替换
- 方法: df.replace(to_replace=None,value=None,inplace=False,limit=None,regex=False,method='pad')

主要参数:

参数	说明
to_replace	替换值: 字符串, 正则, 列表等
value	替换目标值
limit	次数限制
inplace	是否替换原数据
regex	使用正则, 需要设置True

- 将A替换成a

```
In [145]: 1 df.replace('A', 'a')
```

```
Out[145]:   name  math  chinese  class
0      a     90      95    001
1      B     80      78    001
2      C     47      96    002
3      D     69      59    002
```

- 多个值替换成一个值
- 将[A,B]替换成*

```
In [154]: 1 df.replace(['A', 'B'], '*')
```

```
Out[154]:   name  math  chinese  class
0      *     90      95    001
1      *     80      78    001
2      C     47      96    002
3      D     69      59    002
```

- 一组数据替换
- 列表, 列表

```
In [156]: 1 df.replace(['ABCD'], ['abcd'])
```

```
Out[156]:   name  math  chinese  class
0      a     90      95    001
1      b     80      78    001
2      c     47      96    002
3      d     69      59    002
```

- 多个值替换成不同值
- name中的A替换成a,B替换成b

```
In [152]: 1 df.replace({'A': 'a', 'B': 'b'})
```

```
Out[152]:   name  math  chinese  class
0      a     90      95    001
1      b     80      78    001
2      C     47      96    002
3      D     69      59    002
```

- 正则: 将所有字母替换成*

```
In [158]: 1 df.replace(r'[A-Z]', '*', regex=True)
```

```
Out[158]:   name  math  chinese  class
0      *     90      95    001
1      *     80      78    001
2      *     47      96    002
3      *     69      59    002
```

11.5 agg:聚合操作

- 按照设置axis对数据进行聚合操作(mean, max, ...)
- df.agg(func, axis=0, *args, **kwargs)
- axis: 0: func应用到column, 1: func应用到row

```
In [180]: 1 df[['chinese', 'math']].agg(['mean', 'std'])
```

```
Out[180]:      chinese      math  
               mean    std  
mean  82.000000  71.500000  
std   17.416467  18.448125
```

- 数据基本描述

```
In [181]: 1 df[['chinese', 'math']].describe()
```

```
Out[181]:      chinese      math  
               count  mean  std  min  25%  50%  75%  max  
count  4.000000  4.000000  
mean   82.000000  71.500000  
std    17.416467  18.448125  
min    59.000000  47.000000  
25%   73.250000  63.500000  
50%   86.500000  74.500000  
75%   95.250000  82.500000  
max   96.000000  90.000000
```

11.6 transform: 处理数据

- 根据设置方法, 对数据进行处理, 得到新的数据
- df.transform(func, axis=0, *args, **kwargs)
- 每个科目数值减去均值

```
In [186]: 1 df[['chinese', 'math']].transform(lambda x:x-x.mean())
```

```
Out[186]:      chinese      math  
               0     13.0    18.5  
               1     -4.0     8.5  
               2     14.0   -24.5  
               3    -23.0    -2.5
```

11.7 filter: 过滤

- 根据标签过滤符合条件数据
- df.filter(items=None, like=None, regex=None, axis=None)

参数说明

参数	说明
items	labels值, 类似列表
like	标签模糊匹配
regex	正则表达式匹配
axis	设置轴

- 标签

```
In [198]: 1 df.filter(['name'])
```

```
Out[198]:      name  
               0     A  
               1     B  
               2     C  
               3     D
```

```
In [200]: 1 df.filter(['name', 'class'])
```

```
Out[200]:      name  class  
               0     A    001  
               1     B    001  
               2     C    002  
               3     D    002
```

- 设置like: like in label

```
In [205]: 1 df.filter(like = 'n')
```

```
Out[205]:      name  chinese  
               0     A      95  
               1     B      78  
               2     C      96  
               3     D      59
```

- 正则表达式
- 获取e结尾的列

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :)
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace: 替换
11.5 agg: 聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

数据分析之_pandas	1 Pandas
	2 pandas基础
	2.1 pandas简介:
	2.2 pandas安装
	2.3 Series数据结构
	2.4 获取index与value
	2.5 将index与value转成列表
	2.6 dataframe
	2.6.1 一行一列
	2.6.2 多列
	2.6.3 设置index与columns
	2.6.4 使用字典创建DataFrame
	2.6.5 设置列标签
	2.7 DataFrame对象访问
	2.7.1 获取指定列
	2.7.2 loc操作
	2.7.3 获取指定行
	2.7.4 遍历DataFrame对象
	2.8 DataFrame修改
	2.8.1 修改元素
	2.8.2 DataFrame插入列
	2.8.3 DataFrame插入行
	3 pandas数据导入与保存
	3.1 数据导入
	3.1.1 读取excel文件
	3.1.2 读取csv文件:
	3.2 保存文件
	4 缺失数据处理
	4.1 缺失值与空值
	4.2 缺失值判断
	4.3 判断是否有缺失值
	4.4 缺省值处理方式
	4.5 缺省值过滤
	4.6 删除缺省值
	4.7 缺失值填充
	4.8 插入均值, 中位数, 最大值, ...
	5 数据清洗
	5.1 准备数据
	5.2 获取指定列
	5.3 获取指定多列
	5.4 根据指定条件获取数据
	5.5 根据指定多个条件获取数据
	5.6 根据集合获取数据
	5.7 根据数据排序
	6 pandas汇总与描述性统计
	6.1 产生数据:
	6.2 基本描述与计算
	6.3 分位数
	7 索引/多级索引
	7.1 准备数据
	7.2 设置索引
	7.3 多级索引
	7.4 通过多级索引取值
	7.5 索引交换
	7.6 行列变换
	8 时间与时间序列
	8.1 时间戳
	8.2 时间索引
	8.3 周期
	8.4 时间索引
	8.5 时间差值
	8.6 重采样
	8.7 时间迁移
	9 数据清洗
	9.1 产生数据:
	9.2 唯一值与数值出现次数
	9.3 删除指定行列
	9.4 去重
	10 数据合并
	10.1 merge
	10.2 准备数据
	10.3 数据合并
	10.4 join方法:
	10.5 concat
	11 pandas数据处理常用函数
	11.1 apply函数
	11.2 func处理对象
	11.3 map
	11.4 replace替换
	11.5 agg聚合操作
	11.6 transform: 处理数据
	11.7 filter: 过滤
	12 分组处理
	12.1 groupby分组
	12.2 分组基本操作
	12.3 聚合操作(Aggregations)
	12.4 transform
	12.5 filter
	12.6 cut分组
	12.7 透视表
	12.8 str相关方法
	12.8.1 字符串类似方法:
	12.8.2 正则类似方法
	13 pandas可视化:
	13.1 基本使用
	13.2 plot中可视化方法:

```
In [206]: 1 df.filter(regex=r'.*e')
Out[206]:   name  chinese
            0     A      95
            1     B      78
            2     C      96
            3     D      59
```

```
In [ ]: 1
```

```
In [ ]: 1
```

```
In [ ]: 1
```

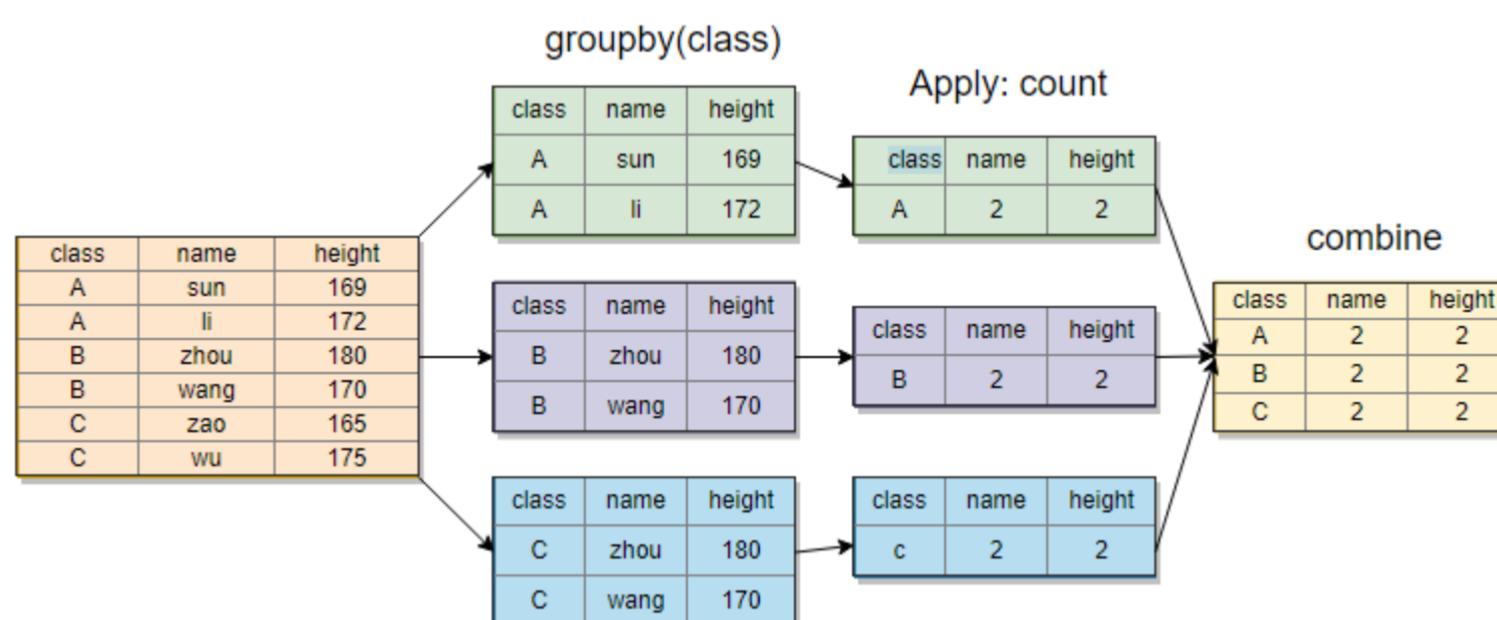
```
In [188]: 1 df.filter?
In [193]: 1 df
```

```
Out[193]:   name  math  chinese  class
            0     A    90       95    001
            1     B    80       78    001
            2     C    47       96    002
            3     D    69       59    002
```

```
In [ ]: 1
```

12 分组处理

目的: 根据指定条件, 对数据进行分组, 然后在依据分组进行计算
 例如: 统计每天活跃用户总数, 用户每天在线时长, 用户平均消费水平等;
 处理过程:



```
In [52]: 1
```

```
In [1]: 1 import pandas as pd
2 classname = ['001', '001', '002', '002', '003', '003']
3 name = ['sun', 'li', 'zhou', 'wang', 'zao', 'wu']
4 height = [169, 172, 180, 170, 165, 175]
5 weights = [61, 53, 75, 64, 50, 58]
6 df = pd.DataFrame({'cname':classname, 'user':name, 'height':height, 'weights':weights})
7 df
```

```
Out[1]:   cname  user  height  weights
          0    001   sun     169      61
          1    001    li     172      53
          2    002   zhou     180      75
          3    002   wang     170      64
          4    003   zao      165      50
          5    003   wu      175      58
```

12.1 groupby分组

- 分组方法:
`df.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=False, observed=False, **kwargs)`

主要参数:

参数	说明
by	分组依据
axis	轴: 0/index, 1/columns, 默认0
group_keys	聚合输出, 以组标签作为索引, 默认True
sort	根据分组标签排序
level	多级索引, 指定索引

```
In [2]: 1 #创建DataFrameGroupBy 对象,
2 dfg = df.groupby('cname')
3 dfg.groups
```

```
Out[2]: {'001': Int64Index([0, 1], dtype='int64'),
         '002': Int64Index([2, 3], dtype='int64'),
         '003': Int64Index([4, 5], dtype='int64')}
```

数据分析之_pandas

- 1 Pandas
- ▼ 2 pandas基础
 - 2.1 pandas简介:
 - 2.2 pandas安装
 - 2.3 Series数据结构
 - 2.4 获取index与value
 - 2.5 将index与value转成列表
- ▼ 2.6 dataframe
 - 2.6.1 一行一列
 - 2.6.2 多列
 - 2.6.3 设置index与columns
 - 2.6.4 使用字典创建DataFrame
 - 2.6.5 设置列标签
- ▼ 2.7 DataFrame对象访问
 - 2.7.1 获取指定列
 - 2.7.2 loc操作
 - 2.7.3 获取指定行
 - 2.7.4 遍历DataFrame对象
- ▼ 2.8 DataFrame修改
 - 2.8.1 修改元素
 - 2.8.2 DataFrame插入列
 - 2.8.3 DataFrame插入行
- ▼ 3 pandas数据导入与保存
 - 3.1 数据导入
 - 3.1.1 读取excel文件
 - 3.1.2 读取csv文件:
 - 3.2 保存文件
- ▼ 4 缺失数据处理
 - 4.1 缺失值与空值
 - 4.2 缺失值判断
 - 4.3 判断是否有缺失值
 - 4.4 缺省值处理方式
 - 4.5 缺省值过滤
 - 4.6 删除缺省值
 - 4.7 缺失值填充
 - 4.8 插入均值, 中位数, 最大值, ...
- ▼ 5 数据清洗
 - 5.1 准备数据
 - 5.2 获取指定列
 - 5.3 获取指定多列
 - 5.4 根据指定条件获取数据
 - 5.5 根据指定多个条件获取数据
 - 5.6 根据集合获取数据
 - 5.7 根据数据排序
- ▼ 6 pandas汇总与描述性统计
 - 6.1 产生数据:
 - 6.2 基本描述与计算
 - 6.3 分位数
- ▼ 7 索引/多级索引
 - 7.1 准备数据
 - 7.2 设置索引
 - 7.3 多级索引
 - 7.4 通过多级索引取值
 - 7.5 索引交换
 - 7.6 行列变换
- ▼ 8 时间与时间序列
 - 8.1 时间戳
 - 8.2 时间索引
 - 8.3 周期
 - 8.4 时间索引
 - 8.5 时间差值
 - 8.6 重采样
 - 8.7 时间迁移
- ▼ 9 数据清洗
 - 9.1 产生数据:
 - 9.2 唯一值与数值出现次数
 - 9.3 删除指定行列
 - 9.4 去重
- ▼ 10 数据合并
 - 10.1 merge
 - 10.2 准备数据
 - 10.3 数据合并
 - 10.4 join方法:
 - 10.5 concat
- ▼ 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace:替换
 - 11.5 agg:聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter:过滤
- ▼ 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
- ▼ 12.8 str相关方法
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
- ▼ 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法:

In [3]: ▾ 1 #分组统计
2 dfg.count()

Out[3]:

cname			
001	2	2	2
002	2	2	2
003	2	2	2

```
In [4]: ▾ 1 #根据多列进行分组:  
2 dfg = df.groupby(['cname', 'height'])  
3 dfg.groups
```

```
Out[4]: {('001', 169): Int64Index([0], dtype='int64'),
         ('001', 172): Int64Index([1], dtype='int64'),
         ('002', 170): Int64Index([3], dtype='int64'),
         ('002', 180): Int64Index([2], dtype='int64'),
         ('003', 165): Int64Index([4], dtype='int64'),
         ('003', 175): Int64Index([5], dtype='int64')}
```

In [5]: ▾ 1 #统计结果为多级索引
2 dfg.count()

Out[5]:

cname	height		
001	169	1	1
	172	1	1
002	170	1	1
	180	1	1
003	165	1	1
	175	1	1

遍历分组：

```
In [6]: ▾ 1 for index, group in dfg:  
      2     print(index, '\n', group)
```

	cname	user	height	weights
0	001	sun	169	61
('001', 172)				
	cname	user	height	weights
1	001	li	172	53
('002', 170)				
	cname	user	height	weight
3	002	wang	170	64
('002', 180)				
	cname	user	height	weight
2	002	zhou	180	75
('003', 165)				
	cname	user	height	weights
4	003	zao	165	50
('003', 175)				
	cname	user	height	weights
5	003	wu	175	58

12.2 分组基本操作

- 选取分组
 - get_group(name, obj=None)

```
In [14]:      1 dfg = df.groupby('cname')  
              2 dfg.get_group('001')
```

Out[14]:

87 / 100

In [15]: 1 dfg.describe()

• 517

001 2.0 170.5

002 2.0 175.0

500 21.5 110.0 Nov. 1983

- 获取某列描述

In [16]: 1 dfg['height'].describe()

Table 1. Summary of the main characteristics of the study population.

cname

001 2.0 170.5 2.121320

002 2.0 175.0 7.071068

600 2.0 170.0 7.071060

113, 113-01

```
In [17]: 1 print(dfg['height'].max())
          2 print(dfg['height'].min())
          3 print(dfg['height'].mean())
```

```
cname
001    172
002    180
003    175
Name: height, dtype: int64
cname
001    169
002    170
003    165
Name: height, dtype: int64
cname
001    170.5
002    175.0
003    170.0
Name: height, dtype: float64
```

12.3 聚合操作(Aggregations)

分组得到groupby对象，可以通过聚合函数对其操作，获取聚合结果；直白理解：对分组数据，进行处理；

```
In [21]: 1 import pandas as pd
          2 import numpy as np
          3 classname = ['001', '001', '002', '002', '003', '003']
          4 name = ['sun', 'li', 'zhou', 'wang', 'zao', 'wu']
          5 height = [169, 172, 180, 170, 165, 175]
          6 weights = [61, 53, 75, 64, 50, 58]
          7 df = pd.DataFrame({'cname':classname, 'user':name, 'height':height, 'weights':weights})
          8 dfg = df.groupby('cname')
```

- 根据指定方式进行计算
- 一次获取最大值，均值，标准差，数量

```
In [22]: 1 dfg.agg(['max', 'mean', np.std, 'count'])
```

cname	height				weights			
	max	mean	std	count	max	mean	std	count
001	172	170.5	2.121320	2	61	57.0	5.656854	2
002	180	175.0	7.071068	2	75	69.5	7.778175	2
003	175	170.0	7.071068	2	58	54.0	5.656854	2

12.4 transform

- 作用：将分组数据处理，得到一组新的数据
- 方法：dfg.transform(func, *args, **kwargs)
- 场景：分组数据中，与分组均值差。

```
In [30]: 1 #每个班级平均身高，体重
          2 dfg.transform('mean')
```

	height	weights
0	170.5	57.0
1	170.5	57.0
2	175.0	69.5
3	175.0	69.5
4	170.0	54.0
5	170.0	54.0

```
In [32]: 1 #每个班级 学生-班级平均值(身高体重)
          2 dfg.transform(lambda x: x-x.mean())
```

	height	weights
0	-1.5	4.0
1	1.5	-4.0
2	5.0	5.5
3	-5.0	-5.5
4	-5.0	-4.0
5	5.0	4.0

12.5 filter

- 过滤数据
- dfg.filter(func, dropna=True, *args, **kwargs)：根据过滤条件返回分组数据

```
In [39]: 1 #返回分组身高均值大于171的数据
          2 dfg['height'].filter(lambda x: np.mean(x)> 171)
```

	height
2	180
3	170

```
In [41]: 1 dfg.count()
```

	user	height	weights
cname	001	2	2
001	2	2	2
002	2	2	2
003	2	2	2

12.6 cut分组

- 根据设置区间进行分段汇总
- 方法：pd.cut(x,bins,right=True,labels=None,retbins=False,precision=3,include_lowest=False,duplicates='raise')

主要参数：

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一维列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, ...
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

参数	说明
x	array-like, 一维数据
bins	分组数据, 类似: [1,2,3,4]
right	闭区间, True:(x1,x2],(x3,x3]
labels	标签
retbins	是否返回分段数据
duplicates	重复分段默认除法异常

```
In [231]: 1 #准备数据, 统计70~79, 80~89, 90~100三个范围对应数量
2 df = pd.Series(np.random.randint(70, 100, size = 10))
3 #统计70~79, 80~89, 90~100三个范围对应数量
4 r = pd.cut(df, [70, 80, 90, 101], right=False)
5 r
```

```
Out[231]: 0 [90, 101)
1 [70, 80)
2 [90, 101)
3 [80, 90)
4 [70, 80)
5 [80, 90)
6 [70, 80)
7 [80, 90)
8 [90, 101)
9 [90, 101)
dtype: category
Categories (3, interval[int64]): [[70, 80) < [80, 90) < [90, 101]]
```

```
In [225]: 1 #统计70~79, 80~89, 90~100三个范围对应数量, 设置标签
2 r = pd.cut(df, [70, 80, 90, 101], right=False, labels=['70+', '80+', '100+'])
3 r.groupby(r).count()
```

```
Out[225]: 70+    9
80+    10
100+   11
dtype: int64
```

```
In [233]: 1 #返回retbins
2 r,bins= pd.cut(df, [70, 80, 90, 101], right=False, labels=['70+', '80+', '100+'], retbins=True)
3 bins
```

```
Out[233]: array([ 70,  80,  90, 101])
```

12.7 透视表

- 透视表是一种可以对数据动态排布并且分类汇总的表格格式, 使用方式与groupby类似, 但是比其简单;
- 方法: pd.pivot_table(data,values=None,index=None,columns=None,aggfunc='mean',fill_value=None,margins=False,...)

主要参数:

参数	说明
data	数据
values	用于计算数据项
index	行分组键(column,grouper,array)
columns	列分组键(column,grouper,array)
aggfunc	聚合函数或函数列表, 字典函数
fill_value	设置缺省值
dropna	删除缺省值, 默认True
margins	是否增加统计列
margins_name	新增统计列列名

```
In [104]: 1 # 导入数据
2 #地址: https://github.com/mwaskom/seaborn-data/blob/master/titanic.csv
3 import pandas as pd
```

```
In [118]: 1 path = r'E:\data\seaborn-data\titanic.csv'
2 df = pd.read_csv(path)
```

```
In [146]: 1 #根据class等级统计男女获救的比例
2 #      1等仓, 2等仓, ...
3 #女性获救比例, x x ...
4 #男性获救比例 x x ...
5 df.pivot_table(values='survived', index='sex', columns=['class'])
```

```
Out[146]: class   First   Second   Third
          sex
female  0.968085  0.921053  0.500000
male   0.368852  0.157407  0.135447
```

```
In [153]: 1 #根据class等级统计男女获救的比例, 统计所有值, 并重命名
2 #设置margins, 统计所有男女获救比例
3 #      1等仓, 2等仓, ...
4 #女性获救比例, x x ...
5 #男性获救比例 x x ...
6 df.pivot_table(values='survived', index='sex', columns=['class'], margins=True, margins_name='all_mean')
```

```
Out[153]: class   First   Second   Third all_mean
          sex
female  0.968085  0.921053  0.500000  0.742038
male   0.368852  0.157407  0.135447  0.188908
all_mean 0.629630  0.472826  0.242363  0.383838
```

数据分析之_pandas

1 Pandas
2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter: 过滤
12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

```
In [154]: 1 #根据class等级与年龄段, 统计男女获救的比例
2 cuts = pd.cut(df.age, [0, 18, 100])
3 df.pivot_table(values='survived', index=['sex', cuts], columns=['class'])
```

```
Out[154]:
```

	class	First	Second	Third
sex	age			
female	(0, 18]	0.909091	1.000000	0.511628
	(18, 100]	0.972973	0.900000	0.423729
male	(0, 18]	0.800000	0.600000	0.215686
	(18, 100]	0.375000	0.071429	0.133663

```
In [155]: 1 #根据class等级与年龄段, 统计男女获救的比例, 数量, 并统计所有值
2 cuts = pd.cut(df.age, [0, 18, 100])
3 df.pivot_table(values='survived', index=['sex'], columns=['class'], aggfunc=['mean', 'count'], margins=True)
```

```
Out[155]:
```

	mean	count						
class	First	Second	Third	All	First	Second	Third	All
sex								
female	0.968085	0.921053	0.500000	0.742038	94	76	144	314
male	0.368852	0.157407	0.135447	0.188908	122	108	347	577
All	0.629630	0.472826	0.242363	0.383838	216	184	491	891

12.8 str相关方法

str方法使Series对象内置方法, 用于对字符串处理, 与字符串相关方法类似;

12.8.1 字符串类似方法:

- cat(others=None, sep=None, na_rep=None, join=None): 拼接字符串
- split(pat=None, n=-1, expand=False): 切分字符串
- get(i): 获取指定位置的字符串
- join(sep): 字符串拼接
- find(sub, start=0, end=None): 查找, 返回第一次出现子集位置
- contains(pat, case=True, flags=0, na=nan, regex=True): 判断是否包含指定的值
- replace(pat, repl, n=-1, case=None, flags=0, regex=True): 替换
- sf.str.repeat(repeats): 重复repeats次
- startswith(pat, na=nan): 判断是否已pat开头
- sf.str.endswith(pat, na=nan): 判断是否已pat结尾
- sf.str.strip(to_strip=None): 根据指定字符掐头去尾

```
In [261]: 1 sf = pd.Series(['a_1', 'b_2', 'c_3'])
2 sf.str.cat(list('ABC'), sep = '-')
```

```
Out[261]: 0 a_1-A
1 b_2-B
2 c_3-C
dtype: object
```

```
In [262]: 1 sf.str.split('_', expand=True)
```

```
Out[262]: 0 1
0 a 1
1 b 2
2 c 3
```

12.8.2 正则类似方法

- match(pat, case=True, flags=0, na=nan): 从头匹配, 符合返回True
- findall(pat, flags=0, **kwargs): 返回出符合匹配规则的所有子集
- extract(pat, flags=0, expand=True): 提取匹配
- extractall(pat, flags=0): 提取所有匹配

```
In [284]: 1 #提取有效字符,
2 sf.str.extract(r'([a-zA-Z]+)')
```

```
Out[284]: 0
0 a
1 b
2 c
```

```
In [301]: 1 #字符开头
2 sf.str.match(r'([a-zA-Z]+)')
```

```
Out[301]: 0 True
1 True
2 True
dtype: bool
```

```
In [302]: 1 #查找所有字符与数字
2 sf.str.findall(r'([a-zA-Z0-9]+)')
```

```
Out[302]: 0 [a, 1]
1 [b, 2]
2 [c, 3]
dtype: object
```

```
In [ ]: 1
```

13 pandas可视化:

pandas可以直接绘制图表, 实现基于matplotlib, 使用方式与其类似, 方法:

- df.plot(*args, **kwargs)

相关参数:

参数	说明
data	Series或者DataFrame对象
x	标签或者索引

数据分析之_pandas

1 Pandas
▼ 2 pandas基础
2.1 pandas简介:
2.2 pandas安装
2.3 Series数据结构
2.4 获取index与value
2.5 将index与value转成列表
▼ 2.6 dataframe
2.6.1 一行一列
2.6.2 多列
2.6.3 设置index与columns
2.6.4 使用字典创建DataFrame
2.6.5 设置列标签
▼ 2.7 DataFrame对象访问
2.7.1 获取指定列
2.7.2 loc操作
2.7.3 获取指定行
2.7.4 遍历DataFrame对象
▼ 2.8 DataFrame修改
2.8.1 修改元素
2.8.2 DataFrame插入列
2.8.3 DataFrame插入行
▼ 3 pandas数据导入与保存
3.1 数据导入
3.1.1 读取excel文件
3.1.2 读取csv文件:
3.2 保存文件
▼ 4 缺失数据处理
4.1 缺失值与空值
4.2 缺失值判断
4.3 判断是否有缺失值
4.4 缺省值处理方式
4.5 缺省值过滤
4.6 删除缺省值
4.7 缺失值填充
4.8 插入均值, 中位数, 最大值, :
▼ 5 数据清洗
5.1 准备数据
5.2 获取指定列
5.3 获取指定多列
5.4 根据指定条件获取数据
5.5 根据指定多个条件获取数据
5.6 根据集合获取数据
5.7 根据数据排序
▼ 6 pandas汇总与描述性统计
6.1 产生数据:
6.2 基本描述与计算
6.3 分位数
▼ 7 索引/多级索引
7.1 准备数据
7.2 设置索引
7.3 多级索引
7.4 通过多级索引取值
7.5 索引交换
7.6 行列变换
▼ 8 时间与时间序列
8.1 时间戳
8.2 时间索引
8.3 周期
8.4 时间索引
8.5 时间差值
8.6 重采样
8.7 时间迁移
▼ 9 数据清洗
9.1 产生数据:
9.2 唯一值与数值出现次数
9.3 删除指定行列
9.4 去重
▼ 10 数据合并
10.1 merge
10.2 准备数据
10.3 数据合并
10.4 join方法:
10.5 concat
▼ 11 pandas数据处理常用函数
11.1 apply函数
11.2 func处理对象
11.3 map
11.4 replace替换
11.5 agg聚合操作
11.6 transform: 处理数据
11.7 filter过滤
▼ 12 分组处理
12.1 groupby分组
12.2 分组基本操作
12.3 聚合操作(Aggregations)
12.4 transform
12.5 filter
12.6 cut分组
12.7 透视表
▼ 12.8 str相关方法
12.8.1 字符串类似方法:
12.8.2 正则类似方法
▼ 13 pandas可视化:
13.1 基本使用
13.2 plot中可视化方法:

参数	说明
y	标签或者索引
kind	绘制图像样式: line, bar, barh, hist...
figsize	大小
use_index	是否使用index作为x轴ticks
grid	是否使用栅格
legend	图例

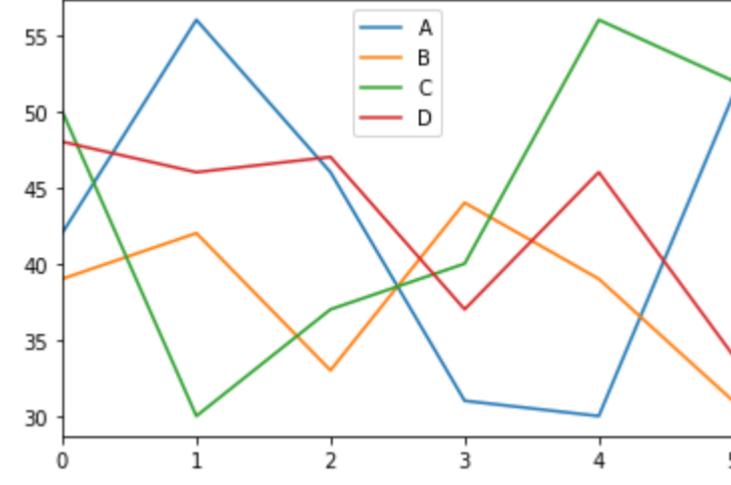
13.1 基本使用

```
In [13]: 1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 %matplotlib inline
```

```
In [ ]: 1 #数据
2 df = pd.DataFrame(np.random.randint(30, 60, size=(6, 4)), columns=list('ABCD'), index=range(6))
```

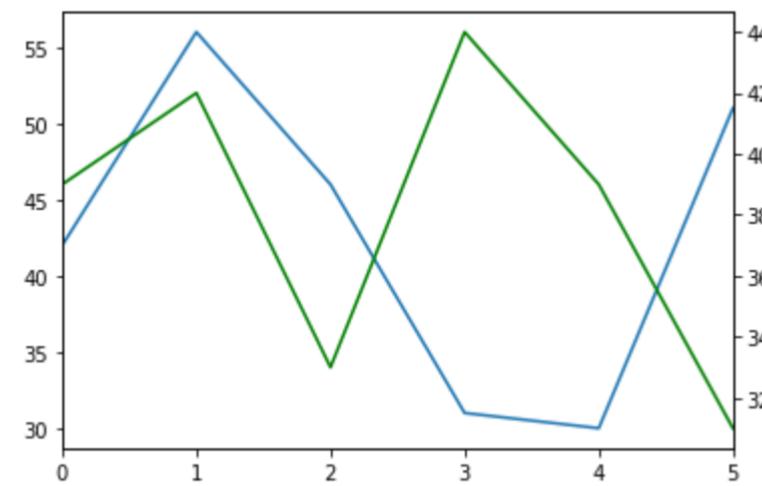
```
In [32]: 1 #绘制折线图
2 #x:index, y:默认所有columns
3 df.plot()
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0xe0cf93ba88>



```
In [43]: 1 #A使用左侧Y轴
2 df.A.plot()
3 #B使用右侧Y周
4 df.B.plot(secondary_y=True, style='g')
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0xe0d114e548>



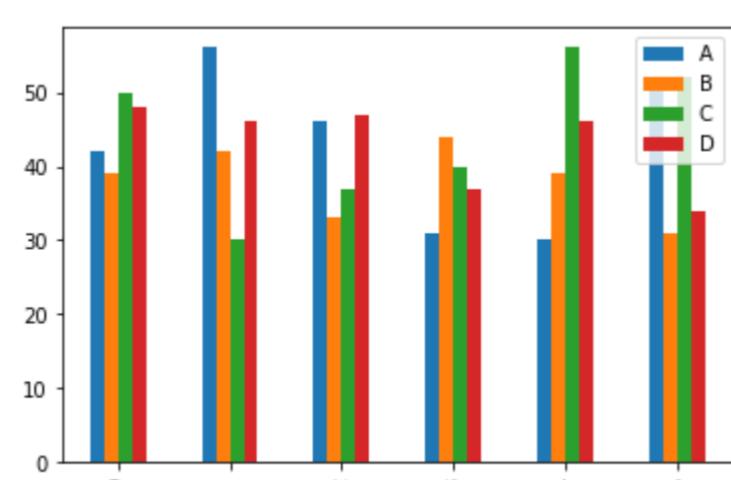
```
In [ ]: 1
```

13.2 plot中可视化方法:

- 折线图: df.plot.line(x=None, y=None, **kwargs)
- 柱状图: df.plot.bar(x=None, y=None, **kwargs)
- 条形图: df.plot.barh(x=None, y=None, **kwargs)
- 直方图: df.plot.hist(by=None, bins=10, **kwargs)
- KDE图: df.plot.kde(bw_method=None, ind=None, **kwargs)
- 饼状图: df.plot.pie(**kwargs)
- 散点图: df.plot.scatter(x, y, s=None, c=None, **kwargs)
- 箱状图: df.plot.box(by=None, **kwargs)
- 区域块状图: df.plot.area(x=None, y=None, **kwargs)

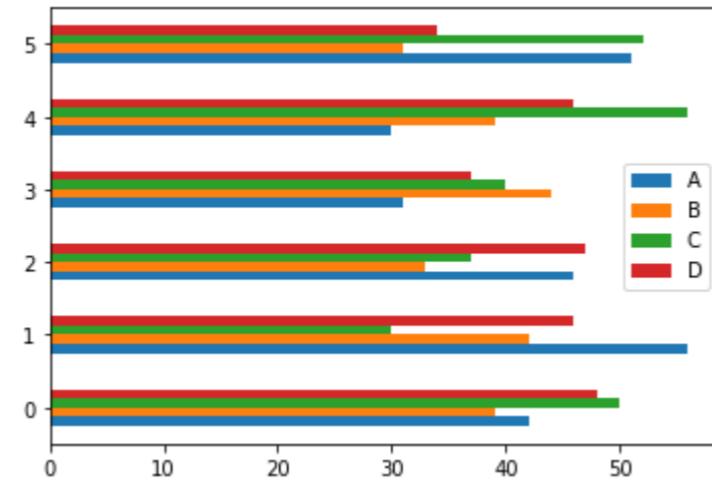
```
In [68]: 1 # 柱状图
2 df.plot.bar()
```

Out[68]: <matplotlib.axes._subplots.AxesSubplot at 0xe0d2b70e08>



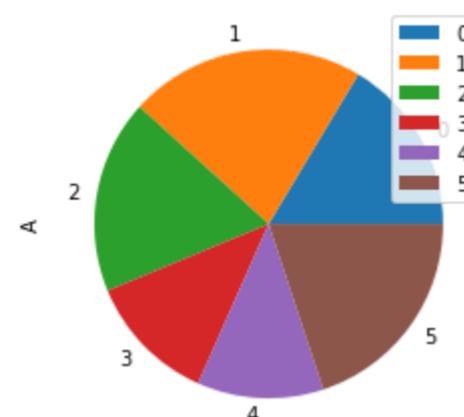
```
In [69]: 1 # 条状图  
          2 df.plot.barh()
```

Out[69]: <matplotlib.axes._subplots.AxesSubplot at 0x1e0d2c2e048>



```
In [74]: 1 # 柱状图  
          2 df.plot.pie(y='A')
```

Out[74]: <matplotlib.axes._subplots.AxesSubplot at 0x1e0d3055f88>



```
In [ ]: 1
```

数据分析之_pandas

- 1 Pandas
- ▼ 2 pandas基础
 - 2.1 pandas简介:
 - 2.2 pandas安装
 - 2.3 Series数据结构
 - 2.4 获取index与value
 - 2.5 将index与value转成列表
- ▼ 2.6 dataframe
 - 2.6.1 一行一列
 - 2.6.2 多列
 - 2.6.3 设置index与columns
 - 2.6.4 使用字典创建DataFrame
 - 2.6.5 设置列标签
- ▼ 2.7 DataFrame对象访问
 - 2.7.1 获取指定列
 - 2.7.2 loc操作
 - 2.7.3 获取指定行
 - 2.7.4 遍历DataFrame对象
- ▼ 2.8 DataFrame修改
 - 2.8.1 修改元素
 - 2.8.2 DataFrame插入列
 - 2.8.3 DataFrame插入行
- ▼ 3 pandas数据导入与保存
 - 3.1 数据导入
 - 3.1.1 读取excel文件
 - 3.1.2 读取csv文件:
 - 3.2 保存文件
- ▼ 4 缺失数据处理
 - 4.1 缺失值与空值
 - 4.2 缺失值判断
 - 4.3 判断是否有缺失值
 - 4.4 缺省值处理方式
 - 4.5 缺省值过滤
 - 4.6 删除缺省值
 - 4.7 缺失值填充
 - 4.8 插入均值, 中位数, 最大值, :
- ▼ 5 数据清洗
 - 5.1 准备数据
 - 5.2 获取指定列
 - 5.3 获取指定多列
 - 5.4 根据指定条件获取数据
 - 5.5 根据指定多个条件获取数据
 - 5.6 根据集合获取数据
 - 5.7 根据数据排序
- ▼ 6 pandas汇总与描述性统计
 - 6.1 产生数据:
 - 6.2 基本描述与计算
 - 6.3 分位数
- ▼ 7 索引/多级索引
 - 7.1 准备数据
 - 7.2 设置索引
 - 7.3 多级索引
 - 7.4 通过多级索引取值
 - 7.5 索引交换
 - 7.6 行列变换
- ▼ 8 时间与时间序列
 - 8.1 时间戳
 - 8.2 时间索引
 - 8.3 周期
 - 8.4 时间索引
 - 8.5 时间差值
 - 8.6 重采样
 - 8.7 时间迁移
- ▼ 9 数据清洗
 - 9.1 产生数据:
 - 9.2 唯一值与数值出现次数
 - 9.3 删除指定行列
 - 9.4 去重
- ▼ 10 数据合并
 - 10.1 merge
 - 10.2 准备数据
 - 10.3 数据合并
 - 10.4 join方法:
 - 10.5 concat
- ▼ 11 pandas数据处理常用函数
 - 11.1 apply函数
 - 11.2 func处理对象
 - 11.3 map
 - 11.4 replace替换
 - 11.5 agg聚合操作
 - 11.6 transform: 处理数据
 - 11.7 filter: 过滤
- ▼ 12 分组处理
 - 12.1 groupby分组
 - 12.2 分组基本操作
 - 12.3 聚合操作(Aggregations)
 - 12.4 transform
 - 12.5 filter
 - 12.6 cut分组
 - 12.7 透视表
 - ▼ 12.8 str相关方法:
 - 12.8.1 字符串类似方法:
 - 12.8.2 正则类似方法
- ▼ 13 pandas可视化:
 - 13.1 基本使用
 - 13.2 plot中可视化方法: