11.3 Pandas 数据处理

简述

> Pandas 是一个基于 Numpy 构建的 Python 模块。Pandas 模块所具有的高性能、高效率和高水平使之成为数据分析最有效的库。

Pandas 中常用的两种数据结构

- **Series**：一维数组，与 Python 基本的数据结构 List 相近。其区别是，List 中的元素可以是不同的数据类型，而 Series 中则只允许存储相同的数据类型，这样是为了更有效地使用内存，提高运算效率。
- **DataFrame**：二维的表格型数据结构。可以将 DataFrame 理解为 Series 的容器。以下的内容主要以 DataFrame 为主。

一、Pandas 安装与数据结构

安装 pandas

```
$ sudo pip3 install pandas
```

使用

```
import pandas as pd

pd.__version__
# '0.23.4'
```

Series

```
ser = pd.Series(['张三', '李四', '王五'])

"""
ser
0    张三
1    李四
2    王五
dtype: object
"""
```

```
# 修改索引
ser = pd.Series(['张三', '李四', '王五'], list(range(1, 4)))
ser = pd.Series(['张三', '李四', '王五'], index=list(range(1, 4))) # 这样会更明确

"""
1    张三
2    李四
3    王五
dtype: object
"""

# Series有个values属性，它的值是 array(['张三', '李四', '王五'], dtype=object)
ser.values
type(ser.values) # 它的类型就是 numpy.ndarray

# Series也可以运算
ser2 = pd.Series([18, 19, 17], index=range(1, 4))

"""
1    18
2    19
3    17
dtype: int64
"""

ser2 + 1

# 字典的key作为索引
data = {'beijing':9240, 'shanghai':8960, 'guangzhou':7400}
ser3 = pd.Series(data)

"""
beijing      9240
guangzhou    7400
shanghai     8960
dtype: int64
"""

ser3['beijing']
'beijing' in ser3

ser4 = pd.Series(data, index=['shanghai', 'beijing', 'chongqing'])

"""
shanghai     8960.0
beijing      9240.0
chongqing       NaN
dtype: float64
"""

ser4.to_dict()  # {'beijing': 9240.0, 'chongqing': nan, 'shanghai': 8960.0}
ser4.tolist()   # [8960.0, 9240.0, nan]
ser4.to_json()  # '{"shanghai":8960.0,"beijing":9240.0,"chongqing":null}'
```

```
ser4.to_frame() # DataFrame 方式展示
```

DataFrame

```
import numpy as np
data = np.arange(100, 109).reshape(3, -1)

"""
array([[100, 101, 102],
       [103, 104, 105],
       [106, 107, 108]])
"""

df = pd.DataFrame(data)


data = {
    'name':['jack', 'mary', 'lily'],
    'age':[19, 19, 17],
    'height': [1.68, 1.72, 1.62]
}
df = pd.DataFrame(data)

df.columns # 查看key，Index(['age', 'height', 'name'], dtype='object')

df.columns = ['age', 'height', 'username'] # 可以这样修改key

df = pd.DataFrame(data, columns = ['age', 'height', 'name', 'email'])

df.index # 查看索引，RangeIndex(start=0, stop=3, step=1)

df = pd.DataFrame(data, columns = ['age', 'height', 'name', 'email'], index=range(1, 4))
```

二、Pandas 中对数据的选取操作

```
data = {
    'name': ['张三', '李四', '王五', '赵六'],
    'age': [18, 19, 17, 20],
    'height': [1.68, 1.73, 1.62, 1.55]
}
```

```
df = pd.DataFrame(data, columns=['name', 'age', 'height'])

df['name']   # 选取name那一列数据
df.name      # 也可以这样
df[['name']] # 这样得到DataFrame格式

df[['name', 'age']]

names = df['name']
names[0]
names[0] = '田七'  # 直接修改了值，df同样被修改了

df.columns
df.columns[1:3]

df[df.columns[1:3]]

import datetime
df['year'] = datetime.datetime.now().year - df.age

df.drop('year', axis=1) # 删除

df.loc[1]
df.loc[[1, 3]]

df.index
df.index[-2:]
df.loc[df.index[-2:], ['name', 'age']]  # 取最后两条

df.shape
df.loc[df.shape[0]] = {'age':21, 'name':'无酒', 'height':1.66, 'year':0}

df2 = df.drop(2)  # drop会复制数据，而不是引用?
df2.iloc[2]  # 按顺序找，而不是索引序号
df2.iloc[1:3]
df.iat[1, 1]

df['height'] >= 1.65
df[df['height'] >= 1.65] # 选取身高大于1.65米的
df[(df['height'] >= 1.65) & (df['age'] <= 20)]
df.query('height >= 1.65 and age <= 20')

age = 20
df.query('age == @age')

df['age'].isin([18, 19])
df[df['age'].isin([18, 19])]

df.T # 转置
```

三、Pandas 加载数据

Text

示例：

```
pd.read_table('data/01.txt')
pd.read_table('data/02.txt')  # 严格按tab键分隔
pd.read_table('data/03.txt', sep=':', header=None)
pd.read_table('data/03.txt', sep=':', names=['name', 'pwd', 'uid', 'gid', 'local',
'home', 'shell'])
```

CSV 文本

示例：

```
pd.read_csv('data/04.csv')
```

Excel

> 后缀 xlsx，Microsoft Excel 2007-2013

依赖 xlrd 模块

示例：

```
pd.read_excel('data/05.xlsx')
```

Html

依赖 lxml 模块

示例：

```
tables = pd.read_html('data/06.html', header=0)
tables[0]
tables[1]

tables = pd.read_html('data/06.html', header=0, attrs={'class':'mydata'})
tables[0]
```

MySQL

依赖 pymysql 模块

示例：

```
import pymysql

conn = pymysql.connect(host='localhost', user='root', password='', db='linuxmm',
charset='utf8')

sql = 'select * from host'
data = pd.read_sql(sql, conn)
conn.close()
```

MongoDB

依赖 pymongo 模块

示例：

```
import pymongo

client = pymongo.MongoClient('localhost', 27017)

db = 'test'
table = 'stu'
cursor = client[db][table].find()

data = pd.DataFrame(list(cursor))
```

截图

Text

In [107]:
```
pd.read_table('data/01.txt')
```

Out[107]:

	email
0	jack@example.com
1	marry@example.com
2	lily@example.com
3	tom@example.com
4	joe@example.com

In [111]:
```
pd.read_table('data/02.txt')  # 严格按tab键分隔
```

Out[111]:

	name	age	email
0	Jack	18	jack@example.com
1	Marry	19	marry@example.com
2	Lily	17	lily@example.com
3	Tom	17	tom@example.com
4	Joe	20	joe@example.com

In [113]:
```
pd.read_table('data/03.txt', sep=':', header=None)
```

Out[113]:

	0	1	2	3	4	5	6
0	root	x	0	0	root	/root	/bin/bash
1	daemon	x	1	1	daemon	/usr/sbin	/usr/sbin/nologin
2	bin	x	2	2	bin	/bin	/usr/sbin/nologin
3	sys	x	3	3	sys	/dev	/usr/sbin/nologin

CSV，逗号分隔文件

In [115]:
```
pd.read_csv('data/04.csv')
```

Out[115]:

	书名	单价
0	我在未来等你	29.9
1	原则	63.0
2	半小时漫画世界史	28.9

Excel

In [116]:
```
pd.read_excel('data/05.xlsx')
```

Out[116]:

	电影名称	上映时间
0	神秘巨星	2018-01-19
1	移动迷宫3	2018-01-26
2	阿凡达2	2018-12-25

HTML

In [119]:
```
tables = pd.read_html('data/06.html', header=0)
```

In [120]:
```
tables[0]
```

Out[120]:

	排名	城市	工资
0	1	北京	9240
1	2	上海	8962
2	3	深圳	8315
3	4	广州	7409
4	5	杭州	7330

In [121]:
```
tables[1]
```

Out[121]:

	aa	bb	cc
0	11	22	33
1	44	55	66

In [124]:
```
tables = pd.read_html('data/06.html', header=0, attrs={'class':'mydata'})
```

In [125]:
```
tables
```

Out[125]:
```
[   排名  城市   工资
 0   1  北京  9240
 1   2  上海  8962
 2   3  深圳  8315
 3   4  广州  7409
 4   5  杭州  7330]
```

MySQL

In [127]:
```
import pymysql
```

In [130]:
```
conn = pymysql.connect(host='localhost', user='root', password='', db='linuxmm', charset='utf8')
```

In [131]:
```
sql = 'select * from host'
data = pd.read_sql(sql, conn)
conn.close()
```

In [132]:
```
data
```

Out[132]:

	id	tag	ip	cpu	mem	disk	stat	cdate
0	1	test01	192.168.3.100	4.0	8.0	500.0	1	2018-10-23 17:50:54
1	2	test02	192.168.3.101	4.0	8.0	1000.0	1	2018-10-23 17:50:54
2	3	席位01	6.24.34.100	NaN	NaN	NaN	0	2018-10-23 20:47:17
3	4	席位02	6.24.34.101	NaN	NaN	NaN	0	2018-10-23 20:47:33
4	5	席位03	6.24.34.102	NaN	NaN	NaN	0	2018-10-23 20:47:45
5	6	席位04	6.24.34.103	NaN	NaN	NaN	0	2018-10-23 20:47:53
6	7	席位05	6.24.34.104	NaN	NaN	NaN	0	2018-10-23 20:47:58
7	8	席位06	6.24.34.105	NaN	NaN	NaN	0	2018-10-23 20:48:06
8	9	席位07	6.24.34.106	NaN	NaN	NaN	0	2018-10-23 20:48:13
9	10	席位08	6.24.34.107	NaN	NaN	NaN	0	2018-10-23 20:48:20
10	11	席位09	6.24.34.108	NaN	NaN	NaN	0	2018-10-23 20:48:28

MongoDB

In [134]:
```
import pymongo
```

In [135]:
```
client = pymongo.MongoClient('localhost', 27017)
```

In [136]:
```
client
```

Out[136]:
```
MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)
```

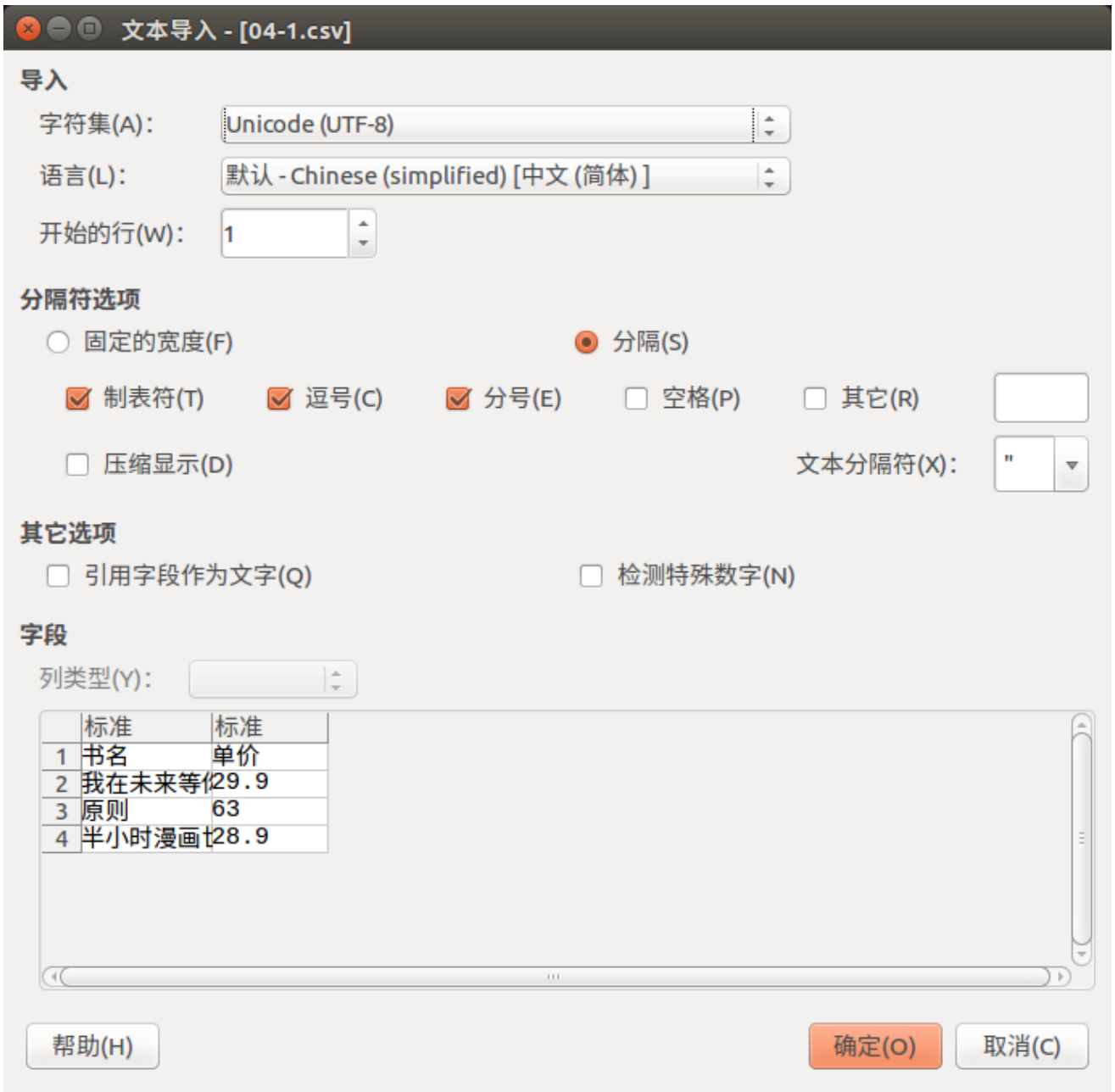
In [137]:
```
db = 'test'
table = 'stu'
cursor = client[db][table].find()
```

In [138]:
```
data = pd.DataFrame(list(cursor))
```

In [139]:
```
data
```

Out[139]:

	_id	gender	name
0	5bb5adbfc5015d63155476a5	1.0	rudy
1	5bb6c2245a8e01a7fcd6168c	0.0	tina
2	20181001	0.0	little


文本导入 - [04-1.csv]
导入
字符集(A):
Unicode (UTF-8)
语言(L):
默认 - Chinese (simplified) [中文 (简体)]
开始的行(W):
1
分隔符选项
固定的宽度(F)
分隔(S)
制表符(T)
逗号(C)
分号(E)
空格(P)
其它(R)
压缩显示(D)
文本分隔符(X):
"
其它选项
引用字段作为文字(Q)
检测特殊数字(N)
字段
列类型(Y):
标准
标准
1 书名 单价
2 我在未来等你 29.9
3 原则 63
4 半小时漫画史 28.9
帮助(H)
确定(O)
取消(C)

2017年中国各城市平均工资排行榜

排名	城市	工资
1	北京	9240
2	上海	8962
3	深圳	8315
4	广州	7409
5	杭州	7330

---

aa bb cc

11 22 33

44 55 66