从文件或数据库，网页中获取数据

**Pandas读写CSV文件或TXT文件和相关参数解释**

csv文件常用逗号和制表符为分隔符

**读csv或txt**

read\_csv(filepath\_or\_buffer,sep=',',header='infer',names=None,encoding=None)

filepath\_or\_buffer：表示文件的路径

sep：表示指定的分隔符默认为','

header：指定哪一行为列索引，默认为0(第一行数据作为列索引)

names：表示DataFrame类的列索引列表

encoding：表示指定的编码格式

df\_csv=pd.read\_csv("phone.csv",encoding='gbk')

df\_txt=pd.read\_csv("phone.txt",encoding='gbk')

**保存csv或txt**

df对象名.to\_csv(io,encoding='utf-8',mode='w',header=None,nrows=None,dtype=None)

io：表示存储路径

encoding是编码方式

header=0表示不保存列名

nrows:表示限制行数

dtype:表示各字段的数据类型

mode=‘w’ 写入模式，默认为w

r : 只能读, 必须存在, 可在任意位置读取

w : 只能写, 可以不存在, 必会擦掉原有内容从头写

a : 只能写, 可以不存在, 必不能修改原有内容, 只能在结尾追加写, 文件指针无效

r+ : 可读可写, 必须存在, 可在任意位置读写, 读与写共用同一个指针

w+ : 可读可写, 可以不存在, 必会擦掉原有内容从头写

a+ : 可读可写, 可以不存在, 必不能修改原有内容, 只能在结尾追加写, 文件指针只对读有效 (写操作会将文件指针移动到文件尾)

csv3.to\_csv('fff.csv',encoding='utf-8',index=False)

**Pandas读写EXCEL文件**

read\_excel(io,sheet\_name=0,header=0,names=None,index\_col=None)

io：表示文件的路径

sheet\_name：表示要读取的工作表(0表示第一个表，也可以表的名字,取多个用数组，全取用None)

header：指定哪一行为列索引，默认为0(第一行数据作为列索引)

names：表示DataFrame类的列索引列表

index\_col：将Excel文件的列标题作为DataFrame类的行索引

df\_excel=pd.read\_excel('aaa.xlsx',sheet\_name=2)

**保存excel**

df对象.to\_excel('aaa.xlsx',sheet\_name='页名',index=False)#保存数据

**Pandas读写JSON文件**

read\_json(path\_or\_buf=None,orient=None,encoding=None)

path\_or\_buf：表示文件路径

encoding：文件编码格式

df\_json=df.read\_json('aaaaa.json',encoding='utf8')

**Pandas读写HTML文件中的表格数据**

read\_html(io,match='.+',flavor=None,index\_col=None,encoding=None,na\_value=None)

io：表示html网页的url路径

match：表示返回与指定的正则表达式或字符串匹配的文本

flavor：表示使用的解析引擎

index\_col：将网页表格中的列标题作为DataFrame类的行索引

encoding：表示网页编码格式

na\_value：表示自定义缺省值

import requests

html\_data=requests.get('https://www.tiobe/tiobe-index/')

df\_html\_table=pd.read\_html(html\_data.content,encoding='utf-8')

**Pandas读写Word文件**

安装第三方库

pip install python-docx

python-docx的Document类相对于Word文件

包括paragraphs(段落对象列表）,tables(表格对象列表）等属性k

可获取相应对象

|  |  |
| --- | --- |
| Paragraph类的属性 | 说明 |
| text | 获取段落中的文本字符串 |
| runs | 获取段落中的节段对象 |
| style | 获取段落中的样式对象 |

|  |  |
| --- | --- |
| Table类的属性方法 | 说明 |
| cell(row\_idx,col\_idx) | 获取表格中指定的单元格 |
| rows | 获取表格中包含行数据的对象 |
| columns | 获取表格中包含列数据的对象 |
| text | 获取表格中的文本字符串 |

**python-docx的基本使用**

from docx import Document

#step1:创建Document类对象

docx=Document('aaa.docx')

#step2:获取段落对象

paragraphs=docx.paragraphs

for i in paragraphs:

print(i.text)#获取段落的文本信息

#step2:获取表格对象

tables=docx.tables

for table in tables:

for row in table.rows:#获取行数据对象

row\_content=[]#保存表格数据的列表

for cell in row.cells[:]:#获取单元格对象

row\_content.append(cell.text)#获取单元格中的字符串

**Pandas读写PDF文件**

安装第三方库

pip install pdfplumber

pdfplumber库提供PDF类(对应一个PDF文件)和Page类(对应PDF中每页)

|  |  |
| --- | --- |
| PDF类的属性 | 说明 |
| metadata | 返回一个包含创建日期,修改日期,作者等信息的字典 |
| pages | 返回一个包含Page类对象的列表 |

|  |  |
| --- | --- |
| Page类的方法 | 说明 |
| extract\_words() | 提取页面中所有单词及其相关信息 |
| extract\_text() | 提取页面中所有的文本数据和表格数据 |
| extract\_tables() | 提取页面中的表格数据 |

**pdfplumber库的基本使用**

import pdfplumber

with pdfplumber.open('aaa.pdf') as pdf:

print(pdf.pagess[0].extract\_text())

**Pandas读写数据库**

安装第三方包

pip install SQLALchemy

pip install PyMySQL

read\_sql(sql,con,index\_col=None,coerce\_float=True)

sql：表示被执行的sql查询语句或数据表名

con：表示使用SQLAlchemy连接数据库

index\_col：将数据表中的列标题作为DataFrame类的行索引

coerce\_float：表示是否将非字符串，非数字对象的值转换为浮点型，默认为True

conn=create\_engine(‘mysql+pymysql://user:passward@IP:3306/test01’)

• root: 用户名

• passward: 密码

• IP : 服务器IP，本地电脑用localhost

• 3306： 端口号

• test01 : 数据库名称

import pandas as pd

from sqlalchemy import create\_engine

engine=create\_engine('mysql+pymysql://'

'root:123456@127.0.0.1:3306/数据库名')

mysql\_data=pd.read\_sql('表名',engine)

**保存到mysql**

df.to\_sql(name, con=engine, if\_exists=‘replace/append/fail’,index=False)

• name是表名

• con是连接

• if\_exists：表如果存在怎么处理。三个选项 append代表追加, replace代表删除原表，建立新表，fail代表什么都不干

• index=False：不插入索引index

mysql\_data.to\_sql('ffff',con=conn,index=False,if\_exists='append')