电子科技大学
University of Electronic Science and Technology of China

Lecture 8 文本读写技术

- 读取文本文件
- 写入文本文件

教学目标

- 认识常见的文本读写技术的特点
- 掌握读取文件、写入文件、连接数据库的方法等

文本读写技术实现

- 本堂课内容涉及到文本读写的技术实现，都是采用Python语言完成的

- 用到了Python语言中的Pandas库，请大家在学习本课前安装Python及其Pandas库

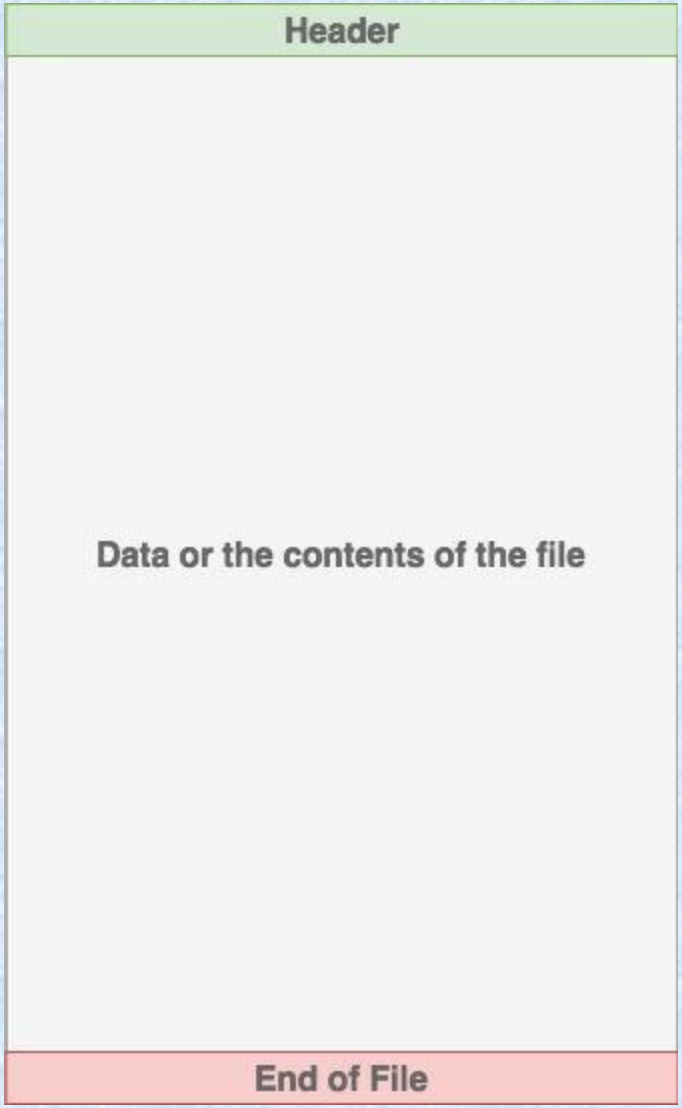
文件基本概念

文件（file）：计算机系统存储和处理数据的方式。现代文件系统上的文件由三个主要部分组成：

标题(Header)：有关文件内容的元数据（文件名，大小，类型等）

数据(Data)： 由创建者或编辑者编写的文件内容

文件结束符（EOF）：表示文件结尾的特殊字符



读写文件步骤

操作步骤

第1步： 打开需要操作的文件

第2步： 对文件读或写操作

第3步： 关闭文件

第一种语法：

```
f = open("文件路径/文件名" ,"读写模式")
读或写
f.close()
```

第二种语法：

```
with open("文件路径/文件名" ,"读写模式") as f:
读或写
自动关闭文件
```

文件基本概念

(1)读文件

read() 将文件的所有内容读取出来，返回一个字符串

readline() 读取文件中的一行内容， 返回一个字符串

readlines()，将文件的所有内容读取出来，返回一个列表，文件中的每行为列表中的一个元素

变量 = f.read()

变量 = f.readline()

列表变量 = f.readlines()

文件基本概念

(2)写文件

write(s) 将一个字符串写入文件

writelines(squ) 将一个序列对象的所有元素依次写入文件

f.write(字符串)

f.writelines(列表名) [ ]

文件基本概念

(3) 追写文件

w模式覆盖写入， 打开文件先清空内容，然后执行写入操作。（文件不存在创建文件 ）

a模块追加写入，打开文件在文件尾执行写入操作。（文件不存在创建文件 ）

```
def fun_write3():
    f=open('古诗.txt','a',encoding="utf-8")
    f.write('好雨知时节\n')
    f.write('当春乃发生\n')
    f.write('随风潜入夜\n')
    f.write('润物细无声\n')
    f.close()
```

文件基本概念

r	打开一个文件为只读。文件指针被放置在文件的开头。这是默认模式。
rb	打开一个文件只能以二进制格式读取。文件指针被放置在文件的开头。这是默认模式。
r+	打开用于读和写文件。文件指针置于该文件的开头。
rb+	打开用于读取和写入二进制格式的文件。文件指针置于该文件的开头。
w	打开一个文件只写。如果文件存在覆盖该文件。如果该文件不存在，则创建写入新文件。
wb	打开一个文件只能以二进制格式写入。如果文件存在覆盖该文件。如果该文件不存在，则创建写入新文件。
w+	打开文件为写入和读取模式。如果文件存在覆盖现有文件。如果该文件不存在，创建用于读写操作的新文件。
wb+	打开用于以二进制格式写入和读出文件。如果文件存在覆盖现有文件。如果该文件不存在，创建用于读写操作的新文件。
a	打开用于追加的文件。文件指针是在文件是否存在该文件的末尾。也就是说，该文件是在追加模式。如果该文件不存在，它会创建一个用于写入的新文件。
ab	打开文件用于二进制格式追加。文件指针是在文件是否存在该文件的末尾。也就是说，文件是在追加模式。 如果该文件不存在，它会创建一个用于写入的新文件。
a+	打开文件为追加和读取方式。文件指针是在文件是否存在该文件的末尾。该文件以追加模式打开。如果该文件不存在，它将创建用于读写操作的新文件。
ab+	打开一个文件以附加和二进制格式读取模式。如果该文件存在文件指针在该文件的末尾。该文件以追加模式打开。如果该文件不存在，它将创建读写操作的新文件。

8.1 读取文本文件

- 鸢尾花数据集，存储在一个文本文件中，源数据的前几行：

	A	B	C	D	E
1	150	4	setosa	versicolor	virginica
2	5.1	3.5	1.4	0.2	0
3	4.9	3	1.4	0.2	0
4	4.7	3.2	1.3	0.2	0
5	4.6	3.1	1.5	0.2	0
6	5	3.6	1.4	0.2	0
7	5.4	3.9	1.7	0.4	0
8	4.6	3.4	1.4	0.3	0
9	5	3.4	1.5	0.2	0
10	4.4	2.9	1.4	0.2	0

读取txt文件

- 读取txt文件的步骤
  - 打开test.txt文档需要的操作是

    >>> *fp = open('iris.txt', 'r')*
  - 当我们输入

    >>> *fp*

    Python Shell中会显示

<_io.TextIOWrapper name='iris.txt' mode='r' encoding='cp936'>

读取txt文件

- 读取txt文件的说明
  - <_io.TextIOWrapper name='iris.txt' mode='r' encoding='cp936'>表示txt文件已经成功打开
  - open函数的第一个参数是需要打开文本的存储路径，第二个参数‘r’指open函数采用的模式为“读取模式”。
  - 编码格式为*cp936*

读取test.txt文档中的某几行

- 读取txt文件的某几行
  - 想要读取一个文档中的某一行或几行，可以采用下面的一组命令：

```
>>> fp.readline()
```

显示一行

'150\t4\tsetosa\tversicolor\tvirginica\n ‘

其中，\t表示横向跳转到下一个制表符，\n表示换行。

```
>>> fp.readlines()
```
显示所有行

```
In [1]: fp = open('iris.txt','r')

In [2]: fp

Out[2]: <_io.TextIOWrapper name='iris.txt' mode='r' encoding='cp936'>

In [3]: fp.readline() # 一次读入文件的一行，然后fp游标自动转到文件的下一行首

Out[3]: '150\t4\tsetosa\tversicolor\tvirginica\n'

In [4]: fp.readline()

Out[4]: '5.1\t3.5\t1.4\t0.2\t0\n'

In [5]: fp.readlines()

Out[5]: ['4.9\t3\t1.4\t0.2\t0\n',
         '4.7\t3.2\t1.3\t0.2\t0\n',
         '4.6\t3.1\t1.5\t0.2\t0\n',
         '5\t3.6\t1.4\t0.2\t0\n',
         '5.4\t3.9\t1.7\t0.4\t0\n',
         '4.6\t3.4\t1.4\t0.3\t0\n',
         '5\t3.4\t1.5\t0.2\t0\n',
         '4.4\t2.9\t1.4\t0.2\t0\n',
         '4.9\t3.1\t1.5\t0.1\t0\n',
         '5.4\t3.7\t1.5\t0.2\t0\n'
```

读取文本常用函数

- open()函数中的第一个参数是打开文本文件的路径，第二个参数r代表读取模式，w代表写入模式，a代表追加模式，r+代表读写模式
- read()表示读取到文件尾，size表示读取大小

读取文本常用函数

- seek(0)表示跳到文件开始位置。
- readline()逐行读取文本文件。
- readlines()读取所有行到列表中，通过for循环可以读出数据。
- close()关闭文件。

- 读取txt文件完整代码（第一种方法）

```
#read txt method one
f = open("./image/abc.txt")
line = f.readline()
while line:
    print line
    line = f.readline()
f.close()
```

- 读取txt文件完整代码（第二种方法）

```
#read txt method two
with open("1.txt", "r", encoding='utf-8') as f:

#read data
data = f.read()

print(data)
```

- 读取txt文件完整代码（第三种方法）

```
1  #read txt method three
2  f2 = open("./image/abc.txt","r")
3  lines = f2.readlines()
4  for line3 in lines:
5      print line3
```

8.2 读取CSV文件

CSV（comma separated values）文件

以纯文本形式存储表格数据（数字和文本）的一种通用文件格式。在实践中“CSV”泛指具有以下特征的任何文件：

- 纯文本，使用某个字符集，比如ASCII、Unicode、EBCDIC或GB2312
- 由记录组成（典型的是每行一条记录）
- 每条记录被分隔符分隔为字段（典型分隔符有逗号、分号或制表符，有时分隔符可以包括可选的空格）
- 每条记录都有同样的字段序列

8.2 读取CSV文件

- 可以采用上节中读取txt文件的所有常用函数
  - open()函数中的第一个参数是打开文本文件的路径，第二个参数r代表读取模式，w代表写入模式，a代表追加模式，r+代表读写模式
  - read()表示读取到文件尾，size表示读取大小
  - readline()逐行读取文本文件。
  - readlines()读取所有行到列表中，通过for循环可以读出数据。

读取CSV文件

- 还可以采用pandas中提供的一些函数，将csv等表格型文件直接读取到一个Python的DataFrame对象里面。
- 最常用的函数包括read_csv和read_table函数

1、read_csv函数

- 首先导入pandas库>>> *import pandas as pd*
- 然后通过>>> *df = pd.read_csv('iris.csv')*
  将iris.csv存储到df这个DataFrame里面。

read_csv函数

- 查看命令>>> *df*

	150	4	setosa	versicolor	virginica
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

2、read_table函数

- 首先导入pandas库

  >>> *import pandas as pd*

- 然后通过

  >>> *df=pd.read_table('iris.csv', sep=',')*

  将iris.csv存储到df这个DataFrame里面。

- 查看命令

>>> *df*

In [2]:
```
import pandas as pd
df = pd.read_table('iris.csv', sep=',')
df
```

Out[2]:

	150	4	setosa	versicolor	virginica
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

3、逐块读取文本文件

- 如果只想读取其中的几行（避免读取整个文件），就可以通过nrows来进行指定：

```
In [5]: df5 = pd.read_table('iris.csv', nrows=5)
        df5
```

Out[5]:

	150,4,setosa,versicolor,virginica
0	5.1,3.5,1.4,0.2,0
1	4.9,3.0,1.4,0.2,0
2	4.7,3.2,1.3,0.2,0
3	4.6,3.1,1.5,0.2,0
4	5.0,3.6,1.4,0.2,0

逐块读取文本文件

- 当我们要逐块读取文件时，还有一种办法是设置chunksize（行数）：

```
>>>
chunck=pd.read_csv('iris.csv',chunksize=5)
```

逐块读取文本文件

- 迭代处理iris.csv文件，统计该文件中的行数，我们可以用下面的操作：

```
>>> tot = 0
>>> for piece in chunck:
        tot = tot + 1
```

- 统计每一块的数据行数后，迭代求出整个iris.csv文件中的数据总行数。

逐块读取文本文件

```
In [8]: chunck = pd.read_csv('iris.csv',chunksize=5)
        chunck
Out[8]: <pandas.io.parsers.TextFileReader at 0x1c4d04ee490>

In [9]: tot=0
        for piece in chunck:
            tot=tot+1

In [10]: tot
Out[10]: 30
```

8.3 写入文本文件

- 要把数据写入txt文件，我们就必须先创建 file 对象。
- 在这情况下，必须用 ‘w’ 模式标记指定要写入的文件。

写入文本文件

- 我们创建一个名叫myfile的文件，首先把mydata list的内容写入文件，关闭文件。

```
>>> mydata = ['Date', 'Time']
>>> myfile = open('testit.txt', 'w')
>>> for line in mydata:
        myfile.write(line + '\n')
>>> myfile.close()
```

写入文本文件

- 重新打开文件，就可以读取文件内容了。

```
>>> myfile = open("testit.txt")
>>> myfile.read()
        'Date\nTime\n'
>>> myfile.close()
```

同时读取和写入文件

- 用 ‘a+’ 模式重新打开文件

```
myfile = open('testit.txt', 'a+') # 以读取和添加模式打开文件
myfile.seek(0)
myfile.read()
```

显示'Date\nTime\n'

同时读取和写入文件

```
for line in mydata:
    myfile.write(line + '\n')
myfile.seek(0)
myfile.read()
```

显示 'Date\nTime\nDate\nTime\n'

```
myfile.close()
```

python读取PDF文件内容

安装第三方工具PyPDF2

```
> pip install PyPDF2
```

PyPDF2是作为PDF工具包构建的python库，它能够：

- 提取文档信息（标题，作者，...）
- 按页拆分文档
- 逐页合并文档
- 裁剪页面
- 合并多个页面到一个页
- 对pdf文档进行加密解密
- 其他

python读取PDF文件内容

从pdf中提取文字

```
import PyPDF2

pdfFile = open('example.pdf','rb')
pdfReader = PyPDF2.PdfFileReader(pdfFile)
print(pdfReader.numPages)
page = pdfReader.getPage(0)
print(page.extractText())
pdfFile.close()
```

合并pdf文件

```python
import PyPDF2

def PDFmerge(pdfs, output):

    #创建一个pdf文件合并对象
    pdfMerger = PyPDF2.PdfFileMerger()
    #逐个添加pdf
    for pdf in pdfs:
        with open(pdf,'rb') as f:
            pdfMerger.append(f)

    #将内存中合并的pdf文件写入
    with open(output,'wb') as f:
        pdfMerger.write(f)

def main():
    #需要合并的pdf名称
    pdfs = ['example.pdf', 'testexample.pdf']
    #合并完成的pdf名称
    output = 'combined_example.pdf'
    #调用PDFmerge函数，进行合并
    PDFmerge(pdfs,output)

if __name__ == '__main__':
    main()
```

Apriori算法概念

- 项集：即项的集合。{牛奶, 面包}，其中牛奶和面包为项，{牛奶、面包}为2项集。
- 关联规则：形如$X \rightarrow Y$的蕴涵表达式（If...Then...），其中X和Y是不相交的项集。
- 支持度：项集X、Y同时发生的概率称之为关联规则的支持度，$s(X \rightarrow Y) = \frac{|X \cup Y|}{N}$
- 置信度：项集X发生的情况下，则项集Y发生的概率，$c(X \rightarrow Y) = \frac{|X \cup Y|}{|X|}$

Apriori算法概念

- 最小支持度：人为按照实际意义规定的阈值，表示项集在统计意义上的最低重要性。
- 最小置信度：人为按照实际意义规定的阈值，表示关联规则最低可靠性。
  - 如果支持度与置信度同时达到最小支持度与最小置信度，则此关联规则为强规则。
- 频繁项集：满足最小支持度的所有项集，称作频繁项集。

Apriori算法的Python实现

算法步骤

1. 算法扫描所有的事务，获得每个项，生成C1（见下文代码中的create_C1函数）。然后对每个项进行计数。然后根据最小支持度从C1中删除不满足的项，从而获得频繁1项集L1。

2. 对L1生成的集合执行剪枝策略产生候选2项集的集合C2，然后扫描所有事务，对C2中每个项进行计数。同样的，根据最小支持度从C2中删除不满足的项，从而获得频繁2项集L2。

3. 对L2的自身连接生成的集合执行剪枝策略产生候选3项集的集合C3，然后，扫描所有事务，对C3每个项进行计数。同样的，根据最小支持度从C3中删除不满足的项，从而获得频繁3项集L3。

4. 以此类推，对Lk-1的自身连接生成的集合执行剪枝策略产生候选k项集Ck，然后，扫描所有事务，对Ck中的每个项进行计数。然后根据最小支持度从Ck中删除不满足的项，从而获得频繁k项集。

Apriori算法的Python实现

由频繁项集产生关联规则

- 对于每个频繁项集itemset，产生itemset的所有非空子集（这些非空子集一定是频繁项集）；
- 对于itemset的每个非空子集s，如果 $\frac{\sup port\_count(l)}{\sup port\_count(s)} \geq \min\_conf$

  则输出 $s \Rightarrow (l - s)$

  其中min_conf是最小置信度阈值。

Apriori算法的Python实现

Apriori算法的Python实现

计算构建候选项集

```
# 构建1-itemsets候选项集C1
def create_C1(data_set):
    item_set = frozenset([item])
    C1.add(item_set)
```

```
# 构建所有包含k-itemsets的候选项集Ck
def create_Ck(Lksub1, k):
    if is_apriori(Ck_item, Lksub1):
        Ck.add(Ck_item)
```

Apriori算法的Python实现

由候选项集生成频繁项集

$C_k \longrightarrow L_k$

```
# 通过对候选项集删减生成频繁项集
def generate_Lk_by_Ck(data_set, Ck, min_support, support_data):
```

```
if (item_count[item] / t_num) >= min_support:
            Lk.add(item)
            support_data[item] = item_count[item] / t_num
```

Apriori算法的Python实现

由频繁项集产生关联规则

L ⟹ Big Rule List

```
# 挖掘产生关联规则
def generate_big_rules(L, support_data, min_conf):
```

```
if conf >= min_conf and big_rule not in big_rule_list:
        big_rule_list.append(big_rule)
```

8.4 数据库的连接

- 引入数据处理模块MySQLdb后，就可以和数据库进行连接

```
db = MySQLdb.connect("localhost","root","123456","myciti" )
```

上述代码中四个关键的参数：第一个参数是服务器的地址；第二个参数是用户名；第三个参数是dbms密码；第四个参数是需要访问的数据库名称。

- 注意：要在Python3中使用MySQLdb，在使用前添加命令：

```
import pymysql
pymysql.install_as_MySQLdb()
```

执行sql语句

- 连接上数据库之后，我们就需要开始执行sql语句了。

```
import MySQLdb
db = MySQLdb.connect("localhost","root","123456","myciti" )
cursor = db.cursor()
```

执行sql语句

```
sql="""insert into article values
(0,"woainimahah","http://www.aa.com")"""
try:
    cursor.execute(sql)
    db.commit()
except:
    db.rollback()
db.close
```

选择和打印

- 连接数据库后，获取数据库中的数据信息，并对信息进行展示和打印。

```
import MySQLdb
db =
MySQLdb.connect("localhost","root","123456",
"myciti" )
cursor = db.cursor()
cursor.execute("select * from article “)
```

选择和打印

```
datas = cursor.fetchall()
for data in datas:
    print data[1]
print cursor.rowcount,"rows in tatal"
db.close
```

Fetchall是取出数据库表中的所有行数据

rowcount是读出数据库表中的行数

动态插入

- 动态插入是用占位符来实现的

```
import MySQLdb
title = "title"
url = "urlofwebpage "
db =
MySQLdb.connect("localhost","root","123456"
,"myciti" )
cursor = db.cursor()
```

动态插入

```
sql = """insert into article values
(0,"%s","%s","2012-9-
8","wo","qq","skjfasklfj","2019","up")"""
try:
    cursor.execute(sql%(title,url))
    db.commit()
except:
    db.rollback()
db.close
```

update操作

- 在update的操作中，占位符的使用和上面是一样的

```
import MySQLdb
title = "title"
id=11
db =
MySQLdb.connect("localhost","root","123
456","myciti" )
```

update操作

```
cursor = db.cursor()
 sql = """update article set title = "%s" where id =
"%d" """
 try:
    cursor.execute(sql%(title,id))
    db.commit()
 except:
    db.rollback()
 db.close
```