

# Python语言基础与应用

基本扩展模块 / 文本文件读写

陈斌 北京大学 [gischen@pku.edu.cn](mailto:gischen@pku.edu.cn)



# 文本文件读写

- › 文件的打开
- › 文件的读写和访问
- › 文件的关闭
- › 结构化文本文件

# 文件的打开

## › 普通文件

数据持久化的最简单类型

仅仅是在一个文件名下的字节流，把数据从文件读入内存，从内存写入文件

## › open()函数

```
f = open(filename[,mode[,buffering]])
```

- f: open()返回的文件对象
- filename: 文件的字符串名
- mode: 可选参数，打开模式和文件类型
- buffering: 可选参数，文件的缓冲区，默认为-1



# 文件的打开模式

## › mode第一个字母表明对其的操作：

- ‘r’表示读模式
- ‘w’表示写模式
- ‘x’表示在文件不存在的情况下新建并写文件
- ‘a’表示在文件末尾追加写内容
- ‘+’表示读写模式

## › mode第二个字母是文件类型

- ‘t’表示文本类型
- ‘b’表示二进制文件

# 文件的读写和访问

## › 文件的写操作

`f.write(str)`

`f.writelines(strlist)`: 写入字符串列表

## › 文件的读操作

`f.read()`

`f.readline()`: 返回一行

`f.readlines()`: 返回所有行、列表

```
>>> f=open("my.txt", "w")
>>> f.writelines(["apple\n", "pie\n"])
>>> f.close()
>>> f=open("my.txt", "r")
>>> f.readlines()
['apple\n', 'pie\n']
>>> f.close()
```

# 文件的关闭

## › 文件打开后要记得关闭

关闭的作用是终止对外部文件的连接，同时将缓存区的数据刷新到硬盘上

## › 调用close()方法

```
f.close()
```

## › 使用上下文管理器(context manager)

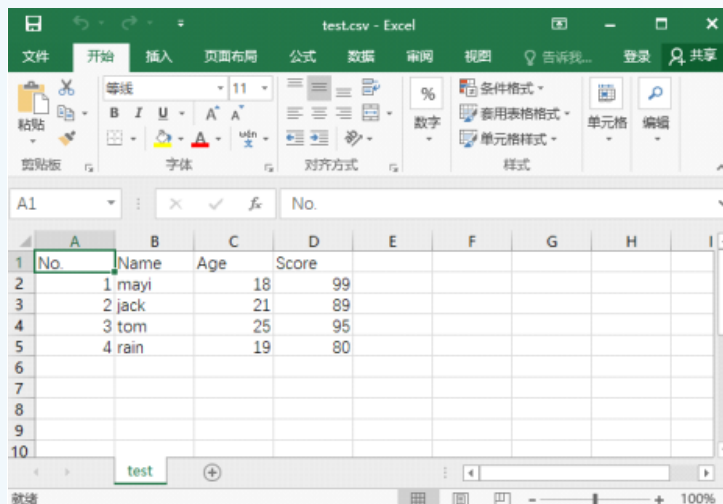
确保在退出后自动关闭文件

```
with open('textfile','rt') as myfile:  
    myfile.read()
```

# 结构化文本文件：CSV

## › 纯文本文件，以 “,” 为分隔符

- 值没有类型，所有值都是字符串
- 不能指定字体颜色等样式
- 不能指定单元格的宽高，不能合并单元格
- 没有多个工作表
- 不能嵌入图像图表



No.	Name	Age	Score
1	mayi	18	99
2	jack	21	89
3	tom	25	95
4	rain	19	80

# 结构化文本文件：CSV

## › 文件读取 - reader

```
re = csv.reader()
```

- 接受一个可迭代对象（比如csv文件），能返回一个生成器，可以从其中解析出内容

## › 文件读取 - DictReader

```
re = csv.DictReader()
```

- 与reader类似
- 但返回的每一个单元格都放在一个元组的值内



# 结构化文本文件：CSV

## › 文件写操作

```
w = csv.writer()  
w.writerow(rows)
```

- 当文件不存在时，自动生成
- 支持单行写入和多行写入

## › 字典数据写入

```
w = csv.DictWriter()  
w.writeheader()  
w.writerow(rows)
```

# 结构化文本文件：Excel

## › openpyxl 模块

- 用来读写扩展名为xlsx/xlsm/xltx/xltm的文件
- Workbook类是对工作簿的抽象
- Worksheet类是对表格的抽象
- Cell类是对单元格的抽象文件写操作

## › 操作之前先导入第三方库

- 安装: `pip install openpyxl`
- 导库: `from openpyxl import workbook`

# 结构化文本文件：Excel

## › 创建Excel文件

- 一个Workbook对象代表一个Excel文档，使用该方法创建一个Worksheet对象后才能打开一个表

```
from openpyxl import Workbook  
wb = Workbook()  
ws = wb.active
```

## › 读取Excel文件

```
from openpyxl import load_workbook  
wb = load_workbook(filename)  
ws = wb.file.active
```

# 结构化文本文件：Excel

## › 获取单元格信息

- 获取Cell对象

```
c = wb['sheet']['A1']
```

```
c = wb['sheet'].cell(row=1,column=1)
```

- `c.coordinate` : 返回单元格坐标
- `c.value` : 返回单元格的值
- `c.row` : 返回单元格所在的行坐标
- `c.column` : 返回单元格所在列坐标



# 结构化文本文件：PDF

## › 轻松处理pdf文件的库 – PyPDF2

- 包含了PdfFileReader、PdfFileMerger、PageObject和PdfFileWriter四个主要类
- 能进行读写、分割、合并、文件转换等多种操作
- 只能从PDF文档中提取文本并返回为字符串，而无法提取图像、图表或其他媒体

# 结构化文本文件：PDF

## › 读取PDF文件

```
readFile = open('test.pdf','rb')  
pdfFileReader = PdfFileReader(readFile)
```

## › pdfFileReader类

- .getNumPages(): 计算PDF文件总页数
- .getPage(index): 检索指定编号的页面

# 结构化文本文件：PDF

## › PDF文件的写操作

```
writeFile = 'output.pdf'  
pdfFileWriter = PdfFileWriter()
```

## › pdfFileWriter类

- .addPage(pageObj): 根据每页返回的 PageObject, 写入到文件
- .addBlankPage(): 在文件的最后一页后面写入一个空白页, 保存到新文件

# 结构化文本文件：PDF

## › 合并多个文档

```
pdf_merger = PdfFileMerger()  
pdf_merger.append('python2018.pdf')  
pdf_merger.merge(20, 'insert.pdf')  
pdf_merger.write('merge.pdf')
```

## › 单个页面操作 – PageObject类

- `.extractText()`: 按照顺序提取文本
- `.getContents()`: 访问页面内容
- `.rotateClockwise(angle)`: 顺时针旋转
- `.scale(sx,sy)`: 改变页面大小