杭州电子科技大学
HANGZHOU DIANZI UNIVERSITY

第5章 文件

5.1 基本概念

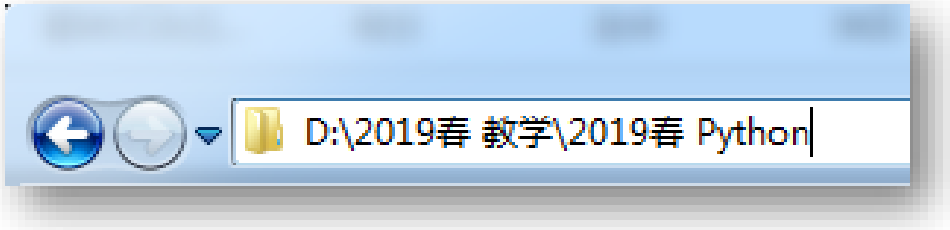
5.2 文件操作

- 理解文件与目录、文件的编码方式。
- 理解文本文件和二进制文件的区别。
- 掌握文件的打开、关闭、定位。
- 掌握文件的读取、写入和追加操作。

基本概念

- 计算机对数据进行管理是以文件为单位的。
- 文件是一个存储在磁盘等外存上的数据序列，可以包含任何数据内容。
- 文件是数据的集合和抽象。用文件形式组织和表达数据更有效、也更为灵活。

文件和目录



- 文件是通过目录来组织和管理的，目录提供了指向对应磁盘空间的路径地址。
- 目录一般采用树状结构，在这种结构中，每个磁盘有一个根目录，它包含若干文件和子目录。

- **绝对路径：**从根目录开始标识文件所在的完整路径。
- **相对路径：**相对于程序所在的目录位置，标识文件所在的路径。

【例5-1】 绝对路径示例

假定文件file.txt保存在D:盘lecture目录的ex子目录下，那么包含绝对路径的文件名表示为：

D:\lecture\ex\file.txt

在Python中用字符串表示为：

"D:\\lecture\\ex\\file.txt"

Python语言中，在表示路径的字符串中，"\"必须转义为"\\"

【例5-2】 相对路径示例

假定源程序保存在D:盘的lecture目录下，那么包含相对路径的文件名表示为：

ex\file.txt

在Python语言中用字符串表示为：

"ex\\file.txt"

```
>>> import os
>>> os.getcwd()
```

文本文件：具有可读性。但存取时需要编/解码，存取速度较

二进制文件：存取直接是二进制值的处理，无需编/解码，存取较快。但通常无法直接读懂。

- **按照文件的编码方式，可分为两种类型：**
  - **文本文件** ——基于**字符编码**的文件，由统一特定编码的字符组成，如UTF-8编码，内容容易统一展示和阅读，如：txt格式的文本文件。
  - **二进制文件** ——基于**值编码**的文件，存储的是二进制数据0和1，没有统一字符编码，文件内部数据的组织格式与文件用途有关，如：png格式的图片文件、avi格式的视频文件。

文本文件和二进制文件

无论文件创建为文本文件或者二进制文件，都可以用“文本文件方式”和“二进制文件方式”打开，打开后的操作不同。

【例】理解文本文件和二进制文件的区别

```
txtFile = open("test.txt", "rt") #t表示文本文件方式
t = txtFile.read()
print(t)
txtFile.close()
print('----')
binFile = open("test.txt", "rb") #b表示二进制文件方式
b = binFile.read()
print(b)
binFile.close()
```

文本文件和二进制文件

【例】理解文本文件和二进制文件的区别

```
>>>
=== RESTART: C:/Users/WHH/AppData/Local/Programs/Pyth
文本文件
二进制文件
b'\xce\xc4\xb1\xbe\xce\xc4\xbc\xfe\r\n\xb6\xfe\xbd\xf
```

采用文本方式读入文件，文件经过编码形成字符串，打印出有含义的字符；

采用二进制方式打开文件，文件被解析为字节（byte）流，其内容为字符的编码，可见字符串中的一个汉字字符由2个字节表示。

文件的操作

Python程序中对文件的操作一般包括：

- 打开文件 `f = open(…)`
- 读取文件 → `f.read(size)` `f.readline()` `f.readlines()`
- 对文件数据进行处理
- 写入文件 → `f.seek(offset)` `f.write(s)` `f.writelines(lines)`
- 关闭文件 `f.close()`

文件的打开和关闭

打开文件：建立文件对象与物理文件的关联。

<文件对象> = open(<文件名>[,打开模式])

open()函数有两个参数：

- 文件名：可包含完整路径
- 打开模式：可选，默认为'r'

文件的打开和关闭

open()函数提供7种基本的打开模式：

模式	含义
'r'	以只读方式打开，如果文件不存在，则报错FileNotFoundError，默认值
'w'	覆盖写方式，文件不存在则创建，存在则覆盖原内容
'x'	创建写方式，文件不存在则创建，存在则报错FileExistsError
'a'	追加写方式，文件不存在则创建，存在则在原文件末尾追加内容
'b'	二进制文件模式，添加在其它控制字符后
't'	文本文件模式，默认值
'+'	与r/w/x/a一同使用，在原功能基础上增加同时读写功能

文件的打开和关闭

<文件对象>.close()

当处理完一个文件后，需要关闭文件，以保证正常释放该文件对象所占用的系统资源。如：关闭文件后才可以使用记事本对文本文件进行编辑等操作。

文件定位

文件打开之后，读/写指针会定位在文件的头部。

- 顺序存取：文件中的信息将按照顺序被访问。
- 随机存取：将读/写指针移动到文件的任何位置进行存取操作。文件定位函数：

```
f.seek(<偏移值>[,<起始值>])
```

- 起始位置：0（文件开头，默认值）、1（当前位置）、2（文件末尾）
- 偏移值：单位是字节，正数表示向右（文件尾方向），负数表示向左（文件头方向）

seek中whence参数的值：

0：

open函数以r，w，带b的二进制模式，就是以任何模式打开文件，都能正常运行

1和2：

open函数只能以二进制模式打开文件，才能正常运行，否则就会报出错误

如果没有以二进制b的方式打开，则offset无法使用负值（即向左侧移动）

文件的读取、写入、追加

❖ 从文件读取数据

1）f.read(size)方法

- 从文件中读入长度为size的字符串或字节流。
- size参数如果省略，则表示读取文件所有内容。
  如果已到达文件的末尾，则返回一个空字符串。

文件的读取、写入、追加

❖ 从文件读取数据

2）f.readline()方法

- 从文件的当前读/写位置，读入一行内容，以\n结尾。

3）f.readlines()方法

- 从文件中读入所有行，以每行为元素形成一个列表。可设参数指定读入的行数。

文件的读取、写

思考：如何统计文件中包含的行数，并逐行打印？

❖ 从文件读取数据

```
>>> f = open('workfile.txt')
>>> f.read()
'宝剑锋从磨砺出\n梅花香自苦寒来\n'
>>> f.read()
''
```

```
>>> f.seek(0)
0
>>> f.readline()
'宝剑锋从磨砺出\n'
>>> f.readline()
'梅花香自苦寒来\n'
>>> f.readline()
```

```
>>> f.seek(0)
0
>>> f.readlines()
['宝剑锋从磨砺出\n', '梅花香自苦寒来\n']
>>> f.readlines()
[]
>>> f.close()
```

文件的读取、写入、追加

❖ 从文件读取数据

从文件中读取行，更高效的方法是在文件对象上循环。这不但可以节省内存，而且代码也更简洁。例如：

```
>>> f = open('workfile.txt')
>>> for line in f:
        print(line, end='')


宝剑锋从磨砺出
梅花香自苦寒来
>>> f.close()
```

文件的读取、写入、追加

❖ 从文件读取数据

对于只需读入数据的场合，Python还提供了快速列表访问方式：

```
<列表> = list(open(<文件名>))
```

文件的打开和读取合二为一，也不必使用文件的关闭操作

```
>>> ls = list(open('workfile.txt'))
>>> ls
['宝剑锋从磨砺出\n', '梅花香自苦寒来\n']
```

文件的读取、写入、追加

❖ 将数据写入文件

1）f.write(string)方法

- 向文件写入一个字符串或字节流。但不会自动换行，如需换行，则要使用'\n'。

2）f.writelines(lines)方法

- 将一个元素为字符串的列表写入文件。

文件的读取、写入、追加

❖ 将数据写入文件

【例】向文件写入一个列表，再读取并显示文件内容。

```
ls = ['唐诗', '宋词', '元曲']
fname = input("请输入要写入的文件：")
```

文件的读取、写入、追加

❖ 将数据追加到文件末尾

```
>>> f=open('workfile.txt', 'a')
>>> f.write('书到用时方恨少\n')
8
>>> f.close()
```

以'a'模式打开文件，指针会移到文件末尾处，写入的内容将追加到文件末尾。但必须关闭文件才能生效。

【例】 从score.txt读取学生成绩数据，统计学科等级水平，并将等级写入level.txt文件。

（1）生物和科学两门课都达到60分，总分达到180分为及格；

（2）每门课达到85分，总分达到260分为优秀；

（3）总分不到180分或有任意一门课不到60分，为不及格。

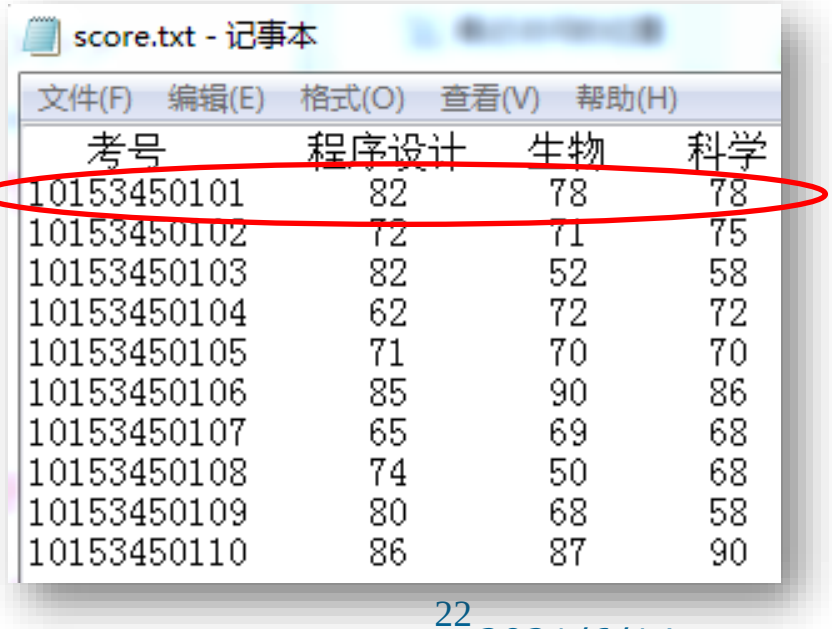
每行是一个字符串

split( )

三门课的成绩

if-else

等级



考号	程序设计	生物	科学
10153450101	82	78	78
10153450102	72	71	75
10153450103	82	52	58
10153450104	62	72	72
10153450105	71	70	70
10153450106	85	90	86
10153450107	65	69	68
10153450108	74	50	68
10153450109	80	68	58
10153450110	86	87	90

score.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

考号	程序设计	生物	科学
10153450101	82	78	78
10153450102	72	71	75
10153450103	82	52	58
10153450104	62	72	72
10153450105	71	70	70
10153450106	85	90	86
10153450107	65	69	68
10153450108	74	50	68
10153450109	80	68	58
10153450110	86	87	90

- **方案一：**

（1）读取文件score.txt数据到列表L中

列表L中的数据项对应着文件中的每条学生记录，通过循环语句遍历L，提取需要的考号和三门课的成绩，并存放在列表x中。

（2）判定学科等级

列表x包含4个数据项，x[0]为考号，x[1]、x[2]和x[3]分别为“程序设计”、“生物”和“科学”三门课的成绩，需要转换为整数类型以便进行求和等数值运算。最后通过分支语句，将求得的等级结果存放在key变量中。

（3）将考号和等级结果按一定格式写入文件level.txt中。

score.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

考号	程序设计	生物	科学
10153450101	82	78	78
10153450102	72	71	75
10153450103	82	52	58
10153450104	62	72	72
10153450105	71	70	70
10153450106	85	90	86
10153450107	65	69	68
10153450108	74	50	68
10153450109	80	68	58
10153450110	86	87	90

```
lines = list(open("score.txt"))
f = open("level.txt", "w")
for s in lines[1:]:
    x = s.split()
    x[1:] = [int(i) for i in x[1:]]
    xSum = sum(x[1:])
    if x[1]>=85 and x[2]>=85 and x[3]>=85 and xSum>=260:
        level = '优秀'
    elif x[2]>=60 and x[3]>=60 and xSum>=180:
        level = '及格'
    else:
        level = '不及格'
    f.write('%s\t%s\n' % (s[:-1], level))
f.close()
```

❖ **方案二：**

方案一利用列表存放文件中的数据，需要占用额外的内存空间。更优的处理方法是：使用readline()语句读取一条记录，判定该学生考核等级，并与记录合并写入文件level.txt中。若某次循环读到空行，则跳出循环，结束对文件的处理。

score.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

考号	程序设计	生物	科学
10153450101	82	78	78
10153450102	72	71	75
10153450103	82	52	58
10153450104	62	72	72
10153450105	71	70	70
10153450106	85	90	86
10153450107	65	69	68
10153450108	74	50	68
10153450109	80	68	58
10153450110	86	87	90

```
f0 = open("score.txt")
f = open("level.txt", "w")
f0.readline()
while True:
    s = f0.readline()
    x = s.split()
    if len(x) == 0:
        break
    x[1:] = [int(i) for i in x[1:]]
    xSum = sum(x[1:])
    if x[1]>=85 and x[2]>=85 and x[3]>=85 and Xsum>=260:
        level = '优秀'
    elif x[2]>=60 and x[3]>=60 and Xsum>=180:
        level = '及格'
    else:
        level = '不及格'
    f.write('%s\t%s\n' % (s[:-1], level))
f0.close()
f.close()
```