字符串

一、 字符串类型

1. 字符串常量：

与 C 语言中储存字符串的字符数组不同，Python 没有 char 这种单个字符的类型，取而代之的是使用一个字符的字符串。定义一个字符串常量有三种方式：单引号、双引号和三引号

```
s='hello world'   #单引号

s="hello world"   #双引号

s='''hello
    world'''      #三引号
```

其中单引号和双引号两者是等效的，且字符序列只能在一行定义；三引号内的字符序列可以分布在连续的多行上。

2. 转义字符与抑制转义字符：

(1)转义字符

Python 中用“\”字符嵌入我们不能通过键盘输入的字节。“\”以及后边的一个或多个字符最终在字符串中会被单个字符替代，这个字符通过转义字符定义了一个二进制值。

```
#常见转义字符:
\n        #换行符
\t        #水平制表符
\         #续行符
\"        #双引号
\\        #反斜杠
```

当 Python 遇到“\”时，会优先考虑将其翻译为转义字符，因此当我们想要打印“\”时要用“\\”

```
>>> s="s\tp\na\x00m"
>>> print(s)
s       p
a
>>> s="s\\tp\\na\\x00m"
>>> print(s)
s\tp\na\x00m
```

（2）抑制转义字符

当一个新手尝试去用这样一行代码打开文件的时候就会遇到问题。这个调用的结果将会是 C:(换行)ew（空格）ext.dat 的文件，而不是我们所期待的结果。

Myfile=open(‘C:\new\text.dat’,’w’)

为了解决这个问题，raw 字符串被引入。如果字母 r 的大写或小写形式出现在字符串的第一个引号前面，他就会关闭转义机制。

Myfile=open(r‘C:\new\text.dat’,’w’)

二、 字符串常用操作

1. 拼接字符串（直接将同类型的 str 相加）

```
>>> str1='今天我一共走了'
>>> num=143412
>>> str2='步'
>>> print(str1+str(num)+str2)
今天我一共走了143412步
```

2. 计算字符串长度（len(str)）

3. 截取字符串(索引和分片)

字符串被定义为字符的有序集合，所以我们能够通过位置获取他们的元素。在 Python 中，字符串中的字符是通过索引提取的。Python 中支持用负数来获取元素，具体来说负偏移可以看作是从结束处反向计数。

（1） 索引（s[i]）获取特定偏移的元素：

----第一个元素的偏移为 0

----负索引意味着从最后一位反向计数

（2） 分片（s[i:j]）提取对应部分作为一个序列：

----上边界不包含在内

----分片的默认边界是 0 和序列的长度

----s[1:3]获取了偏移为 1 的元素，直到偏移不包括 3 的元素

----s[1:]获取了从偏移为 1 到末尾的序列

----s[:-1]获取了从偏移为 0 直到但是不包括最后一个元素的序列

（3） 扩展分片(s[i:j:k])提取给定步长的子序列：

----s[1:10:2]会取出 s 中偏移值在 1-9 之间，间隔了一个元素的序列，也就是收集了 1,3,5,7,9 的元素

----s[::-1]会将全部序列通过负数步提取，实际效果就是将序列进行反转

4. 分割和合并字符串

（1） 分割字符串

str.split(sep,maxsplit) #分割字符串

sep:用于指定分隔符，可以包含多个字符，默认状态下采用空白符进行分割，这时无论有多少空格都将作为一个分隔符进行分割。

Maxsplit：可选参数，用于指定分割的次数，默认状态为-1，即分割次数没有限制

返回值：分割后的字符串储存在列表之中

Eg:

```
>>> s="aaasafasfabcafassfabcfeawojfabcasfjwef"
>>> s.split('abc')
['aaasafasf', 'afassf', 'feawojf', 'asfjwef']
```

（2） 合并字符串

Strnew=string.join(iterable)

Strnew:合并后生成的新字符串

String:用于指定合并时的分隔符

Iterable:可迭代对象，该迭代对象中所有元素将被合并为一个新的字符串。

Eg:

```
>>> List=['明日科技','马云','马化腾','俞敏洪']
>>> s='@'.join(List)
>>> print(s)
明日科技@马云@马化腾@俞敏洪
```

5. 修改字符串

String.replace(str1,str2) #用 str2 替换 str1

三、 格式化字符串

1. 格式化表达式

‘%[-][+][0][m][.n]格式化字符’ %exp

-:可选参数，用于指定左对齐，正数前方无符号，负数前加负号

+：可选参数，用于指定右对齐，正数前方加正号，负数前方加负号

0：可选参数，表示右对齐，正数前方无符号，负数前方加负号，用 0 填充空白处

m:可选参数，表示占有宽度

.n:可选参数，表示小数点后保留的位数

格式化字符：%s %c %d %f

Exp:要转换的项

2. 字符串对象的 format()方法

str.format(args)

str:用于指定字符串的显示样式（即模板）

args：用于指定要转换的项，如果有多项，则用逗号分隔

下面重点介绍模板。在创建模板时，需要使用{}和：指定占位符，语法格式如下：

{[index][:[[fill]align][sign][#][width][.precision][type]}

Index:可选参数，用于指定设置格式的对象在参数列表中的索引位置，如果省略，则根据值前后顺序自动分配。

Fill:可选参数，用于指定空白处填充的字符

Align:可选参数，用于指定对齐方式(‘<’表示左对齐；’>’表示右对齐；’=’表示右对齐，只对数字有效；’^’居中)需要配合 width 一起使用

Sign:用于指定有无符号数（+表示整数加正号，负数加负号；-表示正数不变；空格表示整数加空格，负号加负号）

#：对于二进制、八进制和十六进制数，加上#会显示 0b\0o\0x 前缀

Width：用于指定所占宽度

.precision:指定保留小数位数

Type:用于指定类型

格式化字符	说　明	格式化字符	说　明
s	对字符串类型格式化	b	将十进制整数自动转换成二进制表示再格式化
d	十进制整数	o	将十进制整数自动转换成八进制表示再格式化
c	将十进制整数自动转换成对应的 Unicode 字符	x 或者 X	将十进制整数自动转换成十六进制表示再格式化
e 或者 E	转换为科学计数法表示再格式化	f 或者 F	转换为浮点数(默认小数点后保留 6 位)再格式化
g 或者 G	自动在 e 和 f 或者 E 和 F 中切换	%	显示百分比（默认显示小数点后 6 位）

Eg:

```
>>> template='标号：{:0>9s}'
>>> context=template.format('7')
>>> print(context)
标号：000000007
```

Eg:

```
>>> format_title="{:^6}{:^12}\t{:^8}\t{:^10}\t{:^10}\t{:^10}"
>>> print(format_title.format("ID","名字","英语成绩","python成绩","C语言成绩","总成绩"))
  ID        名字            英语成绩        python成绩        C语言成绩        总成绩
```

四、　字符串编码转换

在 Python 中，有两种常用的字符串类型，分别是 str 和 bytes。其中 str 表示 Unicode 字符，bytes 表示二进制数据。这两种类型无法拼接在一起使用，通常情况下，str 在内存中以 Unicode 表示，一个字符对应多个字节，但是如果在网络上传输，或者存在磁盘上，就需要把 str 转化为字节类型。

Str.encode([encoding=” utf-8”][,errors=” strict”])

Str:表示要进行转换的字符串

Encoding=”utf-8”:用于指定转码时采用的字符编码，默认为 UTF-8，如果使用简体中文，也可以设置为 gb2312.当只有这个参数时，可以省略 encoding=，直接写编码

Errors=“strict”：用于指定错误处理方式，strict（遇到非法字符就抛出异常）ignore（忽略非法字符）replace（用？代替非法字符）默认值为 strict

Eg:

```
>>> verse='野渡无人舟自横'
>>> byte=verse.encode("GBK")
>>> print(byte)
b'\xd2\xb0\xb6\xc9\xce\xde\xc8\xcb\xd6\xdb\xd7\xd4\xba\xe1'
```

Bytes.decode([encoding=' utf-8'][,error=strict])

```
>>> print(byte.decode('GBK'))
野渡无人舟自横
```

五、 正则表达式

在处理字符串中，经常会有查找符合某些复杂规则的字符串的需求。正则表达式就是描述这些规则的工具。正则表达式就是记录文本规则的代码。

1、行定位符：“^”表示行的开始；“$”表示行的结尾。

Eg:^tm:“tm equal tomorrow moon”; tm$:“tomorrow moon equal tm”;tm:匹配字符可出现任意位置

2、 元字符：

代 码	说 明	举 例
.	匹配除换行符以外的任意字符	.可以匹配“mr\nM\tR”中的 m、r、M、\t、R
\w	匹配字母、数字、下划线或汉字	\w 可以匹配“m 中 7r\n”中的“m、中、7、r”，但不能匹配 \n
\W	匹配除字母、数字、下划线或汉字以外的字符	\W 可以匹配“m 中 7r\n”中的 \n，但不能匹配“m、中、7、r”
\s	匹配单个的空白符（包括 Tab 键和换行符）	\s 可以匹配“mr\tMR”中的 \t
\S	除单个空白符（包括 Tab 键和换行符）以外的所有字符	\S 或以匹配“mr\tMR”中的 m、r、M、R
\b	匹配单词的开始或结束，单词的分界符通常是空格，标点符号或者换行	在“I like mr or am”字符串中，\bm 与 mr 中的 m 相匹配，但与 am 中的 m 不匹配
\d	匹配数字	\d 可以与“m7ri”中的字符 7 匹配

Eg:\bmr\w*\b：以 mr 为开头的单词

3、 限定符：

使用\w*匹配任意数量的字母或数字。如果想匹配特定数量的数字，那么就应该使用限定符来实现。

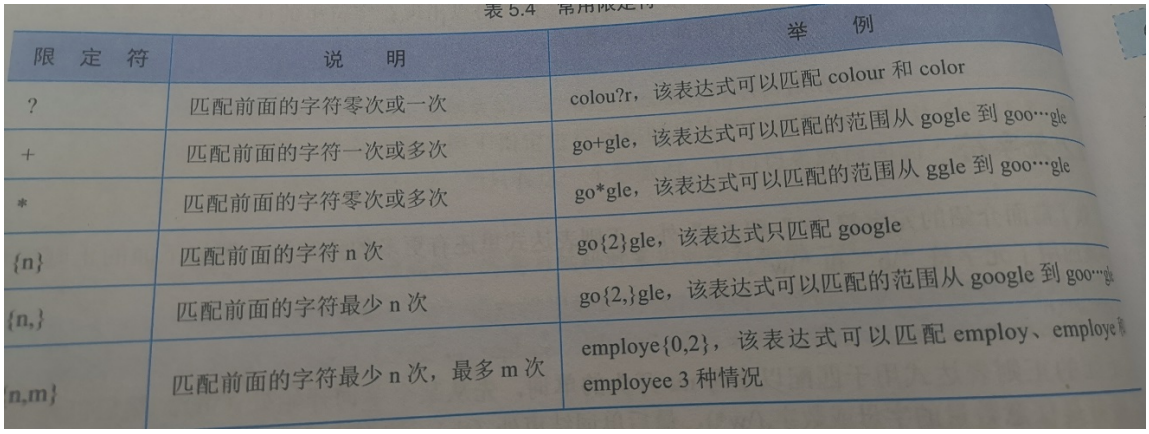

表 5.4 常用限定符

限定符	说明	举例
?	匹配前面的字符零次或一次	colou?r，该表达式可以匹配 colour 和 color
+	匹配前面的字符一次或多次	go+gle，该表达式可以匹配的范围从 gogle 到 goo…gle
*	匹配前面的字符零次或多次	go*gle，该表达式可以匹配的范围从 ggle 到 goo…gle
{n}	匹配前面的字符 n 次	go{2}gle，该表达式只匹配 google
{n,}	匹配前面的字符最少 n 次	go{2,}gle，该表达式可以匹配的范围从 google 到 goo…
{n,m}	匹配前面的字符最少 n 次，最多 m 次	employe{0,2}，该表达式可以匹配 employ、employe、employee 3 种情况

4、 字符和排除字符：

匹配元音：[aeiou]

匹配标点符号：[!.?]

匹配数字：[0-9]

匹配字母和数字:[0-9a-zA-Z]

匹配汉字：[\u4e00-\u9fa5]

匹配不是字母的:[^a-zA-Z]

5、 选择字符：“|”

Eg:身份证长度为 15 位或 18 位，15 位时全部为数字，18 位时前 17 位是数字，最后一位可能为数字或者 X

匹配身份证的正则表达式可以是：

(^\d{15}$)|(^\d{18}$)|(^\d{17}X$)

6、转义字符：”\”

Eg:用正则表达式匹配“127.0.0.1”格式的 IP 地址

^[1-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}

由于.在正则表达式中有特殊含义，所以必须要用转义字符处理

7、分组：小括号

Eg:(six|four)th 匹配 sixth 或者 fourth

(\.[0-9]{1,3}){3}就是对 IP 地址的例子中后边三项的重复操作

六、 Python 中的 re 模块实现正则表达式操作

1、 在 Python 中使用正则表达式

匹配开头是 m 的单词的正则表达式转为模式字符串，不能够直接在两侧添加引号定界符，例如下边的代码是不对的：\bm\w*\b 而是应该将其中的“\”进行转义，结果为:\\bm\\w*\\b.因此推荐使用 raw 原生字符串来生成模式字符串：r’ \bm\w*\b’

2、 Re 模块中匹配字符串

match(pattern,repl):从字符串的开始处进行匹配，如果在起始位置匹配成功，则返回 match 对象，否则 None

search(pattern,repl):用于在整个字符串中搜索出第一个匹配的值，如果匹配成功，则返回 match 对象，否则 None

findall(pattern,repl):用于在整个字符串中搜索所以符合正则表达式的字符串，并以列表的形式返回，如果成功，返回含有匹配结构的列表，否则返回空列表。

Eg:验证是否出现危险字符

```
import re
pattern=r'(黑客)|(抓包)|(监听)|(Trojan)'
about="我是一名程序员，我喜欢看黑客方面的书籍，想研究一下Trojan"
match=re.search(pattern,about)
if match==None:
    print(about,"@安全")
else:
    print(about,"@出现了敏感词汇")

about="我是一名程序员，我在学习Python,喜欢开发网站"
match=re.search(pattern,about)
if match==None:
    print(about,"@安全")
else:
    print(about,"@出现了敏感词汇")
```

```
========== RESTART: C:/Users/liuyi/Desktop/1.py ==========
我是一名程序员，我喜欢看黑客方面的书籍，想研究一下Trojan @出现了敏感词汇
我是一名程序员，我在学习Python,喜欢开发网站 @安全
```

3、 Re 模块中替换字符串

re.sub(pattern,repl,string,count,flags)

pattern:模式字符串

repl：要替换的字符串

string:表示要被查找替换的字符串

count:替换次数，默认为全部

flags:表示标志位，控制匹配模式，如是否区分大小写等。

Eg:隐藏中奖号码

```
import re
pattern=r'1[34578]\d{9}'
about="中奖号码为849789981，联系电话为13488738116"
result=re.sub(pattern,"1xxxxxxxxxx",about)
print(result)
```

```
========== RESTART: C:/Users/liuyi/Desktop/1.py ==========
中奖号码为849789981，联系电话为1xxxxxxxxxx
```

4、 Re 分割字符串

re.split(pattern,string,[maxsplit],[flags])