

用户在使用账号登录微博网站时,需要输入自己的账号和密码。这样，当用户登录时,浏 览器会把输入的账号和密码传递到后台服务器，服务器会把刚输入的这些内容与以前保存的用 户信息进行比对，如果信息匹配，那么用户就可以顺利进入到当前账户的主页。

在整个过程中，会产生有着字母、特殊符号或者数字的一串内容，如密码，这时需要找个 变量暂时保存。在Python中，什么类型能保存像密码这样的数据呢？答案是字符串。程序设计 中，字符串是符号或者数值的一个连续序列，用来表示文本的数据类型。本章将针对字符串进 行详细讲解。

<1宇符鵰奇餾

不同的计算机语言表示字符串的方式是不同的，比如C语言使用双引号标识字符串。在 Pythoii中共有三种表示字符串的方式：单引号、双引号、三引号（三个单引号或双引号），下 面逐个来介绍这些表示方式。

第1种：使用单引号表示

使用单引号包含的内容可以表示一个字符串。需要注意的是，在单引号表示的字符串中可 以嵌套双引号，但是不允许再嵌套单引号。

示例如下:

»> ?Hello! #单引号包含字符

»>，他说：”你好！ J #单引号表示的字符串中包含双引号

\*他说:°你好! ° !

»> vletss gof #单引号表示的字符串中不允许包含单引号

File n<stdin>° r line 1

1. let1s go 1

SyntaxError: invalid syntax

第2种辭使用双引号表示

使用双引号包含的内容也可以表示一个字符串。这里要注意一点，使用双引号表示的字符

|  |  |  |
| --- | --- | --- |
| 串中，允许嵌入单引号，但不允许包含双引号。示例如下： | | |
| >>> "World" | # | 双引号包含字符 |
| 'World , |  |  |
| >>> ° say: \* Hi ? ° | # | 双引号表示的字符串中包含单引号 |
| "say: \* Hi , °  »> "他说:°你在干什么"" | # | 双引号表示的字符串中不允许包含双引号 |

File n<std.in>°, line 1 °他说:°你在干什么f

SyntaxError: invalid syntax

第3种：使用三引号表示

使用三引号（三个单引号或者三个双引号）包含的内容可以表示多行字符串，在这个字符 串中可以包含换行符、制表符或者其他特殊的字符。通常情况下，三引号表示的字符串表示的 是文档字符串，用来说明包、模块类或者函数的功能。

下面是对函数说明的一个示例代码：

def len（\*args, \*\*kwargs）:

n"" Return the number of items in a container a "

pass

跟普通的注释相比，使用三引号标注的注释会作为函数的一个默认属性，可以通过“函数名。 \_doc\_n进行访问。

（気"注意；Pythoi!中没有字符的概念，即使只有一个字母，也属于字符串类型。

<2字綺鵰存儲方或

金21拿務露酶養昌I

字符串就是一个由字符（这里的字符是指计算机中使用的字母、数字、汉字和符号，包括L 2, 3, A, B9 " #9 ¥, %,…，一，\*（,）,——，+,…）组成的序列，每个

字符所处的位置是固定的，有顺序的。字符串的每个字符都对应着一个位置编号，它是从。开 始的，然后依次递增L这个位置编号就是索引或者下标。

为了大家更好地理解，下面通过图4』来进行描述。

在图心 中，索引是按照从左向右的顺序排列的，这是正向索引。如果按照从右向左排列， 就是反向索引。反向索引的示意图如图4・2所示。

|  |  |  |  |  | L————\* |
| --- | --- | --- | --- | --- | --- |
| P | y | t | h | 0 | n |

python

0 1 2 3 4 5

图4-1字符串的索引（正向）

二4 …亠…一

-6 -5 -4 -3 -2 -1

图4-2字符串的索引（反向）

在图4-2中，字符串的末尾字符的反向索引为-L依次向左开始减L首位字符的反向索 引为-6。

4.2.2體瘋養引遮闻值

字符串中的字符是根据索引标记的，如果希望获取字符串的任意字符，可以使用索引来获 取。使用索引访问字符串值的语法格式如下：

字符串[索引]

例如，变量string引用了字符串?abcdef \在内存中存储的方式如图4-3所示。

在图4；中9索引可以看作字符串中各个字符的入口，每个字符对应着两个索引，分别为 正向索引和反向索引。由图4.3可知，如果要从字符串中取出c这个字符，它对应的索引为2和-4,

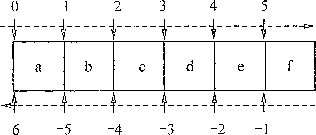
所以可通过如下两种方式获取。 方式一：

图4.3字符串的索引

>>> string[2]

\* c 1

方式二

>>> string[-4]

f c 1

需要注意的是，当使用索引访问字符串值的时候，索引值的范围不能越界，否则程序会报 索引越界的异常。示例如下：

>>> string[6]

Traceback (most recent call last):

File "<stdin>"f line 1, in <module>

IndexError: string index out of range

4o2q3拿播額憂釋爾靈鱸

与C语言不同9 Pythoii中的字符串是不能改变的。如果希望对某个索引位置赋值，就会导 致错误。例如，把字符串sHelloWorld,中索引位置为5的字符W替换为w,示例如下：

>>> string = 'HelloWorld5

>>> string[5] = !w f

Traceback (most recent call last): File "<stdin>"r line 1, in <module>

TypeError: 1 str v obj ect does not support item assignment

上述示例中”尝试着把某个索引位置的字符赋值为其他字符，结果程序报错。出现上述错 误信息，原因在于字符串对象是不可变的。

当对字符串做追加、修改、截取等一些改变字符串内容的操作时，Pythoii会在内存中新建 一个字符串。例如，使用连接操作符(+ )拼接两个字符串的结果歹也是一个新的字符串，示 例如下:

>>> string = f Hello^ '

>>> string += \* Pythons

>>> s

? Helloz Python!

上述示例中，s变量的值最初为IfclkA然后使用+=拼接了 Yython\再次赋值给变量s, 使得s变量的值改为IfcHo,Python，。从表面上来看，字符串的内容发生了变化，事实上在进行 拼接时又创建了新的字符串，原来的字符串fHello?已经被释放掉了。

为了证明上述说法是正确的，下面使用id()函数获取Wing变化前后的内存地址。若两个 内存地址是一样的，表示没有创建新的字符串，反之则表示创建了新的字符串。

>>> string = 5 Hello7 \*

>>> id(string)

4329464864

»>

>>> string += !Pythons

>>> id(string)

4329476336

从输出的结果可以看出，string引用的地址发生了变化，表示生成了新的字符串。

4.3场片職恥字简睹

使用索引能获取单个字符，切片可以获取一个子字符串。那么什么是切片呢？切片是指对 操作的对象截取其中一部分的操作，字符串、列表、元组都支持切片操作。这里9我们以字符 串为例来讲解切片的使用。

切片的语法格式如下：

[起始:结束:步长]

*很"在*意：切片选取的区间属于左闭右开型，即从“起始”位开始，到“结束”位的前一位 结束(不包含结束位本身)。才艮据步长的取值，可以分为如下两种情况：

(1 )步长大于0 .

按照从左到右的顺序，每隔“步长』”(索引间的差值仍为步长值)个字符进行一次截取。 这时，“起始”指向的位置应该在“结束”指向的位置的左边，否则返回值为空。

示例如下:

>>> string = 5 python ®

>>> string [0:6] #没指定步长，默认为1

\* python 1

»> string [2:5:2]

!to5

在上述示例中,如果没有指定步长,则默认为L以第2个示例为例,通过示意图来分析 切片的原理，如图4『4所示。

(2)步长小于0

按照从右到左的顺序，每隔“步长T”(索引间的差值仍为步长值)个字符进行一次截取。 这时，“起始”指向的位置应该在“结束”指向的位置的右边，否则返回值为空。

示例如下:

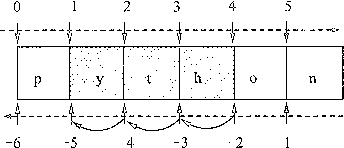
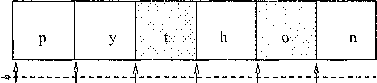
>>> string = 1python 1

>>> string[3:0:-1]

5hty!

>>> string[0:3:-2]

起始位置的索引必须大于结束位置的索引，否则返回空字符串。以第1个示例为例,通过 示意图来分析切片的原理，如图4・5所示。



0 1 2 3 4 5

-6 -5 -4 -3 -2 -1

图4-5切片示意图（步长小于0）

图44切片示意图（步长大于0）

综上可知，利用下标的组合截取原字符串的全部字符或部分字符。如果截取的是字符串的 部分字符，则会开辟新的空间来临时存放这个截取后的字符串。

切片的索引有着非常有用的默认值。如果省略了切片的起始位置，则默认切片的起始值 为0;如果省略了切片的结束位置，则默认切片的结束值为字符串的长度。

示例如下:

>>> word = \* abcdef!

>>> word[:]

9 abcdef'

>>> word[0:]

'abcdef'

>» word [ : 3 ]

'abc ?

»> word [: : -1 ]

f fedcba ?

正向索引可以结合反向索引混用达到截取字符串的结果。示例如下：

>>> word[0:-1]

'abcde 1

>>> word[0:-3]

J abc'

需要注意的是，程序试图使用下标太大的索引（即下标值大于字符串实际的长度）获取字 符时，肯定会导致越界异常。但是，Python处理了那些没有意义的切片索引，一个过大的索引 值将被字符串的实际长度所代替，如果上边界比下边界大时（即切片起始的值大于结束的值）， 则会返回空字符串。

示例如下：

>>> word = 'python \*

>» word [4:40]

?onf

>>> word[40:]

I y

无论使用单引号或者双引号包含的字符串，值都是一样的。那么，为什么要用两种情况表 示字符串呢？这是因为在某些特殊情况下，单引号或者双引号会派上用场。

例如,使用双引号包含带有双引号的字符串，示例如下:

»> ”他说:°你在干什么?°°

File °<stdin>", line 1 "他说:°你在干什么?° °

A

SyntaxError: invalid syntax

在上述示例中,解释器会通过匹配双引号来辨别字符串的结束符,所以会发生错误。同样, 使用单引号中包含带有单引号的字符，会出现相同的错误。为了解决上述问题，需要单独处理 这些比较特殊的符号，我们可以使用对它们进行转义。对上述示例进行修改，具体如下：

»>間他说:\ °你在干什么?\°°

'他说:°你在干什么?" 5

在上述示例中，使用斜线对双引号进行了转义，这样当解释器遇到转义符的时候歹就会 把其后面的字符与转义字符当作一个整体，表示一个双引号的字符，而非再当作字符串语法的 标志。

除了引号以外,Python有一些其他的转义字符,具体如表4』所示。

表转义字符

|  |  |  |
| --- | --- | --- |
| 运算符 | 说 | 明 |
| \（在行尾时） | 续行符 | |
| wiiih | 反斜杠符号 | |
|  | 双引号 | |
| \b | 退格(Backspace) | |
| \e | 转义 | |
| \000 | 空 | |
| \n | 换行 | |
| \v | 纵向制表符 | |

|  |  |
| --- | --- |
| 续表 | |
| 运算符 . | 说 明 |
| \t | 横向制表符 |
| \r | 回车 |
| \f | 换页 |
| \oyy | 八进制数，yy代表的字符，例如；\ol2代表换行 |
| \xyy | 十六进制数，yy代表的字符，例如言WOa代表换行 |
| \other | 其他的字符以普通格式输出 |

4L圍思 録牆原嬉寥簿額

如果你不想使用反斜杠（\）转义特殊字符，可以在字符串的前面添加一个『或者R,当作 原始字符串使用。在原始字符串中，所有的字符都是直接按照字面的意思来使用，没有转义特 殊字符或不能打印的字符。

示例如下:

>>> print（\*Ru\noob1）

Ru

oob

»>

>>> print（r \* Ru\noob1）

Ru\noob

在上述示例中，\n扮演着换行的角色，如第1个示例。当在整个字符串的开头加上『以后, \ri就失去了特殊的含义，仅作为两个普通字符使用，如第2个示例。

衫R表示原始字符串，能够禁止转义字符的实际意义。原始字符串除在字符串的第一个引 号前加上字母”产（可以大小写）以外，与普通字符串有着几乎完全相同的语法。

（W注意:在版本3.3中,为了向后兼容Python2系列,在字符串文字上再次允许使用u前缀, 它'寿字符串文字的含义没有影响，且不能与『前缀进行组合。

4o 5字简鵰的論冊和输入

4.5J 拿籍辭髓檣就継讎器

在Python中内置有字符串的格式化操作,所以print（）函数支持格式化输出。在格式化字 符串时，Python会插入格式操作符（如％s）到字符串中，为真实的数值预留位置，并说明真 实数值需要呈现的格式。例如，"hello%sn使用格式操作符％s为真实的数值占位，由于％s 只能给字符串类型的值占位，所以真实的数值必须为字符串类型。

最简单的用法是，把一个值插入一个有格式化操作符的字符串中，示例如下：

>>> name =° 小明 °

»> print （"大家好,我叫 %s"%name）

大家好，我叫小明

在上述示例print()函数的括号中，“大*家好，*我叫％s”是一个带有格式操作符%s的字符 串，该字符串后面的％表示对字符串进行格式化操作，即把％后面的name作为真实的数值， 插入字符串中％s占用的位置。

如果字符串中有多个格式操作符，那么就需要使用元组来插入相应数目的值。示例如下:

>>> name ="小明"

>>> age=20

»> print ("大家好,我叫 %s!今年％d 岁了 ! "% (name, age))

大家好，我叫小明，今年20岁了 ！

在上述代码中，printO语句中的“大家好，我叫％&今年％(1岁了！”是带有两个格式化 操作符的字符串。其中，％s作为第一个格式操作符标记了要插入的位置，表示真实的值会被 格式化为字符串；％(1作为第二个格式操作符同样标记了要插入的位置，表示真实的值会被格 式化为整数。元组(name, age)中的name和age元素作为真实的数值，分别会插入字符串中%s 和％<1占用的位置。

如果要格式化其他类型的数据，可以使用对应的格式化操作符。常用的格式化操作符如 表4-2所ZK。

» 4-2常见的格式化符号

|  |  |  |  |
| --- | --- | --- | --- |
| 格式化操作符 | 说 明 | 格式化操作符 | 说 明 |
| %c | 字符 | %X | 十六进制整数(大写字母) |
| %s | 通过str()字符串转换来格式化 | %e | 索引符号(小写e) |
| %i | 有符号十进制整数 | %E | 索引符号(大写E) |
| %d | 有符号十进制整数 | %f | 浮点实数 |
| %o | 八进制整数 | %g | % f和％ e的简写 |
| %x | 十六进制整数(小写字母) | %G | %f和％ E的简写 |

此外，Pythoii在格式化字符串时支持用字典作为要插入的真实数值。当要插入字典到有格 式操作符的字符串时，字典的键作为格式化操作符的参数进行占位，而字典的值会根据键所在 的位置进行替换。

示例如下:

>>> die = { 1 name 1 : \* 小明 5 r ' age ? : 2 9}

>>> die

{ \* name 1 : 1 小明 L \* age 1 : 2 9)

»> print ( 1 我是 % (name) s,年龄 % (age) d 岁*。*'%dic)

我是小明，年龄29岁。

像上面在格式化字符串中使用到字典类型时，需要在％后面插入字典的键。除了这种情 况以外，还有一些应用于格式化操作的辅助指令，接下来使用一张表格来列举这些辅助指令， 具体如表4・3所示。

表4-3格式化操作符辅助指令

|  |  |
| --- | --- |
| 辅助符号 | 作用说明 |
| \* | 定义宽度或者小数点精度 |
| - | 用作左对齐 |
| + | 正数前面显示加号 |
| <sp> | 正数前面显示空格 |
| # | 八进制数前面显示零，十六进制前面显示皈或者0X （取决于用的是x还是X） |
| 0 | 显示的数字前面填充0,而不是默认的空格 |
| % | %%输出一个单一的% |
| (var) | 映射变量（字典参数） |
| m.n | m显示的是最小总宽度，n是小数点后的位数（如果可用的话） |

挑选上述几个比较典型的辅助指令举例，代码如下:

»> a=100

»> b=10

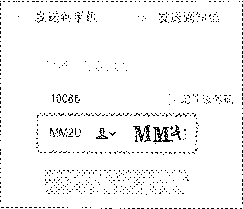
>>> c=a%b

>>> print (，s %d%%%d=%dJ % (az b7 c))

100%10=0

>>> print (5 a 的值为 %+d1 %a) a的值为+100

<5.3寥簿麝餉鑰人

相信很多人经历过网上购票，并且很多用户都在吐槽验证码。

相对而言，有些网站的验证

图4\*6百度地图验证码

码就简单一点，如图4・6所示。

为了检查用户输入的验证码是否正确，首先要获取用户输入 的字符串，然后跟系统显示的验证码进行比对。为了解决这种情 况,Python 3提供了 input函数（函数就是实现某种功能的一段代 码，后面会有介绍）从标准输入读取一行文本，默认的标准输入 是键盘。

使用input函数读取文本的代码如下:

>>> captcha = input （）

123456

»> captcha

'123456®

在上述示例中，使用变量captcha接收了用户输入的内容。从captcha的值可以看出，用户 起初输入的是数值类型，但captcha的结果却是字符串类型的。下面使用type函数进行测试， 具体如下:

>>> type（captcha）

<class \* str1>

由此说明，input函数会把接收到的内容进行包装，转换成字符串类型。

input函数的括号中能放入提示信息，用于获取数据前给用户提示。示例如下: >» captcha = input (° 请输入验证码：° )

请输入验证码:8888

>>> captcha

J8888v *(宓*注意：在Python 2中9共有raw\_input和input两个函数负责接收键盘的输入，而Python *g岫a*

3中的只有一个input函数,其用法与Python 2中的raw input函数的用法是一致的。

:<6字简鵰模板

正如前面所介绍的那样，使用格式操作符可以给字符串传递参数，实现字符串的格式化。 对于Python的新手而言，格式化的字符串不仅不够直观，而且很容易因搞混格式操作符的类型 岀现错误。为此，程序员必须明确记住要转换类型的格式操作符，例如，转换为字符串类型的 格式操作符是％s、转换为整型的格式操作符是％d。

为了解决上述这些问题,Python 24开始加入了新式的字符串模板。字符串模板的工作方 式类似于sh紺里面的变量替换，默认变量是使用$符号定义的。新字符串模板用Temp就e对 象表示，该类对象提供了如下方法替换$定义的变量，具体如下：

-substitute()方法：若不能提供模板所需要的全部变量值时，会发生异常。

• safe\_ substitute()方法:安全替换,该方法不仅会替换模板中存在的变量的值,而且会保 留不存在的变量。.

上述两个方法相比，第1个方法更为严谨一些。当缺少变量的情况下，程序会报一个 KeyErnr异常；而第2个方法在缺少变量时，会直接原封不动地显示字符串。接下来，以第1 个方法为例，介绍字符串模板的使用。

1Q使照substitute方法用值替换相应的参饑

substitute这个模板方法会用传递过来的参数值替换字符串中相应的参数,例如，把foo=29 替换字符串中的$化。。

示例如下:

>>> from string import Template

>>> string = Template ( , 我叫 $name,今年 $age 岁。' )

>>> string. substitute (name= \* 小明 ' r age=28)

'我叫小明，今年28岁。，

从上述示例看出，仅仅使用一个$符号，就能替换多种类型的值歹所以程序员无须再记住 转换的类型。

2.使用substitute有法營渙郭翁寥简嬲

如果要替换的字段只是字符串内容的一部分，则参数的名称必须用花括号括起来，以明确 地指出要替换字符串部分的结束位置。

示例如下:

»> string = Template ( 1 It is $ {x} ful! 1 )

70*临* Pyfthoo囊tt豔麗；映霸掌Pythoo

>>> string.substitute(x= 8wonder')

, It is wonderful! \*

3嗣使用$$牆入$简蜀

如果希望$符号不再具备任何特殊的含义，需要使用$$实现。

示例如下:

»> string=Template(\* make $$ selling $x! 1)

>>> string.substitute(x=f demo 5)

,make $ selling demo! 1

冬使闢享製提供键值对

除了使用关键字以外，还能够使用字典的键充当字符串的参数，而字典的值用于替换参数 的值。

示例如下:

>>> string = Template ( ' 我叫 $namef 今年 $ age。 ' )

>>> dic={}

>>> die [! name J ] =5 小米 J

»> die [ \* age，] =2 9

»> string. substitute (die)

1我叫小米\*今年2 9。5

<7宇简跚的內建函數

Pythoii提供了一些和字符串相关的内建函数，它们能够快速地让对象执行简单的操作，具 体包括序列类型的操作(如 即 等)、类型转换操作(如 血)。下面针对字符串的内建函数进 行介绍。

字符串是一个文本序列，它支持很多跟序列相关的操作，包括获取字符串的长度(kn()函 数)、字符串的最大和最小字符(max()函数和min()函数)等。下面介绍这些函数在字符串中 的运用，具体如下：

1 len()函数

眞()函数能够返回对象(字符串、列表、元组等)的长度或项目个数,语法格式如下： len(str)

示例如下:

>>> word='abede!

>>> len(word)

5

2. max()額 mm()函数 . ' .

max()函数用于返回字符串中ASCII值最大的字母，语法格式如下:

max(str)

相反，min()函数用于返回字符串中ASCH值最小的字母。语法格式如下：

min(str)

这两个函数对于像列表这样的序列更适用一些，在字符串中只能用于返回最大或者最小的 字符。

示例如下:

>» max (\* abc ?)

\* c'

>>> max(1123abcABC f)

! c'

>>> min('efg \*)

¥ e »

»> min ( v 123abcABC?)

fl!

& enumerateO 函数

°niimemtR()函数用于把一个可以遍历的数据对象(如字符串、列表或元组)组合成一个索 引序列，同时列岀数据和数据下标，通常用于版循环中得到计数，语法格式如下：

enumerate(iterable, start=0)

在上述语法中，iterable必须是序列、迭代器或者支持迭代的其他对象，start指定数据下标 的起始值，默认为0,示例如下：

>>> string=f abcdefg1

»> for index, value in enumerate (string):

..• print(index,value)

0 a

1. b
2. c
3. d
4. e
5. f
6. g

盛潔卷 寥摘辭雾饗鶉渙緝美函數

下面介绍几个转换为字符串类型的函数，具体如下：

(1 ) chr()函数

用一个范围在range(256)内(0〜255 )的整数作为参数，返回一个对应的字符串，语法格 式如下：

chr(i)

(冷注意：i的值既可以是十进制的数字，也可以是十六进制形式的数字，返回的是当前整 数所对应的ASCII字符。示例如下：

>» chr (0x31) # 十六进制

? 1 '

»> chr (97) # 十进制

(2 ) ord()函数

ord()函数是chr()或者unichrQ的配对函数，它以一个长度为1的字符串作为参数，返回 其对应的ASCII数值或者Unicode数值，语法格式如下：

ord(c)

上述语法中，。代表长度为1的字符。如果Unicode字符串超出了范围，则会引发 TypeError 的异常。

示例如下:

>>> ord(!a \*)

97

»> ord(s\u234567?)

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: ord() expected a characterz but string of length 8 found

<8字简鵰飽靄O錢

字符串作为最常用的数据类型，Pythoii提供了很多供开发者使用的方法，这些方法封装了 一些关于字符串的简单操作。接下来，通过表4Y来列举目前字符串支持的内建方法。

表&4字符串的常见方法(按字母排序)

|  |  |
| --- | --- |
| 函数名 . | 相关描述 |
| capitalize() | 把字符串的第一个字符转换成大写 |
| casefbld() | 其效果和kwer()类似，把全部字符串转换为小写。如果在汉语和英语环境下可以使用 lower()*方法*§如果要处理其他语言的大小写问题，使用casefoldO方法 |
| center(width[, fillchar]) | 将字符串按照width指定的宽度居中显示，且使用fillchar 认为空格)填充多余的长度。  若指定的长度小于原字符串的长度，则返回原字符串。 |
| count(sub[, startf, end]]) | 返回sub在字符串里面出现的次数。如果指定了 start或者end,则返回指定范围内sub出 现的次数 |
| encode(encoding="utf-8",  errors=" strict") | 以encoding指定的编码格式编码字符串。如果出错默认会报一个ValueError的异常，除 非errors指定的是ignore或者replace |
| endswith(suffix[, startf, end]]) | 检查字符串是否以suffix结束。如果指定了 start或者end,则检查指定的范围内是否以 suffix结束，如果是返回True,否则返回而屐 |
| expandtabs(tabsize=8) | 把字符串中的tab符号转换为空格，默认空格数为8 |
| find(sub[, start[, end]]) | 检查sub是否包含在字符串中。如果指定了 start或者end,则在指定范围内检查。如果包 含，那么返回开始的索引值，否则返回-1 |
| fbrmat(\*args, \*\*kwargs) | 格式化字符串 |
| fbrmat\_map(mapping) | 与str.fbrmat (\*\*映射)类似，除了直接使用映射并且不复制到diet |
| index(sub[, start[? end]]) | 跟find()函数用法一样，只不过sub不在字符串中会报一个异常 |
| isalnum() | 如果字符串中至少有一个字符，并且所有字符都是字母或数字则返回Tme,否则返回  False |

|  |  |
| --- | --- |
|  | 续表 |
| 函数名 | 相关描述 |
| isalpha() | 如果字符串至少有一个字符，并且所有字符都是字母则返回True,否则返回False |
| isdecimalQ | 如果字符串只包含十进制数字，则返回True,否则返回False |
| isdigit() | 如果字符串只包含数字，则返回True,否则返回False |
| isidentifier() | 判断字符串是否是合法的标识符，实际上这里判断的是变量名是否合法 |
| islower() | 如果字符串中包含至少一个区分大小写的字符，并且所有这些（区分大小写的）字符都是 小写，则返回True,否则返回False |
| isnumeric() | 如果字符串中只包含数字字符，则返回True,否则返回False |
| isprintable() | 判断字符串所包含的字符是否全部可以打印。如果字符串中包含不可打印的字符，如转 义字符，则会返回關se |
| isspace() | 如果字符串中只包含空格，则返回True,否则返回False |
| istitle() | 如果字符串是标题化的（见title（））9则返回True,否则返回False |
| isupperQ | 如果字符串中包含至少一个区分大小写的字符，并且所有这些（区分大小写的）字符都是 大写则返回True,否则返回False |
| join(iterable) | 以指定的字符串作为分隔符，将iterable中所有的元素（字符串表示）合并为一个新的字 符串 |
| ljust(width[5 fillchar]) | 将字符串按照斯dEh指定的宽度左对齐显示，且使用fillchar （默认为空格）填充多余的长 度。若指定的长度小于原字符串的长度，则返回原字符串 |
| lowerQ | 将字符串中的所有字母都转为大写 |
| lstrip( [chars]) | 截掉字符串左边的空格 |
| maketrans(x[, y[, z]]) | 返回技anslafe。方法的转换表。如果只有一个参数，它必须是diet类型，键key为长度为 1的字符（Unicode字符码或者字符），值value为任意长度字符串或者None。键key对应的 字符将被转换为值value对应的字符（串）。如果有两个参数，他们长度必须相等，每一个x 字符将被转换为对应的y字符。如果有第三个参数，其对应的字符将被转换为None |
| partition(sep) | 该方法用于拆分字符串，返回一个包含三个元素的元组。如果没有在字符串中找到 则元组的三个元素为；原字符串、空串、空串号否则从字符串中遇到第一个sep字符串开始 拆分，元组的三个元素为f ep之前的字符串、sep、sep之后的字符串 |
| replace(oid, new[, count]) | 把字符串中的old替换成newo如果count指定，则替换不超过count次。 |
| rfind(sub[5 start[5 end]]) | 与find（）函数用法一样，不过是从右边开始査找 |
| rindex(sub[, start[, end]]) | 与index。函数用法一样，不过是从右边开始査找 |
| ijust(width[, fillchar]) | 将字符串按照wid钮指定的宽度右对齐显示，且使用fillchar （默认为空格）填充多余的长 度。若指定的长度小于原字符串的长度，则返回原字符串 |
| rpartition(sep) | 与partitionQ函数的用法一样，不过是从右边开始查找 |
| rsplit(sep=None, maxsplit—1) | 与split（）方法类似，只是它从最右边开始拆分,只有在指定maxsplit的情况下才会看到效果。 |
| rstrip( [chars]) | 删除字符串末尾的空格 |
| split(sep=None, maxsplit—1) | 以sep为分隔符截取字符串，如果maxsplit有指定值,则仅分隔maxsplit个子字符串 |

续表

|  |  |
| --- | --- |
| 函数名 | 相关描述 |
| splitlines( [keepends]) | 按照行(W, W, 分隔，返回一个包含各行作为元素的列表。如果参数keepends为  False,不包含换行符，如果为True,则保留换行符 |
| startswith(prefix[, start], end]]) | 检查字符串是否是以prefix开头。如果是则返回True,否则返回False0如果start和end 指定值，则在指定范围内检查 |
| strip([chars]) | 在字符串上执行lstrip()和rstrip() |
| swapcase() | 删除字符串中的大小写 |
| title() | 返回”标题化”的字符串，就是说所有单词都是以大写开始，其余字母均为小写 |
| translate(table[, deletechars]) | 根据参数table给出的表(包含256个字符)转换字符串的字符，要过滤掉的字符放到 deletechars 参数中 |
| upper。 | 将字符串中的小写字母转换为大写 |
| zfill(width) | 返回长度为width的字符串，原字符串右对齐，前面填充0 |

4D8n1 fmd SB

find方法用于在指定的范围内检测字符串中是否包含子串，语法格式如下：

str .find (sub [ 7 start [ end]])

在上述语法中，可选参数start和end可以理解为像切片标记一样，在切片s[start:end]截取 的范围内查找子串(sub),默认为整个字符串。如果字符串中包含子串，返回子串出现的最小 索引值,否则返回-lo

find方法的使用示例如下所示。

>» word = ? Helloz Welcome to itheima ?

>>> word.find (\* itheima \*)

17 .

>» word,find ( \* itcast' )

-1

>>> word .find ( \* itheima ! f 0,15)

-1 .. . :， :

注意:find方法仅仅适用于查找子串出现的位置,如果要检查sub是否为子字符串,需要 用到iii操作符。示例如下:

>>> , Py \* in v Python

True h方寸注意：由起始和终止值指定的范围包含第1个索引，但不包含第2个索引,这在Python

一.…

中是个惯例。

4L段卷 aodex •

index方法用于在字符串中查找子串第一次岀现的索引位置。如果找到了子串，则返回开 始的索引值，否则会抛出VahiRError异常，语法格式如下：

str.index(sub[f start[r end]])

在上述语法中,如果指定了 start和end的搜索范围,程序会在此范围内查找sub子串,默 认搜索的范围是整个字符串。

index方法的使用示例如下所示。

>>> word= f itheima and itcast \*

>>> word.index('it1)

0

»> word, index ( 1 it \ 2)

12

>>> word.index(? it S 2 f10)

Traceback (most recent call last):

File "<stdin>"z line 1z in <module>

ValueError: substring not found

4d@d3 repfece 有逶

replace方法用于把当前的旧字符串替换成新字符串，可选择地限制替代数量。

语法格式如下：

str„replace(selff old^ newz count=None)

在上述语法中，now作为新的子串会取代old旧子串，方法返回的是字符串中old替换成 new后生成的新字符串。如果给定了可选参数count,则替换不超过count次。

replace方法的使用示例如下:

>>> word = °我是小明,我今年28岁°

>>> word. replace (”我"/ 他")

，他是小明，他今年28岁，

>>> word. replace ("我”/他"，1)

,他是小明，我今年28岁'

如果在字符串中没有找到匹配的子串，那么会直接返回原字符串。示例如下：

>>> word。replace ("他"/ 我“)

1我是小明，我今年28岁!

4L陽盛 spit SB

split方法用于把字符串以分隔符分隔成字符串列表。语法格式如下:

str.split(selff sep=Nonez maxsplit=-l)

在上述语法中，sep (分隔符)默认为空字符，包括空格、换行(\n)、制表符(\t)等。 如果maxsplit有指定值，则会分隔为maxsplit个子字符串学方法返回的是一个分隔以后的字符 串列表。

split方法的使用示例如下:

>>> word ="12345°

>>> word.split()

[11 \ \*31J f5»]

»> word = " a b z c z d7 e"

»> word, split (" 7 ")

[?a\ fbs , ?d\ U]

>>> word., split ! 7 3)

[?a! , sb\ \*c\ Jd,ef ]

4a@D5 tower SB

lower方法用于转换字符串中的所有大写字符为小写。语法格式如下:

str.lower(self)

如果遇到不区分大小写的情况，上述方法可以派上用场了。例如，用户在搜索栏输入关键 字时，总是习惯性地用小写字母搜索，所以这种场景下建议忽略大小写。

lower方法的使用示例如下:

>>> word = \* Lower J

>>> word.lower()

?lower J

>>> word = JLOWER1

>>> word»lower()

!lower \*

4.8.6 strips

strip方法用于去除字符串两侧的空格o语法格式如下:

str.strip(chars^None)

strip跟low©『方法配合使用，可以很好地用于搜索场景。例如，用户输入内容时在末尾多 加了一个空格歹这时就要用氾ip方法删除两头的空格。

strip方法的使用示例如下:

>>> word = ° Strip °

>>> word.strip()

\* Strip3 一 .

另外,可以指定要去除的字符,作为chars参数的值传递。示例如下:

»> word = n\*\*Strip\*\*n

»> word. strip (" \* n)

! Strip 5

4o8a7 format BIS

*从*Python 2.6开始,新增加了一种执行字符串格式化操作的方法——formate那么,它与 使用符号％格式化字符串的方式相比，存在哪些优势呢？大多数情况下，format方法与％操 作符的用法相似。fomat方法会把字符串当作一个模板,通过传入参数进行格式化,这个用来 格式化的模板是使用，(}?作为特殊符号的。

下面是一个示例：

>>> \* H{}\* .format('ello \*)

5 Hello1

如果需要在字符串中包含花括号({})字符，则可以通过两个花括号进行转义，如{{and}}。 花括号里面存放的是替换字段，它可以为整数，表示参数的位置，如{0}或卩}。如果格 式字符串中的数字顺序为饥1, 2,…'则可以全部省略，而是按照该顺序自动插入数字饥1,

示例如下:

»> '{0} {1} ' . format ( ! Hello , z \* Python \* ) \* Hello Python!

#设置参数的位置

#不设置指定位置，按照默认的顺序

#设置参数的位置

>>> \*{} {}1.format('Hello'f 5 Python')

1 Hello Python 1

»> ' 1 {1} { 0 } , o format ( f Hello 1 ? \* Python 1) 5 Python Hello?

{泞注意：在Python 3J版本中位置参数说明符可以省略，因此“{} {厂等价于“{0} {1}"。 除“此之外，字段的名称可以为关键字，表示一个命名的关键字参数。示例如下：

»> "My quest is {name} n . format (name= 1 flower v )

\* My quest is flower1

另外，还能够通过字典和列表索引来设置参数。示例如下：

#通过字典设置参数

>>> dic= ( \* name ' : ' Python 知识大全','produce \* : \* 传智播客 , }

>» 1 书名:(name},出处：{produce} f . format (\* \*my\_did)

!书名:Python知识大全/出处：传智播客f

#通过列表索引设置参数

»> arrays [ \* Python知识大全\* r \*传智播客1 ]

»> ?书名:{ 0 [ 0 ] } z 出处:{ 0 [ 1 ] } ' « format (array)

5书名:Python知识大全/出处：传智播客,

格式字符串可以指定格式限制符显示值的说明，包括字段宽度、对齐方式、填充、小数精 度等信息。指定格式限制符的语法如下：

{替换字段:格式限制符}

在上述语法中，在替换字段的后面使用冒号对其进行补充说明，比如用\*号填充或者向右 对齐。

通常在使用限制符时，填充和对齐是一起使用的，对齐分别用5 y和符号代表居中、 左对齐和右对齐，在这些对齐符号的后面可以跟上字段的宽度。如(1：>3)9表示在第}个位置 参数字段的冒号后面，先设置了向右对齐，再设置字段的宽度为3。

使用限制符的示例如下：

»> word= ' {0:8}$(1:>3} ? #第1个位置参数字段宽度为8 ,默认以空格填充

#第2个位置参数字段宽度为3,向右对齐

>>> word» format(? Potato ? z 35)

\* Potato $ 35'

上述示例中，第1个位置参数限制了字段的宽度为8,默认为左对齐；第2个位置参数明 确指出了向右对齐方式。需要注意的是，冒号后面带有填充的字符，只能是一个字符，如果不 指定，则默认是用空格填充的。所以在上述示例中，默认使用空格进行填充以达到预定的字段 宽度。

CS）弄筒鵰攤作鶴

已知变量a的值为»Hello" 9变量b的值为"iteast«o下面通过一张表列出字符串常用的操 作符，并使用实例演示这些符号所产生的结果，如表4・5所示。

|  |  |  |
| --- | --- | --- |
| 表名5常用的操作符 | | |
| 操作符 | 相关描述 | •实 例 |
| >，<,==，  !=， >=, <= | 按照ASCII值的大小比较字符串 | »> a>b  False |
| + | 字符串连接 | »> a+b  'Helloitcast, |
| \* | 重复输出字符串 | »> a\*2  'HelloHello' |
| [] | 使用索引获取字符串中的字符 | »> a[0]  H |
| [:] | 使用切片截取字符串中的一部分 | »> a[0:2]  'He' |
| in | 成员运算符。如果字符串中包含给定的字符，返回为Tme | »> 'o' in a  True |
| not in | 成员运算符。如果字符串中不包含给定的字符，返回为True | »> \*o' not in b  True |
| r/R | 原始字符串，所有的字符串都是直接按照字面意思使用 | »> printCr'Vn\*)  \n |
| % | 格式化字符串 | »> '%s,Lily!'%a  'Hello,Lily!' |

前面已经介绍了一些操作符，下面对其他操作符进行介绍。

4.9J靈魇曄骚纏篡籍噬驗拿嗇擧

用户使用账户密码登录时，比较字符串会派上用场。那么，根据什么标准来判断两个字符 串是否相等呢？我们知道，字符串是符号和数值的序列，通过遍历这个序列能取出每个字符9 从而可以按照ASCII值的大小进行比较。

常用的比较运算符适用于字符串，下面看一下如何使用这些比较运算符对字符串进行比较, 示例如下:

>>> word = nabc°

»> word2 = ° ABC"

>>> word3 = nabc"

>>> word<word2

False

>>> word <= word3

True

>>> word == word3

True

>>> word != word2

True

多学一招：身份运算待与==操作符的区别

Python支持对象本身之间的比较,比较格式如下:

obj1 is [not] obj 2

身份运算符是用来比较两个对象是否是同一个对象，上面比较运算符中的“==”是用来比 较两个对象的值是否相等。

Python中的变量有3个属性:name. id. valueo其中,name是变量的名字,id是内存地址, value是变量的值。is运算符则是通过这个id来判断的,如果id是一样的返回True,否则就返 回 False o

示例如下:

»> list\_a = [ !aJ , sb\* , 'c\* ]

>>> list\_b = [sa \ \ \*0\*]

>>> list\_a == 1ist\_b

True

>>> list\_a is list\_b

False

上述示例中，变量list\_a和变量list\_b保存的值是一样的，所以用==运算符比较的结果是 True。使用is操作符比较变量list a和变量list\_b的id,由于list a和list\_b引用的是两个不同 的列表对象,所以is运算符比较的id是不一样的,返回的结果为Falseo

Python对小的整数和字符串做了处理，不管使用==操作符还是is操作符进行比较，最终 的结果都是True。

示例如下:

>>> word = f abc1

>» word2 = ' abc \*

>>> word == word2

True

>>> word is word2

True

4S.2德魇毛意翼籥建遷寥简辭

使用+（连接符）可以把两个或者多个字符串连接成一个新的字符串。

示例如下：

>>> \* it? + \* heima'

'itheima 5

4L⑨忑 應陶成曇IS屢籍澄資寥簷罷

Python中的成员运算符（in和notin）,可以用来判断一个元素是否存在于某个序列中。例如, 检查某个字符或者子串是否在字符串中。语法格式如下：

obj [not] in sequence

in或者not in的返回值是True或者Falseo示例如下:

>>> ,it \* in \*itheima'

True

»> , cast1 not in \* itheima \*

True

>>> \* s? in 1itheima'

False