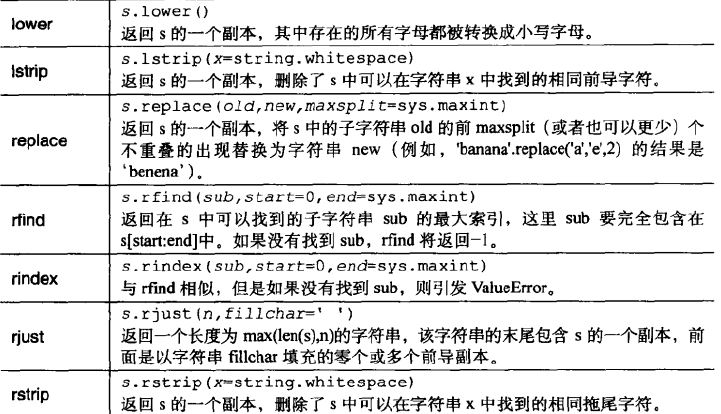
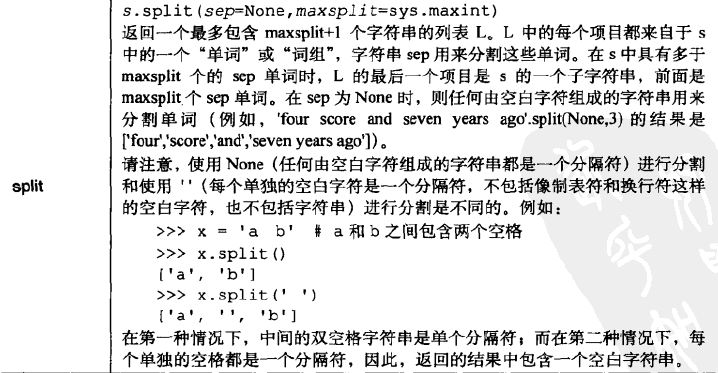
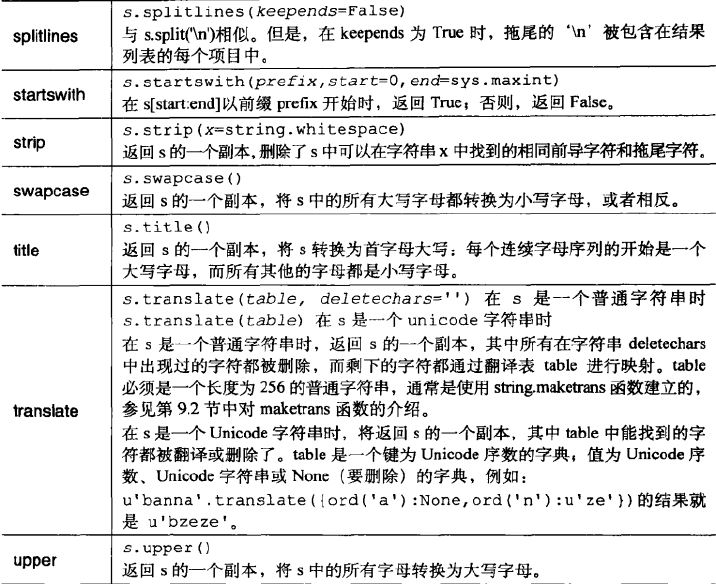
## 1字符串对象的方法

普通和Unicode字符串都是不可变序列。所有不可变序列的操作（重复、串联、索引和切片）都可以应用于字符串。

|  |  |
| --- | --- |
| capitalize | s.capitalize()返回s的一个副本，这个副本的第一个字符（也就是字母）是大写字母，而**其他**字符都是**小写**字母。 |
| center | s.center(n, fillchar= ‘ ’)返回一个长度为max(len(s), n)的字符串，如果n小于len(s),则返回该字符串S的副本，n大于len(s),返回该字符串的最中间包含s一个副本，前后包含的是相同数量的字符fillchar的副本。 |
| count | s.count(sub, start=0, end=sys.maxint)返回s[start:end）中子字符串sub不互相重叠的出现次数。 |
| decode | s.decode(codec=None, errors=’strict’)使用给定的编码器和错误处理方式返回从s获得的一个字符串(通常是Unicode字符串) |
| encode | S.encode(codec=None, errors=’strict’)使用给定的编码器和错误处理方式返回从s中获得的一个普通字符串。 |
| endswith | S.endswith(suffix,start=0, end = sys.maxint)在s[start:end)以suffix结束时，返回True；否则，返回False。 |
| expandtabs | S.expandtabs(tabsize=8)返回s的一个副本，其中的每个制表符被更改为一个或多个空格，制表位在每个tabsize个字符之后结束。 |
| find | S.find(sub, start=0,end=sys.maxint )返回在s中找到的子字符串sub的最小索引号，sub要完整的包含在s[start:end)中。如果没有找到sub，则find将返回-1 |
| index | S.index(sub, start=0,end=sys.maxint)与find类似，但是没有找到sub将引发ValueError。 |
| isalnum | S.isalnum()在len(s)大于0，并且s中的所有字符都是字母或十进制数字时，返回True。在S为空，或者s中至少有一个字符既不是字母也不是十进制数字时，isalnum将返回False。 |
| isalpha | S.isalpha()在len(s)大于0，并且s中的所有字符都是字母时，返回True。在s为空，或者在s中至少有一个字符不是字母时，isalpha将返回False |
| isdigit | S.isdigit()在len(s)大于0，并且s中的所有字符都是数字时，返回True。在s为空，或者在s中至少有一个字符不是字母时，isdigit将返回False |
| islower | S.islower在s中的所有字母都是小写时，返回True。在s不包含任何字母，或者s中至少有一个字母时大写时，islower将返回False |
| isspace | S.isspace()在len(s)大于0，并且s中所有的字符都是空白字符时，返回True。在s为空，或者s中至少有一个字符不是空白字符时，isspace将返回False。 |
| istitle | S.istitle()在s中的单词都是首字母大写时返回True，首字母大写就是每个连续字符序列的开始是一个大写字母，其他字母都是小写字母。在s不包含任何字母，或者s中至少有一个字母违反了首字母大写的条件时，istitle将返回False |
| isupper | S.isupper在s中的所有字母都是大写时，返回True。在s不包含任何字母，或者s中至少有一个字母小写时，isupper将返回False |
| partition | partition() 方法用来根据指定的分隔符将字符串进行分割。如果字符串包含指定的分隔符，则返回一个3元的元组，第一个为分隔符左边的子串，第二个为分隔符本身，第三个为分隔符右边的子串。 |
| rpartition | rpartition() 方法用来根据指定的分隔符将字符串进行分割。如果字符串包含指定的分隔符，则返回一个3元的元组，第一个为分隔符左边的子串，第二个为分隔符本身，第三个为分隔符右边的子串。 |
| join | S.join(seq)返回包含串联的seq中的项目的字符串，seq必须是一个序列或者其他项目都是字符串的可迭代对象，并且返回的字符串将在seq的每个项目对之间插入s的一个副本。 |
| ljust | S.ljust(n, fillchar=’’)返回一个长度为max(len(s), n)的字符串，这个字符串以s的一个副本开始，后面是以字符fillchar填充的零个或多个拖尾副本。 |







函数原型：

strip(...)

S.strip([chars]) -> string or unicode

Return a copy of the string S with leading and trailing # 返回去除头尾空格的字符串S的一个副本

whitespace removed.

If chars is given and not None, remove characters in chars instead. # 如果chars不是None，去掉头尾包含的chars字符。

If chars is unicode, S will be converted to unicode before stripping # 如果chars是unicode，字符串将被转换为unicode，在进行strip操作。

例：

print *'aabbssss'*.lstrip(*'ab'*) # ssss

print *'aabbssssaabb'*.rstrip(*'ab'*) # aabbssss

print *'aabbssabssaabb'*.strip(*'ab'*) # ssabss

# -\*- coding:utf-8 -\*-

S = *"tesT"*

print S.capitalize() # Test

print S.center(8, *'\_'*) # \_\_tesT\_\_

print *"Hkijjkijhvbhlo"*.count(*'ki'*, 2) # 1

print S.endswith(*'sT'*) # True

print *"S"*.expandtabs(1) # tesT

>>> s1 = '\t'

>>> len(s1)

1

>>> s2 = s1.expandtabs()

>>> len(s2)

8

>>> s2 '

print *"banana"*.find(*'na'*), *"banana"*.find(*'na'*, 1), *"banana"*.find(*'na'*, 3), *"banana"*.find(*'na'*, -2) #2,2,4,4

print *"banana"*.index(*'na'*), *"banana"*.index(*'na'*, 1), *"banana"*.index(*'na'*, 3), *"banana"*.index(*'na'*, -2) #2,2,4,4

print *"#isanum10"*.isalnum(), *"isanum10"*.isalnum() #False, True

print *'\_'*.join([*'a'*, *'b'*, *'cd'*]) #a\_b\_cd

print S.ljust(8, *'\_'*) #tesT\_\_\_\_

print S.rjust(8, *'\_'*) #\_\_\_\_tesT

print S.lower() #test

print S.upper() #TEST

print *"hello world!"*.title() #Hello World!

print *'a b'*.split() #['a', 'b']

print *' a b '*.strip() #a b 删除字符串两边的空格，中间空格删除不了

print *' a b '*.lstrip() #a b 删除字符串左边空格

print *' a b '*.rstrip() # a b 删除字符串右边空格

print *'abcd'*.partition(*'a'*), *'abcd'*.partition(*'a'*) # ('', 'a', 'bcd'), ('ab', 'c', 'd')

>>> s = 's\tp\na\rm'.split()

>>> s

['s', 'p', 'a', 'm']

split默认值相当于正则表达式的r‘\s’

**translate方法**

**描述**：Python translate() 方法根据参数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 del 参数中。

**语法**： S.translate(table [,deletechars]) -> string

**参数**：

* table -- 翻译表，翻译表是通过maketrans方法转换而来。
* deletechars -- 字符串中要过滤的字符列表。

例：

#!/usr/bin/env python

# -\*- coding:utf-8 -\*-

import string

instr = *'ABCDE'*

outstr = *'12345'*

table = string.maketrans(instr, outstr)

print *"ABCDE is abcde"*.translate(table) # 12345 is abcde

print *"ABCDE is abcde"*.translate(table, *'ABis'*) # 345 abcde

字符串逆序:步进-1表示分片将从右至左而不是从左至右。

>>> s = "abcde"

>>> s[::-1]

'edcba'

>>> s

'abcde'

>>> s[4:1:-1]

'edc'

>>>

## 2 unicode和字节字符串

### 2.1 Python3.0中的字符串修改

python2.x的str和unicode类型已经融入了python 3.0的str和bytes类型，并且增加了一种新的可变的类型bytearray。

Python2.X：

>>> str = "string"

>>> type(str)

<type 'str'>

>>> type(unicode(str,'utf-8'))

<type 'unicode'>

>>>

>>> sys.getdefaultencoding()

'ascii'

>>>

Python 3.x：

>>> str = "string"

>>> type(str)

<class 'str'>

>>> str = unicode(str, 'utf-8')

Traceback (most recent call last):

File "<pyshell#2>", line 1, in <module>

str = unicode(str, 'utf-8')

NameError: name 'unicode' is not defined

>>>

>>> sys.getdefaultencoding()

'utf-8'

>>>

### 2.2字符串基础知识

#### 2.2.1字符编码方法

* 编码是根据一个想要的编码名称，把一个字符串翻译为其原始字节的形式。
* 解码是根据其编码名称，把一个原始字节串翻译为字符串形式的过程。

ACSII是Latin-1和UTF-8的子集。

Python带有大约100种不同的编码， 参见Python库参考可以找到一个完整的列表。 导入e n c o d i n g s模块并运行help(encodings)也会显示很多的编码名称， 一些是在Python中实现的， 一些是在C中实现的。

#### 2.2.2 Python的字符串类型

python2.x：

* str表示8位文本和二进制数据
* unicode用来表示宽字符unicode文本

python3.x：

* str表示unicode文本（8位和更宽的）
* bytes表示二进制数据（每个Python 3.0 b y t e s对象其实只是较小整数的一个序列，其中每个整数的范围都在0到255之间；索引一个b y t e s将返回一个i n t，分片一个b y t e s将将返回另一个b y t e s，并且在一个b y t e s上运行内置函数l i s t将返回整数，而不是字符的一个列表）。
* bytearray是一种可变的bytes类型

python3.x之后关于这一修改的主要目标是把python2.x中常规的和unicode字符串类型合并到一个单独的字符串类型中去。

#### 2.2.3文本和二进制文件

文件I/O（输入和输出）在**Python 3.0**中也有所改进，以反映str/bytes的区分以及对编码

Unicode文本的自动支持。

**文本文件：**

当一个文件以文本模式打开的时候， 读取其数据会自动将其内容解码（每个平台

一个默认的或一个提供的编码名称），并且将其返回为一个s t r，写入会接受一个

s t r， 并且在将其传输到文件之间自动编码它。 文本模式的文件还支持统一的行尾

转换和额外的编码特定参数。 根据编码名称， 文本文件也自动处理文件开始处的字

节顺序标记序列。

**python3.x**

>>> s = "python 之禅"

>>> s

'python 之禅'

>>> type(s)

<class 'str'>

>>> fp = open(r'c:\e\dir0\python3.txt', 'w')

>>> fp.write(s)

9

>>> fp.close()

>>> s1 = open(r'c:\e\dir0\python3.txt').read()

>>> s1

'python 之禅'

>>> type(s1)

<class 'str'>

>>> len(s1)

9

**二进制文件：**

通过在内置的open调用的模式字符串参数添加一个b（只能小写） ， 以二进制模式

打开一个文件的时候， 读取其数据不会以任何方式解码它， 而是直接返回其内容

raw并且未经修改， 作为一个**bytes对象**； 写入类似地接受一个bytes对象， 并且将

其传送到文件中而未经修改。 二进制模式文件也接受一个bytearray对象作为写入

文件中的内容。

>>> s = b"python 之禅"

SyntaxError: bytes can only contain ASCII literal characters.

>>> s = 'python 之禅'

>>> b = s.encode('utf8')

>>> b

b'python \xe4\xb9\x8b\xe7\xa6\x85'

>>> fp = open(r'c:\e\dir0\python3.txt', 'w')

>>> fp.write(b)

Traceback (most recent call last):

File "<pyshell#15>", line 1, in <module>

fp.write(b)

TypeError: write() argument must be str, not bytes

>>> fp.close()

>>> fp = open(r'c:\e\dir0\python3.txt', 'wb')

>>> fp.write(b)

13

>>> fp.close()

>>> b1 = open(r'c:\e\dir0\python3.txt', 'rb').read()

>>> b1

b'python \xe4\xb9\x8b\xe7\xa6\x85'

>>> fp.close()

>>> b == b1

True

>>> len(b1)

13

>>> b1.decode('utf8')

'python 之禅'

#### 2.2.4 python3.0中的字符串应用

>>> B = b'spam'

>>> S = 'eggs'

>>> type(B), type(S)

(<class 'bytes'>, <class 'str'>)

>>> B[0],S[0]

(115, 'e')

>>> list(B),list(S)

([115, 112, 97, 109], ['e', 'g', 'g', 's'])

>>> B[1:],S[1:]

(b'pam', 'ggs')

>>>

>>> str(B, encoding='ascii') # 将字节码解码为str字符串

'spam'

>>> bytes(S, encoding='ascii') # 将字符串编码为字节码

b'spam'

**bytes对象是不可修改的**。

>>> B = b'spam'

>>> B.find(b'pa')

1

>>> B.replace(b'pa', b'XY')

b'sXYm'

>>> B.split(b'pa')

[b's', b'm']

>>> b'%s' % B

b'spam'

>>> b'%s' % 100

Traceback (most recent call last):

File "<pyshell#53>", line 1, in <module>

b'%s' % 100

**TypeError: %b requires bytes, or an object that implements \_\_bytes\_\_, not 'int'**

>>> B = bytes([97, 98, 99])

>>> B

b'abc'

>>> B = bytes('abc', 'utf8')

>>> B

b'abc'

>>> B = b'abc'

>>> B

b'abc'

>>> B = b'spam'

>>> B.replace('pa', 'XY')

Traceback (most recent call last):

File "<pyshell#68>", line 1, in <module>

B.replace('pa', 'XY')

TypeError: a bytes-like object is required, not 'str'

#### 2.2.5转换

python3.0基本上要求遵守一种类型或另一种类型，或者手动执行显式转换：

* str.encode()和bytes(S, encoding)把一个字符串转换为其raw bytes形式， 并且

在此过程中根据一个str创建一个bytes。

bytes.decode()和str(B, encoding)把raw bytes转换为其字符串形式， 并且在此

过程中根据一个bytes创建一个str。

#### 2.2.6 使用python3.x（和python2.6）bytearray对象

**python 2.x**

>>> S = 'spam'

>>> C = bytearray(S)

>>> C

bytearray(b'spam')

**python 3.x**

>>> S = 'spam'

>>> C = bytearray(S)

Traceback (most recent call last):

File "<pyshell#73>", line 1, in <module>

C = bytearray(S)

TypeError: string argument without an encoding

>>> C = bytearray(S, 'utf8')

>>> C

bytearray(b'spam')

或者

>>> B = b'spam'

>>> C = bytearray(B)

>>> C

bytearray(b'spam')

>>> C[0]

115

>>> C[0] = 'x'

Traceback (most recent call last):

File "<pyshell#77>", line 1, in <module>

C[0] = 'x'

TypeError: an integer is required

>>> C[0] = b'x'

Traceback (most recent call last):

File "<pyshell#78>", line 1, in <module>

C[0] = b'x'

TypeError: an integer is required

>>> C[0] = ord('x')

>>> C

bytearray(b'xpam')

>>> C[1] = b'Y'[0]

>>> C

bytearray(b'xYam')

>>> C.append(b'LMN')

Traceback (most recent call last):

File "<pyshell#86>", line 1, in <module>

C.append(b'LMN')

TypeError: an integer is required

>>> C.append(b'L')

Traceback (most recent call last):

File "<pyshell#87>", line 1, in <module>

C.append(b'L')

TypeError: an integer is required

>>> C.append(ord('L'))

>>> C

bytearray(b'xYamL')

>>> C.extend(b'MNO')

>>> C

bytearray(b'xYamLMNO')

>>> C + b'!#'

bytearray(b'xYamLMNO!#')

>>> C[1:]

bytearray(b'YamLMNO')

>>> len(C)

8

>>> C.replace(b'Y', b'sp')

bytearray(b'xspamLMNO')

>>> C \* 4

bytearray(b'xYamLMNOxYamLMNOxYamLMNOxYamLMNO')

>>> list(C)

[120, 89, 97, 109, 76, 77, 78, 79]