第7章 字符串

第7章 字符串

- 在Python中，字符串属于不可变有序序列，使用单引号、双引号、三单引号或三双引号作为定界符，并且不同的定界符之间可以互相嵌套。

```
'abc'、'123'、'中国'

"Python"

'''Tom said,"Let's go"'''
```

第7章 字符串

- 除了支持序列通用方法（包括双向索引、比较大小、计算长度、元素访问、切片、成员测试等操作）以外，字符串类型还支持一些特有的操作方法，例如字符串格式化、查找、替换、排版等等。
- 字符串属于不可变序列，不能直接对字符串对象进行元素增加、修改与删除等操作，切片操作也只能访问其中的元素而无法使用切片来修改字符串中的字符。

7.2 转义字符与原始字符串

转义字符	含义	转义字符	含义
\b	退格，把光标移动到前一列位置	\\	一个斜线\
\f	换页符	\’	单引号’
\n	换行符	\”	双引号”
\r	回车	\ooo	3位八进制数对应的字符
\t	水平制表符	\xhh	2位十六进制数对应的字符
\v	垂直制表符	\uhhhh	4位十六进制数表示的Unicode字符

7.2 转义字符与原始字符串

❖转义字符用法

```
>>> print('Hello\nWorld')                #包含转义字符的字符串
Hello
World
```

7.2 转义字符与原始字符串

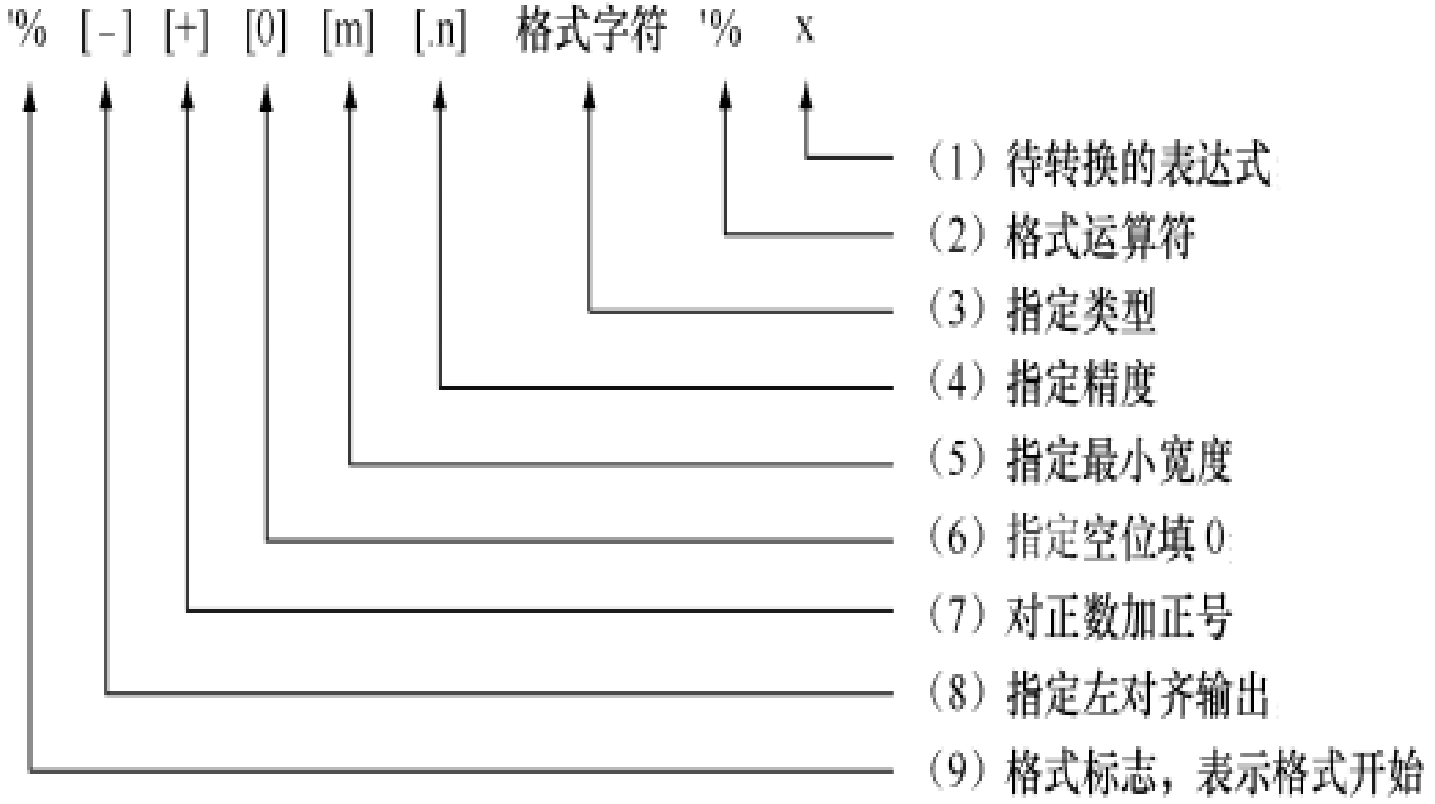
- 为了避免对字符串中的转义字符进行转义，可以使用原始字符串，在字符串前面加上字母r或R表示原始字符串，其中的所有字符都表示原始的含义而不会进行任何转义。

```
>>> path = 'C:\Windows\notepad.exe'
>>> print(path)                          #字符\n被转义为换行符
C:\Windows
otepad.exe
>>> path = r'C:\Windows\notepad.exe'     #原始字符串，任何字符都不转义
>>> print(path)
C:\Windows\notepad.exe
```

7.3 字符串格式化

- 7.3.1 使用%运算符进行格式化
- 7.3.2 使用format方法进行格式化
- 7.3.3 格式化的字符串常量

7.3.1 使用%运算符进行格式化



7.3.1 使用%运算符进行格式化

- 常用格式字符

格式字符	说明
%s	字符串（采用str()的显示）
%r	字符串（采用repr()的显示）
%c	单个字符
%d	十进制整数
%i	十进制整数
%o	八进制整数
%x	十六进制整数
%e	指数（基底写为e）
%E	指数（基底写为E）
%f、%F	浮点数
%g	指数(e)或浮点数（根据显示长度）
%G	指数(E)或浮点数（根据显示长度）
%%	一个字符"%"

7.3.1 使用%运算符进行格式化

```
>>> x = 1235
>>> so = "%o" % x
>>> so
"2323"
>>> sh = "%x" % x
>>> sh
"4d3"
>>> se = "%e" % x
>>> se
"1.235000e+03"
>>> chr(ord("3")+1)
"4"
>>> "%s" % 65
"65"
>>> "%s" % 65333
"65333"
>>> "%d" % "555"
TypeError: %d format: a number is required, not str
```

7.3.2 使用format()方法进行格式化

- 使用format方法进行格式化更加灵活，不仅可以使用位置进行格式化，还支持使用与位置无关的参数名字进行格式化
- 字符串format()方法的基本使用格式如下：
  <模板字符串>.format(<逗号分隔的参数>)
- 模板字符串由一系列槽组成，用来控制修改字符串中嵌入值出现的位置，其基本思想是将format()方法中逗号分隔的参数按照序号关系替换到模板字符串的槽中。槽用大括号（{}）表示，如果大括号中没有序号，则按照出现顺序替换。

例如

```
>>> "{}：计算机{}的占用率为{}%。".format("2016-12-31","PYTHON",10)
'2016-12-31：计算机PYTHON的占用率为10%。'
```

7.3.2 使用format()方法进行格式化

- format()方法中模板字符串的槽除了包括参数序号，还可以包括格式控制信息，此时，槽的内部样式如下：

  {<参数序号>:<格式控制标记>}

- 其中，格式控制标记用来控制参数显示时的格式，格式内容如图所示

:	<填充>	<对齐>	<宽度>	<,>	<精度>	<类型>
引导	用于填充的	<左对齐	槽的设定输	数字的千位	浮点数小数	整数类型
符号	单个字符	>右对齐	出宽度	分隔符	部分的精度	b,c,d,o,x,X
		^居中对齐		适用于整数	或	浮点数类型
				和浮点数	字符串的最	e,E,f,%
					大输出长度	

7.3.2 使用format()方法进行格式化

- 在<类型>中
  - b：输出整数的二进制方式
  - c：输出整数对应的Unicode字符
  - d：输出整数的十进制方式
  - o：输出整数的八进制方式
  - x：输出整数的小写十六进制方式
  - X：输出整数的大写十六进制方式

  - e：输出浮点数对应的小写字母e的指数形式
  - E：输出浮点数对应的小写字母E的指数形式
  - f：输出浮点数的标准浮点形式
  - %：输出浮点数的百分形式

7.3.2 使用format()方法进行格式化

```
>>> 1/3
0.3333333333333333
>>> print('{0:.3f}'.format(1/3))          #保留3位小数
0.333
>>> '{0:%}'.format(3.5)                   #格式化为百分数
 '350.000000%'
```

7.3.2 使用format()方法进行格式化

```
>>> print("The number {0:,} in hex is: {0:#x}, the number {1} in oct is
  {1:#o}".format(5555,55))
The number 5,555 in hex is: 0x15b3, the number 55 in oct is 0o67
>>> print("The number {1:,} in hex is: {1:#x}, the number {0} in oct is
  {0:o}".format(5555,55))
The number 55 in hex is: 0x37, the number 5555 in oct is 12663
>>> print("my name is {name}, my age is {age}, and my QQ is {qq}".format(name
  = "Dong Fuguo",age = 40,qq = "30646****"))
my name is Dong Fuguo, my age is 40, and my QQ is 30646****
>>> position = (5, 8, 13)
>>> print("X:{0[0]};Y:{0[1]};Z:{0[2]}".format(position))
X:5;Y:8;Z:13
```

7.4 字符串常用操作

- Python字符串对象提供了大量方法用于字符串的切分、连接、替换和排版等操作，另外还有大量内置函数和运算符也支持对字符串的操作。
- 字符串对象是不可变的，所以字符串对象提供的涉及到字符串“修改”的方法都是返回修改后的新字符串，并不对原始字符串做任何修改，无一例外。

7.4.1 find()、rfind()、index()、rindex()、count()

- find()、rfind()、index()、rindex()、count()

✓ find()和rfind方法分别用来查找一个字符串在另一个字符串指定范围（默认是整个字符串）中首次和最后一次出现的位置，如果不存在则返回-1；

✓ index()和rindex()方法用来返回一个字符串在另一个字符串指定范围中首次和最后一次出现的位置，如果不存在则抛出异常；

✓ count()方法用来返回一个字符串在当前字符串中出现的次数。

7.4.1 find()、rfind()、index()、rindex()、count()

```
>>> s="apple,peach,banana,peach,pear"
>>> s.find("peach")
6
>>> s.find("peach",7)
19
>>> s.find("peach",7,20)
-1
>>> s.rfind('p')
25
>>> s.index('p')
1
>>> s.index('pe')
6
```

```
>>> s.index('pear')
25
>>> s.index('ppp')
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in
<module>
    s.index('ppp')
ValueError: substring not found
>>> s.count('p')
5
>>> s.count('pp')
1
>>> s.count('ppp')
0
```

7.4.2 split()、rsplit()、partition()、rpartition()

- split()、rsplit()、partition()、rpartition()

✓ split()和rsplit()方法分别用来以指定字符为分隔符，把当前字符串从左往右或从右往左分隔成多个字符串，并返回包含分隔结果的列表；

✓ partition()和rpartition()用来以指定字符串为分隔符将原字符串分隔为3部分，即分隔符前的字符串、分隔符字符串、分隔符后的字符串，如果指定的分隔符不在原字符串中，则返回原字符串和两个空字符串。

7.4.2 split()、rsplit()、partition()、rpartition()

```
>>> s = "apple,peach,banana,pear"
>>> s.split(",")
["apple", "peach", "banana", "pear"]
>>> s.partition(',')
('apple', ',', 'peach,banana,pear')
>>> s.rpartition(',')
('apple,peach,banana', ',', 'pear')
>>> s.rpartition('banana')
('apple,peach,', 'banana', ',pear')
>>> s = "2017-10-31"
>>> t = s.split("-")
>>> print(t)
['2017', '10', '31']
>>> print(list(map(int, t)))
[2017, 10, 31]
```

7.4.2 split()、rsplit()、partition()、rpartition()

- split()和rsplit()方法还允许指定最大分割次数。

```
>>> s = '\n\nhello\t\t world \n\n\n My name is Dong   '
>>> s.split(None, 1)
['hello', 'world \n\n\n My name is Dong   ']
>>> s.rsplit(None, 2)
['\n\nhello\t\t world \n\n\n My name', 'is', 'Dong']
>>> s.split(maxsplit=6)
['hello', 'world', 'My', 'name', 'is', 'Dong']
>>> s.split(maxsplit=100)      #最大分隔次数大于可分隔次数时无效
['hello', 'world', 'My', 'name', 'is', 'Dong']
```

7.4.2 split()、rsplit()、partition()、rpartition()

- 对于split()和rsplit()方法，如果不指定分隔符，则字符串中的任何空白符号（空格、换行符、制表符等）都将被认为是分隔符，把连续多个空白字符看作一个分隔符。

```
>>> s = 'hello world \n\n My name is Dong    '
>>> s.split()
['hello', 'world', 'My', 'name', 'is', 'Dong']
>>> s = '\n\nhello world \n\n\n My name is Dong    '
>>> s.split()
['hello', 'world', 'My', 'name', 'is', 'Dong']
>>> s = '\n\nhello\t\t world \n\n\n My name\t is Dong    '
>>> s.split()
['hello', 'world', 'My', 'name', 'is', 'Dong']
```

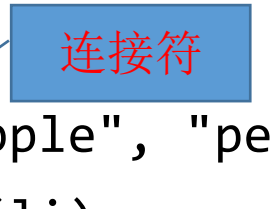
7.4.2 split()、rsplit()、partition()、rpartition()

◆然而，明确传递参数指定split()使用的分隔符时，情况是不一样的。

```
>>> 'a,,,bb,,ccc'.split(',')          #每个逗号都被作为独立的分隔符
['a', '', '', 'bb', '', 'ccc']
>>> 'a\t\t\tbb\t\tccc'.split('\t') #每个制表符都被作为独立的分隔符
['a', '', '', 'bb', '', 'ccc']
>>> 'a\t\t\tbb\t\tccc'.split()        #连续多个制表符被作为一个分隔符
['a', 'bb', 'ccc']
```

7.4.3 join()

- 字符串连接join()




```
>>> li = ["apple", "peach", "banana", "pear"]
>>> ','.join(li)
'apple,peach,banana,pear'
>>> '.'.join(li)
'apple.peach.banana.pear'
>>> '::'.join(li)
'apple::peach::banana::pear'
```

7.4.3 join()

- **问题解决：**使用split()和join()方法删除字符串中多余的空白字符，连续多个空白字符只保留一个。

```
>>> x = 'aaa      bb      c d e   fff    '
>>> ' '.join(x.split())                    #使用空格作为连接符
'aaa bb c d e fff'
```

7.4.4 lower()、upper()、capitalize()、title()、swapcase()

- lower()、upper()、capitalize()、title()、swapcase()

```
>>> s = "What is Your Name?"
>>> s.lower()                          #返回小写字符串
'what is your name?'
>>> s.upper()                          #返回大写字符串
'WHAT IS YOUR NAME?'
>>> s.capitalize()                     #字符串首字符大写
'What is your name?'
>>> s.title()                          #每个单词的首字母大写
'What Is Your Name?'
>>> s.swapcase()                       #大小写互换
'wHAT IS yOUR nAME?'
```

7.4.5 replace()

- 查找替换replace()，类似于Word中的“全部替换”功能。

```
>>> s = "中国，中国"
>>> print(s)
中国，中国
>>> s2 = s.replace("中国", "中华人民共和国")  #两个参数都作为一个整理
>>> print(s2)
中华人民共和国，中华人民共和国
```

7.4.5 replace()

- 问题解决：测试用户输入中是否有敏感词，如果有的话就把敏感词替换为3个星号***。

```
>>> words = ('测试', '非法', '暴力', '话')
>>> text = '这句话里含有非法内容'
>>> for word in words:
    if word in text:
        text = text.replace(word, '***')
>>> text
'这句***里含有***内容'
```

7.4.6 strip()、rstrip()、lstrip()

- strip()、rstrip()、lstrip()

```
>>> s = "  abc   "
>>> s.strip()                                  #删除空白字符
'abc'
>>> '\n\nhello world   \n\n'.strip()           #删除空白字符
'hello world'
>>> "aaaassddf".strip("a")                     #删除指定字符
'ssddf'
>>> "aaaassddf".strip("af")
'ssdd'
>>> "aaaassddfaaa".rstrip("a")                 #删除字符串右端指定字符
'aaaassddf'
>>> "aaaassddfaaa".lstrip("a")                 #删除字符串左端指定字符
'ssddfaaa'
```

7.4.6 strip()、rstrip()、lstrip()

- 这三个函数的参数指定的字符串并不作为一个整体对待，而是在原字符串的两侧、右侧、左侧删除参数字符串中包含的所有字符，一层一层地从外往里扒。

```
>>> 'aabbccddeeeffg'.strip('af')  #字母f不在字符串两侧，所以不删除
'bbccddeeeffg'
>>> 'aabbccddeeeffg'.strip('gaf')
'bbccddeee'
>>> 'aabbccddeeeffg'.strip('gaef')
'bbccdd'
>>> 'aabbccddeeeffg'.strip('gbaef')
'ccdd'
>>> 'aabbccddeeeffg'.strip('gbaefcd')
''
```

7.4.7 startswith()、endswith()

- s.startswith(t)、s.endswith(t)，判断字符串是否以指定字符串开始或结束

```
>>> s = 'Beautiful is better than ugly.'
>>> s.startswith('Be')          #检测整个字符串
True
>>> s.startswith('Be', 5)       #指定检测范围起始位置
False
>>> s.startswith('Be', 0, 5)    #指定检测范围起始和结束位置
True
```

7.4.8 isalnum()、isalpha()、isdigit()、isspace()、isupper()、islower()

- isalnum()、isalpha()、isdigit()、isdecimal()、isnumeric()、isspace()、isupper()、islower()，用来测试字符串是否为数字或字母、是否为字母、是否为数字字符、是否为空白字符、是否为大写字母以及是否为小写字母。

```
>>> '1234abcd'.isalnum()
True
>>> '1234abcd'.isalpha()          #全部为英文字母时返回True
False
>>> '1234abcd'.isdigit()          #全部为数字时返回True
False
>>> 'abcd'.isalpha()
True
>>> '1234.0'.isdigit()
False
```

7.4.8 isalnum()、isalpha()、isdigit()、isspace()、isupper()、islower()

```
>>> '1234'.isdigit()
True
>>> '九'.isdigit()
False
>>> 'ⅣⅢX'.isdigit()
False
```

7.4.9 center()、ljust()、rjust()

- center()、ljust()、rjust()，返回指定宽度的新字符串，原字符串居中、左对齐或右对齐出现在新字符串中，如果指定宽度大于字符串长度，则使用指定的字符（默认为空格）进行填充。

```
>>> 'Hello world!'.center(20)          #居中对齐，以空格进行填充
'    Hello world!    '
>>> 'Hello world!'.center(20, '=')     #居中对齐，以字符=进行填充
'====Hello world!===='
>>> 'Hello world!'.ljust(20, '=')      #左对齐
'Hello world!========'
>>> 'Hello world!'.rjust(20, '=')      #右对齐
'========Hello world!'
```

7.4.10 字符串对象支持的运算符

- Python**字符串支持**加法运算符，表示两个字符串连接，生成新字符串。

```
>>> 'hello ' + 'world'
'hello world'
```

7.4.10 字符串对象支持的运算符

- 成员判断，关键字in

```
>>> "a" in "abcde"      #测试一个字符中是否存在于另一个字符串中
True
>>> 'ab' in 'abcde'
True
>>> 'ac' in 'abcde'     #关键字in左边的字符串作为一个整体对待
False
>>> "j" in "abcde"
False
```

7.4.10 字符串对象支持的运算符

- Python字符串支持与整数的乘法运算，表示序列重复，也就是字符串内容的重复，得到新字符串。

```
>>> 'abcd' * 3
'abcdabcdabcd'
```

7.4.11 适用于字符串对象的内置函数

```
>>> x = 'Hello world.'
>>> len(x)                          #字符串长度
12
>>> max(x)                          #最大字符
'w'
>>> min(x)
' '
>>> list(zip(x,x))                  #zip()也可以作用于字符串
[('H', 'H'), ('e', 'e'), ('l', 'l'), ('l', 'l'), ('o', 'o'), (' ', ' '), ('w',
'w'), ('o', 'o'), ('r', 'r'), ('l', 'l'), ('d', 'd'), ('.', '.')]
>>> sorted(x)
[' ', '.', 'H', 'd', 'e', 'l', 'l', 'l', 'o', 'o', 'r', 'w']
>>> list(reversed(x))
['.', 'd', 'l', 'r', 'o', 'w', ' ', 'o', 'l', 'l', 'e', 'H']
>>> list(enumerate(x))
[(0, 'H'), (1, 'e'), (2, 'l'), (3, 'l'), (4, 'o'), (5, ' '), (6, 'w'), (7,
'o'), (8, 'r'), (9, 'l'), (10, 'd'), (11, '.')]
>>> list(map(add, x, x))
['HH', 'ee', 'll', 'll', 'oo', '  ', 'ww', 'oo', 'rr', 'll', 'dd', '..']
```

7.4.11 适用于字符串对象的内置函数

- 内置函数eval()用来把任意字符串转化为Python表达式并进行求值。

```
>>> eval("3+4")                          #计算表达式的值
7
>>> a = 3
>>> b = 5
>>> eval('a+b')                          #这时候要求变量a和b已存在
8
>>> import math
>>> eval('math.sqrt(3)')
1.7320508075688772
```

7.4.11 适用于字符串对象的内置函数

- Python的内置函数eval()可以计算任意合法表达式的值，如果有恶意用户巧妙地构造并输入非法字符串，可以执行任意外部程序或者实现其他目的，例如下面的代码运行后可以启动记事本程序：

```
>>> a = input('Please input a value:')
Please input a
value:__import__('os').startfile(r'C:\Windows\\notepad.exe')
>>> eval(a)
```

- 下面的代码则会导致屏幕一闪，而就在那一瞬间在当前文件夹中创建了一个子文件夹testtest：

```
>>> eval("__import__('os').system('md testtest')")
```

7.4.12 字符串对象的切片操作

- 切片也适用于字符串，但仅限于读取其中的元素，不支持字符串修改。

```
>>> 'Explicit is better than implicit.'[:8]
'Explicit'
>>> 'Explicit is better than implicit.'[9:23]
'is better than'
>>> path = 'C:\\Python35\\test.bmp'
>>> path[:-4] + '_new' + path[-4:]
'C:\\Python35\\test_new.bmp'
```

7.5 字符串常量

- Python标准库string中定义数字字符、标点符号、英文字母、大写字母、小写字母等常量。

```
>>> import string
>>> string.digits
'0123456789'
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
>>> string.ascii_letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> string.ascii_lowercase
'abcdefghijklmnopqrstuvwxyz'
>>> string.ascii_uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

7.5 字符串常量

- **问题解决：**生成指定长度的随机密码。

```
>>> import string
>>> characters = string.digits + string.ascii_letters
>>> import random
>>> ''.join([random.choice(characters) for i in range(8)])
'J5Cuofhy'
>>> ''.join([random.choice(characters) for i in range(10)])
'RkHA3K3tNl'
>>> ''.join([random.choice(characters) for i in range(16)])
'zSabpGltJ0X4CCjh'
```