桂林理工大学
GUILIN UNIVERSITY OF TECHNOLOGY

第七章
字符串

测绘地理信息学院

第七章 字符串

- 7.1字符串编码格式简介
- 7.2转义字符与原始字符串
- 7.3字符串格式化
- 7.4字符串常用操作
- 7.5字符串常量

第七章 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"'''
```

第七章 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。

第七章 字符串

- 除了支持Unicode编码的str类型之外，Python还支持字节串类型bytes，str类型字符串可以通过encode()方法使用指定的字符串编码格式编码成为bytes对象，而bytes对象则可以通过decode()方法使用指定编码格式解码成为str字符串。

7.1 字符串编码格式简介

- 最早的字符串编码是美国标准信息交换码ASCII，仅对10个数字、26个大写英文字母、26个小写英文字母及一些其他符号进行了编码。ASCII码采用1个字节来对字符进行编码，最多只能表示256个符号。

7.1 字符串编码格式简介

- GB2312是我国制定的中文编码，使用1个字节表示英语，2个字节表示中文；GBK是GB2312的扩充，而CP936是微软在GBK基础上开发的编码方式。GB2312、GBK和CP936都是使用2个字节表示中文。
- UTF-8对全世界所有国家需要用到的字符进行了编码，以1个字节表示英语字符(兼容ASCII)，以3个字节表示中文，还有些语言的符号使用2个字节（例如俄语和希腊语符号）或4个字节。

7.1 字符串编码格式简介

- 不同编码格式之间相差很大，采用不同的编码格式意味着不同的表示和存储形式，把同一字符存入文件时，写入的内容可能会不同，在试图理解其内容时必须了解编码规则并进行正确的解码。如果解码方法不正确就无法还原信息，从这个角度来讲，字符串编码也具有加密的效果。

7.1 字符串编码格式简介

```
>>> '董付国'.encode('utf8')
b'\xe8\x91\xa3\xe4\xbb\x98\xe5\x9b\xbd'
>>> '董付国'.encode('cp936')
b'\xb6\xad\xb8\xb6\xb9\xfa'
>>> '董付国'.encode('cp936').decode('cp936')
'董付国'
>>> 'Python可以这样学'.encode('utf8').decode('cp936')
Traceback (most recent call last):
  File "<pyshell#63>", line 1, in <module>
    'Python可以这样学'.encode('utf8').decode('cp936')
UnicodeDecodeError: 'gbk' codec can't decode byte 0xaf in position 8:
illegal multibyte sequence
>>> 'Python程序设计开发宝典'.encode('cp936').decode('utf8')
Traceback (most recent call last):
  File "<pyshell#64>", line 1, in <module>
    'Python程序设计开发宝典'.encode('cp936').decode('utf8')
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb3 in position
6: invalid start byte
```

7.1 字符串编码格式简介

- Python 3.x完全支持中文字符，默认使用UTF8编码格式，无论是一个数字、英文字母，还是一个汉字，在统计字符串长度时都按一个字符对待和处理。

```
>>> s = '中国山东烟台'
>>> len(s)                          #字符串长度，或者包含的字符个数
6
>>> s = '中国山东烟台ABCDE'        #中文与英文字符同样对待，都算一个字符
>>> len(s)
11
>>> 姓名 = '张三'                   #使用中文作为变量名
>>> print(姓名)                     #输出变量的值
张三
```

7.2 转移字符与原始字符串

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 转移字符与原始字符串

❖ 转义字符用法

```
>>> print('Hello\nWorld')                    #包含转义字符的字符串
Hello
World
>>> print('\101')                            #三位八进制数对应的字符
A
>>> print('\x41')                            #两位十六进制数对应的字符
A
>>> print('我是\u8463\u4ed8\u56fd')#四位十六进制数表示Unicode字符
我是董付国
```

7.2 转移字符与原始字符串

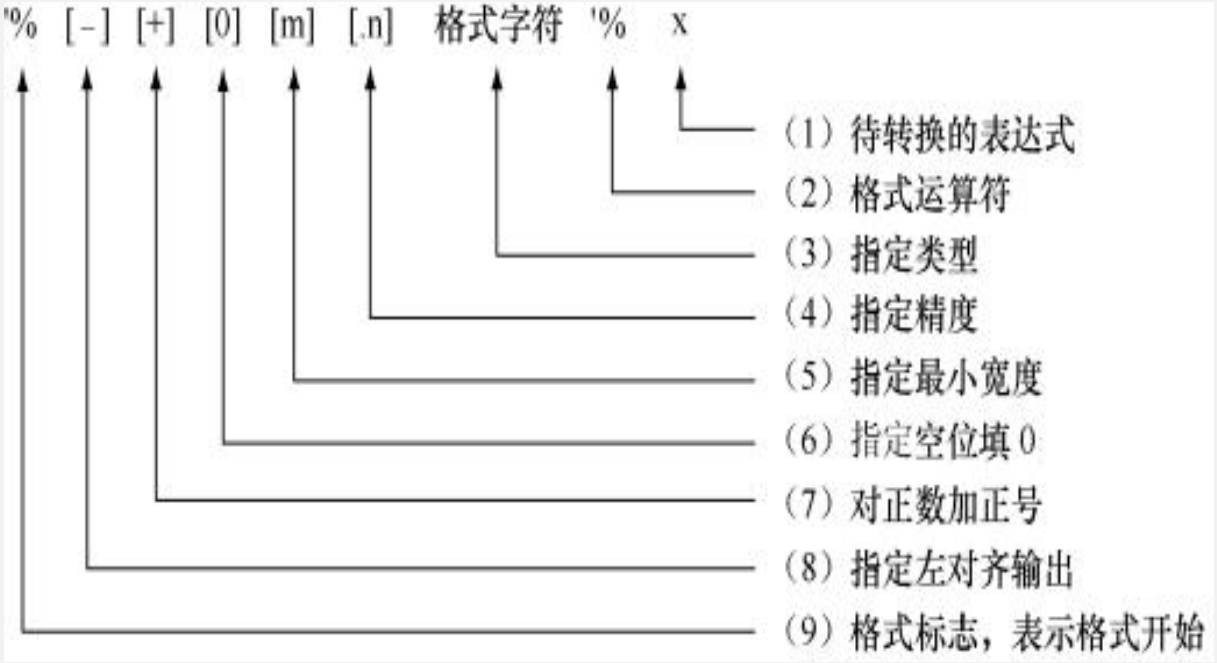
- 为了避免对字符串中的转义字符进行转义，可以使用原始字符串，在字符串前面加上字母r或R表示原始字符串，其中的所有字符都表示原始的含义而不会进行任何转义。

```
>>> 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

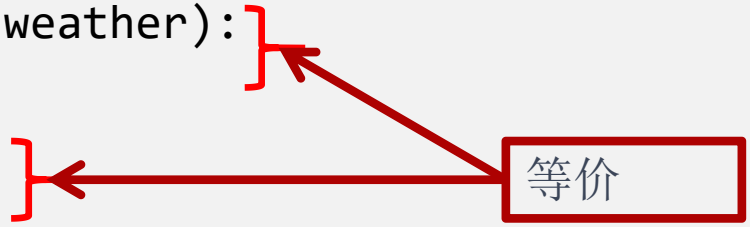
```
>>> 1/3
0.3333333333333333
>>> print('{0:.3f}'.format(1/3))          #保留3位小数
0.333
>>> '{0:%}'.format(3.5)                   #格式化为百分数
'350.000000%'
>>> '{0:_},{0:_x}'.format(1000000)        #Python 3.6.0及更高版本支持
'1_000_000,f_4240'
>>> '{0:_},{0:_x}'.format(10000000)       #Python 3.6.0及更高版本支持
'10_000_000,98_9680'
```

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
```

7.3.2 使用format()方法进行格式化

```
weather = [("Monday","rainy"),("Tuesday","sunny"),
           ("Wednesday", "sunny"),("Thursday","rainy"),
           ("Friday","cloudy")]
formatter = "Weather of '{0[0]}' is '{0[1]}'".format
for item in map(formatter,weather):
    print(item)
for item in weather:
    print(formatter(item))
```



运行结果:

```
Weather of 'Monday' is 'rainy'
Weather of 'Tuesday' is 'sunny'
Weather of 'Wednesday' is 'sunny'
Weather of 'Thursday' is 'rainy'
Weather of 'Friday' is 'cloudy'
```

7.3.3 格式化的字符串常量

- 从Python 3.6.x开始支持一种新的字符串格式化方式，官方叫做Formatted String Literals，在字符串前加字母f，含义与字符串对象format()方法类似。

```
>>> name = 'Dong'
>>> age = 39
>>> f'My name is {name}, and I am {age} years old.'
'My name is Dong, and I am 39 years old.'
>>> width = 10
>>> precision = 4
>>> value = 11/3
>>> f'result:{value:{width}.{precision}}'
'result:     3.667'
```

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()

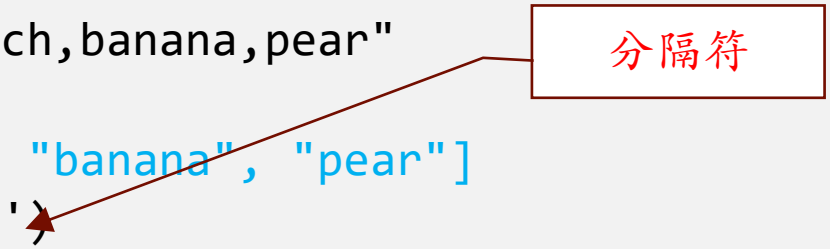
- split()、rsplit()、partition()、rpartition()

✓ 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'
>>> def equavilent(s1, s2):                #判断两个字符串在Python意义上是否等价
    if s1 == s2:
        return True
    elif ' '.join(s1.split()) == ' '.join(s2.split()):
        return True
    elif ''.join(s1.split()) == ''.join(s2.split()):
        return True
    else:
        return False
>>> equavilent('pip list', 'pip    list')
True
```

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()、maketrans()、translate()

- 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()、maketrans()、translate()

- **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'
```

7.4.5 replace()、maketrans()、translate()

- 字符串对象的maketrans()方法用来生成字符映射表，而translate()方法用来根据映射表中定义的对应关系转换字符串并替换其中的字符，使用这两个方法的组合可以同时处理多个字符。

这两个参数不是作为整体进行处理的

```
#创建映射表，将字符"abcdef123"一一对应地转换为"uvwxyz@#$"
>>> table = ''.maketrans('abcdef123', 'uvwxyz@#$')
>>> s = "Python is a greate programming language. I like it!"
#按映射表进行替换
>>> s.translate(table)
'Python is u gryuty progrumming lunguugy. I liky it!'
```

7.4.5 replace()、maketrans()、translate()

- **问题解决**：凯撒加密，每个字母替换为后面第k个。

```
>>> import string
>>> def kaisa(s, k):
    lower = string.ascii_lowercase              #小写字母
    upper = string.ascii_uppercase              #大写字母
    before = string.ascii_letters
    after = lower[k:] + lower[:k] + upper[k:] + upper[:k]
    table = ''.maketrans(before, after)         #创建映射表
    return s.translate(table)

>>> s = "Python is a greate programming language. I like it!"
>>> kaisa(s, 3)
'Sbwkrq lv d juhdwh surjudpplqj odqjxdjh. L olnh lw!'
```

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
>>> import os
>>> [filename for filename in os.listdir(r'c:\\') if filename.endswith(('.bmp','.jpg','.gif'))]
```

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

```
>>> '1234'.isdigit()
True
>>> '九'.isnumeric()                #isnumeric()方法支持汉字数字
True
>>> '九'.isdigit()
False
>>> '九'.isdecimal()
False
>>> 'ⅣⅢX'.isdecimal()
False
>>> 'ⅣⅢX'.isdigit()
False
>>> 'ⅣⅢX'.isnumeric()               #支持罗马数字
True
```

7.4.9 center()、ljust()、rjust()、zfill()

- 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。zfill()返回指定宽度的字符串，在左侧以字符0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'
```

7.6 中英文分词

```
>>> import jieba                              #导入jieba模块
>>> x = '分词的准确度直接影响了后续文本处理和挖掘算法的最终效果。'
>>> jieba.cut(x)                              #使用默认词库进行分词
<generator object Tokenizer.cut at 0x000000000342C990>
>>> list(_)
['分词', '的', '准确度', '直接', '影响', '了', '后续', '文本处理', '和',
'挖掘', '算法', '的', '最终', '效果', '。']
>>> list(jieba.cut('纸杯'))
['纸杯']
>>> list(jieba.cut('花纸杯'))
['花', '纸杯']
>>> jieba.add_word('花纸杯')                  #增加词条
>>> list(jieba.cut('花纸杯'))                 #使用新题库进行分词
['花纸杯']
>>> import snownlp                            #导入snownlp模块
>>> snownlp.SnowNLP('学而时习之，不亦说乎').words
['学而', '时习', '之', '，', '不亦', '说乎']
>>> snownlp.SnowNLP(x).words
['分词', '的', '准确度', '直接', '影响', '了', '后续', '文本', '处理', '
和', '挖掘', '算法', '的', '最终', '效果', '。']
```

7.7 汉字到拼音的转换

```
>>> from pypinyin import lazy_pinyin, pinyin
>>> lazy_pinyin('董付国')                    #返回拼音
['dong', 'fu', 'guo']
>>> lazy_pinyin('董付国', 1)                 #带声调的拼音
['dǒng', 'fù', 'guó']
>>> lazy_pinyin('董付国', 2)                 #另一种拼音形式，数字表示前面
字母的声调
['do3ng', 'fu4', 'guo2']
>>> lazy_pinyin('董付国', 3)                 #只返回拼音首字母
['d', 'f', 'g']
>>> lazy_pinyin('重要', 1)                   #能够根据词组智能识别多音字
['zhòng', 'yào']
>>> lazy_pinyin('重阳', 1)
['chóng', 'yáng']
>>> pinyin('重阳')                           #返回拼音
[['chóng'], ['yáng']]
>>> pinyin('重阳节', heteronym=True) #返回多音字的所有读音
[['zhòng', 'chóng', 'tóng'], ['yáng'], ['jié', 'jiē']]
```

7.7 汉字到拼音的转换

```
>>> import jieba                                    #其实不需要导入jieba,
这里只是说明已安装
>>> x = '中英文混合test123'
>>> lazy_pinyin(x)                                  #自动调用已安装的
jieba扩展库分词功能
['zhong', 'ying', 'wen', 'hun', 'he', 'test123']
>>> lazy_pinyin(jieba.cut(x))
['zhong', 'ying', 'wen', 'hun', 'he', 'test123']
>>> x = '山东烟台的大樱桃真好吃啊'
>>> sorted(x, key=lambda ch: lazy_pinyin(ch))
                                                    #按拼音对汉字进行排
序
['啊', '吃', '大', '的', '东', '好', '山', '台', '桃', '
烟', '樱', '真']
```

7.8 精彩案例赏析

- **例7-1** 编写函数实现字符串加密和解密，循环使用指定密钥，采用简单的异或算法。

```
def crypt(source, key):
    from itertools import cycle
    result = ''
    temp = cycle(key)
    for ch in source:
        result = result + chr(ord(ch) ^ ord(next(temp)))
    return result
```

7.8 精彩案例赏析

```
source = 'Shandong Institute of Business and Technology'

key = 'Dong Fuguo'


print('Before Encrypted:'+source)

encrypted = crypt(source, key)

print('After Encrypted:'+encrypted)

decrypted = crypt(encrypted, key)

print('After Decrypted:'+decrypted)
```


7.8 精彩案例赏析

- **例7-2** 编写程序，生成大量随机信息，这在需要获取大量数据来测试或演示软件功能的时候非常有用，不仅能真实展示软件功能或算法，还可以避免泄露真实数据或者引起不必要的争议。

code\randomInformation.py

7.8 精彩案例赏析

- **例4-3** 检查并判断密码字符串的安全强度。

```
import string


def check(pwd):
    #密码必须至少包含6个字符
    if not isinstance(pwd, str) or len(pwd)<6:
        return 'not suitable for password'
    #密码强度等级与包含字符种类的对应关系
    d = {1:'weak', 2:'below middle', 3:'above middle', 4:'strong'}
    #分别用来标记pwd是否含有数字、小写字母、大写字母和指定的标点符号
    r = [False] * 4
```

7.8 精彩案例赏析

```
    for ch in pwd:
        #是否包含数字
        if not r[0] and ch in string.digits:
            r[0] = True
        #是否包含小写字母
        elif not r[1] and ch in string.ascii_lowercase:
            r[1] = True
        #是否包含大写字母
        elif not r[2] and ch in string.ascii_uppercase:
            r[2] = True
        #是否包含指定的标点符号
        elif not r[3] and ch in ',.!;?<>':
            r[3] = True
    #统计包含的字符种类，返回密码强度
    return d.get(r.count(True), 'error’)

print(check('a2Cd,'))
```