

第7章 字符串



第7章 文本处理(一)：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
- 7.6中英文分词
- 7.8精彩案例赏析

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。

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

◆一句话的字符串

第7章 字符串

```
>>> s = """STRAY birds of summer come to my window  
to sing and fly away.  
And yellow leaves of autumn,  
which have no songs,  
flutter and fall there with a sigh.
```

O TROUPE of little vagrants of the world,
leave your footprints in my words.

THE world puts off its mask of vastness to its lover.
It becomes small as one song,
as one kiss of the eternal.

IT is the tears of the earth
that keep her smiles in bloom.

THE mighty desert is burning
for the love of a blade of grass
who shakes her head and laughs
and flies
away.

IF you shed tears when you miss the sun,
you also miss the stars.

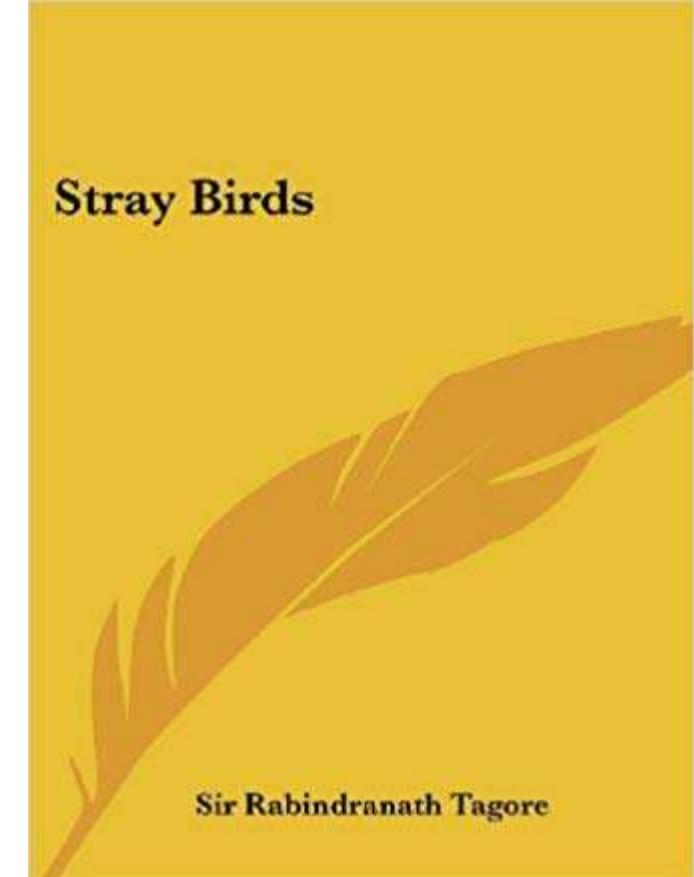
THE sands in your way beg for your song
and your movement,
dancing water.
Will you carry the burden of their lameness?

HER wistful face haunts my dreams
like the rain at night.

ONCE we dreamt that we were strangers.
We wake up to find that we were dear to each other.

SORROW is hushed into peace in my heart
like the evening among the silent trees."""

◆ 多行字符串



第7章 字符串

◆一本书的字符串

```
>>> fp = open(r'C:\Users\xiong\Nutstore\1\我的坚果云\2025秋冬  
_python程序设计\PPT\py\红楼梦.txt', 'r', encoding='utf-8')
```

```
>>> hlm = fp.read()
```

```
>>> fp.close()
```

```
>>> len(hlm)
```

```
873792
```

```
>>> print(hlm[:500]) #前500字
```

第一回甄士隐梦幻识通灵贾雨村风尘怀闺秀

一此开卷第一回也。作者自云：曾经历过一番梦幻之后，故将真事隐去，而借



第7章 字符串

- 字符串支持序列的通用操作：包括双向索引、比较大小、计算长度、切片、成员测试等操作

```
>>> s = '浙江工业大学'  
>>> s[-2:]  
'大学'
```

- 字符串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
- 字符串属于不可变序列，不能直接对字符串对象进行原地字符增加、修改与删除等操作，各操作均创建新的字符串

```
>>> s.replace('工业', '')  
'浙江大学'
```

第7章 字符串

| 操作类别 | 操作名 | 语法示例 (Python) | 适用序列类型 |
|--------|---------|---------------------------------|--------------|
| 通用操作 | 索引 | seq[0] | 所有序列 |
| | 切片 | seq[1:4] | 所有序列 |
| | 成员检查 | 'a' in seq | 所有序列 |
| | 长度 | len(seq) | 所有序列 |
| | 最小值/最大值 | min(seq), max(seq) | 元素可比较的序列 |
| | 连接/重复 | seq1 + seq2, seq * 3 | 所有序列 |
| | 迭代 | for x in seq: | 所有序列 |
| | 查找索引 | seq.index('a') | 所有序列 |
| | 计数 | seq.count('a') | 所有序列 |
| 可变序列操作 | 赋值 | seq[0] = 1, seq[1:3] = [4,5] | 可变序列 (如list) |
| | 添加元素 | seq.append(x), seq.insert(i, x) | 可变序列 |
| | 删除元素 | del seq[0], seq.remove(x) | 可变序列 |
| | 排序/反转 | seq.sort(), seq.reverse() | 可变序列 |

第7章 字符串

- 字符串中的字符属于**Unicode字符集**, 每个字符都有一个唯一的编号 (**码点**)

```
>>> for ch in s:  
    print(ch, ord(ch))
```

浙 27993
江 27743
工 24037
业 19994
大 22823
学 23398

- Python默认使用**UTF-8**(8-bit Unicode Transformation Format)编码格式以二进制的形式存储字符

```
>>> import sys  
>>> sys.getdefaultencoding()  
'utf-8'
```

第7章 字符串

- 除了支持str类型之外，Python还支持字节串类型bytes
 - 字符串可以通过**encode()**方法使用指定的编码格式编码成为**bytes**对象
 - 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

```
>>> s = '浙江工业大学'  
>>> s.encode()  
b'\xe6\xb5\x99\xe6\xb1\x9f\xe5\xb7\xa5\xe4\xb8\x9a\xe5\xa4\xa7\xe5\xad\xa6 '  
>>> len(s)  
6  
>>> len(s.encode())  
18 #三个字节构成一个汉字  
>>> s.encode().decode()  
'浙江工业大学'
```

第3章 文本处理(一)：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
- 7. 6中英文分词
- 7. 8精彩案例赏析

| 字符 | 码点 | UTF-8 编码 | 说明 |
|----------------|--------|----------|-------------|
| m (拉丁文小写m) | U+006D | 6D | ASCII字符，1字节 |
| م (阿拉伯文字母Meem) | U+062D | D8 AD | 阿拉伯字符，2字节 |
| S(拉丁文大写S带逗号) | U+0219 | C8 99 | 欧洲字符，2字节 |
| 看 (中文字符"看") | U+770B | E7 9C 8B | 中文字符，3字节 |

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，共编码128个字符

7.1 字符串编码格式简介

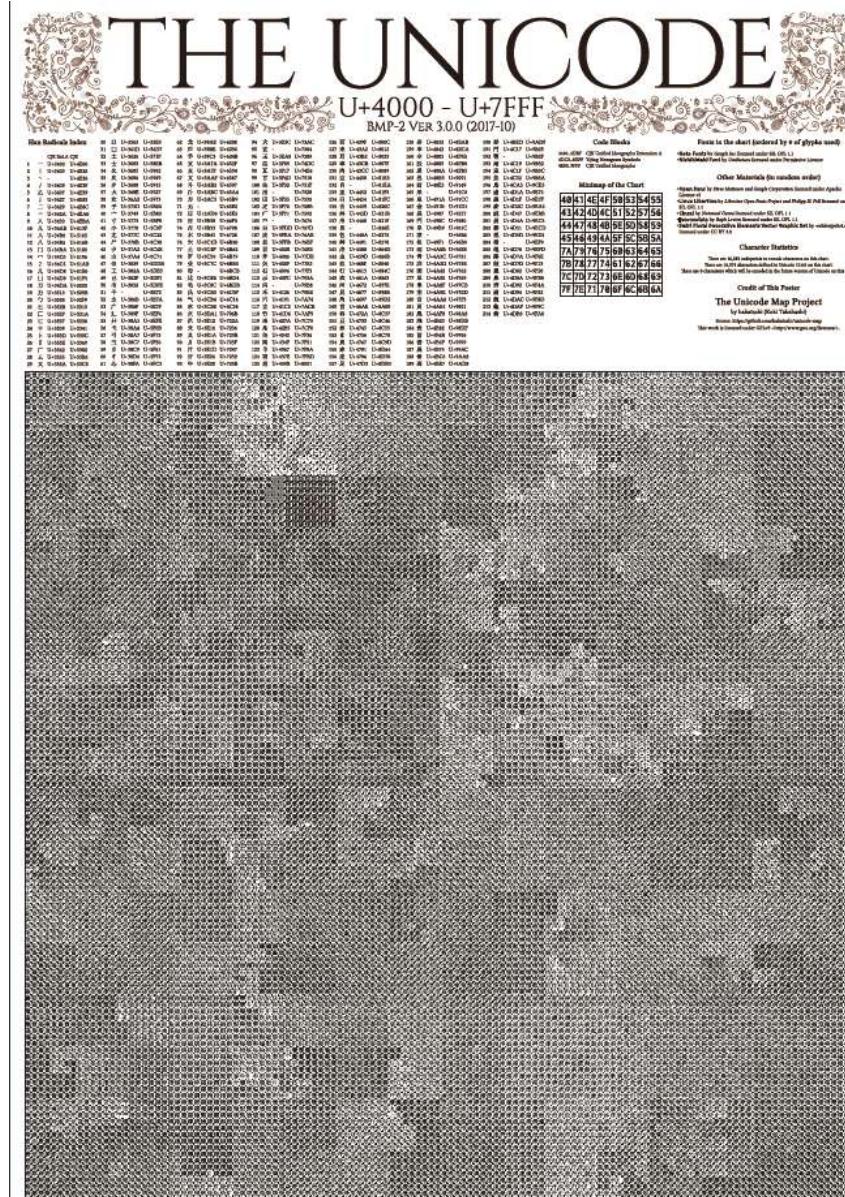
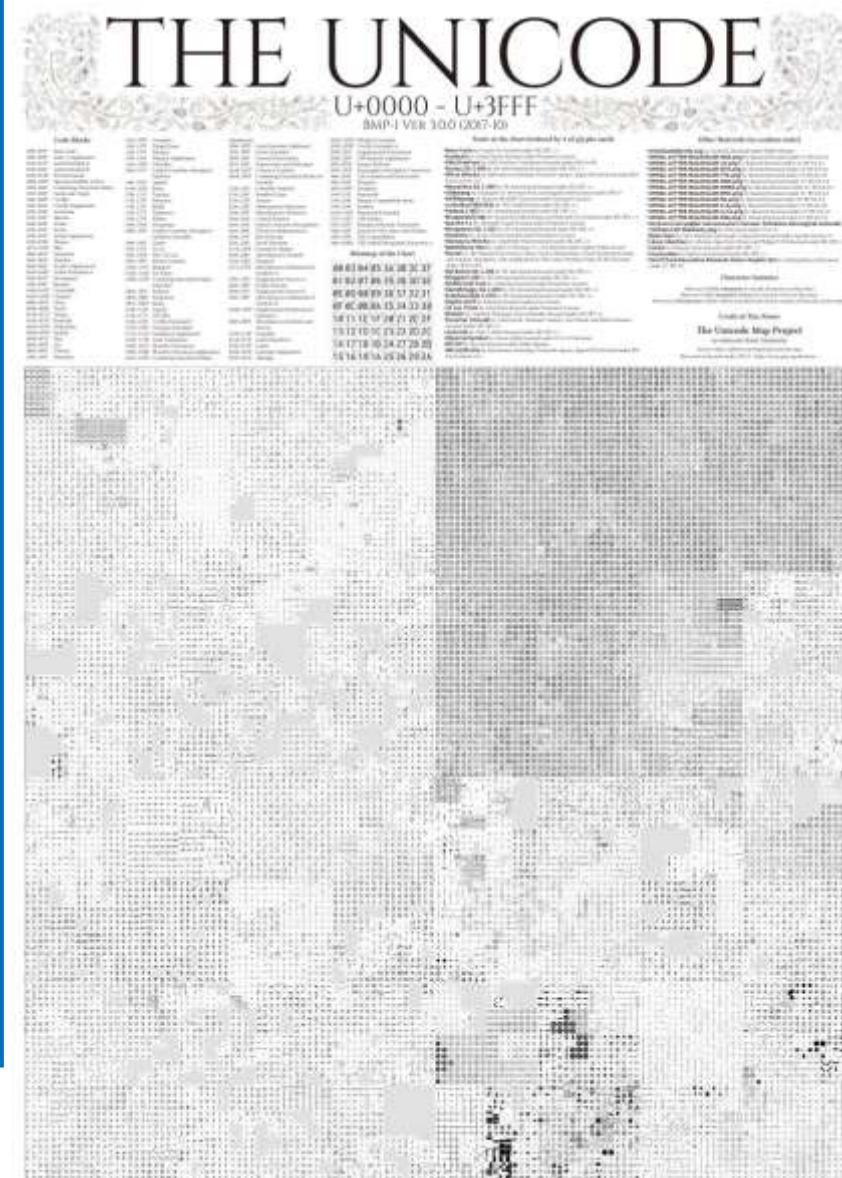
| Dec | Hx | Oct | Char | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr | Dec | Hx | Oct | Html | Chr |
|-----|--------|-----|------------------------------------|-----|--------|-----|-------|--------------|-----|--------|-----|-------|----------|-----|--------|-----|--------|------------|
| 0 | 0 000 | 000 | NUL (null) | 32 | 20 040 | 040 | | Space | 64 | 40 100 | 100 | @ | Ø | 96 | 60 140 | 140 | ` | ` |
| 1 | 1 001 | 001 | SOH (start of heading) | 33 | 21 041 | 041 | ! | ! | 65 | 41 101 | 101 | A | A | 97 | 61 141 | 141 | a | a |
| 2 | 2 002 | 002 | STX (start of text) | 34 | 22 042 | 042 | " | " | 66 | 42 102 | 102 | B | B | 98 | 62 142 | 142 | b | b |
| 3 | 3 003 | 003 | ETX (end of text) | 35 | 23 043 | 043 | # | # | 67 | 43 103 | 103 | C | C | 99 | 63 143 | 143 | c | c |
| 4 | 4 004 | 004 | EOT (end of transmission) | 36 | 24 044 | 044 | $ | \$ | 68 | 44 104 | 104 | D | D | 100 | 64 144 | 144 | d | d |
| 5 | 5 005 | 005 | ENQ (enquiry) | 37 | 25 045 | 045 | % | % | 69 | 45 105 | 105 | E | E | 101 | 65 145 | 145 | e | e |
| 6 | 6 006 | 006 | ACK (acknowledge) | 38 | 26 046 | 046 | & | & | 70 | 46 106 | 106 | F | F | 102 | 66 146 | 146 | f | f |
| 7 | 7 007 | 007 | BEL (bell) | 39 | 27 047 | 047 | ' | ' | 71 | 47 107 | 107 | G | G | 103 | 67 147 | 147 | g | g |
| 8 | 8 010 | 010 | BS (backspace) | 40 | 28 050 | 050 | (| (| 72 | 48 110 | 110 | H | H | 104 | 68 150 | 150 | h | h |
| 9 | 9 011 | 011 | TAB (horizontal tab) | 41 | 29 051 | 051 |) |) | 73 | 49 111 | 111 | I | I | 105 | 69 151 | 151 | i | i |
| 10 | A 012 | 012 | LF (NL line feed, new line) | 42 | 2A 052 | 052 | * | * | 74 | 4A 112 | 112 | J | J | 106 | 6A 152 | 152 | j | j |
| 11 | B 013 | 013 | VT (vertical tab) | 43 | 2B 053 | 053 | + | + | 75 | 4B 113 | 113 | K | K | 107 | 6B 153 | 153 | k | k |
| 12 | C 014 | 014 | FF (NP form feed, new page) | 44 | 2C 054 | 054 | , | , | 76 | 4C 114 | 114 | L | L | 108 | 6C 154 | 154 | l | l |
| 13 | D 015 | 015 | CR (carriage return) | 45 | 2D 055 | 055 | - | - | 77 | 4D 115 | 115 | M | M | 109 | 6D 155 | 155 | m | m |
| 14 | E 016 | 016 | SO (shift out) | 46 | 2E 056 | 056 | . | . | 78 | 4E 116 | 116 | N | N | 110 | 6E 156 | 156 | n | n |
| 15 | F 017 | 017 | SI (shift in) | 47 | 2F 057 | 057 | / | / | 79 | 4F 117 | 117 | O | O | 111 | 6F 157 | 157 | o | o |
| 16 | 10 020 | 020 | DLE (data link escape) | 48 | 30 060 | 060 | 0 | 0 | 80 | 50 120 | 120 | P | P | 112 | 70 160 | 160 | p | p |
| 17 | 11 021 | 021 | DC1 (device control 1) | 49 | 31 061 | 061 | 1 | 1 | 81 | 51 121 | 121 | Q | Q | 113 | 71 161 | 161 | q | q |
| 18 | 12 022 | 022 | DC2 (device control 2) | 50 | 32 062 | 062 | 2 | 2 | 82 | 52 122 | 122 | R | R | 114 | 72 162 | 162 | r | r |
| 19 | 13 023 | 023 | DC3 (device control 3) | 51 | 33 063 | 063 | 3 | 3 | 83 | 53 123 | 123 | S | S | 115 | 73 163 | 163 | s | s |
| 20 | 14 024 | 024 | DC4 (device control 4) | 52 | 34 064 | 064 | 4 | 4 | 84 | 54 124 | 124 | T | T | 116 | 74 164 | 164 | t | t |
| 21 | 15 025 | 025 | NAK (negative acknowledge) | 53 | 35 065 | 065 | 5 | 5 | 85 | 55 125 | 125 | U | U | 117 | 75 165 | 165 | u | u |
| 22 | 16 026 | 026 | SYN (synchronous idle) | 54 | 36 066 | 066 | 6 | 6 | 86 | 56 126 | 126 | V | V | 118 | 76 166 | 166 | v | v |
| 23 | 17 027 | 027 | ETB (end of trans. block) | 55 | 37 067 | 067 | 7 | 7 | 87 | 57 127 | 127 | W | W | 119 | 77 167 | 167 | w | w |
| 24 | 18 030 | 030 | CAN (cancel) | 56 | 38 070 | 070 | 8 | 8 | 88 | 58 130 | 130 | X | X | 120 | 78 170 | 170 | x | x |
| 25 | 19 031 | 031 | EM (end of medium) | 57 | 39 071 | 071 | 9 | 9 | 89 | 59 131 | 131 | Y | Y | 121 | 79 171 | 171 | y | y |
| 26 | 1A 032 | 032 | SUB (substitute) | 58 | 3A 072 | 072 | : | : | 90 | 5A 132 | 132 | Z | Z | 122 | 7A 172 | 172 | z | z |
| 27 | 1B 033 | 033 | ESC (escape) | 59 | 3B 073 | 073 | ; | : | 91 | 5B 133 | 133 | [| [| 123 | 7B 173 | 173 | { | { |
| 28 | 1C 034 | 034 | FS (file separator) | 60 | 3C 074 | 074 | < | < | 92 | 5C 134 | 134 | \ | \ | 124 | 7C 174 | 174 | | | |
| 29 | 1D 035 | 035 | GS (group separator) | 61 | 3D 075 | 075 | = | = | 93 | 5D 135 | 135 |] |] | 125 | 7D 175 | 175 | } | } |
| 30 | 1E 036 | 036 | RS (record separator) | 62 | 3E 076 | 076 | > | > | 94 | 5E 136 | 136 | ^ | ^ | 126 | 7E 176 | 176 | ~ | ~ |
| 31 | 1F 037 | 037 | US (unit separator) | 63 | 3F 077 | 077 | ? | ? | 95 | 5F 137 | 137 | _ | _ | 127 | 7F 177 | 177 | | DEL |

Dec: Decimal → 10
 Hx: Hexadecimal → 16
 Oct: Octal → 8

7.1 字符串编码格式简介

- UTF-8对全世界所有国家需要用到的字符（Unicode字符集）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
- 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（国标扩展）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
- GB2312等不支持一些其他语言的符号

7.1 字符串编码格式简介



7.1 字符串编码格式简介

■ UTF-8编码是如何识别一个字符使用几个字节存储的(了解)

```
>>> ord('熊')  
29066 #unicode码点  
#默认十进制显示  
>>> bin(ord('熊')) #二进制  
'0b0111000110001010'
```

| 特性 | Unicode码点 | UTF-8编码 |
|-----|-----------------|---------------|
| 本质 | 字符的抽象编号 | 字符的具体字节表示 |
| 用途 | 字符的唯一标识 | 存储和传输格式 |
| 值类型 | 整数 (0-0x10FFFF) | 字节序列 (1-4个字节) |
| 确定性 | 一个字符对应一个码点 | 一个码点对应一种编码 |

7.1 字符串编码格式简介

■ UTF-8编码是如何识别下一个字符使用几个字节存储的(了解)

```
>>> s = '熊'.encode() #编译为字节  
>>> s  
b'\xe7\x86\x8a' #默认显示十六进制字节值  
>>> bin(int('0x' + s.hex(), base=16))  
# 添加 0x 前缀表示这是一个十六进制字符串  
#s.hex()字节对象转为十六进制字符串  
#int(base=16)将十六进制字符串转为十进制整数  
'0b111001111000011010001010' #二进制的UTF-8编码
```

11100111 10000110 10001010 ← 分成3个字节
e7 / _86_ / _8a_ / ← 对应的十六进制

7.1 字符串编码格式简介

| 表示方式 | 值 | 说明 |
|-----------|----------------------------|------------------|
| 字符 | '熊' | 实际的汉字 |
| Unicode码点 | 29066 (0x718c) | 字符在Unicode表中的编号 |
| 码点二进制 | 0b0111000110001010 | 编号的二进制表示 |
| UTF-8编码 | b'\xe7\x86\x8a' | 字符在UTF-8编码中的字节序列 |
| UTF-8二进制 | 0b111001111000011010001010 | 编码的二进制表示 |

Unicode码点：字符的抽象编号，与具体编码无关

UTF-8编码：具体的字节表示，**用于存储和传输**

UTF-8是一种变长编码，汉字通常需要3个字节：

首字节：1110xxxx (表示3字节序列)

后续字节：10xxxxxx

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存储形式
- 把同一字符以不同编码格式存入文件时，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

```
>>> s = '浙江工业大学'  
>>> s.encode('utf-8').decode('gbk')  
Traceback (most recent call last):  
  File "<pyshell#23>", line 1, in <module>  
    s.encode('utf-8').decode('gbk')  
UnicodeDecodeError: 'gbk' codec can't decode byte 0xad in position 16:  
illegal multibyte sequence  
  
>>> 'abc'.encode('gbk').decode('utf-8')  

```

UTF-8编码 -> UTF-8解码 (正确)

GBK编码 -> GBK解码 (正确，但前提是s包含的字符都在GBK字符集中)

第3章 文本处理(一)：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
- 7. 6中英文分词
- 7. 8精彩案例赏析

7.2 转义字符与原始字符串

保持字符串原样输出有3种方法：元字符串、`repr`函数和转义字符（\）。

元字符串

`repr`函数

转义字符（\）

元字符串即在任意字符串之前添加字母r或R，那么，当前字符串中所有转义字符在使用时都**不会进行转义操作**。正则表达式中常见该格式。 eg. `print(r'abc\n')`

如果希望按原始字符串输出，也可使用**repr函数**，不过，此时原字符串会使用一对单引号括起来。eg. `print(repr("abc\n"))`

如果只想原样输出转义字符，如\n、\t等，则可以在前面**再加一个反斜杠**，也就是使用两个反斜杠，这样，第一个反斜杠后的\n、\t符号会被认为是普通字符，而不是转义字符。 eg. `print('abc\\n')`

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  
>>> 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('\u6d59\u6c5f\u5de5\u4e1a\u5927\u5b66')四位十六进制数表示Unicode  
字符  
浙江工业大学
```

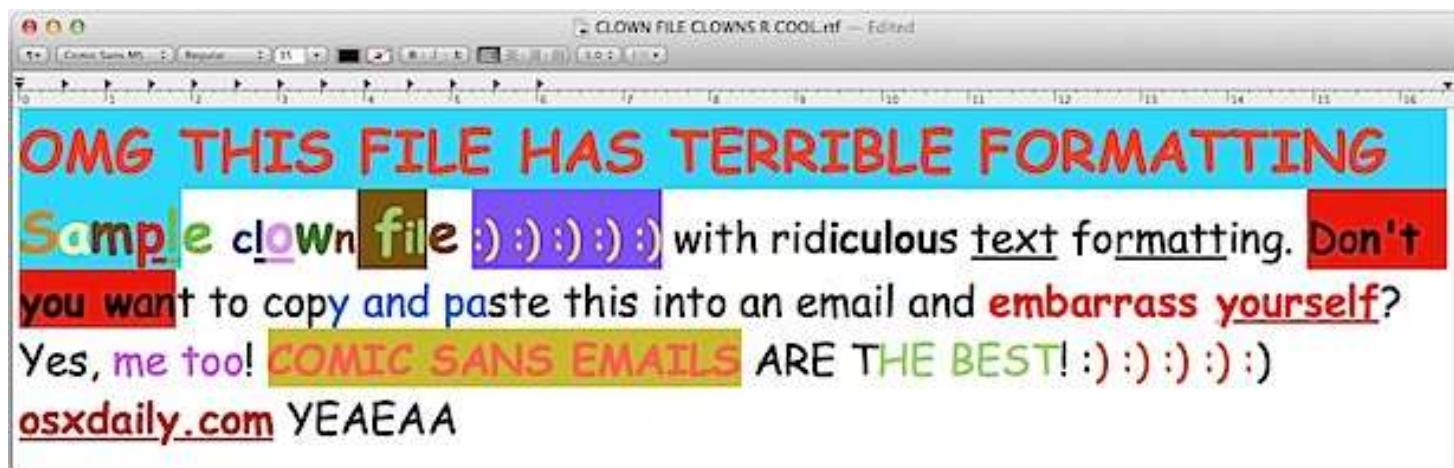
7.2 转义字符与原始字符串

- 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  
notepad.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  
  
>>> path = 'C:\\Windows\\notepad.exe'  #r不是万能的，也可以用\\转义  
>>> print(path)  
C:\Windows\notepad.exe
```

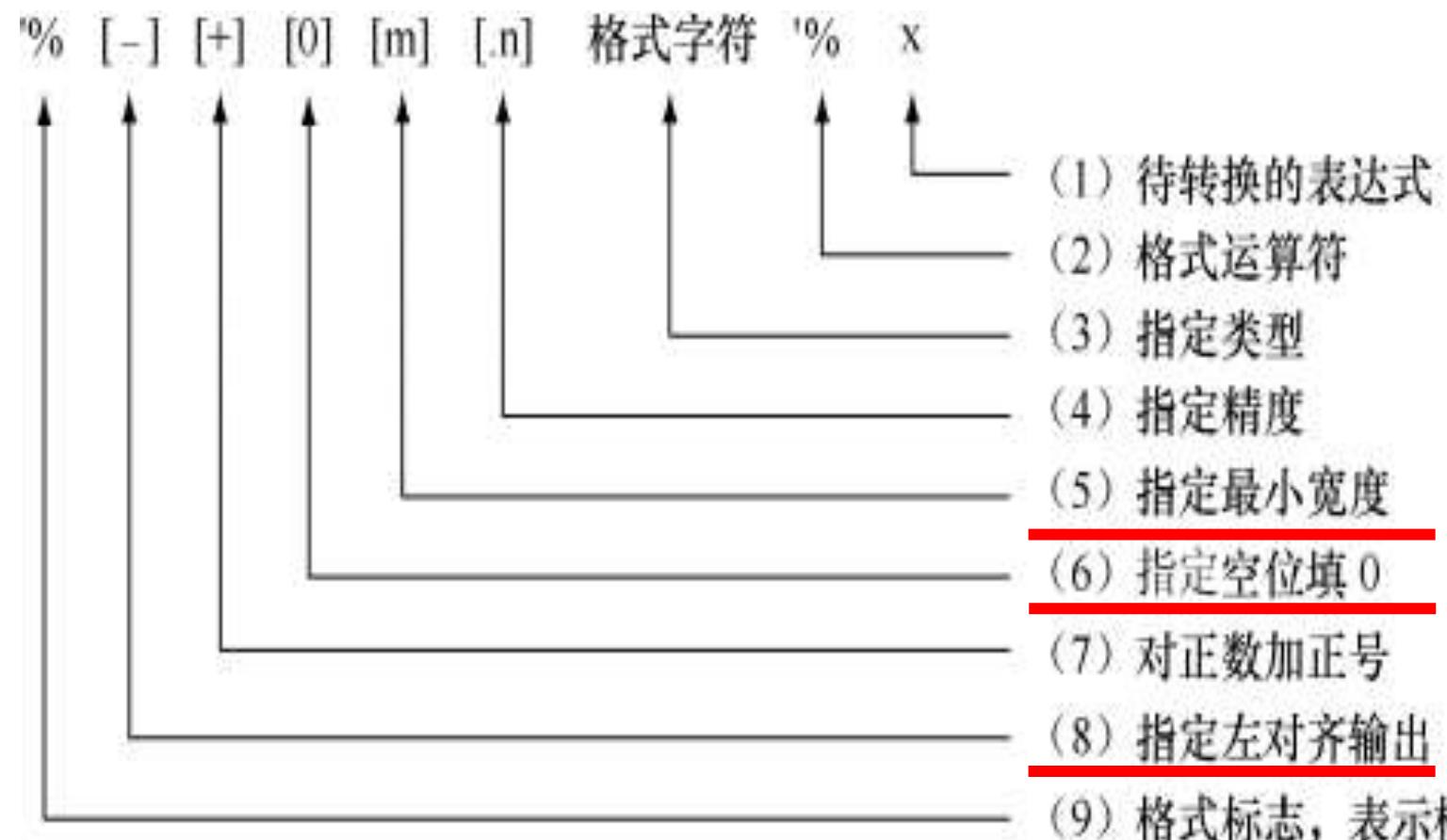
第3章 文本处理(一)：字符串

- 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. 3. 1使用%符号进行格式化
 - 7. 3. 2使用format()方法进行字符串格式化
 - 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

■ 通过格式化创建以特定格式表示数据的字符串



""将一个浮点数格式化为小数点后6位，并在一个至少20字符宽的字段中左对齐""

```
>>>s = '%-20.6f'%12.5  
>>> s  
'12.500000        '  
>>>s ='%020.6f'%12.5  
>>> s  
'000000000012.500000'
```

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

```
>>> for i in range(9,12):
    print('f%02d'%i)

f09
f10
f11

>>> '(%d, %d)'%(65,65)
'(65, 65)'
>>> '[%d, %d]'%[65,65]
Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    '[%d, %d]'%[65,65]
TypeError: %d format: a number is required, not list
# % 操作符实际上期望的是一个元组 (tuple) 作为右侧的参数
```

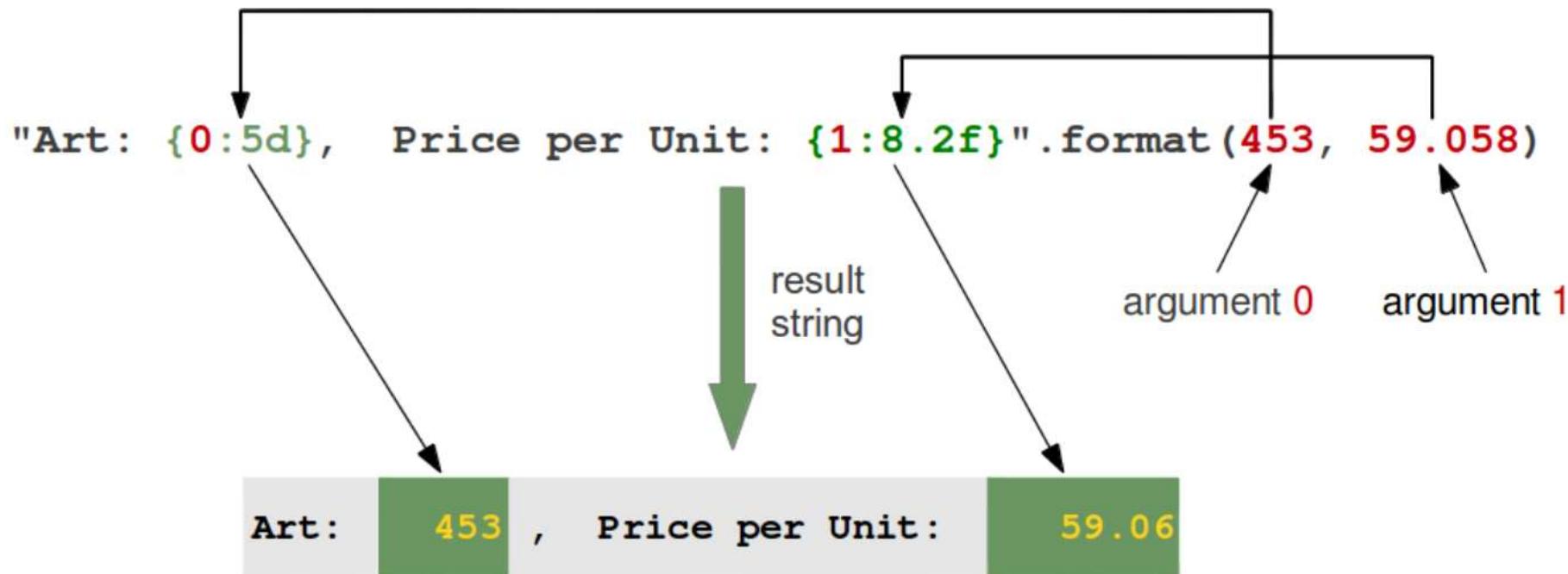
第3章 文本处理(一)：字符串

- 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. 3. 1使用%符号进行格式化
 - 7. 3. 2使用**format()**方法进行字符串格式化
 - 7. 3. 3格式化的字符串常量

7.3.2 使用format()方法进行格式化

- 用format()方法格式化字符串更符合Python的风格，且更灵活，**推荐使用**

str.format()



7.3.2 使用format()方法进行格式化

■ 占位符 (placeholder) 的形式为 {参数:格式码}

```
>>> 1/3  
0.3333333333333333  
  
>>> print('{0:.3f}'.format(1/3))      #保留3位小数  
0.333  
  
>>> print(':.3f'.format(1/3))        #参数0可省略  
0.333  
  
>>> print('{0}'.format(1/3))  
0.3333333333333333  
  
>>> print('{}'.format(1/3))          #0可省略，只要占位即可  
0.3333333333333333  
  
  
>>> '{0:.2%}'.format(0.035)        #格式化为百分数  
'3.50%'    # %: 格式类型。这是一个特殊的格式类型，它会将数字乘以100，并添加一个百分号%。
```

7.3.2 使用format()方法进行格式化

- 参数可以是序号或关键字参数的名字，且若参数表示序列时，可使用下标

```
>>> 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
```

```
>>> position = (5, 8, 13)
```

```
>>> print('X:{}; Y:{}; Z:{}'.format(position[0],position[1],position[2]))
```

```
X:5;Y:8;Z:13
```

```
>>> print("X:{};Y:{};Z{}".format(*position)) #用*做序列解包，前面下标可省略
```

```
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]}'"
>>> for item in weather:
    print(formatter.format(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'
```

#或用序列解包方式

```
>>> formatter = "Weather of '{}' is '{}'"
>>> for item in weather:
    print(formatter.format(*item))
```

第3章 文本处理(一)：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. 3. 1使用%符号进行格式化
 - 7. 3. 2使用format()方法进行字符串格式化
 - **7. 3. 3格式化的字符串常量**

7.3.3 格式化的字符串常量（强烈推荐！）

- 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，常被称为f字符串，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.'  
>>> print(_)  
My name is Dong, and I am 39 years old.
```

7.3.3 格式化的字符串常量

```
>>> width = 10  
>>> precision = 4  
>>> value = 11/3  
>>> f'result:{value:{width}.{precision}f}'  
'result: 3.6667'  
  
>>> 'result:{0:{1}.{2}f}'.format(value, width, precision)  
'result: 3.6667'
```

Practice

- UTF-8和GB2312分别使用几个字节存储英文字母和汉字?
✓ utf-8: 1和3字节
✓ gb2312: 1和2字节
- ‘f{…}-{…}.txt’ .format(12, 2) 的输出为 ‘f012-02.txt’，两个占位符分别怎么写?
✓ 'f{0:03d}-{1:02d}.txt'.format(12, 2) #参数0和1可省

习题

7.1 已知列表对象 `x = ['11', '2', '3']`, 则表达式 `max(x)` 的值为 _____。

7.2 已知列表对象 `x = ['11', '2', '3']`, 则表达式 `max(x, key=len)` 的值为 _____。

7.6 表达式 `'a' + 'b'` 的值为 _____。

7.7 表达式 `len('中国共产党'.encode('utf-8'))` 的值为 _____。

7.11 单选题: 表达式 `'{},{}'.format(3, 5)` 的值为 ()。

- A. '3.5'
- B. '3,5'
- C. '35'
- D. '5,3'

7.12 单选题: 表达式 `'{1},{0}'.format(3, 5)` 的值为 ()。

- A. '1,0'
- B. '3,5'
- C. '0,1'
- D. '5,3'

第3章 文本处理(一)：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
- 7. 6中英文分词
- 7. 8精彩案例赏析

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 #dir(str)
- **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 字符串常用操作

| 方法 | 功能描述 |
|---|--|
| <code>capitalize()</code> | 返回新字符串，第一个英文字母大写，其余小写 |
| <code>center(width, fillchar=' ', /)</code> <code>ljust(width, fillchar=' ', /)</code> <code>rjust(width, fillchar=' ', /)</code> | 返回指定长度的新字符串，当前字符串在新字符串中居中/居左/居右，如果参数width指定的新字符串长度大于当前字符串长度就在两侧/右侧/左侧使用参数fillchar指定的字符（默认为空格）进行填充；如果参数width指定的长度小于或等于当前字符串的长度，直接返回当前字符串，不会进行截断 |
| <code>count(sub[, start[, end]])</code> | 返回字符串sub在当前字符串下标范围[start, end)内不重叠出现的次数，参数start默认值为0，参数end默认值为字符串长度。例如，'abababab'.count('aba')的值为2 |
| <code>encode(encoding='utf-8', errors='strict')</code> | 返回当前字符串使用参数encoding指定的编码格式编码后的字节串。对于非ASCII字符，UTF-8编码得到的字节串较长，网络传输时占用带宽较大，保存为文件时占用空间较大；GBK编码得到的字节串更短一些，但有些字符不在GBK字符集中，无法使用GBK编码 |
| <code>endswith(suffix[, start[, end]])</code> <code>startswith(prefix[, start[, end]])</code> | 如果当前字符串下标范围[start, end)的子串以某个字符串suffix/prefix或元组suffix/prefix指定的几个字符串之一结束/开始则返回True，否则返回False |

可dir(str)
或'a'.
查看所有

7.4 字符串常用操作

| | |
|---|---|
| <code>expandtabs(tabsize=8)</code> | 返回新字符串，所有 tab 键都替换为指定数量的空格。 例如，'abcd\t e'.expandtabs()返回'abcd e' 'abcdef\t e'.expandtabs()返回'abcdef e' |
| <code>find(sub[, start[, end]])</code> <code>rfind(sub[, start[, end]])</code> | 返回字符串sub在当前字符串下标范围[start,end)内出现的最小/最大下标，不存在时返回-1 |
| <code>format(*args, **kwargs)</code> | 返回对当前字符串进行格式化（格式化是指把字符串中大括号以及内部变量或编号指定的占位符替换为实际值并以指定的格式呈现）后的新字符串，其中args表示位置参数，kwargs表示关键参数，见5.2节 |
| <code>index(sub[, start[, end]])</code> <code>rindex(sub[, start[, end]])</code> | 返回字符串sub在当前字符串下标范围[start,end)内出现的最小/最大下标，不存在时抛出ValueError异常并提示子串不存在 |
| <code>isalnum()</code> 、 <code>isalpha()</code> 、 <code>isascii()</code> 、 <code>isdecimal()</code> 、 <code>isdigit()</code> 、 <code>isidentifier()</code> 、 <code>islower()</code> 、 <code>isnumeric()</code> 、 <code>isprintable()</code> 、 <code>isspace()</code> 、 <code>istitle()</code> 、 <code>isupper()</code> | 测试字符串中字符的类型 |

7.4 字符串常用操作

| | |
|---|---|
| join(iterable, /) | 使用当前字符串作为连接符把参数iterable中的所有字符串连接成为一个长字符串并返回连接之后的长字符串，要求参数iterable指定的可迭代对象中所有元素全部为字符串 |
| lower() upper() | 返回当前字符串中所有字母都变为小写/大写之后的新字符串，非字母字符保持不变 |
| lstrip(chars=None, /) rstrip(chars=None, /) strip(chars=None, /) | 返回当前字符串删除左侧/右侧/两侧的空白字符或参数chars中所有字符之后的新字符串 |
| maketrans(...) | 根据参数给定的字典或者两个等长字符串对应位置的字符，构造并返回字符映射表（形式上是字典，“键”和“值”都是字符的Unicode编码），如果指定了第三个参数（必须为字符串）则该参数中所有字符都被映射为空值None。该方法是字符串类str的静态方法，可以通过任意字符串进行调用，也可以直接通过字符串类str进行调用 |
| partition(sep, /) rpartition(sep, /) | 使用参数sep指定字符串的第一次/最后一次出现作为分隔符，把当前字符串分隔为3个子串，返回包含3个子串的元组。如果指定的子串不存在，返回当前字符串和两个空串组成的元组 |

7.4 字符串常用操作

| | |
|---|---|
| <code>removeprefix(prefix, /)</code> <code>removesuffix(suffix, /)</code> | 返回删除前缀/后缀之后的新字符串 |
| <code>replace(old, new, count=-1, /)</code> | 返回当前字符串中所有子串old都被替换为子串new之后的新字符串，参数count用来指定最大替换次数，默认值-1表示全部替换 |
| <code>rsplit(sep=None, maxsplit=-1)</code> <code>split(sep=None, maxsplit=-1)</code> | 使用参数sep指定的字符串对当前字符串从后向前/从前向后进行切分，返回包含切分后所有子串的列表。参数sep=None表示使用所有空白字符作为分隔符并丢弃切分结果中的所有空字符串，参数maxsplit表示最大切分次数，默认值-1表示没有限制 |
| <code>splitlines(keepends=False)</code> | 使用换行符切分字符串，返回列表 |
| <code>swapcase()</code> | 交换英文字母大小写，返回新字符串 |
| <code>title()</code> | 返回新字符串，每个单词的首字母大写，其余字母小写 |
| <code>translate(table, /)</code> | 根据参数table指定的映射表对当前字符串中的字符进行替换并返回替换后的新字符串，不影响原字符串，参数table一般为字符串方法maketrans()创建的映射表，其中映射为空值None的字符将会被删除而不出现在新字符串中 |

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

- `find(sub[, start[, end]])`和`rfind(sub[, start[, end]])`方法分别用来查找另一个字符串在当前字符串指定范围（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(sub[, start[, end]])`和`rindex(sub[, start[, end]])`方法用来返回另一个字符串在当前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(sub[, start[, end]])`方法用来返回另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) #7-19之间  
-1 #不存在  
>>> 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
```

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | p | p | l | e | , | p | e | a | c | h | , | b | a | n | a | , | p | e | a | c | h | , | p | e | a | r |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 |

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | |

7.4.2 split()、rsplit()

- `split(sep=None, maxsplit=-1)`和`rsplit(sep=None, maxsplit=-1)`方法分别用来以参数字符串为分隔符，把当前字符串从左往右或从右往左分隔成多个字符串，返回包含分隔结果的列表。

7.4.2 split()、rsplit()

```
>>> s = 'apple,peach,banana,pear'  
>>> s.split(',')  
['apple', 'peach', 'banana', 'pear']  
  
>>> s = '2025-11-10'  
>>> t = s.split('-')  
>>> print(t)  
['2025', '11', '10']  
>>> print(list(map(int, t)))  
[2025, 11, 10]
```

分隔符

7.4.2 split()、rsplit()

- `split()`和`rsplit()`方法允许通过参数`maxsplit`指定最大分割次数，该参数默认值-1表示没有限制。

```
>>> s = '\n\nhello\t\t world \n\n\n My name is Dong '
>>> s.split(None, 1)          # 第一个参数None表示使用任意空白字符做分隔符
['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)       # 使用关键参数直接指定第二个参数
                               # 这时第一个参数使用默认值None
['hello', 'world', 'My', 'name', 'is', 'Dong']
>>> s.split(maxsplit=100)     # 最大分隔次数大于可分隔次数时，相当于-1
['hello', 'world', 'My', 'name', 'is', 'Dong']
```

7.4.2 split()、rsplit()

- 对于split()和rsplit()方法，如果不指定分隔符，则字符串中的**任何空白字符（空格、换行符、制表符等）**都将被认为是分隔符，并删除分割结果中的所有空字符串。

```
>>> 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()

◆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  
                                         # 相邻逗号直接会得到一个空字符串  
['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  
                                         # 相邻制表符之间会得到一个空字符串  
['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()

- 字符串方法 `join(iterable, /)` 用于连接可迭代对象中的字符串，以当前字符串作为连接符，返回连接后的新字符串。

连接符

```
>>> li = ["apple", "peach", "banana", "pear"]
```

```
>>> ','.join(li)
```

```
'apple,peach,banana,pear'
```

```
>>> '.'.join(li)
```

```
'apple.peach.banana.pear '
```

```
>>> '.'.join(map(str, [192, 168, 1, 0]))
```

```
'192.168.1.0'
```

```
>>> ' '.join(map(str, [192, 168, 1, 0]))
```

```
'192 168 1 0'
```

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 '
```

当 split() 不提供分隔符时，它会：

- ✓ 使用任何空白字符作为分隔符
- ✓ 自动合并连续的空白字符
- ✓ 忽略开头和结尾的空白字符

这是一个非常经典且实用的字符串处理技巧，用于规范化空白字符！

7.4.3 join()

■问题解决：判断两个字符串在Python意义上是否等价

```
>>> def equivalent(s1, s2):          #
    if s1 == s2:
        return True #字符串内容完全相同
    elif ' '.join(s1.split()) == ' '.join(s2.split()): #此句冗余，可不要
        return True #多个连续空格被压缩为单个空格后相等
    elif ''.join(s1.split()) == ''.join(s2.split()):
        return True #移除所有空格后相等
    else:
        return False
>>> equivalent('pip list', 'pip      list')
True
```

Practice

- 字符串的find()和index()方法的区别?
 - ✓ find()方法当被查找字符串不存在时，返回-1
 - ✓ index()则抛出异常
- 'a, b,, d,, '.split(',')返回的列表包含几个元素?
 - ✓ 6个 #5个逗号分出来是6个元素 #['a', 'b', '', 'd', '', '']
- 设x为一个整数列表，将其转化为逗号分隔的字符串
 - ✓ ', '.join(map(str, x))

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**字符串对象的replace(old, new, count=-1, /)方法把指定子字符串的所有出现都替换为另一个字符串，类似于Word中的“**全部替换**”功能，返回替换后的新字符串。必要时可以通过参数count指定替换次数。

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

```
>>> s.replace("中国", "中华人民共和国", 1)  
'中华人民共和国, 中国'
```

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（了解）
- translate()方法用来根据映射表中定义的对应关系转换字符串中的字符。（了解）

```
#创建映射表，将字符"abcdef123"——对应地转换为"uvwxyz@#$"  
>>> table = ''.maketrans('abcdef123', 'uvwxyz@#$')  
>>> type(table)  
<class 'dict'>  
>>> table  
{97: 117, 98: 118, 99: 119, 100: 120, 101: 121, 102: 122, 49: 64, 50: 35, 51:  
36}
```

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

```
>>> s = "Python is a greate programming language. I like it!"  
>>> s.translate(table)
```

```
'Python is u gryuty programming lunguugy. I liky it!'
```

```
>>> table1 = dict(zip(table.values(), table.keys()))
```

```
>>> s = 'Python is u gryuty programming lunguugy. I liky it!'
```

```
>>> s.translate(table1)
```

```
'Pethon is a greate programming langaage. I like 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（所有字母排成一个圆）。

```
>>> 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!"  
>>> s1 = kaisa(s, 3) #加密  
>>> s1  
'Sbwkrq lv d juhdwh surjudpplqj odqjxdjh. L olnh lw!'  
  
>>> s2 = kaisa(s1, -3) #解密  
>>> s2  
"Python is a greate programming language. I like it!"
```

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

示例：k = 3

```
lower = "abcdefghijklmnopqrstuvwxyz"  
upper = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

分解计算：

```
lower[3:] = "defghijklmnopqrstuvwxyz" # 从第3位开始到结尾  
lower[:3] = "abc" # 从开始到第3位 (不包括)  
upper[3:] = "DEFGHIJKLMNOPQRSTUVWXYZ" # 从第3位开始到结尾  
upper[:3] = "ABC" # 从开始到第3位 (不包括)
```

组合结果：

```
after = "defghijklmnopqrstuvwxyz" + "abc" + "DEFGHIJKLMNOPQRSTUVWXYZ" + "ABC"  
= "defghijklmnopqrstuvwxyzabc" + "DEFGHIJKLMNOPQRSTUVWXYZABC"  
= "defghijklmnopqrstuvwxyzabcDEFGHIJKLMNOPQRSTUVWXYZABC"
```

```
a b c d e f g h i j k l m n o p q r s t u v w x y z  
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓  
d e f g h i j k l m n o p q r s t u v w x y z a b c  
  
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓  
D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
```

7.4.6 strip()、rstrip()、lstrip()

| 方法 | 说明 |
|-------------------|-------------------------------|
| s.strip([chars]) | 在其 左侧和右侧 去除chars中列出的字符 |
| s.lstrip([chars]) | 在其 左侧 去除chars中列出的字符 |
| s.rstrip([chars]) | 在其 右侧 去除chars中列出的字符 |

- chars为可选参数，用于指定需要去除的字符，可以指定多个。如果不指定此参数，则默认去除空格、制表符\t、回车符\r和换行符\n等。
- **但不能删除中间部分的字符。**

7.4.6 strip()、rstrip()、lstrip()

■ strip()、rstrip()、lstrip()

```
>>> s = " abc "
>>> s.strip()                      #删除两侧空白字符
'abc'
>>> '\n\nhello world  \n\n'.strip()   #删除两侧空白字符以及换行符\n, 但中间部分不删除
'hello world'
>>> "aaaassddf".strip("a")        #删除两侧指定字符
'sddf'

>>> "aaaassddfaaa".rstrip("a")    #删除字符串右侧指定字符
'aaaassddf'
>>> "aaaassddfaaa".lstrip("a")    #删除字符串左侧指定字符
'sddf'aaa'
```

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
```

#应用1：检测文件名

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

#应用2：检测文件内容

```
>>> [line for line in s.split('\\n') if line.strip().endswith('.')] #以Stray Birds为例
```

第7章 字符串

```
>>> s = """STRAY birds of summer come to my window  
to sing and fly away.  
And yellow leaves of autumn,  
which have no songs,  
flutter and fall there with a sigh.
```

O TROUPE of little vagrants of the world,
leave your footprints in my words.

THE world puts off its mask of vastness to its lover.
It becomes small as one song,
as one kiss of the eternal.

IT is the tears of the earth
that keep her smiles in bloom.

THE mighty desert is burning
for the love of a blade of grass
who shakes her head and laughs
and flies
away.

IF you shed tears when you miss the sun,
you also miss the stars.

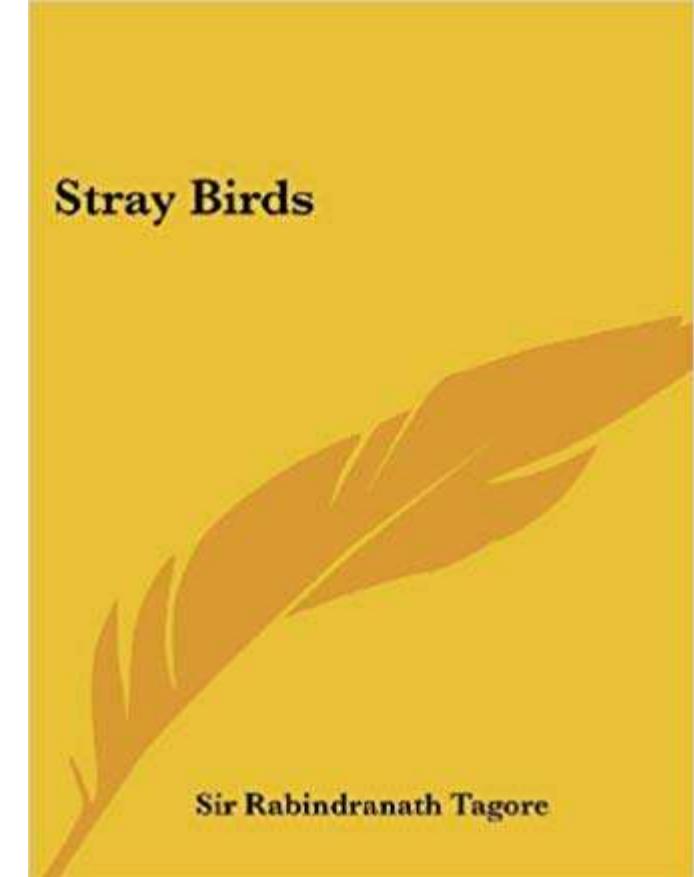
THE sands in your way beg for your song
and your movement,
dancing water.
Will you carry the burden of their lameness?

HER wistful face haunts my dreams
like the rain at night.

ONCE we dreamt that we were strangers.
We wake up to find that we were dear to each other.

SORROW is hushed into peace in my heart
like the evening among the silent trees."""

◆ 多行字符串



7.4.7 startswith()、endswith()

#应用2：检测文件内容

```
fp = open(r'C:\Users\xiong\Nutstore\1\我的坚果云\2025秋冬_python程序设计\PPT\py\红楼梦.txt', 'r', encoding='utf-8')
hlm = fp.read()
for line in hlm.split('\n'):
    if line.startswith('第'):
        print(line)
```

第二回贾夫人仙逝扬州城冷子兴演说荣国府

第三回托内兄如海荐西宾接外孙贾母惜孤女

第二个削肩细腰，长挑身材，鸭蛋脸儿，俊眼修眉，顾盼神飞，文彩精华，见之忘

第四回薄命女偏逢薄命郎葫芦僧判断葫芦案

第十五回贾宝玉神游太虚境警幻仙曲演红楼梦

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

- 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()          #检查字符串是否只包含字母和数字
```

```
True
```

```
>>> '1234abcd'.isalpha()         #全部为英文字母时返回True
```

```
False
```

```
>>> '1234'.isdigit()            #是否为数字字符
```

```
True
```

```
>>> '九'.isnumeric()           #isnumeric()方法支持汉字数字
```

```
True
```

```
>>> 'IVIII X'.isnumeric()      #支持罗马数字
```

```
True
```

| | |
|-------------|--|
| isdigit() | 检查字符串是否只包含数字字符 |
| isdecimal() | 检查字符串是否只包含十进制数字字符，比 isdigit() 更严格 |
| isnumeric() | 检查字符串是否只包含数字字符（包括分数、上标等），范围最广，包含汉字数字、罗马数字等 |

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!'
```

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

```
weather = [("Monday", "rainy"), ("Tuesday", "sunny"),
           ("Wednesday", "sunny"), ("Thursday", "rainy"),
           ("Friday", "cloudy")]

>>> for item in weather:
    print('Weather of {0} is {1}'.format(item[0].rjust(10), item[1].rjust(8)))
```

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
```

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

```
>>> for line in s.split('\n'):
    print(line.strip().rjust(50))
```

STRAY birds of summer come to my window
 to sing and fly away.
And yellow leaves of autumn,
 which have no songs,
flutter and fall there with a sigh.

O TROUPE of little vagrants of the world,
 leave your footprints in my words.

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

```
for i in range(10):
    print('f' + str(i).zfill(4))
    # print('f' + str(i).rjust(4,'0')) #亦可
f0000
f0001
f0002
f0003
f0004
f0005
f0006
f0007
f0008
f0009
```

Practice

- 使用字符串的哪个方法去掉字符串右端的空白字符?
✓ `rstrip()`
- 设fn为一文件名，如何判断它的后缀为'.exe'??
✓ `fn.endswith('.exe')`
- 设`s = 'IF you shed tears when you miss the sun, you also miss the stars.'`，如何得到其中以字母's'开头的单词组成的列表?
✓ `s_words = [word for word in s.split() if word.lower().startswith('s')]`
✓ `print(s_words)`
✓ `['shed', 'sun', 'stars.']` #可进一步用`replace`把', ' ' .' 去掉
✓ `s_words = [word for word in s.replace(',', '').replace('.', '').split() if word.lower().startswith('s')]`
✓ `print(s_words)`
✓ `['shed', 'sun', 'stars']`

习题

7.5 表达式 `len('Hello world!'.ljust(20))` 的值为_____。

7.8 已知 `text = '延安精神是中国共产党创造的一种革命精神,主要内容包括:实事求是、理论联系实际的精神,全心全意为人民服务的精神和自力更生艰苦奋斗的精神。'`, 那么表达式 `text.count('精神')` 的值为_____。

7.9 表达式 `len('::'.join(['a', 'b', 'c']))` 的值为_____。

7.13 单选题: 表达式 `'Beautiful is better than ugly.'.index('beautiful')` 的值为()。

- A. 0
- B. -1
- C. 1
- D. 引发异常

习题

7.14 单选题：表达式 `len('a,,b'.split(','))` 的值为（ ）。

- A. 1 B. 2 C. 3 D. 4

- A. 1 B. 2 C. 3 D. 4

7.16 单选题：表达式 '1234'.upper() 的值为（ ）。

- A. '1234'
 - B. '一二三四'
 - C. '壹貳叄肆'
 - D. 表达式错误

7.17 单选题：表达式 'ababababa'.rindex('aba') 的值为（ ）。

- A. 0 B. 2 C. 6 D. -2

习题

7.18 单选题：表达式 'ababababa'.count('aba') 的值为（ ）。

- A. 1
- B. 2
- C. 3
- D. 4

7.20 单选题：已知 `table = ''.maketrans('abc', 'ABC')`，那么表达式
`'dong fuguo'.translate(table)` 的值为（ ）。

- A. 'dong fuguo'
- B. 'Dong Fuguo'
- C. 'Dong FuGuo'
- D. 'Dong fuguo'

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'
```

- 成员判断，关键字in

```
>>> "a" in "abcde"      #测试一个字符中是否存在另一个字符串中  
True  
>>> 'ac' in 'abcde'    #关键字in左边的字符串作为一个整体对待  
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.'  
>>> list(zip(x,x))          # 组合对应位置上的字符  
[('H', 'H'), ('e', 'e'), ('l', 'l'), ('l', 'l'), ('o', 'o'), (' ', ' '), ('w', 'w'),  
 ('o', 'o'), ('r', 'r'), ('l', 'l'), ('d', 'd'), ('.', '.')]  
>>> sorted(x)              # 按字符Unicode编码排序  
[' ', '.', 'H', 'd', 'e', 'l', 'l', 'l', 'o', 'o', 'r', 'w']  
>>> list(reversed(x))      # 翻转  
['.', 'd', 'l', 'r', 'o', 'w', ' ', 'o', 'l', 'l', 'e', 'H']  
>>> list(enumerate(x))     # 枚举字符串中的字符  
[(0, 'H'), (1, 'e'), (2, 'l'), (3, 'l'), (4, 'o'), (5, ' '), (6, 'w'), (7, 'o'), (8,  
 'r'), (9, 'l'), (10, 'd'), (11, '.')]  
>>> list(map(lambda i,j: i+j, x, x))  
['HH', 'ee', 'll', 'll', 'oo', ' ', 'ww', 'oo', 'rr', 'll', 'dd', '...']  
>>> len(x)                 # 字符串长度  
12  
>>> max(x)                # 最大字符  
'w'  
>>> min(x)  
' '
```

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  
  
>>> eval('[1,2,3]')  
[1, 2, 3]
```

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'
```

```
>>> path = 'C:\\Python35\\test.bmp' #修改文件名  
>>> path[:-4] + '_new' + path[-4:]  
'C:\\Python35\\test_new.bmp'
```

第3章 文本处理(一)：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
- 7. 6中英文分词
- 7. 8精彩案例赏析

7.5 字符串常量（了解）

- 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(16)])  
#''.join(random.choices(characters, k=16))  
'zSabpGltJ0X4CCjh'
```

7.6 中英文分词(了解)

```
>>> import jieba          #导入jieba模块
>>> x = '浙江工业大学真漂亮'
>>> jieba.cut(x)          #使用默认词库进行分词
<generator object Tokenizer.cut at 0x000001353B6CA140>
>>> list(_)
['浙江', '工业', '大学', '真', '漂亮']

>>> 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'))    #使用新词库进行分词
['花纸杯']

>>> import snownlp      #导入snownlp模块,Natural Language Processing自然语言处理
>>> snownlp.SnowNLP(x).words
['浙江', '工业', '大学', '真', '漂亮']
```

7.6 中英文分词

s = '一个是阆苑仙葩，一个是美玉无瑕。若说没奇缘，今生偏又遇着他；\n若说有奇缘，如何心事终虚话？一个枉自嗟呀，一个空劳牵挂。一个水中月，一\n个是镜中花。想眼中能有多少泪珠儿，怎禁得秋流到冬，春流到夏！\n'

```
import jieba  
list(jieba.cut(s.replace('\n', '')))
```

```
['一个', '是', '阆', '苑仙葩', '，', '一', '个', '是', '美玉无瑕', '，', '若', '说', '没',  
'奇缘', '，', '今', '生', '偏', '又', '遇', '着', '他', '，', '若', '说', '有', '奇缘',  
'，', '如', '何', '心', '事', '，', '终', '虚', '话', '?', '一', '个', '枉', '自', '嗟', '呀', '，', '一',  
'空', '劳', '牵', '挂', '，', '一', '个', '是', '水', '中', '月', '，', '一', '个', '是', '镜', '中', '花', '，', '  
想', '眼', '中', '能', '有', '多', '少', '泪', '珠', '儿', '，', '怎', '禁', '得', '秋', '流', '到', '冬', '，', '  
春', '流', '到', '夏', '！']
```

```
import snownlp  
snownlp.SnowNLP(s.replace('\n', '')).words  
['一', '个', '是', '阆苑', '仙', '葩', '，', '一', '个', '是', '美', '玉', '无瑕', '，', '若',  
'说', '没', '奇', '缘', '，', '今', '生', '偏', '又', '遇', '着', '他', '，', '若', '说',  
'有', '奇', '缘', '，', '如', '何', '心', '事', '，', '终', '虚', '话', '?', '一', '个', '枉', '自', '嗟', '呀',  
'，', '一', '个', '空', '劳', '牵', '挂', '，', '一', '个', '是', '水', '中', '月', '，', '一', '个', '是', '镜', '中', '花', '，', '  
想', '眼', '中', '能', '有', '多', '少', '泪', '珠', '儿', '，', '怎', '禁', '得', '秋', '流', '到', '冬', '，', '  
禁', '得', '秋', '流', '到', '冬', '，', '春', '流', '到', '夏', '！']
```

7.6 中英文分词

| | | |
|------|---------------|--------------------------|
| 特性 | jieba | snownlp |
| 主要用途 | 中文分词 | 中文 自然语言处理 |
| 核心功能 | 分词、词性标注、关键词提取 | 分词、 情感分析 、文本分类、摘要 |
| 算法基础 | 基于词典 + HMM | 基于贝叶斯等传统机器学习 |
| 分词精度 | | |
| 情感分析 | 无内置 | |
| 速度 | | |
| 自定义性 | | |
| 学习成本 | 低 | 低 |

第3章 文本处理(一)：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
- 7. 6中英文分词
- 7. 8精彩案例赏析

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。（了解，此法过于简单，一般不用）
 - 异或加密的特点：
 1. 可逆性：加密和解密使用相同的算法：明文 \wedge 密钥 = 密文，密文 \wedge 密钥 = 明文
 2. 自反性： $0 \wedge 0 = 1 \wedge 1 = 0$, $0 \wedge 1 = 1 \wedge 0 = 1$, $(A \wedge B) \wedge B = A$ (A 和 B 为任意整数)
 3. 相同密钥：加密和解密使用相同的密钥：(明文 \wedge 密钥) \wedge 密钥 = 明文

```
from itertools import cycle  
cycle(iterable) --> cycle object(迭代器)  
| Return elements from the iterable until it is exhausted.  
| Then repeat the sequence indefinitely.
```

7.8 精彩案例赏析

实现1：

```
def crypt(source, key):
    from itertools import cycle
    result = ''

    temp = cycle(key) # 创建循环迭代器，使密钥可重复使用

    for ch in source:      #谨慎：循环会降低效率，不适合大量字符处理
        result = result + chr(ord(ch) ^ ord(next(temp)))
                           # 对每个字符进行异或运算
    return result
```

7.8 精彩案例赏析

```
source = 'Hello123'  
key = 'key'  
  
encrypted = crypt(source, key)  
decrypted = crypt(encrypted, key)  
print(source[-100:], repr(encrypted[-100:]), decrypted[-100:], sep='\n')
```

对称加密！

7.8 精彩案例赏析

第一步：准备阶段

- source = "Hello123" (长度: 8个字符)
- key = "key" (长度: 3个字符)
- cycle(key) 创建无限循环: 'k' → 'e' → 'y' → 'k' → 'e' → 'y' → ...

第二步：逐字符处理

| 步骤 | 明文字符 | ASCII | 密钥字符 | ASCII | 异或运算 | 结果字符 |
|----|------|-------|------|-------|-----------------------|--------|
| 1 | 'H' | 72 | 'k' | 107 | $72 \wedge 107 = 35$ | '#' |
| 2 | 'e' | 101 | 'e' | 101 | $101 \wedge 101 = 0$ | '\x00' |
| 3 | 'l' | 108 | 'y' | 121 | $108 \wedge 121 = 21$ | '\x15' |
| 4 | 'l' | 108 | 'k' | 107 | $108 \wedge 107 = 7$ | '\x07' |
| 5 | 'o' | 111 | 'e' | 101 | $111 \wedge 101 = 10$ | '\n' |
| 6 | '1' | 49 | 'y' | 121 | $49 \wedge 121 = 72$ | 'H' |
| 7 | '2' | 50 | 'k' | 107 | $50 \wedge 107 = 89$ | 'Y' |
| 8 | '3' | 51 | 'e' | 101 | $51 \wedge 101 = 86$ | 'V' |

第三步：加密结果

- 加密前: "Hello123"
- 加密后: '#\x00\x15\x07\nHYV'

7.8 精彩案例赏析

第四步：验证解密

使用相同的密钥解密：

| 步骤 | 密文字符 | ASCII | 密钥字符 | ASCII | 异或运算 | 结果字符 |
|----|--------|-------|------|-------|-----------------------|------|
| 1 | '#' | 35 | 'k' | 107 | $35 \wedge 107 = 72$ | 'H' |
| 2 | '\x00' | 0 | 'e' | 101 | $0 \wedge 101 = 101$ | 'e' |
| 3 | '\x15' | 21 | 'y' | 121 | $21 \wedge 121 = 108$ | 'l' |
| 4 | '\x07' | 7 | 'k' | 107 | $7 \wedge 107 = 108$ | 'l' |
| 5 | '\n' | 10 | 'e' | 101 | $10 \wedge 101 = 111$ | 'o' |
| 6 | 'H' | 72 | 'y' | 121 | $72 \wedge 121 = 49$ | '1' |
| 7 | 'Y' | 89 | 'k' | 107 | $89 \wedge 107 = 50$ | '2' |
| 8 | 'V' | 86 | 'e' | 101 | $86 \wedge 101 = 51$ | '3' |

解密成功！
得到原始明文："Hello123"

7.8 精彩案例赏析

实现2：（了解）#用字节串处理，无for循环，效率提高

```
def new_crypt(source, key):
    #参数source为字节串，key为字符串

    key = key.encode() # 将字符串密钥转换为字节串

    #使key的字节串与source一样长
    key = key * (len(source)//len(key)) + key[:len(source)%len(key)]
    # 示例：source长8字节，key长3字节 → key扩展为 key*2 + key[0:2]

    # 将source和key转换为16进制字符串，再转为整数进行异或
    result = int(source.hex(), base=16) ^ int(key.hex(), base=16)
    result = hex(result)[2:] # 将结果转换回16进制字符串，去掉'0x'前缀

    #保证16进制数字字符串的长度为偶数，这样才能作为fromhex的参数
    if len(result)%2 == 1:
        result = '0' + result
    return bytes.fromhex(result) # 将16进制字符串转换回字节串
```

7.8 精彩案例赏析

==== 加密过程 ===

1. 原始source: b'Hello123'
2. 原始key: 'key'
3. key编码为字节: b'key'
4. 扩展后的key: b'keykeyke'
5. source的16进制: 48656c6c6f313233
6. key的16进制: 6b65796b65796b65
7. 异或结果(十进制): 945300936690293328
8. 异或结果(16进制): d20c05070b045a46
9. 16进制长度正常: d20c05070b045a46
10. 最终字节结果:
b'\xd2\x0c\x05\x07\x0b\x04ZF'

加密结果: b'\xd2\x0c\x05\x07\x0b\x04ZF'

==== 解密过程 ===

1. 原始source:
b'\xd2\x0c\x05\x07\x0b\x04ZF'
- ...
- 解密结果: b'Hello123'
- 解密文本: Hello123

7.8 精彩案例赏析

#实现2，无循环，可以高效地运行大量字符

```
source = '浙江工业大学是一所非常漂亮的大学'  
key = 'abcdefg'  
  
encrypted = new_crypt(source.encode(), key)  
decrypted = new_crypt(encrypted, key).decode()  
print(source[-100:], encrypted[-100:], decrypted[-100:],  
sep='\n')
```

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。
 - 汉字unicode范围：19968 = < 汉字 <= 40959

```
#三个随机汉字所组成的名字
```

```
>>> ''.join(chr(random.randint(19968, 40959)) for i in range(3))
```

```
#11位随机数字组成的电话号码
```

```
>>> ''.join(str(random.randint(0,9)) for i in range(11))
```

7.8 精彩案例赏析

■ 例7-3 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('a2Cd3e,'))
```

7.8 精彩案例赏析

测试密码分析

密码: 'a2Cd3e,'

包含的字符类型:

数字: '2', '3' → r[0] = True

小写字母: 'a', 'e' → r[1] = True

大写字母: 'C', 'D' → r[2] = True

标点符号: ';' → r[3] = True

r.count(True) = 4 → 返回 'strong'

7.8 精彩案例赏析

■ 使用集合运算实现更简洁：

```
r = [bool(set(pwd) & set(string.digits)),      #交集不为空，则bool为True  
     bool(set(pwd) & set(string.ascii_lowercase)),  
     bool(set(pwd) & set(string.ascii_uppercase)),  
     bool(set(pwd) & set(',.!;?<>'))]
```

#列表推导式，更符合python风格

```
r = [bool(set(pwd) & set(s)) for s in (string.digits,  
                                         string.ascii_lowercase, string.ascii_uppercase, ',.!;?<>')]
```

作业

```
>>> s = """STRAY birds of summer come to my window  
to sing and fly away.  
And yellow leaves of autumn,  
which have no songs,  
flutter and fall there with a sigh.
```

O TROUPE of little vagrants of the world,
leave your footprints in my words.

THE world puts off its mask of vastness to its lover.
It becomes small as one song,
as one kiss of the eternal.

IT is the tears of the earth
that keep her smiles in bloom.

THE mighty desert is burning
for the love of a blade of grass
who shakes her head and laughs
and flies
away.

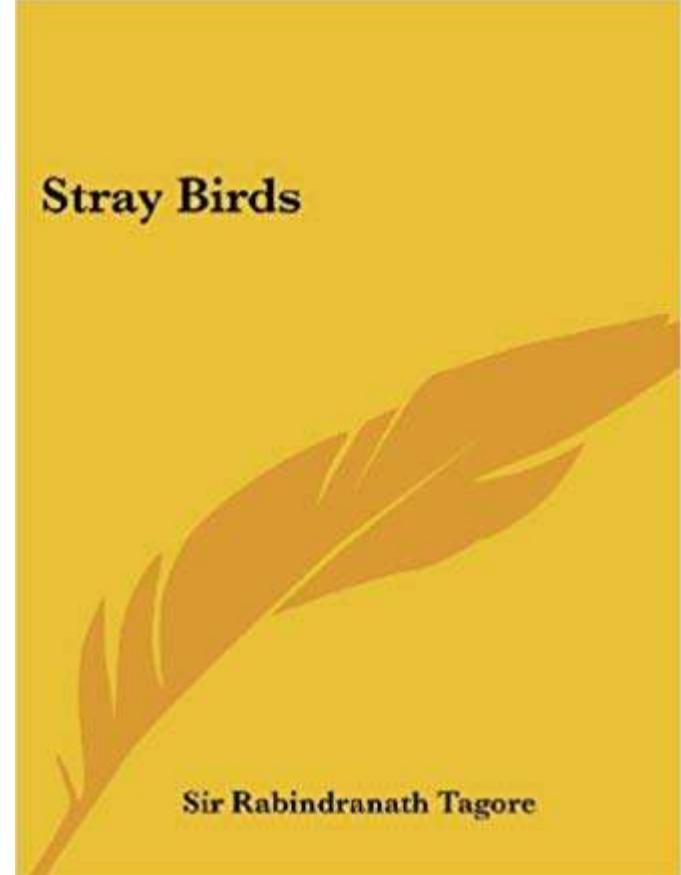
IF you shed tears when you miss the sun,
you also miss the stars.

THE sands in your way beg for your song
and your movement,
dancing water.
Will you carry the burden of their lameness?

HER wistful face haunts my dreams
like the rain at night.

ONCE we dreamt that we were strangers.
We wake up to find that we were dear to each other.

SORROW is hushed into peace in my heart
like the evening among the silent trees."""



作业

1. 设字符串s为上一页的诗句，写下程序语句和运行结果：
 - (1) 计算s中所有单词的数目，以变量n_word表示
 - (2) 计算s中所有非空的行的数目，以变量n_line表示
 - (3) 使用字符串的format()方法输出n_word和n_line：格式要求每个数字占6个格

2. 设字符串s为上一页的诗句，写下程序语句和运行结果：
 - (1) 将s中所有的标点符号(., ?)替换为空字符，然后以/将s中所有非空的行连起来，并输出所生成字符串的前50个字符
 - (2) 将s中所有的标点符号(., ?)替换为空字符，然后以, 将s中所有的单词连起来，并输出所生成字符串的前50个字符