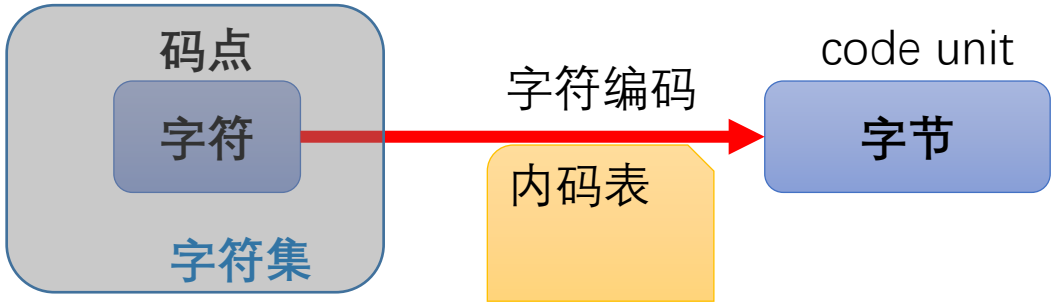
字符串

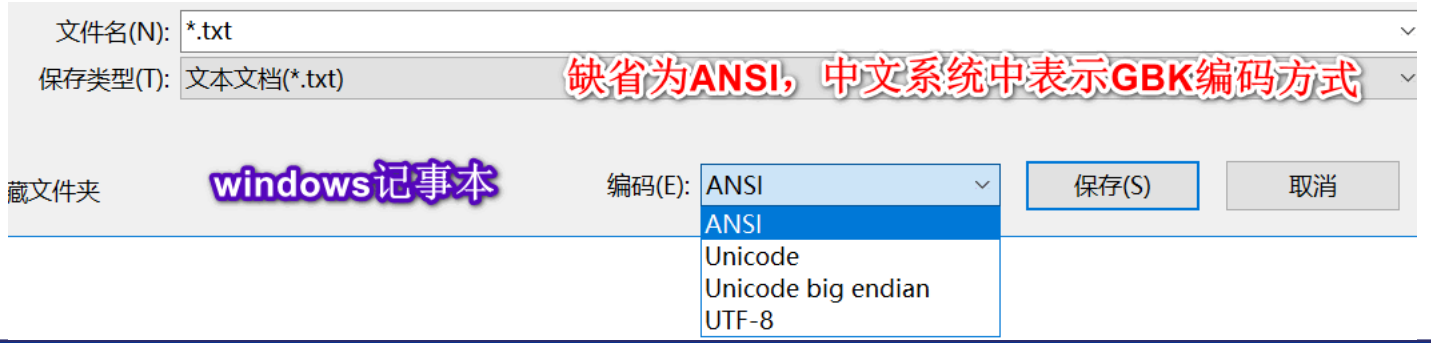
主要内容

- 字符集和编码方式
- 字符串的方法

字符集和字符编码

- 字符集(Charset)：系统支持的多个抽象字符的集合，每个字符在该字符集中可以通过一个称为码点(Code Point)的数字唯一标识
- 字符编码(Encoding)：字符集中的字符在计算机内部存储时如何表示，可映射为1个字节或者多个字节（称为Code Unit)
- **Code Page: 内码表**
  - 最初IBM引入的概念，描述字符与计算机内部存储的字节之间映射的表格
  - 微软/SAP/Apple等通过不同编号的内码表来描述字符集与内部存储之间的映射（即字符编码）
- 可将内码表、字符集和字符编码看成同一实体的不同角度描述
- Python3的字符串中的字符属于Unicode字符集
- 我们通过IDLE等编写的源文件(.py)采用Unicode字符集，其编码方式为UTF-8编码方式





常见字符集

- ASCII**或**US-ASCII：American Standard Code for Information Interchange
  - 128个字符组成，前面32个字符为控制字符，后面的96个字符包括了数字0-9、大写和小写英文字母和标点符号等
  - 其CodePoint(也称为ASCII码)就是字符集中的序号，编码时对应1个字节，保存其相应的CodePoint
- ISO-8859-1(Latin-1)：西欧采用的 8 位字符集，共256个字符
  - 前面128个字符对应着ASCII字符集中的字符，后面的128个字符包括了大部分使用变音符号的拉丁字母语言中的字符
- GB2312：**中国国家标准简体中文字符集，可编码成1或2个字节**
  - **最高位为0时，1个字节对应着ASCII字符集中的字符**
  - 中文使用2个字节表示，每个字节的最高位都为1
- GBK(cp936)：GB2312的扩充，支持更多的汉字，包括繁体汉字等
- GB18030(cp54936)：加入少数民族字符

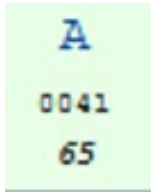
Windows的命令行中可通过chcp命令来查看以及更改code page

ASCII字符集[00, 7F]

code point顺序： .....数字 .... A-Z ... a-z.....

十进制数字/大写英文字母/小写英文字母的ASCII码是连续的

表格中每个字符的表示方法：



字符含义
十六进制的ASCII码
十进制的ASCII码

ASCII (1977/1986)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	NUL 0000 0	SOH 0001 1	STX 0002 2	ETX 0003 3	EOT 0004 4	ENQ 0005 5	ACK 0006 6	BEL 0007 7	BS 0008 8	HT 0009 9	LF 000A 10	VT 000B 11	FF 000C 12	CR 000D 13	SO 000E 14	SI 000F 15
1_	DLE 0010 16	DC1 0011 17	DC2 0012 18	DC3 0013 19	DC4 0014 20	NAK 0015 21	SYN 0016 22	ETB 0017 23	CAN 0018 24	EM 0019 25	SUB 001A 26	ESC 001B 27	FS 001C 28	GS 001D 29	RS 001E 30	US 001F 31
2_	SP 0020 32	! 0021 33	" 0022 34	# 0023 35	$ 0024 36	% 0025 37	& 0026 38	' 0027 39	(0028 40	) 0029 41	* 002A 42	+ 002B 43	, 002C 44	- 002D 45	. 002E 46	/ 002F 47
3_	0 0030 48	1 0031 49	2 0032 50	3 0033 51	4 0034 52	5 0035 53	6 0036 54	7 0037 55	8 0038 56	9 0039 57	: 003A 58	; 003B 59	< 003C 60	= 003D 61	> 003E 62	? 003F 63
4_	@ 0040 64	A 0041 65	B 0042 66	C 0043 67	D 0044 68	E 0045 69	F 0046 70	G 0047 71	H 0048 72	I 0049 73	J 004A 74	K 004B 75	L 004C 76	M 004D 77	N 004E 78	O 004F 79
5_	P 0050 80	Q 0051 81	R 0052 82	S 0053 83	T 0054 84	U 0055 85	V 0056 86	W 0057 87	X 0058 88	Y 0059 89	Z 005A 90	[005B 91	\ 005C 92	] 005D 93	^ 005E 94	_ 005F 95
6_	` 0060 96	a 0061 97	b 0062 98	c 0063 99	d 0064 100	e 0065 101	f 0066 102	g 0067 103	h 0068 104	i 0069 105	j 006A 106	k 006B 107	l 006C 108	m 006D 109	n 006E 110	o 006F 111
7_	p 0070 112	q 0071 113	r 0072 114	s 0073 115	t 0074 116	u 0075 117	v 0076 118	w 0077 119	x 0078 120	y 0079 121	z 007A 122	{ 007B 123	| 007C 124	} 007D 125	~ 007E 126	DEL 007F 127

常见字符集：Unicode

Unicode(统一码): http://www.unicode.org/

- Unicode统一了全球不同语言采用的字符集，通过Unicode Code Point(Unicode码)描述
- 起初Unicode编码为16个比特，1996的Unicode 2.0扩展到21比特：U+0000..U+10FFFF
- ASCII码(0x00-0x7f)和Latin-1(0x80-0xff)是Unicode字符集的子集
- 微软操作系统采用cp65001表示采用UTF-8编码的Unicode字符集
- unicodedata模块给出了访问unicode数据库的一些方法，包括所属的category、name、字符宽度(east_asian_width) 等
- 内置函数ord返回字符对应的unicode。已知unicode，可使用内置函数chr返回对应的字符
- 字符串字面量定义中可以通过\ddd、\xhh、\uxxxx或\Uxxxxxxxx等描述相应码点对应的字符

```
>>> print('八进制(\141) 十六进制(\xf7) 十六进制Unicode(\u7a0b)以及(\U00005e8f)')
八进制(a) 十六进制(÷) 十六进制Unicode(程)以及(序)
```

\ddd	(最多)3位八进制数(ddd)对应的字符	\xhh	2位十六进制数(hh)对应的字符
\uxxxx	4位十六进制数对应的UNICODE字符	\Uxxxxxxxx	8位十六进制数对应的UNICODE字符

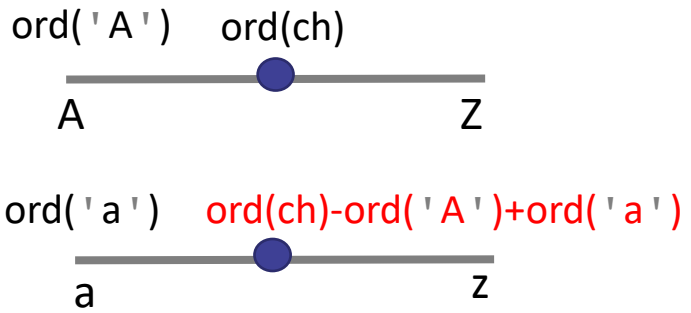
常见字符集：Unicode

- 内置函数ord返回字符对应的unicode。已知unicode，可使用内置函数chr返回对应的字符

```
>>> ord('\t')
9
>>> ord('\n')
10
>>> ord('0')
48
```

```
>>> ord('A')
65
>>> ord('a')
97
>>> chr(48)
'0'
```

```
>>> chr(0x03b1)
'α'
>>> chr(0x03b2)
'β'
>>> chr(0x2200)
'∀'
```




```
if 'a' <= ch <= 'z':
    ch2 = chr(ord(ch) - ord('a') + ord('A'))
else:
    if 'A' <= ch <= 'Z':
        ch2 = chr(ord(ch) - ord('A') + ord('a'))
    else:
        ch2 = ch
```

- Unicode字符可采用UTF-8,UTF-16,UTF-32进行编码,将unicode对应的字符转换成字节序列
- UTF-8: Unicode transformation format, Code page: 65001
  - 目前使用最为广泛的Unicode编码方式。Python3的字符串采用UTF-8编码
  - 编码成1～6个字节。ASCII字符占1个字节，其编码与ASCII编码一致。中文字符被编码成3个字节

字符串字面量定义回顾

- 字符串字面量定义：
  - 单引号和双引号：不能跨越多行

'abc'　"What's your name?"　'"Thank you!" she said. '

  - 三单引号和三双引号：用于长字符串
    - 可以跨越多行，其余与短字符串一致
  - 两个以上连续（中间可用空格等分隔）的字符串字面量被合并为一个字符串

```
>>> a ='''first line \nline 2
line 3
last line.'''
>>> a
'first line \nline 2\nline 3\nlast line.'
>>> print(a)
first line
line 2
line 3
last line.
```

- 字符转义：转义字符(\)表示后面的字符不是原有的含义而是要特殊处理)，如\' \n \t　\141　\x09 \u7a0b　\U00005e8f
  - 如果转义字符后面的字符并没有特殊的含义，则转义字符仍然保留，如'\c'

```
a = '2233' '456'
a = '2233456'
```

- 原始字符串：在字符串界定符前面加字母r或R
  - 可以是短字符串或长字符串(三引号形式)的原始字符串形式
  - 原始字符串中的字符都为字面上的含义，不再需要进行转义（特殊情况除外）
  - 原始字符串的最后一个字符不能是\，因为增加转义字符确定了原始字符串的开始和结束位置后，转义字符又失去了原来的意义

```
>>> print("one\ttwo\n\141b\x09\u7a0b\U00005e8f")
one	two
ab	程序
```

```
>>> print(r'C:\Program Files (x86)\Python35\Lib')
C:\Program Files (x86)\Python35\Lib
```

中文字符串格式化：中文字符占2个英文字符的宽度

format_eastasia.py

```
import unicodedata
def width_east_str(text):
    """ 返回text等价的英文字符的宽度 """
    t = [ 2 if unicodedata.east_asian_width(ch) in
'WFA' else 1 for ch in text ]
    return sum(t)
```

```
def format_east_str(text, min_width=10, justify=None,
fillchar=' ' ):
    width = width_east_str(text)
    if width >= min_width: return text
    fill_width = min_width - width
    if justify == '>' :  # right justified
        return fill_width * fillchar + text
    elif justify == None or justify == '<':
        return  text + fill_width * fillchar
    else:  # center
        return (fill_width+1) // 2 * fillchar + text
+ fill_width // 2 * fillchar
```

```
format_east_str('中国12上海', 20, '^', '*')
```

*****中国12上海*****

```
# 中文unicode码 [0x4e00, 0x9fa5]
>>> 0x4e00 <= ord('中') <= 0x9fa5
True
>>> 0x4e00 <= ord('a') <= 0x9fa5
False
```

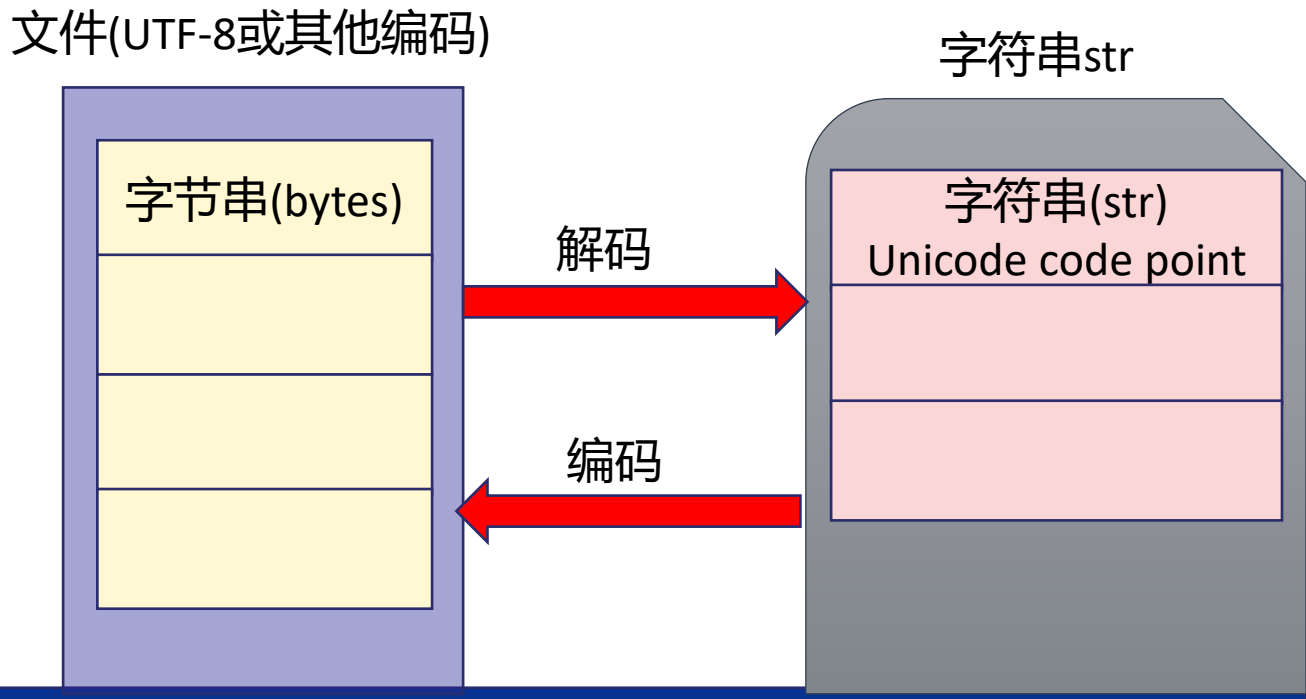
east_asia_width	意义
A: Ambiguous	不确定
F: Fullwidth	全宽
H:Halfwidth	半宽
N: Neutral	中性
Na: Narrow	窄
W: Wide	宽

format_east_str函数按照相应的格式格式化字符串text

- 最小宽度缺省为10，
- 对齐方式缺省为左对齐，可取的值 '<', '>', '^', None
- 填充字符缺省为空格

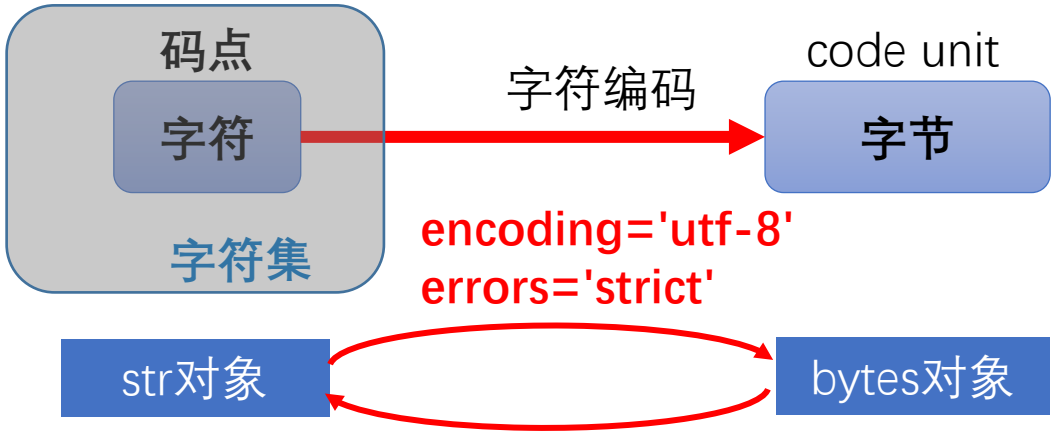
字符串编码和解码

- 保存在外部存储(硬盘/U盘等）中的数据以字节方式保存
- 我们通过IDLE等编写的源文件(.py)，其编码方式缺省为UTF-8编码。
  - 编辑器在加载文件时，基于UTF-8编码方式，将外部存储的字节流解释为Unicode字符集中的字符串str
  - 编辑器在保存文件时，基于UTF-8编码方式，将字符串编码成字节流后保存到外部存储
- **类型bytes**类似于str，也是一个**不可变**的有序序列类型，只是**其元素为单个字节**，而str中每个元素为单个字符
  - 字面量定义：和str类似，只是在之前添加**b或者B**即可，比如b'abcd'
- 类型bytearray与bytes类似，只是其为可变序列类型



unicode字符串str与字节串bytes之间的转换

- encoding参数给出编码方式，缺省为utf-8，常用的编码还包括ascii, latin1, gbk等
- errors参数指出在编码/解码过程中无法转换时怎么办？缺省‘strict’，即抛出异常。比如某些字符不属于相应字符集中的字符，或者某些字节不是合法的字节序列



含义	方法	例子
str→bytes	str_obj.encode(encoding='utf-8', errors='strict') str_obj.encode(encoding='ascii',errors='replace') **bytes(strobj, encoding='utf-8', errors='strict')**	**'C语言'.encode('gbk')** 采用GBK编码方式将字符串编码成字节串 **bytes('C语言', encoding='ascii', errors='replace')** 字符串转换成ASCII码，字符不能转换时用?代替
bytes→str	bytesobj.decode(encoding='utf-8', errors='strict) **str(bytesobj, encoding, errors='strict')** 至少有两个参数，否则str(bytesobj)表示将bytesobj以友好方式转换为字符串	**'C语言'.encode('gbk').decode('gbk')** **str('C语言'.encode('gbk'), 'gbk')**

编码：unicode字符串→字节串

errors参数指出在编码过程中由于某些字符不属于相应字符集中的字符从而无法转换时怎么办

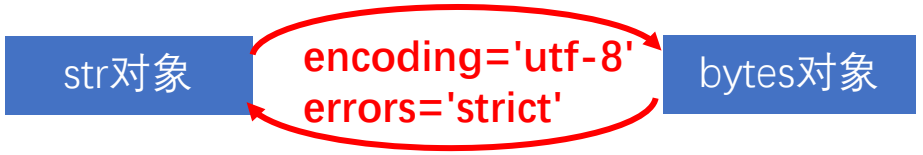
- strict(缺省):抛出异常UnicodeEncodeError
- ignore：忽略并跳过无法编码的字符
- replace:无法编码时替换为缺省替换字符，一般为?
- backslashreplace:无法编码时替换为通过\进行转义的转义序列
- xmlcharrefreplace：无法编码时替换为XML表示方式
- namereplace: 替换为用名字描述的Unicode字符特性\N{...}

str_encoding.py

```
def test_unicode_encode():
    s1 = 'I love Python语言!'
    try:
        print(s1.encode('ascii',errors='strict'))
    except Exception as e:
        print(e)
    for error in ('ignore', 'replace',
'backslashreplace', 'xmlcharrefreplace',
'namereplace'):
        print("encoding with errors '%s'" % error)
        print(s1.encode('ascii', errors=error))
```

```
<class 'UnicodeEncodeError'> 'ascii' codec
can't encode characters in position 13-14:
ordinal not in range(128)

encoding with errors 'ignore'
b'I love Python!'

encoding with errors 'replace'
b'I love Python??!'

encoding with errors 'backslashreplace'
b'I love Python\\u8bed\\u8a00!'

encoding with errors 'xmlcharrefreplace'
b'I love Python&#35821;&#35328;!'

encoding with errors 'namereplace'
b'I love Python\\N{CJK UNIFIED IDEOGRAPH-
8BED}\\N{CJK UNIFIED IDEOGRAPH-8A00}!'
```

编码：字节串→unicode字符串



bytes_obj.decode(encoding='utf-8', errors='strict') -> str

str(bytes_or_buffer[, encoding='utf-8' [, errors='strict']]) -> str

- 解码时也有可能出现无法解码的情况， errors可取值'strict','ignore','replace', 'backslashreplace'等
- errors取值'strict'，解码出错时抛出异常 UnicodeDecodeError

```
>>> s1 = 'Python程序设计'
>>> bs1 = s1.encode() # 转换为字节串bs1,utf-8编码方式
>>> print(bs1)
b'Python\xe7\xa8\x8b\xe5\xba\x8f\xe8\xae\xbe\xe8\xae\xa1'
>>> s2 = str(bs1)   # 将字节串转换为友好的字符串(知道其是一个bytes对象)
>>> print(s2)
b'Python\xe7\xa8\x8b\xe5\xba\x8f\xe8\xae\xbe\xe8\xae\xa1'
>>> s3 = str(bs1,'utf-8')   #或者 bs1.decode()
>>> print(s3)
Python程序设计
>>> s1.encode('gbk').decode('utf-8')
Traceback (most recent call last):...
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xb3 in position 6: invalid start byte
```

locale模块和sys模块

- locale描述了本地化设置，**语言_区域.字符集**，表示在指定区域说某种语言的人的一些基本的习惯(时间、数字、字符类型和排序等)，比如zh_CN.UTF-8，en_US.UTF-8等

```
>>> import locale
>>> locale.getdefaultlocale()        # 获得本地化设置，(语言_区域，  编码方式)
('zh_CN', 'cp936')
>>> locale.getpreferredencoding()  # 获得用户打开文本文件缺省的编码方式，  cp936=GBK
'cp936'
```

- sys模块：解释器相关的属性和方法

```
>>> import
>>> sys.getdefaultencoding()     # 解释器所采用的缺省字符串编码方式，python3一定是'utf-8'
'utf-8'
>>> sys.getfilesystemencoding()    # 操作系统文件名与unicode文件名之间转换采用的编码方式
# 给出了如何将str文件名转换为bytes文件名，以进行进一步的系统调用
'utf-8'
# 每个系统有可能返回不同的值，大部分情况下会返回'utf-8'
```

主要内容

- 字符集和编码方式
- 字符串的方法

字符串相关内置函数

转换为字符串

转换为数字类型

函数	含义	实例	结果
bin(x)	把数字x转换为二进制字符串	bin(4)	'0b100'
oct(x)	把数字x转换为八进制字符串	oct(25)	'0o31'
hex(x)	把数字x转换为十六进制字符串	hex(75)	'0x4b'
str(obj)	把对象obj转换为字符串	str(16), str('程序')	('16', '程序')
repr(obj)	把对象obj转换为内部表示的字符串	repr(16), repr('程序')	('16', "'程序'")
ascii(obj)	把对象obj转换为内部表示的ascii字符串	ascii('程序')	"'\\u7a0b\\u5e8f'"
int(x[,d])	把数字x截取为整数，或将基数为d的数字型字符串x转换为整数，如果d没有缺省为十进制	int(3.14), int('314') int('ff',16)	(3,314) 255
float(x)	将对象(数字或字符串)x转换为浮点数	float(3),float('3.14') float('12e-2')	(3.0,3.14) 0.12
len(obj)	返回序列类对象obj包含的元素个数	len('Hello World'),len('')	(11, 0)
ord(s)	返回长度为1的字符串s中的字符对应的Unicode码（ASCII码是其子集）	ord('a'),ord('A'),ord('你') ord('0'), ord('\n')	(97, 65, 20320) (48, 10)
chr(x)	返回Unicode编码为x的字符串(len=1)	chr(65),chr(10),chr(20320) chr(ord('D')-ord('A')+ord('a'))	('A', '\n', '你') 'd'

字符串作为有序序列和iterble对象支持的方法

字符串是不可变的有序对象类型。

- 作为有序序列对象：
  - 下标和切片访问
  - 连接运算 + 、重复运算 *
  - 比较运算
  - 求字符串长度 len
  - 有序对象支持的通用方法
    - 成员关系：in 和not in
    - 元素出现情况：index和count方法
    - 判断**连续多个字符（子串）**出现在原有字符串中
    - 字符串还引入了rindex、find、rfind方法
- 作为iterable对象，当然也支持max/min/sorted/reversed/enumerate/zip等内置函数

字符串的方法 strobj.method(…)

dir(str)

描述	方法
去首尾的空白字符	strip([chars]) lstrip([chars]) rstrip([chars])
字符串类型判断	isalnum() isalpha() isdecimal() isascii() isdigit() isidentifier() islower() isupper() isnumeric() isprintable() isspace() istitle()
大小写转换	lower() upper() swapcase() capitalize() title() casefold()
测试和查找	• startswith(prefix[,start[,end]]) endswith(suffix[,start[,end]]) • index(sub[,start[,end]]) rindex(sub[,start[,end]]) find(sub[,start[,end]]) rfind(sub[,start[,end]]) count(sub[,start[,end]])
替换	replace(old,new[,count])
拆分和组合	• split(sep=None,maxsplit=-1) rsplit(sep=None, maxsplit=-1) partition(sep) rpartition(sep) splitlines([keepends]) • join(iterable)
填充和对齐	zfill(width) center(width[,fillchar]) ljust(width[,fillchar]) rjust(width[,fillchar]) expandtabs([tabsize])
翻译和转换	maketrans translate

字符串填充和对齐示例：图案问题

triangle.py

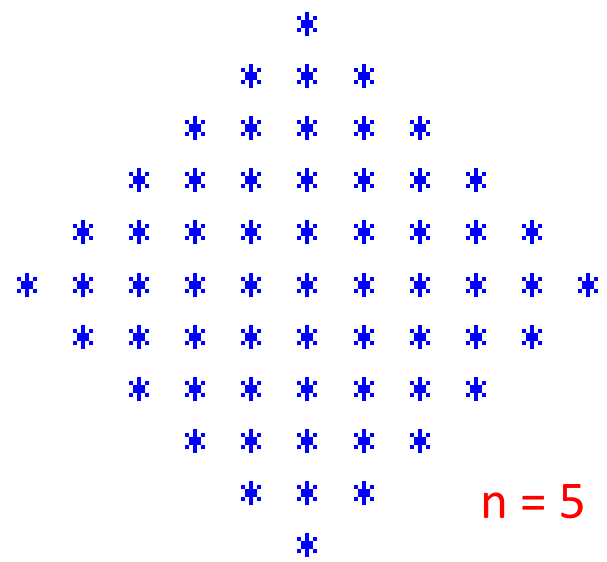
```
total = int(input('请输入n:'))
for i in range(total):
    for j in range(i + 1):
        print('*', end='')
    print()
```

```
for i in range(total):
    stars = '*' * (i + 1)
    print(stars)
```

```
for i in range(0, total):
    stars = ' '.join('*' * (i + 1))
    print(stars)
```

```
# n stars + n-1 spaces
for i in range(total):
    stars = ' '.join('*' * (i + 1))
    line = stars.rjust(total * 2 - 1, ' ')
#   line = '{: >{}}'.format(stars, lines * 2 - 1)
    print(line)
```

```
*                   *                                   *
**                  * *                               * *
***                 * * *                           * * *
****                * * * *                       * * * *
*****               * * * * *                   * * * * *
```



用户输入一个正整数n，
输出菱形图案

字符串的方法：去首尾的（空白）字符

`strip([chars])`　`lstrip([chars])`　`rstrip([chars])`

- 不传递参数，等价于chars为’ \t\n\r\x0b\x0c’。将字符串的两端、左端或右端的**空格类字符**(包括回车\r、换行\n、空格、制表\t、\v和\f）去掉，返回一个新的字符串
- 如果指定chars，则去掉字符串两端、左端或右端开始处chars中的那些字符

```
>>> s=″ abc  ″
>>> s1=s.strip( )
>>> s1
″abc″
>>> s2 = s.lstrip( )
>>> s2
″abc  ″
>>> s3 = s.rstrip( )
>>> s3
″ abc″
>>> '\t  line 1 \t  \n'.strip()
'line 1'
```

```
>>> 'aaaassddfaaa'.strip('asf')
'dd'
>>> 'aaaassddfaaa'.lstrip('a')
'ssddfaaa'
>>> 'aaaassddfaaa'.rstrip('a')
'aaaassddf'
```

```
>>> '{1456};'.strip('{};')
'1456'
>>> from string import punctuation
>>> '"<pyshell#110>"'.strip(punctuation)
'pyshell#110'
```

字符串的方法：类型判断，根据Unicode字符集中的类型判断

string模块中的属性给出了ASCII字符集中的字母、数字、空格等

isalpha()	字符串的字符是否**都为**字母
isdigit()	字符串的字符是否都为十进制数字
isalnum()	字符串的字符是否都为字母或者numeric
isidentifier():	字符串是否为合法的python标识符
islower() isupper()	字符串的那些**可区分大小写**的字符是否都为小写或者大写字母
isprintable()	字符串的字符是否都为可打印字符
isspace()	字符串的字符是否都为空格类字符（空格、制表符\t、回车\n等）
istitle()	字符串中的每个单词的首字母是否为大写

```
>>> ''.isalpha()
False
>>> 'abcd'.isalpha()
True
>>> '1234'.isalpha()
False
>>> '一千三百'.isalpha()
True
>>> '1240'.isdigit()
True
>>> '一千三百'.isdigit()
False
```

- 要求所有的字符都满足指定的特性，而不是仅仅其中某个字符满足
- is类方法一般要求至少有一个属于那一类型的字符

```
>>> ''.isspace()
False
```

```
>>> '1234abcDE'.isalnum()
True
>>> '中国一千三百'.isidentifier()
True
>>> '1234abcDE'.isidentifier()
False
>>> '1234abcDE'.islower()
False
```

```
>>> '1234abcde'.islower()
True
>>> 'ABCDE'.isupper()
True
>>> ' \t\n'.isspace()
True
>>> 'How Are You?'.istitle()
True
```

字符串的方法：分割 split和rsplit

`split(sep=None, maxsplit=-1)`　　`rsplit(sep=None, maxsplit=-1)`

- 以指定字符串sep为分隔符分割字符串，返回一个字符串列表，其元素为分割出来的字符串(不包括分隔符)。如果字符串中sep连续出现，这些sep之间会分割出空字符串
- sep一般为单个字符，但也可以是多个字符
- 如果maxsplit给出且大于-1，则最多只进行maxsplit次分割，即列表元素个数最多为maxsplit+1。如果小于等于-1，则不限次数
- rsplit函数表示从右边开始进行分割，但结果仍然按照出现在原字符串的顺序排列

```
>>> s="  apple , peach , banana,,peach ,pear ,"
>>> s.split(',')
['  apple ', ' peach ', ' banana', '', 'peach ', 'pear ', '']
>>> [i.strip() for i in s.split(',')]
['apple', 'peach', 'banana', '', 'peach', 'pear', '']
>>> [i.strip() for i in s.split(',') if i != '']
['apple', 'peach', 'banana', 'peach', 'pear']
>>> s.split(',',2)
['  apple ', ' peach ', ' banana,,peach ,pear ,']
>>> s.split('peach')
['  apple , ', ' , banana,,', ' ,pear ,']
```

```
>>> t = "2014-10-31"
>>> t.split('-')
['2014', '10', '31']
>>> [int(c) for c in t.split('-')]
[2014, 10, 31]
>>> t.rsplit('-')
['2014', '10', '31']
>>> t.rsplit('-',1)
['2014-10', '31']
```

字符串的方法：分割 split和rsplit

`split(sep=None, maxsplit=-1)`　　`rsplit(sep=None, maxsplit=-1)`

- 以指定字符串sep为分隔符分割字符串，返回一个字符串列表，其元素为分割出来的字符串(不包括分隔符)。如果字符串中sep连续出现，这些sep之间会分割出空字符串
- sep一般为单个字符，但也可以是多个字符
- **如果**sep==None：
  - 所有空白类字符（包括空格、换行符、制表符等等）都将被认为是分隔符
  - 多个连续的空白类字符当成一个整体，即相当于1个空白分隔符。中间不会分割出空字符串
  - 字符串最前面和最后面分割出来的空字符串会被移走
  - **所有空字符串会被移走，每个字符串首尾都没有空格类字符**

```
>>> s2 = '\nHello world \n\n My  name is Mike   '
>>> s2.split()
['Hello', 'world', 'My', 'name', 'is', 'Mike']
>>> s2.split(' ')
['\nHello', 'world', '\n\n', 'My', '', 'name', 'is', 'Mike', '', '', '']
```

字符串的方法：分割 splitlines

splitlines([keepends=False])

- 根据换行类字符(包括'\n')将字符串分割成多个子字符串，返回这些**子字符串的列表**。字符串最后为换行字符时，在该分隔符之后不会出现空字符串
- 空字符串调用本函数时返回空列表
- keepends=True表示保留分隔符(换行字符)

	splitlines()	split('\n')
空字符串	返回空列表	返回包含空字符串的列表
最后为\n	该分隔符\n后面没有空字符串	该分隔符后面会分割出一个空字符串
keepends	可保留换行字符	换行符不保留

```
>>> 'line1\rline2\nline3\r\nline4\n\nline6\n'.splitlines()
['line1', 'line2', 'line3', 'line4', '', 'line6']
>>> 'line1\rline2\nline3\r\nline4\n\nline6\n'.split('\n')
['line1\rline2', 'line3\r', 'line4', '', 'line6', '']
```

```
>>> s3 = '\nab c\n\nde fgkl\n'
>>> print(s3, end='')

ab c

de fgkl
>>> s3.split('\n')
['', 'ab c', '', 'de fgkl', '']
>>> ''.split('\n')
['']
>>> s3.splitlines()
['', 'ab c', '', 'de fgkl']
>>> s3.splitlines(True)
['\n', 'ab c\n', '\n', 'de fgkl\n']
>>> ''.splitlines()
[]
```

字符串的方法：join 组合多个字符串

strobj.join(iterable)

- 将iterable对象(列表,字符串等)中的多个元素组合在一起，元素间插入相应的**分割字符串strobj**，返回新的字符串
- iterable对象中的**元素必须是字符串**

```
>>> s = list('abc')
>>> s
['a', 'b', 'c']
>>> ''.join(s)
'abc'
>>> ','.join(s)
'a,b,c'
>>> print('\n'.join(s))
a
b
c
>>> list2 = list(range(5))
>>> list2
[0, 1, 2, 3, 4]
>>> '+'.join([str(i) for i in list2])
'0+1+2+3+4'
```

字符串的方法：大小写转换

lower()	返回小写格式的字符串
upper()	返回大写格式的字符串
capitalize()	返回字符串中**首字母大写**，其他字母小写
title()	返回字符串中**每个单词**首字母大写，其他小写
swapcase()	返回字符串中大小写互换

```
>>> s="What is YoUr NaMe?"
>>> s2 = s.lower()
>>> s2
'what is your name?'
>>> s
'What is YoUr NaMe?'
>>> s.upper()
'WHAT IS YOUR NAME?'
>>> s.capitalize()
'What is your name?'
>>> s.title()
'What Is Your Name?'
>>> s.swapcase()
'wHAT IS yOuR nAmE?'
```

自己实现大小写英文字母转换swapcase

swapcase.py

```
def swapcase(text):
    text2 = []
    for ch in text:
        if 'a' <= ch <= 'z':
            text2.append(chr(ord(ch) - ord('a') + ord('A')))
        elif 'A' <= ch <= 'Z':
            text2.append(chr(ord(ch) - ord('A') + ord('a')))
        else:
            text2.append(ch)
    return ''.join(text2)
```

a	b	c	...	x	y	z
A	B	C		X	Y	Z
0	1	2	...	23	24	25

如何实现凯撒密码？一个字母替代以偏移n个位置的另一个字母

字符串的方法：replace 替换

replace(old,new[,count])

- 返回新的字符串，原始字符串中的所有子串old被替换为new 。如果count给出，则仅替换前面的count个子串

```
>>> s = 'apple, peach, banana, peach, pear'
>>> s.replace('peach', 'PEACH')
'apple, PEACH, banana, PEACH, pear'
>>> s.replace('peach', 'PEACH', 1)
'apple, PEACH, banana, peach, pear'
>>> s.replace(',', ' ')
'apple  peach  banana  peach  pear'
>>> s.replace(',', ' ').split()
['apple', 'peach', 'banana', 'peach', 'pear']
```

```
>>> filename = 'C:/Program Files/Microsoft Office/Office16/winword.exe'
>>> filename.replace('/', '\\')
'C:\\Program Files\\Microsoft Office\\Office16\\winword.exe'
```

字符串的方法：startswith/endswith

startswith(prefix[,start[,end]])　　endswith(suffix[,start[,end]])

判断字符串是否以prefix开头或suffix结尾

- prefix和suffix可以是字符串

```
startswith:   strobj[:len(prefix)] == prefix
endswith:     strobj[-len(sufffix):] == suffix
```

- **也可以是一个元组**，其每个元素为一个字符串，表示是否其中任意一个字符串开头或结尾

```
>>> import os
>>> [filename for filename in os.listdir(r'c:\windows' ) if filename.endswith(('.exe','.bat'))]
['bfsvc.exe', 'evim.bat', 'explorer.exe', 'fveupdate.exe', 'gview.bat', 'gvim.bat', 'gvimdiff.bat',
'HelpPane.exe', 'hh.exe', 'notepad.exe',...]
```

```
def _get_normal_name(orig_enc):
    # Only care about the first 12 characters.
    enc = orig_enc[:12].lower().replace("_", "-")
    if enc == "utf-8" or enc.startswith("utf-8-"):
        return "utf-8"
    if enc in ("latin-1", "iso-8859-1", "iso-latin-1") or \
       enc.startswith(("latin-1-", "iso-8859-1-", "iso-latin-1-")):
        return "iso-8859-1"
    return orig_enc
```

str_find.py

orig_enc编码方式可能不是很标准

- 有的使用大写
- 有的使用下划线而不是减号
- 如果以utf-8-开始，则等价于utf-8
- 如果以latin-1等开始时，等价于iso-8859-1

字符串的元素访问和计数

```
>>> 'pppppp'.count('pp')
3
```

- 字符串是有序序列，可以通过下标访问指定位置的元素(即由对应位置的字符组成的字符串），可支持in, count, index，只是其参数除了是单个字符组成的字符串外，还可是长度超过1的**子字符串**
- strobj.index(sub[,start[,end]]) 返回子串sub(而不仅仅是单个字符)在指定范围内**首次出现**的位置（下标），**如果不存在则抛异常ValueError**
- strobj.count(sub[,start[,end]]) 返回子串sub出现的次数，找到子串后，在之后的下一个位置继续...
- sub in S　sub not in S 判断子串sub是否在S中出现
- rindex(sub[,start[,end]]) 返回子串sub在指定范围内**最后一次**出现的位置，**如果不存在则ValueError**
- find(sub[,start[,end]]) rfind(sub[,start[,end]]) 与index/rindex类似，返回子串sub在指定范围内首次或最后一次出现的位置，**如果不存在则返回-1**

```
>>> s = "apple,peach,banana,peach,pear"
>>> s.index('peach')
6
>>> s.index('ppp')
Traceback (most recent call last):
...
ValueError: substring not found
>>> s.rindex('peach')
19
>>> s.count('an')
2
```

```
>>> s.find("peach")
6
>>> s.find("peach", 6 + 5)
19
>>> s.find("peach", 11, 24)
19
>>> s.rfind('p')
25
>>> s.rfind('ppp')
-1
```

字符串的方法：成员判断和查找示例

import os
os.path.split(filename)

str_find.py

```
text = '''key1=value1
key2 = value2
key3 = value3=
  '''

lines = text.count('\n') + (text[-1] != '\n')
```

 # 统计字符串的行数
len(text.splitlines())
3 + True 3 + False

```
# 分析配置文件
def parse_config(text):
    for line in text.splitlines():
        if '=' in line:
            i = line.index('=')
            key, value = line[:i], line[i+1:]
            print('key={} value={}'.format(
              key.strip(), value.strip()))
```

```
def split_filename(filename=__file__, sep='/'):
    if sep not in filename:
        return '', filename
    i = filename.rindex(sep)
    dirname, basename = filename[:i], filename[i+1:]
    if dirname != sep * len(dirname):
        dirname = dirname.rstrip(sep)
    return dirname, basename
```

```
def split_filename1(filename=__file__, sep='/'):
    i = filename.rfind(sep)
    if i == -1:
        return '', filename
    dirname, basename = filename[:i], filename[i+1:]
    if dirname != sep * len(dirname):
        dirname = dirname.rstrip(sep)
    return dirname, basename
```

sep不在文件名中出现时，rfind返回-1

```
filename = r'C:\Program Files\Microsoft Office\Office16\winword.exe'
split_filename(filename, '\\')
# C:\Program Files\Microsoft Office\Office16, winword.exe
```

字符串的方法： partition rpartition分割一次

partition(sep)　　rpartition(sep)

- 以指定字符串sep为分隔符**分割最多一次**，将原字符串分割为包括3部分的元组，即sep前的字符串、sep、sep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
- rpartition表示从右边开始分割，但是返回的顺序仍然是原字符串中的顺序

```
>>> s = "  apple , peach , banana,,peach ,pear ,"
>>> s.partition(',')
('  apple ', ',', ' peach , banana,,peach ,pear ,')
>>> s.rpartition(',')
('apple , peach , banana,,peach ,pear ', ',', '')
>>> s.partition('\n')
('  apple , peach , banana,,peach ,pear ,', '', '')
```

```
text = '''key1=value1
key2 = value2
key3 = value3=
'''
```

str_find.py

```
def parse_config2(text):
    for line in text.splitlines():
        if '=' in line:
            key, _, value = line.partition('=')
            print('key={} value={}'.format(key.strip(), value.strip()))
```

字符串的方法： maketrans/translate 并行转换

- replace方法只能串行替代，而不能并行替代！ `s.replace(',', ' ').replace(';', ' ')`
- 将字符串中的某些字符并行替代为另外一些字符或删除
  - 首先maketrans构造映射表，然后translate进行实际的替换
- 静态方法 `maketrans(x,y[,z])` 如果有两个以上参数，x和y必须长度一致，每一个在x出现的字符转换为y对应位置的字符，如果有第三个参数，表示在z中出现的字符转换为None，即删除
- `strobj.translate(table)` 按照映射表对strobj进行转换，返回转换后的新字符串

```
>>> s = 'Great hopes make great man.'
>>> tab = str.maketrans('abcde', 'uvwxy')
>>> ss = s.translate(tab)
>>> ss
'Gryut hopys muky gryut mun.'
>>> tab2 = str.maketrans('uvwxy', 'abcde')
>>> ss.translate(tab2)
'Great hopes make great man.'
>>> tab3 = str.maketrans('abcde','uvwxy', '.?')
>>> s.translate(tab3)
'Gryut hopys muky gryut mun'
```

```
>>> s1 = '我的银行账号为3141516,密码为123456'
>>> tab1 = str.maketrans('账号密码','九八姑婆
','123456789')
>>> s11 = s1.translate(tab1)
>>> s11
'我的银行九八为,姑婆为'
>>> tab2 = str.maketrans('123456789', '*' * 9)
>>> s1.translate(tab2)
'我的银行账号为*******,密码为******'
```

字符串的方法： maketrans/translate 并行转换

- maketrans创建的映射表，实际上就是1个dict，该字典的key为字符的unicode(整数)，而value为
  - 如果为整数，则是要替换为的字符的Unicode
  - 如果为None，则表示删除
  - 如果为字符串，则原来的字符替换为相应字符串

```
>>> str.maketrans('abcde','uvwxy', '.?')
{97: 117, 98: 118, 99: 119, 100: 120,
101: 121, 46: None, 63: None}
```

- maketrans(mapping)，只传递一个字典参数，将某个字符映射为一个字符或多个字符或None
  - 该字典的key应该为一个单个字符ch或者ord(ch)返回的整数(即unicode码点)
  - 值可以是：
    - None，表示删除该字符
    - 一个字符串，表示替代为该字符串
    - 整数n，表示替换为chr(n)

```
>>> s2 = 'dlmao@fudan.edu.cn'
>>> tab3 = str.maketrans({'@':' AT ', '.':' DOT '})
>>> s2.translate(tab3)
'dlmao AT fudan DOT edu DOT cn'
```

字符串应用示例：统计文章的单词个数

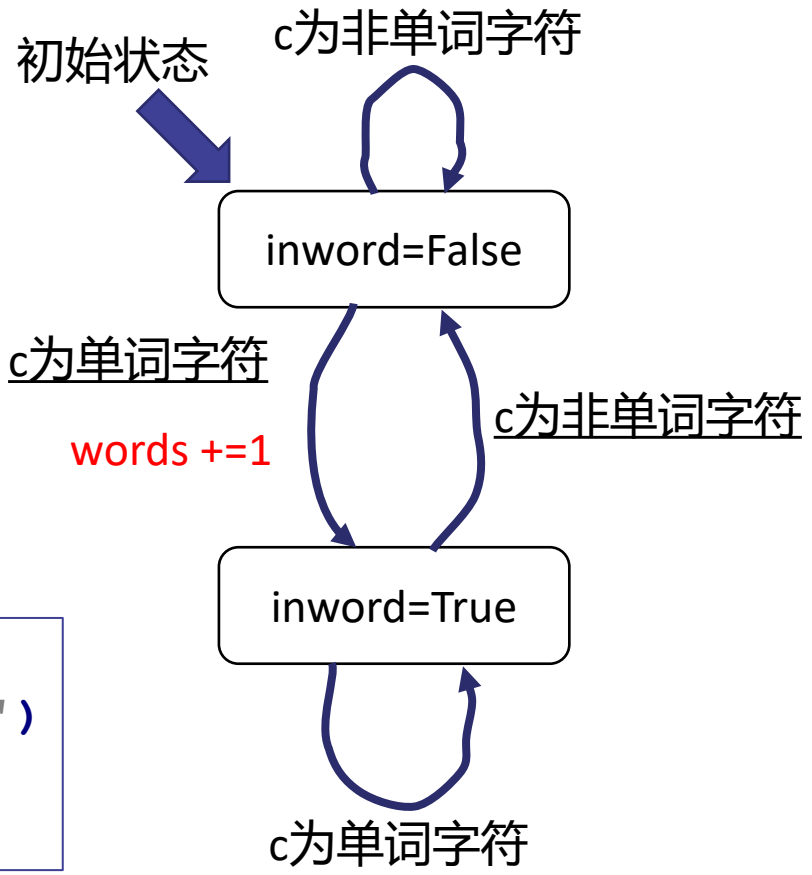
单词的定义：单词由英文字母组成，其他字符只是用来分隔单词

```
import string
def count_words(sentence='If winter comes, can spring be far behind?'):
    words = 0
    inword = False
    for c in sentence:
        if inword:
            if c not in string.ascii_letters:
                inword = False
        else:
            if c in string.ascii_letters:
                inword = True
                words += 1
    return words
```

count_words.py

```
def test_count_words():
    s = input('请输入多个英文单词，可以用任何非英文字母分割单词\n')
    words = count_words(s)
    print('英文单词个数:%d' % words)
```

- 单词的开始：刚开始或前一个字符为非单词字符，当前字符为单词字符
- 单词中间：前一个字符为单词字符，当前字符为单词字符
- 单词的结束：前一个字符为单词字符，当前字符为非单词字符或者结束



参考编程题1

- 检查两个单词是否为相似词。两个单词如果包含相同的字母，则它们是相似词。例如：silent和listen是相似词。
- 图为程序分三次执行，每次输入及输出，蓝色为程序输出，黑色为用户输入。

```
请输入两个单词，以空格分隔:listen  silent
单词listen与单词silent是相似词！
```

```
请输入两个单词，以空格分隔:banana  anbaan
单词banana与单词anbaan是相似词！
```

```
请输入两个单词，以空格分隔:listen  silenta
单词listen与单词silenta不是相似词！
```

参考编程题2

- 用户随意输入一串字符，程序依次在相邻位置插入一个空格，逐行输出。
- 如右图，黑色为用户输入，蓝色为程序输出：

```
>>>
请输入：University
Universit y
Universi ty
Univers ity
Univer sity
Unive rsity
Univ ersity
Uni versity
Un iversity
U niversity
-----
U niversity
Un iversity
Uni versity
Univ ersity
Unive rsity
Univer sity
Univers ity
Universi ty
Universit y
>>>
```