字节串

产生背景

计算机底层都是0或1，并不能保存字符，但是程序需要识别/保存各种字符。于是大佬们就想了一个办法，为每个字符编个号，当程序保存字符的时候，实际保存的是该字符的编号；当程序读取字符时，也是读取的编号，然后，通过编号去查一下码表(比如：utf-8)得到实际的字符

字符集

为了将字符和字节关联起来，美国人整了一套码表，将咱们键盘上能敲出来的字符都跟字节一一对应，结果发现一共就100来个字符，1个字节（8位，支持256个字符编号）就能搞定。这就是ASCII字符集

其他字符集

但是，美国并没有考虑其他国家的字符怎么编码，美国26个字母就可以搞定一切，但其他国家语言根本不够，例如：中国汉字就有85568个。于是各个国家，就各自定制了一套字符集，但是这些字符集互相不兼容。不兼容的意思是：中国把`00000001`编成`中`（即：`gbk`编码），而泰国编成`ก`，所以信息进行交流很麻烦

万国码

于是，美国人又整了一套"万国码"，把世界上所有书面语言的字符，进行了统一编号，这次用了2字节（16位，支持65536个字符编号），这就是Uniocode字符集。准确来说是`utf-16`

utf-8与utf-16

但是计划赶不上变化，美国人万万没想到，Unicode收录的字符很快超过了 65536个，所以编号又不够用了，于是，继续采取`utf-32`来存储，结果发现了个大问题，一个英文字母`a`也需要4个字节来存储，空间浪费太大了。为了解决这个问题，于是才有了`utf-8`可变长编码方式，英文字母只需要1个字节，而一般的汉字可以用3个字节，更稀奇古怪的字符，也可以用4，5或更多字节编号

定长编码，如：`utf-16`的好处是可以快速定位字符，而`utf-8`需要从头开始一个字符一个字符解析，所以运算速度会比`utf-16`要慢。但目前文件编码是`utf-8`，数据库编码是`utf-8`，网络流编码是`utf-8`等等，基本业内已全部统一使用`utf-8`

字节串

即一堆0或1组成的串，如：`11111111111111111010101010101010`，为了打印记录方便，常用用16进制来表示：`0xFFFFAAAA`

	字节串(bytes)	字符串(str)
操作单位	字节(二进制)	字符
不可变序列	是	是

bytes对象只负责以字节(二进制)序列来记录数据，至于这个数据到底表示什么内容，完全由程序决定

字节串转字符串

- 编码过程：字符串 -> 字符集映射表 -> 字节串

```
a = "你好".encode("utf-8")  # 默认也是utf-8
print(a, type(a))
b = "你好".encode("gbk")
print(b)
c = "你好".encode("unicode-escape")
print(c)

# 转换的字节串不一样
# b'\xe4\xbd\xa0\xe5\xa5\xbd'
# b'\xc4\xe3\xba\xc3'
# b'\\u4f60\\u597d'
```

```
b'\xe4\xbd\xa0\xe5\xa5\xbd' <class 'bytes'>
b'\xc4\xe3\xba\xc3'
b'\\u4f60\\u597d'
```

- 字符串转成字节串，在python中有三种方式：
  1. 如果字符串内容都ASCII字符，则可以直接通过`b'hello'`来构建
  2. 调用`bytes()`函数来构造，如：`bytes('我和我的祖国',encoding='utf-8')`，默认`encoding`为`utf-8`
  3. 通过`.encode()`来转换，如：`a = "我和我的祖国".encode('utf-8')`

```
a = b'hello'
print(a)
b = bytes("我和我的祖国", encoding="utf-8")
print(b)
c = "我和我的祖国".encode("utf-8")
print(c)
```

```
b'hello'
b'\xe6\x88\x91\xe5\x92\x8c\xe6\x88\x91\xe7\x9a\x84\xe7\xa5\x96\xe5\x9b\xbd'
b'\xe6\x88\x91\xe5\x92\x8c\xe6\x88\x91\xe7\x9a\x84\xe7\xa5\x96\xe5\x9b\xbd'
```

- 在python中打印字节串：

```
b3 = b'hello'
print(b3)  # 控制台输出 b'hello'，即原样输出
print(b3[0])  # 控制台输出：104，即：字母"h"在ascii表中对应的十进制数，其二进制数为：01101000
print(b3[2:4])  # 控制台输出：b'll'
```

```
b'hello'
104
b'll'
```

ascii码对照表：http://c.biancheng.net/c/ascii

解释：

1. 1个字节(byte)对应8位(bit)，`b'hello'`即有5个字节，40位
2. `\xe4`也是1个字节，`\x`表示16进制，`e6`是16进制数

练习：使用utf-8编码一下自己的姓名，输出一下字节串

字符串转字节串

字节串 -> 字符集映射表 -> 字符串

```
a = b'\xe4\xbd\xa0\xe5\xa5\xbd'
print(a.decode("utf-8"))  # 得到正确结果
print(a.decode("gbk"))  # 得到错误结果

# 你好
# 浣犲ソ
```

```
你好
浣犲ソ
```

- 同一个字节串，使用不同的字符集解析出来的是不同的内容

```
```