#!/usr/bin/env python3
# -\*- coding: utf-8 -\*-
//在python文件开头加入这两行，指定字符编码

背景：

ASCII:

最早只有127个字符被编码到计算机里，也就是大小写英文字母、数字和一些符号，这个编码表被称为ASCII编码，比如大写字母A的编码是65，小写字母z的编码是122。

Unicode:

使用两个字节来表示一个字符。ASCII采用一个字节

要处理中文显然一个字节是不够的，至少需要两个字节，而且还不能和ASCII编码冲突，所以，中国制定了GB2312编码，用来把中文编进去。

日本把日文编到Shift\_JIS里，韩国把韩文编到Euc-kr里，因此，Unicode把所有语言都统一到一套编码里，这样就不会再有乱码问题

UTF-8:

统一成Unicode编码，乱码问题从此消失了。但是，如果你写的文本基本上全部是英文的话，用Unicode编码比ASCII编码需要多一倍的存储空间，在存储和传输上就十分不划算。所以，本着节约的精神，又出现了把Unicode编码转化为“可变长编码”的UTF-8编码。

UTF-8编码把一个Unicode字符根据不同的数字大小编码成1-6个字节，常用的英文字母被编码成1个字节，汉字通常是3个字节，只有很生僻的字符才会被编码成4-6个字节。如果你要传输的文本包含大量英文字符，用UTF-8编码就能节省空间.

在计算机内存中，统一使用Unicode编码

当需要保存到硬盘或者需要进行传输时，转换为UTF-8编码。

用记事本编辑的时候，从文件读取的UTF-8字符被转换为Unicode字符到内存里，编辑完成后，保存的时候再把Unicode转换为UTF-8保存到文件

浏览网页的时候，服务器会把动态生成的Unicode内容转换为UTF-8再传输到浏览器

**字符串**

python支持多语言（在内存中使用Unicode进行保存，是字符串形式的）

对于单个字符的编码，python提供ord()获取字符的整数表示，使用char(）将字符编码转换为对应的字符

>>>ord('A')
65
>>>char(65)
'A'

2. Python的字符串类型是str，在内存中以Unicode表示，一个字符对应若干个字节。如果要在网络上传输，或者保存到磁盘上，就需要把str变为以字节为单位的bytes。

python对于bytes类型的数据可以采用带b前缀的单引号或者双引号表示

x=b'ABC'
y='ABC'
//x与y显示的内容一样，但是x中每个字符之占一个字节，而y占用两个字节

**str转为Bytes: 使用encode()方法**

**Bytes转为str:使用decode()方法**

>>>'ABC'.encode('ascii')
b'ABC'
>>>b'ABC'.decode('ascii')
'ABC'

纯英文的str可以用ASCII编码为bytes，含有中文的str可以用UTF-8编码为bytes。含有中文的str无法用ASCII编码，因为中文编码的范围超过了ASCII编码的范围，Python会报错。

在bytes中，无法显示为ASCII字符的字节，用\x##显示。

计算str中包含多少字符。调用len()方法

1个中文字符经过UTF-8编码后通常会占用3个字节

>>> len('ABC')
3

格式化：1.与c语言一样，才用%实现

2.format()，它会用传入的参数依次替换字符串内的占位符{0}、{1}

>>> 'Hello, %s' % 'world'
'Hello, world'
>>> 'Hi, %s, you have $%d.' % ('Michael', 1000000)
'Hi, Michael, you have $1000000.'

字符串里面的%是一个普通字符怎么办？这个时候就需要转义，用%%来表示一个%：

>>> 'growth rate: %d %%' % 7
'growth rate: 7 %'