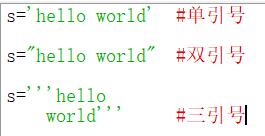
字符串

1. 字符串类型
2. **字符串常量：**

与C语言中储存字符串的字符数组不同，Python没有char这种单个字符的类型，取而代之的是使用一个字符的字符串。定义一个字符串常量有三种方式：单引号、双引号和三引号

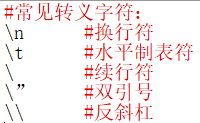


其中单引号和双引号两者是等效的，且字符序列只能在一行定义；三引号内的字符序列可以分布在连续的多行上。

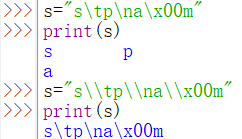
1. **转义字符与抑制转义字符：**

(1)转义字符

Python中用“\”字符嵌入我们不能通过键盘输入的字节。“\”以及后边的一个或多个字符最终在字符串中会被单个字符替代，这个字符通过转义字符定义了一个二进制值。



当Python遇到“\”时，会优先考虑将其翻译为转义字符，因此当我们想要打印“\”时要用“\\”



（2）抑制转义字符

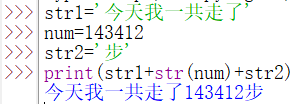
当一个新手尝试去用这样一行代码打开文件的时候就会遇到问题。这个调用的结果将会是C:(换行)ew（空格）ext.dat的文件，而不是我们所期待的结果。

Myfile=open(‘C:\new\text.dat’,’w’)

为了解决这个问题，raw字符串被引入。如果字母r的大写或小写形式出现在字符串的第一个引号前面，他就会关闭转义机制。

Myfile=open(r‘C:\new\text.dat’,’w’)

1. 字符串常用操作
2. **拼接字符串（直接将同类型的str相加）**

****

1. **计算字符串长度（len(str)）**
2. **截取字符串(索引和分片)**

字符串被定义为字符的有序集合，所以我们能够通过位置获取他们的元素。在Python中，字符串中的字符是通过索引提取的。Python中支持用负数来获取元素，具体来说负偏移可以看作是从结束处反向计数。

1. 索引（s[i]）获取特定偏移的元素：

----第一个元素的偏移为0

----负索引意味着从最后一位反向计数

1. 分片（s[i:j]）提取对应部分作为一个序列：

----上边界不包含在内

----分片的默认边界是0和序列的长度

----s[1:3]获取了偏移为1的元素，直到偏移不包括3的元素

----s[1:]获取了从偏移为1到末尾的序列

----s[:-1]获取了从偏移为0直到但是不包括最后一个元素的序列

1. 扩展分片(s[i:j:k])提取给定步长的子序列：

----s[1:10:2]会取出s中偏移值在1-9之间，间隔了一个元素的序列，也就是收集了1,3,5,7,9的元素

----s[::-1]会将全部序列通过负数步提取，实际效果就是将序列进行反转

1. **分割和合并字符串**
2. 分割字符串

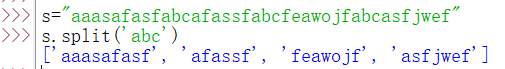
str.split(sep,maxsplit) #分割字符串

sep:用于指定分隔符，可以包含多个字符，默认状态下采用空白符进行分割，这时无论有多少空格都将作为一个分隔符进行分割。

Maxsplit：可选参数，用于指定分割的次数，默认状态为-1，即分割次数没有限制

返回值：分割后的字符串储存在列表之中

Eg:



1. 合并字符串

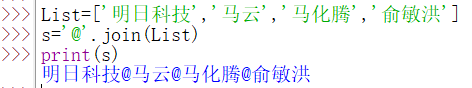
Strnew=string.join(iterable)

Strnew:合并后生成的新字符串

String:用于指定合并时的分隔符

Iterable:可迭代对象，该迭代对象中所有元素将被合并为一个新的字符串。

Eg:



1. **修改字符串**

String.replace(str1,str2) #用str2替换str1

1. 格式化字符串
2. **格式化表达式**

‘%[-][+][0][m][.n]格式化字符’%exp

-:可选参数，用于指定左对齐，正数前方无符号，负数前加负号

+：可选参数，用于指定右对齐，正数前方加正号，负数前方加负号

0：可选参数，表示右对齐，正数前方无符号，负数前方加负号，用0填充空白处

m:可选参数，表示占有宽度

.n:可选参数，表示小数点后保留的位数

格式化字符：%s %c %d %f

Exp:要转换的项

1. **字符串对象的format()方法**

str.format(args)

str:用于指定字符串的显示样式（即模板）

args：用于指定要转换的项，如果有多项，则用逗号分隔

下面重点介绍模板。在创建模板时，需要使用{}和：指定占位符，语法格式如下：

{[index][:[[fill]align][sign][#][width][.precision][type]}

Index:可选参数，用于指定设置格式的对象在参数列表中的索引位置，如果省略，则根据值前后顺序自动分配。

Fill:可选参数，用于指定空白处填充的字符

Align:可选参数，用于指定对齐方式(‘<’表示左对齐；’>’表示右对齐；’=’表示右对齐，只对数字有效；’^’居中)需要配合width一起使用

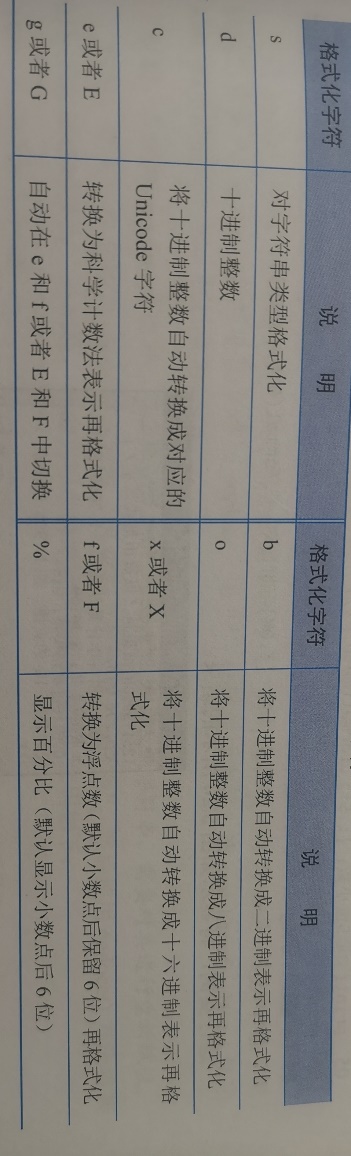
Sign:用于指定有无符号数（+表示整数加正号，负数加负号；-表示正数不变；空格表示整数加空格，负号加负号）

#：对于二进制、八进制和十六进制数，加上#会显示0b\0o\0x前缀

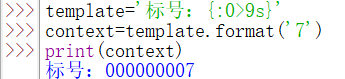
Width：用于指定所占宽度

.precision:指定保留小数位数

Type:用于指定类型



Eg:



Eg:



1. 字符串编码转换

在Python中，有两种常用的字符串类型，分别是str和bytes。其中str表示Unicode字符，bytes表示二进制数据。这两种类型无法拼接在一起使用，通常情况下，str在内存中以Unicode表示，一个字符对应多个字节，但是如果在网络上传输，或者存在磁盘上，就需要把str转化为字节类型。

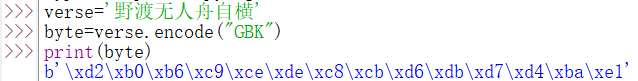
Str.encode([encoding=”utf-8”][,errors=”strict”])

Str:表示要进行转换的字符串

Encoding=”utf-8”:用于指定转码时采用的字符编码，默认为UTF-8，如果使用简体中文，也可以设置为gb2312.当只有这个参数时，可以省略encoding=，直接写编码

Errors=“strict”：用于指定错误处理方式，strict（遇到非法字符就抛出异常）ignore（忽略非法字符）replace（用？代替非法字符）默认值为strict

Eg:



Bytes.decode([encoding=’utf-8’][,error=strict])



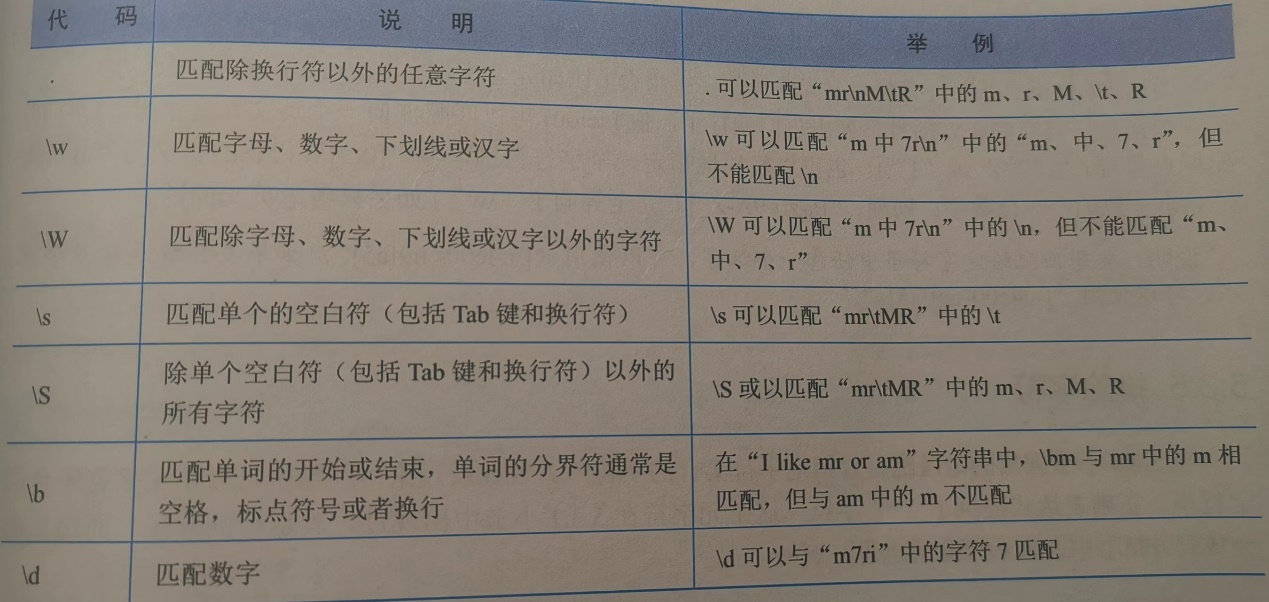
1. 正则表达式

在处理字符串中，经常会有查找符合某些复杂规则的字符串的需求。正则表达式就是描述这些规则的工具。正则表达式就是记录文本规则的代码。

1、行定位符：“^”表示行的开始；“$”表示行的结尾。

Eg:^tm:“tm equal tomorrow moon”; tm$:“tomorrow moon equal tm”;tm:匹配字符可出现任意位置

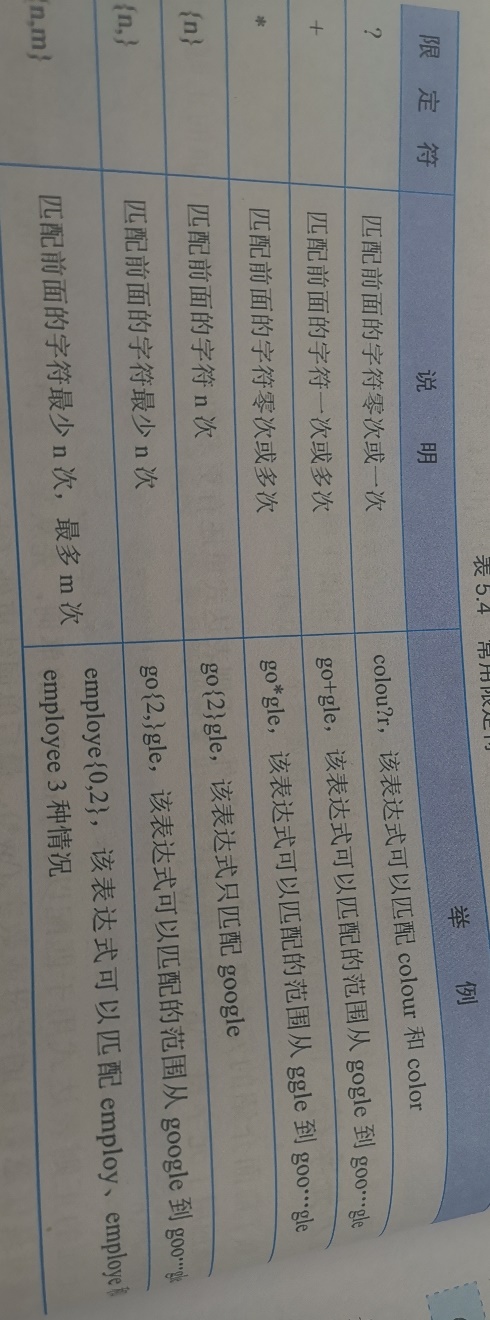
2、 元字符：



Eg:\bmr\w\*\b：以mr为开头的单词

3、 限定符：

使用\w\*匹配任意数量的字母或数字。如果想匹配特定数量的数字，那么就应该使用限定符来实现。



4、 字符和排除字符：

匹配元音：[aeiou]

匹配标点符号：[!.?]

匹配数字：[0-9]

匹配字母和数字:[0-9a-zA-Z]

匹配汉字：[\u4e00-\u9fa5]

匹配不是字母的:[^a-zA-Z]

5、 选择字符：“|”

Eg:身份证长度为15位或18位，15位时全部为数字，18位时前17位是数字，最后一位可能为数字或者X

匹配身份证的正则表达式可以是：

(^\d{15}$)|(^\d{18}$)|(^\d{17}X$)

6、转义字符：”\”

Eg:用正则表达式匹配“127.0.0.1”格式的IP地址

^[1-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}

由于.在正则表达式中有特殊含义，所以必须要用转义字符处理

7、分组：小括号

Eg:(six|four)th匹配sixth或者fourth

(\.[0-9]{1,3}){3}就是对IP地址的例子中后边三项的重复操作

1. Python中的re模块实现正则表达式操作
2. 在Python中使用正则表达式

匹配开头是m的单词的正则表达式转为模式字符串，不能够直接在两侧添加引号定界符，例如下边的代码是不对的：\bm\w\*\b而是应该将其中的“\”进行转义，结果为:\\bm\\w\*\\b.因此推荐使用raw原生字符串来生成模式字符串：r’\bm\w\*\b’

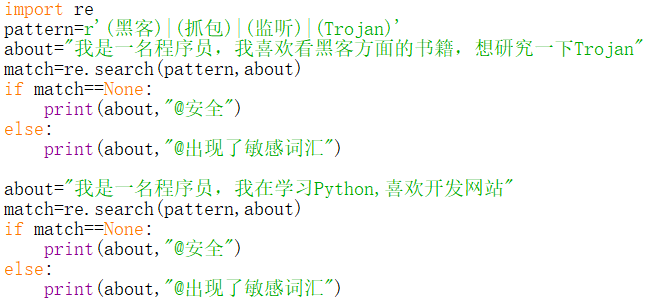
1. Re模块中匹配字符串

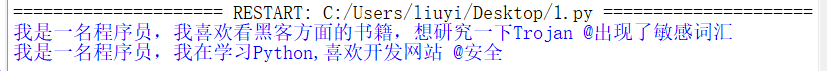
match(pattern,repl):从字符串的开始处进行匹配，如果在起始位置匹配成功，则返回match对象，否则None

search(pattern,repl):用于在整个字符串中搜索出第一个匹配的值，如果匹配成功，则返回match对象，否则None

findall(pattern,repl):用于在整个字符串中搜索所以符合正则表达式的字符串，并以列表的形式返回，如果成功，返回含有匹配结构的列表，否则返回空列表。

Eg:验证是否出现危险字符





1. Re模块中替换字符串

re.sub(pattern,repl,string,count,flags)

pattern:模式字符串

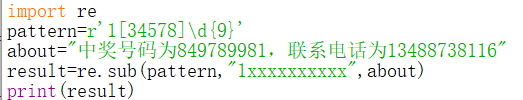
repl：要替换的字符串

string:表示要被查找替换的字符串

count:替换次数，默认为全部

flags:表示标志位，控制匹配模式，如是否区分大小写等。

Eg:隐藏中奖号码





1. Re分割字符串

re.split(pattern,string,[maxsplit],[flags])