**re模块**

正则表达式

匹配单个字符

|  |  |
| --- | --- |
| 记号 | 说明 |
| . | 匹配任意字符（换行符除外） |
| [...x-y...] | 匹配字符组里的任意字符 |
| [^...x-y...] | 匹配不在字符组里的任意字符 |
| \d | 匹配任意数字，与[0-9]同义 |
| \w | 匹配任意数字字母字符，与[0-9a-zA-Z]同义 |
| \s | 匹配空白字符，与[\r\v\f\t\n]同义 |

匹配一组字符

|  |  |
| --- | --- |
| 记号 | 说明 |
| literal | 匹配字符串的值 |
| re1|re2 | 匹配正则表达式re1或re2 |
| \* | 匹配前面出现的正则表达式零次或多次 |
| + | 匹配前面出现的正则表达式一次或多次 |
| ？ | 匹配前面出现的正则表达式零次或一次 |
| {M,N} | 匹配前面出现的正则表达式至少M次最多N次 |

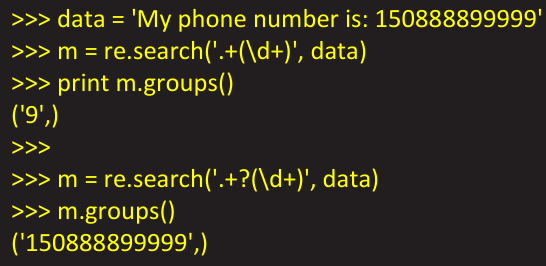
其它元字符

|  |  |
| --- | --- |
| 记号 | 说明 |
| ^ | 匹配字符串的开始 |
| $ | 匹配字符串的结尾 |
| \b | 匹配单词的边界 |
| () | 对正则表达式分组 |
| \nn | 匹配已保存的子组 |

贪婪匹配

\*、+和?都是贪婪匹配操作符，在其后加上?科一取消其贪婪匹配行为

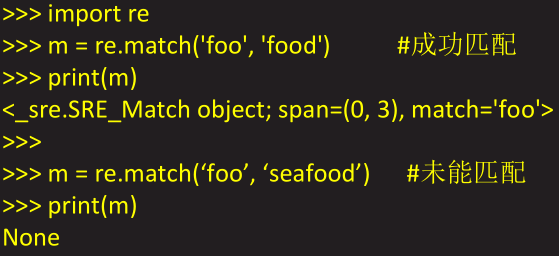
正则表达式匹配对象通过groups函数获取子组



核心函数和方法

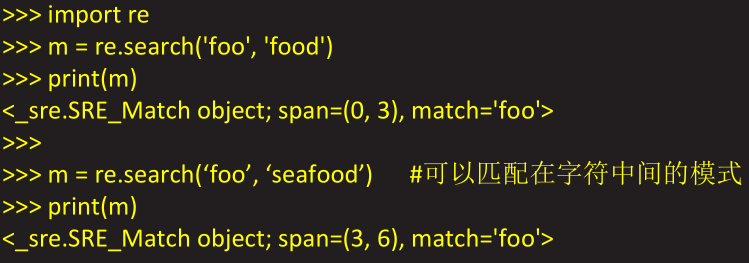
match函数

尝试用正则表达式模式从字符串的开头匹配，如果匹配成功，则返回一个对象；否则返回NONE



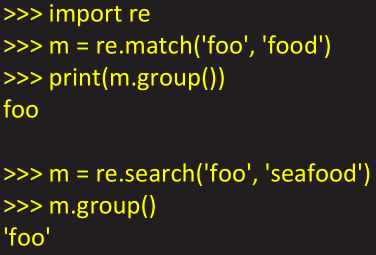
search函数

在字符串中查找正则表达式模式的第一次出现，如果匹配成功，则返回一个匹配对象；否则返回NONE



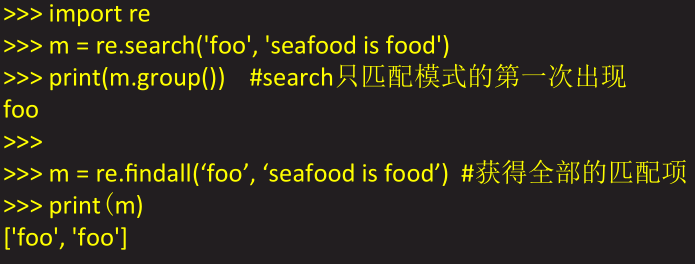
group方法

使用match或search匹配成功后，返回的匹配对象可以通过group方法获得匹配内容



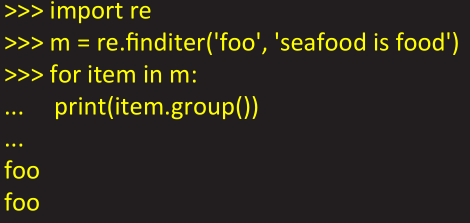
findall函数

在字符串中查找正则表达式模式的所有（非重复）出现；返回一个匹配对象的列表



finditer函数

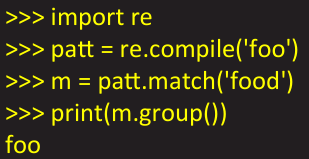
和findall()函数有相同的功能，但返回的不是列表而是迭代器；对于每个匹配，该迭代器返回一个匹配对象



compile函数

对正则表达式模块进行编译，返回一个正则表达式对象

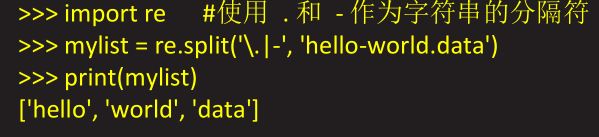
不是必须要用这种方式，但是在大量匹配的情况下，可以提升效率



split方法

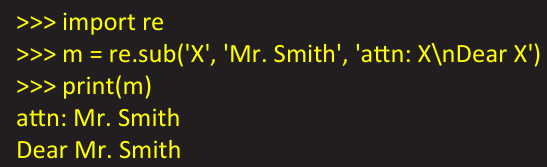
根据正则表达式中的分隔符把字符分割为一个列表，并返回成功匹配的列表

字符串也有类似的方法，但是正则表达式更加灵活



sub方法

把字符串中所有匹配正则表达式的地方替换成新的字符串



**socket模块**

C/S架构

什么是C/S架构

服务器是一个软件或硬件，用于提供客户需要的“服务”

硬件上，客户端常见的就是平时所使用的PC机，服务器常见的有联想、DELL等厂商生产的各种系列服务器

软件上，服务器提供的服务主要是程序的运行，数据的发送与接收、合并、升级或其它的程序或数据的操作

套接字

套接字是一种具有“通信端点”概念的计算机网络数据结构

套接字起源于20世纪70年代加利福尼亚大学伯克利分校版本的UNIX

一种套接字是UNIX套接字，其“家族名”为AF\_UNIX

另一种套接字是基于网络的，“家族名”为AF\_INET

如果把套接字比作电话的插口，那么主机与端口就像区号和电话号码的一对组合

面向连接与无连接

无论使用哪一种地址家族，套接字的类型只有两种。一种是面向连接的套接字，另一种是面向无连接的套接字

面向连接的主要协议九十传输控制协议TCP，套接字类型为SOCK\_STREAM

无连接的主要协议是用户数据报协议UDP，套接字类型SOCK\_DGRAM

python种使用socket模块中的socket函数实现套接字的创建

socket函数与方法

创建TCP服务器

创建TCP服务器的主要步骤如下：

1. 创建服务器套接字：s=socket.socket
2. 绑定地址到套接字：s.bind()
3. 启动监听：s.listen()
4. 接收客户端连接：s.accept()
5. 与客户端通信：recv()/send()
6. 关闭套接字：s.close()

创建TCP客户机

创建TCP客户端的步骤主要如下：

1. 创建客户端套接字：cs=socket.socket()
2. 尝试连接服务器：cs.connect()
3. 与服务器通信：cs.send()/cs.recv()
4. 关闭客户端套接字：cs.close

创建UDP服务器

创建UDP服务器的主要步骤如下：

1. 创建服务器套接字：s=socket.socket()
2. 绑定服务器套接字：s.bind()
3. 接收、发送数据：s.recvfrom()/ss.sendto()
4. 关闭套接字：s.close()

创建UDP客户端

创建UDP客户端的主要步骤如下：

1. 创建客户端套接字：cs=socket.socket()
2. 与服务器通信：cs.sendto()/cs.recvfrom()
3. 关闭客户端套接字：cs.close()