# 课程介绍

* 正则表达式的基本原理（了解）
* 正则表达式匹配规则（重点掌握）
* re模块（重点掌握）
* 用正则表达式处理HTML（掌握）

1. 正则表达式
   1. 简介

爬虫抓取的海量信息中，我们需要提取有用的部分数据，而这些数据可能以各种类型存在着，比如：JSON、HTML、XML、CSV（Comma-Separated Values，逗号分隔值）等。不同类型的数据，我们需要采用不同的方式和工具来处理，正则表达式就是其中一种强大的工具。

正则表达式是一个特殊的字符序列，它能帮助你方便的检查一个字符串是否与某种模式匹配。

正则表达式本身就是一种语言，这在其它语言是通用的.

正则表达式是对[字符](http://baike.baidu.com/view/263416.htm)串操作的一种逻辑公式，就是用事先定义好的一些特定字符、及这些特定字符的组合，组成一个“规则字符串”，这个“规则字符串”用来表达对其他字符串的一种过滤逻辑。可以用来检查一个串是否含有某种子串、将匹配的子串做替换或者从某个串中取出符合某个条件的子串等。

正则表达式的两个例子：

**匹配Email地址的正则表达式：**

\w+([-+.]\w+)\*@\w+([-.]\w+)\*\.\w+([-.]\w+)\*  
评注：表单验证时很实用

**匹配中国邮政编码：**

[1-9]\d{5}  
评注：中国邮政编码为6位数字

**国内固定电话号码：**

\d{2,4}-?\d{8}

* 1. 正则表达式能做什么

1. 检查字符串的合法性
   * 1. 验证用户名 （a-z，0-9，不能全是数字，不能全是字母）
     2. 验证邮箱格式 (xxx@qq.com)
     3. 验证电话号码 (11位数字)
     4. 验证身份证 (18位 )
     5. 验证QQ号码格式（5-12纯数字，第一位不能为0）；
2. 提取字符串中信息
   * 1. 提取一条短信中数字；
     2. 提取文件名的后缀；
     3. 采集器(网路爬虫)
3. 替换字符串
   * 1. 替换字符串中的非法字符；
     2. 对电话号码进行屏蔽；（1852\*\*\*\*0102）
     3. 替换占位符 “hello {{name}} ” hello 王老二 （模板框架）
4. 分割字符串
   * 1. 将一个字符串按照指定的规则进行分割；

例如: 按照任意空格数进行分割.

* 1. re.match()

re.match(*pattern*, *string*)

*pattern*指用做匹配的正则表达式规则字符串，*string*指的是待匹配的字符串。

该函数尝试用pattern表示的正则表达式规则，去匹配字符串string，返回匹配结果类型。匹配需从待匹配字符串第一个字符开始，默认为贪婪模式，试图匹配尽可能长的子串。如果匹配成功，则立即返回匹配结果对象，不再继续匹配；如果匹配失败，则立即返回None。

1. 正则表达式匹配规则
   1. 普通字符

使用正则表达式中的普通字符串匹配来使用。

所有的大写和小写字母字符，所有数字，以及一些符号。

* 1. 元字符

元字符就是正则表达式中具有特殊含义的单个字符或字符组合，这种单个字符或字符组合是作为正则表达式的一个不可分割的意义单元出现。

* + 1. 一般字符

|  |  |
| --- | --- |
| . | 匹配除换行符以外的任意一个字符 |
| \ | 转义字符 |

* + 1. 单字符簇

通过[]定义单个字符,[]中的是该字符的可选字符.

|  |
| --- |
| [abc] |
| [^abc] |
| [a-zA-Z] |

* + 1. 预定义字符集

|  |  |
| --- | --- |
| \d | 数字:[0-9] |
| \D | 非数字[^0-9] |
| \s | 空白字符[<空格>\t\r\n\f\v] |
| \S | 非空白字符[^\s] |
| \w | 单词字符[A-Za-z0-9\_] |
| \W | 非单词字符[^\w] |

注意以上的预定义字符只代表一个字符.只是定义了该字符可以出现的范围.

例如:\d表示一个字符,但是该字符只能够是0-9之间的一个字符.

* + 1. 数量词

可以限制 正则表达式匹配上的内容 必须要出现多少次才能满足匹配。

|  |  |
| --- | --- |
| **\*** | 匹配前一个字符**0**或者多次 |
| + | 匹配前一个字符1次或者多次 |
| ? | 匹配前一个字符0次或者1次 |
| {m} | 匹配前一个字符m次 |
| {m,n} | 匹配前一个字符m至n次 |
| 数量词? | 变成非贪婪模式 |

* + 1. 边界

可以将一个正则表达式固定在一行的开始或结束。也可以创建只在单词内或只在单词的开始或结尾处出现的正则表达式。

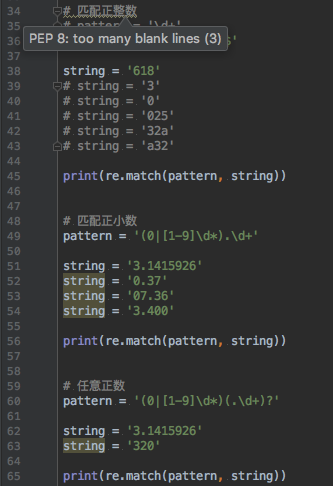
|  |  |
| --- | --- |
| ^ | 匹配字符串开头，多行匹配每一行开头 |
| $ | 匹配字符串末尾，多行匹配每一行末尾 |
| \A | 仅匹配字符串开头 |
| \Z | 仅匹配字符串末尾 |
| \b | 匹配\w和\W之间 |

* + 1. 逻辑、分组

|  |  |
| --- | --- |
| | | 左右表达式任意匹配一个 先匹配左边一旦成功则跳过匹配右边 如果|没有包含在()中，匹配整个正则表达式 |
| (...) | 分组匹配，分组后面可加数量词 |
| (?P<name>...) | 除了分组序号外，指定一个 name的别名 |

* 1. 案例 正则表达式基础规则应用

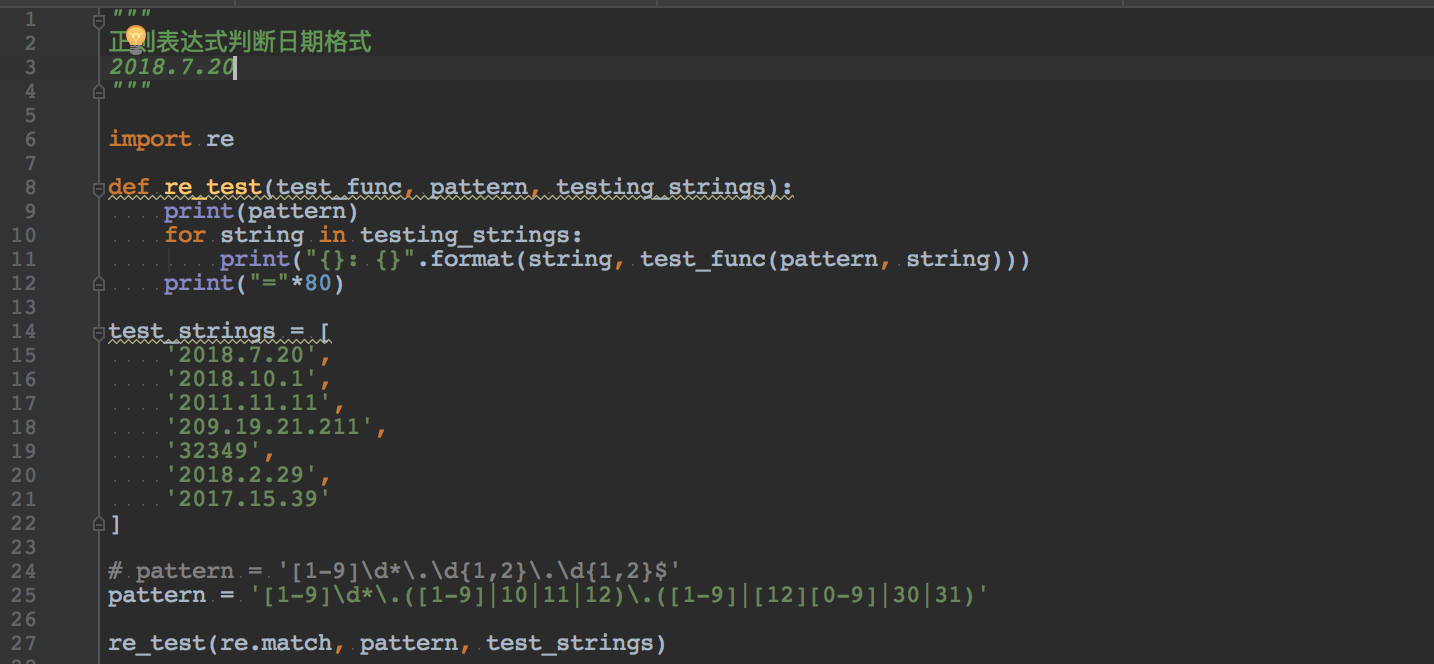




* 1. 案例 正则表达式判断邮箱地址是否合法



* 1. 案例 正则表达式判断日期格式



1. re模块的使用
   1. re模块的对象式使用：
      * 1. 使用 compile() 函数将正则表达式的字符串形式编译为一个 Pattern 对象
        2. 通过 Pattern 对象提供的一系列方法对文本进行匹配查找，获得匹配结果，一个 Match 对象。
        3. 最后使用 Match 对象提供的属性和方法获得信息，根据需要进行其他的操作
      1. compile()函数

compile 函数用于编译正则表达式，生成一个 Pattern 对象，它的一般使用形式如下：

import re

# 将正则表达式编译成 Pattern 对象

pattern = re.compile(r'\d+')

在上面，我们已将一个正则表达式编译成 Pattern 对象，接下来，我们就可以利用 pattern 的一系列方法对文本进行匹配查找了。

* + 1. Pattern 对象的一些常用方法主要有：
* match 方法：从起始位置开始查找，一次匹配

match(string[, flags])

* search 方法：从任何位置开始查找，一次匹配

search(string[, flags])

* findall 方法：全部匹配，返回列表

findall(string[, flags])

* finditer 方法：全部匹配，返回迭代器

finditer(string[, flags])

* split 方法：分割字符串，返回列表

split(string[, maxsplit, flags])

* sub 方法：替换

sub(repl, string[, count, flags])

* 1. re模块的函数式使用

Pattern对象的每一种常用匹配方法都有对应的函数式版本，该版本封装了如下行为：通过传入的正则表达式字符串创建Pattern对象，然后调用对应的匹配方法作为在目标字符串上。

通常，re模块的匹配函数用起来更方便。

* + 1. re.match()

match()用于查找字符串的头部（也可以指定起始位置），它是一次匹配，只要找到了一个匹配的结果就返回，而不是查找所有匹配的结果。它的一般使用形式如下：

match(pattern, string[, flag])

其中，pattern是正则表达式规则字符串，string 是待匹配的字符串，flag 是可选参数。

当匹配成功时，返回一个 Match 对象，如果没有匹配上，则返回 None。

* + 1. re.search()

search()用于查找字符串的任何位置，它也是一次匹配，只要找到了一个匹配的结果就返回，而不是查找所有匹配的结果，它的一般使用形式如下：

search(pattern, string[, flag])

当匹配成功时，返回一个 Match 对象，如果没有匹配上，则返回 None。

* + 1. re.findall()

上面的 match 和 search 方法都是一次匹配，只要找到了一个匹配的结果就返回。然而，在大多数时候，我们需要搜索整个字符串，获得所有匹配的结果。

findall() 的使用形式如下：

findall(pattern, string[, flag])

findall 以列表形式返回全部能匹配的子串，如果没有匹配，则返回一个空列表。

* + 1. re.finditer()

finditer() 的行为跟 findall 的行为类似，也是搜索整个字符串，获得所有匹配的结果。但它返回一个顺序访问每一个匹配结果（Match 对象）的迭代器。

使用形式如下：

finditer(pattern, string[, flag])

* + 1. re.split()

split()按照能够匹配的子串将字符串分割后返回列表，它的使用形式如下：

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

其中，maxsplit 用于指定最大分割次数，不指定将全部分割。

* + 1. re.sub()

sub()用于替换，使用形式如下：

sub(pattern, repl, string[, count, flags])

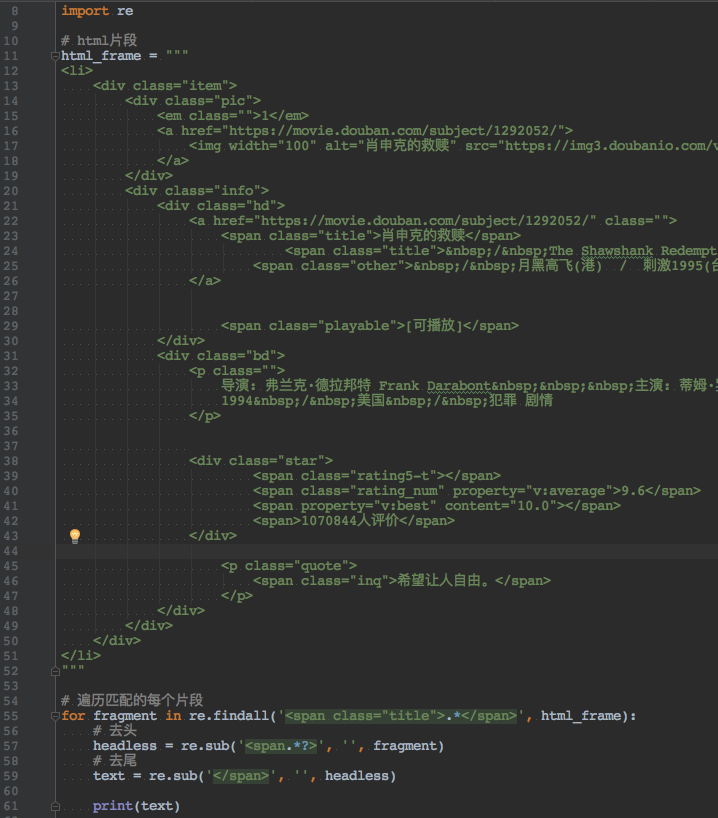
其中，repl 可以是字符串也可以是一个函数：

如果 repl 是字符串，则会使用 repl 去替换字符串每一个匹配的子串，并返回替换后的字符串，另外，repl 还可以使用 id 的形式来引用分组，但不能使用编号 0；

如果 repl 是函数，这个方法应当只接受一个参数（Match 对象），并返回一个字符串用于替换（返回的字符串中不能再引用分组）。

count 用于指定最多替换次数，不指定时全部替换。

1. 使用正则表达式从HTML片段中提取数据
   1. 从HTML片段中提取数据（此处未使用捕获组）



1. 扩展知识: 正则表达式的捕获组和引用（可选）
   1. 引入的原因

生活中有时会遇到需要判断字符串反复出现的情况，比如：我们想得到含有同一字母连续出现两次以上序列的单词表，例如”soon”, ”see”。又比如：我们想判断一个电话号码是不是靓号，例如含有连续三个相同数字序列的电话号码就是靓号。

* + 1. 试图通过已学知识解决

**案例：**使用正则表达式判断一个字符串是否由连续三个相同的字母组成。

比如"aaa, “zzz”, “AAA”满足要求。

而”abc”, “aba”, “aAa”不满足要求。



正则表达式\w{3}在此处相当于\w\w\w，这三处\w做匹配每处都是相互独立完成的，没有记忆匹配后的字符串实例的功能，所以这三次匹配的字符串实例之间没有任何联系，无法达到预想效果。

* + 1. 怎么办？

添加“记忆功能”：捕获组+反向引用。

捕获组保存记忆，反向引用使用保存的记忆。

|  |  |
| --- | --- |
| (...) | 分组匹配，从左到右，每遇到一个 ( 编号+1 分组后面可加数量词 |
| (?P<name>...) | 除了分组序号外，指定一个 name的别名 |
| \<number> | 引用编号为<number>的分组匹配到的字符串 |
| (?P=name) | 引用别名为<name>的分组匹配到的串 |

* 1. 捕获组
     1. 概念：捕获组保存“记忆”

捕获组就是，在正则表达式匹配过程中，把其中指定子表达式匹配的内容，保存到内存中以数字编号的组里，方便后面引用。

我们可以这样理解：捕获组中所匹配的内容，就是通过一种特定计算产生的值，捕获组把所产生的值赋值给一种特殊变量。

* 1. 语法

**(Expression)**

**案例1：(\w)=(\d)**

案例2：(\w)\+((\w)\\*(\w))

案例3：(\w){3}

* + 1. 捕获组的编号规则

**捕获组的编号：**

**按照“(”出现的顺序，从左到右，从1开始进行编号。**

**编号0指的是正则表达式整体。**

**将来按数字顺序编号来访问捕获组。**

**案例1：(\w)=(\d)**

**编号 1 2**

**上面的正则表达式可以用来匹配格式为*x*=*num*的简单等式。**

**用以上正则表达式匹配字符串：n=6，匹配结果为：**

|  |  |  |
| --- | --- | --- |
| **编号** | **子表达式** | **匹配内容** |
| **0** | **(\w)=(\d)** | **n=6** |
| **1** | **(\w)** | **n** |
| **2** | **(\d)** | **6** |

**案例2：**(\w)\+((\w)\\*(\w))

编号 1 2 3 4

**上面的正则表达式可以用来匹配格式为*x+y*\*z的简单代数式。**

**用以上正则表达式匹配字符串：a+b\*c，匹配结果为：**

|  |  |  |
| --- | --- | --- |
| **编号** | **捕获组子表达式** | **匹配内容** |
| **0** | (\w)\+((\w)\\*(\w)) | **a+b\*c** |
| **1** | **第一处(\w)** | **a** |
| **2** | ((\w)\\*(\w)) | **b\*c** |
| **3** | 第二处(\w) | **b** |
| **4** | 第三处(\w) | **c** |

案例3：(\w){3}

编号 1

上面的正则表达式可以用来匹配三个字母的字符串。

**用以上正则表达式匹配字符串：abc，匹配结果为：**

|  |  |  |
| --- | --- | --- |
| **编号** | **捕获组子表达式** | **匹配内容** |
| **0** | (\w){3} | **abc** |
| **1** | **第一次(\w)** | **a** |
| **1** | 第二次(\w) | **b** |
| **1** | 第三次(\w) | **c** |

* 1. 捕获组在程序中的引用
     1. 概念

捕获组捕获到的内容，可以在正则表达式外部通过程序进行引用。

* + 1. 作用

可以用来获取正则表达式中指定的局部表达式所匹配的字符串内容，达到一次匹配，多次使用的效果。

* + 1. 语法

**result.group(***number***)**

其中result是re.match(...)或者re.search(...)返回的结果对象，number是十进制的数字，即捕获组的编号。

* 1. 捕获组的反向引用
     1. 概念

捕获组捕获到的内容，不仅可以在正则表达式外部通过程序进行引用，也可以在正则表达式内部进行引用，后者就是反向引用。

* + 1. 作用

反向引用的作用通常是用来查找或限定重复，限定指定标识配对出现等等。

* + 1. 语法

**\number**

其中number是十进制的数字，即捕获组的编号。

* 1. 捕获组和反向引用的过程
     1. 图示

待匹配字符串：aaa

\1 表示第一个捕获组匹配到的内容 a

\0 表示整个正则表达式匹配到的内容 aaa

匹配过程中存储匹配上的内容

在正则表达式中 \n引用捕获组所匹配的内容

匹配中

正则表达式：(\w)\1{2} (\w)是第一个捕获组

* 1. 案例：使用正则表达式判断一个字符串是否满足三连字母的模式
     1. 需求

使用正则表达式判断一个字符串是否满足三连字母的模式，比如”aaa”, “zzz”就是三连字母。

* + 1. 代码



* + 1. 运行结果



* 1. 案例：使用正则表达式从可用电话号码池中过滤掉“靓号”
     1. 需求

使用正则表达式判断一个电话号码是不是靓号，这里简单认为含有连续三个相同数字的序列，这样的电话号码就是靓号。

* + 1. 代码



这里刻意涉及了列表产生式（list comprehension）和.join()方法。

* + 1. 运行结果



* 1. 用正则表达式从douban完整html页面上提取数据



1. 课程总结
   1. 重点
      * 1. 正则表达式的匹配规则。
        2. Python的re模块的用法。
   2. 难点
      * 1. 转义字符：特别是在Python字符串和正则表达式规则的两次转义
        2. 捕获组和反向引用
   3. 避坑宝典
      1. 注意回归测试
      2. 注意完善测试集
      3. 注意特殊字符（特别是点号.）的转义
      4. 注意正则表达式默认的贪婪匹配的特性
2. 课后练习
   1. **必做题**
      * 1. 实现上课代码，尽量领会后独立实现，包括自己简要的注释
        2. 用正则表达式验证国内用户手机号是否合法

主号码区11位数字组成，规定第一位为1，第二位在3,5,8中间取。

包括153-xxxxxxxx和153xxxxxxxx两种格式都接受。

另外，主号码前可以有“(+86)”或者“+86-”的可选项（86是中国大陆的国际区号）。

* + - 1. 找出所有“靓词”

“靓词”就是含有同一字母连续出现两次以上模式的单词，例如”soon”, ”see”。要求读入给定的包含常见英文单词的文本文件words.txt，其中每行一个单词；然后输出所有“靓词”到nice\_words.txt文件中，每行一个单词，并打印“靓词”总数。

* + - 1. 完成从豆瓣电影<https://movie.douban.com/top250>完整html页面中，提取电影名、评分、评价人数和名言金句，每个条目一个字典，组织成列表。
      2. 完成注册表单验证功能块，判断用户是否填写用户名（长度6~20，以字母开头，包含字母、数字、-、汉字），邮箱（邮箱格式），电话号码（长度和格式），密码（字母开头，包含大小写字母、数字和特殊字符!@#$%^&\*，长度8-20位），密码和确认密码是否一致，如果验证通过返回True，如果验证失败显示错误信息。
  1. 选做题（请在能于12点前睡觉的前提下尝试）
     + 1. 用正则表达式判断一个字符串形式的十进制自然数是否能被3整除。
       2. 用正则表达式判断一个字符串形式的十进制自然数是否能被7整除。

1. 面试题
   * + 1. 用 Python 匹配 HTML的tag 的时候，<.\*> 和 <.\*?> 有什么区别？
2. 每日一练
   * + 1. 对于常见功能需求，完成使用urllib库和requests库的对比。