





首页
所有文章
观点与动态
基础知识
系列教程
实践项目
工具与框架
工具资源
Python小组

- 导航条 -

伯乐在线 > [Python - 伯乐在线](#) > [所有文章](#) > [基础知识](#) > 众里寻她千百度，正则表达式

众里寻她千百度，正则表达式

2016/10/02 · [基础知识](#) · [6 评论](#) · [正则表达式](#)

分享到： 18 本文作者：[伯乐在线](#) - [selfboot](#)。未经作者许可，禁止转载！
欢迎加入伯乐在线 [专栏作者](#)。

先来看一个让人震撼的小故事，故事来自知乎问题[PC用户的哪些行为让你当时就震惊了？](#)

同学在一个化妆品公司上班，旁边一个大妈（四十多岁）发给他一个exl表，让他在里面帮忙找一个经销商的资料。

表格里面大约有几百个客户资料，我同学直接筛选填入信息，然后没找到，就转头告诉大妈，说这个表里没有。

大妈很严厉的批评了我同学，说年轻人干工作一定要沉的住气，心浮气躁可不行。这才几分钟啊，我才看了二十行，你怎么就找完了。

同学过去一看，大妈在一行一行的精挑细选，顿时一身冷汗。把筛选办法告知后，大妈不但不领情，还召集办公司其他老职员，一起声讨我同学，我们平时都是这么找的，你肯定是偷工减料，我们找一个小时没找完，你几分钟就找完了。

不知道是否确有此事，不过看起来好吓人的样子。仔细想想，大多数人都是用以往的经验来分析遇见的新问题的。就上面的大妈而言，在接触计算机之前的几十年里，她面对的都是纸质的客户资料，此时，要查找某一客户资料，只能一行一行看下去了。

现在，虽然有了计算机，但是只是简单的把它看做一个比较大的纸质资料库罢了，并没有认识到计算机的强大之处。这里的强大主要就是说计算机在处理电子文档时的强大的搜索功能了。

当然，对于大部分年轻人来说，计算机中的搜索功能是再熟悉不过了。我们可以在word、excel、网页中搜索特定内容，可以在整个计算机文件系统中搜索文件名，甚至搜索文件中的内容（Win下的everthing，Mac下的Spotlight）。

这些搜索主要用到了两种技术：

1. 正则表达式
2. 数据库索引

这里我们先介绍一下正则表达式。

正则表达式介绍

简单来说，正则表达式就是用来匹配特定内容的字符串。举个例子来讲，如果我想找出由a、b组成的，以abb结尾的字符串，比如ababb，那么用正则表达式来表示就是`[ab]*abb`。

正则表达的理念是由数学家[Stephen Kleene](#)在1950年首次提出来的，开始时主要用于UNIX下文本编辑器ed和过滤器grep中。1968年开始广泛应用于文本编辑器中的模式匹配和编译器中的词法分析。1980年，一些复杂的正则表达式句开始出现在Perl中，使用了由[Henry Spencer](#)实现的正则表达解析器。而Henry Spencer后来写了更高效的正则解析器Tcl，Tcl混合使用了[NFA](#)（非确定有限自动机）/[DFA](#)（确定有限自动机）来实现正则表达语法。

正则表达式有以下优点：

- 容易理解
- 能高效实现
- 具有坚实的理论基础

正则表达式的语法十分简单，虽然各种编程语言在正则表达式的语法上有细节上的区别，不过主要部分如下：

1. `[a-z]`表示所有小写字母，`[0-9]`表示所有数字，`[amk]`表示a、m或k。
2. `+`表示字符重复1或者多次，`*`表示字符重复0或者多次。在使用`+`或者`*`时，正则表达式遵从maximal munch的原则，也就是说它匹配能够匹配到的最大字符串。
3. `a|z` 表示匹配字符‘a’或者‘z’
4. `?`表示字符出现0次或者1次
5. 是正则表达式中的escape符号，`*`表示的就是‘*’这个字符，而不是它在正则表达式中的功能。
6. `.`表示出了换行符之外的任何字符，而`^`表示出了紧接它的字符以外的任何字符
7. `^` 匹配字符串的开始，`$` 匹配字符串的结尾。

回到我们前面的例子中，我们用正则表达式`[ab]*abb`来匹配由a、b组成的，以abb结尾的字符串。这里`[ab]*abb`即可以这样解读：a或者b重复0或者多次，然后是abb的字符串。

下面用python在“aababbaxz abcabb abbbbabb”中搜索`[ab]*abb`：

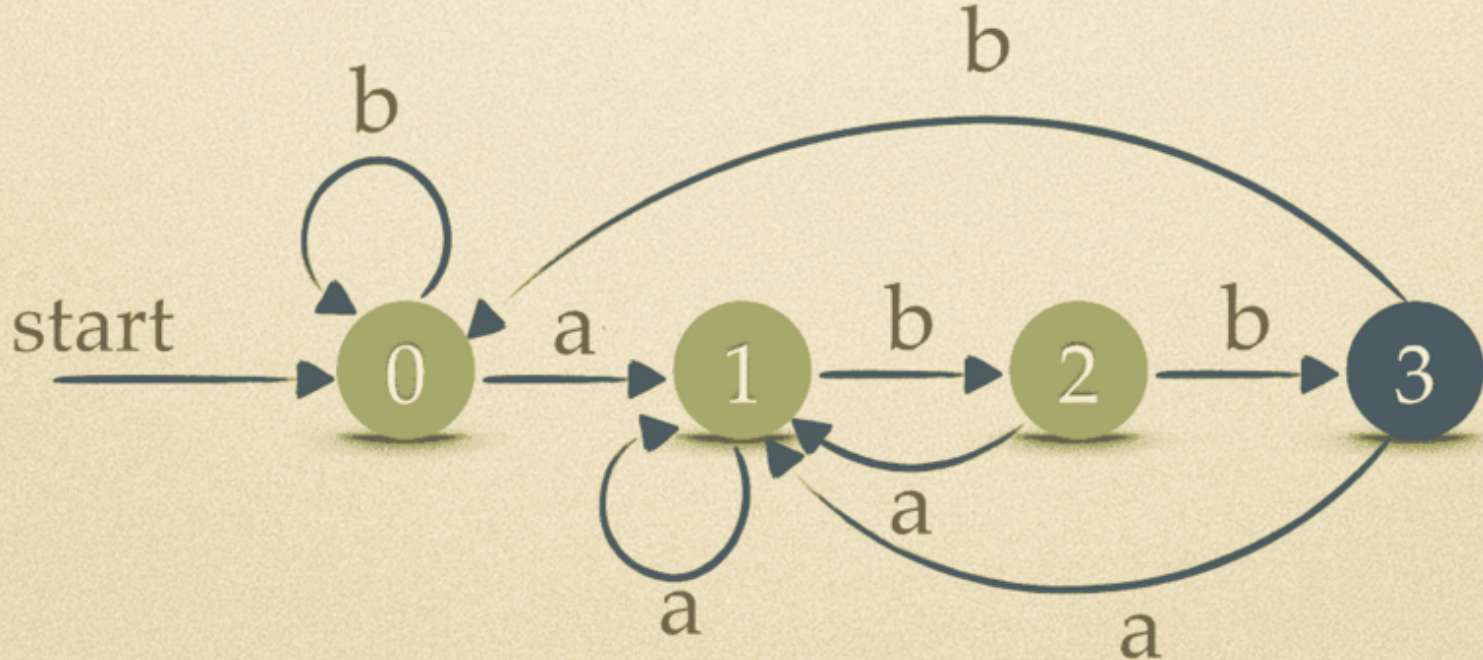
```
1 import re
2 content = "aababbaxz abcabb abbbbabb"
3 pattern = re.compile("[ab]*abb")
4 print pattern.findall(content)
5 # outputs: ['aababb', 'abb', 'abbbbabb']
```

其实，正则表达式不只用于文本搜索和模糊匹配，还可以用于以下场景：

1. 合法性检查
2. 文本的自动更正和编辑
3. 信息提取

正则表达式实现原理

正则表达式便于我们理解使用，但是如何让计算机识别用正则表达式描述的语言呢？仍然以前面的`[a|b]*abb`为例，计算机如何识别`[a|b]*abb`的意义呢？首先我们来看判断输入内容是否匹配正则表达式的流程图：



图中一共有4个状态S0, S1, S2, S3，在每个状态基础上输入字符a或者b就会进入下一个状态。如果经过一系列输入，最终如果能达到状态S3，则输入内容一定满足正则表达式[a|b]*abb。

为了更清晰表述问题，将上图转换为状态转换表，第一列为当前状态，第二列为输入a后当前状态的跳转，第三列为输入b后当前状态的跳转。其中S0为起始状态，S3为接受状态，从起始状态起经过一系列输入到达接受状态，那么输入内容即满足[a|b]*abb。

状态	a	b
S0	S1	S0
S1	S1	S2
S2	S1	S3
S3	S1	S0

其实上图就是一个DFA实例（确定有限自动机），下面给出DFA较为严格的定义。一个确定的有穷自动机(DFA) M 是一个五元组：M = (K, Σ, f, S, Z)，其中：

1. K是一个有穷集，它的每个元素称为一个状态；
2. Σ是一个有穷字母表，它的每个元素称为一个输入符号，所以也称Σ为输入符号表；
3. f是转换函数，是在 $K \times \Sigma \rightarrow K$ 上的映射，如 $f(k_i, a) \rightarrow k_j$, $k_i \in K$, $k_j \in K$ 就意味着当前状态为 k_i ，输入符号为a时，将转换为下一个状态 k_j ，我们将 k_j 称作 k_i 的一个后继状态；
4. $S \in K$ 是唯一的一个初态；
5. $Z \subseteq K$ 是一个状态集，为可接受状态或者结束状态。

DFA的确定性表现在转换函数 $f: K \times \Sigma \rightarrow K$ 是一个单值函数，也就是说对任何状态 $k_i \in K$ 和输入符号 $a \in \Sigma$ ， $f(k, a)$ 唯一地确定了下一个状态，因此DFA很容易用程序来模拟。

下面用字典实现[a|b]*abb的确定有限自动机，然后判断输入字符串是否满足正则表达式。

```

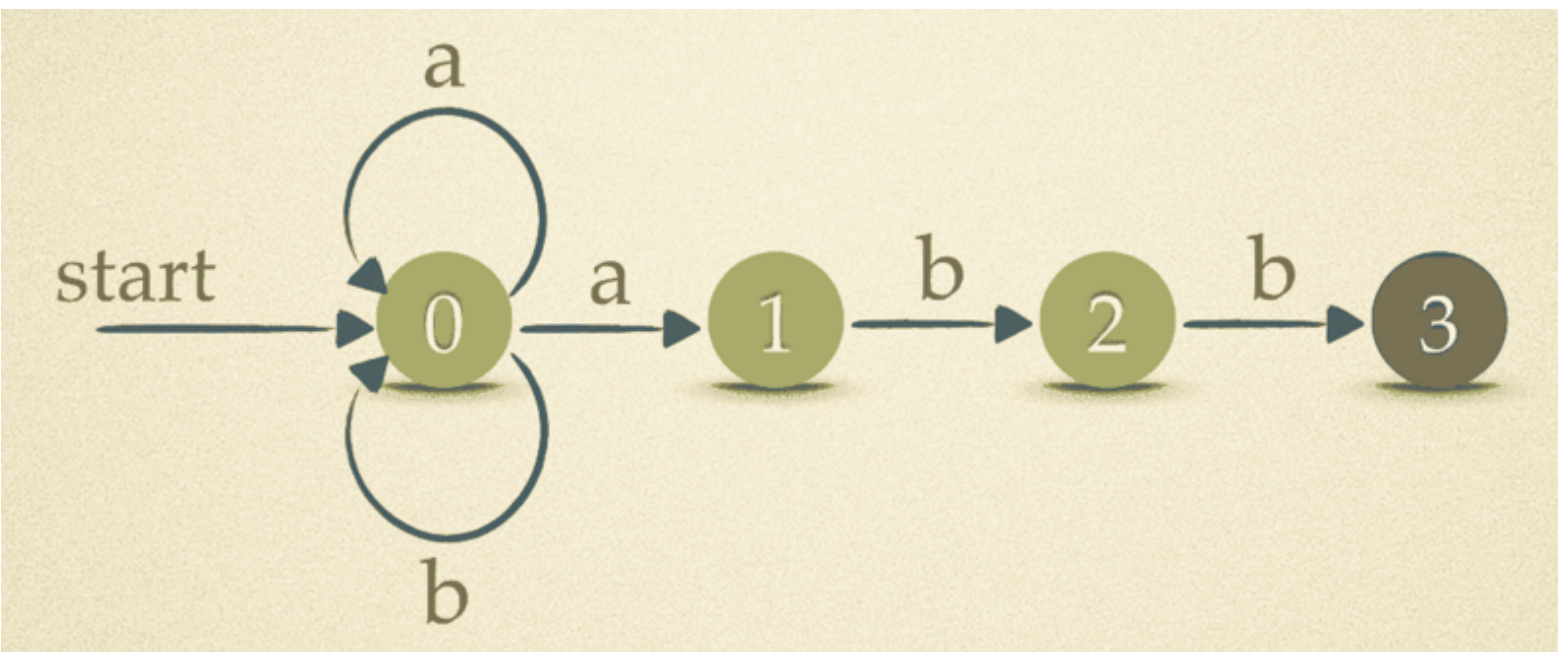
1 DFA_func = {0: {"a": 1, "b": 0},
2             1: {"a": 1, "b": 2},
3             2: {"a": 1, "b": 3},
4             3: {"a": 1, "b": 0}
5         }
6 input_symbol = ["a", "b"]
7 current_state = 0
8 accept_state = 3
9
10 strings = ["ababaaabb",
11            "ababcaabb",
  
```



```
12     "abbab"]
13 for string in strings:
14     for char in string:
15         if char not in input_symbol:
16             break
17         else:
18             current_state = DFA_func[current_state][char]
19
20 if current_state == 3:
21     print string, "---> Match!"
22 else:
23     print string, "--->No match!"
24 current_state = 0
25 """outputs:
26 ababaaabb ---> Match!
27 ababcaabb --->No match!
28 abbab --->No match!
29 """
```

上面的例子可以看出DFA识别语言简单直接，便于用程序实现，但是DFA较难从正则表达式直接转换。如果我们能找到一种表达方式，用以连接正则表达式和DFA，那么就可以让计算机识别正则表达式了。事实上，确实有这么一种表达方式，可以作为正则表达式和DFA的桥梁，而且很类似DFA，那就是非确定有限自动机(NFA)。

还是上面的例子，如果用NFA表示流程图，就如下图所示：



看上去很直观，很有 $[a|b]^*abb$ 的神韵。它转换为状态转换表如下：

状态	a	b
S0	S0, S1	S0
S1	Φ	S2
S2	Φ	S3
S3	Φ	Φ

NFA的定义与DFA区别不大， $M = (K, \Sigma, f, S, Z)$ ，其中：

1. K 是一个有穷集，它的每个元素称为一个状态；
2. Σ 是一个有穷字母表，它的每个元素称为一个输入符号， ϵ 表示输入为空，且 ϵ 不存在于 Σ ；
3. f 是转换函数，是在 $K \times \Sigma^* \rightarrow K$ 上的映射， Σ^* 说明存在遇到 ϵ 的情况， $f(k_i, a)$ 是一个多值函数；
4. $S \in K$ 是唯一的一个初态；
5. $Z \subseteq K$ 是一个状态集，为可接受状态或者结束状态。

数学上已经证明：

1. **DFA，NFA和正则表达式三者的描述能力是一样的。**
2. **正则表达式可以转换为NFA**，已经有成熟的算法实现这一转换。
3. **NFA可以转换为DFA**，也有完美的实现。

这里不做过多陈述，想了解详情可以参考《编译原理》一书。至此，计算机识别正则表达式的过程可以简化为：正则表达式→NFA→DFA。不过有时候NFA转换为DFA可能导致状态空间的指数增长，因此直接用NFA识别正则表达式。

正则表达式应用实例

前面已经使用python的re模块，简单展示了正则表达式[ab]*abb的匹配过程。下面将结合几个常用的正则表达式例子，展示正则表达式的强大之处。

开始之前，先来看下python中正则表达的一些规定。

1. `\w` 匹配单词字符，即`[a-zA-Z0-9_]`，`\W` 则恰好相反，匹配`[^a-zA-Z0-9_]`；
2. `\s` 匹配单个的空白字符：space, newline(`\n`), return(`\r`), tab(`\t`), form(`\f`)，即`[\n\r\t\f\v]`，`\S` 相反。
3. `\d` 匹配数字`[0-9]`，`\D` 恰好相反，匹配`[^0-9]`。
4. `(...P<name>...)` 会产生一个分组，在后面需要时可以用数组下标引用。
5. `(?P...)` 会产生命名组，需要时直接用名字引用。
6. `(?!...)` 当...不出现时匹配，这叫做后向界定符
7. `r" pattern"` 此时`pattern`为原始字符串，其中的`"\"` 不做特殊处理，`r" \n"` 匹配包含`"\"` 和`" n"` 两个字符的字符串，而不是匹配新行。当一个字符串是原始类型时，Python编译器不会对其尝试做任何的替换。关于原始字符串更多的内容可以看stackflow上问题[Python regex – r prefix](#)

python中常用到的正则表达式函数主要有`re.search`, `re.match`, `re.findall`, `re.sub`, `re.split`。

1. `re.findall`: 返回所有匹配搜索模式的字符串组成的列表；
2. `re.search`: 搜索字符串直到找到匹配模式的字符串，然后返回一个`re.MatchObject`对象，否则返回`None`；
3. `re.match`: 如果从头开始的一段字符匹配搜索模式，返回`re.MatchObject`对象，否则返回`None`。
4. `re.sub(pattern, repl, string, count=0, flags=0)`: 返回`repl`替换`pattern`后的字符串。
5. `re.split`: 在`pattern`出现的地方分割字符串。

`re.search`和`re.match`均可指定开始搜索和结束搜索的位置，即`re.search(string[, pos[, endpos]])`和`re.match(string[, pos[, endpos]])`，此时从`pos`搜索到`endpos`。需要注意的是，`match`总是从起始位置匹配，而`search`则从起始位置扫描直到遇到匹配。

`re.MatchObject`默认有一个boolean值`True`，`match()`和`search()`在没有找到匹配时均返回`None`，因此可以用简单的if语句判断是否匹配。

```
1 match = re.search(pattern, string)
2 if match:
3     process(match)
```

`re.MatchObject`对象主要有以下方法：`group([group1, ...])`和`groups([default])`。`group`返回一个或多个分组，`groups`返回包含所有分组的元组。

例子1：匹配Hello，当且仅当后面没有紧跟着World。

```
1 strings = ["HelloWorld!",
2            "Hello World!"]
3 import re
4 pattern = re.compile(r"Hello(?!World).*")
5 for string in strings:
6     result = pattern.search(string)
```

```

7     if result:
8         print string, "> ", result.group()
9     else:
10        print string, "> ", "Not match"
11
12 '''
13 HelloWorld! > Not match
14 Hello World! > Hello World!
15 '''

```

例子2：匹配邮箱地址。目前没有可以完美表达邮箱地址的正则表达式，可以看stackflow上问题[Using a regular expression to validate an email address](#)。这里我们用 `[w.-]+@[w-]+\.[w.-]+` 来简单地匹配邮箱地址。

```

1 content = """
2     alice@google.com
3     alice-bob@gmail..com gmail
4     alice.bob@apple.com apple
5     alice.bob@gmailcom invalid gmail
6 """
7 import re
8 address = re.compile(r'[w.-]+@[w-]+\.[w.-]+')
9 print address.findall(content)
10 '''

```

例子3：给函数添加装饰器。

```

1 original = """
2 def runaway():
3     print "running away..."
4 """
5 import re
6 pattern = re.compile(r"def (\w+\(\\):)")
7 wrapped = pattern.sub(r"@get_car\\ndef \\1", original)
8 print original, "---->", wrapped, "----"
9 """outputs
10 def runaway():
11     print "running away..."
12 ---->
13 @get_car
14 def runaway():
15     print "running away..."
16 ----
17 """

```

看起来正则表达式似乎无所不能，但是并不是所有的场合都适合用正则表达式，许多情况下我们可以找到替代的工具。比如我们想解析一个html网页，这时候应该使用使用 HTML 解析器，stackflow上有一个[答案](#)告诉你此时为什么不要使用正则表达式。python有很多html解析器，比如：

- [BeautifulSoup](#) 是一个流行的第三方库
- [lxml](#) 是一个功能齐全基于 c 的快速的库

[博客中文章](#)已更新，修复一些错误。 更多文章移步博客： <http://selfboot.cn/>

参考

- [Wiki: Regular expression](#)
- [正则表达式和有限状态机](#)
- [Python Regular Expressions](#)
- [Python check for valid email address?](#)
- [Python正则表达式的七个使用范例](#)

打赏支持我写出更多好文章，谢谢！

¥ 打赏作者

👍 2 赞

🔖 20 收藏

💬 6 评论



关于作者：selfboot



热爱计算机技术的学生...书中寻求心灵的平静...selfboot，自启动，只有自己能启动自己所以不要寄希望于别人，自我蜕变展翅飞翔吧！
[👤 个人主页](#) · [📁 我的文章](#) · [🎓 33](#) · [🔗](#) [🔔](#) [📢](#) [🐦](#)

相关文章

- [用 Python 的魔术方法做出更好的正则表达式 API](#)
- [Python 正则模式中 search\(\) 和 match\(\) 有什么区别？](#)
- [Python正则表达式学习摘要及资料](#)
- [python-文本处理和正则表达式 · 1](#)

可能感兴趣的话题


- [DPDK如今都有哪些项目使用到了 · 1](#)
- [想组建一个关于深度学习的小组，不知道有... · 4](#)
- [春涌，程序员](#)
- [幸运数](#)
- [张江爱情故事3： b/s模式和c/s模式](#)
- [如何在github使用svg图片？](#)

登录后评论

新用户注册

直接登录    

最新评论

 [Rif.南理汉子](#) 10/07

还是没看懂...(逃.....)

[👍 赞](#) [👤 回复](#)



selfboot (33 · 0 0 0 0)

10/19

不懂就再看

👍 赞 回复 ↩



郭淮 (1 · 0)

10/09

楼主，[a|b]的写法匹配a或b是有问题的，[a|b]是匹配a或|或b

👍 赞 回复 ↩



selfboot (33 · 0 0 0 0)

10/19

多谢指出!!!抱歉才看到!

👍 赞 回复 ↩



Mr.Riaht

10/19

首页 资讯 文章 频道 ✕ 资源 小组 ♡ 相亲

频道 ✕

🔑 登录

👤 注册



第一次看到把正则表达式讲的这么详细的文章

👍 赞 回复 ↩



selfboot (33 · 0 0 0 0)

10/19

多谢支持

👍 赞 回复 ↩



Python小组话题

我有新话题 💬



[一个noob默默来问问题了 \(Python相...](#)
[iPixelOldC](#) 发起 • 7 回复



[关于比较运算符的问题](#)
[NYNL](#) 发起



[Python使用multiprocessing库时，队列...](#)
[隔夜茶](#) 发起



[python3 和python2 hashlib同样的代码...](#)
[疯狂的兔子](#) 发起 • 10 回复



[python图片抓取的问题](#)
[wx_tDz3wzoQ](#) 发起



[前辈们指点一下python 的入门学习](#)
[So.](#) 发起 • 3 回复

程序员专属
极客卫衣

第二件减12

- [本周热门Python文章](#)
- [本月热门](#)
- [热门标签](#)

- [0 \[Ubuntu+Django+Nginx+uWSGI+Mysql搭...\]\(#\)](#)
- [1 \[一起读 Gevent 源码\]\(#\)](#)
- [2 \[Python开发者又一次扩大圈内影响力...\]\(#\)](#)
- [3 \[理解 Python 的 LEGB\]\(#\)](#)
- [4 \[从零开始搭建论坛（3）：Flask框架...\]\(#\)](#)
- [5 \[使用 Python 进行分布式系统协调\]\(#\)](#)
- [6 \[说说Python中的闭包\]\(#\)](#)
- [7 \[Python 装饰器之 functools.wraps\]\(#\)](#)
- [8 \[Python中不尽如人意的断言Assertion\]\(#\)](#)
- [9 \[详解Python的装饰器\]\(#\)](#)





[cclib](#)：一个用来解析和解释计算化学软件包输出结果的库

科学计算与分析



[Open Mining](#)：使用 Python 挖掘商业情报 (Pandas W...

科学计算与分析



[bpython](#)：界面丰富的Python解析器

交互式解析器



[django-elastic-transcoder](#)：Django + Amazon Elastic T...

音频



[MRQ](#)：一个 Python 的分布式 worker 任务队列，使用 Red...

Queue



最新评论

-  Re: [Spark for python developers ...](#)
尝试一下...
-  Re: [一行 Python 代码](#)
有意思
-  Re: [一行 Python 代码](#)
额，好吧。我也是半吊子全栈。
-  Re: [一行 Python 代码](#)
学习了



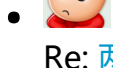
Re: [两个命令把 Vim 打造成 Pyth...](#)

查了下，用NERDTree可以调出文件路径。其他问题依然没有答案，希望能得到答案，减少探索时间，谢谢...



Re: [一行 Python 代码](#)

一行代码更多是对python 的列表推导，匿名函数lambda 等多重使用，提高对这些技能的理解，纯...



Re: [两个命令把 Vim 打造成 Pyth...](#)

觉得很好用。有几个问题：1、如何去掉那条竖线？因为我在mac下都是全屏编辑，如果按照pep的规则，那...



Re: [一行 Python 代码](#)

额？所谓一行代码？.....还不如写一个库，然后就 something.run();

[首页](#) [资讯](#) [文章](#) [频道](#) [资源](#) [小组](#) [❤ 相亲](#)

[频道](#) [登录](#)

[注册](#)



关于 Python 频道

Python频道分享 Python 开发技术、相关的行业动态。

快速链接

[网站使用指南](#) »

[加入我们](#) »

[问题反馈与求助](#) »

[网站积分规则](#) »

[网站声望规则](#) »

关注我们

新浪微博：[@Python开发者](#)

RSS：[订阅地址](#)

推荐微信号



Python开发者



Linux爱好者



数据库开发

合作联系

Email：bd@jobbole.com

QQ：2302462408（加好友请注明来意）

更多频道

[小组](#) – 好的话题、有启发的回复、值得信赖的圈子

[头条](#) – 分享和发现有价值的内容与观点

[相亲](#) – 为IT单身男女服务的征婚传播平台

[资源](#) – 优秀的工具资源导航

[翻译](#) – 翻译传播优秀的外文文章

[文章](#) – 国内外的精选文章

[设计](#) – UI,网页，交互和用户体验

[iOS](#) – 专注iOS技术分享

[安卓](#) – 专注Android技术分享

[前端](#) – JavaScript, HTML5, CSS

[Java](#) – 专注Java技术分享

[Python](#) – 专注Python技术分享

