目录

[Python 学习笔记 1](#_Toc501959728)

[正则表达式 1](#_Toc501959729)

[编译 2](#_Toc501959730)

# Python 学习笔记

正则表达式

用\d可以匹配一个数字，\w可以匹配一个字母或数字

用\*表示任意个字符（包括0个），用+表示至少一个字符，用?表示0个或1个字符，用{n}表示n个字符，用{n,m}表示n-m个字符

* [0-9a-zA-Z\\_]可以匹配一个数字、字母或者下划线；
* [0-9a-zA-Z\\_]+可以匹配至少由一个数字、字母或者下划线组成的字符串，比如'a100'，'0\_Z'，'Py3000'等等；
* [a-zA-Z\\_][0-9a-zA-Z\\_]\*可以匹配由字母或下划线开头，后接任意个由一个数字、字母或者下划线组成的字符串，也就是Python合法的变量；
* [a-zA-Z\\_][0-9a-zA-Z\\_]{0, 19}更精确地限制了变量的长度是1-20个字符（前面1个字符+后面最多19个字符）。

A|B可以匹配A或B，所以(P|p)ython可以匹配'Python'或者'python'。

^表示行的开头，^\d表示必须以数字开头。

$表示行的结束，\d$表示必须以数字结束。

你可能注意到了，py也可以匹配'python'，但是加上^py$就变成了整行匹配，就只能匹配'py'了。

除了简单地判断是否匹配之外，正则表达式还有提取子串的强大功能。用()表示的就是要提取的分组（Group）

>>> m = re.match(r'^(\d{3})-(\d{3,8})$', '010-12345')

>>> m

<\_sre.SRE\_Match object at 0x1026fb3e8>

>>> m.group(0)

'010-12345'

>>> m.group(1)

'010'

>>> m.group(2)

'12345'

正则匹配默认是贪婪匹配，也就是匹配尽可能多的字符。举例如下，匹配出数字后面的0：

>>> re.match(r'^(\d+)(0\*)$', '102300').groups()

('102300', '')

由于\d+采用贪婪匹配，直接把后面的0全部匹配了，结果0\*只能匹配空字符串了。

必须让\d+采用非贪婪匹配（也就是尽可能少匹配），才能把后面的0匹配出来，加个?就可以让\d+采用非贪婪匹配：

>>> re.match(r'^(\d+?)(0\*)$', '102300').groups()

('1023', '00')

编译

当我们在Python中使用正则表达式时，re模块内部会干两件事情：

1. 编译正则表达式，如果正则表达式的字符串本身不合法，会报错；
2. 用编译后的正则表达式去匹配字符串。

如果一个正则表达式要重复使用几千次，出于效率的考虑，我们可以预编译该正则表达式，接下来重复使用时就不需要编译这个步骤了，直接匹配：

>>> **import** re

*# 编译:*

>>> re\_telephone = re.compile(r'^(\d{3})-(\d{3,8})$')

*# 使用：*

>>> re\_telephone.match('010-12345').groups()

('010', '12345')

>>> re\_telephone.match('010-8086').groups()

('010', '8086')

编译后生成Regular Expression对象，由于该对象自己包含了正则表达式，所以调用对应的方法时不用给出正则字符串。

#### 常用第三方模块PIL

PIL的ImageDraw提供了一系列绘图方法，让我们可以直接绘图。比如要生成字母验证码图片：

**import** Image, ImageDraw, ImageFont, ImageFilter

**import** random

*# 随机字母:*

**def** **rndChar**():

**return** chr(random.randint(65, 90))

*# 随机颜色1:*

**def** **rndColor**():

**return** (random.randint(64, 255), random.randint(64, 255), random.randint(64, 255))

*# 随机颜色2:*

**def** **rndColor2**():

**return** (random.randint(32, 127), random.randint(32, 127), random.randint(32, 127))

*# 240 x 60:*

width = 60 \* 4

height = 60

image = Image.new('RGB', (width, height), (255, 255, 255))

*# 创建Font对象:*

font = ImageFont.truetype('Arial.ttf', 36)

*# 创建Draw对象:*

draw = ImageDraw.Draw(image)

*# 填充每个像素:*

**for** x **in** range(width):

**for** y **in** range(height):

draw.point((x, y), fill=rndColor())

*# 输出文字:*

**for** t **in** range(4):

draw.text((60 \* t + 10, 10), rndChar(), font=font, fill=rndColor2())

*# 模糊:*

image = image.filter(ImageFilter.BLUR)

image.save('code.jpg', 'jpeg');

我们用随机颜色填充背景，再画上文字，最后对图像进行模糊，得到验证码图片如下：

验证码

要详细了解PIL的强大功能，请请参考PIL官方文档：

<http://effbot.org/imagingbook/>

#### 网络编程

