python零星笔记

正则

1.直接给出字符，就是精确匹配。用\d可以匹配一个数字，\w可以匹配一个字母或数字，

2. .可以匹配任意字符。

3. 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：

4. '-'这样的是特殊字符，在正则表达式中，要用'\'转义。

4.\s可以匹配一个空格（也包括Tab等空白符），所以\s+表示至少有一个空格，例如匹配' '，' '等

5. [0-9a-zA-Z\\_]可以匹配一个数字、字母或者下划线。匹配的是一个。

[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；

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

7. ^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。

8. 我们强烈建议使用Python的r前缀，就不用考虑转义的问题了

9. 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）。比如：

^(\d{3})-(\d{3,8})$分别定义了两个组

10. 正则匹配默认是贪婪匹配，也就是匹配尽可能多的字符。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')

11. 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')

4、pd.DataFrame空值填充为0

round\_r1.fillna(0,inplace = True) #空值填充

5、np.random.choice（5，3）从【0，1，2，3，4】中选3个，有放回的选，返回值为列表

6/在编程的时候有时候我们想要比较 两种算法哪个更快 或者自己的代码哪一段最慢 这时候就可以使用 magic函数  
magic有行魔法%time 和单元魔法%%time  
行魔法 显示这一行代码运行的时长 单元魔法显示这一个cell 运行的时长

cputime=user用户态cpu耗时 +sys内核态cpu耗时  
wall time = real time 墙上钟表时间 包含堵塞等多种情况的实际运行时长

7、pwd显示当前路径

8、data['w']  #选择表格中的'w'列，使用类字典属性,返回的是Series类型

data[['w']]  #选择表格中的'w'列，返回的是DataFrame属性

9、**fillna()函数详解 fillna(0,inplace=True)**

inplace参数的取值：True、False

**True**：直接修改原对象

**False**：创建一个副本，修改副本，原对象不变（缺省默认）

10、awk‘{print $4,$8}‘ 功能是打印第4和8列，以空格间隔

11、ps –ef |grep 命令

命令拆解：

ps:将某个进程显示出来  
-A 　显示所有程序。   
-e 　此参数的效果和指定"A"参数相同。  
-f 　显示UID,PPIP,C与STIME栏位。   
grep命令是查找  
中间的|是管道命令 是指ps命令与grep同时执行  
这条命令的意思是显示有关Apachejetspeed有关的进程

12、make是用来编译的，它从Makefile中读取指令，然后编译。

make install是用来安装的，它也从Makefile中读取指令，安装到指定的位置。

13、which命令 查看可执行文件的位置。