**Pandas 读取txt**，在txt文件中，有时候解析的数据并非逗号或者分号分隔，对于此种情况，正则表达式就能派上用场。在tead\_table()函数中，可以使用sep选项指定正则表达式。

**正则表达式**

为了更好理解正则表达式的用法，以及用它分割多个元素的方法，先从一个简单例子入手。  
假如有一个txt文件，它里面的元素是以空格或制表符分隔的，且没有规律，在这种情况下，就可以用正则表达式兼顾两种分隔符。可以使用通配符\s\*，\s匹配空格和制表符，’\*’表示1个或者多个字符。更多正则表达式的知识可以参考网页<https://geek-docs.com/regexp>，常用元字符如下表所示:

| **通配符** | **描述** |
| --- | --- |
| . | 除换行符（\n、\r）之外的任何单个字符 |
| \d | 一个数字字符，等价于[0-9] |
| \D | 一个非数字字符，等价于 [^0-9] |
| \s | 任何空白字符，包括空格、制表符、换页符等等，等价于 [\f\n\r\t\v] |
| \S | 任何非空白字符，等价于 [^ \f\n\r\t\v] |
| \n | 换行符 |
| \t | 制表符 |
| \uxxxx | 用十六进制数字xxxx表示的Unicode字符 |

如果所有的元素随机以制表符或空格分隔，顺序随机，如工作目录中存放文件data1.txt信息如下:  


import pandas as pd

df=pd.read\_table('data1.txt', sep='\s+')

print(df)

执行结果如下：

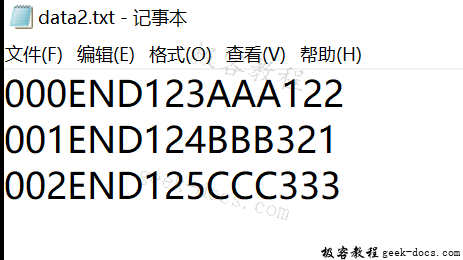
white red blue green

0 1 5 2 3

1 2 7 8 5

2 3 3 6 7

如上所示，我们得到一个完美的DataFrame对象，所有的元素均处在正确的位置。

通常情况把逗号、空格和制表符等看作分隔符，但是实际应用中，字母数字组合或者整数均可作为分隔符，比如数字0。接下来这个例子中，txt文件中的数字和字母杂糅在一起，你需要从中抽取数字部分。若txt文件中的数据无表头，记得将header选项设置为None。如下所示:  


import pandas as pd

df=pd.read\_table('data2.txt', sep='\D+', header=None)

print(df)

输出结果如下:

0 1 2

0 0 123 122

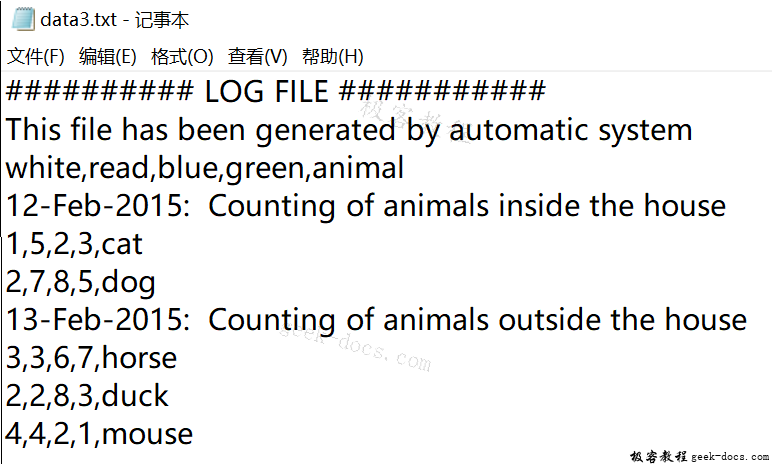
1 1 124 321

2 2 125 333

**skiprows**

另外一种常见情况是，解析数据时把空行排除在外，文件中的表头或没有必要的注释也用不到。使用skiprows选项，可以排除多余的行，把要删除的行和行号放到数字中，赋值给该参数即可。例如:

* skiprows=5：排除前五行
* skiprows=[5]：排除第五行



import pandas as pd

df=pd.read\_table('data3.txt', sep=',', skiprows=[0,1,3,6])

print(df)

输出结果如下:

white read blue green animal

0 1 5 2 3 cat

1 2 7 8 5 dog

2 3 3 6 7 horse

3 2 2 8 3 duck

4 4 4 2 1 mouse