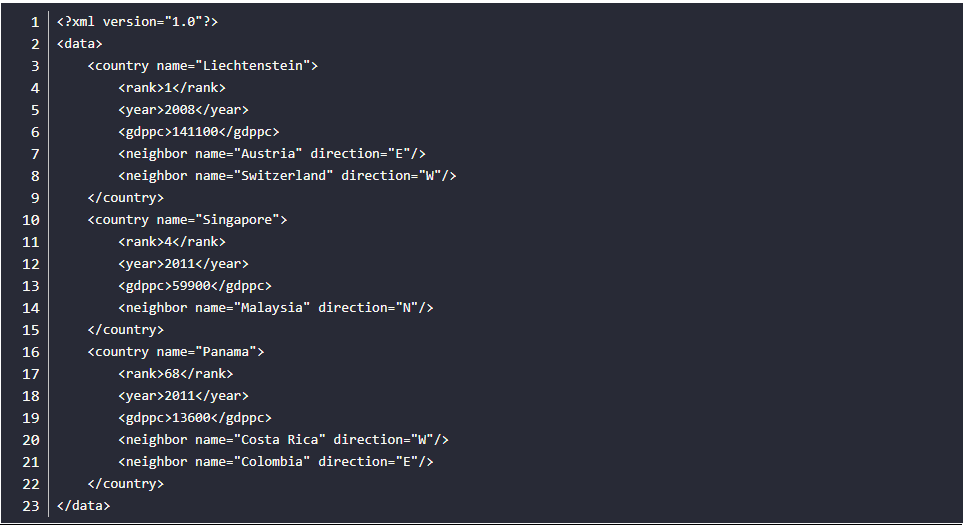
# XML tree and elements

XML（eXtensible Markup Language）指可扩展标记语言，被设计用来传输和存储数据。ElementTree生来就是为了处理XML，它在Python标准库中有两种实现：一种是纯Python实现的，如xml.etree.ElementTree，另一种是速度快一点的xml.etree.cElementTree。注意：尽量使用C语言实现的那种，因为它速度更快，而且消耗的内存更少。



# XML基本元素：

1. tag：获取节点名称，返回string对象；

2. text：获取元素的内容，返回string对象。

3. attrib：获取节点的属性，返回dict对象。

4. tail：获取元素闭合之后的尾迹，返回string对象。



ElementTree.parse：读取路径下的XML文件, 并返回ElementTree对象，输入参数为XML文件路径；

ElementTree.getroot：获取根节点。

ElementTree.iter：遍历Element下面的所有子树(它的子树，它们的子树，等等)，返回迭代对象；

ElementTree.findall：在Element的下一级子节点中找出与括号里面的tag匹配的元素，参数支持xpath方式”./.../...”；

ElementTree.find：在Element的下一级子节点找到第一个带有括号内特定标签tag的子元素；

ElementTree.text：访问节点元素内容；

ElementTree.get：访问节点元素属性，也就是那个字典的键值；

# XML文件结构及基本语法

XML文件的结构性内容，包括节点关系以及属性内容等等。

元素是组成XML的最基本的单位，它由开始标记，属性和结束标记组成。

#--------------------------------------------------------------------------------------------------------------------------------------------------------------#

　　现在我们暂且使用“记事本”来创建我们的xml文件吧。先看一个xml文件：

　　〈?xml version="1.0" encoding="gb2312" ?〉

　　〈参考资料〉

　　 〈书籍〉

　　 〈名称〉xml入门精解〈/名称〉

　　 〈作者〉张三〈/作者〉

　　 〈价格 货币单位="人民币"〉20.00〈/价格〉

　　 〈/书籍〉

　　 〈书籍〉

　　 〈名称〉xml语法〈/名称〉

　　 〈!--此书即将出版--〉

　　 〈作者〉李四〈/作者〉

　　 〈价格 货币单位="人民币"〉18.00〈/价格〉

　　 〈/书籍〉

〈/参考资料〉

#--------------------------------------------------------------------------------------------------------------------------------------------------------------#

　xml文件可分为文件序言（prolog）和文件主体两个大的部分。

1. xml解析器

解析器的主要功能就是检查xml文件是否有结构上的错误，剥离xml文件中的标记，读出正确的内容，以交给下一步的应用程序处理。xml是一种用来结构化文件信息的标记语言，xml规范中对于如何标记文件的结构性有一个详细的法则，解析器就是根据这些法则写出来的软件。

2. xml文件需满足的准则

第一，xml文件的第一行必须是声明该文件是xml文件以及它所使用的xml规范版本。在文件的前面不能够有其它元素或者注释。

第二，在xml文件中有且只能够有一个根元素。例如〈参考资料〉... 〈/参考资料〉就是此xml文件的根元素。

第三，在xml文件中的标记必须正确地关闭，也就是说，在xml文件中，控制标记必须有与之对应的结束标记。如：〈名称〉标记必须有对应的〈/名称〉结束标记，不像html，某些标记的结束标记可有可无。如果在xml文件中遇到自成一个单元的标记，就是类似于html 中的〈img src=.....〉的这些没有结束标记的时候，xml把它称为“空元素”，必须用这样的写法：〈空元素名/〉，如果元素中含有属性时写法则为：〈空元素 名 属性名=“属性值”/〉。

第四，标记之间不得交叉。在以前的html文件中，可以这样写：〈b〉〈h〉xxxxxxx〈/b〉〈/h〉,〈b〉和〈h〉标记之间有相互重叠的区域，而在xml中，是严格禁止这样标记交错的写法，标记必须以规则性的次序来出现。

第五，属性值必须要用“ ”号括起来。如第一个例子中的“1.0”、“gb2312”、“人民币”。都是用“ ”号括起来了的，不能漏掉。

第六，控制标记、指令和属性名称等英文要区分大小写。而html不区分。

第七，我们知道，在html文件中，如果我们要浏览器原封不动地将我们所输入的东西显示出来，可以将这些东西放到〈pre〉xxx〈/pre〉或者〈xmp〉xxx〈/xmp〉标记中间，而在xml中，要实现这样的功能，就必须使用 cdata标记。在cdata标记中的信息被解析器原封不动地传给应用程序，并且不解析该段信息中的任何控制标记。cdata区域是由：“〈! [cdata[”为开始标记，以“]]〉”为结束标记。例如：例2中的源码，除了“〈![cdata[”和“]]〉”符号，其余的内容解析器将原封不动地 交给下游的应用程序，即使cdata区域中的开始和结尾的空白以及换行字符等，都同样会被转交（注意cdata是大写的字符）。

3. xml文件的结构

a. XML 的声明：<?xml version=”1.0” standalone=”yes” encoding=”UTF-8”?>

这是一个XML处理指令。处理指令以 <? 开始，以 ?> 结束。<? 后的第一个单词是指令名，如xml, 代表XML声明。

version, standalone, encoding 是三个属性，属性是由等号分开的名称-数值对，等号左边是特性名称，等号右边是特性的值，用引号引起来。

version: 说明这个文档符合1.0规范；standalone: 用于说明文档在这一个文件里是否还需要从外部导入, standalone 的值设为yes 说明所有的文档都在这一文件里完成 ；encoding: 指文档字符编码

b. XML根元素定义

XML文档的树形结构要求必须有一个根元素。根元素的起始标记要放在所有其它元素起始标记之前，根元素的结束标记根放在其它所有元素的结束标记之后，如

<?xml version=”1.0” standalone=”yes” encoding=”UTF-8”?>

<Settings>

<Person>Zhang San</Person>

</Settings>

c. XML元素

元素的基本结构由开始标记，数据内容，结束标记组成，如

<Person>

<Name>Zhang San</Name>

<Sex>Male</Sex>

</Person>

需要注意的是：元素标记区分大小写；结束标记必须有反斜杠；XML元素标记命名规则如下；名字中可以包含字母，数字及其它字母；名字不能以数字或下划线开头；名字不能用xml开头；名字中不能包含空格和冒号

d. XML中的注释

XML中注释：<!-- this is comment -->

需要注意的是：注释中不要出现“--”或“-”；注释不要放在标记中；注释不能嵌套

e. PI (Processing Instruction)

PI 指 Processing Instruction, 处理指令。PI以“<?”开头，以“?>”结束，用来给下游的文档传递信息。

<?xml:stylesheet href=”core.css” type=”text/css” ?>

例子表明这个XML文档用core.css控制显示。

f. PCDATA

#PCDATA: specifies that an element will contain parsed character data.

举例说明PCDATA的用法, 其中movies.xml 存储电影内容数据，movies.dtd对movies.xml进行验证。

示例文件(movies.dtd)

<?xml version="1.0" encoding="GB2312"?>

<!ELEMENT movies (id, name, brief, time)>

<!ATTLIST movies type CDATA #REQUIRED>

<!ELEMENT id (#PCDATA)>

<!ELEMENT name (#PCDATA)>

<!ELEMENT brief (#PCDATA)>

<!ELEMENT time (#PCDATA)>

id, name, brief, time只能包含非标记文本(不能有自己的子元素)。

XML文件如下所示(movies.xml):

<?xml version="1.0" encoding="GB2312"?>

<!DOCTYPE movies SYSTEM "movies.dtd">

<movies type="动作片">

<id>1</id>

<name>致命摇篮</name>

<brief>李连杰最新力作</brief>

<time>2003</time>

</movies>

7 CDATA

CDATA用于需要把整段文本解释成纯字符数据而不是标记的情况。当一些文本中包含很多“<”,“>”,“&”,“””等字符而非标记时，CDATA会非常有用。

<Example>

<![CDATA[

<Person>

<Name>ZhangSan</Name>

<Sex>Male</Sex>

</Person>

]]>

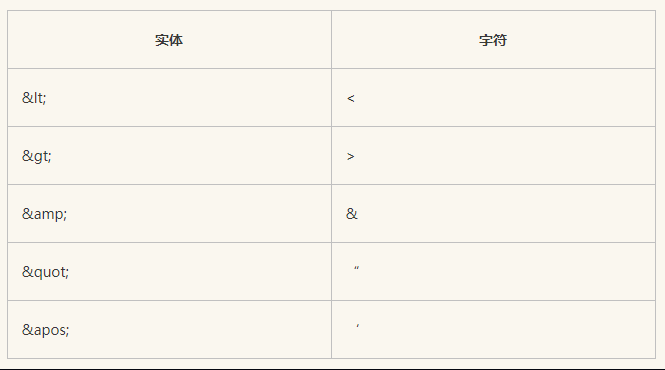
</Example>

以“<![CDATA[” 开始，以“]]>”结束。注意，在CDATA段中不要出现结束定界符“]]>”。

8 Entities(实体)

Entities(实体)是XML的存储单元，一个实体可以是字符串，文件，数据库记录等。实体的用处主要是为了避免在文档中重复输入，我们可以为一个文档定义一个实体名，然后在文档里引用实体名来代替这个文档，XML解析文档时，实体名会被替换成相应的文档。

XML为五个字符定义了实体名：



9 DOCTYPE

“<!DOCTYPE[]>”紧随XML声明，包括所有实体的声明，如下所示：

<!DOCTYPE example [

declare your entities here…

]>

<example>

Body of document…

</example>