XML Processing:

1.SAX - Simple API for XML. This works by reading XML little bit at a time and calling a method for each element.

2.DOM - Document Object Model. This works by reading in whole XML at once and creates internal representation of it

using native python classes.

In XML a node is the basic entity.

Every element, document, attribute, comment, text is a node.

API Comparison:

1)xml.dom : **This API is closer to DOM implementation. It provides all the basic elements**

**of DOM like Node, Element, Comment, Attribute, TextNode out-of-the box.**

**Anyone who is well aware of the DOM structure can use this API to process xml**

**in python.**

**For any element or doc text is a childNode of type TEXT\_NODE**

**This library many good API's such has importNode, next sibling, previous sibling,**

**lstChild, firstChild**

**Case sensitive search**

**Finding elements by tagname returns a list of all matching elements.**

**To get value/data/text of a node you have to iterate through that node**

**and print its childnode's nodeValue/data.**

**To add a node with text and attributes at runtime you need to create**

**a seperate node, then attach an attribute then a text node to it.**

**To remove a childNode you have to be the parent of that node**

**To insert a node you have to adjust many things in order to maintain the**

**formatting of the xml. Text nodes create a lot of problems.**

**While merging two xmls you need to take care of trailing text nodes otherwise you will**

**lose the formatting and xml won't look formatted after printing.**

**No XPATH support out-of-the-box. You need to install PyXML library**

**but this library is not maintained now.**

**2)ElementTree : This API tries to emulate DOM in terms of ElementTree and Element**

**This implementation is much closer to Python and users can process xmls using this**

**API.**

**For any Element text or tail is an argument.**

**You need not have a special text node to represent text in XML, text and tail**

**arguments are enough to represent any text contained inside XML document.**

**Element created by ElementTree act as python lists. So each subelement is like**

**an item inside list.**

**Single element can appear in multiple trees/xmls so a change in one tree**

**will reflect in other trees also.So you need to be casreful while appending**

**same tag to multiple trees.**

**Case sensitive search**

**There two methods for finding elements by tagName**

**find - returns first element that matches**

**findall - returns list of all elements that matches**

**To get value/data/text of a node you can use findtext method.**

**findtext accepts two types of inputs:**

**1. tagname : searches for tagname in childnodes and returns the text argument**

**of first matched node**

**2. XPATH : searches for given node and returns text argument of matched node**

**To add a node with text and attributes at runtime you can rightaway create**

**an element with attributes specified and you can easily set the text attribute**

**To remove an element you need to be at the parent of the element.**

**Insertion of node requires index. But there are no problems of text nodes or**

**new line nodes.**

**There is no direct API for replacing a node. You will have to remove a node and insert new**

**node at that index.**

**Merging two xmls disturbs the formatting but could not find any way to**

**retain the formatting.**

**There is a minimalistic XPATH support but it can be useful.**

**You can search elements based on its path address.**

**3)lxml:**

**This is pythonic library which is built up on libxml2 and libxslt.**

**Similar to ElementTree API this library tries to to provide many features**

**but by staying close to Python.**

**You can perform every operation that you could do with ElementTree by**

**just importing lxml.etree.** There are some extra operations supported by

lxml.etree

**copying element from one tree to other will delink the element from first tree**

**So you need to use deep copy for copying element from one tree to other.**

**lxml.etree provides getparent(an element has one one parent),**

**getprevious and getnext (to access siblings)**

**For recursively traversing xml you can use root.iter() which will go through**

**each and every child.**

**There are many useful APIs for iterating over xml like**

**tag.getroot, tag.getroottree, root.iterchildren, tag.itersiblings, tag.iterancestors, tag.iterdescendants,**

**tag.iterchildren**

**Merging of two xmls still has same problem.**