Introduction to XML

What is XML?

- XML stands for **Ex**tensible **M**arkup **L**anguage and is a text-based markup language derived from Standard Generalized Markup Language (SGML).
-

outline

- XML Basics,
- Advanced XML, and
- XML tools.

XML overview

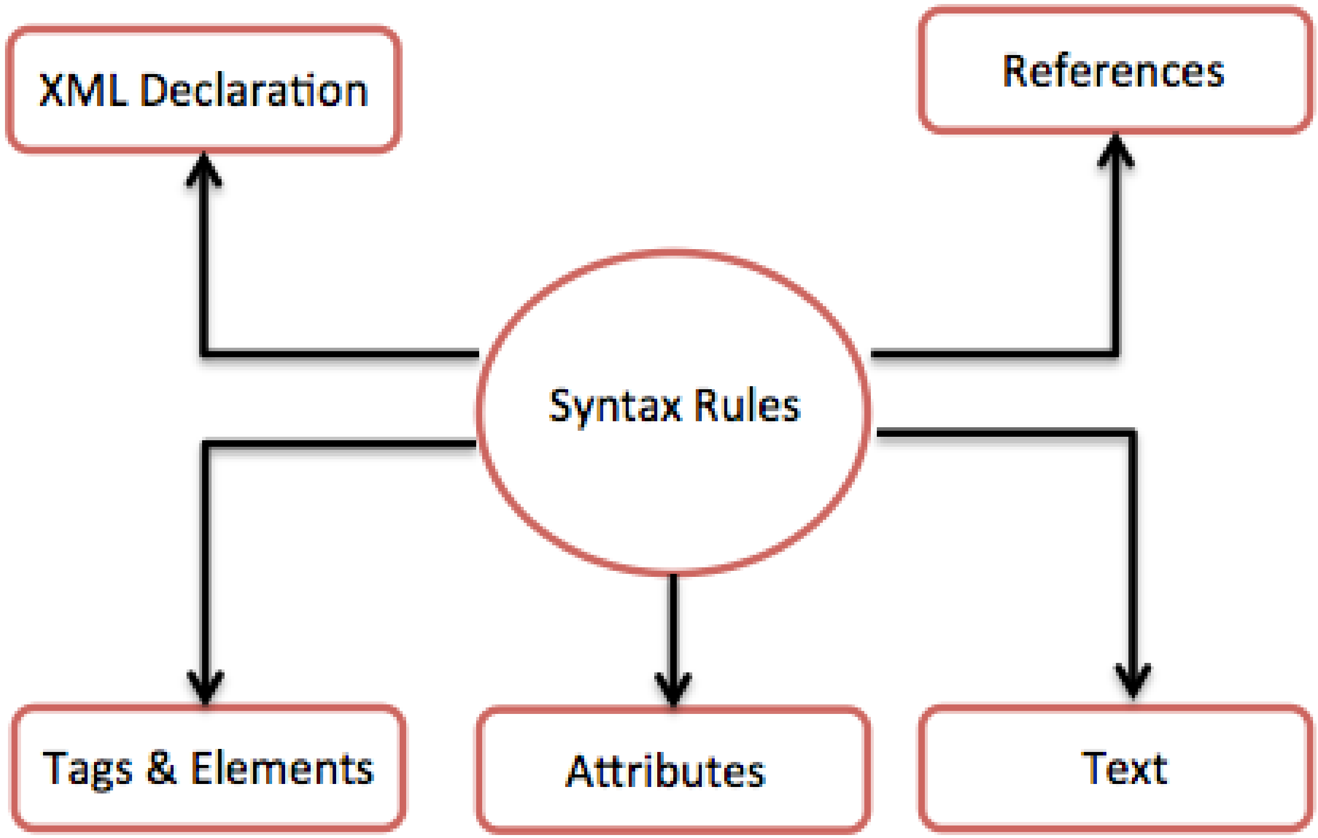
- XML tags identify the data and are used to store and organize the data, rather than specifying how to display it like HTML tags, which are used to display the data.
- XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.
- There are three important characteristics of XML that make it useful in a variety of systems and solutions −

Characteristics of XML

- **XML is extensible** − XML allows you to create your own self-descriptive tags, or language, that suits your application.
- **XML carries the data, does not present it** − XML allows you to store the data irrespective of how it will be presented.
- **XML is a public standard** − XML was developed by an organization called the World Wide Web Consortium (W3C) and is available as an open standard.

XML - Syntax

```
<?xml version = "1.0"?>
<contact-info>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</contact-info>
```


XML Declaration
References
Syntax Rules
Tags & Elements
Attributes
Text

```
<?xml version = "1.0" encoding = "UTF-8"?>
```

XML Declaration

The XML document can optionally have an XML declaration. It is written as follows:

<?xml version = "1.0" encoding = "UTF-8"?>

Where *version* is the XML version and *encoding* specifies the character encoding used in the document.

Syntax Rules for XML Declaration

- **The XML declaration is case sensitive and must begin with "<?xml>" where "xml" is written in lower-case.**
- **If document contains XML declaration, then it strictly needs to be the first statement of the XML document**
- **The XML declaration strictly needs be the first statement in the XML document.**
- **An HTTP protocol can override the value of *encoding* that you put in the XML declaration**

Tags and Elements

Syntax Rules for Tags and Elements:

Element Syntax – Each XML-element needs to be closed either with start or with end elements as shown below –

```
<element>....</element>
```

Nesting of Elements

- – An XML-element can contain multiple XML-elements as its children, but the children elements must not overlap. i.e., an end tag of an element must have the same name as that of the most recent unmatched start tag.

The Following example shows incorrect nested tags –

```
<?xml version = "1.0"?>
<contact-info>
 <company>Networks
</contact-info>
</company>
```

The Following example shows correct nested tags –

```
<?xml version = "1.0"?>
 <contact-info>
      <company>Networks</company>
<contact-info>
```

Root Element

– An XML document can have only one root element. For example, following is not a correct XML document, because both the **x** and **y** elements occur at the top level without a root element –

```
<x>...</x>

<y>...</y>
```

```
<root>

    <x>...</x>

    <y>...</y>

</root>
```

Case Sensitivity –

- The names of XML-elements are case-sensitive. That means the name of the start and the end elements need to be exactly in the same case.
- For example, **<contact-info>** is different from **<Contact-Info>**

XML Attributes

An **attribute** specifies a single property for the element, using a name/value pair. An XML-element can have one or more attributes. For example −

<a href = “http://uomosul.edu.iq”>Tutorialspoint!</a>

Here **href** is the attribute name and http://uomosul.edu.iq/ is attribute value.

Syntax Rules for XML Attributes

- Attribute names in XML (unlike HTML) are case sensitive. That is, *HREF* and *href* are considered two different XML attributes.
- Same attribute cannot have two values in a syntax. The following example shows incorrect syntax because the attribute *b* is specified twice

```
<a b = "x" c = "y" b = "z">....</a>
```

- Attribute names are defined without quotation marks, whereas attribute values must always appear in quotation marks. Following example demonstrates incorrect xml syntax

```
<a b = x>....</a>
```

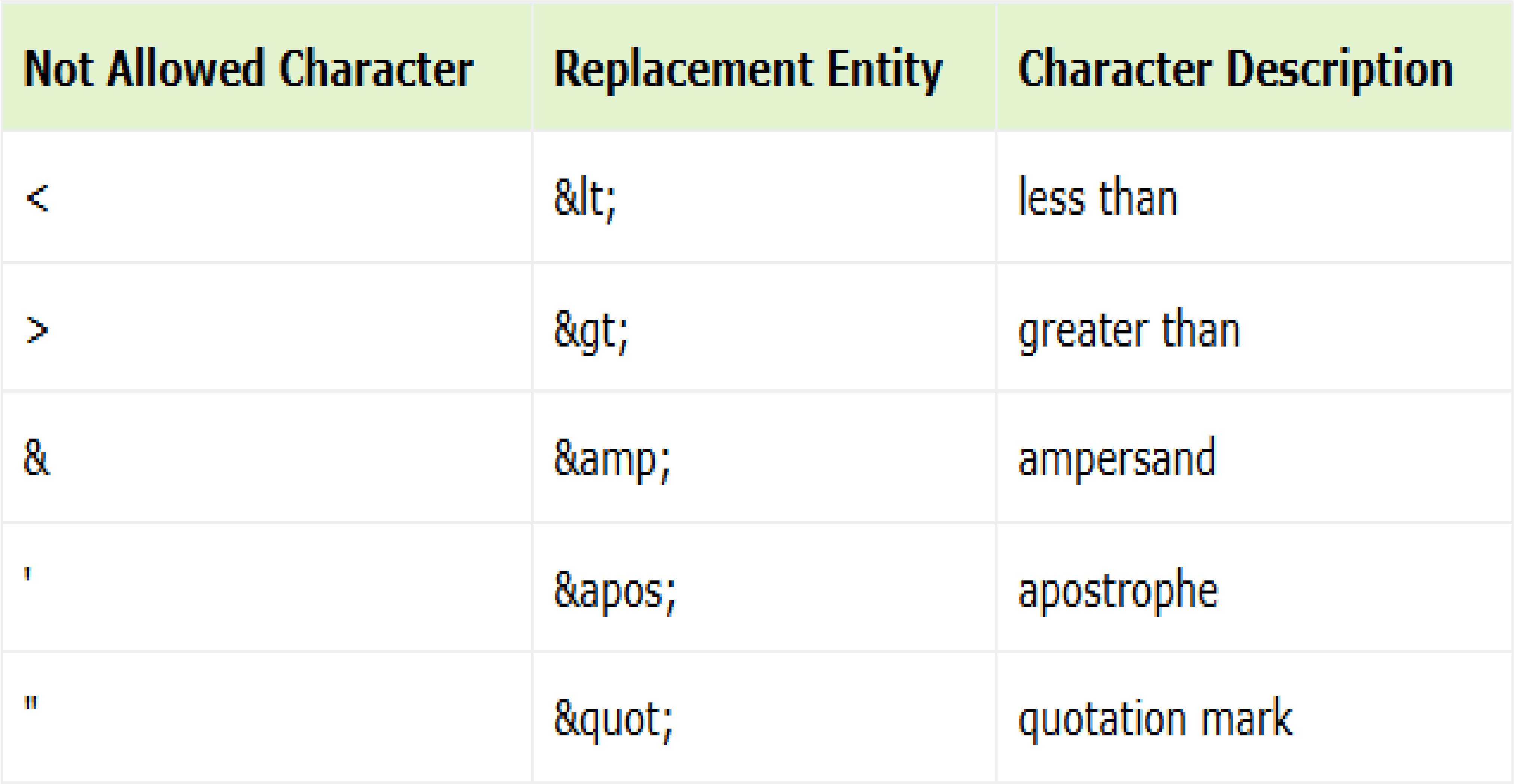
XML References

References usually allow you to add or include additional text or markup in an XML document. References always begin with the symbol "**&**“ which is a reserved character and end with the symbol ";". XML has two types of references −

- **Entity References** − An entity reference contains a name between the start and the end delimiters. For example **&amp;** where *amp* is *name*. The *name* refers to a predefined string of text and/or markup.
- **Character References** − These contain references, such as **&#65;**, contains a hash mark (“#”) followed by a number. The number always refers to the Unicode code of a character. In this case, 65 refers to alphabet "A".

XML Text

- The names of XML-elements and XML-attributes are case-sensitive, which means the name of start and end elements need to be written in the same case. To avoid character encoding problems, all XML files should be saved as Unicode UTF-8 or UTF-16 files.
- Whitespace characters like blanks, tabs and line-breaks between XML-elements and between the XML-attributes will be ignored.
- Some characters are reserved by the XML syntax itself. Hence, they cannot be used directly. To use them, some replacement-entities are used, which are listed below −



Not Allowed Character	Replacement Entity	Character Description
<	&lt;	less than
>	&gt;	greater than
&	&amp;	ampersand
'	&apos;	apostrophe
"	&quot;	quotation mark

XML - Tags

XML tags form the foundation of XML. They define the scope of an element in XML. They can also be used to insert comments, declare settings required for parsing the environment, and to insert special instructions.

We can broadly categorize XML tags as follows −Start Tag

- The beginning of every non-empty XML element is marked by a start-tag. Following is an example of start-tag −

```
<address>
```

End Tag

- Every element that has a start tag should end with an end-tag. Following is an example of end-tag −

```
</address>
```

Empty Tag

The text that appears between start-tag and end-tag is called content.

An element which has no content is termed as empty.

An empty element can be represented in two ways as follows –

1. A start-tag immediately followed by an end-tag as shown below –

```
<hr></hr>
```

2. A complete empty-element tag is as shown below –

```
<hr />
```

3. Empty-element tags may be used for any element which has no content.

XML Tags Rules

Rule 1

XML tags are case-sensitive.

Following line of code is an example of wrong syntax </Address>, because of the case difference in two tags, which is treated as erroneous syntax in XML.

```
<address>This is wrong syntax</Address>
```

- Following code shows a correct way, where we use the same case to name the start and the end tag.

```
<address>
This is correct syntax
</address>
```

Rule 2

XML tags must be closed in an appropriate order, i.e., an XML tag opened inside another element must be closed before the outer element is closed. For example –

```
<outer_element>
    <internal_element>
        This tag is closed before the outer_element
    </internal_element>
</outer_element>
```

XML-Elements

- **XML elements** can be defined as building blocks of an XML. Elements can behave as containers to hold text, elements, attributes, media objects or all of these.
- Each XML document contains one or more elements, the scope of which are either delimited by start and end tags, or for empty elements, by an empty-element tag.

```
<element-name attribute1 attribute2>
 ....content
</element-name>
```

- **element-name** is the name of the element.

The *name* its case in the start and end tags must match.

- **attribute1, attribute2** are attributes of the element separated by white spaces. An attribute defines a property of the element. It associates a name with a value, which is a string of characters. An attribute is written as −

Name = “value”

name is followed by an = sign and a string *value* inside double(" ") or single(' ') quotes.

Empty Element

An empty element (element with no content) has following syntax –

```
<name attribute1 attribute2.../>
```

example (various XML element

```
<?xml version = "1.0"?>
<contact-info>
    <address category = "residence">
        <name>Ahmed Ali</name>
        <company>Mosul Co.</company>
        <phone>(077) 123-4567</phone>
    </address>
</contact-info>
```

XML Elements Rules

Following rules are required to be followed for XML elements –

An element *name* can contain any alphanumeric characters. The only punctuation mark allowed in names are the hyphen (-), under-score (_) and period (.).

Names are case sensitive. For example, Address, address, and ADDRESS are different names.

Start and end tags of an element must be identical.

An element, which is a container, can contain text or elements as seen in the above example.

name = "value"

XML - Attributes

Attributes are part of XML elements. An element can have multiple unique attributes. Attribute gives more information about XML elements. To be more precise, they define properties of elements. An XML attribute is always a name-value pair.

Syntax

```
<element-name attribute1 attribute2 >
 ....content..
< /element-name>
```

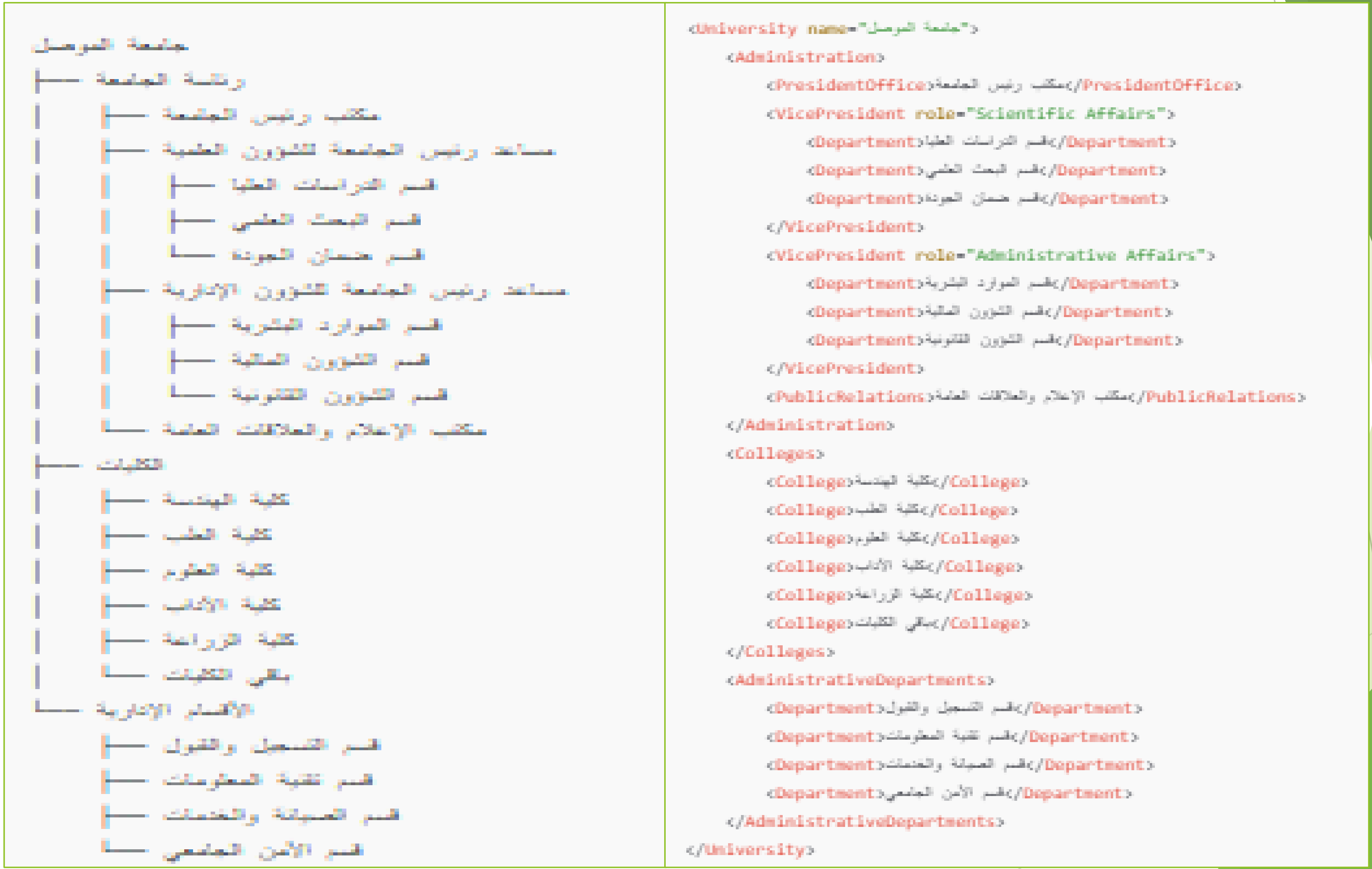
where *attribute1* and *attribute2* has the following form –

name = "value"

value has to be in double (" ") or single (' ') quotes. Here, *attribute1* and *attribute2* are unique attribute labels.

Attributes are used to add a unique label to an element, place the label in a category, add a Boolean flag, or otherwise associate it with some string of data. Following example demonstrates the use of attributes −

Example: University of Mosul



```
جامعة الموصل
├── رئاسة الجامعة
│   ├── مكتب رئيس الجامعة
│   ├── مساعد رئيس الجامعة للشؤون العلمية
│   │   ├── قسم الدراسات العليا
│   │   ├── قسم البحث العلمي
│   │   └── قسم ضمان الجودة
│   ├── مساعد رئيس الجامعة للشؤون الإدارية
│   │   ├── قسم الموارد البشرية
│   │   ├── قسم الشؤون المالية
│   │   └── قسم الشؤون القانونية
│   └── مكتب الإعلام والعلاقات العامة
├── الكليات
│   ├── كلية الهندسة
│   ├── كلية الطب
│   ├── كلية العلوم
│   ├── كلية الآداب
│   ├── كلية الزراعة
│   └── باقي الكليات
└── الأقسام الإدارية
    ├── قسم التسجيل والقبول
    ├── قسم تقنية المعلومات
    ├── قسم الصيانة والخدمات
    └── قسم الأمن الجامعي
```

```xml
<University name="جامعة الموصل">
    <Administration>
        <PresidentOffice>مكتب رئيس الجامعة</PresidentOffice>
        <VicePresident role="Scientific Affairs">
            <Department>قسم الدراسات العليا</Department>
            <Department>قسم البحث العلمي</Department>
            <Department>قسم ضمان الجودة</Department>
        </VicePresident>
        <VicePresident role="Administrative Affairs">
            <Department>قسم الموارد البشرية</Department>
            <Department>قسم الشؤون المالية</Department>
            <Department>قسم الشؤون القانونية</Department>
        </VicePresident>
        <PublicRelations>مكتب الإعلام والعلاقات العامة</PublicRelations>
    </Administration>
    <Colleges>
        <College>كلية الهندسة</College>
        <College>كلية الطب</College>
        <College>كلية العلوم</College>
        <College>كلية الآداب</College>
        <College>كلية الزراعة</College>
        <College>باقي الكليات</College>
    </Colleges>
    <AdministrativeDepartments>
        <Department>قسم التسجيل والقبول</Department>
        <Department>قسم تقنية المعلومات</Department>
        <Department>قسم الصيانة والخدمات</Department>
        <Department>قسم الأمن الجامعي</Department>
    </AdministrativeDepartments>
</University>
```

Xml manipulation in python

Experiment 1: Parse and Read XML

Objective: parse an XML string and read its elements.

```
import xml.etree.ElementTree as ET

# Sample XML string

xml_data = """<library>

    <book isbn="12345">

        <title>XML Basics</title>

        <author>John Doe</author>

    </book>

    <book isbn="67890">

        <title>Advanced XML</title>

        <author>Jane Smith</author>

    </book>

</library>"""
```

```
# Parse the XML string
root = ET.fromstring(xml_data)

# Print the root tag
print(f"Root tag: {root.tag}")

# Iterate over elements
for book in root.findall('book'):
    title = book.find('title').text
    author = book.find('author').text
    isbn = book.get('isbn')
    print(f"Title: {title}, Author: {author}, ISBN: {isbn}")
```

Xml manipulation in python

Experiment 2: Create and Write XML

Objective: Create an XML structure and save it to a file.

```
import xml.etree.ElementTree as ET


# Create root element

library = ET.Element("library")


# Add books

book1 = ET.SubElement(library,
"book", isbn="12345")

ET.SubElement(book1, "title").text =
"XML Basics"

ET.SubElement(book1,
"author").text = "John Doe"
```

```
book2 = ET.SubElement(library, "book", isbn="67890")
ET.SubElement(book2, "title").text = "Advanced XML"
ET.SubElement(book2, "author").text = "Jane Smith"

# Convert to a string
tree = ET.ElementTree(library)
tree.write("library.xml", encoding="utf-8",
xml_declaration=True)
print("XML file created: library.xml")
```

Xml manipulation in python

Experiment 3: Modify XML

Objective: Update existing elements or attributes in an XML file.

```
import xml.etree.ElementTree as ET
# Load XML from a file
tree = ET.parse("library.xml")
root = tree.getroot()
# Update author of the first book
first_book = root.find("book")
if first_book:
    first_book.find("author").text = "Updated Author"
# Add a new attribute to the second book
second_book = root.findall("book")[1]
second_book.set("genre", "Technology")
# Save changes back to the file
tree.write("library_updated.xml", encoding="utf-8", xml_declaration=True)
print("XML file updated: library_updated.xml")
```

Xml manipulation in python

Experiment 4: Delete Elements

Objective: Remove an element from the XML.

```
import xml.etree.ElementTree as ET
# Load XML from a file
tree = ET.parse("library_updated.xml")
root = tree.getroot()

# Remove the second book
second_book = root.find("book[@isbn='67890']")
if second_book:
    root.remove(second_book)

# Save changes back to the file
tree.write("library_deleted.xml", encoding="utf-8", xml_declaration=True)
print("Second book removed. File saved: library_deleted.xml")
```