XML

Introduction to XML

XML - XML Stands for eXtensible Markup Language

- XML is designed to send, store, receive and display data
- XML is platform and programming language independent
- unlinke HTML where most of the tags are predefined, XML doesn't have predefined tags

Features

- Extensible and human readable
- Preserves white spaces
- Overall Simplicity
- Separates data from HTML
- Undefined tags
- XML was designed to carry data
- Well Structured format
- Self descriptive in nature

Advantages of XML

- Platform Independent
- XML supports Unicode
- The data and transported using XML can be changed at any point of time
- XML is completely with many programming languages and 100% portable
- XML simplifies data sharing between various systems
- XML allows validation using DTD and Schema

Limitations of XML

- Verbose and Redundant compared to JSON
- High storage and transportation costs for large volumes of data
- XML doesn't support array
- XML file sizes are usually very large due to its verbose nature

Example ,

XML File, Before Altering "employee.xml"

```xml
<employees company="tesla">
    <employee id = 'ABC123'>
        <firstname>Rajath</firstname>
        <lastname>Kumar</lastname>
        <lastname>KS</lastname>
        <email>rajathkumarks@gmail.com</email>
        <title>Thought Leader</title>
    </employee>
    <employee id = 'ABC456'>
        <firstname>Elon</firstname>
        <lastname>Musk</lastname>
        <email>elon.musk@gmail.com</email>
        <title>CEO</title>
    </employee>
</employees>
```

Basics XML Operations with Python

Working with XML in Python typically involves parsing, modifying, and creating XML documents. Python provides several libraries for these tasks, with the most commonly used ones being:

1. **xml.etree.ElementTree** (part of the Python standard library)
2. **minidom**

1. Xml.etree.ElementTree

This is a simple and efficient module for parsing and creating XML data. It's part of the Python standard library.

```python
import xml.etree.ElementTree as ET

tree = ET.parse("employee.xml")
root = tree.getroot()

# Root
print(root.tag)

# Len of Root
print(len(root))
```

```python


# Access 1st Root Child
print(root[0].tag)

# Access len of 1st Root Child
print(len(root[0]))

# Loop over the Root Childeren
for child in root:
    print(child[0].text)
    print(child[1].text)
    print(child[2].text)
    print(child[3].text)

# Rajath
# Kumar
# rajathkumarks@gmail.com
# Thought Leader
# Elon
# Musk
# elon.musk@gmail.com
# CEO


# Root - Children - GrandChildren
print(root[0][0].tag)  # firstname
print(root[0][0].text)  # Rajath

print(root[1][2].text)  # elon.musk@gmail.com


# Get an Attribute
print(root.attrib)  # {'company': 'tesla'}


# Loop over 'email' Tag
for mailid in root.iter("email"):
    print(mailid.text)

# rajathkumarks@gmail.com
# elon.musk@gmail.com

# Find 1st Root Child with how many 'lastname' Tag
print(len(root[0].findall("lastname")))

# Printing the 1st root child with 'lastname' Tag
print(root[0].findall("lastname")[0].text)  # Kumar
print(root[0].findall("lastname")[1].text)  # KS

# Change the email address of Elon
root[1][2].text = "elon.musk@company.com"
```

```
63
64  # Remove the extra last name in the 1st root child
65  root[0].remove(root[0].findall("lastname")[1])
66  tree.write("xml_altered.xml")
67
```

XML File, After Altering "xml_altered.xml"

```
1   <employees company="tesla">
2       <employee id="ABC123">
3           <firstname>Rajath</firstname>
4           <lastname>Kumar</lastname>
5           <email>rajathkumarks@gmail.com</email>
6           <title>Thought Leader</title>
7       </employee>
8       <employee id="ABC456">
9           <firstname>Elon</firstname>
10          <lastname>Musk</lastname>
11          <email>elon.musk@company.com</email>
12          <title>CEO</title>
13      </employee>
14  </employees>
```

2. Minicom

The `minidom` module is another part of the Python standard library for working with XML. It's a minimal implementation of the Document Object Model (DOM) interface, which can be more convenient for certain tasks, especially if you're familiar with DOM.

Parsing XML,

```
1   from xml.dom import minidom
2
3   # Load and parse an XML file
4   dom = minidom.parse("employee.xml")
5
6   # Getting elements
7   elements = dom.getElementsByTagName("firstname")
8   for elem in elements:
9       print(elem.firstChild.data)
10
```

Creating XML,

```
1   from xml.dom.minidom import Document
2
3   doc = Document()
4
5   # Create elements
6   root = doc.createElement("root")
```

```
doc.appendChild(root)
child = doc.createElement("child")
root.appendChild(child)
text = doc.createTextNode("This is a child element")
child.appendChild(text)

# Write to a file
with open("new_file.xml", "w") as f:
    f.write(doc.toprettyxml())

# <?xml version="1.0" ?>
# <root>
#   <child>This is a child element</child>
# </root>

```