XML

Document Object Model (DOM) es una API (Application Programming Interface) que nos permite tanto ingresar, como editar cualquier tipo de documento que posea la extensión ".xml". De este modo, permite reestructurar un documento xml y presentarlo con una estructura de árbol, de igual manera, permite que el código del programador sea capaz de construir el documento con una estructura de árbol sin tener un documento previamente, esto quiere decir que, se iniciará desde cero a construir dicha estructura.

Cuando se habla de una aplicación de DOM, se dice que hay niveles en los cuales el nivel 1 no logra trabajar totalmente con los archivos xml. El nivel 2 proporciona mejoras significativas pero limitadas para el trabajo con los documentos. El nivel 3 permite agregar una especificación para cargar archivos, sin embargo, esta no se encuentra en la biblioteca estándar de Python.

Cuando hablamos de la creación, análisis y manejo de los documentos xml, podemos hablar de xPath xml de Python, la cual es técnicamente un módulo implementado con el fin de poder manejar y crear archivos con la extensión de datos xml. Con este módulo es posible utilizar dos clases, las cuales permiten ver el documento xml como un árbol o como un nodo del árbol en específico. Dichas clases son "Element" la cual permite ver un nodo del árbol, y "ElementTree" la cual permite ver el documento como un árbol completo

Librerías:

- Xml.dom: Esta solo da la definición de lo que es la API DOM.
- Xml.etree.ElementTree: Esta es un simple procesador de xml para el manejo de archivos en forma de árbol.
- Xml.dom.pulldom: Esta brinda un cierto soporte para la construcción de árboles.
- Xml.dom.minidom: Esta es una pequeña implementación de la API DOM.
- Xml.parsers.expat: Esta sirve como comunicación con el analizador Expat.

Ejemplos

Primer Ejemplo

```
Import xml.etree.ElementTree as ElementT
arbol = ElementT.parse('archivo.xml')
raíz_de_Arbol = tree.getroot()
```

Segundo Ejemplo

```
From xml.dom.minidom import parse, parseString
D1 = parse('C:\\user\documento.xml')
fuenteDatos = open('C:\\user\documento.xml')
```

Tercer Ejemplo

```
From xml.dom.minidom import getDOMImplementation
Implementacion = getDOMImplementation()

nuevoDocumento = implementación.createDocument(None, "some_tag", None)
elementos = nuevoDocumento.documentElement
texto = nuevoDocumento.createTextNode('Some textual content. ')
elementos.appendChild(text)
```

Cuarto Ejemplo

```
from xml.dom.pulldom import parse
from xml.sax import make_parser

analizador = make_parser()
```

Quinto Ejemplo

```
From xml.parsers.expat import ParserCreate, ExpatError, errors


Analizador = ParserCreate()
try:
        analizador.Parse(‘document.xml’)
except ExpatError as error:
        print(“Error:”, errors.messages[err.code])
```