Modulo DOM

Se le conoce como “DOM” a el Document Object Model el cual es la base de XML. Los documentos XML tienen una jerarquía de unidades llamadas “nodes”, la cual es una forma de describir los nodos y las relaciones entre ellos. Estas estructuras de información se forman a partir de un árbol dando la posibilidad de poder crear estructuras desde cero y acceso a estas mismas a través de un conjunto de objetos. Este pertenece a una librería la cual se importa de la siguiente manera:

```
Import xml.dom.minidom
```

Un documento DOM es una colección de datos los cuales poseen una jerarquía, esta jerarquía nos permite navegar por un árbol de información para poder buscar información específica.

Este también proporciona una API que permite a los desarrolladores agregar, editar, mover o eliminar los nodos de un árbol en cualquier momento con el objetivo de crear una aplicación.

Algunos objetos del DOM son los siguientes:

Interfaz	Propósito
DomImplementation	Interfaz para las implementaciones subyacentes
Node	Interfaz base para la mayoría de objetos en un documento
NodeList	Interfaz para una secuencia de nodos
DocumentType	Información acerca de la declaraciones necesarias para procesar un documento
Document	Objeto que representa un documento entero
Element	Nodos elemento en la jerarquía del documento
Attr	Nodos de los valores de los atributos en los elementos nodo
Comment	Representación de los comentarios en el documento fuente
Text	Nodos con contenido textual del documento
ProcessingInstruction	Representación de instrucción del procesamiento

Módulo Xpath

Xpath es un módulo que es parte de la librería xml.etree.ElementTree, esta librería se importa de la siguiente manera:

```
Import xml.etree.ElementTree as <nombre que se desea>
```

Xpath se define como un lenguaje que permite construir expresiones que recorren y procesan documentos XML. Este concepto es parecido al de las expresiones regulares las cuales se utilizan en Python para la búsqueda de valores sin el uso de parámetros. Por lo tanto, Xpath nos permite buscar y seleccionar información de un XML teniendo en cuenta su jerarquía.

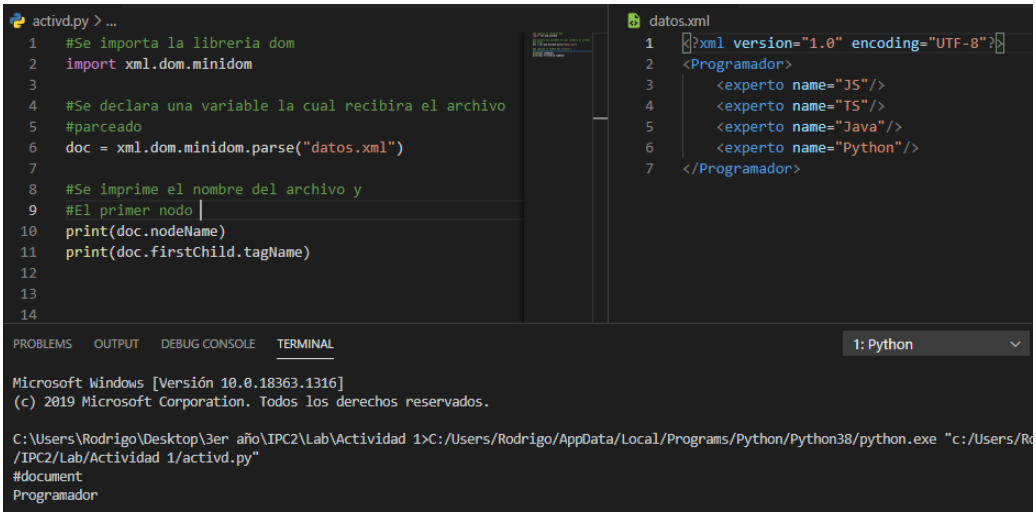
La forma en que XPath selecciona partes del documento XML se basa precisamente en la representación de árbol que se genera del documento. De hecho, los "operadores" de que consta este lenguaje recordarán la terminología que se utiliza a la hora de hablar de árboles en informática: raíz, hijo, ancestro, descendiente, etc. Esta terminología nos lleva a trabajar con nodos los cuales representan la jerarquía en el XML teniendo como ejemplo: los nodos raíz, nodos elemento, nodos texto, nodos atributo, nodos comentario, etc.

A continuación, se presenta la sintaxis de Xpath:

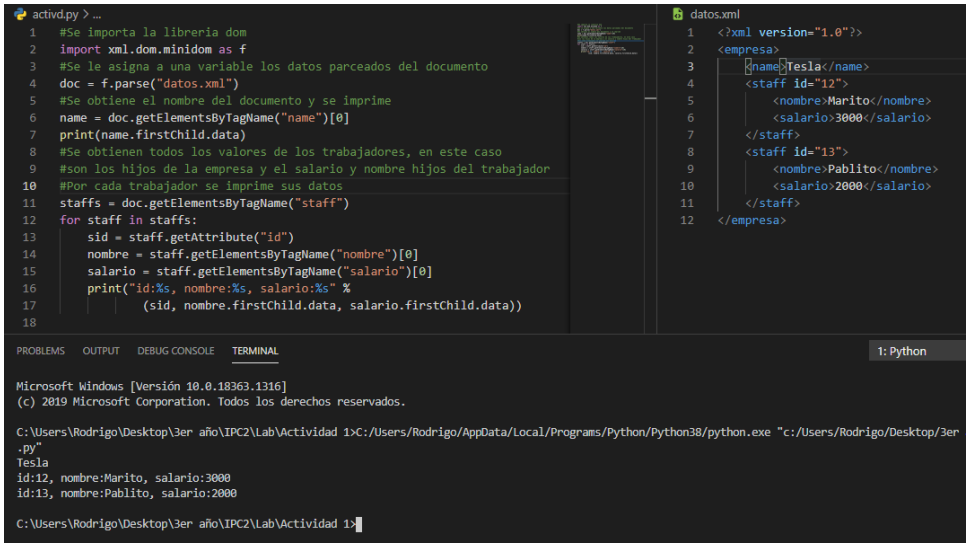
Sintaxis	Descripción
tag	Selecciona todos los elementos hijos contenidos en la etiqueta “tag”
*****	Selecciona todos los elementos hijos
.	Selecciona el nodo actual, este es muy usado en el inicio del path, para indicar que es un path relativo
//	Selecciona todos los sub elementos de todos los niveles bajo el nodo expresado
..	Selecciona el elemento padre
[@attrib]	Selecciona todos los elementos que contienen el atributo tras el “@”
[@attrib=’valor’]	eleccione todos los elementos para los cuales el atributo dado tenga un valor dado, el valor no puede contener comillas
[tag]	Selecciona todos los elementos que contienen una etiqueta hijo llamada tag. Solo los hijos inmediatos son admitidos
[tag=’text’]	Selecciona todos los elementos que tienen una etiqueta hijo llamada tag incluyendo descendientes que sean igual al texto dado
[position]	Selecciona todos los elementos que se encuentran en la posición dada. La posición puede contener un entero siendo 1 la primera posición, la expresión last( ) para la última, o la posición relativa con respecto a la última posición last( )-1

Ejemplos:

1. Imprimir el primer hijo de un documento xml



2. Obtener los salarios y nombres de los trabajadores de la empresa obteniendo el nombre de esta:



```
#Se importa la libreria dom
import xml.dom.minidom as f
#Se le asigna a una variable los datos parceados del documento
doc = f.parse("datos.xml")
#Se obtiene el nombre del documento y se imprime
name = doc.getElementsByTagName("name")[0]
print(name.firstChild.data)
#Se obtienen todos los valores de los trabajadores, en este caso
#son los hijos de la empresa y el salario y nombre hijos del trabajador
#Por cada trabajador se imprime sus datos
staffs = doc.getElementsByTagName("staff")
for staff in staffs:
    sid = staff.getAttribute("id")
    nombre = staff.getElementsByTagName("nombre")[0]
    salario = staff.getElementsByTagName("salario")[0]
    print("id:%s, nombre:%s, salario:%s" %
          (sid, nombre.firstChild.data, salario.firstChild.data))
```

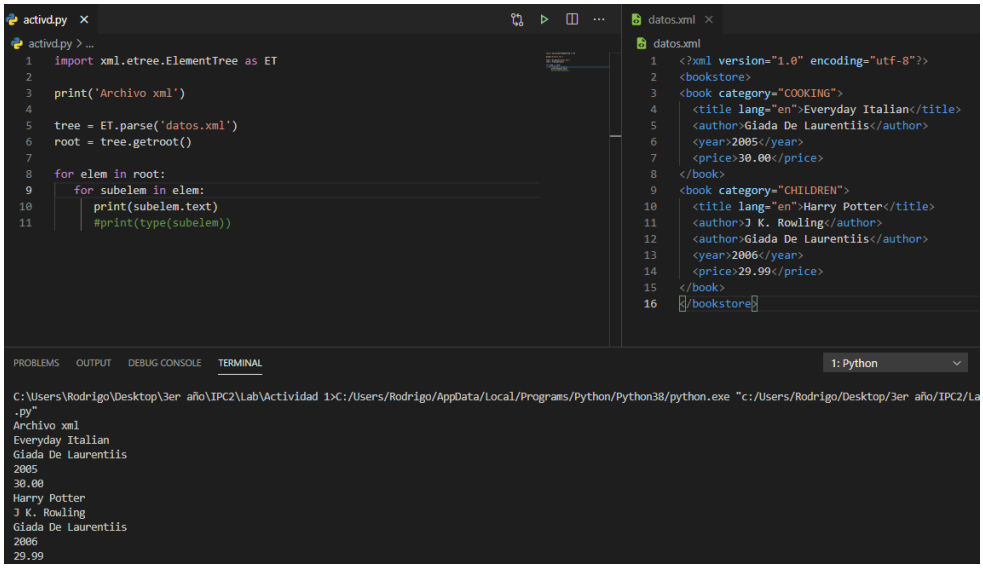
```
<?xml version="1.0"?>
<empresa>
    <name>Tesla</name>
    <staff id="12">
        <nombre>Marito</nombre>
        <salario>3000</salario>
    </staff>
    <staff id="13">
        <nombre>Pablito</nombre>
        <salario>2000</salario>
    </staff>
</empresa>
```

```
Microsoft Windows [Versión 10.0.18363.1316]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Rodrigo\Desktop\3er año\IPC2\Lab\Actividad 1>C:/Users/Rodrigo/AppData/Local/Programs/Python/Python38/python.exe "c:/Users/Rodrigo/Desktop/3er
.py"
Tesla
id:12, nombre:Marito, salario:3000
id:13, nombre:Pablito, salario:2000

C:\Users\Rodrigo\Desktop\3er año\IPC2\Lab\Actividad 1>
```

3. Desplegar todos los datos de un archivo xml:



```
import xml.etree.ElementTree as ET

print('Archivo xml')

tree = ET.parse('datos.xml')
root = tree.getroot()

for elem in root:
    for subelem in elem:
        print(subelem.text)
        #print(type(subelem))
```

```
<?xml version="1.0" encoding="utf-8"?>
<bookstore>
<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <author>Giada De Laurentiis</author>
  <year>2006</year>
  <price>29.99</price>
</book>
</bookstore>
```

```
C:\Users\Rodrigo\Desktop\3er año\IPC2\Lab\Actividad 1>C:/Users/Rodrigo/AppData/Local/Programs/Python/Python38/python.exe "c:/Users/Rodrigo/Desktop/3er año/IPC2/La
.py"
Archivo xml
Everyday Italian
Giada De Laurentiis
2005
30.00
Harry Potter
J K. Rowling
Giada De Laurentiis
2006
29.99
```

4. Eliminar el nodo ropa en el archivo xml:

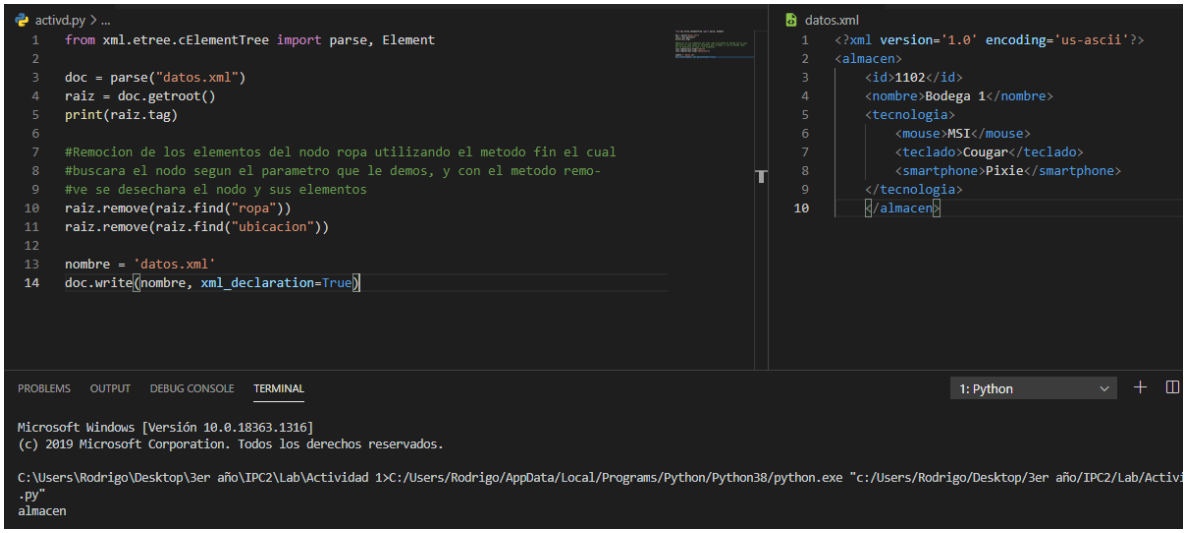
```
from xml.etree.cElementTree import parse, Element

doc = parse("datos.xml")
raiz = doc.getroot()
print(raiz.tag)

#Remocion de los elementos del nodo ropa utilizando el metodo fin el cual
#buscara el nodo segun el parametro que le demos, y con el metodo remo-
#ve se desechara el nodo y sus elementos
raiz.remove(raiz.find("ropa"))
raiz.remove(raiz.find("ubicacion"))

nombre = 'datos.xml'
doc.write(nombre, xml_declaration=True)
```

```
<?xml version="1.0" encoding="utf-8"?>
<almacen>
    <id>1102</id>
    <nombre>Bodega 1</nombre>
    <tecnologia>
        <mouse>MSI</mouse>
        <teclado>Cougar</teclado>
        <smartphone>Pixie</smartphone>
    </tecnologia>
    <ropa>
        <deporte>Camisa</deporte>
        <jean>Vaqueros</jean>
    </ropa>
    <ubicacion>Zona 11</ubicacion>
</almacen>
```



5. Agregar un nuevo nodo para correo al archivo XML y guardar los cambios en un archivo nuevo