Parsing XML Document

We can parse XML data using below two method-

1. parse(file\_path) --- this is used for parsing filedata
2. fromstring(str) ----- this is used for parsing from strings.

NOTE:

fromstring() parses XML from a string directly into an Element, which is the root element of the parsed tree.

Other parsing functions may create an ElementTree. Check the documentation to be sure.

**class xml.etree.ElementTree.ElementTree(element=None, file=None)**

**getroot()**

Returns the root element for this tree.

Once we call this method on any ElementTree it converts that into tree and we can get all child node one by one.

import xml.etree.ElementTree as ET

tree=ET.parse('edureak.xml')

print(type(tree))#<class 'xml.etree.ElementTree.ElementTree'>

#print(myroot)

root=tree.getroot() #Get the root of tree

#now iterate the all child nodes of tree

for child in root:

    print(type(child)) #<class 'xml.etree.ElementTree.ElementTree'>

**find(match, namespaces=None)**

Same as Element.find(), starting at the root of the tree

**findall(match, namespaces=None)**

Same as Element.findall(), starting at the root of the tree

**findtext(match, default=None, namespaces=None)**

Same as Element.findtext(), starting at the root of the tree

**class xml.etree.ElementTree.Element(tag, attrib={}, \*\*extra)**

fromstring() parses XML from a string directly into an Element, which is the root element of the parsed tree. Other parsing functions may create an ElementTree. Check the documentation to be sure.

Attributes and methods-

tag:

A string identifying what kind of data this element represents (the element type, in other words)

Example:

Get the tag and attribute of child node of root.

|  |  |
| --- | --- |
| tree=ET.parse('edureak.xml')  print(type(tree))#<class 'xml.etree.ElementTree.ElementTree'>  #print(myroot)  root=tree.getroot()  for child in root:      print(child.tag,child.attrib) | <class 'xml.etree.ElementTree.ElementTree'>  food {'food\_name': 'breakfast'}  food {}  food {'food\_name': 'upma'}  food {}  food {} |

text:

To get the data/text data of that element

tail:

These attributes can be used to hold additional data associated with the element. Their values are usually strings but may be any application-specific object.

**get(key, default=None)**

Gets the element attribute named key/tag.

keys()

Returns the elements attribute names as a list. The names are returned in an arbitrary order

**set(key, value)**

Set the attribute key on the element to value.

items()

Returns the element attributes as a sequence of (name, value) pairs. The attributes are returned in an arbitrary order.

**Iteration through all node/child node**

If nodes are nested then in that case to avoid nesting of for loop we can use **iter() function.**

Iter() gives the each node from a tree(not element Tree) in list.

Iter(element\_name) --- Gives all elements with specified name

tree=ET.parse('edureak.xml')

print(type(tree))#<class 'xml.etree.ElementTree.ElementTree'>

for i in tree.iter():#will give all nodes of tree in list form

    print('\*\*\*\*\*\*\*\*\*\*\*')

    print(i.tag,i.text)

Example: Finding elements with given name and getting it’s attribute

Find all element by tag name food and print it’s attribute

import xml.etree.ElementTree as ET

tree=ET.parse('edureak.xml')

print(type(tree))#<class 'xml.etree.ElementTree.ElementTree'>

#Find all element with food and anywhere

#after finding get the attribute

for i in tree.iter('food'):

    print(i.tag,i.attrib)

#XPATH in XML#

This is generally used to find element in xml.

It’s almost same as Selenium XPATH

|  |  |
| --- | --- |
| **XPath Expression** | **Result** |
| /bookstore/book[1] | Selects the first book element that is the child of the bookstore element |
| /bookstore/book[last()] | Selects the last book element that is the child of the bookstore element |
| /bookstore/book[last()-1] | Selects the last but one book element that is the child of the bookstore element |
| /bookstore/book[position()<3] | Selects the first two book elements that are children of the bookstore element |
| //title[@lang] | Selects all the title elements that have an attribute named lang |
| //title[@lang='en'] | Selects all the title elements that have a "lang" attribute with a value of "en" |
| /bookstore/book[price>35.00] | Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00 |
| /bookstore/book[price>35.00]/title | Selects all the title elements of the book elements of the bookstore element that have a price element with a value greater than 35.00 |

Example:

Find all element from xml file with ‘price’ tag.

print(type(tree))#<class 'xml.etree.ElementTree.ElementTree'>

#find all element with tag item

xpath='//item'

for each in tree.findall(xpath):

    print(each.tag,each.attrib,each.text)

**Changing the value of any element.**

We can change the element value using set(value\_to\_change,new\_value).

**Question** – From edureka file change the value of <item > element for which description is "Idly"

to “Idly replaced by Poori”

Solution:

tree=ET.parse('edureak.xml')

print(type(tree))#<class 'xml.etree.ElementTree.ElementTree'>

#find all element with tag item

xpath='//item'

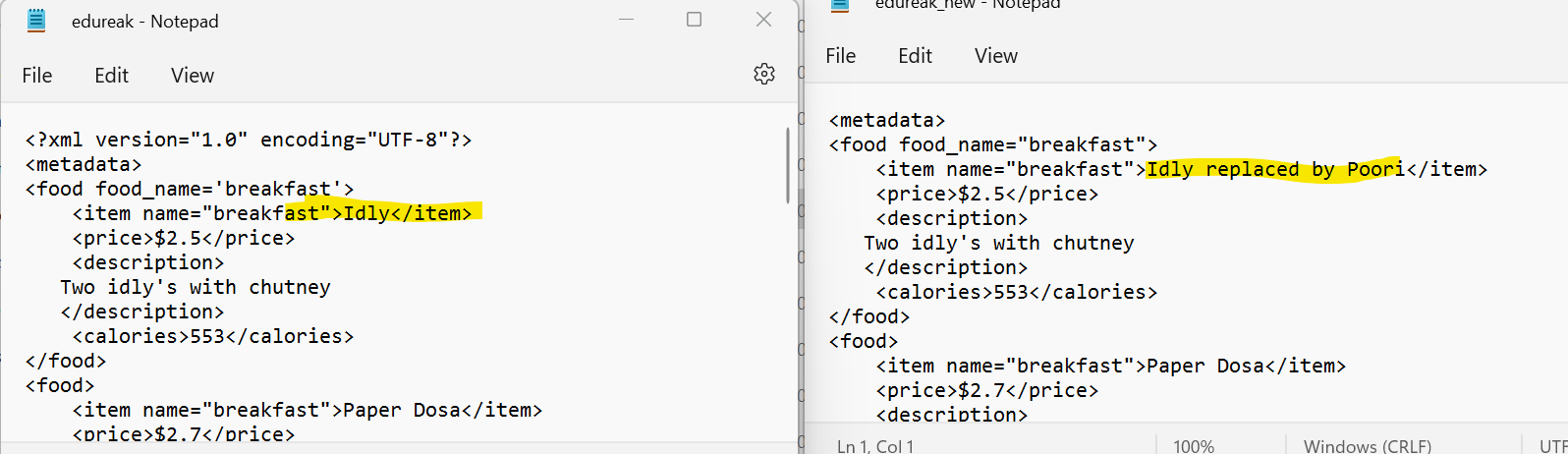
for each in tree.findall(xpath):

    print(each.tag,each.attrib,each.text)

    if each.text=="Idly":

        each.text='Idly replaced by Poori'

tree.write('edureak\_new.xml')



Creating/saving xml data into file

tree.write(file\_path)