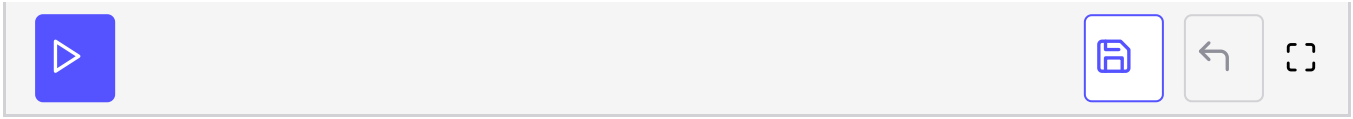
How to Parse XML with ElementTree

WE'LL COVER THE FOLLOWING

Now we get to learn how to do some basic parsing with ElementTree. First we'll read through the code and then we'll go through bit by bit so we can understand it. Note that this code is based around the original example, but it should work on the second one as well.

```
import xml.etree.cElementTree as ET

def parseXML(xml_file):
    """
    Parse XML with ElementTree
    """
    tree = ET.ElementTree(file=xml_file)
    print(tree.getroot())
    root = tree.getroot()
    print("tag=%s, attrib=%s" % (root.tag, root.attrib))

    for child in root:
        print(child.tag, child.attrib)
        if child.tag == "appointment":
            for step_child in child:
                print(step_child.tag)

    # iterate over the entire tree
    print("-" * 40)
    print("Iterating using a tree iterator")
    print("-" * 40)
    iter_ = tree.getiterator()
    for elem in iter_:
        print(elem.tag)

    # get the information via the children!
    print("-" * 40)
    print("Iterating using getchildren()")
    print("-" * 40)
    appointments = root.getchildren()
    for appointment in appointments:
        appt_children = appointment.getchildren()
        for appt_child in appt_children:
```

```
            print("%s=%s" % (appt_child.tag, appt_child.text))



if __name__ == "__main__":
    parseXML("appt.xml")
```



You may have already noticed this, but in this example and the last one, we've been importing cElementTree instead of the normal ElementTree. The main difference between the two is that cElementTree is C-based instead of Python-based, so it's much faster. Anyway, once again we create an ElementTree object and extract the root from it. You'll note that we print out the root and the root's tag and attributes. Next we show several ways of iterating over the tags. The first loop just iterates over the XML child by child. This would only print out the top level child (appointment) though, so we added an if statement to check for that child and iterate over its children too.

Next we grab an iterator from the tree object itself and iterate over it that way. You get the same information, but without the extra steps in the first example. The third method uses the root's **getchildren()** function. Here again we need an inner loop to grab all the children inside each appointment tag. The last example uses the root's **iter()** method to just loop over any tags that match the string "begin".

As mentioned in the last section, you could also use **find()** or **findall()** to help you find specific tags or sets of tags respectively. Also note that each Element object has a **tag** and a **text** property that you can use to acquire that exact information.

Wrapping Up

Now you know how to use minidom to parse XML. You have also learned how to use ElementTree to create, edit and parse XML. There are other libraries outside of Python that provide additional methods for working with XML too. Be sure you do some research to make sure you're using a tool that you understand as this topic can get pretty confusing if the tool you're using is obtuse.