

# XML redux

Week 5, day 1

# Extensible markup Language

- Structured markup of nested tags
- Produces a tree of tags with attributes
- No fixed tags
- Universal, can be read and written from just about anywhere

# Example

```
<? xml version="1.0" encoding="utf8"?>
<SHOPPING_LIST>
  <TO_GET>
    <ITEM >Eggs</ITEM>
    <ITEM type='low fat'>Milk</ITEM>
    <ITEM type='fresh'>Bread</ITEM>
  </TO_GET>
  <DONT_NEED>
    <ITEM>Tacos</ITEM>
    <ITEM>Onions</ITEM>
  </DONT_NEED>
</SHOPPING_LIST>
```

# Rules

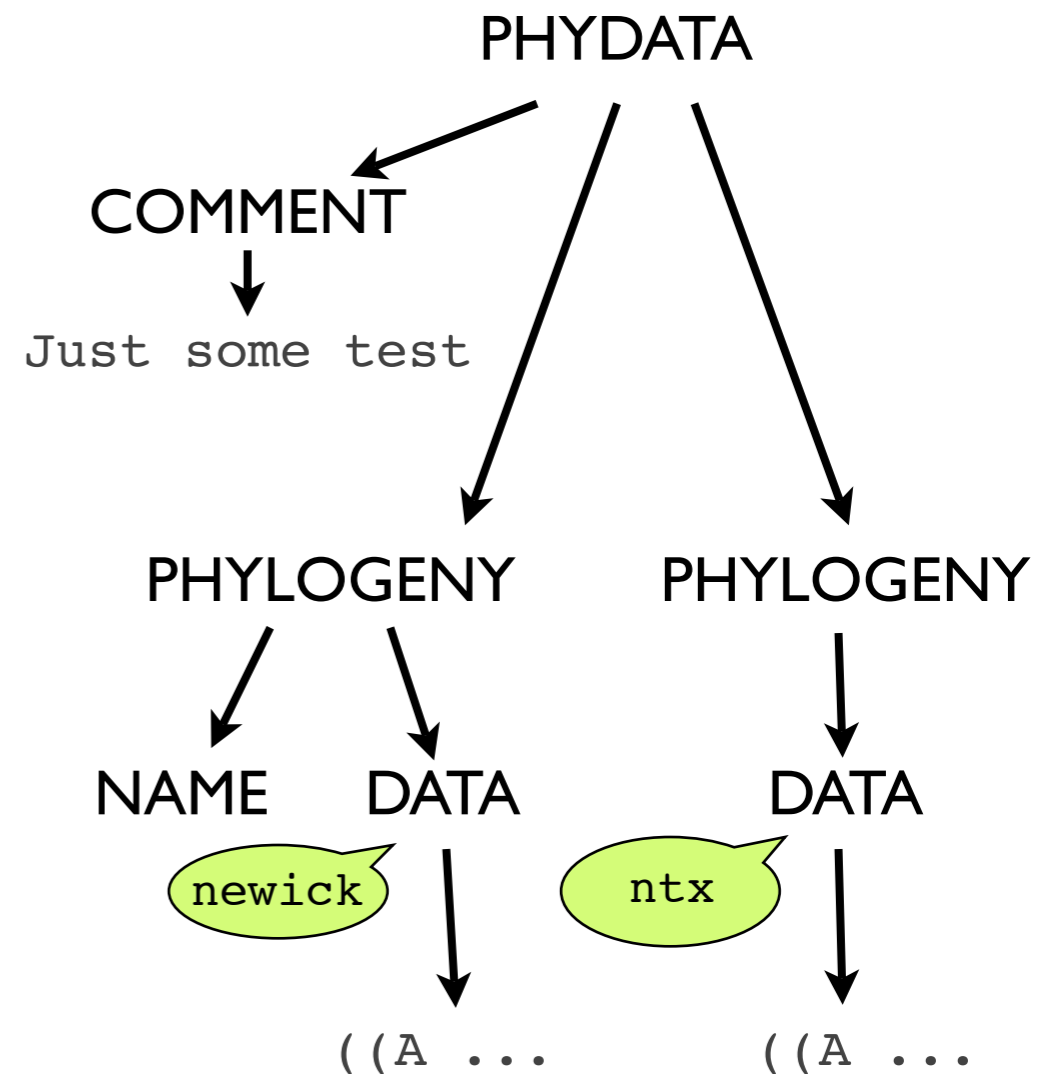
- Must be one root tag, non-repeated
- Tags nest & match
  - `<TAG />` equals `<TAG></TAG>`
- Tags can occur multiple times
- Tags may have attributes: key-value pairs
- There are no standard tags
- May be a directive at root

# The result

- A **well-formed** XML document is an ordered labelled tree
- A tree of **nodes** that are:
  - **character data** (text) ...
  - or **elements** with a **type** or **name** and labels or **attributes**
- We can traverse a tree in order

# The tree

```
<PHYDATA>
<COMMENT>Just some test
data</COMMENT>
<PHYLOGENY>
<NAME />
<DATA format="newick">
((A,B), (C,D))</DATA>
</PHYLOGENY>
<PHYLOGENY>
<DATA format="ntx">
((A,B,C), D)</DATA>
</PHYLOGENY>
</PHYDATA>
```



# Creating an XML document

```
import xml.dom.minidom

doc = Document()

root = doc.createElement ("PHYDATA")
doc.appendChild (root)

newElem = doc.createElementNS ("", "PHYDATA")
root.appendChild (newElem)
newElem.setAttribute("name", "Hyrax")

c = doc.createComment ("An example file")
t = doc.createTextNode ("This is text data")
root.appendChild (t)
root.appendChild (c)
```

# The calls

- `import xml.dom.minidom`
- `Document()`
- `doc.createElement (namespace, tag)`
- `elem.appendChild (node)`
- `doc.createComment (str)`
- `doc.createTextNode (str)`
- `elem.setAttribute (key, value)`

# Writing it out

```
# print to screen in nice format
import xml.dom.ext
xml.dom.ext.PrettyPrint (doc)

# to file
outfile = open ("out.xml")
xml.dom.ext.PrettyPrint (doc, outfile)
outfile.close()

# unpretty print
outfile = open ("out2.xml")
xml.dom.ext.Print (doc, outfile)
outfile.close()
```

# Reading / parsing

- DOM
  - Document Object Model
  - Read whole document and then process
- SAX
  - Simple API for XML
  - Element by element as it is read

# Reading XML

```
import xml.dom.minidom

tree = xml.dom.minidom.parse (file)

# or ...

tree = xml.dom.minidom.parse (filename)
```

# Parsing tree

```
import xml.dom.minidom
tree = xml.dom.minidom.parse ("eg.xml")

theRoot = tree.childNodes[0]

for child in theRoot.childNodes:
    if child.tagName == "DATA":
        # do something
```

# Interrogating nodes

- *element.childNodes* : return list of children
- *node.tagName*, *node.nodeName*: return tag
- *element.getAttribute*
- *element.firstChild*
- *element.lastChild*
- *node.parentNode*
- *node.nodeValue*



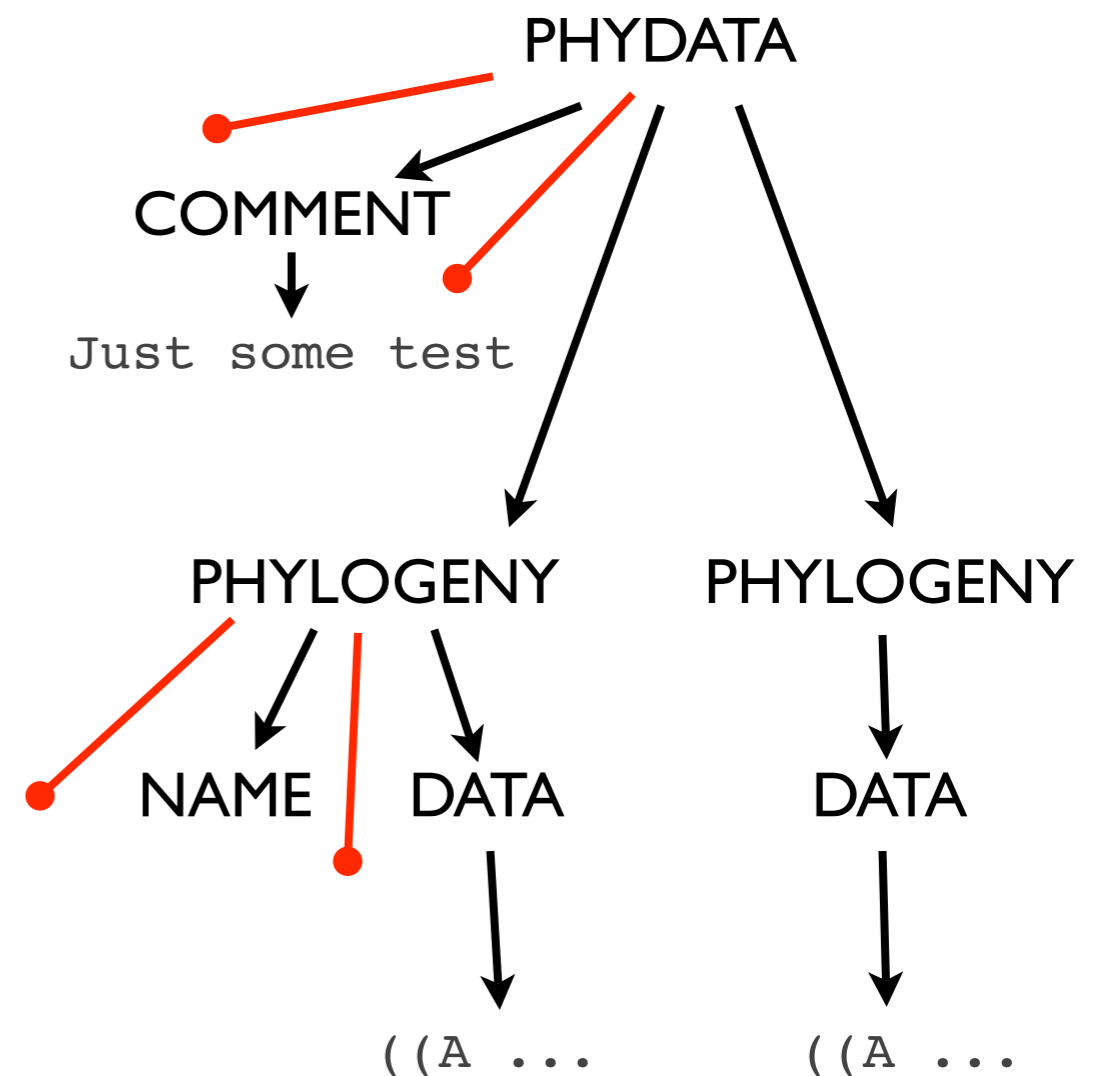
# Whitespace is significant

```
<PHYDATA>
<PHYLOGENY>
<NAME />
<DATA format="newick">
((A,B), (C,D))</DATA>
</PHYLOGENY>
<PHYLOGENY>
<DATA format="ntx">
((A,B,C), D)</DATA>
</PHYLOGENY>
</PHYDATA>
```

```
<PHYDATA><PHYLOGENY><NAME />
<DATA format="newick">((A,B),
(C,D))</DATA></
PHYLOGENY><PHYLOGENY><DATA format="ntx">
((A,B,C), D)</DATA></
PHYLOGENY></PHYDATA>
```

# Silent whitespace in prettyprinting

```
<PHYDATA>
<COMMENT>Just some test
data</COMMENT>
<PHYLOGENY>
<NAME />
<DATA format="newick">
((A,B), (C,D))</DATA>
</PHYLOGENY>
<PHYLOGENY>
<DATA format="ntx">
((A,B,C), D)</DATA>
</PHYLOGENY>
</PHYDATA>
```



# Dealing with whitespace

- Don't prettyprint
- Pass over in parse
  - `getElementByTagName`
- Transform before parse
  - `removeChild`

# Dealing with whitespace 2

- DTDs
- XSL
- Sax parser

# Elementtree

```
from elementtree import Elementtree
tree = Elementtree.parse ("input.xml")

root = tree.getRoot()

for node in root:
    print root

for node in root.findall ('PHYLOGENY'):
    print root

# node.text, node.attrib['key']
```

# SAX, briefly

- Event based parser
- Define a SAX-based class
- Subclass methods to handle tags

# Resources

- Recipe at Python Cookbook <<http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/303061>>
- Elementtree at <<http://effbot.org/zone/element-index.htm>>
- Practical XML for Python <<http://www.a-cooke.org/andrew/writing/python-xml.html>>