# xml.dom.minidom

## Reading (and writing) XML from Python

Prof. Andrew C.R. Martin
andrew.martin@ucl.ac.uk
http://www.bioinf.org.uk/

# Aims and objectives

- A brief introduction to **XML (and XHTML) documents**
- Know problems in **reading and writing XML**
- Understand the **requirements of XML parsers** and the **two main types**
- Know how to **write code using the DOM parser**
- **PRACTICAL**: write a script to read XML

# What is XML?

- **XML doesn’t DO anything**
  - XML is used to structure, store and transport data.
- A simple **markup** language to allow structured representation of data

# What is XML?

- Uses '**tags**' (or 'elements') to wrap pieces of data.

```
<note>
   <from>Andrew Martin</from>
   <to>Adrian Shepherd</to>
   <date>1st April 2020</date>
   <content>
      The students are very good this year!
   </content>
</note>
```

- Looks a bit like **HTML**, but you invent your own tags to structure the data.

# What is XML?

- Tags may also contain **attributes**

```
<note date='1st April 2020'>
    <from>Andrew Martin</from>
    <to>Adrian Shepherd</to>
    <content>
       The students are very good this year!
    </content>
</note>
```

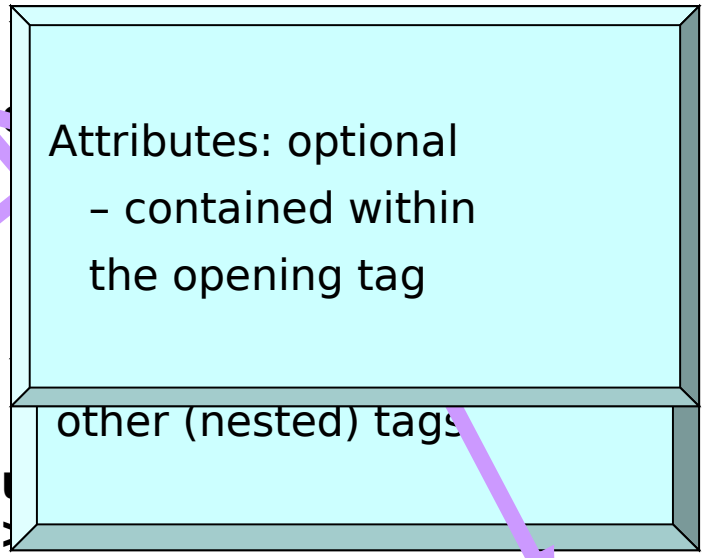
# What is XML?

- Unbalanced tags are special

```
<note date=’1st April 2020’>
   <from>Andrew Martin</from>
   <to>Adrian Shepherd</to>
   <content>
      The students are very good this year!
      <separator />
      April fool!
   </content>
</note>
```

# A more complex example

```
<mutants>
   <mutant_group native='1abc01'>
      <structure>
         <method>x-ray</method>
         <resolution>1.8</resolution>
         <rfactor>0.20</rfactor>
      </structure>

      <mutant domid='2bcd01'>
         <structure>
            <method>x-ray</method>
            <resolution>1.8</resolu
            <rfactor>0.20</rfactor>
         </structure>
         <mutation res='L24' native='ALA' subs='ARG' />
      </mutant>
   </mutant_group>
</mutants>
```



# XML technologies

- DTDs / XML-Schema
- Stylesheets (XSL/CSS)
- XSLT
- Native XML databases (e.g. eXist or BaseX)

# Writing XML

## Writing XML is straightforward

- Generate XML from a Python script using **`print()`** statements.
- However:
  - tags correctly **nested**
  - quote marks correctly **paired**
  - international **character sets**
  - Need to handle **special characters**

# Reading XML

## As simple or complex as you wish!

- Full control over XML:
  - simple pattern matching (regular expressions) may suffice
- Otherwise, may be dangerous
  - may rely on un-guaranteed formatting

```
<mutants><mutant_group native='1abc01'><structure><method>
x-ray</method><resolution>1.8</resolution><rfactor>0.20
</rfactor></structure><mutant domid='2bcd01'><structure>
<method>x-ray</method><resolution>1.8</resolution>
<rfactor>0.20</rfactor></structure><mutation res='L24’
native='ALA’ subs='ARG'/></mutant></mutant_group></mutants>
```

UCL

# XML Parsers

- Clear **rules** for data boundaries and hierarchy
- Predictable; unambiguous
- **Parser translates XML** into
  - stream of events
  - complex data object

# XML Parsers

## Good parser will handle:

- Different **data sources** of data
  - files
  - character strings
  - remote references
- different character **encodings**
  - standard Latin
  - Japanese
- checking for well-formedness **errors**

# XML Parsers

- **Read stream** of characters
  - differentiate markup and data
- Optionally replace **entity references**
  - (e.g. < with <)
- **Assemble** complete document
  - disparate (perhaps remote) sources
- Report syntax and validation **errors**
- Pass **data to client** program

# XML Parsers

- If XML has no syntax errors it is **'well formed'**

- With a DTD, a validating parser will check it matches: **'valid'**

# XML Parsers

- Writing a good parser is a **lot of work**!
- A lot of **testing** needed
- Fortunately, many parsers available

# Getting data to your program

- Parser can generate **'events'**
  - Tags are converted into events
  - Events triggered in your program as the document is read
- Parser acts as a **pipeline** converting XML into processed chunks of data sent to your program:
  - an **'event stream'**

# Getting data to your program

**OR…**

- XML converted into a **tree structure**

- Reflects **organization** of the XML
- Whole document **read into memory** before your program gets access

# Pros and cons

## Data structure

- ✓ More convenient
- ✓ Can access data in any order
- ✓ Code usually simpler
- ✘ May be impossible to handle very large files
- ✘ Need more processor time
- ✘ Need more memory

## Event stream

- ✓ Faster to access limited data
- ✓ Use less memory
- ✘ Parser loses data at the next event
- ✘ More complex code

- In the parser, everything is likely to be event-driven
- tree-based parsers create a data structure from the event stream

# SAX and DOM

## de facto standard APIs for XML parsing

- **SAX (Simple API for XML)**
  - event-stream API
  - originally for Java, but now for several programming languages (including Python)
  - development promoted by Peter Murray Rust, 1997-8
- **DOM (Document Object Model)**
  - W3C standard tree-based parser
  - platform- and language-neutral
  - allows update of document content and structure as well as reading

# XML parsers

## Many parsers available

- Differ in three major ways:
  - parsing style (event driven or data structure)
  - 'standards-completeness'
  - speed (implementation in C or pure Python)

# XML parsers

## expat

- C XML parser
  - Written by James Clark
  - Probably the first C parser

# Python XML parsers

**http://wiki.python.org/moin/PythonXml**

- A discussion of various Python XML parsers

# Python XML parsers

## xml.dom.minidom

- Probably the most popular
- Quite complete
- May be slow and memory hungry (but this is typical of DOM parsers)

**See**

http://docs.python.org/3/library/xml.dom.minidom.html

http://wiki.python.org/moin/MiniDom

# Python XML parsers

## xml.sax

- Implements the event-driven SAX standard

**See**

http://pyxml.sourceforge.net/topics/howto/section-SAX.html

https://wiki.python.org/moin/Sax

# Python XML parsers

## xml.dom.pulldom

- A compromise between DOM and SAX
- Passes through the file in an event-driven manner
- When an event of interest is found, provides a DOM version of the document tree beyond that point

**See**

https://wiki.python.org/moin/PullDom

# Python XML parsers

## ElementTree

- A more Python-esque approach to an event-driven parser (not SAX)
- Available as pure Python and a (faster) C-based implementation

**See**

https://wiki.python.org/moin/ElementTree

# Python XML parsers

## lxml

- Another more Python-esque approach to an event-driven parser (not SAX)
- Built on the libxml2 C library
- “...combines the speed and XML feature completeness of these libraries with the simplicity of a native Python API, mostly compatible but superior to the well-known ElementTree API..”

**See**

http://lxml.de/

# XML-DOM

- DOM is a **standard API**
  - once learned moving to a different language is straightforward
  - moving between implementations also easy
- Suppose we want to extract some data from an XML file...

# XML-DOM

```
<data>
   <species name='Felix domesticus'>
      <common-name>cat</common-name>
      <conservation status='not endangered' />
   </species>
   <species name='Drosophila melanogaster'>
      <common-name>fruit fly</common-name>
      <conservation status='not endangered' />
   </species>
</data>
```

**We want:**

```
cat (Felix domesticus) not endangered
fruit fly (Drosophila melanogaster) not endangered
```

```
#!/usr/bin/env python3

from xml.dom import minidom

doc = minidom.parse('test.xml')

for species in doc.getElementsByTagName('species'):

    speciesName = species.getAttribute('name')



    commonName = species.getElementsByTagName('common-name')[0].firstChild.data



    conservation = species.getElementsByTagName('conservation')[0].getAttribute('status')

    print ("%s (%s) %s" % (commonName, speciesName, conservation))
```

```
#!/usr/bin/env python3

# Import the minidom module
from xml.dom import minidom

# Parse the document
doc = minidom.parse('test.xml')

# getElementsByTagName() returns a list of
# elements with the given name. Here <species>
for species in doc.getElementsByTagName('species'):


    # getAttribute() returns a value associated with a given key
    # within a tag. Here <species name='xxxxx'>
    speciesName = species.getAttribute('name')

    # 1. Again getElementsByTagName() returns a list.
    # 2. We use just the first element of the list ([0]).
    # 3. From that we obtain the first child (could be a data node
    #    or another element)
    # 4. From the data node we obtain the actual data
    # Here: <common-name>xxxx</common-name>
    commonName = species.getElementsByTagName('common-name')[0].firstChild.data
    #                          >>1<<                          >>2<<  >>3<<   >>4<<

    # 1. Again getElementsByTagName() returns a list.
    # 2. We use just the first element of the list ([0]).
    # 3. From the element we use getAttribute to obtain the status
    # Here: <conservation status='not endangered' />
    conservation = species.getElementsByTagName('conservation')[0].getAttribute('status')
    #                             >>1<<                          >>2<<  >>3<<   >>4<<

    # Finally print the results
    print ("%s (%s) %s" % (commonName, speciesName, conservation))
```

## `.getElementsByTagName` returns a list

Here we work through the list:

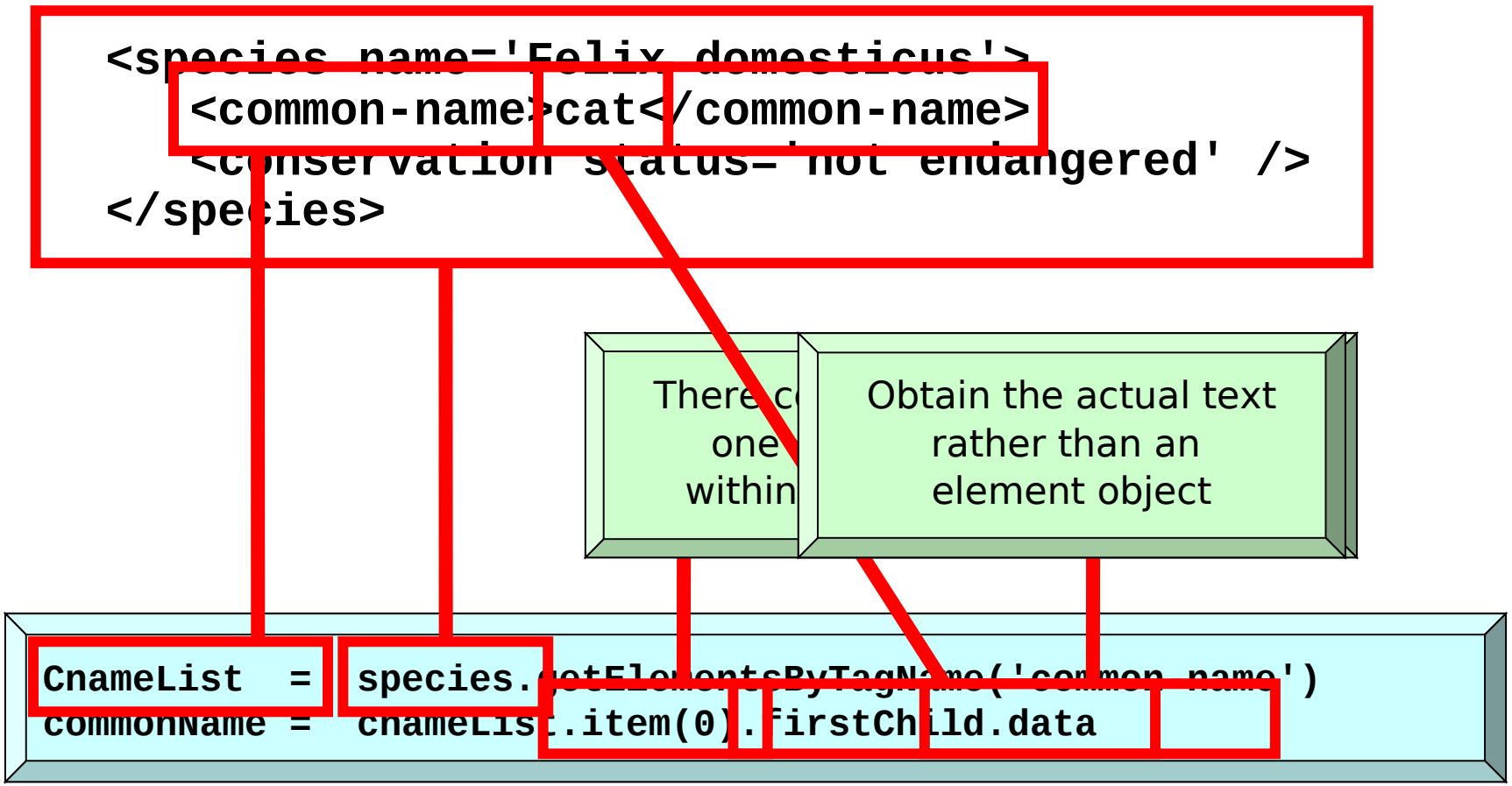
```
for species in doc.getElementsByTagName('species'):
    ... Do something ...
```

**species** is now a document object so we can use it in the same way as **doc** to obtain a list of elements:

```
commonName = species.getElementsByTagName('common-name')[0].firstChild.data
```

Splitting this up:

```
cnameList  = species.getElementsByTagName('common-name')
commonName = cnameList[0].firstChild.data
```


<species name='Felix domesticus'>
<common-name>cat</common-name>
<conservation status='not endangered' />
</species>
There c
one
within
Obtain the actual text
rather than an
element object
CnameList =
species.
commonName =
.item(0).firstChild.data

# Attributes

Attributes are much simpler!
Can only contain text (no nested elements)
Simply specify the attribute you wish to retrieve

```
#!/usr/bin/env python3

from xml.dom import minidom

doc = minidom.parse('test.xml')

for species in doc.getElementsByTagName('species'):
    speciesName = species.getAttribute('name')
    commonName = species.getElementsByTagName('common-name')[0].firstChild.data
    conservation = species.getElementsByTagName('conservation')[0].getAttribute('status')
    print ("%s (%s) %s" % (commonName, speciesName, conservation))
```

```
<species name='Felix domesticus'>
```

# Attributes

```
#!/usr/bin/env python3

from xml.dom import minidom

doc = minidom.parse('test.xml')

for species in doc.getElementsByTagName('species'):
    speciesName = species.getAttribute('name')
    commonName = species.getElementsByTagName('common-name')[0].firstChild.data
    conservation = species.getElementsByTagName('conservation')[0].getAttribute('status')
    print "%s (%s) %s" % (commonName, speciesName, conservation)
```

## Other ways to do it:

```
   conservationElement  = species.getElementsByTagName('conservation')[0]
   conservation         = conservationElement.getAttribute('status')
---
   conservationElement  = species.getElementsByTagName('conservation').item(0)
   conservation         = conservationElement.getAttribute('status')
---
   conservationElements = species.getElementsByTagName('conservation')
   conservation         = conservationElements[0].getAttribute('status')
---
   conservationElements = species.getElementsByTagName('conservation')
   conservation         = conservationElements.item(0).getAttribute('status')
```

UCL

# Alternatively...

```
#!/usr/bin/env python3

from xml.dom import minidom

doc = minidom.parse('test.xml')

for species in doc.getElementsByTagName('species'):
    speciesName = species.getAttribute('name')

    for commonNameElement in species.getElementsByTagName('common-name'):
        commonName = commonNameElement.firstChild.data

    for conservationElement in species.getElementsByTagName('conservation'):
        conservation = conservationElement.getAttribute('status')

    print ("%s (%s) %s" % (commonName, speciesName, conservation))
```

# XML-DOM

## Note

- Not necessary to use variable names that match the tags, **but it is a very good idea!**

- There are many many more functions, but this set covers most needs

# Summary - reading XML

**Load the module**

```
from xml.dom import minidom
```

**Parse a file**

```
doc = minidom.parse('filename')
```

**Extract all elements matching tag-name**

```
elementSet = doc.getElementsByTagName('tag-name')
```

**Extract first element of a set**

```
element = elementSet.item(0)
element = elementSet[0]
```

**Extract first child of an element**

```
childElement = element.firstChild
```

**Extract text from an element**

```
text = element.data
```

**Get the value of a tag's attribute**

```
text = element.getAttribute('attribute-name')
```

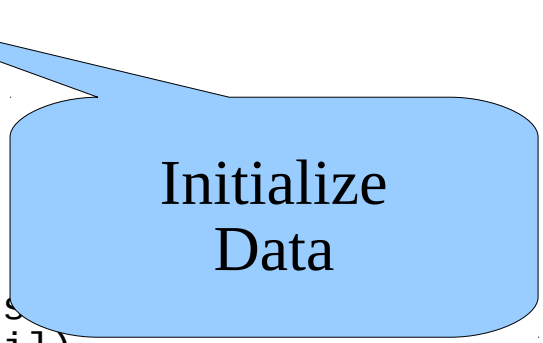
# Writing XML with XML-DOM

```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus', 'Drosophila melanogaster')
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'not endangered')

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('species')
    species.setAttribute('name', names[i])

    data.appendChild(species)

    cname  = doc.createElement('common-name')
    text = doc.createTextNode(commonNames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```

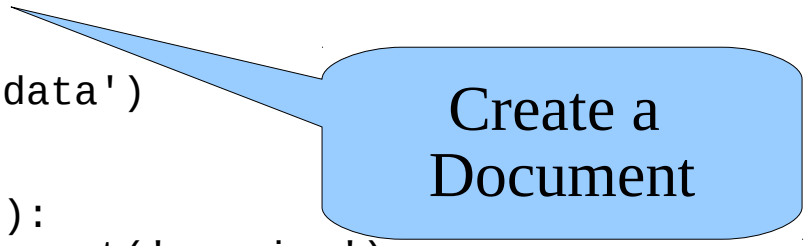
```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus',
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'not

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('species')
    species.setAttribute('name', names[i])

    data.appendChild(species)

    cname  = doc.createElement('common-name')
    text = doc.createTextNode(commonNames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```

Import modules

```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus', 'Drosophila melanogaster')
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'not endangered')

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('species')
    species.setAttribute('name', names[i])

    data.appendChild(species)

    cname  = doc.createElement('common-name')
    text = doc.createTextNode(commonNames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```



```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus', 'Drosophila melanogaster')
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'not endangered')

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('species')
    species.setAttribute('name', names[i])

    data.appendChild(species)

    cname  = doc.createElement('common-name')
    text = doc.createTextNode(commonNames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```



```
<?xml version=”1.0” ?>
```

```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus', 'Dr
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'not e

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('species')
    species.setAttribute('name', names[i])

    data.appendChild(species)

    cname  = doc.createElement('common-name')
    text = doc.createTextNode(commonNames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```

Create the data Element & attach to the document

```
<?xml version=”1.0” ?>
<data />
```

```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus',                    ster')
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'n

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('species')
    species.setAttribute('name', names[i])

    data.appendChild(species)

    cname  = doc.createElement('common-name')
    text = doc.createTextNode(commonNames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```

Step through the data

```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus', 'Dro
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'not e

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('species')
    species.setAttribute('name', names[i])

    data.appendChild(species)

    cname  = doc.createElement('common-name')
    text = doc.createTextNode(commonNames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```

Create <element>
Set name attribute
Attach to <data>

```
<?xml version=”1.0” ?>
<data>
   <species name=”Felix domesticus” />
</data>
```

```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesti
commonNames = ('cat', 'fruit
consStatus  = ('not endangere

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('s      es')
    species.setAttribute('name', n     s[i])

    data.appendChild(species)

    cname  = doc.createElement('common-name')
    text = doc.createTextNode(commonNames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```

Create <common-name>
Create a text node
Add the text to <common-name>
Add it to the <species>

```
<?xml version=”1.0” ?>
<data>
   <species name=”Felix domesticus”>
      <common-name>cat</common-name>
   </species>
</data>
```

```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus', 'Drosophila melanogaster')
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'not endangered')

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('spe
    species.setAttribute('name', names[i

    data.appendChild(species)

    cname  = doc.createElement('comm    name')
    text = doc.createTextNode(commo    ames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```

Create <conservation>
Set `status=` attribute
Add to species

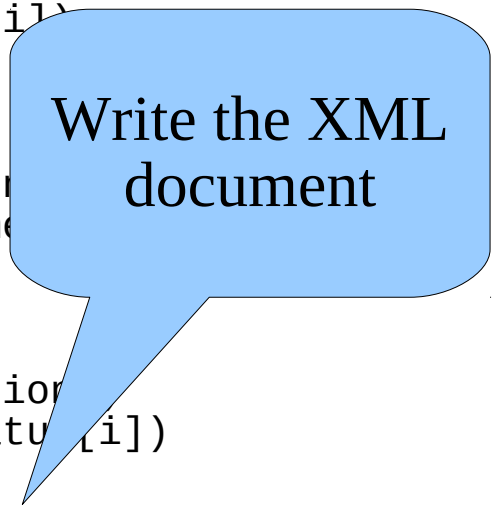
```
<?xml version=”1.0” ?>
<data>
   <species name=”Felix domesticus”>
      <common-name>cat</common-name>
      <conservation status=”not endangered” />
   </species>
</data>
```

```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus',                    ster')
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'n

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('species')
    species.setAttribute('name', names[i])

    data.appendChild(species)

    cname  = doc.createElement('common-name')
    text = doc.createTextNode(commonNames[i])
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservation')
    cons.setAttribute('status', consStatus[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```

Step through the data

```
<?xml version=”1.0” ?>
<data>
   <species name=”Felix domesticus”>
      <common-name>cat</common-name>
      <conservation status=”not endangered” />
   </species>
   <species name="Drosophila melanogaster">
      <common-name>fruit fly</common-name>
      <conservation status="not endangered"/>
   </species>
</data>
```

```
#!/usr/bin/env python3

from xml.dom import minidom
import sys

nspecies    = 2
names       = ('Felix domesticus', 'Drosophila melanogaster')
commonNames = ('cat', 'fruit fly')
consStatus  = ('not endangered', 'not endangered')

doc = minidom.Document()

data = doc.createElement('data')
doc.appendChild(data)

for i in range(0, nspecies):
    species = doc.createElement('species')
    species.setAttribute('name', names[i])

    data.appendChild(species)

    cname  = doc.createElement('common-n
    text = doc.createTextNode(commonName
    cname.appendChild(text)
    species.appendChild(cname)

    cons = doc.createElement('conservatio
    cons.setAttribute('status', consStatu[i])
    species.appendChild(cons)

doc.writexml(sys.stdout, addindent='   ', newl='\n')
```



# Summary - writing XML

**Create an XML document structure**

```
doc = minidom.Document()
```

**Create a tagged element**

```
element = doc.createElement('tag-name')
```

**Set an attribute for an element**

```
element.setAttribute('attrib-name', 'value')
```

**Append a child element to an element**

```
parent_element.appendChild(child_element)
```

**Create a text node element**

```
element = doc.createTextNode('text')
```

**Print a document structure as a string**

```
doc.writexml(file,
             addindent='   ', # Indent size
             newl='\n')       # Newline if needed
```

# Summary

- Two **types of parser**
  - Event-driven
  - Data structure
- **Writing** a good parser is difficult!
- **Many** parsers available
- **xml.dom.minidom** for reading and writing data

# References

## XML

http://www.xml.com/pub/a/98/10/guide0.html

http://www.w3schools.com/xml/xml_whatis.asp

http://en.wikipedia.org/wiki/XML

## Python and XML

https://www.mkyong.com/python/python-read-xml-file-dom-example/

https://docs.python.org/3/library/xml.dom.minidom.html

https://wiki.python.org/moin/MiniDom

https://wiki.python.org/moin/ElementTree

https://wiki.python.org/moin/Sax

https://wiki.python.org/moin/PythonXml

https://wiki.python.org/moin/PullDom

http://lxml.de/