# XML Schema

# XML Schema

- XML itself does not restrict what elements may occur in a document.
- In a given application, you want to fix a vocabulary -- what elements make sense, what their types are, etc.
- Use a **Schema** to define an XML dialect
  - MusicXML, ChemXML, VoiceXML, WSDL, SOAP, etc.
- Restrict documents to those tags.
- Schema can be used to validate a document -- i.e. to see if it obeys the rules of the dialect.
- W3C recommendation since 2001

# Schema determine ...

- What sort of elements can appear in the document.
- What elements MUST appear
- Which elements can appear as part of another element
- What attributes can appear or must appear
- What kind of values can/must be in an attribute.

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
   <book id="b0836217462" available="true">
      <isbn> 0836217462 </isbn>
      <title lang="en"> Being a Dog is a Full-Time Job </title>
      <author id="CMS">
         <name> Charles Schulz </name>
         <born> 1922-11-26 </born>
         <dead> 2000-02-12 </dead>
      </author>
      <character id="PP">
         <name> Peppermint Patty </name>
         <born> 1966-08-22 </born>
         <qualification> bold,brash, and tomboyish </qualification>
      </character>
      <character id="Snoopy">
         <name> Snoopy</name>
         <born>1950-10-04</born>
         <qualification>extroverted beagle</qualification>
      </character>
      <character id="Schroeder">
         <name>Schroeder</name>
         <born>1951-05-30</born>
         <qualification>brought classical music to the Peanuts Strip</qualification>
      </character>
      <character id="Lucy">
         <name>Lucy</name>
         <born>1952-03-03</born>
         <qualification>bossy, crabby, and selfish</qualification>
      </character>
   </book>
</library>
```

- We start with a sample XML document and reverse engineer a schema as a simple example

First identify the elements:
*author, book, born, character, dead, isbn, library, name, qualification, title*

Next categorize by *content model*
*Empty*: contains nothing
*Simple*: only text nodes
*Complex*: only sub-elements
*Mixed*: text nodes + sub-elements

Note: content model independent of comments and attributes.

# Content models

- Simple content model: *name, born, title, dead, isbn, qualification*
- Complex content model: libarary, character, book, author

# Content Types

- We further distinguish between complex and simple content **Types:**
  - Simple Type: An element with only text nodes and no child elements or attributes
  - Complex Type: All other cases
- We also say (and require) that all attributes themselves have simple type

# Content Types

- Simple content type: *name, born, dead, isbn, qualification*
- Complex content type: library, character, book, author, title

# Building the schema

- Schema are XML documents
- They must contain a schema root element as such

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">
... ...
</xs:schema>
```

- We will discuss details in a bit -- yellow part can be excluded for now.

# Flat schema for library

Start by defining all of the simple types (including attributes):

```
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
  <xs:element name=“name” type=“xs:string”/>
  <xs:element name=“qualification” type=“xs:string”/>
  <xs:element name=“born” type=“xs:date”/>
  <xs:element name=“dead” type=“xs:date”/>
  <xs:element name=“isbn” type=“xs:string”/>
  <xs:attribute name=“id” type=“xs:ID”/>
  <xs:attribute name=“available” type=“xs:boolean”/>
  <xs:attribute name=“lang” type=“xs:language/>
   .../...
</xs:schema>
```

# Complex types with simple content

Now to complex types:

```
<title lang="en">
  Being a Dog is  ...
</title>

<xs:element name="title">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute ref="lang"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

"the element named *title* has a complex type which is a simple content obtained by extending the predefined datatype *xs:string* by adding the attribute defined in this schema and having the name *lang*."

# Complex Types

All other types are complex types with complex content. For example:

```
<xs:element name="library">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="book" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="author">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="name"/>
      <xs:element ref="born"/>
      <xs:element ref="dead" minOccurs=0/>
    </xs:sequence>
    <xs:attribute ref="id"/>
  </xs:complexType>
</xs:element>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="name" type="xs:string"/>
   <xs:element name="qualification" type="xs:string"/>
   <xs:element name="born" type="xs:date"> </xs:element>
   <xs:element name="dead" type="xs:date"> </xs:element>
   <xs:element name="isbn" type="xs:string"> </xs:element>
   <xs:attribute name="id" type="xs:ID"> </xs:attribute>
   <xs:attribute name="available" type="xs:boolean"> </xs:attribute>
   <xs:attribute name="lang" type="xs:language"> </xs:attribute>
   <xs:element name="title">
      <xs:complexType>
         <xs:simpleContent>
            <xs:extension base="xs:string">
               <xs:attribute ref="lang"> </xs:attribute>
            </xs:extension>
         </xs:simpleContent>
      </xs:complexType>
   </xs:element>
   <xs:element name="library">
      <xs:complexType>
         <xs:sequence>
            <xs:element maxOccurs="unbounded" ref="book"> </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
   <xs:element name="author">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="name"> </xs:element>
            <xs:element ref="born"> </xs:element>
            <xs:element ref="dead" minOccurs="0"> </xs:element>
         </xs:sequence>
         <xs:attribute ref="id"> </xs:attribute>
      </xs:complexType>
   </xs:element>
```

```
   <xs:element name="book">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="isbn"> </xs:element>
            <xs:element ref="title"> </xs:element>
            <xs:element ref="author" minOccurs="0"
                        maxOccurs="unbounded”/>
            <xs:element ref="character" minOccurs="0"
                        maxOccurs="unbounded"/>
         </xs:sequence>
         <xs:attribute ref="available"> </xs:attribute>
         <xs:attribute ref="id"> </xs:attribute>
      </xs:complexType>
   </xs:element>
   <xs:element name="character">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref="name"/>
            <xs:element ref="born"/>
            <xs:element ref="qualification"/>
         </xs:sequence>
         <xs:attribute ref="id"> </xs:attribute>
      </xs:complexType>
   </xs:element>

</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="library">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="book" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="isbn" type="xs:integer"> </xs:element>
                     <xs:element name="title">
                        <xs:complexType>
                           <xs:simpleContent>
                              <xs:extension base="xs:string">
                                 <xs:attribute name="lang" type="xs:language"
                                 > </xs:attribute>
                              </xs:extension>
                           </xs:simpleContent>
                        </xs:complexType>
                     </xs:element>
                     <xs:element name="author" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="name" type="xs:string"> </xs:element>
                              <xs:element name="born" type="xs:date"> </xs:element>
                              <xs:element name="dead" type="xs:date"> </xs:element>
                           </xs:sequence>
                           <xs:attribute name="id" type="xs:ID"> </xs:attribute>
                        </xs:complexType>
                     </xs:element>
                     <xs:element name="character" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="name" type="xs:string"> </xs:element>
                              <xs:element name="born" type="xs:date"> </xs:element>
                              <xs:element name="qualification" type="xs:string"
                              > </xs:element>
                           </xs:sequence>
                           <xs:attribute name="id" type="xs:ID"> </xs:attribute>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
                  <xs:attribute type="xs:ID" name="id"> </xs:attribute>
                  <xs:attribute name="available" type="xs:boolean"> </xs:attribute>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Same schema but with everything defined locally!

Here the root is forced to be library, before any element could be root

# Dissecting Schema

# What’s in a Schema?

- A Schema is an XML document
- Because it is an XML document, it must have a root element
  - The root element is `<schema>`
- Within the root element, there can be different kinds of information
  - To define the prescribed structure of corresponding XML documents
  - To combine different schemas
  - To add documentation information
- We are interested in
  - Simple and complex data type definitions
  - Element and attribute definitions

# Structure of a Schema

```
<schema>
   <!- any number of the following -->

   <include .../>
   <import> ... </import>
   <redefine> ... </redefine>
   <annotation> ... </annotation>

   <!- any number of following definitions -->

   <simpleType> ... </simpleType>
   <complexType> ... </complexType>
   <element> ... </element>
   <attribute/>
   <attributeGroup> ... </attributeGroup>
   <group> ... </group>
   <annotation> ... </annotation>
</schema>
```

# Simple Types

# Elements

- What is an element with simple type?
  - A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.
- Can also add restrictions to a data type in order to limit its content, and you can require the data to match a defined pattern.

# Example Simple Element

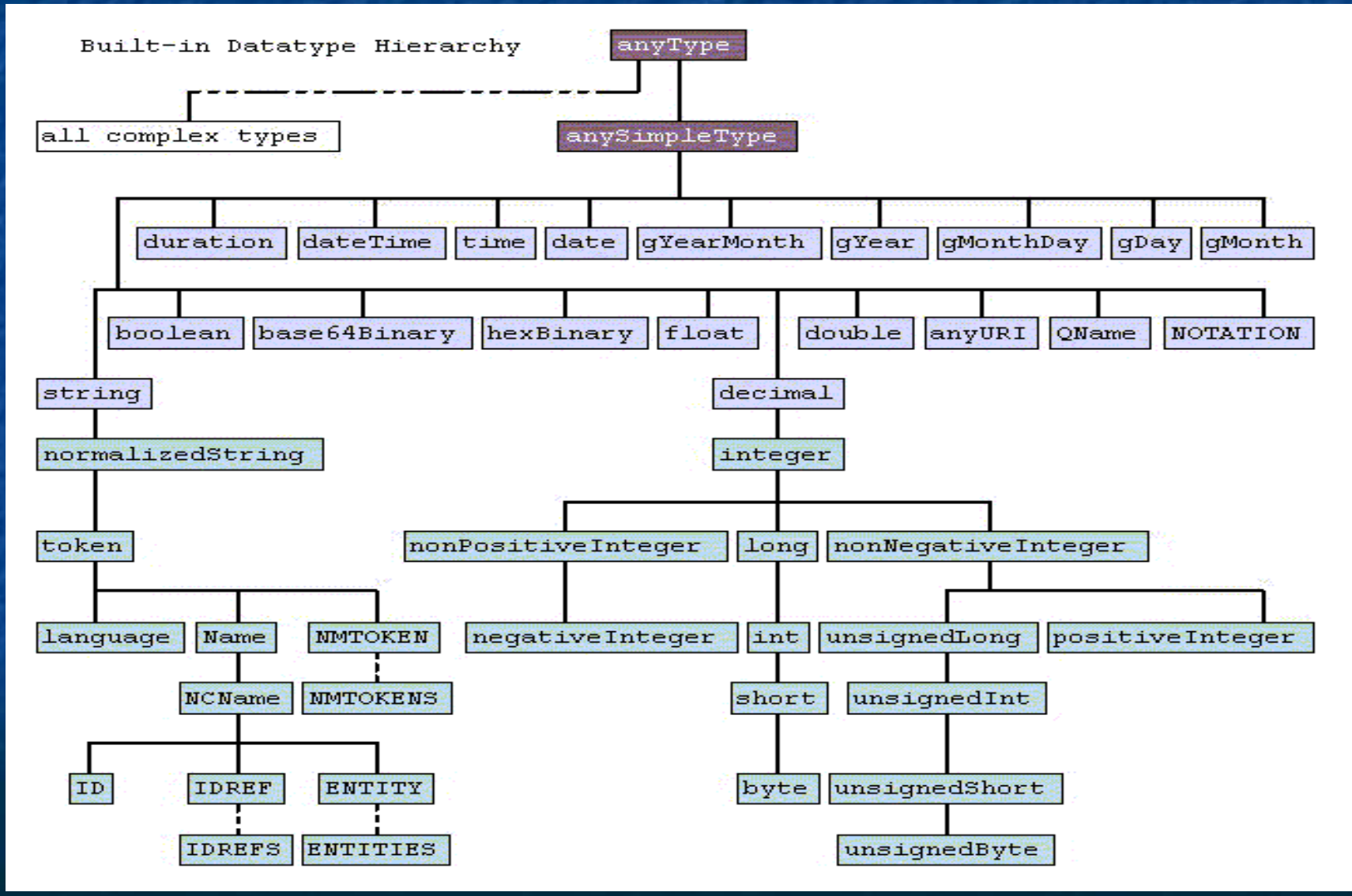
- The syntax for defining a simple element is:
  - **<xs:element name="xxx" type="yyy"/>**
    where xxx is the name of the element and yyy is the data type of the element. Here are some XML elements:
    - <lastname>Refsnes</lastname>
    - <age>37</age>
    - <dateborn>1968-03-27</dateborn>

  And here are the corresponding simple element definitions:
    - <xs:element name="lastname" type="xs:string"/>
    - <xs:element name="age" type="xs:integer"/>
    - <xs:element name="dateborn" type="xs:date"/>

# Common XML Schema Data Types

- XML Schema has a lot of built-in data types. Here is a list of the most common types:
  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

# XML Schema Data Types



# Attributes (Another simple type)

- All attributes are declared as simple types.
- Only elements with complex type can have attributes!

# How to Define an Attribute

- The syntax for defining an attribute is:
  - <xs:attribute name="xxx" type="yyy"/> where xxx is the name of the attribute and yyy is the data type of the attribute. Here is an XML element with an attribute:
  - <lastname lang="EN">Smith</lastname>

    And here is a corresponding simple attribute definition:
  - <xs:attribute name="lang" type="xs:string"/>

# Creating Optional and Required Attributes

- All attributes are optional by default. To explicitly specify that the attribute is optional, use the "use" attribute:
  - <xs:attribute name="lang" type="xs:string" use="optional"/>

  To make an attribute required:

  - <xs:attribute name="lang" type="xs:string" use="required"/>

# Complex Types

# Elements with Complex Type

- An element with complex type is an XML element that contains other elements and/or attributes.

- There are four kinds of elements with complex type:
  - empty elements
  - elements that contain only other elements
  - elements that contain only text
  - elements that contain both other elements and text

- **Note:** Each of these elements may (or must) contain attributes as well!

# Examples of XML Elements with Complex Type

- An element, "product", which is empty:
  - <product pid="1345"/>
- An element, "employee", which contains only other elements:
  - <employee>
    <firstname>John</firstname>
    <lastname>Smith</lastname>
    </employee>
- An element, "food", which contains only text:
  - <food type="dessert">Ice cream</food>
- An element, "description", which contains both elements and text:
  - <description> It happened on <date

    lang="norwegian">03.03.99</date>..</description>

# Empty Elements

- An element, "product", which is empty and has no attribute:
  - <product/>
  - Defined as:

```
<xs:element name="product">
  <xs:complexType/>
</xs:element>
```

- An element, "product", which is empty:
  - <product pid="1345"/>
  - Defined as:

```
<xs:element name="product">
    <xs:complexType>
        <xs:attribute name="pid" type="xs:positiveInteger"/>
    </xs:complexType>
</xs:element>
```

# Simple Content in Complex Type

- If a type contains only simple content (text and attributes), a <simpleContent> element can be put inside the <complexType>

- <simpleContent> must have an <extension>

- <xs:element name="dialog">
  <xs:complexType>
  <xs:simpleContent>
  <xs:extension base="xs:string">
  <xs:attribute name="speaker"
  type="xs:string" use="required"/>
  </xs:extension>
  </xs:simpleContent>
  </xs:complexType>
  </xs:element>

# Elements Containing Other Elements

- We need to specify how other elements can occur
  - There are 3 combinators
    - All: all the elements specified must occur, in any order
    - Choice: one of the elements specified must occur
    - Sequence: all the elements specified must appear, in the specified order
  - Each element is recursively described
  - The attribute mixed="true" allows for having also text

# Example

- The following schema specifies 3 elements and mixed content

```
<xs:element name="BookCover">
    <xs:complexType mixed="true">
        <xs:all>
            <xs:element name="BookTitle" type="xs:string"/>
            <xs:element name="Author" type="xs:string"/>
            <xs:element name="Publisher" type="xs:string"/>
        </xs:all>
    </xs:complexType>
</xs:element>
```

- The following XML block is valid in the above schema

```
<BookCover>
    Title:          <BookTitle>The Holy Grail</BookTitle>
    Published:      <Publisher>Moose</Publisher>
    Author:         <Author>Monty Python</Author>
</BookCover>
```

# Referencing XML Schema in XML documents

# Sample Schema header

- The <schema> element may contain some attributes. A schema declaration often looks something like this:
  - <?xml version="1.0"?>

    <xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.w3schools.com"
    elementFormDefault="qualified">

    ... ...

    </xs:schema>

# Schema headers, cont.

- The following fragment:

**xmlns:xs="http://www.w3.org/2001/XMLSchema"**

indicates that the elements and data types used in the schema (schema, element, complexType, sequence, string, boolean, etc.) come from the "http://www.w3.org/2001/XMLSchema" namespace.

It also specifies that the elements and data types that come from the "http://www.w3.org/2001/XMLSchema" namespace should be prefixed with xs:

# Schema header, cont.

- This fragment:
  - targetNamespace="http://www.w3schools.com"
    indicates that the elements defined by this schema will belong the "http://www.w3schools.com"  namespace.
- This fragment:
  - elementFormDefault="qualified“
    indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified, the other possibility is elementFormDefault="unqualified“

# Referencing schema in XML

- This XML document has a reference to an XML Schema:

```
<?xml version="1.0"?>
<note xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

# Referencing schema in xml, cont.

- xmlns="http://www.w3schools.com" specifies the default namespace.
- xmlns:xsi=" http://www.w3.org/2001/XMLSchema-instance " makes the XML Schema Instance namespace available
- xsi:schemaLocation="http://www.w3schools.com note.xsd" specifies to use the schema note.xsd for namespace  http://www.w3schools.com

# Using References

# Using References

- You don't need to have the content of an element defined in the nested fashion as

```
<xs:element name="rooms">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="room">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="capacity" type="xs:decimal"/>
```

- You can define the element globally and use a reference to it instead

```
<xs:element name="rooms">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="room"/>
</xs:sequence></xs:complexType></xs:element>

<xs:element name="room">
  ...
</xs:element>
```

# Rooms Schema using References

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

<xs:element name="rooms">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="room" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="room">
   <xs:complexType>
       <xs:sequence>
            <xs:element name="capacity" type="xs:decimal"/>
            <xs:element ref="features" minOccurs="0" maxOccurs="1"/>
       </xs:sequence>
       <xs:attribute name="name" type="xs:string" use="required"/>
   </xs:complexType>
</xs:element>

<xs:element name="features">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="feature" type="xs:string"
                        maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

</xs:schema>
```

# Types

Both elements and attributes have types, which are defined in the Schema.

One can reuse types by giving them names.

```
<xsd:element name="Robot">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="Sensor_List" minOccurs="0"/>
            <xsd:element ref="Specification_List" minOccurs="0"/>
            <xsd:element ref="Note" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

OR

```
<xsd:element name="Robot" type="RoboType">
<xsd:complexType  name="RoboType" >
        <xsd:sequence>
            <xsd:element ref="Sensor_List" minOccurs="0"/>
            <xsd:element ref="Specification_List" minOccurs="0"/>
            <xsd:element ref="Note" minOccurs="0"/>
        </xsd:sequence>
</xsd:complexType>
</xsd:element>
```

# Validating a Schema

- One can check
  - that an XML file is correct
  - that a schema is correct
  - that an XML file conforms to a given schema

Try the online validator at https://www.liquid-technologies.com/online-xsd-validator

# Exercise

- Define an XML Schema and an XML document conforming to the schema about university degrees.
- A degree is a set of courses
- Each course has one or more professors, a title and a number of CFUs
- Each professor has a name and a matricula id