LAB # 1

Many goals for this lab: identity, classify and retrieve data from data banks; import data into applications; practice with XML language to acquire skills of semi-structured data model.

Data Source

Commonly speaking **data banks** are repository of different types of data, classified on the basis of their main characteristics and purposes. Typically a data bank has a web interface, such the user can search for data of interest.

Commonly speaking data banks equal **databases**, **but it is not the reality**. Data banks offer data typically registered in text file (even in a spreadsheet), also called **flat file**.

Flat files generally includes only primary data types (number, words, date, time) and are commonly identified as "**raw**".

Interesting data banks (offering a web interface):

(1) **Eurostat** (`http://ec.europa.eu/eurostat`). Web tools help users to select data and gather raw data in a file having one of the common format `.csv`, `.tsv`, `.html`, `.pdf`;

(2) **European Social Survey** (`https://www.europeansocialsurvey.org/`). Offers a full data set, including all themes. A selection of themes can be download as a `.csv` text file.

(3) **Organisation for Economic Co-operation and Development (OECD)** (`https://data.oecd.org/`). Data is available in a visual form, but raw data can be downloaded as a `.csv` text file.

(4) **World Health Organization (WHO)** `https://www.who.int/`. The organization manages and maintains a wide range of data collections related to global health and well-being. Full raw data are generally available as a `.csv` text file.

Data Integration

Data integration, in common sense, means to exchange data among different software applications. For example data available in the spreadsheet could be statistically analyzed by means of R software. Mostly, software applications have the capability to recognize common data format.

Which format?

(a) Comma-Separated Value (CSV). Each record is located on a separate line, delimited by a line break. Within the header and each record, there may be one or more fields, separated by commas. Each line should contain the same number of fields throughout the file. The last field in the record must not be followed by a comma.

- DATA-FOOD-COMMA.CSV

```
Food,Weight,Unit,Price
Fresh Chopped Tomatoes,340,g,2.4
Black Olive Paste,90,g,2.7
Organic Extra Virgin Olive Oil,0.51,l,8.9
Pistachio Pesto,180,g,8.9
Artichokes in Oil,180,g,6.98
```

- DATA-FOOD-SEMICOLON.CSV

```
Food;Weight;Unit;Price
Fresh Chopped Tomatoes;340;g;2,4
Black Olive Paste;90;g;2,7
Organic Extra Virgin Olive Oil;0,51;l;8,9
Pistachio Pesto;180;g;8,9
Artichokes in Oil;180;g;6,98
```

(b) Text (txt). Each record is represented as a single line. fields in a record are separated from each other by a space character. Each line must contain the same number of fields. Data looks like arranged in a table.

DATA-FOOD-TEXT.TXT

```
Food Weight Unit Price
"Fresh Chopped Tomatoes" 340 g 2.40
"Black Olive Paste" 90 g 2.70
"Organic Extra Virgin Olive Oil" 0.51 l 8.90
"Pistachio Pesto" 180 g 8.90
"Artichokes in Oil" 180 g 6.98
```

(c) **Hyper Text Markup Language (HTML)**. It is a markup language for describing web documents (web pages). A markup language consists in a set of markup tags (HTML tags). Each HTML tag describes different document content and how to visualize it.

DATA-FOOD-HTML.HTML

```
<!DOCTYPE html>
<html>
   <head>
      <title> Selection of Traditional Italian Food </title>
   </head>
   <body>
      <table>
         <tr>
            <th> Food </th>
            <th> Weight </th>
            <th> Unit </th>
            <th> Price </th>
         </tr>
         <tr>
            <th>Fresh Chopped Tomatoes</th>
            <th>340</th>
            <th>g</th>
            <th>2.4</th>
         </tr>
         <tr>
            <th>Black Olive Paste</th>
            <th>90</th>
            <th>g</th>
            <th>2.7</th>
         </tr>
         <tr>
            <th>Organic Extra Virgin Olive Oil</th>
            <th>0.51</th>
            <th>l</th>
            <th>8.9</th>
         </tr>
```

```
            <tr>
               <th>Pistachio Pesto</th>
               <th>180</th>
               <th>g</th>
               <th>8.9</th>
            </tr>
            <tr>
               <th>Artichokes in Oil</th>
               <th>180</th>
               <th>g</th>
               <th>6.98</th>
            </tr>
         </table>
      </body>
   </html>
```

Getting Data

Data is helpful if it can be analyzed by means of software applications.

Spreadsheet

Spreadsheet gives us few functionalities aiming to get data - import - from different data formats. The wizard tool helps user to interpret correctly text file sources. You have to pay attention at primary data formats. Select options as it is more convenient.

Note: Using Excel spreadsheet it is kindly suggested to apply **text column process** **(DATA TAB ribbon)**

Copy and paste the file content:

(a) `csv` data source, comma separated;

(b) `csv` data source, semicolon separated;

(c) `txt` data source, space separated.

Software R

R Software gets data from different data sources:

(a) `df <- read.table("data-food-text.txt", header=TRUE, sep=" ");`

(b) `df <- read.csv("data-food-comma.csv", header=TRUE);`

(c) `df <- read.csv2("data-food-semicolon.csv", header=TRUE);`

These functions create and return a *data frame* object.

Semi-structured Model

It resembles trees or graphs, rather than tables. The principal manifestation of this viewpoint is XML, a way to represent data by hierarchically nested tagged elements. **The tags define the role played by different pieces of data**.

eXtensible Markup Language (XML)

XML is a text based markup language which is standard for **data interchange on the Web**. Data are identified by tags. An example of some XML data for a messaging application:

```
<message>
     <to>you@yourAddress.com</to>
     <from>me@myAddress.com</from>
     <subject>XML is really cool></subject>
     <text>
          How many ways is XML cool? Le me count the ways ...
     </text>
</message>
```

What XML offers?

- Flexible development of user-defined document;
- It can be used for the storage and transmission of data;
- It provides data identification by use of tags;
- It provides styleability using XSLT (eXtensible StyLesheeT);
- It provides a language for querying (XQUERY) and locating (XPATH) elements in an XML document.

Its broad set of applications makes a compelling case for why anyone working with data needs to have some familiarity with it.

Basics

The Language

- The basic unit is the element (node);
- An element has a name and may have attributes;
- An element may have child elements forming a treelike structure;
- Element and attribute names make up an XML vocabulary, **giving meaning to elements and relationship between them**.

Tags

- Each element begins with the **start-tag** (`<name>`) and must close with a corresponding **end-tag** (`</name>`), that is tags are paired and they delimit an **element** and its **contents**;
- Elements can have content made up of other elements, treated as child elements, hence defining a nested structure;
- Child elements are "regular" elements, with start and end tag, or simple text content that have no children;
- A "regular" element can have zero or more attributes, which are `name = "value"`. **Attributes are specified in the start tag of an element**;
- If an XML document obeys to the basic syntax rules it is said to be **well-formed**.

Hierarchical Structure

- The hierarchical nature of XML can be represented as a tree;
- In the tree, the lines connecting elements are called **branches**, and the elements are referred to as **nodes**;
- The **root** is the node at the top of the tree;
- **Leaf-node** are nodes with no children;
- This conceptual model of an XML document as a tree can be very helpful when **processing and navigating the document**.

Application

The report below shows the names of participants to a "special" social group. Data is registered in a text file applying respectively `.csv` and `.xml` format.

name	email	password	birthday	live
Galileo Galilei	ggalileo@gmail.com	glGL!1564	15-02-1564	Pisa
Leonardo da Vinci	leovinci@gmail.com	LDV*1452	15-4-1452	
Michelangelo Merisi	mmerisi@gmail.com	caravaggio1571		

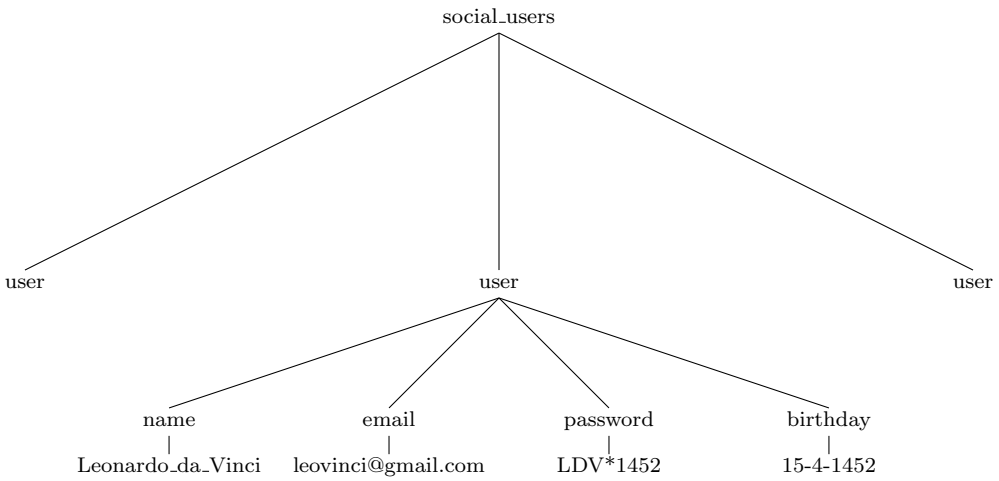
.csv format

```
name,email,password,birthday,live
Galileo Galilei,ggalileo@gmail.com,glGL!1564,15-02-1564,Pisa
Leonardo da Vinci,leovinci@gmail.com,LDV*1452,15-4-1452,
Michelangelo Merisi,mmerisi@gmail.com,caravaggio1571,,
```

.xml format

```
<social_users>
   <user>
      <name>Galileo Galilei</name>
      <email>ggalileo@gmail.com</email>
      <password>glGL!1564</password>
      <birthday>15-02-1564</birthday>
      <live>Pisa</live>
   </user>
   <user>
      <name>Leonardo da Vinci</name>
      <email>leovinci@gmail.com</email>
      <password>LDV*1452</password>
      <birthday>15-4-1452</birthday>
   </user>
   <user>
      <name>Michelangelo Merisi</name>
      <email>mmerisi@gmail.com</email>
      <password>caravaggio1571</password>
   </user>
</social_users>
```

XML tree



XML vs. XSLT

XSLT is a language for transforming XML documents into **HTML**, in order to visualize them by a browser in a nice and clear fashion.

Consider the list.xml file describing the data of a list of food products.

```
<?xml version = "1.0"?>
<?xml-stylesheet type = "text/xsl" href = "list.xsl"?>
<list_products>
   <food>
      <name> Fresh Chopped Tomatoes </name>
      <weight unit="g"> 340 </weight>
      <price currency="euro"> 2,4 </price>
   </food>
   <food>
      <name> Black Olive Paste </name>
      <weight unit="g"> 90 </weight>
      <price currency="euro"> 2,7 </price>
   </food>
   <food>
      <name> Organic Extra Virgin Olive Oil </name>
      <weight unit="l"> 0,51 </weight>
      <price currency="euro"> 8,9 </price>
   </food>
   <food>
      <name> Pistachio Pesto </name>
      <weight unit="g"> 180 </weight>
      <price currency="euro"> 8,9 </price>
   </food>
   <food>
      <name> Artichokes in Oil </name>
      <weight unit="g"> 180 </weight>
      <price currency="euro"> 6,98 </price>
   </food>
</list_products>
```

Note: Observe the two lines at the top, the **declaration** tags. Relevant is the declaration tag `<?xml-stylesheet type = "text/xsl" href = "list.xsl"?>`, which specifies a XSL Style Sheet for the XML document.

The file LIST.XSL describes the XSL Style Sheet.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsl:stylesheet version = "1.0"  xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
   <xsl:template match = "/">
      <html>
         <body>
            <h2> Some Foods </h2>
            <table border = "1">
               <tr bgcolor = "#ffd700">
                  <th> Name </th>
                  <th> Weight </th>
                  <th> Unit </th>
                  <th> Price </th>
                  <th> Currency </th>
               </tr>

               <xsl:for-each select = "list_products/food">
                  <tr>
                     <td><xsl:value-of select = "name"/></td>
                     <td><xsl:value-of select = "weight"/></td>
                     <td><xsl:value-of select = "weight/@unit"/></td>
                     <td><xsl:value-of select = "price"/></td>
                     <td><xsl:value-of select = "price/@currency"/></td>
                  </tr>
               </xsl:for-each>
            </table>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

Note: XSLT uses XPATH to find information navigating in the XML document. The **select** attribute contains XPATH expression to locate the data, for example for each **list_products/food** elements it locates **price//@currency** (price element, attribute currency)

Now, the browser displays the data in LIST.XML formatted, as below.

Some Foods

Name	Weight	Unit	Price	Currency
Fresh Chopped Tomatoes	340	g	2,4	euro
Black Olive Paste	90	g	2,7	euro
Organic Extra Virgin Olive Oil	0,51	l	8,9	euro
Pistachio Pesto	180	g	8,9	euro
Artichokes in Oil	180	g	6,98	euro

Getting Data

Software R

R Software gets data from XML data source (`.xml`).

```
library(XML);
df <- xmlToDataFrame("list.xml");
```

These functions create and return a *data frame* object.

Another Language for Data: JSON

JavaScript Object Notation (JSON) has rapidly become the format for **client web-based** applications. It represents a lightweight data-interchange format easy for humans to read and write. JSON is based on features that are inherent in JavaScript language, the language of web pages.

A JSON instance is an **object**:

- An object is an unordered set of name / value pairs. An object begins with "{" left brace and ends with "}" right brace. Each name is followed by : (colon) and the name / value pairs are separated by , (comma).
- An array is an ordered collection of values. An array begins with [left bracket and ends with] right bracket. Values are separated by , (comma).
- A value can be a string in double quotes, or a number, or true or false or null, or an object or an array.
- A string is a sequence of zero or more characters, wrapped in double quotes, using backslash escapes.

```
{
   "list_products" :
   {
      "food" :
         {
          "name" : "Fresh Chopped Tomatoes",
          "weight" : {"value": 340, "unit": "g"},
          "price" : {"value": 2.4, "currency": "euro"}
         },
      "food" :
         {
          "name" : "Black Olive Paste",
          "weight" : {"value": 90, "unit": "g"},
          "price" : {"value": 2.7, "currency": "euro"}
         },
      "food" :
         {
          "name" : "Organic Extra Virgin Olive Oil",
          "weight" : {"value": 0.51, "unit": "l"},
          "price" : {"value": 8.9, "currency": "euro"}
         },
      "food" :
         {
          "name" : "Pistachio Pesto",
          "weight" : {"value": 180, "unit": "g"},
          "price" : {"value": 8.9, "currency": "euro"}
         }
   }
}
```

Table 1: File `data-food.json`

Exercises (XML Applications)

Ex. # 1

"Eurostat offers different types of data related to social aspects of life. We aim to investigate on intentional homicide victims - distinct by male and female - trends in the European countries and in the main cities. Firstly we compare the number of victims in Germany and Italy in the year 2018. Data is arranged in the table".

2018	Total	Male (%)	Female (%)
Germany	788	53.4%	46.6%
Italy	345	61.4%	38.6%

Table 2: Intentional homicide victims

Describe by means of the XML language the data. Specify only one country (at your choice), but **do not omit to specify all deduced information**.

We need male and female total units! Is it necessary to describe them?

```
<homicide>
   <country>
      <name> Germany </name>
      <year> 2018 </year>
      <numbers>
         <total> 788 </total>
         <male unit="%"> 53.4 </male>
         <female unit="%"> 46.6 </female>
      </numbers>
   </country>
</homicide>
```

It is not necessary to describe male and female as total units because these numbers can be easily computed.

Ex. # 2

"Every year, millions of people visit museums, from the most famous to the lesser known, representing a significant industry of a country's economy. The table compare the **number of museums and visitors in 2017** in some countries of the European Union, with different historical, cultural and geographical characteristics."

		# Museums			
Nation	**Population**	**Art, Archaeology, History**	**Science and Technology**	**Others**	**Visitors**
France	64.694.497	805	341	78	63.199.181
Italy	60.589.445	2.656	1.184	1.136	50.263.520

Table 3: Cultural Statistics - 2017

Describe by means of the XML language the data. Specify only one country (at your choice), but **do not omit to specify all deduced information**.

A significant information is the average visitors per museum. Do we have to add this information to the XML description?

```
<statistics year="2017">
   <nation>
      <name> France </name>
      <population> 64694497 </population>
      <museums unit="number">
         <art-arch-hist> 805 </art-arch-hist>
         <science-tech> 341 </science-tech>
         <others> 78 </others>
      </museums>
      <visitors unit="number"> 63199181 </visitors>
   </nation>
</statistics>
```

It is not necessary to add this information because the average visitors per museum is the number of visitors over the total number of museums.

Ex. # 3

"In the **Big Data** era traditional model to describe data are not sufficient, and therefore we have to consider **semi-structured** and **unstructured** model".

Consider the following message:

Date - time	2019-12-20 18:40
From:	elisabeth.collins@abc.com
To:	alice.may@abc.com, peter.smith@abc.com
Cc:	facility.service@abc.com, administration@abc.com
Object:	Meeting January, 6 2020
Text:	Dears, we inform you that the meeting planned on January, 6 2020 at 17:00 is moved to January, 10 2020 at 18:00. Please update your agenda!

It seems reasonable to describe this message assuming the **semi-structured** model. Write down the corresponding XML file.

```
<message>
   <date-time>
      <date> 2019-12-20 </date>
      <time> 18:40 </time>
   </date-time>
   <from> elisabeth.collins@abc.com </from>
   <to>
      <first> alice.may@abc.com </first>
      <second> peter.smith@abc.com </second>
   </to>
   <cc>
      <first> facility.service@abc.com </first>
      <second> administration@abc.com </second>
   </cc>
   <object> Meeting January, 6 2020 </object>
   <text> Dears, we inform you that the meeting planned on January, 6 2020
          at 17:00 is moved to January, 10 2020 at 18:00.
          Please update your agenda!
   </text>
</message>
```

Ex. # 4

Consider the two **semi-structured** data models (XML) describing data about **Italy** in the year **2014**.

```
<statistical_profile>
   <data>
      <year> 2014 </year>
      <country> Italy </country>
      <population>
         <inhabitants unit="thousands"> 60447 </inhabitants>
         <growth_rate> 0.3705803 </growth_rate>
      </population>
   </data>
</statistical_profile>
```

```
<statistical_profile>
   <country>
      <name> Italy </name>
      <production>
         <year> 2014 </year>
         <GDP>
            <pro_capita> 36070.848 </pro_capita>
            <growth_rate> 0.11367 </growth_rate>
         </GDP>
     </production>
   </country>
</statistical_profile>
```

It seems reasonable to describe the given data into a single, lossless information, **semi-structured** data model. Write down the corresponding XML file.

```
<statistical_profile>
   <country>
      <name> Italy </name>
      <year> 2014 </year>
      <population>
          <inhabitants unit="thousands"> 60447 </inhabitants>
          <growth_rate> 0.3705803 </growth_rate>
      </population>
      <production>
         <GDP>
            <pro_capita> 36070.848 </pro_capita>
            <growth_rate> 0.11367 </growth_rate>
         </GDP>
      </production>
   </country>
</statistical_profile>
```

Ex. # 5

"Currently the Web offers a variety of data, even if it is embodied into HTML code". Marketing needs to compare Internet service offers of different providers. Consider the following Internet offers data and describe it by means of the XML language (**consider just one offer at your choice!**)

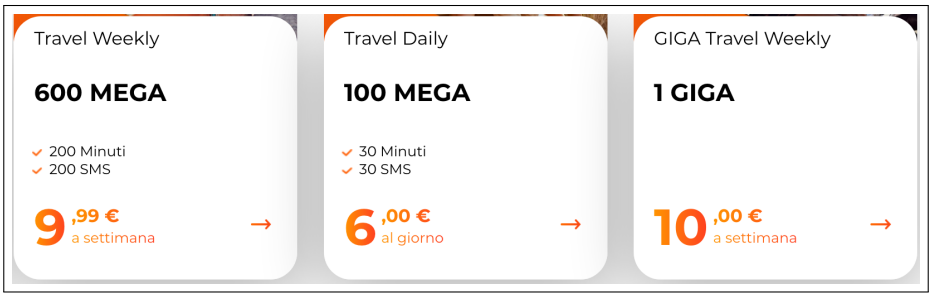


Figure 1: source: **www.windtre.it**: Internet offers to travel abroad

---

```
<Internet-offers>
   <offer>
      <name> Travel Weekly </name>
      <options>
         <internet unit="mega"> 600 </internet>
         <minutes> 200 </minutes>
         <SMS> 200 </SMS>
      </options>
      <price frequency="weekly" currency="euro"> 9,99 </price>
   </offer>
</Internet-offers>
```

Ex. # 6

"Are European people really aware of climate change? And what do they think about causes, impact and responsibilities of global warming?". A significant sample of European people were interviewed proposing the following question and the codified answers.

# clmchng: Do you think world's climate is changing	
Information: [Type= discrete] [Format=numeric] [Range= 1-4] [Missing=*/7/8/9]	
Value	**Label**
1	Definitely changing
2	Probably changing
3	Probably not changing
4	Definitely not changing
7	Refusal
8	Don't know
9	No answer

Describe the data through **XML** language.

```
<climate>
   <question> Do you think world's climate is changing </question>
   <information>
      <type> discrete </type>
      <format> numeric </format>
      <range> 1-4 </range>
      <missing> */7/8/9 </missing>
   </information>
   <answers>
      <a>
         <code> 1 </code>
         <label> Definitely changing </label>
      </a>
      <a>
         <code> 2 </code>
         <label> Probably changing </label>
      </a>
      <a>
         <code> 3 </code>
         <label> Probably not changing </label>
      </a>
      <a>
         <code> 4 </code>
         <label> Definitely not changing </label>
```

```
      </a>
      <a>
         <code> 7 </code>
         <label> Refusal </label>
      </a>
      <a>
         <code> 8 </code>
         <label> Don't know </label>
      </a>
      <a>
         <code> 9 </code>
         <label> No answer </label>
      </a>
   </answers>
</climate>
```

Ex. # 7

"Online shops are mostly used to buy cloths". Consider the promotional advertisement below of an online shop for kids. Report data in a semi-structured data model (XML Language). **Describe at least two cloths!**

[**Tip:** the small numbers at the bottom correspond to cloth sizes.]

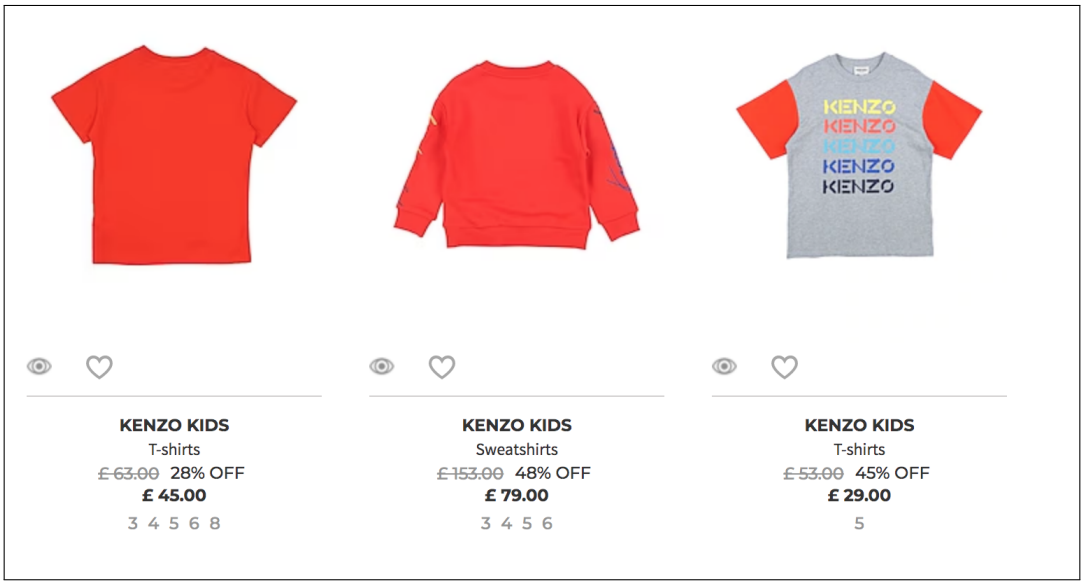


Figure 2: source: **YOOX Online Shop**

```
<shop-kids>
   <brand>
      <name> Kenzo Kids </name>
      <cloths>
         <item>
            <type> Sweatshirts </type>
            <price currency="£"> 153.00 </price>
            <save unit="%"> 48 </save>
            <sizes>
               <s1> 3 </s1>
               <s2> 4 </s2>
               <s3> 5 </s3>
               <s4> 6 </s4>
            </sizes>
         </item>
```

```
        <item>
           <type> T-shirts </type>
           <price currency="£"> 53.00 </price>
           <save unit="%"> 45 </save>
           <sizes>
              <s1> 5 </s1>
           </sizes>
        </item>
      </cloths>
   </brand>
</shop-kids>
```

Ex. # 8

"**Metadata** is defined as the data that provides information about other data". Assume the data set **pollution** below, and write the **XML** file which captures its main features. For example we observe that it has five variables, the first one has the heading *Lat.* and it is numerical (real numbers with six decimal digits). Further the variable with heading **Date** holds dates having the format `dd/mm/yyyy`. Your metadata file must describes all variables.

Lat.;Long.;Altitude;Date; PM10 - $\mu g/m^3$
45,462894;9,195176;130;16/12/2019;40
45,462894;9,195176;130;17/12/2019;45
45,462894;9,195176;130;18/12/2019;44
45,462894;9,195176;130;19/12/2019;28
45,462894;9,195176;130;20/12/2019;15

```
<metadata>
   <data-set> pollution </data-set>
   <nr-var> 5 </nr-var>
   <variables>
      <var>
         <name> Lat. </name>
         <type> numeric </type>
         <decimals> 6 </decimals>
      </var>
      <var>
         <name> Long. </name>
         <type> numeric </type>
         <decimals> 6 </decimals>
      </var>
      <var>
         <name> Altitude </name>
         <type> integer </type>
      </var>
      <var>
         <name> Date </name>
         <format> dd/mm/yyyy </format>
      </var>
      <var>
         <name unit="micro-gr/m^3"> PM10 </name>
         <type> integer </type>
      </var>
   </variables>
</metadata>
```

Ex. # 9

"Currently the Web shows a variety of data in effective way". Marketing needs to compare one night stay offers in luxury hotels. Consider the following data of a room availability and describe it by means of the XML language. Data are arranged in a table, hence you could describe it by row or by column (describe a very **small** part of the data).

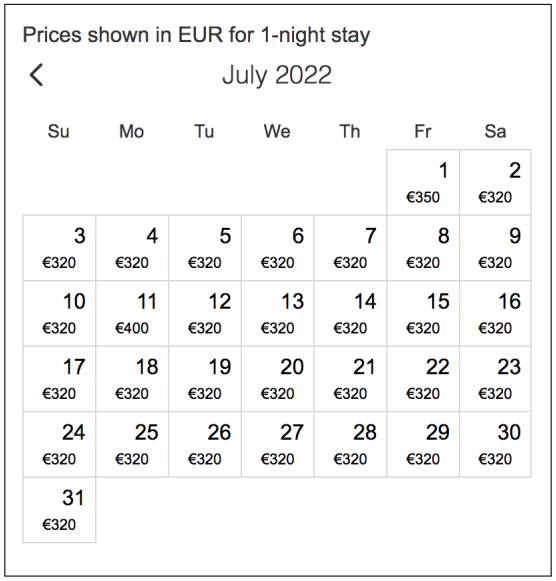


Figure 3: source: **https://grandhotelmajestic.duetorrihotels.com/**

```
<availability>
   <ref currency="euro"> 1-night stay </ref>
   <time>
      <year> 2022 </year>
      <month> July </month>
   </time>
   <table>
      <column>
         <day> Monday </day>
         <case>
            <num> 4 </num>
            <price> 320 </price>
         </case>
         <case>
            <num> 11 </num>
            <price> 400 </price>
```

```
         </case>
         <case>
            <num> 18 </num>
            <price> 320 </price>
         </case>
         <case>
            <num> 25 </num>
            <price> 320 </price>
         </case>
      </column>
   </table>
</availability>
```

Ex. # 10

"Mostly restaurants show menus on the web". Consider part of the menu of "Ristorante Bologna" where courses are in two languages. Report shown data in a semi-structured data model (XML Language). **Describe at least two courses!**

RISTORANTE BOLOGNA
Menù
PRIMI PIATTI
[FIRST COURSE]
Agnolotti burro e salvia
Agnolotti with butter and sage
€ 20.00
Tagliolini al ragù di carne
Tagliolini with meat sauce
€ 16.00
DESSERT
[DESSERT]
Sformato di cioccolato caldo
Hot chocolate flan
€ 10.00
Creme brulé
€ 10.00
Mousse al caffé
Coffee Mousse
€ 10.00

```
<menu_web>
   <title type="restaurant"> RISTORANTE BOLOGNA </title>
   <document> menù </document>
   <item>
      <category>
         <ita> PRIMI PIATTI </ita>
         <eng> FIRST COURSE </eng>
      </category>
      <course>
         <ita> Agnolotti burro e salvia </ita>
         <eng> Agnolotti with butter and sage </eng>
         <price currency="Euro"> 20.00 </price>
      </course>
   </item>
```

```
   <item>
      <category>
         <ita> DESSERT </ita>
         <eng> DESSERT </eng>
      </category>
      <course>
         <ita> Sformato di cioccolato caldo </ita>
         <eng> Hot chocolate flan </eng>
         <price currency="Euro"> 10.00 </price>
      </course>
   </item>
</menu_web>
```

References

This lab is mainly based on:

(1) Request for Comments: 4180, Common Format and MIME Type for Comma-Separated Values (CSV) Files

(2) Definition of tab-separated-values (tsv), www.iana.org

(3) HTML(5) Tutorial, `http://www.w3schools.com`

(3) XML Tutorial, `http://www.w3schools.com`

(4) *D. Nolan* and *D. Tample*, "XML and Web Technologies for Data Sciences with R", 2014

(5) JSON Tutorial, `http://www.w3schools.com`