COMP20008
Elements of data processing

Semester 1 2020

Lecture 3: Data formats

Student representatives

Lucas Fern lfern@student.unimelb.edu.au

Sanjog Gururaj sgururaj@student.unimelb.edu.au

Please contact them with any feedback and suggestions you have about this subject

Changes to assessments

- Assignment one will be worth 25%
- Assignment two also 25%
- No oral presentation (originally worth 10 %)

- Final exam remains 50%
- Hurdle requirements unchanged
  - 20/50 for assignments
  - 20/50 for exam

A data scenario

Next week we are going to make a presentation to the CEO and we need to include a profile of all our customers: who they are, their past purchasing behaviour and all the types of interactions they've had with our company.

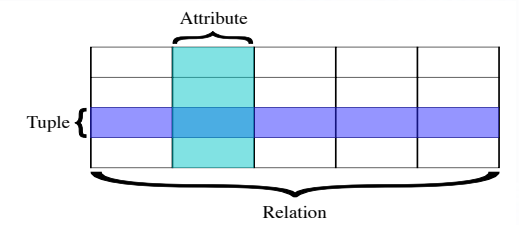
The data exists, kind of, but it is spread across multiple systems and is in **many** types of **formats***: CSV, XML, JSON, HTML, spreadsheets, ….*

Data formats

- How is data stored and in what formats?
  - *Structured*: Relational databases, excel, …
  - *Unstructured*: text, images
  - *Semi-structured*: CSV,HTML, XML, JSON, …
- Why do we have different data formats
- Why do we wish to transform between different formats?

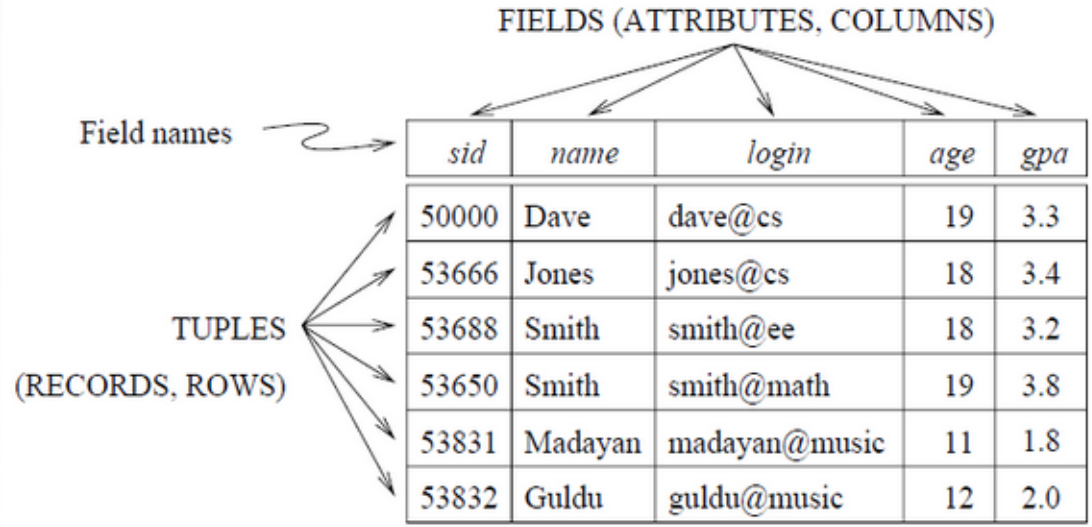
Structured data – Relational databases

- It is good to have structure for data!
  - Easier to analyse, easier to query, easier to store
  - Easier to clean, maintain consistency and security, especially with multiple users



- Relational databases, the classic method of storing structured data (banking, sales, airlines …), as a table
- Can query the data using a high-level language such as SQL

Example – a relation table

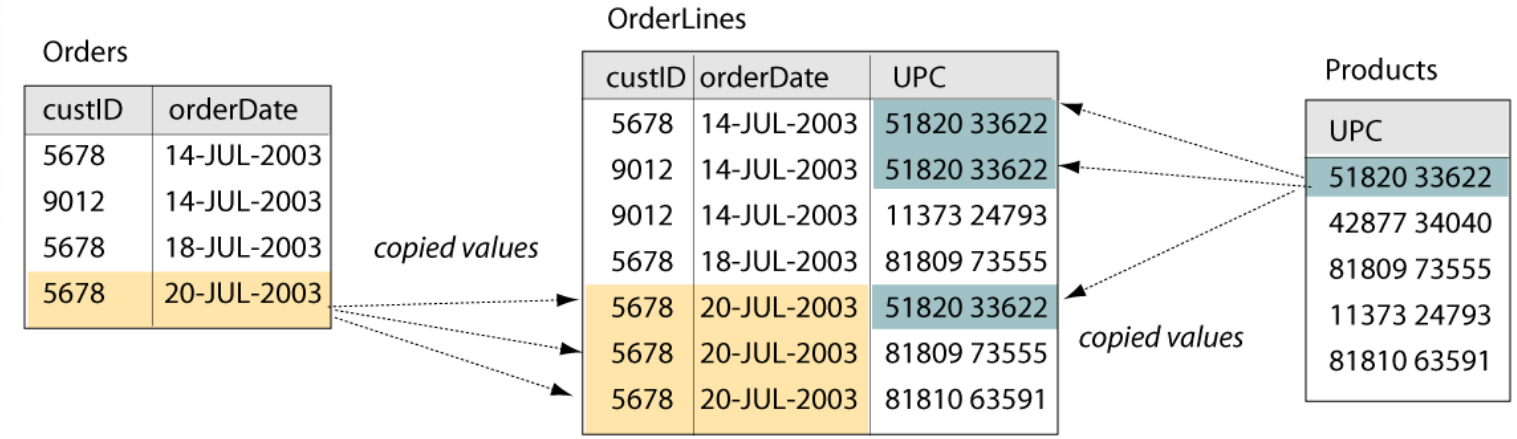


sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure 3.1 An Instance $S1$ of the Students Relation

https://www.sites.google.com/site/merasemester/dbm/relational-model?tmpl=%2Fsystem%2Fapp%2Ftemplates%2Fprint%2F&showPrintDialog=1

Sample relational database



https://web.csulb.edu/colleges/coe/cecs/dbdesign/dbdesign.php?page=manymany.php

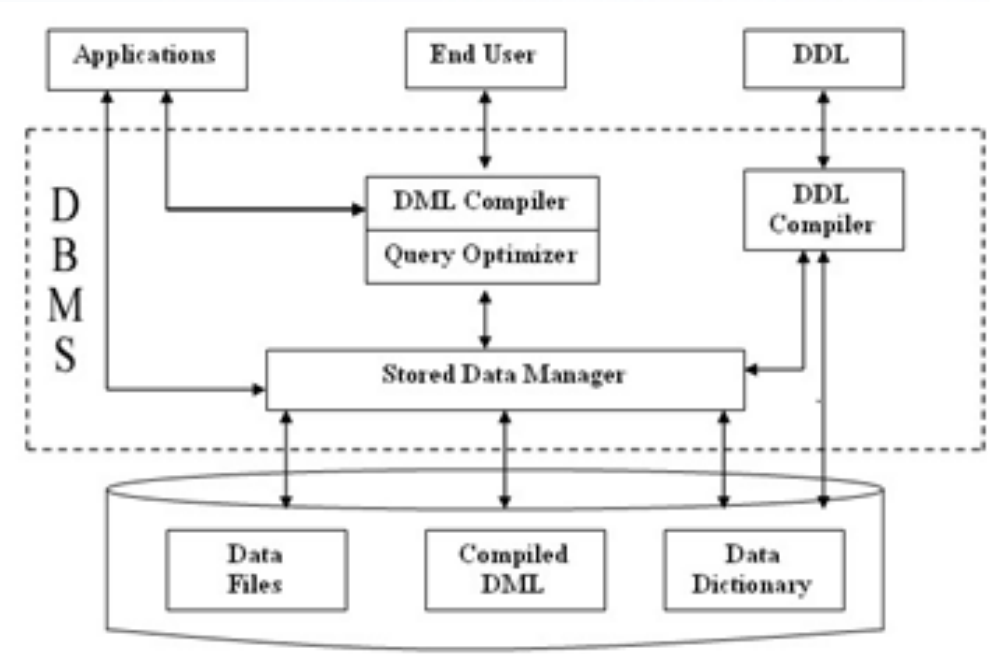
SQL – a language used for relational databases

create table *branch*
(*branch_name* char(15) **not null,**
branch_city char(30),
assets integer,
primary key (branch_name))

SQL

select *account.balance*
from *depositor, account*
where *depositor.customer_id* = ‘192-83-7465’ **and**
depositor.account_number = account.account_number

DBMS – Database management system



http://www.expertsmind.com/questions/physical-dbms-architecture-30139088.aspx

Database systems – (INFO20003)

- INFO20003 covers related topics including
  - SQL
  - Specification of integrity constraints
  - Data modelling and relational database management systems
  - Transactions and concurrency control
  - Storage management
  - Web-based databases

- Highly relevant to data wrangling!
  - Useful to do INFO20003 as part of a data science specialisation

Challenges

- Once data is into a relational database, it is easier to wrangle.
- But may be difficult to load it there in the first place …

More structure - Spreadsheets

- Huge amounts of data is in spreadsheets
  - Businesses
  - Hospitals
  - ….
- Microsoft (Excel), OpenOffice (Calc), Google docs

Information in tabular format (transactional, simple but many entries)

	A	B	C	D	E	F	G	H	I	J	K	L
1		0-4	5-9	10-14	15-19	20-24	25-29	30-34	35-39	40-44	45-49	50-54
2	1901	434741	456981	434152	378115	350993	320544	292071	272710	221190	155009	119072
3	1911	525633	453246	428161	448536	446270	388376	330960	291432	269518	241616	192919
4	1921	603600	597300	530800	470600	450100	462000	448200	390200	332500	283600	255100
5	1922	611900	602500	546100	482200	456000	457900	458700	402700	344400	287800	261700
6	1923	623200	601800	560600	497300	461500	456800	465200	419800	355600	296300	266800
7	1924	633100	593800	579300	512500	468200	455600	469700	434800	368400	305900	271900
8	1925	642600	590100	596400	529800	480100	465100	472600	446100	379900	317800	274000
9	1926	640800	602200	606100	545700	493400	472500	475000	455900	391600	329000	276100
10	1927	637000	613200	613000	563800	510500	485000	474400	468100	405300	341000	279900
11	1928	634700	6267									
12	1929	631600	6375									
13	1930	621400	6440									
14	1931	611500	6378									
15	1932	594000	6295									
16	1933	574000	6240									
17	1934	555100	6201									
18	1935	539700	6112									
19	1936	529700	6025									
20	1937	536000	5865									
21	1938	545200	5679									
22	1939	559000	5508									
23	1940	572400	5364									
24	1941	588600	5280									
25	1942	610200	5354									

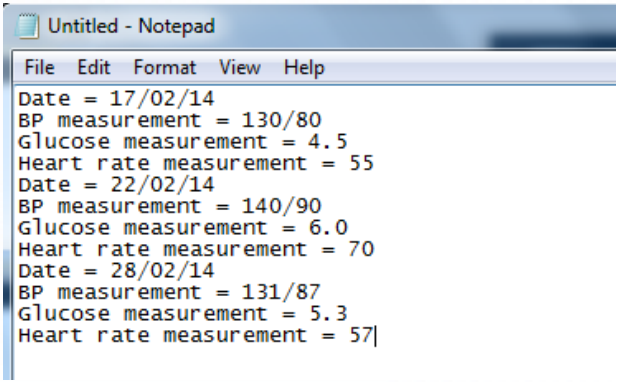
	A	B	C	D	E	F	G
1	Rank	Album	Artist	Year	Total Sales	Origin	Genre
2	1	GREATEST HITS	QUEEN	1981	5678610	UK	Rock
3	2	SGT. PEPPER'S LONELY HEARTS CLUB BAND	BEATLES	1967	4908288	UK	Rock,Pop
4	3	GOLD - GREATEST HITS	ABBA	1992	4610813	Sweden	Pop
5	4	WHAT'S THE STORY MORNING GLORY	OASIS	1996	4421505	UK	Rock
6	5	BROTHERS IN ARMS	DIRE STRAITS	1985	4069764	UK	Rock
7	6	THE DARK SIDE OF THE MOON	PINK FLOYD	1973	3956177	UK	Rock
8	7	THRILLER	MICHAEL JACKSON	1982	3825857	USA	Pop
9	8	GREATEST HITS II	QUEEN	2000	3746404	UK	Rock
10	9	BAD	MICHAEL JACKSON	1987	3554301	USA	Pop
11	10	THE IMMACULATE COLLECTION	MADONNA	1990	3402160	USA	Pop
12	11	STARS	SIMPLY RED	1991	3401092	UK	Pop
13	12	COME ON OVER	SHANIA TWAIN	1998	3358941	Canada	Country,Pop
14	13	RUMOURS	FLEETWOOD MAC	1977	3253818	UK	Rock
15	14	BACK TO BEDLAM	JAMES BLUNT	2004	3172069	UK	Pop
16	15	URBAN HYMNS	VERVE	1997	3167875	UK	Pop
17	16	NO ANGEL	DIDO	2003	3048208	UK	Pop
18	17	BRIDGE OVER TROUBLED WATER	SIMON & GARFUNKEL	1970	3047242	USA	Folk
19	18	BACK TO BLACK	AMY WINEHOUSE	2006	2985303	UK	Retro Soul
20	19	TALK ON CORNERS	THE CORRS	1997	2947666	Ireland	Rock
21	20	BAT OUT OF HELL	MEAT LOAF	1978	2942717	USA	Rock
22	21	SPICE	SPICE GIRLS	1996	2928739	UK	Pop
23	22	WHITE LADDER	DAVID GRAY	2000	2906785	UK	Alternative Rock,Folk
24	23	DIRTY DANCING	ORIGINAL SOUNDTRACK	1987	2892247	UK	Soundtrack

Unstructured data - Text

Text files…

- No structure
- Harder to index
- Harder to organise
- Lacks regularity and decomposable internal structure
- How can we process and search for textual information?

More on text data next week.



Semi-structured data

CSV: comma separated values

- Also very popular
- Also stores tabular information
- Just a **delimited text file** with extension .csv
  - human readable, versus binary XLS format (Excel)
  - Manipulate with any text editor
- Lacks formatting information
- Does not contain formulas and macros (data verification, transformation)

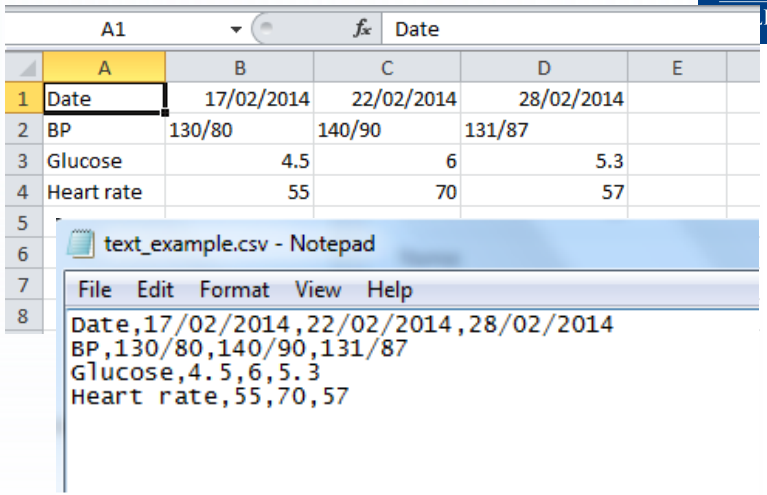
Example – CSV

Spreadsheets

Easy to use

Structured, but not

like excel or a relational DB



A1 | Date

	A	B	C	D	E
1	Date	17/02/2014	22/02/2014	28/02/2014	
2	BP	130/80	140/90	131/87	
3	Glucose	4.5	6	5.3	
4	Heart rate	55	70	57	
5					
6					
7					
8					

text_example.csv - Notepad

File Edit Format View Help

```
Date,17/02/2014,22/02/2014,28/02/2014
BP,130/80,140/90,131/87
Glucose,4.5,6,5.3
Heart rate,55,70,57
```

HTML – Hypertext Markup language

- Marked up with *elements*, delineated by start and end *tags*.
- Elements correspond to logical units, such as a heading, paragraph or itemised list.
- *Tags*: Keywords contained in pairs of angle brackets.
  - **Not** case sensitive.
- Browser determines how to display/present the logical units
- Not all elements need both start and end tags.
- Elements can have *attributes;* ordering of attributes is not significant.

HTML Example

```
<div class="icon section5">
<h2><a href="about/index.html">About the Melbourne School of Engineering</a></h2>
<ul>
<li><a href="about/dean_welcome.html">Dean's Welcome</a></li>
<li><a href="about/staff.html">Leadership &amp;amp; Professional Staff</a></li>
<li><a href="about/contact.html">Contact Us</a></li>
<li><a href="http://www.ecr.unimelb.edu.au">ECR: Computer Resources</a></li>
<li><a href="intranet/index.html">For Staff (intranet)</a></li>
<li><a href="casual_staff/index.html">For Casual Staff</a></li>
<li><a href="intranet/review/prof_staff.html">Professional Staff Review</a></li>
<li><a href="/about/safety/index.html">Environment, Health &amp;amp; Safety</a></li>
<li><a href="/about/committees/index.html">Committees</a></li>
</ul>
```

Try it yourself: https://www.w3schools.com/html/tryit.asp?filename=tryhtml5_browsers_myhero

HTML examples: https://www.w3schools.com/html/html_lists.asp

Limitations of HTML

- HTML was designed for pure presentation
- **HTML is concerned with formatting not meaning**
  it doesn't matter what it is about, HTML will format it
- HTML is not extensible
  - can't be modified to meet specific domain knowledge
  - browsers have developed their own tags (*<bgsound>*, *<layer>*)
- HTML can be inconsistently applied
  almost everything is rendered somehow
  e.g. <b>is this acceptable?</i>

XML: eXtensible Markup Language

- A ‘meta’ mark-up language
- Extensible: user defined tags
- Facilitate better encoding of semantics

Year of offer	2019
Subject level	Undergraduate Level 2
Subject code	COMP20008
Campus	Parkville
Availability	Semester 1 Semester 2

Subject guide in plain-text

Year of offer = 2019

Subject level = Undergraduate level 2

Subject code = COMP20008

Campus = Parkville

Availability = Semester 1, Semester 2

Subject guide in HTML

```
<html>
    <head><style>.....</style></head>
<body>
<table>
    <tr>
        <th>Year of offer</th>  <td>2019</td>
    </tr>
    <tr>
        <th>Subject level</th>  <td>Undergraduate Level 2</td>
    </tr>
    <tr>
        <th>Subject code</th>   <td>COMP20008</td>
    </tr>
    <tr>
        <th>Campus</th>          <td>Parkville</td>
    </tr>
    <tr>
        <th>Availability</th>   <td><div>Semester 1</div>
                                    <div>Semester 2</div></td>
    </tr>
</table>
</body>
</html>
```

Subject guide in HTML – cont.

```
<html>
<head>
<style>
    table {
        border-collapse: collapse;
        width: 100%;
        font-family: Arial, Helvetica, sans-serif;
    }
    th, td {
        text-align: left;
        padding: 8px;
    }
    tr:nth-child(odd) {background-color: #f1f1f4;}
</style>
</head>
<body>
<table>
    ...
</table>
</body>
</html>
```

Subject guide in HTML – cont.

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  border-collapse: collapse;
  width: 100%;
  font-family: Arial, Helvetica, sans-serif;
}
th, td {
  text-align: left;
  padding: 8px;
}
tr:nth-child(odd) {background-color: #f1f1f4;}
</style>
</head>

<body>
<table >
    <tr>    <th>Year of offer</th>  <td>2019</td>   </tr>
    <tr>    <th>Subject level</th>  <td>Undergraduate Level 2</td>  </tr>
    <tr>    <th>Subject code</th>   <td>COMP20008</td>  </tr>
    <tr>    <th>Campus</th>         <td>Parkville</td>  </tr>
    <tr>    <th>Availability</th>   <td><div>Semester 1</div>
                                        <div>Semester 2</div></td>  </tr>
</table>
</body>
</html>
```

Subject guide in xml


```
<?xml version="1.0"?>
<university name="University Of Melbourne">
    <handbook>
        <subject name="Elements of Data Processing" other_name="Data Wrangling">
            <section>
                <title>Overview</title>

                <year_of_offer>2019</year_of_offer>

                <subject_level>Undergraduate Level 2</subject_level>

                <subject_code>COMP20008</subject_code>

                <campus>Parkville</campus>

                <avability>
                       <semester>1</semester>
                       <semester>2</semester>
                </avability>
            </section>
        </subject>
    </handbook>
</university>
```

XML syntax – well formed

- xml files **must begin with declaration**

  <?xml version="1.0"?>

- xml elements
  - One root element
  - Appropriately nested
  - Start/end tags or Empty tags
  - Attributes in quotes

    <campus> Parkville *</campus>* or *<campus location = "Parkville" />*
  - Case sensitive
- comments

  <!-- comments do not affect the document,
  it's not part of the data that you want to represent -->

XML syntax – cont.

- some characters have special meaning
  - '<' and '&' are strictly illegal inside an element
    - <text>all books & videos are now < AUD 10</text>
    - <text>all books & videos are now < AUD 10</text>

- CDATA (character data) section may be used inside XML element to include large blocks of text, which may contain these special characters such as &, >
  - <![CDATA [… … …]]>
  - <![CDATA [all books & videos are now < AUD 10]]>

Using HTML/XML (Python)

- For HTML scraping, the Beautiful Soup library is good
- For XML, a good library is http://lxml.de/
- Import the XML file into your program as a tree structure:

```
import xml.etree.ElementTree as ET
tree = ET.parse('yourfile.xml')
root = tree.getroot()
```

- Then loop through root with the various methods available:

```
for child in root:
 print child.tag, child.attrib
```

XML applications

- Mathematical Markup Language (MathML)
- ChemML (Chemical Markup Language)
- RSS, SOAP, SVG, …

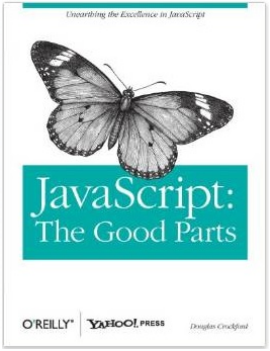
MathML example: markup an equation in terms of presentation and semantics

In MathML, x^3+6x+6 is represented as

```
<mrow>
 <msup>
        <mi>x</mi> <mn>3</mn>
 </msup>
 <mo>+</mo>
 <mrow>
        <mn>6</mn> <mo>&InvisibleTimes;</mo> <mi>x</mi>
 </mrow>
 <mo>+</mo>
 <mn>6</mn>
</mrow>
```

JavaScript Object Notation (JSON)

- JSON (www.json.org)
- Douglas Crockford (pretty much alone)
  - c.f the development of XML by committee
- *"Javascript: the good parts"*
  - O' Reilly, Yahoo Press



JSON

```
{
"Catalog":
  { "CD": {
        "title":  "Empire Burlesque",
        "artist": "Bon Dylan",
        "price": {
            "currency": "USD",
            "value": 10.90
        },
        "year": 1985
    }
  },
  { "CD":  {
        "title": "Hide your heart",
        "artist": "Bonnie Taylor",
        "price": {
            "currency": "USD",
            "value": 9.90
        },
        "year": 1988
    }
  }
]
}
```

```
<CATALOG>
 <CD>
  <TITLE>Empire Burlesque</TITLE>
  <ARTIST>Bob Dylan</ARTIST>
  <PRICE currency = "USD">10.90</PRICE>
  <YEAR>1985</YEAR>
  </CD>

 <CD>
  <TITLE>Hide your heart</TITLE>
  <ARTIST>Bonnie Tyler</ARTIST>
  <PRICE currency = "USD">9.90</PRICE>
  <YEAR>1988</YEAR>
 </CD>
</CATALOG>
```

JSON object example (cont. next slide)


```
{
  "firstName": "David",
  "lastName": "Lynn",
  "isAlive": true,
  "age": 25,
  "height_cm": 167.6,
  "address": {
        "streetAddress": "211 Fox Street",
        "city": "Greenville",
        "state": "NH",
        "postalCode": "80021"
 },
```

JSON example (cont.)


```
"phoneNumbers": [
    {
        "type": "home",
        "number": "315 555-1812"
    },
   {
        "type": "office",
        "number": "646 555-4567"
   }
],
 "email": "dlynn@nhs.net"
}
```

XML representation

```
<?xml version="1.0"?>
<customers>
        <customer>
                <firstname>David</firstname>
                <lastname>Lynn</lastname>
                …..
                <address>
                   <staddress>211 Fox Street</staddress>
                   <city>Greenville</city>
                …..
                </address>
                ….
                <email>dlynn@nhs.net</email>
        </customer>
</customers>
```

JSON syntax rules

- Object data is in name/value pairs

```
"firstName":"John"
```

- JSON values
  - A number (integer or floating point)
  - A string (in double quotes)
  - A Boolean (true or false)
  - An array (in square brackets)
  - An object (in curly braces)
  - null

JSON syntax rules

- JSON Objects

```
{"firstName":"John", "lastName":"Doe"}
```

- JSON Arrays

```
"employees":[
   {"firstName":"John", "lastName":"Doe"},
   {"firstName":"Anna", "lastName":"Smith"},
   {"firstName":"Peter", "lastName":"Jones"}
]
```

- These objects repeat recursively down a hierarchy as needed.
- **In terms of syntax that's pretty much it!**

Using JSON (Python)

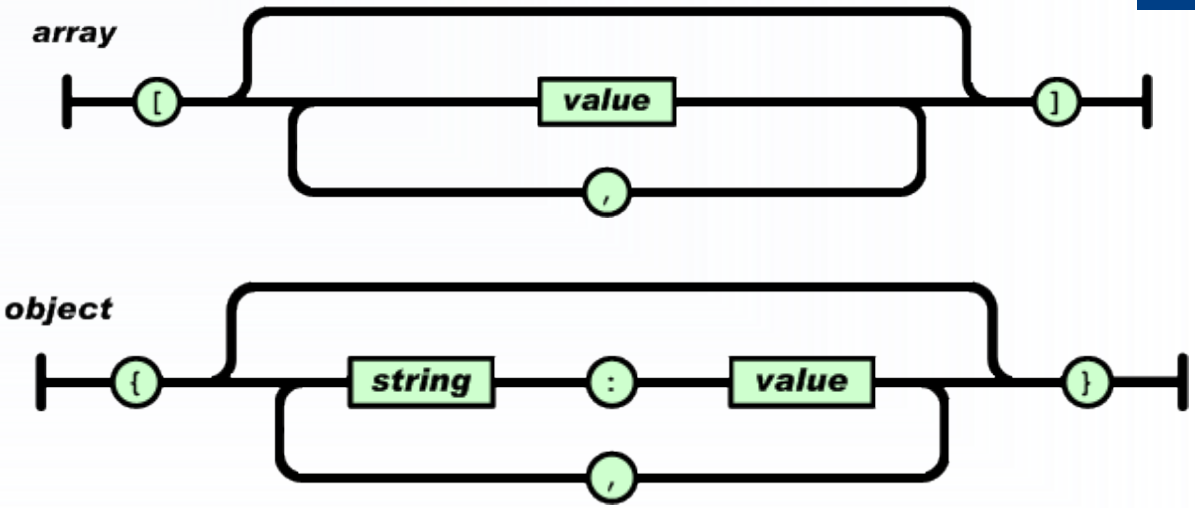
```
import json
json.loads(
    '["foo",
        {"bar":
            ["baz", null, 1.0, 2]
        }
    ]')
json.dumps(
    ['foo',
        {'bar':
            ('baz', None, 1.0, 2)
        }
    ]
)
```

- Load JSON format to Python Dictionary

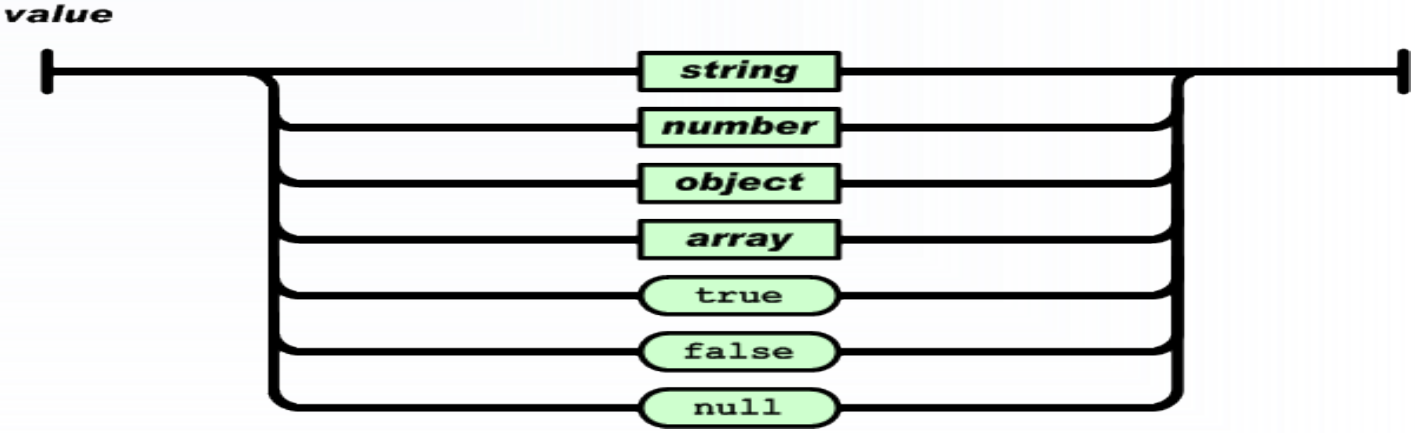
- Convert to JSON format from Python Dictionary

Note: white space and indentation is for human purposes only!

JSON format (from json.org)



JSON format (json.org)



JSON compared to XML

- JSON is simpler and more compact/lightweight than XML; easy to parse.

- Common JSON application – read and display data from a webserver using javascript.
  - https://www.w3schools.com/js/js_json.asp

- XML comes with a large family of other standards for querying and transforming (XQuery, XML Schema, XPATH, XSLT, namespaces, …)

JSON vs XML cont.

- XML allows complex schema definitions (via regular expressions)
  - allows formal validation
  - makes you consider the data design more closely

- JSON is more **streamlined, lightweight and compressed**
  - Which appeals to programmers looking for speed and efficiency
  - Widely used for storing data in noSQL databases

Exercise

Represent the following information in JSON

Check it is well formed: http://jsonlint.com

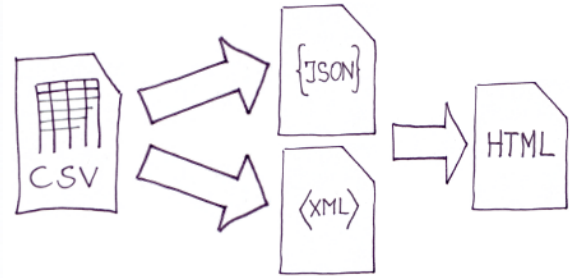
```
<Person>
  <FirstName>Homer</FirstName>
  <LastName>Simpson</LastName>
  <Relatives>
      <Relative>Grandpa</Relative>
     <Relative>Marge</Relative>
     <Relative>Lisa</Relative>
      <Relative>Bart</Relative>
   </Relatives>
   <FavouriteBeer>Duff</FavouriteBeer>
</Person>
```

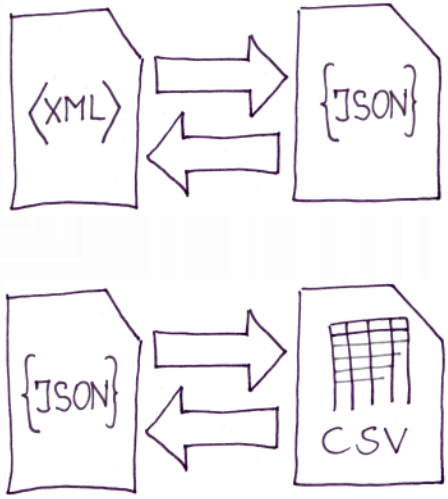
JSON: Summary

- JavaScript Object Notation
- Lightweight, streamlined, standard method of data exchange
- Originally designed to speed up client/server interactions:
  - By running in the client browser
- Native Javascript, so can be executed as code
- Can be used to represent any kind of semi structured data
- Lacks context and schema definitions

Python libraries for JSON and XML

- json
- lxml





XML, JSON, CSV and HTML conversion tools

What you should know

- Why do we have different data formats and why do we wish to transform between different formats?
- Motivation for using relational databases to manage information
- What is a csv, what is a spreadsheet, what is the difference?
- Difference between HTML and XML and when to use each
- Be able to read and write data in XML (elements, attributes)
- Be able to read and write data in JSON
- Difference between XML and JSON; applications where each can be used.

Further reading

- Relational databases

  Pages 403-409 of http://i.stanford.edu/~ullman/focs/ch08.pdf

- XML

  http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html