Text Encoding and Semantic Representation

.* to RDF

marilena.daquino2@unibo.it | https://github.com/marilenadaquino/tesr_dhdk

Outline

RDFLib

XML to RDF - etree + RDFLib

JSON to RDF - json + RDFLib

CSV to RDF - csv + RDFLib

RDFLib

Graphs and Python

The python library RDFLib allows you to manipulate RDF data with some helpful functions. In particular, with RDFLib you can:

- Parse RDF data from strings, files, or URIs into **RDFLib.Graph()** objects
- Iterate over triples and query the data
- Create/delete RDF data from scratch
- Serialize data into any allowed notation and store data in files

RDFLib

How does it work

In Python a graph can be seen as a **list of tuples**.

Each tuple includes 3 elements, namely: the subject URI, the predicate URI, and the Literal/URI of the object.

```python
subj1 = '<http://example.org/robert_capa>'
subj2 = '<http://example.org/gerda_taro>'
pred1 = '<http://example.org/knows>'
pred2 = '<http://example.org/hasSpouse>'
obj1 = '<http://example.org/henri_cartier_bresson>'

# the basic behaviour in RDFlib
example_graph = [
    (subj1, pred1, obj1),
    (subj2, pred2, subj1)
]
```

RDFLib

Parse RDF

RDFlib loads triples in an object called **Graph()**, which comes with a number of convenient methods.

To load triples into a RDFLib graph, you can parse data from a string, a file, or a URI.

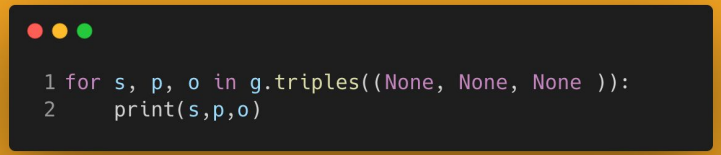
```python
import rdflib

example_data = """
    <http://example.org/robert_capa> <http://example.org/knows>
<http://example.org/henri_cartier_bresson> .
    <http://example.org/gerda_taro> <http://example.org/hasSpouse>
<http://example.org/robert_capa> .
    """
g = rdflib.Graph()

# parse data from string
g.parse(data=example_data, format='nt')
# get the number of statements (triples)
print(len(g))

# parse data from file
g.parse("7_robert_capa.rdf", format="ttl")
print(len(g))

# parse data from URI
g.parse("https://dbpedia.org/resource/Robert_Capa")
print(len(g))
```

RDFLib

Iterate over triples

To access the content of triples/tuples, we can use the method **triples()**, which accepts as parameter a tuple with 3 elements.

The placeholder **None** is used to specify which variable in the triple we want to return.



RDFLib

Iterate over triples

We can replace placeholders with URIs or literal values to see triples in which those terms appear.

To do so, RDFLib requires you to specify the type of resource, i.e. a **URIRef()** or a **Literal()**

```python
from rdflib import URIRef, Literal

# print triples where a certain URI is subject
for s, p, o in g.triples(( URIRef("http://dbpedia.org/resource/Photojournalism"), None, None )):
    print(s,p,o)

# print all the values of a certain property
for s, p, o in g.triples(( URIRef("http://dbpedia.org/resource/Robert_Capa"),
 URIRef("http://dbpedia.org/property/birthName"), None )):
    print(s,p,o)
```

RDFLib

Namespaces

To specify the type of a literal, e.g. a **date**, we need to specify the datatype, which requires us to use the namespace of <xsd>.

RDFLib allows you to directly import some of the most common **namespaces**, e.g. RDFS, RDF, XSD

```python
from rdflib import URIRef, Literal
from rdflib.namespace import XSD

# print triples where a certain URI is subject
for s, p, o in g.triples(( URIRef("http://dbpedia.org/resource/Photojournalism"), None, None )):
    print(s,p,o)

# print all the values of a certain property
for s, p, o in g.triples(( URIRef("http://dbpedia.org/resource/Robert_Capa"),
  URIRef("http://dbpedia.org/property/birthName"), None )):
    print(s,p,o)

# print all the properties of a certain value
for s, p, o in g.triples(( None, None, Literal("1954-05-25",datatype=XSD.date) )):
    print(s,p,o)
```

RDFLib

Namespaces

Less common namespaces can be declared too, which also allow us to make more readable the iteration over triples.

```python
from rdflib import URIRef, Literal
from rdflib.namespace import XSD
from rdflib import Namespace

DBO = Namespace("http://dbpedia.org/ontology/")

# print all the values of a certain property
for s, p, o in g.triples(( None, DBO.birthDate, None )):
    print(s,p,o)
```

Exercise

Practise RDFLib

In a terminal window, install the library **`pip install rdflib`**

Create a new python file

- Parse data from the URI https://dbpedia.org/resource/Gerda_Taro
- Iterate over the triples and return:
  - Her birth date
  - All the properties relating her to http://dbpedia.org/resource/Robert_Capa
  - The resting places

Exercise

Practise RDFLib

- Find a way to prune the resting places to the cemetery only (e.g. parse the RDF data from those URIs in the graph, check their classes, then select only the entity that belongs to the class identifying a cemetery)
- Retrieve all the people resting there (return URI and label)

RDFLib

Create triples

With RDFLib you can add new triples manually using the method **.add()**

To add a new triple you need to specify all 3 placeholders and their data types.

RDFLib

Delete triples

Likewise, you can remove triples using the method **.remove()**

```
1 g.add(( URIRef("http://dbpedia.org/resource/Robert_Capa"),
2         DBO.occupation,
3         URIRef("http://dbpedia.org/resource/Photographer") ))
4
5 g.remove(( URIRef("http://dbpedia.org/resource/Robert_Capa"),
6         DBO.occupation,
7         URIRef("http://dbpedia.org/resource/Photographer") ))
8
9 for s,p,o in g.triples(( None, DBO.occupation, None )):
10     print(s,p,o)
```

RDFLib

Serialize triples

The Graph() object is an in-memory graph, that is, once the code is run, the object is lost.

To preserve data in a graph we must serialize it and store RDF in a file.
RDFLib has a method that allows you to create a file and serialize data in any allowed format.

Exercise

Practise RDFLib

Create a list like the following one:

```
photos = [“http://dbpedia.org/resource/The_Falling_Soldier”,
“http://dbpedia.org/resource/The_Shaved_Woman_of_Chartres”,
“https://dbpedia.org/page/Sicilian_Peasant_Telling_an_American_Officer_Which_Way_the_Germans_Had_Gone”]
```

- For each photo in the list, add a triple to the graph using the property **DBO.creator**

.* to RDF

Transform anything to RDF

In the last decades, cultural heritage institutions and scholars have been transforming their data into RDF to facilitate data exchange and semantic interoperability.

The transformation can be performed automatically, e.g. using **XSLT** stylesheets to transform XML to RDF/XML files, **python** to transform multiple formats to RDF (e.g. XML, CSV, JSON). **Tools and mapping languages** exist for this task too (SPARQL Anything, RML, etc).

.* to RDF

.* to RDF

Transforming data in other formats to RDF requires to first work out the conceptual mapping between original data and expected output data:

- **Selection**. What gets transformed and what not
- **Normalisation**. What needs data cleaning (e.g. dates)
- **Granularity**. What gets transformed as-is or needs some refinement/extraction
- **Data type mapping**. What becomes an entity or a literal, and which data type
- **Ontology mapping**. Select classes and properties and map them to pieces of original data.
- **URI design**. Mint persistent unique URIs for the entities

.* to RDF

XML to RDF

To transform XML-like files into RDF we can use **XSLT** if we plan to transform them into RDF/XML files. Nonetheless, using python libraries can be a convenient way to manipulate data:

- Libraries like **lxml.etree** allow you to manipulate XML documents with more flexibility
- **RDFLib** allows you transform your data into any serialisation you need

Since not all XML files are the same, and the output RDF depends on the factors outlined in the previous slide, there is **no straightway transformation template** to be used.

.* to RDF

XML to RDF

We can explore a structured XML/EAC-CPF file describing George Orwell, from the project SNAC.

We **select** what gets transformed into RDF.

```
1 <nameEntry>
2   <part localType="Surname">Orwell</part>
3   <part localType="Forename">George</part>
4   <part localType="Date">1903-1950</part>
5   <authorizedForm>rda</authorizedForm>
6 </nameEntry>
7 ...
8 <existDates>
9   <dateRange>
10    <fromDate localType="http://socialarchive.iath.virginia.edu/control/term#Birth"
11              standardDate="1903-06-25">1903-06-25</fromDate>
12    <toDate localType="http://socialarchive.iath.virginia.edu/control/term#Death"
13            standardDate="1950-01-21">1950-01-21</toDate>
14  </dateRange>
15 </existDates>
16 ...
17 <biogHist>
18   <p>George Orwell (b. 25 June 1903, Motihari, India–d. 21 January 1950, London, England) is the pen
   name for British author Eric Arthur Blair. Orwell attended Eton College and he joined the Imperial
   police force taking a job in Burma (modern Myanmar). After returning to England, he settled in
   London and started writing and became a teacher. He is best known for novels 1984 and Animal Farm.
   </p>
19 </biogHist>
20 ...
21 <relations>
22   <cpfRelation xlink:type="simple"
23                xlink:role="http://socialarchive.iath.virginia.edu/control/term#Person"
24                xlink:arcrole="http://socialarchive.iath.virginia.edu/control/term#associatedWith"
25                xlink:href="http://n2t.net/ark:/99166/w6bp02mq">
26   <relationEntry>Blunden, Edmund, 1896-1974.</relationEntry>
27   </cpfRelation>
28     ...
29  </relations>
30
```

Name and lastname

Birth and death dates

Biography

People he knew

.* to RDF

XML to RDF

We look how **granular** the data is (e.g. name), and what **data types** to associate to pieces of information.

Dates of known people must be **normalised** (xsd:gYear > xsd:date)

```xml
<nameEntry>
  <part localType="Surname">Orwell</part>
  <part localType="Forename">George</part>
  <part localType="Date">1903-1950</part>
  <authorizedForm>rda</authorizedForm>
</nameEntry>
...
<existDates>
  <dateRange>
    <fromDate localType="http://socialarchive.iath.virginia.edu/control/term#Birth"
              standardDate="1903-06-25">1903-06-25</fromDate>
    <toDate localType="http://socialarchive.iath.virginia.edu/control/term#Death"
            standardDate="1950-01-21">1950-01-21</toDate>
  </dateRange>
</existDates>
...
<biogHist>
  <p>George Orwell (b. 25 June 1903, Motihari, India–d. 21 January 1950, London, England) is the pen name for British author Eric Arthur Blair. Orwell attended Eton College and he joined the Imperial police force taking a job in Burma (modern Myanmar). After returning to England, he settled in London and started writing and became a teacher. He is best known for novels 1984 and Animal Farm.
  </p>
</biogHist>
...
<relations>
  <cpfRelation xlink:type="simple"
               xlink:role="http://socialarchive.iath.virginia.edu/control/term#Person"
               xlink:arcrole="http://socialarchive.iath.virginia.edu/control/term#associatedWith"
               xlink:href="http://n2t.net/ark:/99166/w6bp02mq">
  <relationEntry>Blunden, Edmund, 1896-1974.</relationEntry>
  </cpfRelation>
    ...
</relations>
```

An entity with one string composed of two concatenated strings

xsd:date

A string

Entities with a string associated

We can also extract years/dates from the string using a simple rule (text after 2nd comma)

.* to RDF

XML to RDF

FOAF is an ontology born to describe people's social relations. FOAF lacks some properties which we can integrate from the DBpedia ontology.

NB. Check for inconsistencies when using multiple ontologies.

```xml
<nameEntry>
  <part localType="Surname">Orwell</part>
  <part localType="Forename">George</part>
  <part localType="Date">1903-1950</part>
  <authorizedForm>rda</authorizedForm>
</nameEntry>
...
<existDates>
  <dateRange>
    <fromDate localType="http://socialarchive.iath.virginia.edu/control/term#Birth"
              standardDate="1903-06-25">1903-06-25</fromDate>
    <toDate localType="http://socialarchive.iath.virginia.edu/control/term#Death"
            standardDate="1950-01-21">1950-01-21</toDate>
  </dateRange>
</existDates>
...
<biogHist>
  <p>George Orwell (b. 25 June 1903, Motihari, India–d. 21 January 1950, London, England) is the pen
  name for British author Eric Arthur Blair. Orwell attended Eton College and he joined the Imperial
  police force taking a job in Burma (modern Myanmar). After returning to England, he settled in
  London and started writing and became a teacher. He is best known for novels 1984 and Animal Farm.
  </p>
</biogHist>
...
<relations>
  <cpfRelation xlink:type="simple"
               xlink:role="http://socialarchive.iath.virginia.edu/control/term#Person"
               xlink:arcrole="http://socialarchive.iath.virginia.edu/control/term#associatedWith"
               xlink:href="http://n2t.net/ark:/99166/w6bp02mq">
  <relationEntry>Blunden, Edmund, 1896-1974.</relationEntry>
  </cpfRelation>
    ...
</relations>
```

Class ***foaf:Person***
Property ***foaf:name***

dbo:birthDate
dbo:deathDate

dbo:abstract

Property ***foaf:knows***

dbo:birthDate
dbo:deathDate

Exercise

Learn lxml.etree

Let's have a look at lxml library

- Parse XML
- Access root element
- Find children elements via slicing notation, find(), findall(), and XPath (notice limitations)
- Use of namespaces
- Get attributes (in dictionary), text value

Exercise

Practise etree

- Install lxml **`pip install lxml`**
- Download the XML/EAC-CPF file describing George Orwell (also 8_orwell.xml on our repo)
- Create a new python file and import the library import **`xml.etree.ElementTree as ET`**
- Parse the XML file, access the root element
- Use find() to access unique element, and findall to iterate over sequences of elements
- Pay attention to namespaces

Exercise

XML to RDF

Once you have extracted all the text elements you need

- Design opaque URIs (invent a URI base and use hashlib library to create a unique ID)
  - George Orwell
  - All people he knows
- Add triples to a graph using the predicates chosen
- Print the triples to see your graph growing
- Serialize the graph in a .ttl file

.* to RDF

JSON to RDF

JSON (JavaScript Object Notation) is a format for data exchange. Similar to XML, it allows to store structured information. Unlike XML, it has a more compact, less verbose, syntax.

It presents two data structures common to several programming languages:

- **Object** (or dictionary)
- **Array** (or list)

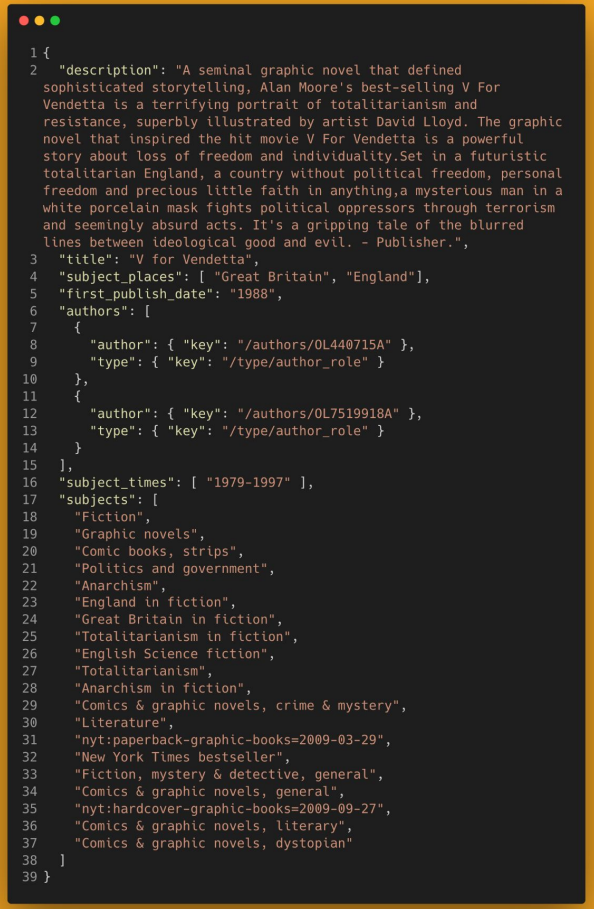
.* to RDF

JSON to RDF

To transform JSON to RDF, we can rely on:

- standard python libraries to manipulate JSON documents
- RDFLib to manipulate RDF data

Like with XML data, we need to perform a preliminary analysis on the data to be transformed. As an example we use a JSON file from the Open Library describing the book V for Vendetta.



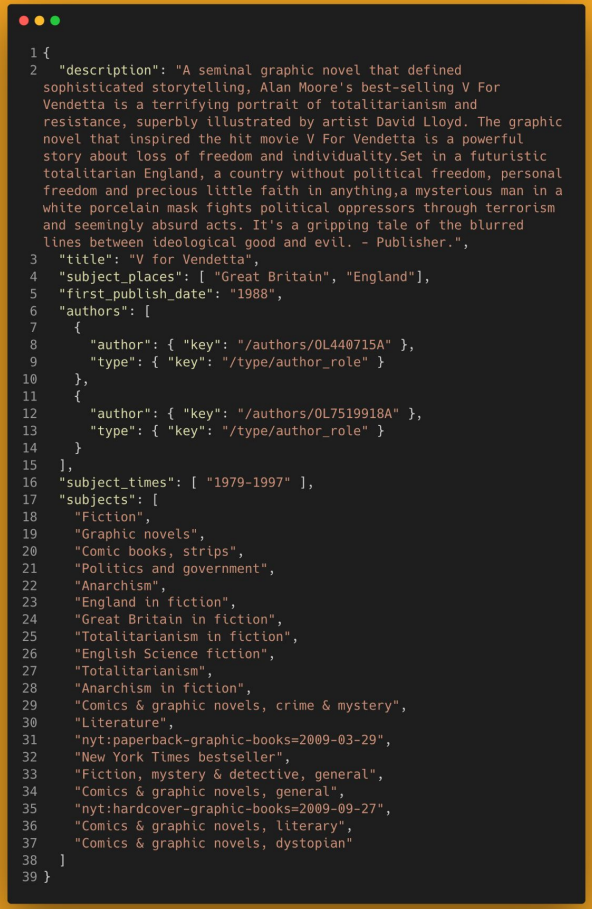
.* to RDF

JSON to RDF

We **select** what fields get transformed.

We identify **entities** and strings (and data types:

- **Entities**: A work, two people, two places, n subjects
- **Strings**: description (string), title (string), date (gYear)



.* to RDF

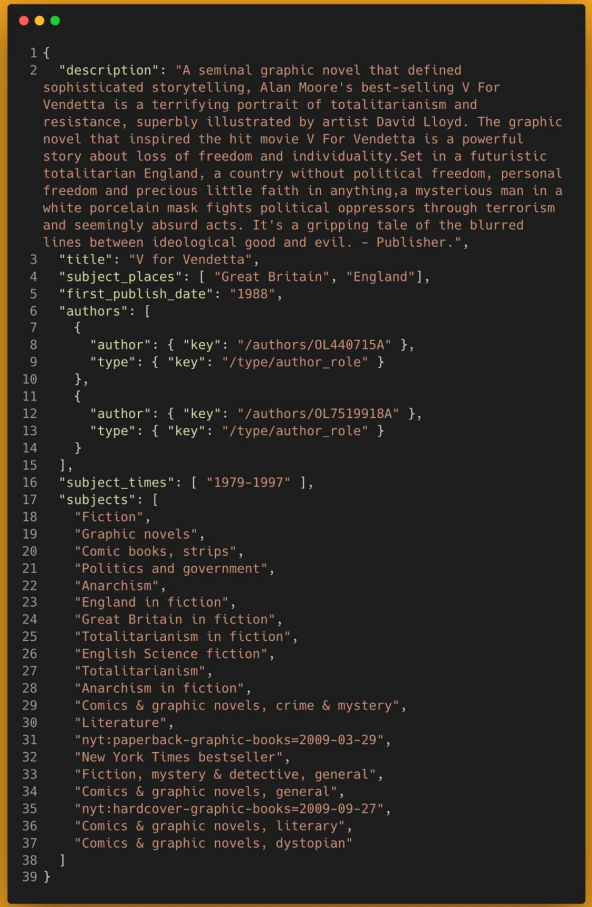
JSON to RDF

Interesting **relations** to be encoded in RDF are the following:

- Work > description, title, publication year, author, subject

We are going to use terms from the Schema.org ontology

- schema:CreativeWork > schema:abstract, schema:headline, schema:datePublished, schema:creator, schema:genre or schema:about
- schema:Person




```
{
  "description": "A seminal graphic novel that defined sophisticated storytelling, Alan Moore's best-selling V For Vendetta is a terrifying portrait of totalitarianism and resistance, superbly illustrated by artist David Lloyd. The graphic novel that inspired the hit movie V For Vendetta is a powerful story about loss of freedom and individuality.Set in a futuristic totalitarian England, a country without political freedom, personal freedom and precious little faith in anything,a mysterious man in a white porcelain mask fights political oppressors through terrorism and seemingly absurd acts. It's a gripping tale of the blurred lines between ideological good and evil. - Publisher.",
  "title": "V for Vendetta",
  "subject_places": [ "Great Britain", "England"],
  "first_publish_date": "1988",
  "authors": [
    {
      "author": { "key": "/authors/OL440715A" },
      "type": { "key": "/type/author_role" }
    },
    {
      "author": { "key": "/authors/OL7519918A" },
      "type": { "key": "/type/author_role" }
    }
  ],
  "subject_times": [ "1979-1997" ],
  "subjects": [
    "Fiction",
    "Graphic novels",
    "Comic books, strips",
    "Politics and government",
    "Anarchism",
    "England in fiction",
    "Great Britain in fiction",
    "Totalitarianism in fiction",
    "English Science fiction",
    "Totalitarianism",
    "Anarchism in fiction",
    "Comics & graphic novels, crime & mystery",
    "Literature",
    "nyt:paperback-graphic-books=2009-03-29",
    "New York Times bestseller",
    "Fiction, mystery & detective, general",
    "Comics & graphic novels, general",
    "nyt:hardcover-graphic-books=2009-09-27",
    "Comics & graphic novels, literary",
    "Comics & graphic novels, dystopian"
  ]
}
```

Exercise

Learn json for python

Let’s have a look at basic functions in the json library (tutorial)

- How to access JSON from file
- How to store JSON in a file

Exercise

JSON to RDF

- Download the JSON file
- Create a python file
- Read contents from the JSON file and transform it in a dictionary
- Access key/value pairs of the dictionary and extract contents to be transformed to RDF
- Add triples to a graph according to the ontology terms selected
- Serialize the graph in json-ld file

.* to RDF

CSV to RDF

CSV is the open format to store tabular data.

Every row of a table is represented by a line in the text file. Values of cells are separated by comma. See an example from the Tate Modern

```
id,name,gender,dates,yearOfBirth,yearOfDeath,placeOfBirth,placeOfDeath,url
10093,"Abakanowicz, Magdalena",Female,born 1930,1930,,Polska,,http://www.tate.org.uk/art/artists/magdalena-abakanowicz-10093
0,"Abbey, Edwin Austin",Male,1852–1911,1852,1911,"Philadelphia, United States","London, United
Kingdom",http://www.tate.org.uk/art/artists/edwin-austin-abbey-0
2756,"Abbott, Berenice",Female,1898–1991,1898,1991,"Springfield, United States","Monson, United
States",http://www.tate.org.uk/art/artists/berenice-abbott-2756
1,"Abbott, Lemuel Francis",Male,1760–1803,1760,1803,"Leicestershire, United Kingdom","London, United
Kingdom",http://www.tate.org.uk/art/artists/lemuel-francis-abbott-1
622,"Abrahams, Ivor",Male,born 1935,1935,,"Wigan, United Kingdom",,http://www.tate.org.uk/art/artists/ivor-abrahams-622
2606,Absalon,Male,1964–1993,1964,1993,"Tel Aviv-Yafo, Yisra'el","Paris, France",http://www.tate.org.uk/art/artists/absalon-2606
9550,"Abts, Tomma",Female,born 1967,1967,,"Kiel, Deutschland",,http://www.tate.org.uk/art/artists/tomma-abts-9550
623,"Acconci, Vito",Male,born 1940,1940,,"New York, United States",,http://www.tate.org.uk/art/artists/vito-acconci-623
624,"Ackling, Roger",Male,1947–2014,1947,2014,"Isleworth, United Kingdom",,http://www.tate.org.uk/art/artists/roger-ackling-624
625,"Ackroyd, Norman",Male,born 1938,1938,,"Leeds, United Kingdom",,http://www.tate.org.uk/art/artists/norman-ackroyd-625
2411,"Adam, Robert",Male,1728–1792,1728,1792,"Kirkcaldy, United Kingdom","London, United Kingdom",http://www.tate.org.uk/art/artists/robert-
adam-2411
```

1	id	name	gender	dates	yearOfBirth	yearOfDeath	placeOfBirth
2	10093	Abakanowicz, Magdalena	Female	born 1930	1930		Polska
3	0	Abbey, Edwin Austin	Male	1852–1911	1852	1911	Philadelphia, United States
4	2756	Abbott, Berenice	Female	1898–1991	1898	1991	Springfield, United States
5	1	Abbott, Lemuel Francis	Male	1760–1803	1760	1803	Leicestershire, United Kingdom
6	622	Abrahams, Ivor	Male	born 1935	1935		Wigan, United Kingdom
7	2606	Absalon	Male	1964–1993	1964	1993	Tel Aviv-Yafo, Yisra'el
8	9550	Abts, Tomma	Female	born 1967	1967		Kiel, Deutschland
9	623	Acconci, Vito	Male	born 1940	1940		New York, United States
10	624	Ackling, Roger	Male	1947–2014	1947	2014	Isleworth, United Kingdom
11	625	Ackroyd, Norman	Male	born 1938	1938		Leeds, United Kingdom

.* to RDF

CSV to RDF

```
id,name,gender,dates,yearOfBirth,yearOfDeath,placeOfBirth,placeOfDeath,url
10093,"Abakanowicz, Magdalena",Female,born 1930,1930,,Polska,,http://www.tate.org.uk/art/artists/magdalena-abakanowicz-10093
0,"Abbey, Edwin Austin",Male,1852–1911,1852,1911,"Philadelphia, United States","London, United
Kingdom",http://www.tate.org.uk/art/artists/edwin-austin-abbey-0
2756,"Abbott, Berenice",Female,1898–1991,1898,1991,"Springfield, United States","Monson, United
States",http://www.tate.org.uk/art/artists/berenice-abbott-2756
1,"Abbott, Lemuel Francis",Male,1760–1803,1760,1803,"Leicestershire, United Kingdom","London, United
Kingdom",http://www.tate.org.uk/art/artists/lemuel-francis-abbott-1
622,"Abrahams, Ivor",Male,born 1935,1935,,"Wigan, United Kingdom",,http://www.tate.org.uk/art/artists/ivor-abrahams-622
2606,Absalon,Male,1964–1993,1964,1993,"Tel Aviv-Yafo, Yisra'el","Paris, France",http://www.tate.org.uk/art/artists/absalon-2606
9550,"Abts, Tomma",Female,born 1967,1967,,"Kiel, Deutschland",,http://www.tate.org.uk/art/artists/tomma-abts-9550
623,"Acconci, Vito",Male,born 1940,1940,,"New York, United States",,http://www.tate.org.uk/art/artists/vito-acconci-623
624,"Ackling, Roger",Male,1947–2014,1947,2014,"Isleworth, United Kingdom",,http://www.tate.org.uk/art/artists/roger-ackling-624
625,"Ackroyd, Norman",Male,born 1938,1938,,"Leeds, United Kingdom",,http://www.tate.org.uk/art/artists/norman-ackroyd-625
2411,"Adam, Robert",Male,1728–1792,1728,1792,"Kirkcaldy, United Kingdom","London, United Kingdom",http://www.tate.org.uk/art/artists/robert-
adam-2411
```

Again, to transform CSV to RDF, we can rely on:

- standard python libraries to manipulate CSV documents
- RDFLib to manipulate RDF data

1	id	name	gender	dates	yearOfBirth	yearOfDeath	placeOfBirth
2	10093	Abakanowicz, Magdalena	Female	born 1930	1930		Polska
3	0	Abbey, Edwin Austin	Male	1852–1911	1852	1911	Philadelphia, United States
4	2756	Abbott, Berenice	Female	1898–1991	1898	1991	Springfield, United States
5	1	Abbott, Lemuel Francis	Male	1760–1803	1760	1803	Leicestershire, United Kingdom
6	622	Abrahams, Ivor	Male	born 1935	1935		Wigan, United Kingdom
7	2606	Absalon	Male	1964–1993	1964	1993	Tel Aviv-Yafo, Yisra'el
8	9550	Abts, Tomma	Female	born 1967	1967		Kiel, Deutschland
9	623	Acconci, Vito	Male	born 1940	1940		New York, United States
10	624	Ackling, Roger	Male	1947–2014	1947	2014	Isleworth, United Kingdom
11	625	Ackroyd, Norman	Male	born 1938	1938		Leeds, United Kingdom

.* to RDF

CSV to RDF

```
id,name,gender,dates,yearOfBirth,yearOfDeath,placeOfBirth,placeOfDeath,url
10093,"Abakanowicz, Magdalena",Female,born 1930,1930,,Polska,,http://www.tate.org.uk/art/artists/magdalena-abakanowicz-10093
0,"Abbey, Edwin Austin",Male,1852–1911,1852,1911,"Philadelphia, United States","London, United
Kingdom",http://www.tate.org.uk/art/artists/edwin-austin-abbey-0
2756,"Abbott, Berenice",Female,1898–1991,1898,1991,"Springfield, United States","Monson, United
States",http://www.tate.org.uk/art/artists/berenice-abbott-2756
1,"Abbott, Lemuel Francis",Male,1760–1803,1760,1803,"Leicestershire, United Kingdom","London, United
Kingdom",http://www.tate.org.uk/art/artists/lemuel-francis-abbott-1
622,"Abrahams, Ivor",Male,born 1935,1935,,"Wigan, United Kingdom",,http://www.tate.org.uk/art/artists/ivor-abrahams-622
2606,Absalon,Male,1964–1993,1964,1993,"Tel Aviv–Yafo, Yisra'el","Paris, France",http://www.tate.org.uk/art/artists/absalon-2606
9550,"Abts, Tomma",Female,born 1967,1967,,"Kiel, Deutschland",,http://www.tate.org.uk/art/artists/tomma-abts-9550
623,"Acconci, Vito",Male,born 1940,1940,,"New York, United States",,http://www.tate.org.uk/art/artists/vito-acconci-623
624,"Ackling, Roger",Male,1947–2014,1947,2014,"Isleworth, United Kingdom",,http://www.tate.org.uk/art/artists/roger-ackling-624
625,"Ackroyd, Norman",Male,born 1938,1938,,"Leeds, United Kingdom",,http://www.tate.org.uk/art/artists/norman-ackroyd-625
2411,"Adam, Robert",Male,1728–1792,1728,1792,"Kirkcaldy, United Kingdom","London, United Kingdom",http://www.tate.org.uk/art/artists/robert-
adam-2411
```

We **select** what fields get transformed. We identify **entities** and strings (and data types:

- **Entities**: a person, two places, gender
- **Strings**: birth/death year (gYear), url (string)

1	id	name	gender	dates	yearOfBirth	yearOfDeath	placeOfBirth
2	10093	Abakanowicz, Magdalena	Female	born 1930	1930		Polska
3	0	Abbey, Edwin Austin	Male	1852–1911	1852	1911	Philadelphia, United States
4	2756	Abbott, Berenice	Female	1898–1991	1898	1991	Springfield, United States
5	1	Abbott, Lemuel Francis	Male	1760–1803	1760	1803	Leicestershire, United Kingdom
6	622	Abrahams, Ivor	Male	born 1935	1935		Wigan, United Kingdom
7	2606	Absalon	Male	1964–1993	1964	1993	Tel Aviv–Yafo, Yisra'el
8	9550	Abts, Tomma	Female	born 1967	1967		Kiel, Deutschland
9	623	Acconci, Vito	Male	born 1940	1940		New York, United States
10	624	Ackling, Roger	Male	1947–2014	1947	2014	Isleworth, United Kingdom
11	625	Ackroyd, Norman	Male	born 1938	1938		Leeds, United Kingdom

.* to RDF

CSV to RDF

```
id,name,gender,dates,yearOfBirth,yearOfDeath,placeOfBirth,placeOfDeath,url
10093,"Abakanowicz, Magdalena",Female,born 1930,1930,,Polska,,http://www.tate.org.uk/art/artists/magdalena-abakanowicz-10093
0,"Abbey, Edwin Austin",Male,1852–1911,1852,1911,"Philadelphia, United States","London, United
Kingdom",http://www.tate.org.uk/art/artists/edwin-austin-abbey-0
2756,"Abbott, Berenice",Female,1898–1991,1898,1991,"Springfield, United States","Monson, United
States",http://www.tate.org.uk/art/artists/berenice-abbott-2756
1,"Abbott, Lemuel Francis",Male,1760–1803,1760,1803,"Leicestershire, United Kingdom","London, United
Kingdom",http://www.tate.org.uk/art/artists/lemuel-francis-abbott-1
622,"Abrahams, Ivor",Male,born 1935,1935,,"Wigan, United Kingdom",,http://www.tate.org.uk/art/artists/ivor-abrahams-622
2606,Absalon,Male,1964–1993,1964,1993,"Tel Aviv-Yafo, Yisra'el","Paris, France",http://www.tate.org.uk/art/artists/absalon-2606
9550,"Abts, Tomma",Female,born 1967,1967,,"Kiel, Deutschland",,http://www.tate.org.uk/art/artists/tomma-abts-9550
623,"Acconci, Vito",Male,born 1940,1940,,"New York, United States",,http://www.tate.org.uk/art/artists/vito-acconci-623
624,"Ackling, Roger",Male,1947–2014,1947,2014,"Isleworth, United Kingdom",,http://www.tate.org.uk/art/artists/roger-ackling-624
625,"Ackroyd, Norman",Male,born 1938,1938,,"Leeds, United Kingdom",,http://www.tate.org.uk/art/artists/norman-ackroyd-625
2411,"Adam, Robert",Male,1728–1792,1728,1792,"Kirkcaldy, United Kingdom","London, United Kingdom",http://www.tate.org.uk/art/artists/robert-
adam-2411
```

We are going to use yet another ontology, i.e. **Wikidata** (see an example record of an artist).

There are two namespaces, respectively for classes and properties

wd: <http://www.wikidata.org/entity/>
wdt: <http://www.wikidata.org/prop/direct/>

	id	name	gender	dates	yearOfBirth	yearOfDeath	placeOfBirth
2	10093	Abakanowicz, Magdalena	Female	born 1930	1930		Polska
3	0	Abbey, Edwin Austin	Male	1852–1911	1852	1911	Philadelphia, United States
4	2756	Abbott, Berenice	Female	1898–1991	1898	1991	Springfield, United States
5	1	Abbott, Lemuel Francis	Male	1760–1803	1760	1803	Leicestershire, United Kingdom
6	622	Abrahams, Ivor	Male	born 1935	1935		Wigan, United Kingdom
7	2606	Absalon	Male	1964–1993	1964	1993	Tel Aviv-Yafo, Yisra'el
8	9550	Abts, Tomma	Female	born 1967	1967		Kiel, Deutschland
9	623	Acconci, Vito	Male	born 1940	1940		New York, United States
10	624	Ackling, Roger	Male	1947–2014	1947	2014	Isleworth, United Kingdom
11	625	Ackroyd, Norman	Male	born 1938	1938		Leeds, United Kingdom

Exercise | Homework

Which Wikidata properties?

Have a look at this artist and make a list of all the properties that fit our case. Notice also the property “Tate ID”.

Exercise | Homework

Learn CSV in python

Let’s have a look at basic functions in the json library (tutorial)

- How to access CSV from file
- Read the CSV as a list of lists or as a list of dictionaries
- How to store CSV in a file

Exercise | Homework

Learn CSV in python

- Download the csv file from Tate Modern repository (or from our repository)
- Create a python file and read the CSV content as a dictionary
- Access row contents and create URIs and triples on spot.
    - Notice not all cells have data. You must check for contents of cells before adding triples
- Write triples in a .ttl file