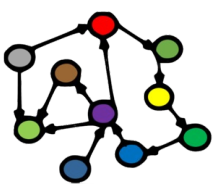
Welcome to INFO216:
Knowledge Graphs
Spring 2023

Andreas L Opdahl
<Andreas.Opdahl@uib.no>

About me

- Background:
  - siv ing, dr ing from NTNU
  - University of Bergen (since early 1990-ies)
  - part-time programmer / consulting for industry
  - several “Forskningsråd” and EU projects and networks
- Central research interest:
  - modelling of information systems and enterprises
  - semantic modelling and modelling languages
  - semantic technologies: ontologies and knowledge graphs
  - knowledge graphs for the news



Recent project: BDEM

- Leveraging *Big Data for Emergency Management*
  - how can semantic technologies play a part?
  - developed a new Master course: INFO319 on Big Data

SAN DIEGO STATE UNIVERSITY

VESTLANDSFORSKING

Recent project: UBIMOB

Ubiquitous Data-Driven Urban Mobility

WESTERN NORWAY RESEARCH INSTITUTE
VESTLANDSFORSKING

NTNU
Norwegian University of Science and Technology

UNIVERSITAS BERGENSIS

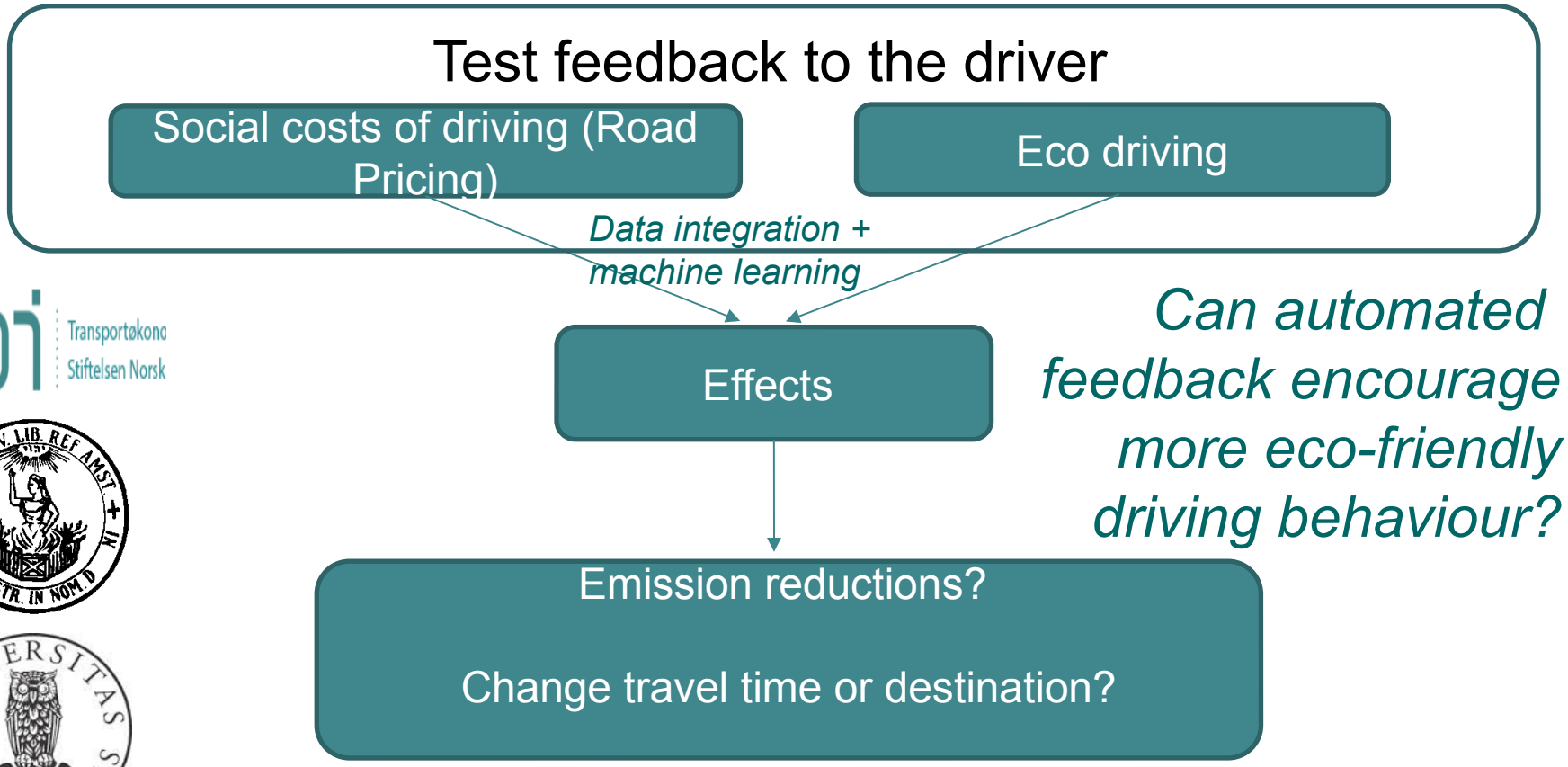
toi
Transportøkonomisk institutt
Stiftelsen Norsk senter for samferdselsforskning

Forschungszentrum · Research Center
L3S

The University Of Sheffield.

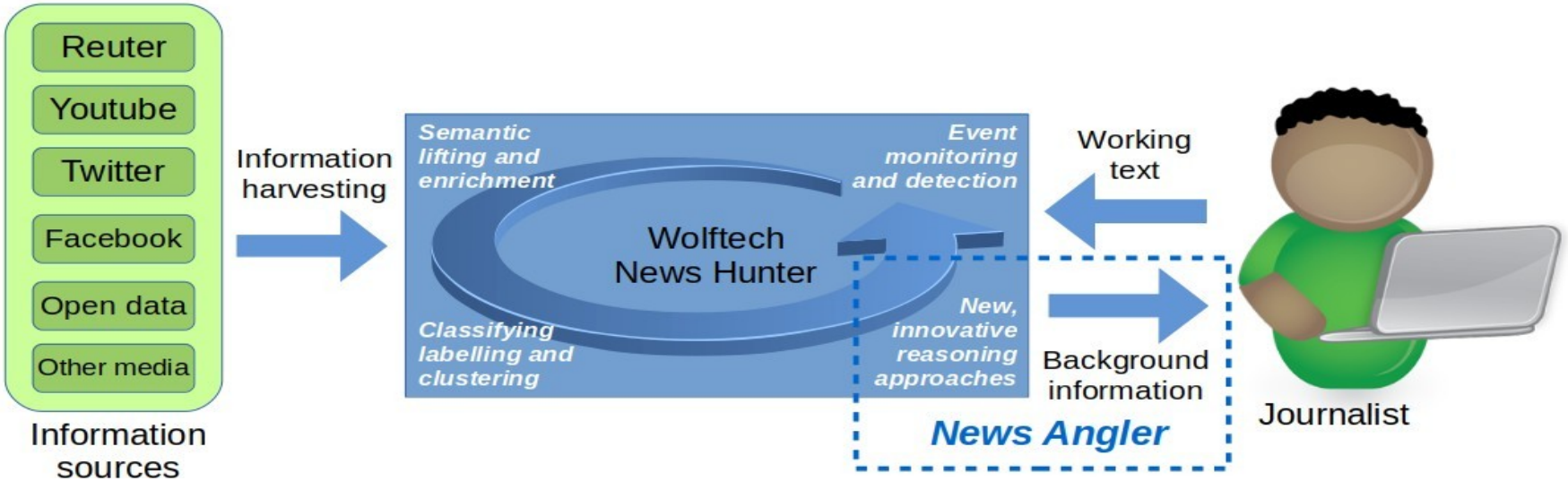
telenor

Recent project: Transfeed



Can automated feedback encourage more eco-friendly driving behaviour?

Ongoing project: News Angler

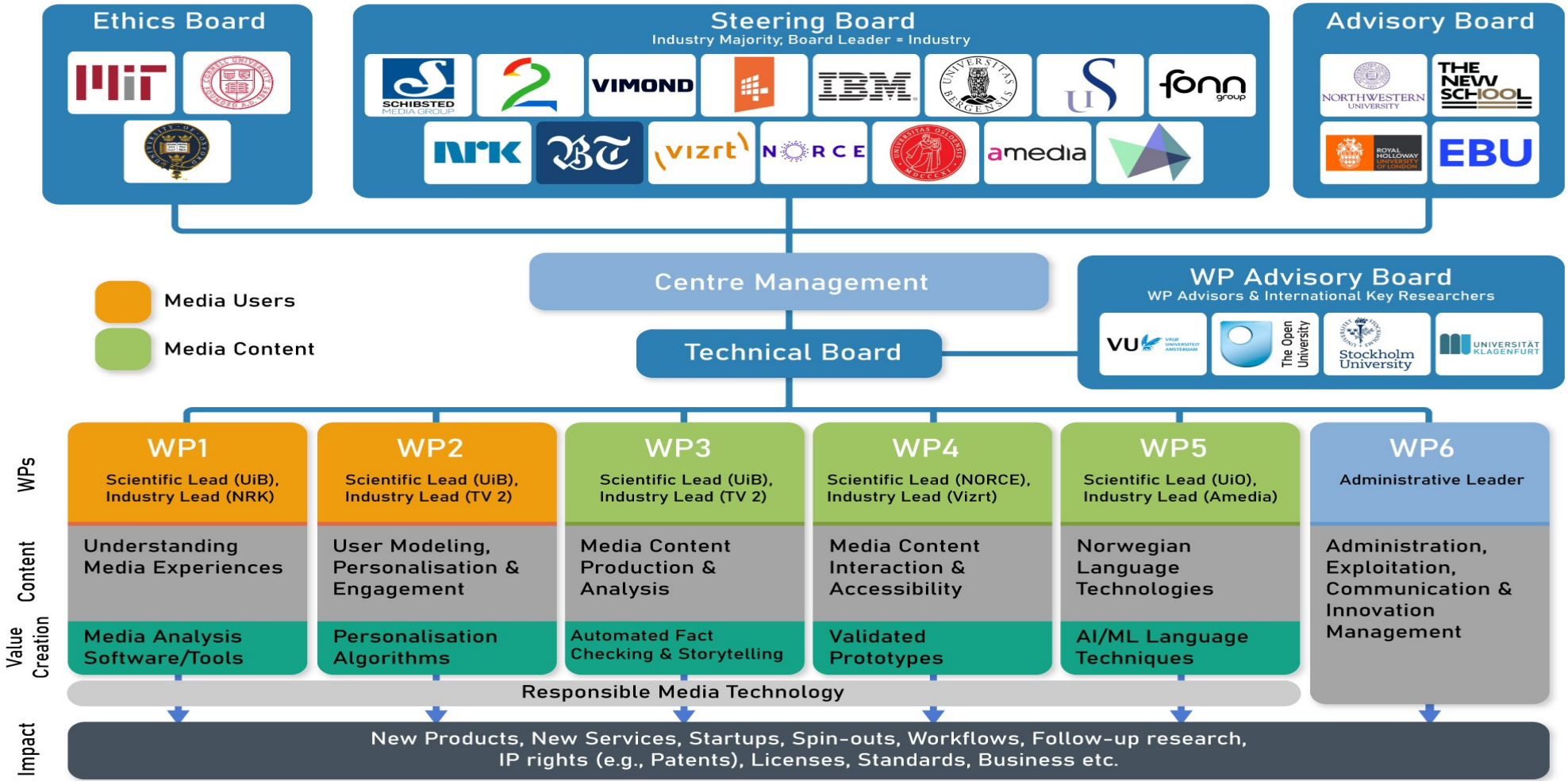


"Wolftech News supports and improves the workflows in a newsroom through mobile solutions for field work that are integrated with central systems for news monitoring, resource management, news editing, and multi-platform publishing"

1) Harvesting and analysing messages
2) Growing a semantic news graph
   - concepts, named entities, context…
3) Analysing working texts (stories)
4) Identifying background information
5) Prioritising and preparing
6) Journalistic and editorial preferences

Research: graph, searches, preparation, preferences, language, scaling

Active centre: Media Futures

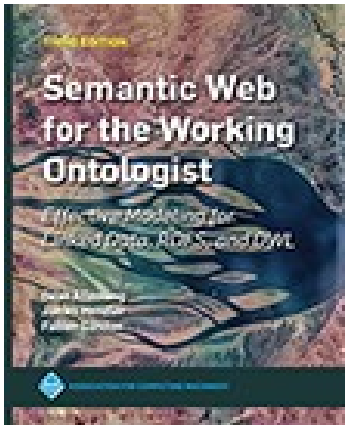


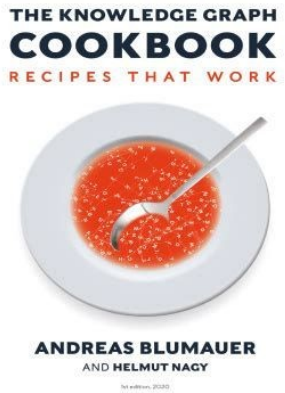
Session 1: Introduction to Knowledge Graphs

- Themes:
  - *what are knowledge graphs (KGs)?*
    - *what is the problem?*
    - *who uses them?*
    - examples of important *open KGs*
  - *exercise 1: Getting started...*
  - *about INFO216*
    - organisation of the course
    - practical information

Readings

- Sources:
  - Allemang, Hendler & Gandon (2020): Semantic Web for the Working Ontologist, 3rd edition: chapters 1-2
  - Blumauer & Nagy (2020): Knowledge Graph Cookbook – Recipes that Work: pages 27-55, 105-122 (*supplementary)*
- Materials in the wiki <http://wiki.uib.no/info216>:
  - Tim Berners-Lee talks about the semantic web
  - links to a few important open KGs





What are knowledge graphs?

What is the problem?

- *Lots* of data around
  - internal, public, social, open
- *Enormous potential to solve, simplify, speed up many critical information handling problem*
  - but data are mostly not *linked* (think of a world wide web without document links!)
- *What if linkable data could always be linked automatically?*
- Tim Berners-Lee's *semantic web vision* (ca 2000)

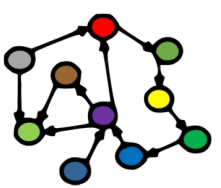
Tim Berners-Lee
Inventor of the
World Wide Web
(WWW, 1989)

Tim Berners-Lee: <http://www.youtube.com/watch?v=HeUrEh-nqtU>

What is the problem?

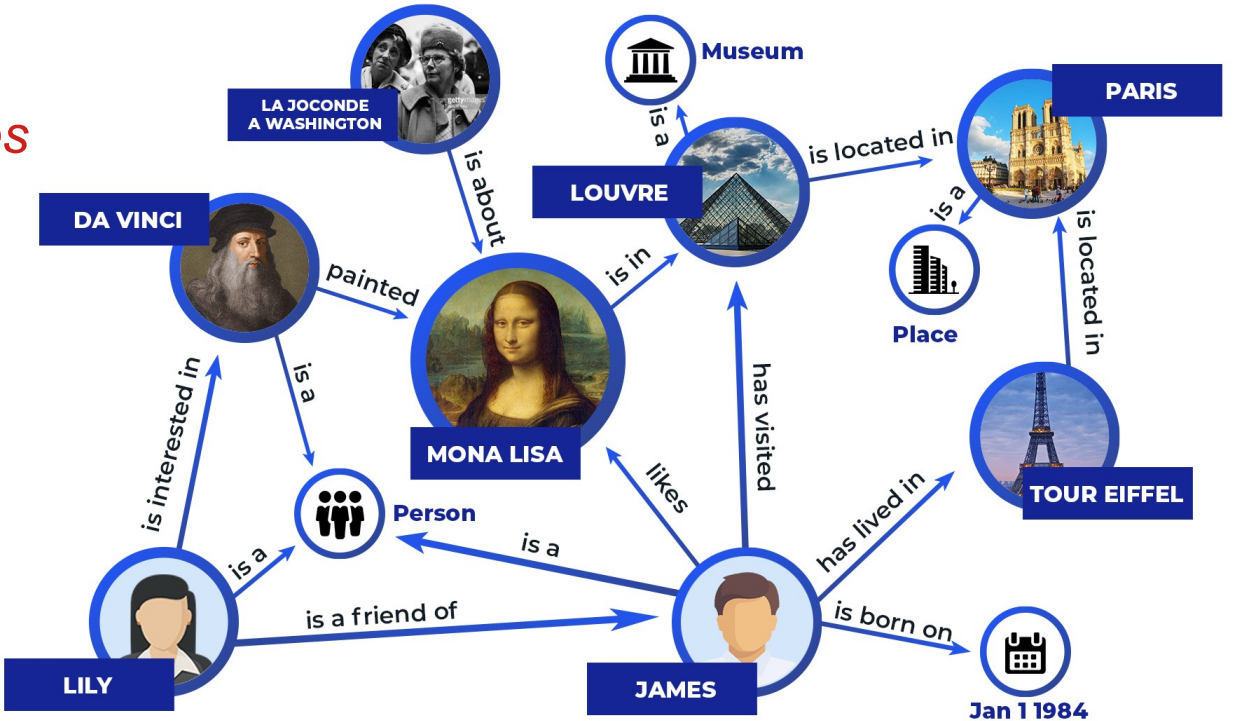
- *Lots* of data around
  - internal, public, social, open
- *Enormous potential to solve, simplify, speed up many critical information handling problem*
  - but data are mostly not *linked* (think of a world wide web without document links!)
- *What if linkable data could always be linked automatically?*
- The *semantic web vision* (ca 2000)

- Need *standard ways of representing data and knowledge*:
  - technical
    - standard formats, languages, and techniques to *share data*
  - semantic
    - standard identifiers and terms to *share meaning*
  - formal
    - support formal *rules and reasoning*



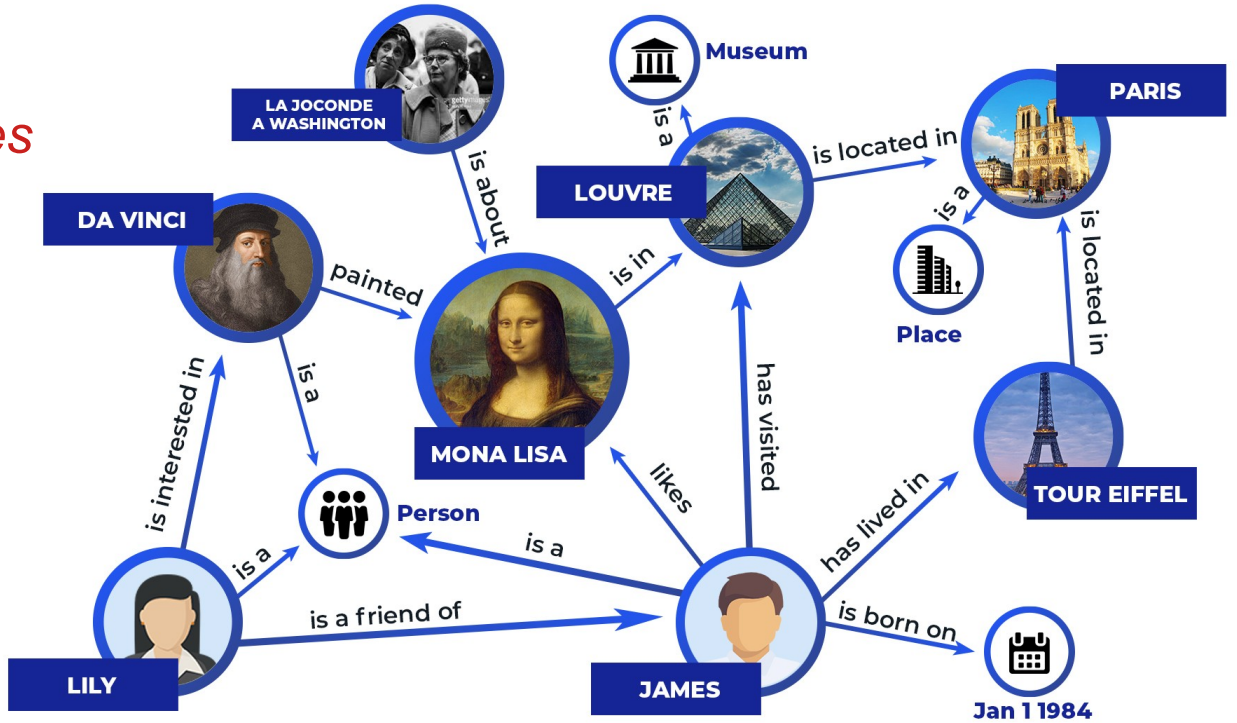
Knowledge graph

- A *graph* of *nodes*
  connected by directed *edges*



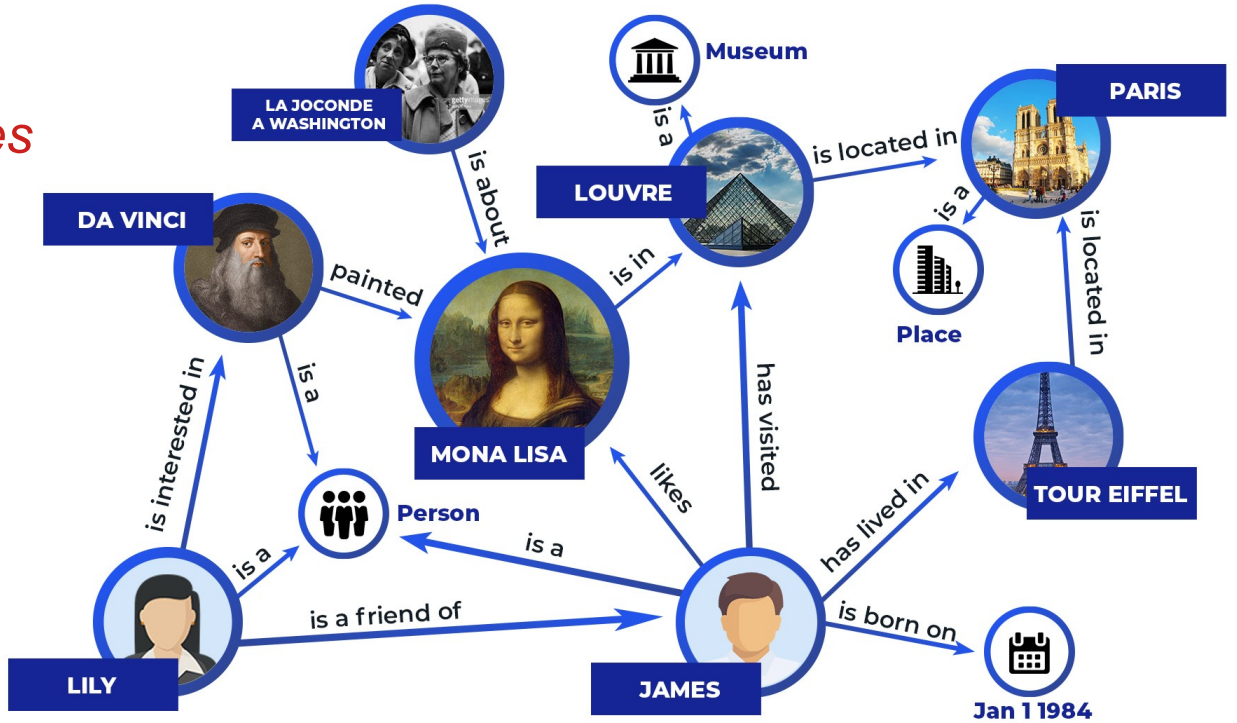
Knowledge graph

- A *graph* of *nodes* connected by directed *edges*
- Nodes can represent *resources* or *values*



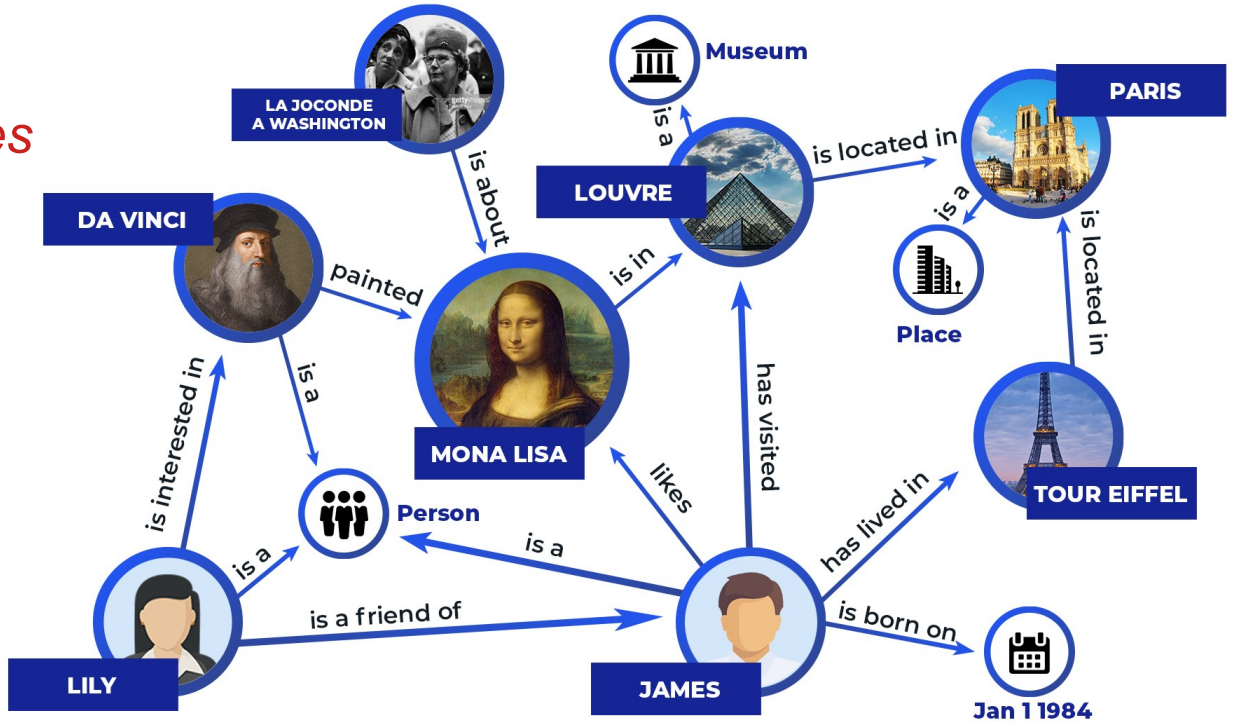
Knowledge graph

- A *graph* of *nodes* connected by directed *edges*
- Nodes can represent *resources* or *values*
- Edges represent *relations*



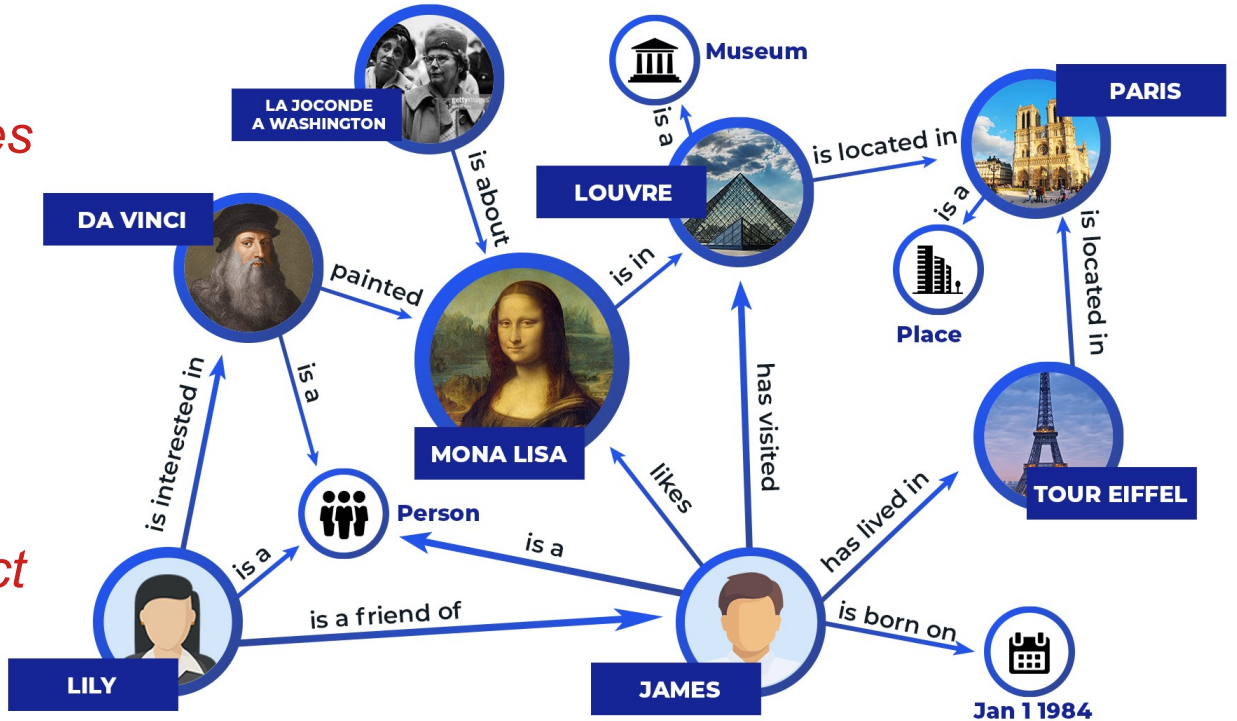
Knowledge graph

- A *graph* of *nodes* connected by directed *edges*
- Nodes can represent *resources* or *values*
- Edges represent *relations*
- Each node–edge–node *triple* represents a *fact*



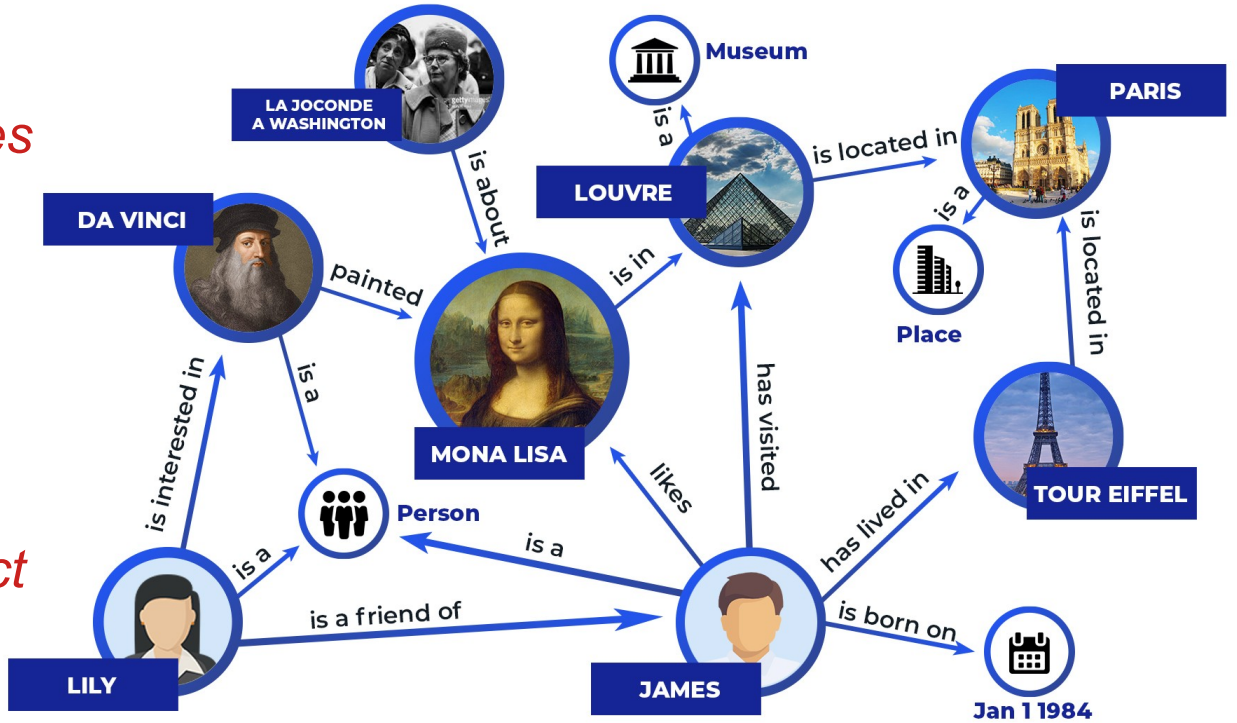
Knowledge graph

- A *graph* of *nodes* connected by directed *edges*
- Nodes can represent *resources* or *values*
- Edges represent *relations*
- Each node–edge–node *triple* represents a *fact*
  - *subject–predicate–object*



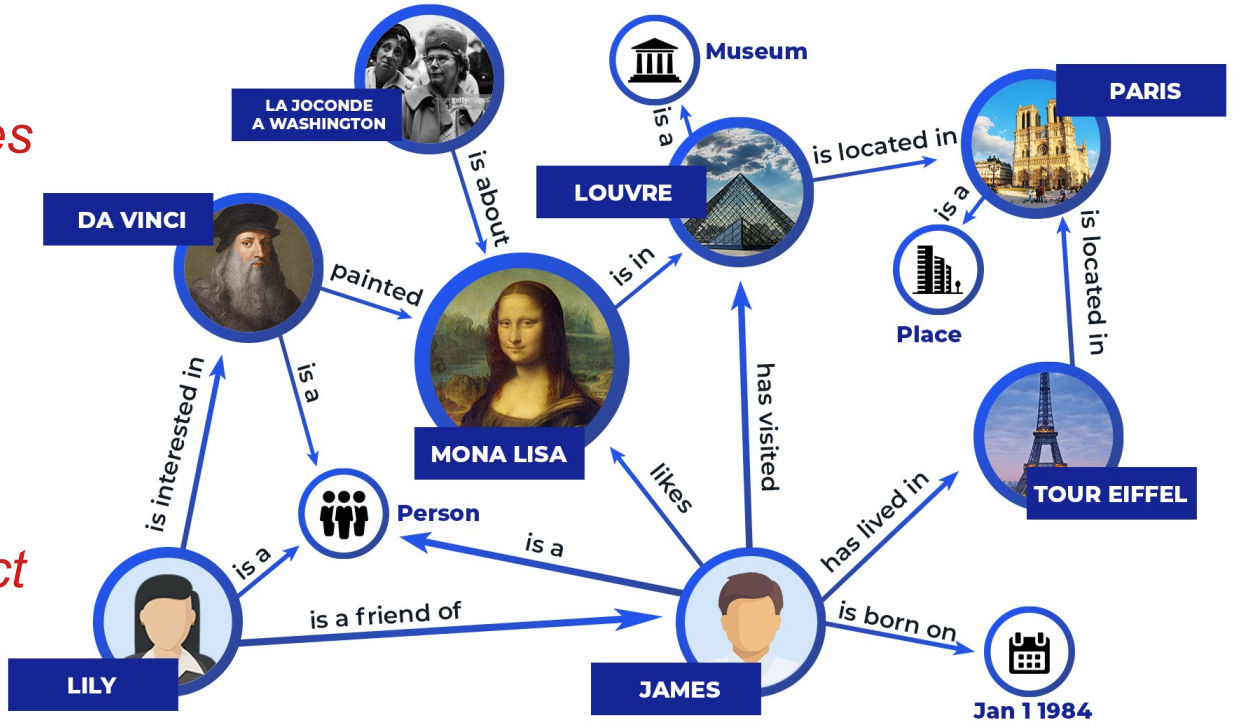
Knowledge graph

- A *graph* of *nodes* connected by directed *edges*
- Nodes can represent *resources* or *values*
- Edges represent *relations*
- Each node–edge–node *triple* represents a *fact*
  - *subject–predicate–object*
  - *head–relation–tail*

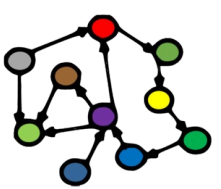


Knowledge graph

- A *graph* of *nodes* connected by directed *edges*
- Nodes can represent *resources* or *values*
- Edges represent *relations*
- Each node–edge–node *triple* represents a *fact*
  - *subject–predicate–object*
  - *head–relation–tail*
- A *knowledge graph* represents *knowledge* as *triples* connected by *nodes*

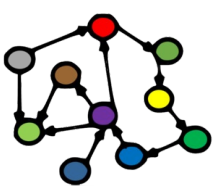


And there is more...



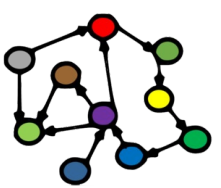
And there is more...

- Technical:
  - standard *formats* for storing and exchanging graphs
    - including types of values (strings, numbers, times, dates, etc.)
  - specialised *databases* and standard *query languages*
  - *APIs* for all major programming languages



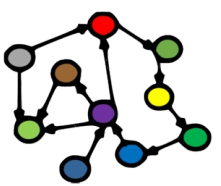
And there is more...

- Technical:
  - standard *formats* for storing and exchanging graphs
    - including types of values (strings, numbers, times, dates, etc.)
  - specialised *databases* and standard *query languages*
  - *APIs* for all major programming languages
- Semantic:
  - large repositories of *unique identifiers* for individual resources
  - *vocabularies* with unique identifiers for resource types and relations
  - *graph embeddings* and *graph neural networks* for machine learning



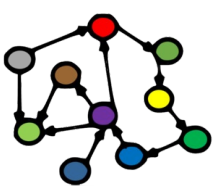
And there is more...

- Technical:
  - standard *formats* for storing and exchanging graphs
    - including types of values (strings, numbers, times, dates, etc.)
  - specialised *databases* and standard *query languages*
  - *APIs* for all major programming languages
- Semantic:
  - large repositories of *unique identifiers* for individual resources
  - *vocabularies* with unique identifiers for resource types and relations
  - *graph embeddings* and *graph neural networks* for machine learning
- Formal:
  - rule languages and *inference engines*
  - *formal logic* systems and reasoning engines



Why knowledge graphs?

- Ease of *exchanging, reusing* information
  - inherent semantics become clearer
  - less dependency on context
- Ease of *interlinking, enriching* information
  - semantic data can be combined in new ways
  - open reference datasets
  - general and specialised knowledge bases
- Ease of *extending*
  - no pre-defined data schemas (“schema-on-read”)
  - easy to add new types of resources and new relations
- *Well-matched with the needs of big data and machine learning!*



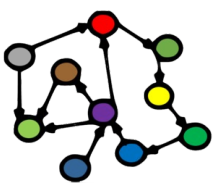
Who is using this?

- *All the big players!*
- Google's Knowledge Graph
- Microsoft's Satori
- Amazon's Product Graph
- ...and (almost) everyone else

From Frank van Harmelen's keynote at CAiSE 2018

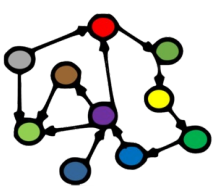
Knowledge graphs at Amazon (→S06)

- Let shoppers find the best products that fit their needs
  - allow greater variation in search terms
  - allow complex queries
- Ambition: *to structure all of the world's information as it relates to everything available on Amazon*
- Describe every product on Amazon
  - both products and non-products
  - both concrete and abstract concepts
  - link related entities, both internal and external
- Enhanced customer experience
  - visit Amazon to see what's new or interesting
  - discover ways to simplify and enrich their lives



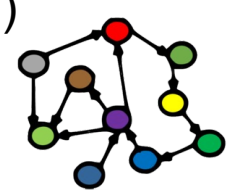
Knowledge graphs everywhere

- BBC's content management, ontologies, BBC Things
- Google, Bing, Yahoo… (schema.org) (2011)
- Google's Knowledge Graph (2012), Microsoft's Satori
- Facebook's Open Graph and Graph Search (2013)
- Thomson Reuters, Bloomberg...
- Uber Eats' food graph

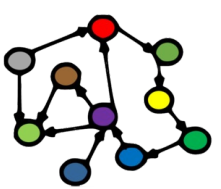


Some knowledge graphs we will look at in INFO216

- You have already seen Google's KG many times:
  - the "knowledge panels" in search results
- Wikidata (https://www.wikidata.org/)
  - part of the Wikimedia family, feeds factual information to Wikipedia
- DBpedia (https://www.dbpedia.org, https://dbpedia.org/page/Bergen)
  - extracts factual information from Wikipedia
- GeoNames (https://www.geonames.org/)
  - global database of place names (toponyms), relations and types
- BabelNet (https://babelnet.org/)
  - a multi-lingual dictionary and thesaurus
- Linked Open Vocabularies (LOV, https://lov.linkeddata.es/dataset/lov/)
  - a collection of knowledge graphs that describe vocabularies (also called ontologies) for other knowledge graphs

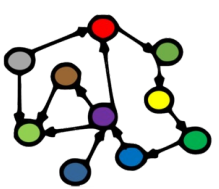


Exercise 1:
Getting started with VSCode, Python and RDFlib



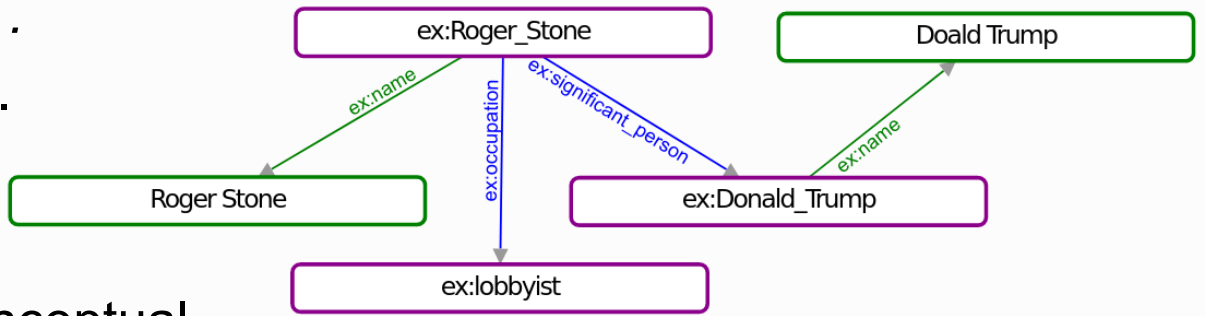
How can we represent semantic KGs?

- Resource Description Framework (RDF → S02)
- RDF models (KGs) consist of statements (triples)
  - of *subject predicate object* .
  - or *subject predicate literal* .



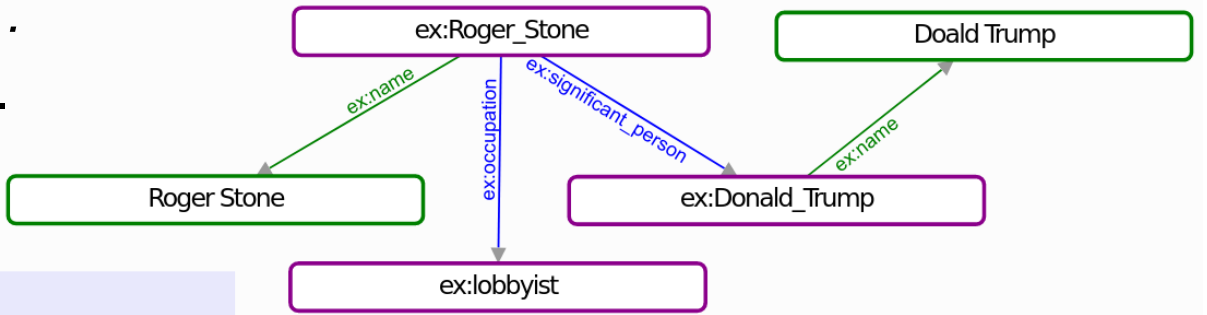
How can we represent semantic KGs?

- Resource Description Framework (RDF → S02)
- RDF models (KGs) consist of statements (triples)
  - of *subject predicate object* .
  - or *subject predicate literal* .
- The subject:
  - must be a *resource*
  - physical, informational, conceptual...
- The predicate:
  - must be a *property* (subtype of *resource*)
- The object:
  - is either a *resource*
  - or a *literal* (or a *value*: string, number... – *not* a *resource*)



How can we represent semantic KGs?

- Resource Description Framework (RDF → S02)
- RDF models (KGs) consist of statements (triples)
  - of *subject predicate object* .
  - or *subject predicate literal* .
- Serialisations, e.g., in *Turtle*:

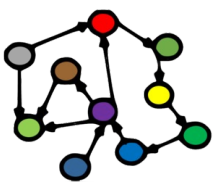



```
@prefix ex: <http://example.org/> .

ex:Roger_Stone
    ex:name "Roger Stone" ;
    ex:occupation ex:lobbyist ;
    ex:significant_person ex:Donald_Trump .

ex:Donald_Trump
    ex:name "Doald Trump" .
```

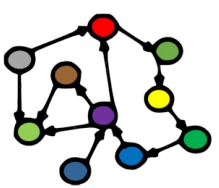
Uniform Resource Identifiers (URIs) identify resources, including types and relations



RDFLib graphs

- RDFLib:
  - an API for programming RDF and SPARQL in Python
  - simple, powerful and *pythonic*
  - parsers and serialisers for most RDF formats
  - a *Graph* interface:
    - a graph holds an RDF model
    - is a Python collection (set) of triples
    - supports adding, removing, listing, and searching for triples
    - supports writing to and reading from RDF files

```
>>> from rdflib import Graph
>>> model = Graph()
```



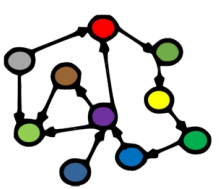
RDFLib resources

- URIRef:
  - a node with a URI (represents resources, types, and properties)

```
>>> from rdflib import URIRef
>>> donaldTrump = URIRef('http://example.org/Donald_Trump')
```

- Namespace:
  - a more compact way to create resources, types, and properties

```
>>> from rdflib import Namespace
>>> ex = Namespace('http://example.org/')
>>> donaldTrump = ex.Donald_Trump
```



RDFLib triples

- Triples / statements:
  - ordinary 3-item Python tuples
    - immutable sequences

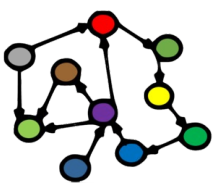
```
>>> triple = (s, p, o)   # creates a triple
>>> s[0]                 # returns the subject, etc.
```

  - add/remove triples to/from graph

```
>>> g.add( (res1, prop, res2) )      or:  >>> g.add( (res, prop, lit) )
>>> g.remove( (res1, prop, res2) )   or:  >>> g.remove( (res, prop, lit) )
```

  - close persisted model:

```
>>> g.close()
```



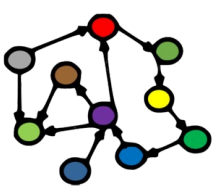
RDFLib literals

- Literal:
  - a typed value

```
>>> from rdflib import Literal
>>> lit = Literal('President of the United States')
```

  - strings can be language-tagged

```
>>> lit = Literal('President of the United States', 'en')
>>> lit = Literal(' 美利堅合眾國的國家元首、政府首腦兼三軍統帥'
                                                    , 'zh-ch')
```



Serialising and parsing

- Serialising:

  >>> g.serialize(destination=*fileNameStr,* format='ttl')

  >>> ttl_str = g.serialize(format='ttl').decode()

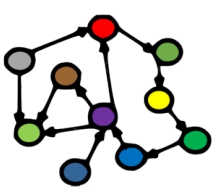
  >>> ttl_str = g.serialize(format='json-ld').decode()

- Parsing:

  >>> g.parse(location=*fileNameStr,* format='ttl')

  >>> g.parse(source=*webURLStr*, format='ttl')

  >>> g.parse(data=*pythonStr,* format='ttl')

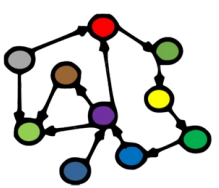


Listing statements

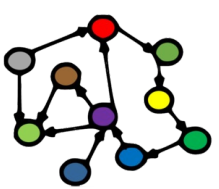
- Retrieving statements (triples):

```
>>> for triple in g:
>>>         do_something(triple)  # s = triple[0], p = triple[1], o = triple[2]
>>> for s, p, o in g:
>>>         do_something(s, p, o)
>>> for s, p, o in g.triples( (sub, pred, obj) ):
>>>         do_something(s, p, o)  # sub, pred, obj can be None
>>> for s, p, o in g[ sub : pred : obj ]:
>>>         do_something(s, p, o)  # sub, pred, obj can be empty:
                                   # s, p, o must match
```

Python overloading!

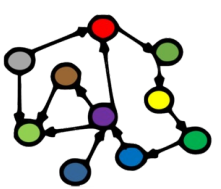


About INFO216



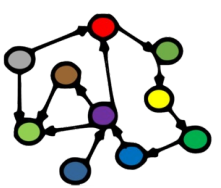
Purpose

- To learn theories, techniques, tools, and best practices for managing knowledge graphs.
- To acquire understanding and skills for programming applications that use and produce such data and metadata.
- To learn about existing sources of and standards for big, open, and semantic data.
- To gain practical experience in developing knowledge graph-based applications using technologies such as RDF, RDFS, OWL, SPARQL, and JSON-LD.



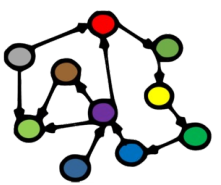
Curriculum

- Course book (*the whole book is mandatory*):
  - Allemang, Hendler & Gandon (2020).
    Semantic Web for the Working Ontologist,
    Effective Modeling for Linked Data, RDFS and OWL (Third Edition)
- Supplementary course book (*suggested, not mandatory*):
  - Blumauer & Nagy (2020).
    The Knowledge Graph Cookbook - Recipes that Work
- Additional readings (both *mandatory* and *suggested*) will be made available in the course wiki: https://wiki.uib.no/info216
- The lectures and lectures notes are also *mandatory* parts of the curriculum.



Practical

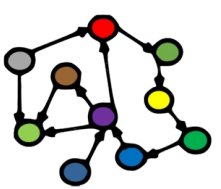
- 14 lectures:
  - Tuesdays 1215-1400
- 14 lab weeks:
  - 2 hours of weekly lab groups
  - starting this week, no labs week 10 and 14 (Easter)
  - seminar/lab leader: Robin Johansen Bøe <Robin.Boe@student.uib.no>
- Evaluation:
  - individual, written 4-hour exam
- Requirements:
  - participation in 75% of labs
- Course wiki:
  - http://wiki.uib.no/info216



Lecture plan (tentative)

1. Introduction to KGs
2. Representing KGs (RDF)
3. Querying and updating KGs (SPARQL)
4. Open KGs 1
5. Open KGs 2
6. Enterprise KGs
7. Rules (RDFS)
8. Ontologies (OWL)
9. Vocabularies
10. Reasoning about KGs (DL)
11. Formal ontologies (OWL-DL)
12. KG embeddings 1
13. KG embeddings 2
14. Knowledge engineering

You learn KGs best through practice: do the lab exercises thoroughly!



Next week:
Representing KGs
(more about RDF)