University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Architecture of large projects in bioinformatics (ADP)

Lecture 01

Łukasz P. Kozłowski

Warsaw, 2024

lukaskoz@mimuw.edu.pl

UNIVERSITY OF WARSAW
MIM
FACULTY
OF MATHEMATICS, INFORMATICS
AND MECHANICS

UNIWERSYTET WARSZAWSKI

Architecture of large projects in bioinformatics (ADP)

Lecture 01

https://www.mimuw.edu.pl/~lukaskoz/teaching/adp/

Łukasz P. Kozłowski

Warsaw, 2024

lukaskoz@mimuw.edu.pl

Lectures

1) Every Tuesday 14:15-16:00

2) The slides from the lectures will be provided at:

https://www.mimuw.edu.pl/~lukaskoz/teaching/adp/

or

https://moodle.mimuw.edu.pl/course/view.php?id=2110

Key: **pV2C6>.9**

During the course, laboratories it is assumed that you use **Ubuntu/Debian**

(no help for Windows, Mac users will be provided)

Outline

1. Data formats in bioinformatics.

2. Popular software libraries (BioPerl, BioPython).

3. Most important bioinformatics databases (UniProt, PDB, RefSeq, GenBank, ENA, InterPro, etc.)

4. Software licensing for scientific purposes. Free-software licensing. Patents.

5. Generic model Organism database (GMOD) project - assumptions, history and usage.

6. Genome browsers, problem description and state of the solutions.

Outline

7. Version control systems (CVS, SVN, git), and online collaboration ad distribution platforms (github, sourceforge).

8. Software testing, automated testing frameworks.

9. Scientific workflow systems - taverna and galaxy. MyExperiment platform. Reproducible research.

10. Literate programming idea and sweave, markdown, software documentation.

11. Interactive scripting platforms, Rstudio, Jupyter.

12. High-performance computing, HPC

Rules, grading, etc.

The final score (grade) for the course depends on:

1) **Homeworks – 25%**
2) **Essay** - **25%**
3) **Project – 50%**

The homework: after some laboratories there will be homework that need to be sent before next unit

The essay: *mini-review about specific bioinformatics topic*

*Before the student will be allowed to pass **the project** (s)he need to get 60% from **the homeworks and essay**. During last two-three laboratories students will present their topic in front of the class.*

***The project**: given the subject the student(s) will need to prepare the presentation explaining the scope and the work done. The project is about making some work with selected the software.*

The will be no checking of the presence during lectures

Exemplary subjects

Review about available software for:

– Structural biology (proteins, RNA, drugs)
– Phylogenetics
– Chemoinformatics
– Data warehouse in bioinformatics (e.g. Biomart)
– Genomics (e.g. chip-seq)
– Machine learning (clustering, classification, deep learning, etc.)
– Image processing from microscopes/scanners etc.
– own suggestions ...?

You have 1 week to decide/find/book the subject. At the next laboratory, you should make suggestions about the topic and then you will have 1 month to prepare it.

Homeworks

1) Frequently, there will homework after laboratory (it will be either the solutions from the laboratory itself or some extra exercise(s) similar to the one done during laboratories).

2) In order to pass given laboratory you need to send email with the solutions to lukaskoz@mimuw.edu.pl

Homeworks

3) The email with homework need to have specific structure:

a) The subject: ADP24_labN_hw_Surname_Name
e.g. ADP24_lab1_hw_Kozlowski_Lukasz

Note underscores, lack of special letters and the order of the parts

b) no text body in the email

c) single **standard** attachment with the same name compressed with 7z (**do not use** external services for the attachments like dropbox, google drive, aspera, usos, etc.)

e.g. ADP24_lab1_hw_Kozlowski_Lukasz.7z

The content and the structure of the attachment is laboratory specific and it will be explained separately

Homeworks

Deadline

For the homework there is only one (for all weeks) and it is
Monday 23:59 CET (GMT+1 Winter and GMT+2 Summer)

Each week delay will be a awarded by handicap/offset of -7.5% of the score (and remember that you need on average 60% to pass)

Rules, grading, etc.

- optimize the size of files
- do not use special letters in file names e.g. Polish
- do not use Polish (everything should be in English)
- always add legends and descriptions for the plots
- README
- LICENSE
- etc.

follow the golden rule:

one functionality, one (python) script

Thus:

- do not overuse jupyter

- do not overuse any web browser solutions follow the golden rule:

one functionality, one (python) script

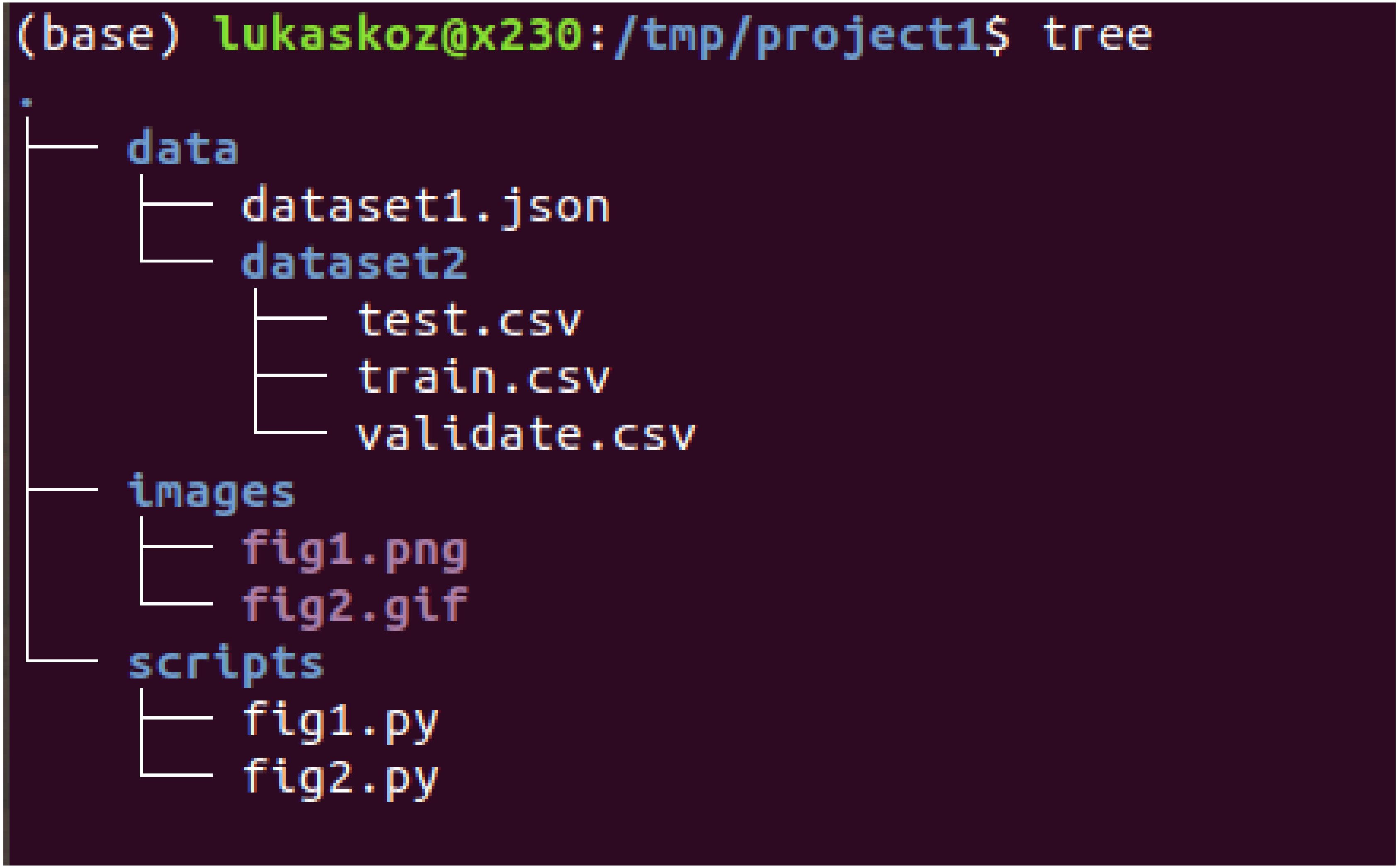
Thus:

- do not overuse jupyter

- do not overuse any web browser solutions

- make proper project structure

Be prepared to present everything as static file
(preferably pdf)



```
(base) lukaskoz@x230:/tmp/project1$ tree
.
├── data
│   ├── dataset1.json
│   └── dataset2
│       ├── test.csv
│       ├── train.csv
│       └── validate.csv
├── images
│   ├── fig1.png
│   └── fig2.gif
└── scripts
    ├── fig1.py
    └── fig2.py
```

- make proper project structure

Be prepared to present everything as static file
(preferably pdf)

Python

Guido van Rossum

Python was conceived in the late 1980s

https://en.wikiversity.org/wiki/Python_Concepts

https://en.wikiversity.org/wiki/Python_Programming

https://pl.wikibooks.org/wiki/Zanurkuj_w_Pythonie

Coursera

LinkedIn Learning

Datacamp, Inc.

edX

Udacity

Python

Guido van Rossum

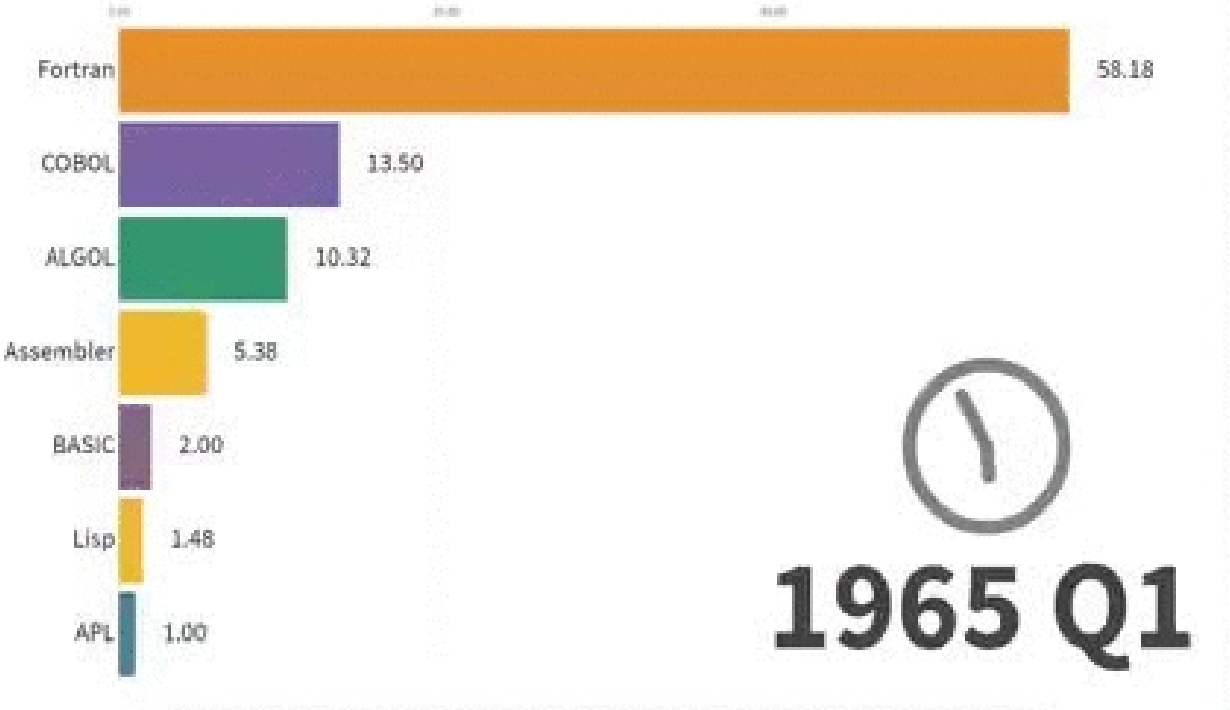
Python was conceived in the late 1980s

- scripting language (no compilation)

- uses whitespace indentation, rather

 than curly brackets or keywords,

 to delimit blocks



- we use **python3** instead **python2**

https://www.youtube.com/watch?v=Og847HVwRSI

Programming language

This is not python programming course and it means that you know basic programming or you must learn the basics within short time

Programming language

This is not python programming course and it means that you know basic programming or you must learn the basics within short time

1. Dive Into Python 3 (also in Polish, check on wikibooks)

2. Python Data Analysis, Ivan Idris, 2014

3. Python for Data Analysis, Wes MacKinney, 2013

Programming language

This is not python programming course and it means that you know basic programming or you must learn the basics within short time

We will use a number of general python libraries useful for bioinformatics such as:

`jupyter, numpy, scipy, pandas, matplotlib, sckit-learn, seaborn, and many others`

Additionally, we will use domain specific libraries e.g. **`biopython`**

Data processing

We will use (and compare) different formats such as general ones (csv, xml, json, etc.) and domain specific

This means that you will need to write custom parsers* as well.

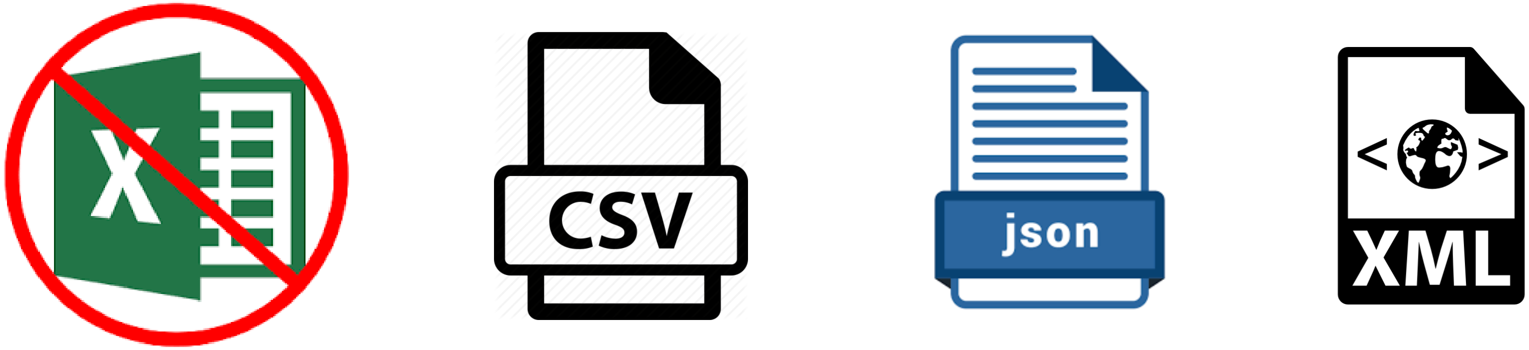
We will use (and compare) different formats such as general ones (csv, xml, json, etc.) and domain specific

This means that you will need to write custom parsers* as well.

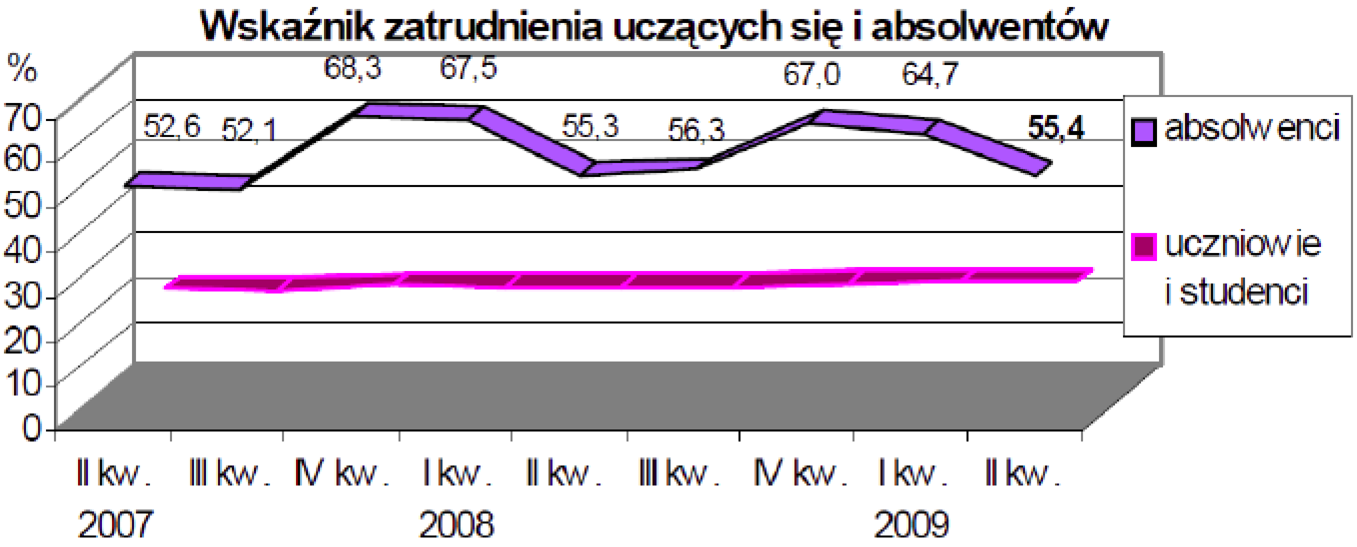
* during labs frequently you will be asked to implement simple functionalities even if some libraries are available

DATA FORMATS

Data formats

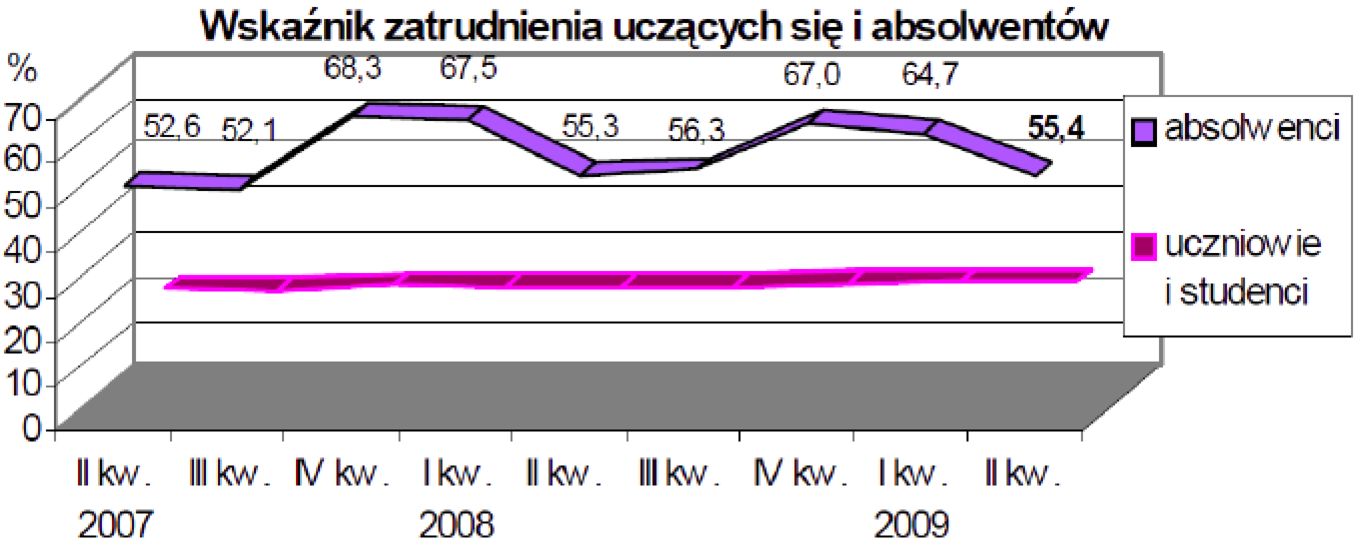


Data formats

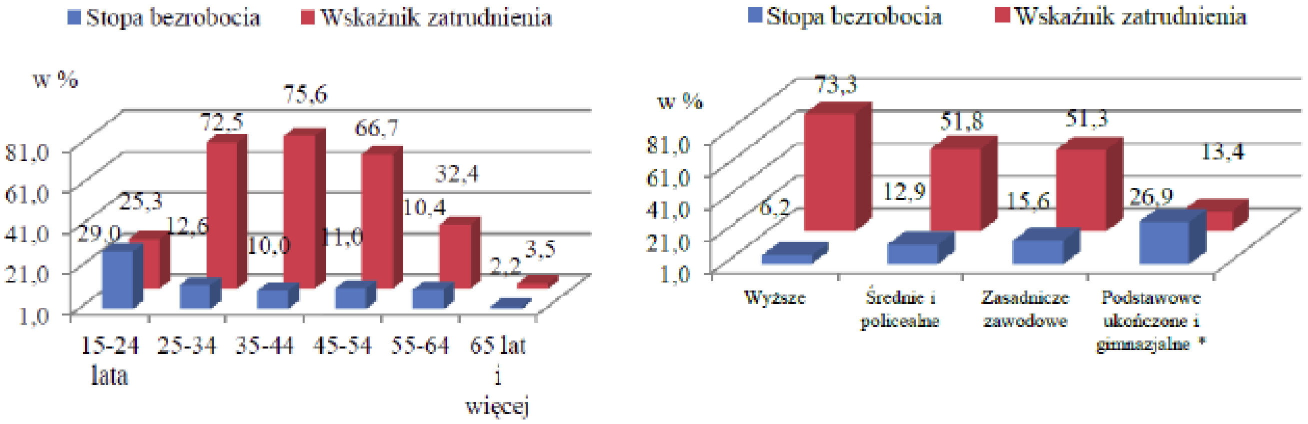


Data formats

ADP


Wskaźnik zatrudnienia uczących się i absolwentów

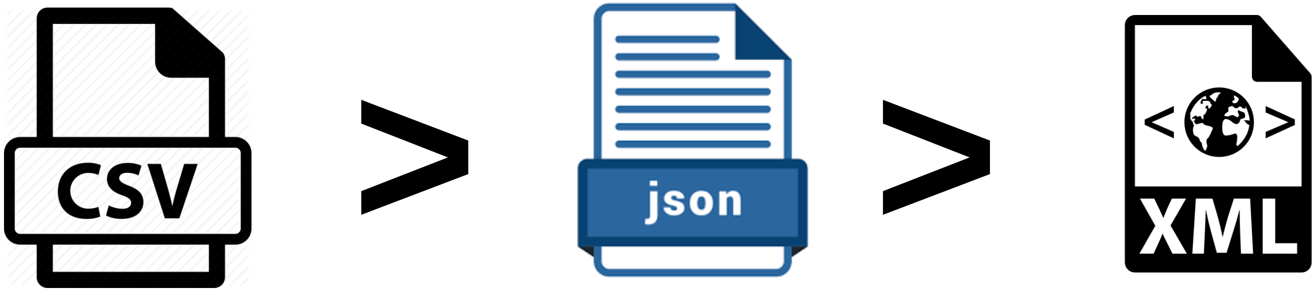
Wykres 7. Wskaźnik zatrudnienia oraz stopa bezrobocia według grup wieku i poziomu wykształcenia w 2011 r.



* łącznie z wykształceniem podstawowym nieukończonym i bez wykształcenia szkolnego

Data formats

Data formats

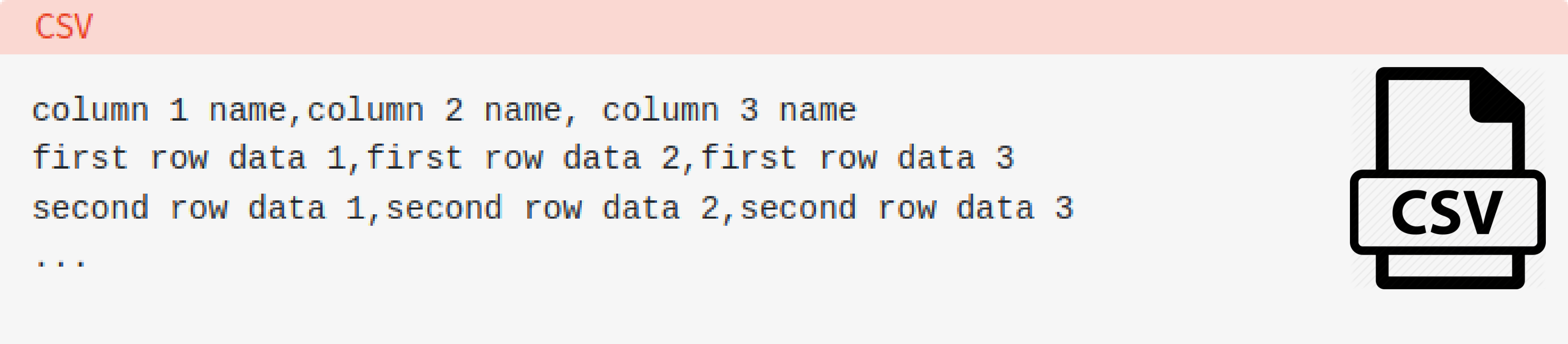


Data formats

CSV > json > XML

Data formats

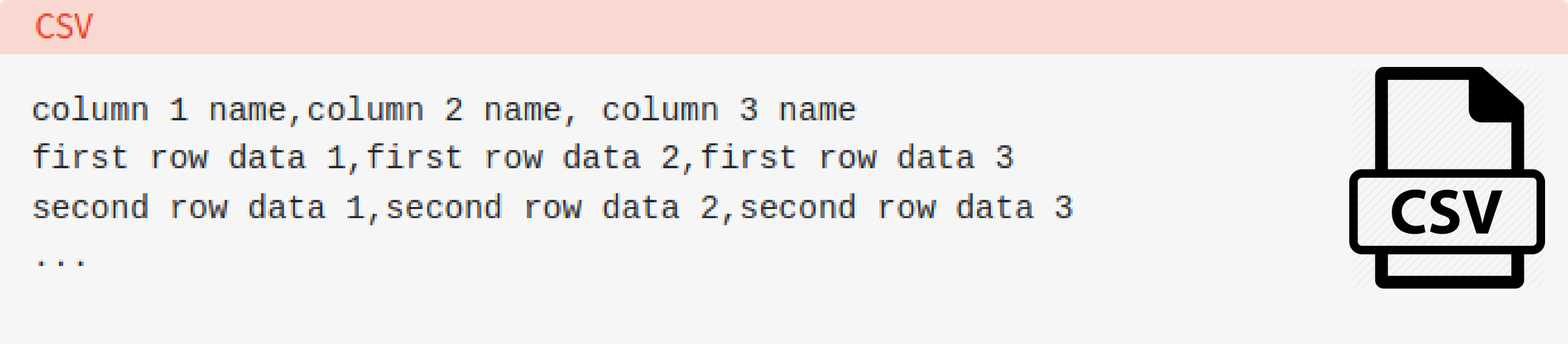
Data formats



CSV

```
column 1 name,column 2 name, column 3 name
first row data 1,first row data 2,first row data 3
second row data 1,second row data 2,second row data 3
...
```

Data formats

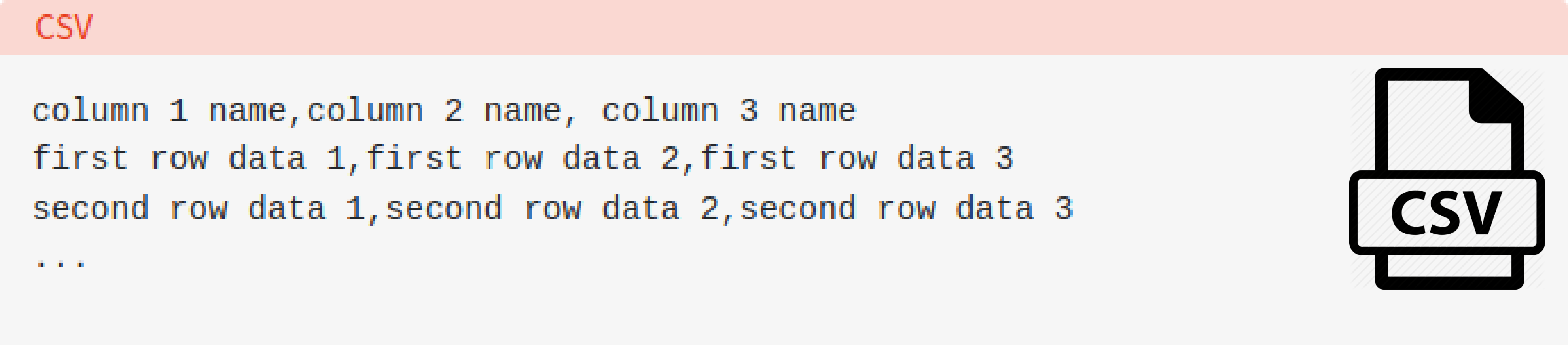


CSV

```
column 1 name,column 2 name, column 3 name
first row data 1,first row data 2,first row data 3
second row data 1,second row data 2,second row data 3
...
```

printable ASCII or Unicode characters.

Data formats

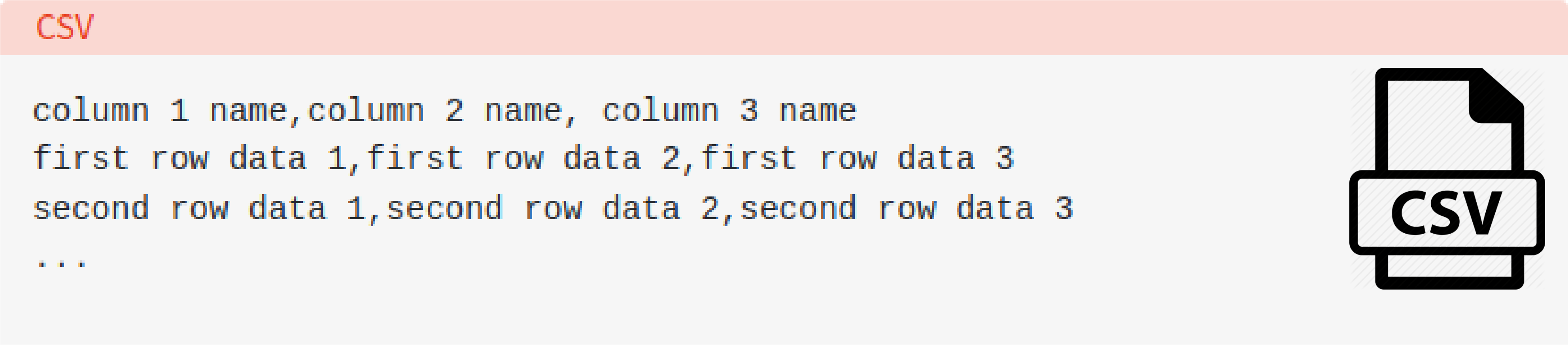


CSV

```
column 1 name,column 2 name, column 3 name
first row data 1,first row data 2,first row data 3
second row data 1,second row data 2,second row data 3
...
```

printable ASCII or Unicode characters.

Other popular delimiters include the tab (\t), colon (:) and semi-colon (;) characters. Properly parsing a CSV file requires us to know which delimiter is being used.



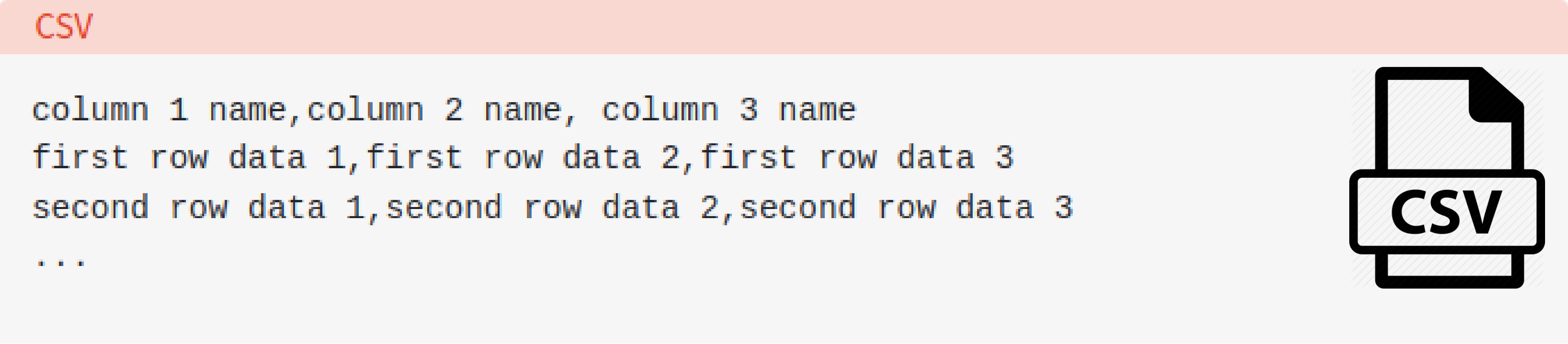
CSV

```
column 1 name,column 2 name, column 3 name
first row data 1,first row data 2,first row data 3
second row data 1,second row data 2,second row data 3
...
```

printable ASCII or Unicode characters.

Other popular delimiters include the tab (\t), colon (:) and semi-colon (;) characters. Properly parsing a CSV file requires us to know which delimiter is being used.

CSV files are very easy to work with programmatically. Any language that supports text file input and string manipulation (like Python) can work with CSV files directly.



CSV

```
column 1 name,column 2 name, column 3 name
first row data 1,first row data 2,first row data 3
second row data 1,second row data 2,second row data 3
...
```

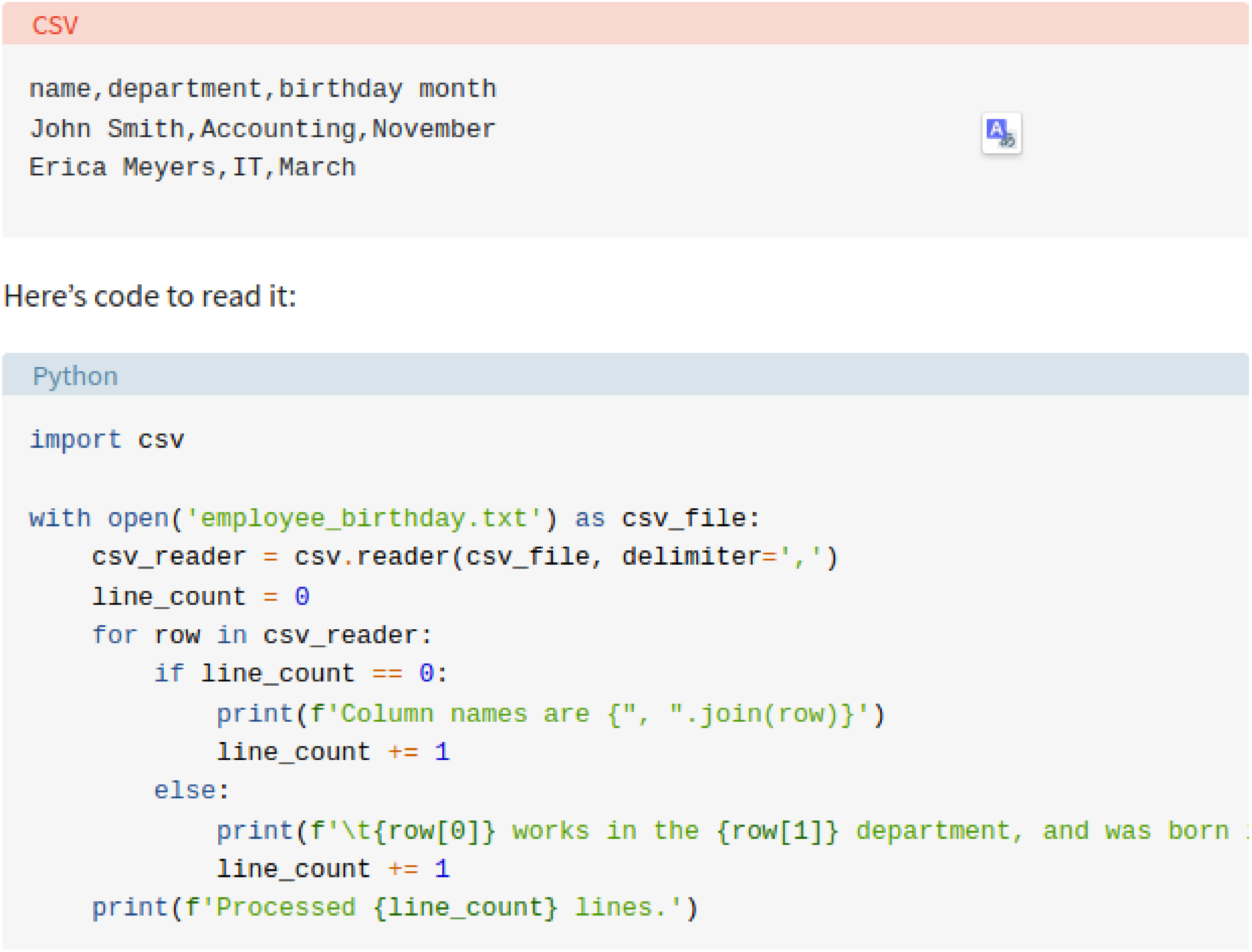
printable ASCII or Unicode characters.

Other popular delimiters include the tab (\t), colon (:) and semi-colon (;) characters. Properly parsing a CSV file requires us to know which delimiter is being used.

CSV files are very easy to work with programmatically. Any language that supports text file input and string manipulation (like Python) can work with CSV files directly.

Nice to work with unix commands like: **head, tail, split, cat,** etc.

Data formats



CSV

```
name,department,birthday month
John Smith,Accounting,November
Erica Meyers,IT,March
```

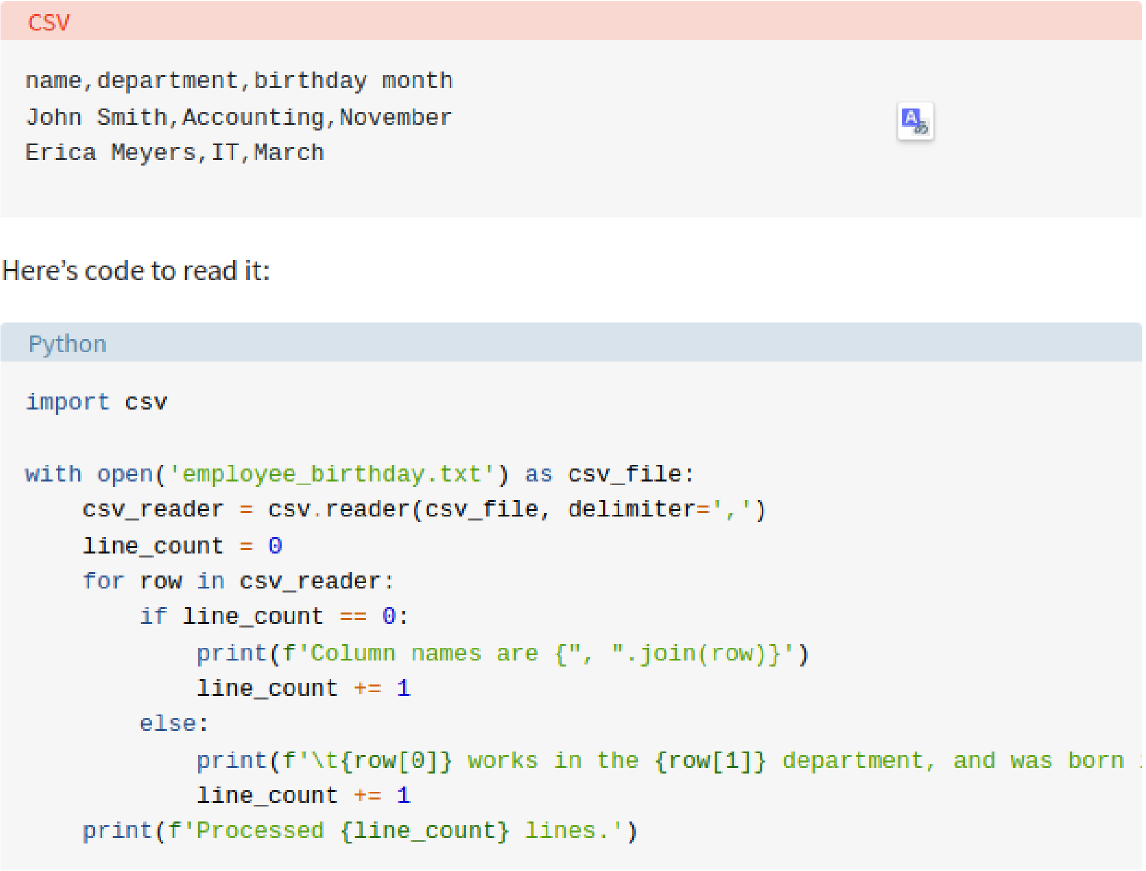
Here's code to read it:

Python

```python
import csv

with open('employee_birthday.txt') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            line_count += 1
        else:
            print(f'\t{row[0]} works in the {row[1]} department, and was born
            line_count += 1
    print(f'Processed {line_count} lines.')
```

https://realpython.com/python-csv/

Data formats



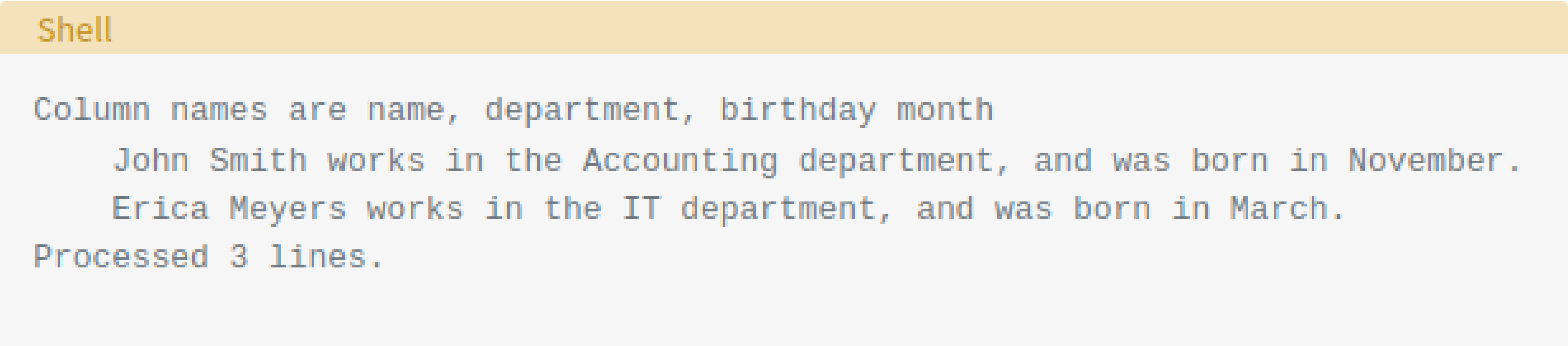
CSV

```
name,department,birthday month
John Smith,Accounting,November
Erica Meyers,IT,March
```

Here's code to read it:

Python

```python
import csv

with open('employee_birthday.txt') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    line_count = 0
    for row in csv_reader:
        if line_count == 0:
            print(f'Column names are {", ".join(row)}')
            line_count += 1
        else:
            print(f'\t{row[0]} works in the {row[1]} department, and was born
            line_count += 1
    print(f'Processed {line_count} lines.')
```



Shell

```
Column names are name, department, birthday month
    John Smith works in the Accounting department, and was born in November.
    Erica Meyers works in the IT department, and was born in March.
Processed 3 lines.
```

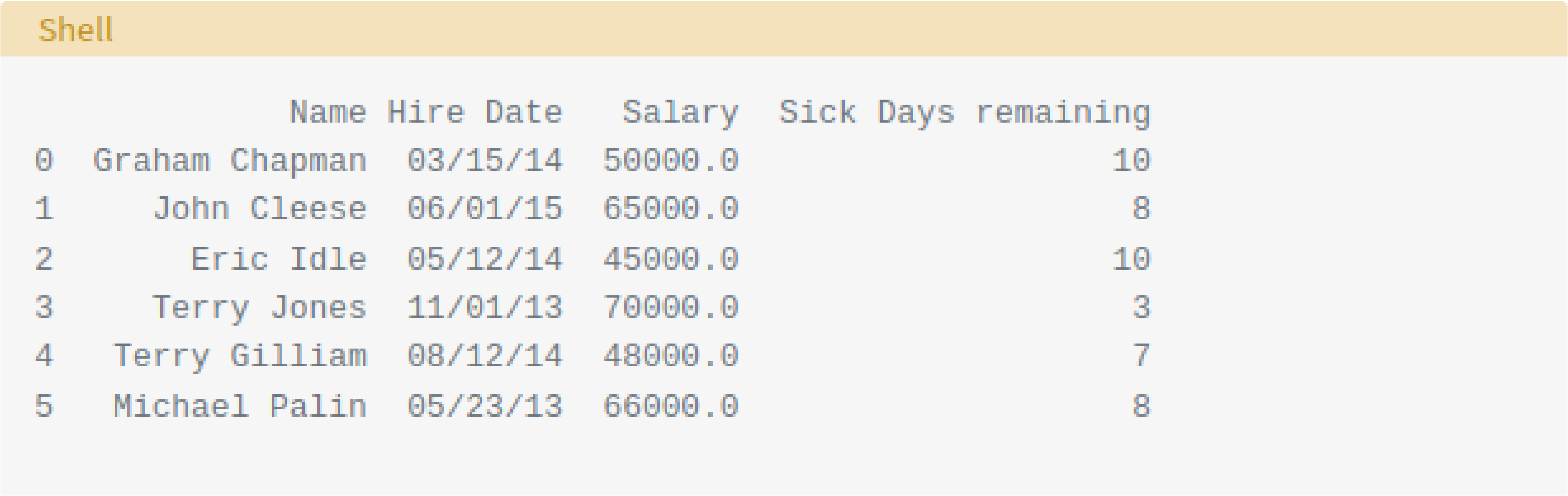
https://realpython.com/python-csv/

Data formats

```
import csv

import pandas
```

https://realpython.com/python-csv/

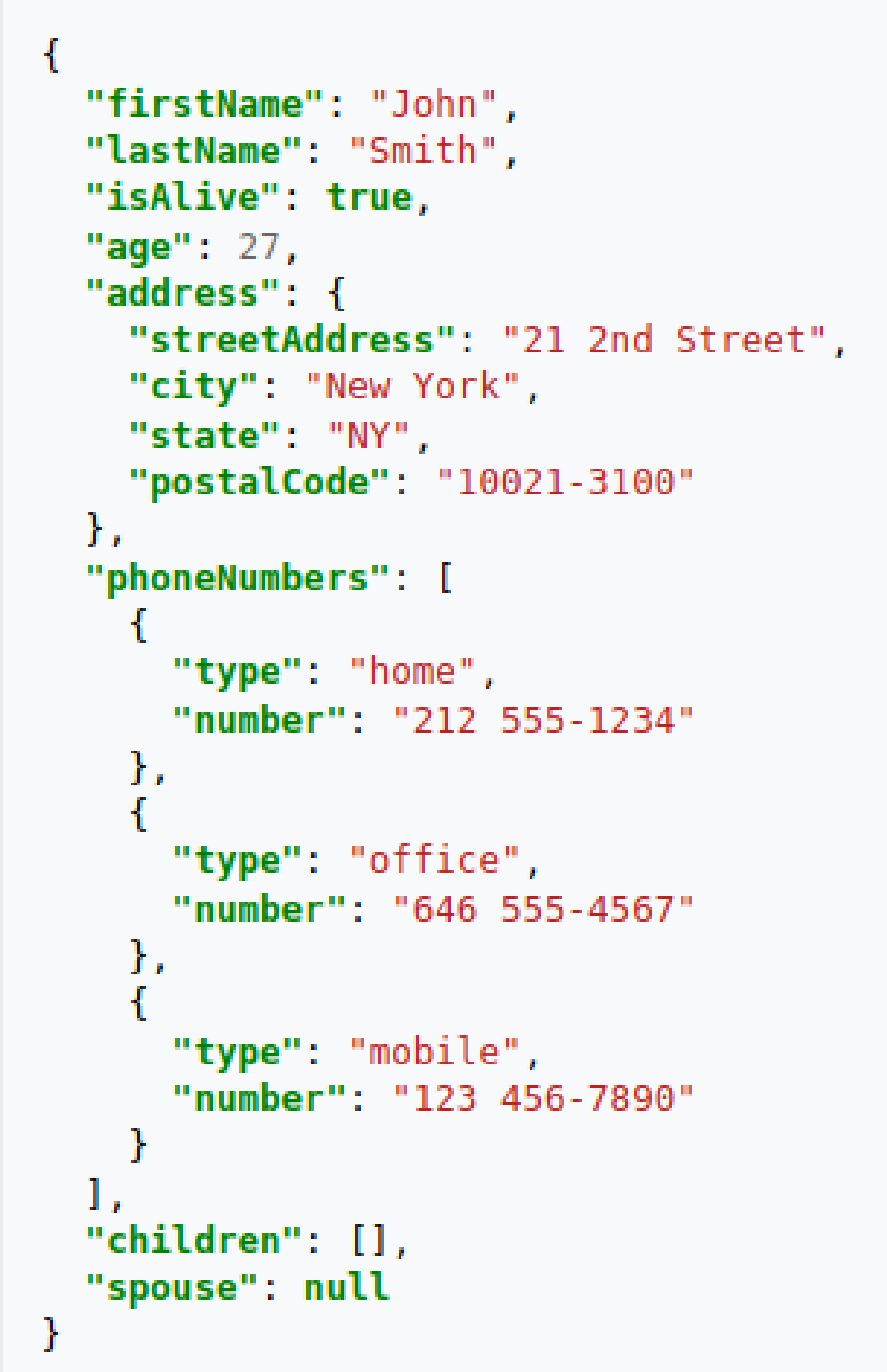
```
import csv

import pandas
df = pandas.read_csv('hrdata.csv')
print(df)
```



```
             Name Hire Date   Salary  Sick Days remaining
0  Graham Chapman  03/15/14  50000.0                   10
1     John Cleese  06/01/15  65000.0                    8
2       Eric Idle  05/12/14  45000.0                   10
3     Terry Jones  11/01/13  70000.0                    3
4   Terry Gilliam  08/12/14  48000.0                    7
5   Michael Palin  05/23/13  66000.0                    8
```

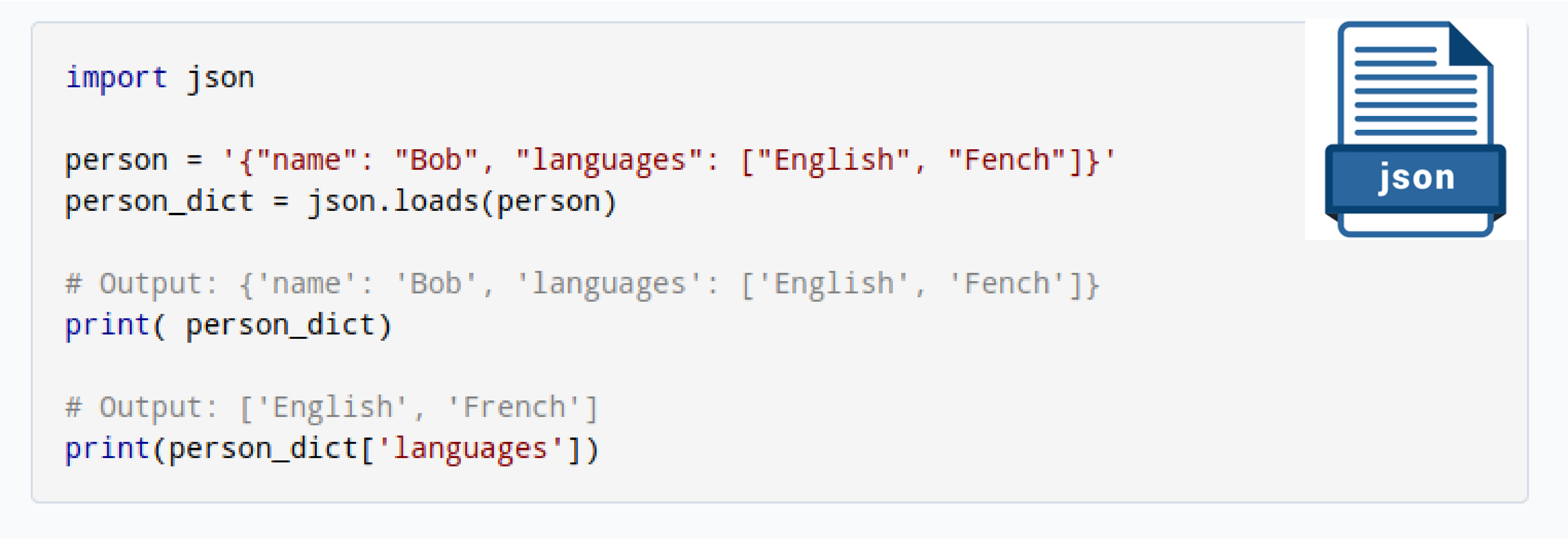
https://realpython.com/python-csv/

Data formats



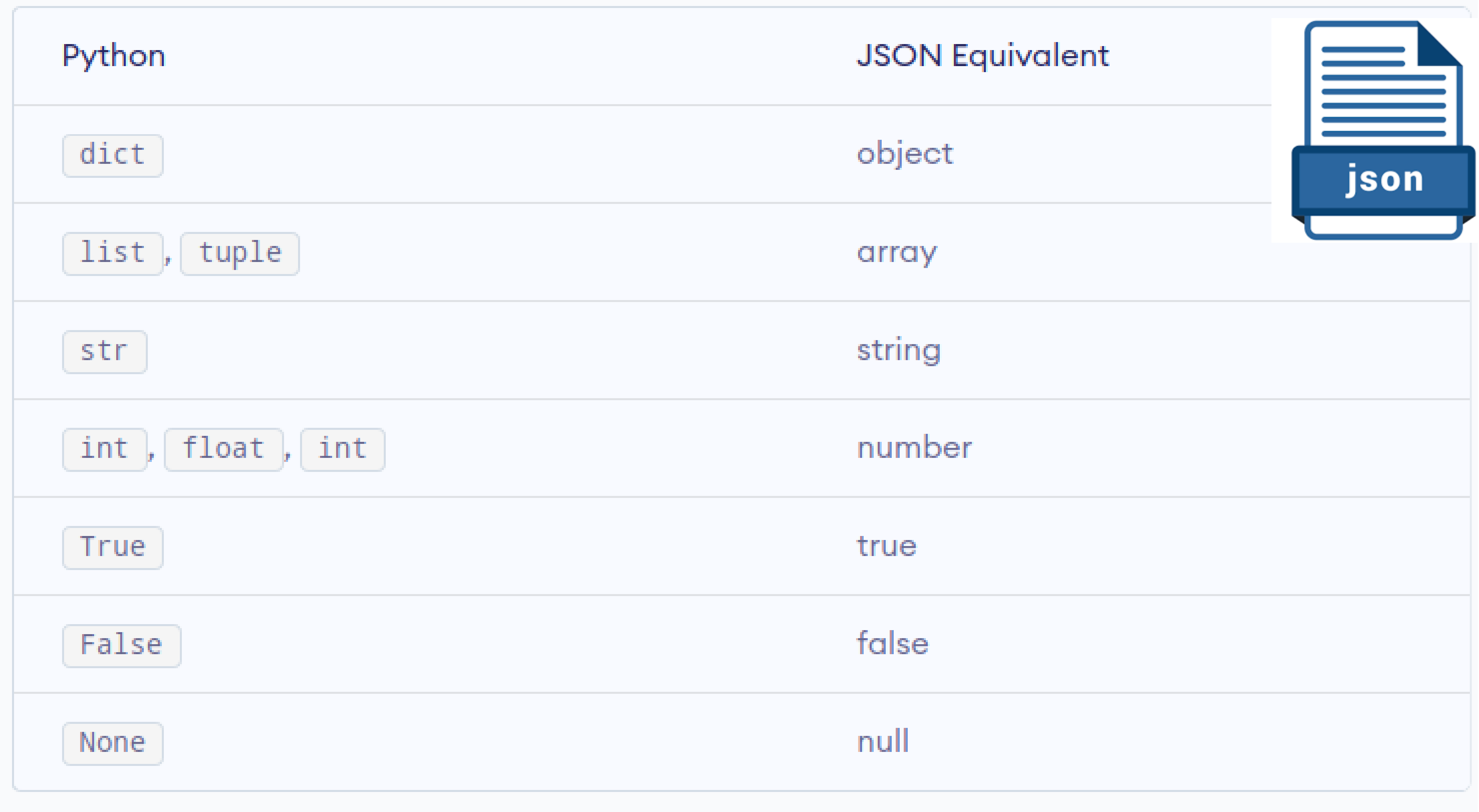
```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Data formats



```
import json

person = '{"name": "Bob", "languages": ["English", "Fench"]}'
person_dict = json.loads(person)

# Output: {'name': 'Bob', 'languages': ['English', 'Fench']}
print( person_dict)

# Output: ['English', 'French']
print(person_dict['languages'])
```

Data formats

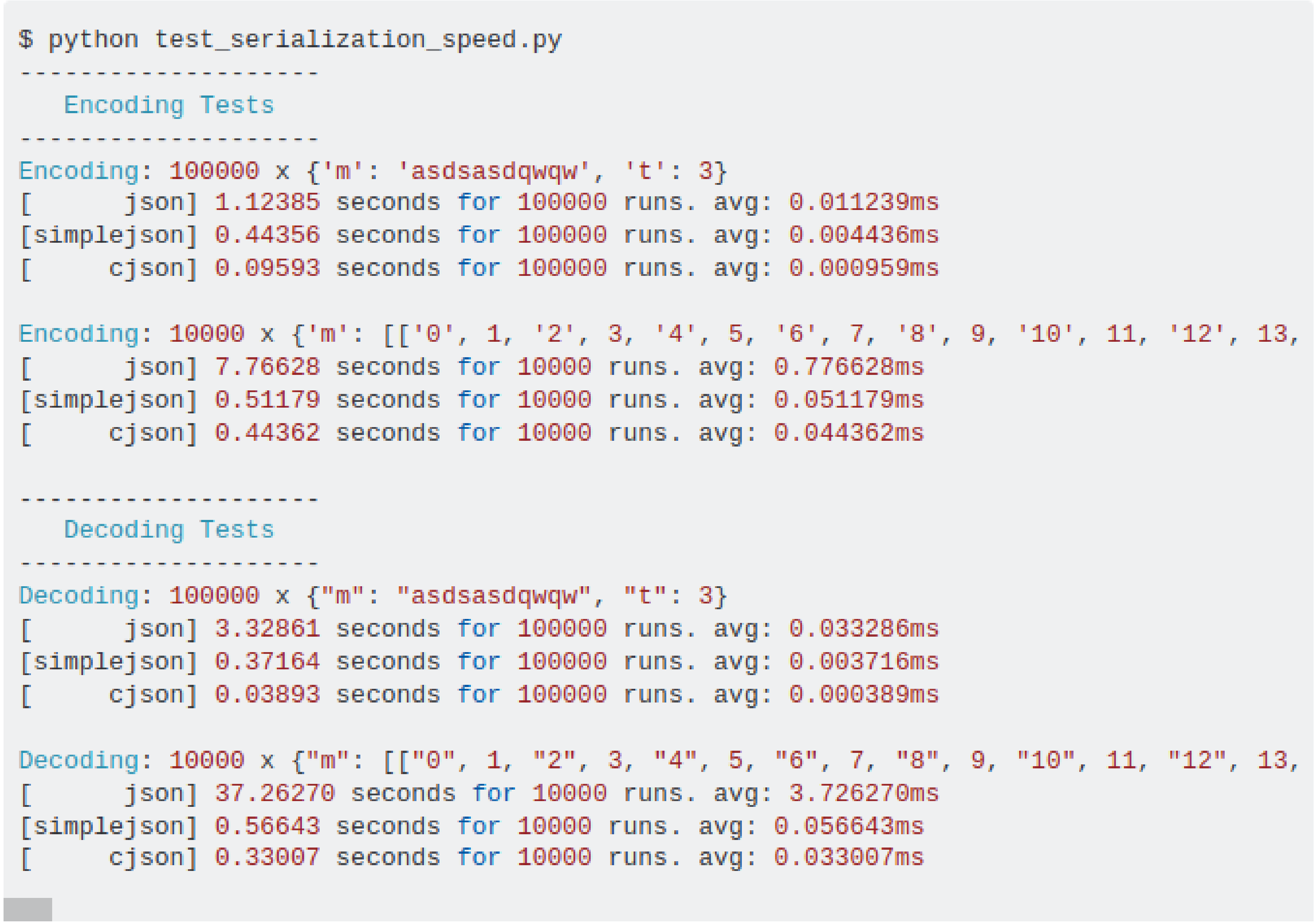


Python	JSON Equivalent
`dict`	object
`list`, `tuple`	array
`str`	string
`int`, `float`, `int`	number
`True`	true
`False`	false
`None`	null

MUST be in UTF-8

https://www.programiz.com/python-programming/json

Data formats



```
$ python test_serialization_speed.py
--------------------
   Encoding Tests
--------------------
Encoding: 100000 x {'m': 'asdsasdqwqw', 't': 3}
[      json] 1.12385 seconds for 100000 runs. avg: 0.011239ms
[simplejson] 0.44356 seconds for 100000 runs. avg: 0.004436ms
[     cjson] 0.09593 seconds for 100000 runs. avg: 0.000959ms

Encoding: 10000 x {'m': [['0', 1, '2', 3, '4', 5, '6', 7, '8', 9, '10', 11, '12', 13,
[      json] 7.76628 seconds for 10000 runs. avg: 0.776628ms
[simplejson] 0.51179 seconds for 10000 runs. avg: 0.051179ms
[     cjson] 0.44362 seconds for 10000 runs. avg: 0.044362ms

--------------------
   Decoding Tests
--------------------
Decoding: 100000 x {"m": "asdsasdqwqw", "t": 3}
[      json] 3.32861 seconds for 100000 runs. avg: 0.033286ms
[simplejson] 0.37164 seconds for 100000 runs. avg: 0.003716ms
[     cjson] 0.03893 seconds for 100000 runs. avg: 0.000389ms

Decoding: 10000 x {"m": [["0", 1, "2", 3, "4", 5, "6", 7, "8", 9, "10", 11, "12", 13,
[      json] 37.26270 seconds for 10000 runs. avg: 3.726270ms
[simplejson] 0.56643 seconds for 10000 runs. avg: 0.056643ms
[     cjson] 0.33007 seconds for 10000 runs. avg: 0.033007ms
```

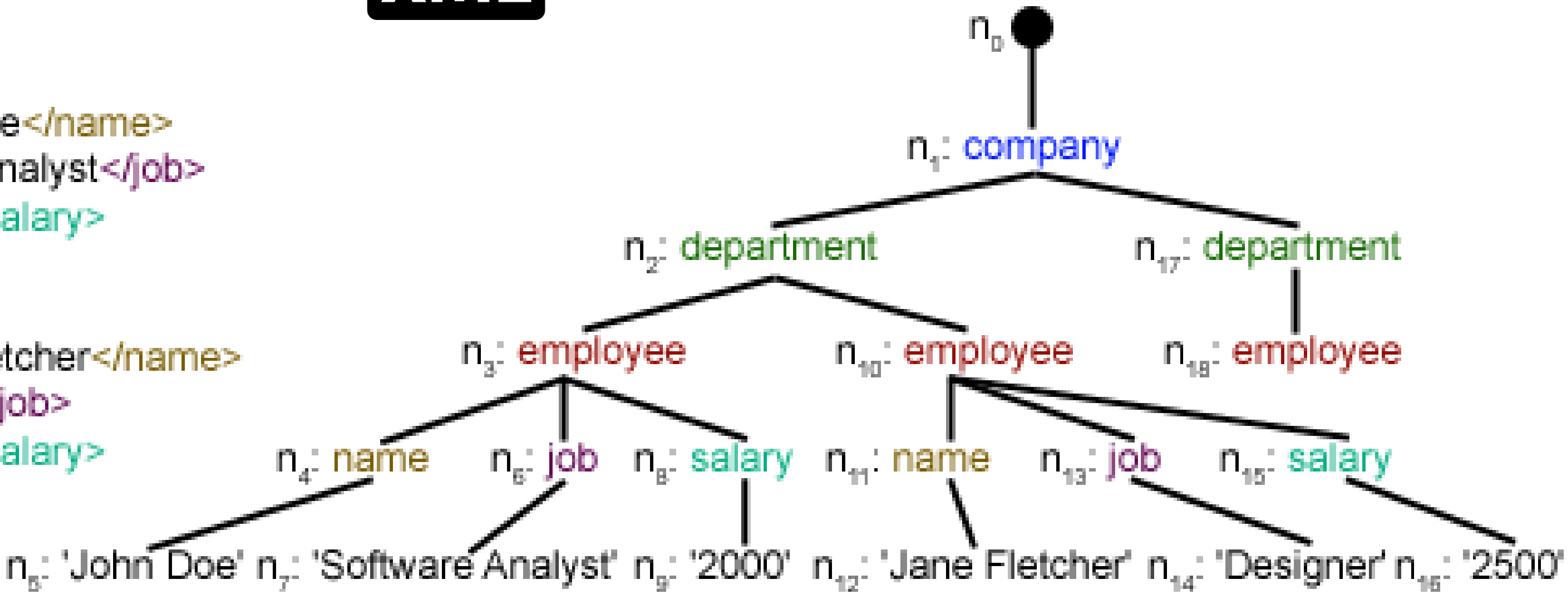
JSON libraries in python:

simplejson, ujson, cjson, ...

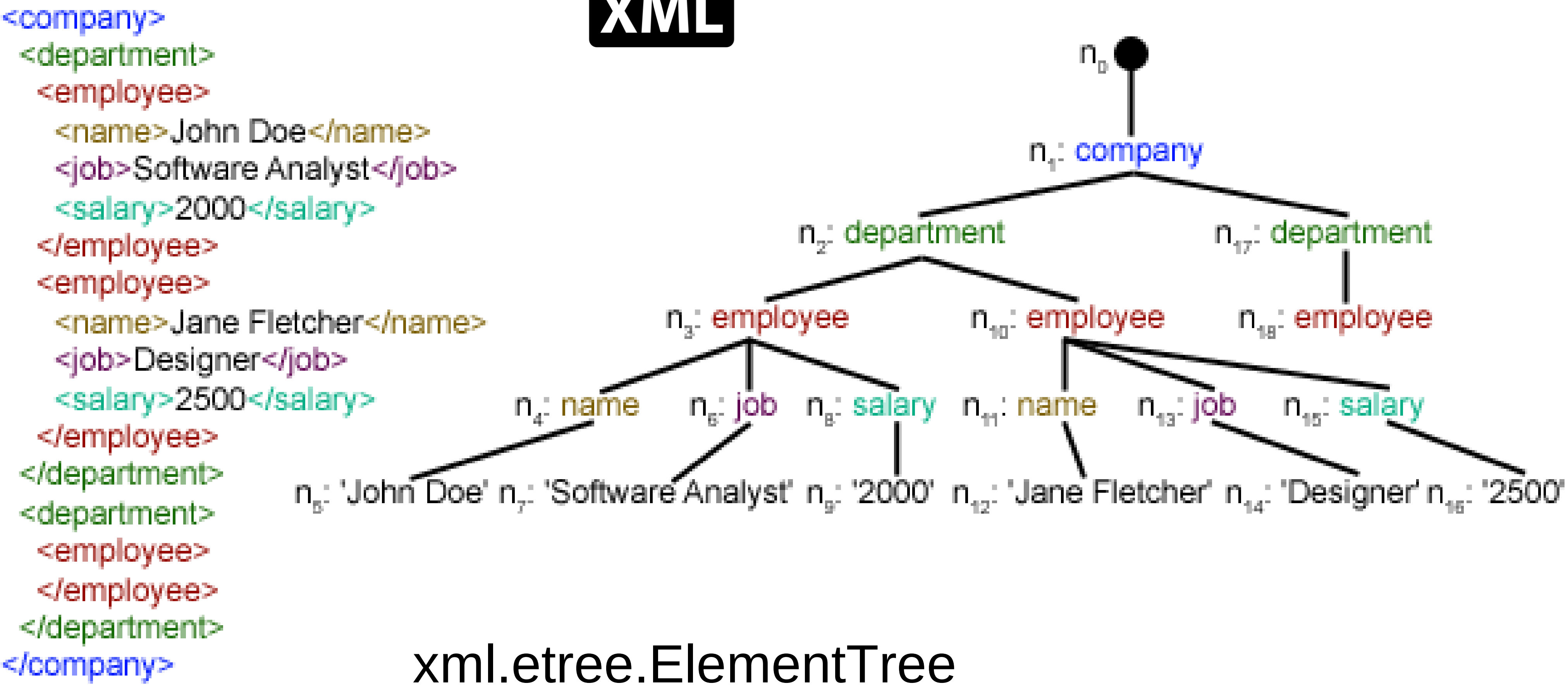
https://stackoverflow.com/questions/712791/what-are-the-differences-between-json-and-simplejson-python-modules

Data formats

```
<company>
  <department>
    <employee>
      <name>John Doe</name>
      <job>Software Analyst</job>
      <salary>2000</salary>
    </employee>
    <employee>
      <name>Jane Fletcher</name>
      <job>Designer</job>
      <salary>2500</salary>
    </employee>
  </department>
  <department>
    <employee>
    </employee>
  </department>
</company>
```



Data formats

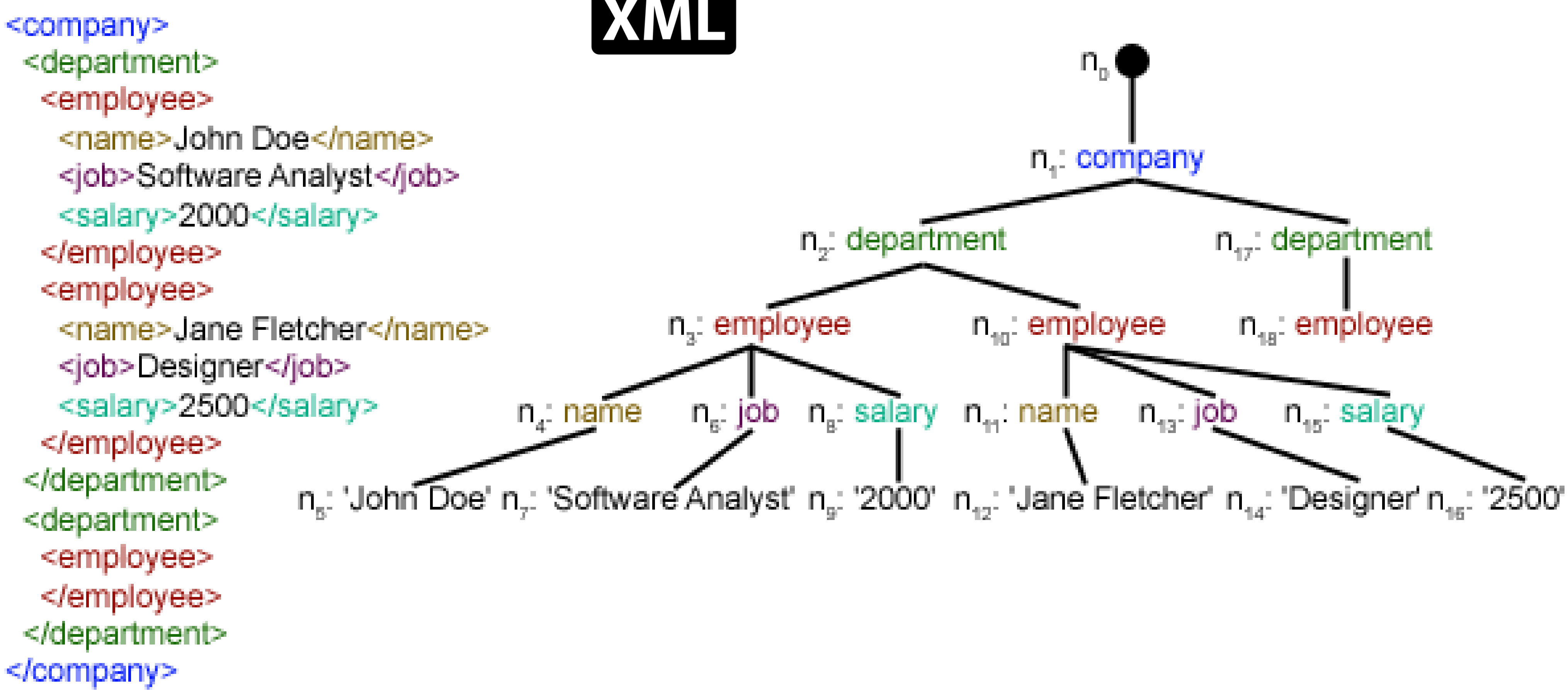


```
<company>
  <department>
    <employee>
      <name>John Doe</name>
      <job>Software Analyst</job>
      <salary>2000</salary>
    </employee>
    <employee>
      <name>Jane Fletcher</name>
      <job>Designer</job>
      <salary>2500</salary>
    </employee>
  </department>
  <department>
    <employee>
    </employee>
  </department>
</company>
```

xml.etree.ElementTree

xml.dom.minidom

etree.XMLParser() [from lxml)



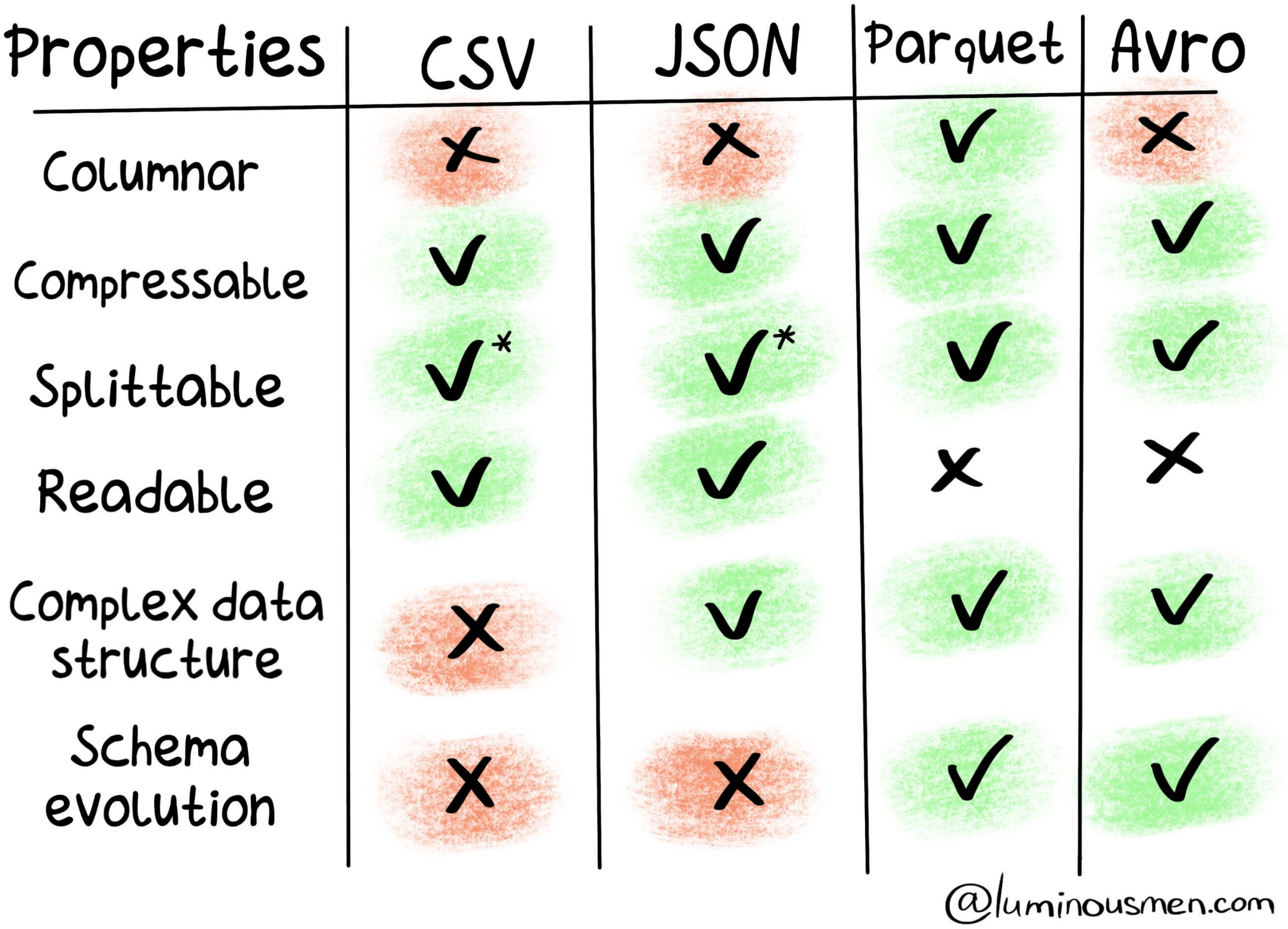
```xml
<company>
 <department>
  <employee>
   <name>John Doe</name>
   <job>Software Analyst</job>
   <salary>2000</salary>
  </employee>
  <employee>
   <name>Jane Fletcher</name>
   <job>Designer</job>
   <salary>2500</salary>
  </employee>
 </department>
 <department>
  <employee>
  </employee>
 </department>
</company>
```

XML is not known for being short and sweet

Human-readable, although …

you can get lost in-between all the tags in-front of your eyes

Data formats



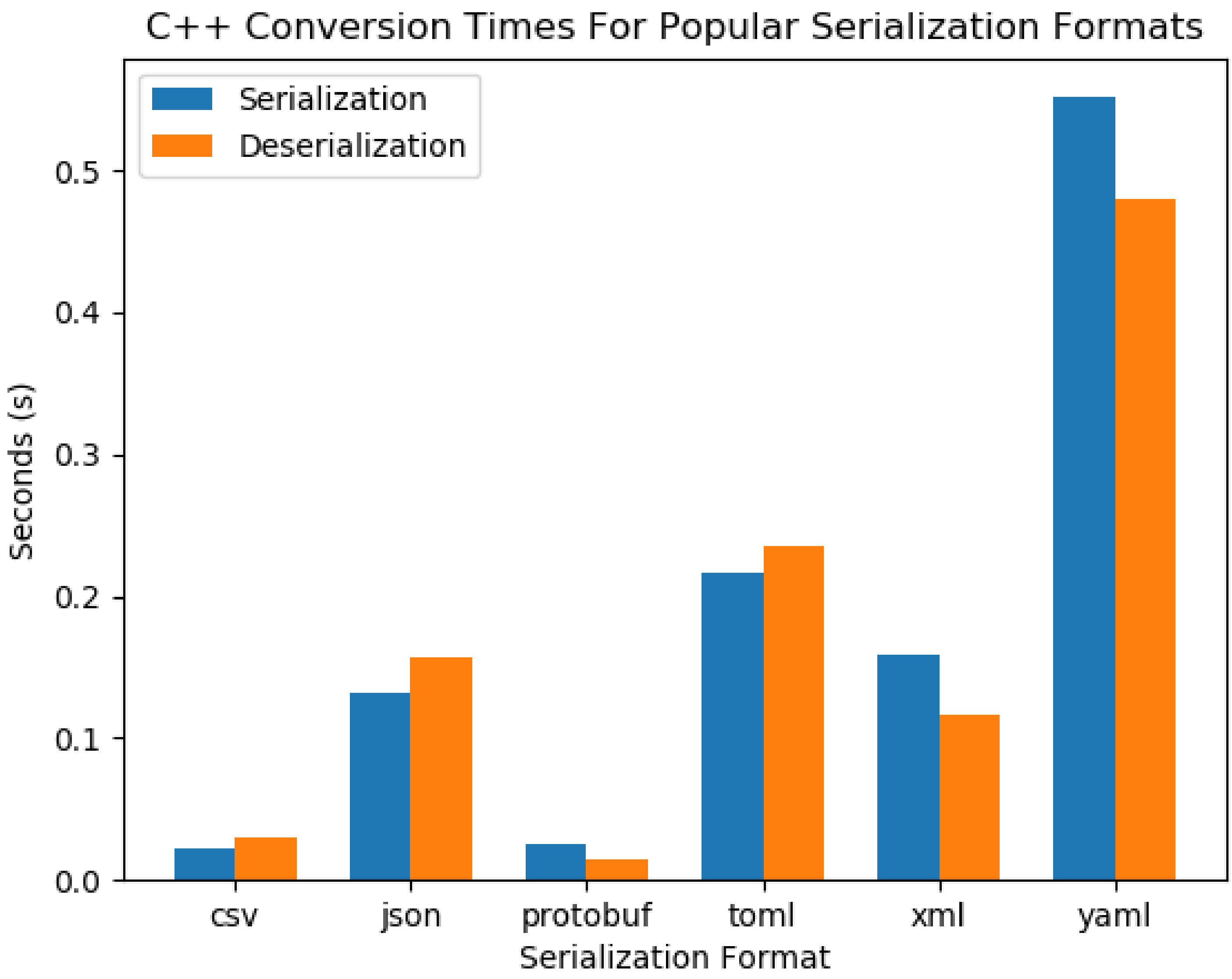
Properties	CSV	JSON	Parquet	Avro
Columnar	✗	✗	✓	✗
Compressable	✓	✓	✓	✓
Splittable	✓*	✓*	✓	✓
Readable	✓	✓	✗	✗
Complex data structure	✗	✓	✓	✓
Schema evolution	✗	✗	✓	✓

@luminousmen.com

Parquet (Apache) – column based format

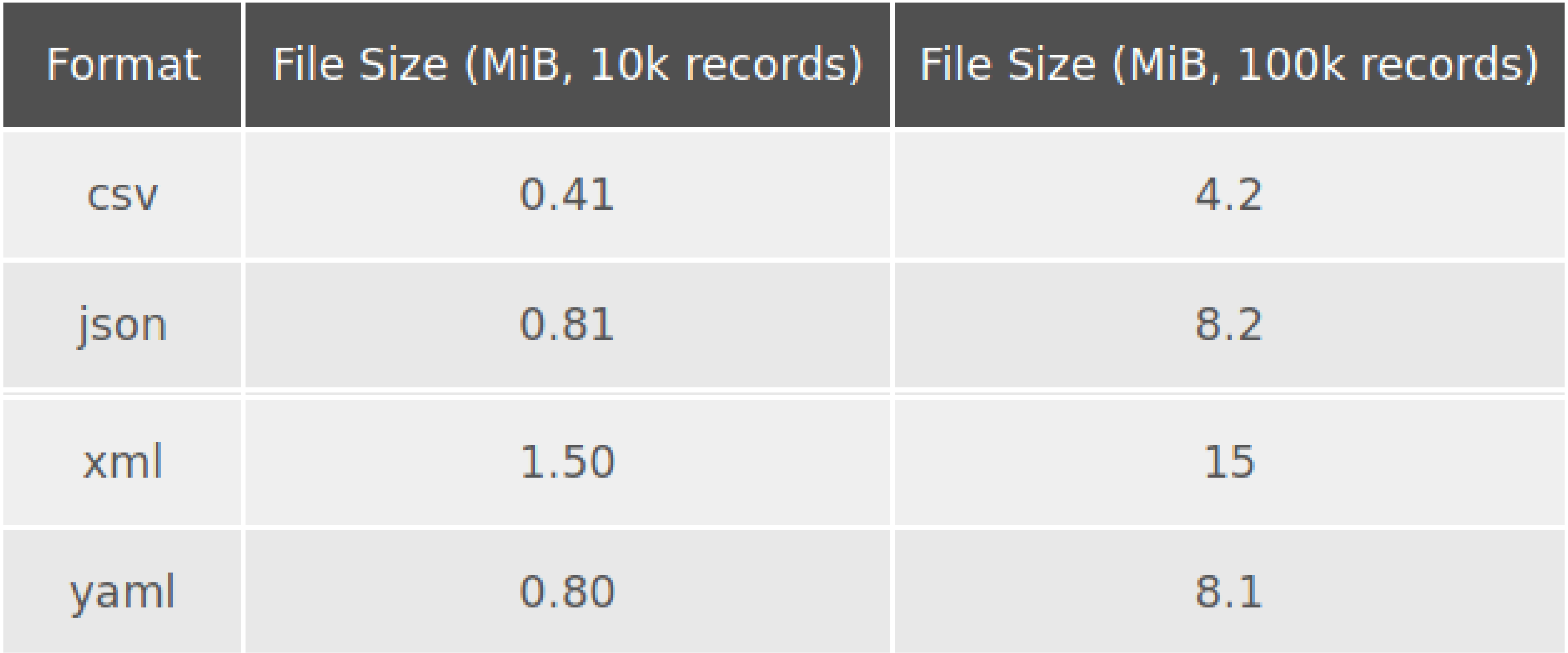
Avro (Hadoop) – row based format

Data formats



https://blog.mbedded.ninja/programming/serialization-formats/a-comparison-of-serialization-formats/

Data formats

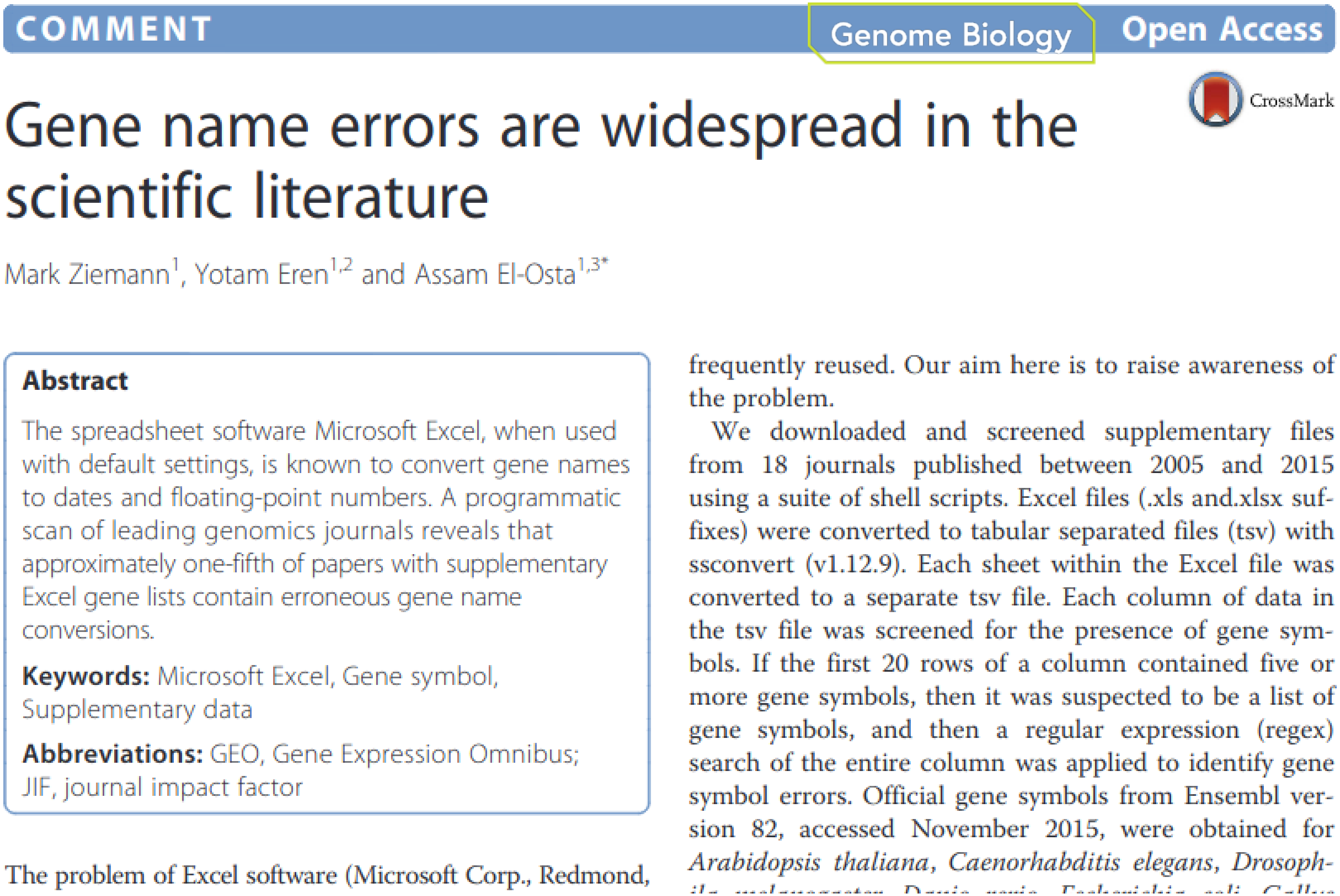


Format	File Size (MiB, 10k records)	File Size (MiB, 100k records)
csv	0.41	4.2
json	0.81	8.2
xml	1.50	15
yaml	0.80	8.1

https://blog.mbedded.ninja/programming/serialization-formats/a-comparison-of-serialization-formats/

Data formats

Data formats



COMMENT

Genome Biology

Open Access

Gene name errors are widespread in the scientific literature

Mark Ziemann[1], Yotam Eren[1,2] and Assam El-Osta[1,3*]

Abstract

The spreadsheet software Microsoft Excel, when used with default settings, is known to convert gene names to dates and floating-point numbers. A programmatic scan of leading genomics journals reveals that approximately one-fifth of papers with supplementary Excel gene lists contain erroneous gene name conversions.

Keywords: Microsoft Excel, Gene symbol, Supplementary data

Abbreviations: GEO, Gene Expression Omnibus; JIF, journal impact factor

The problem of Excel software (Microsoft Corp., Redmond,

frequently reused. Our aim here is to raise awareness of the problem.

We downloaded and screened supplementary files from 18 journals published between 2005 and 2015 using a suite of shell scripts. Excel files (.xls and.xlsx suffixes) were converted to tabular separated files (tsv) with ssconvert (v1.12.9). Each sheet within the Excel file was converted to a separate tsv file. Each column of data in the tsv file was screened for the presence of gene symbols. If the first 20 rows of a column contained five or more gene symbols, then it was suspected to be a list of gene symbols, and then a regular expression (regex) search of the entire column was applied to identify gene symbol errors. Official gene symbols from Ensembl version 82, accessed November 2015, were obtained for *Arabidopsis thaliana*, *Caenorhabditis elegans*, *Drosophila melanogaster*, *Danio rerio*, *Escherichia coli*, *Gallus*

Data formats

Gene symbols:
SEPT2 (Septin 2) --> ‘2-Sep’
MARCH1 [Membrane-Associated Ring Finger (C3HC4) 1] --> ‘1-Mar’

RIKEN identifiers:
‘2310009E13’ --> ‘2.31E+13’

Data formats

Gene symbols:
SEPT2 (Septin 2) --> ‘2-Sep’
MARCH1 [Membrane-Associated Ring Finger (C3HC4) 1] --> ‘1-Mar’

RIKEN identifiers:
‘2310009E13’ --> ‘2.31E+13’

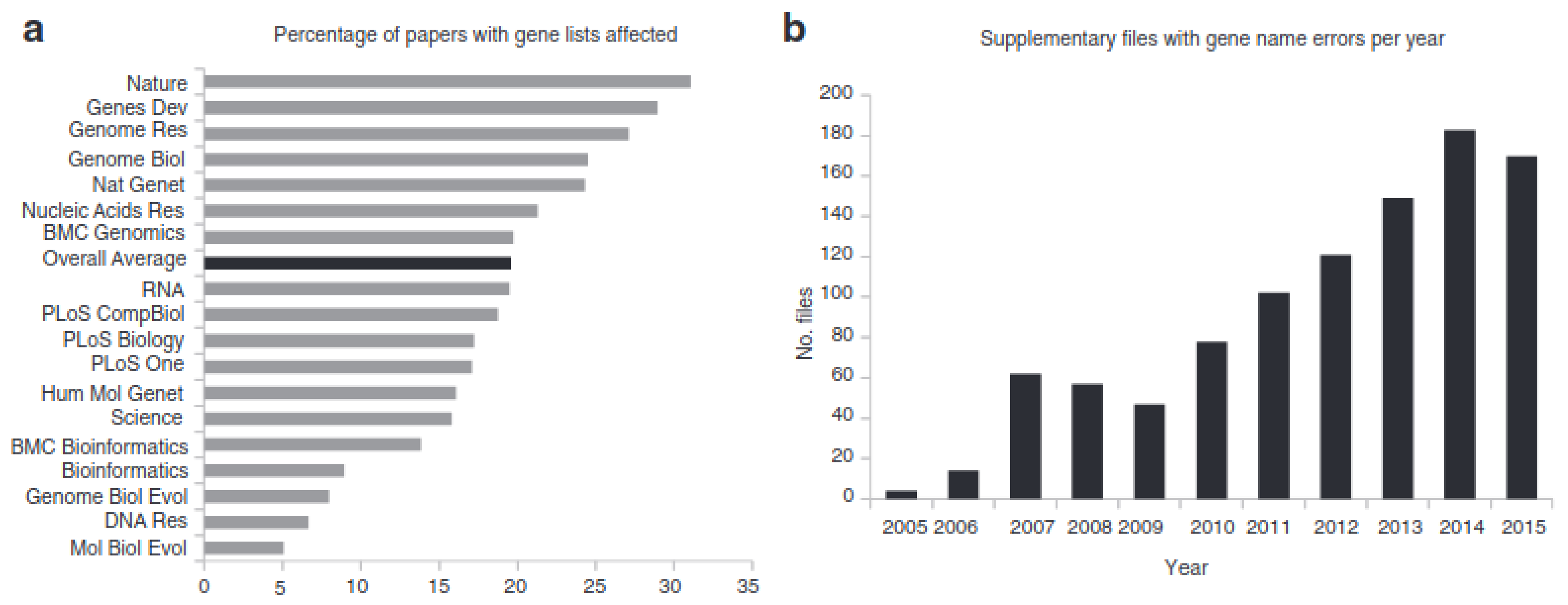
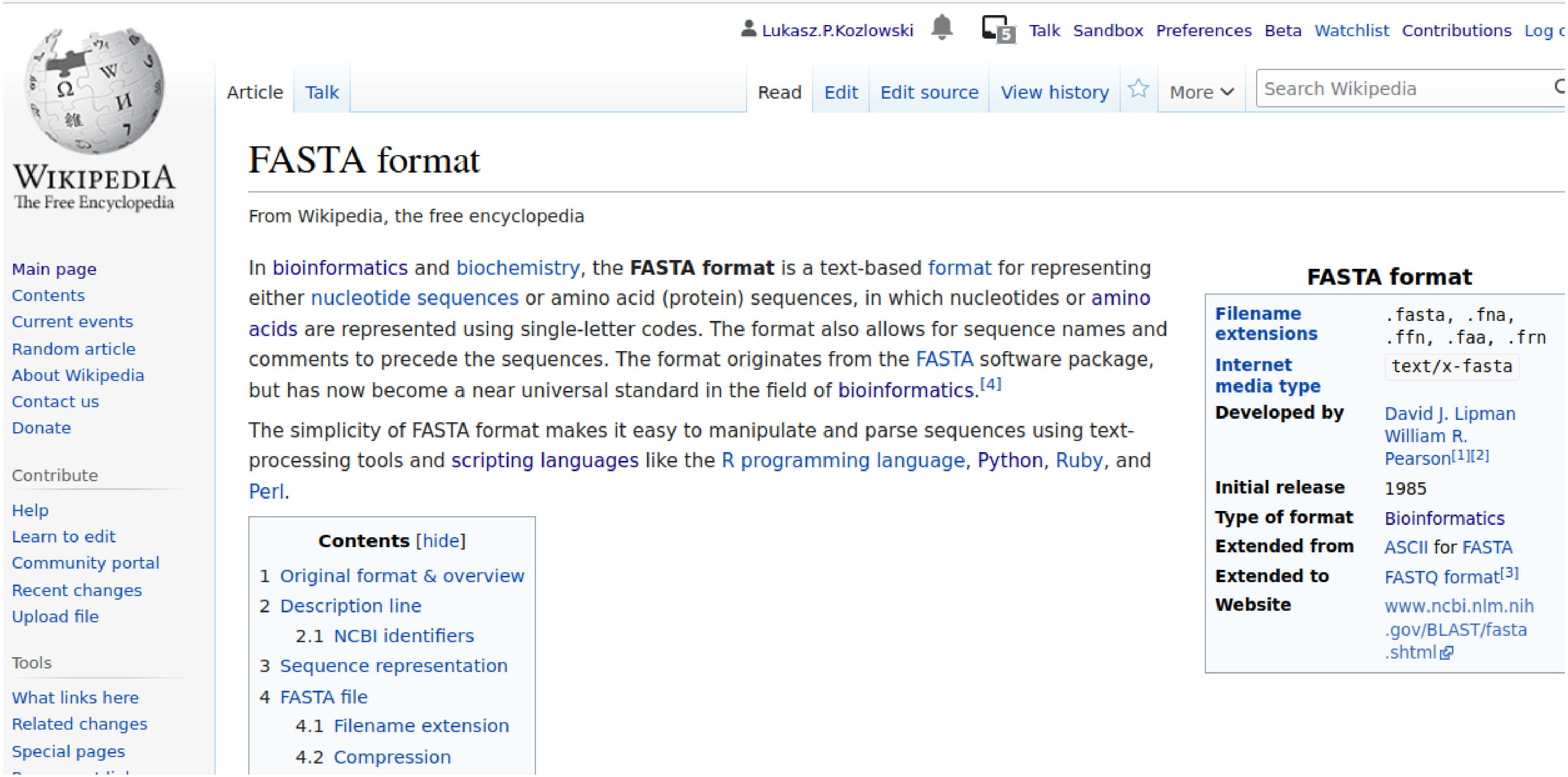


Fig. 1 Prevalence of gene name errors in supplementary Excel files. **a** Percentage of published papers with supplementary gene lists in Excel files affected by gene name errors. **b** Increase in gene name errors by year

Domain specific formats

Bioinformatics

Data formats



FASTA format

From Wikipedia, the free encyclopedia

In bioinformatics and biochemistry, the **FASTA format** is a text-based format for representing either nucleotide sequences or amino acid (protein) sequences, in which nucleotides or amino acids are represented using single-letter codes. The format also allows for sequence names and comments to precede the sequences. The format originates from the FASTA software package, but has now become a near universal standard in the field of bioinformatics.[4]

The simplicity of FASTA format makes it easy to manipulate and parse sequences using text-processing tools and scripting languages like the R programming language, Python, Ruby, and Perl.

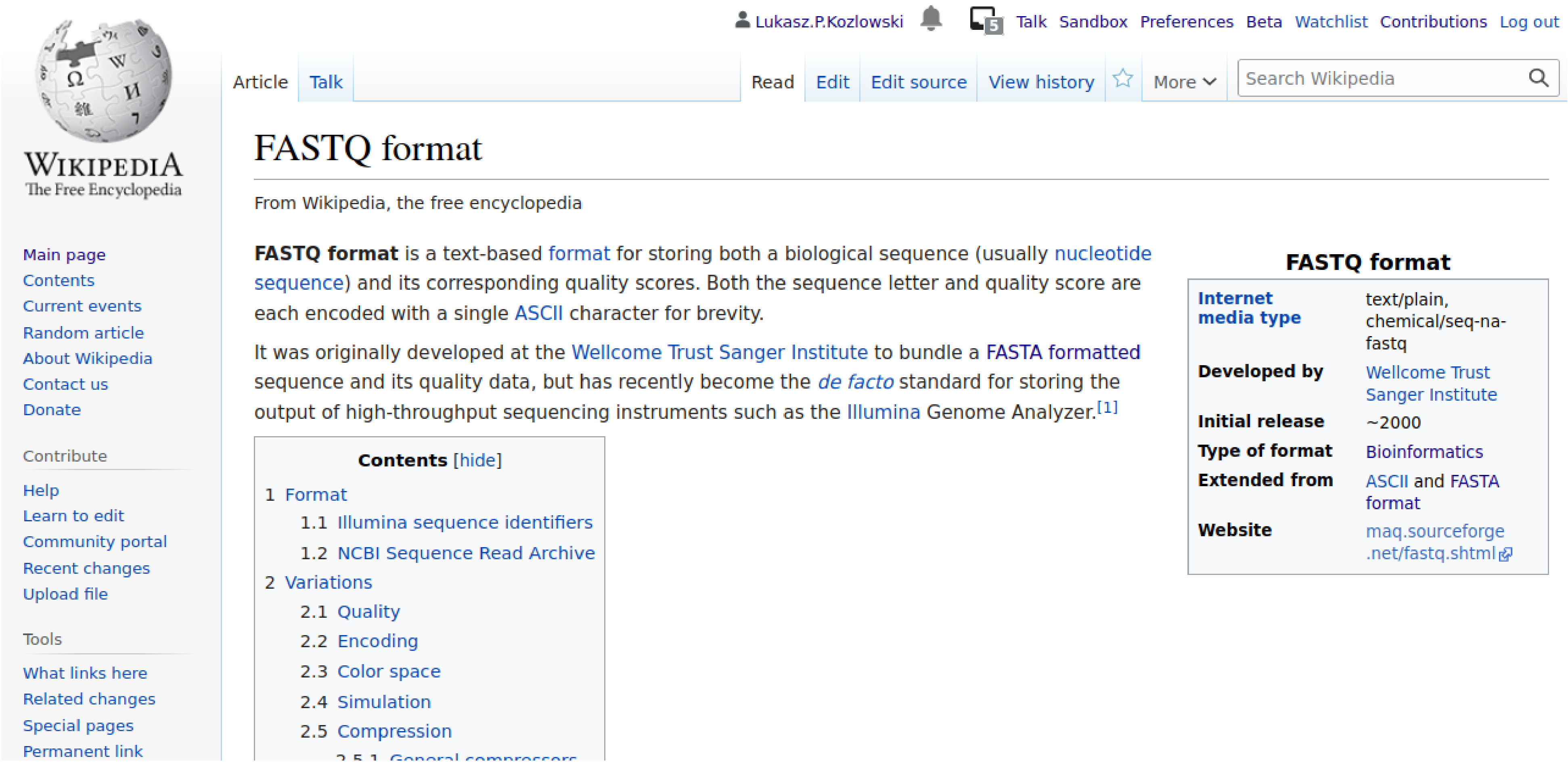
FASTA format

Filename extensions	`.fasta, .fna, .ffn, .faa, .frn`
Internet media type	`text/x-fasta`
Developed by	David J. Lipman William R. Pearson[1][2]
Initial release	1985
Type of format	Bioinformatics
Extended from	ASCII for FASTA
Extended to	FASTQ format[3]
Website	www.ncbi.nlm.nih.gov/BLAST/fasta.shtml

Contents [hide]

https://en.wikipedia.org/wiki/FASTA_format

Data formats



FASTQ format

From Wikipedia, the free encyclopedia

FASTQ format is a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores. Both the sequence letter and quality score are each encoded with a single ASCII character for brevity.

It was originally developed at the Wellcome Trust Sanger Institute to bundle a FASTA formatted sequence and its quality data, but has recently become the *de facto* standard for storing the output of high-throughput sequencing instruments such as the Illumina Genome Analyzer.[1]

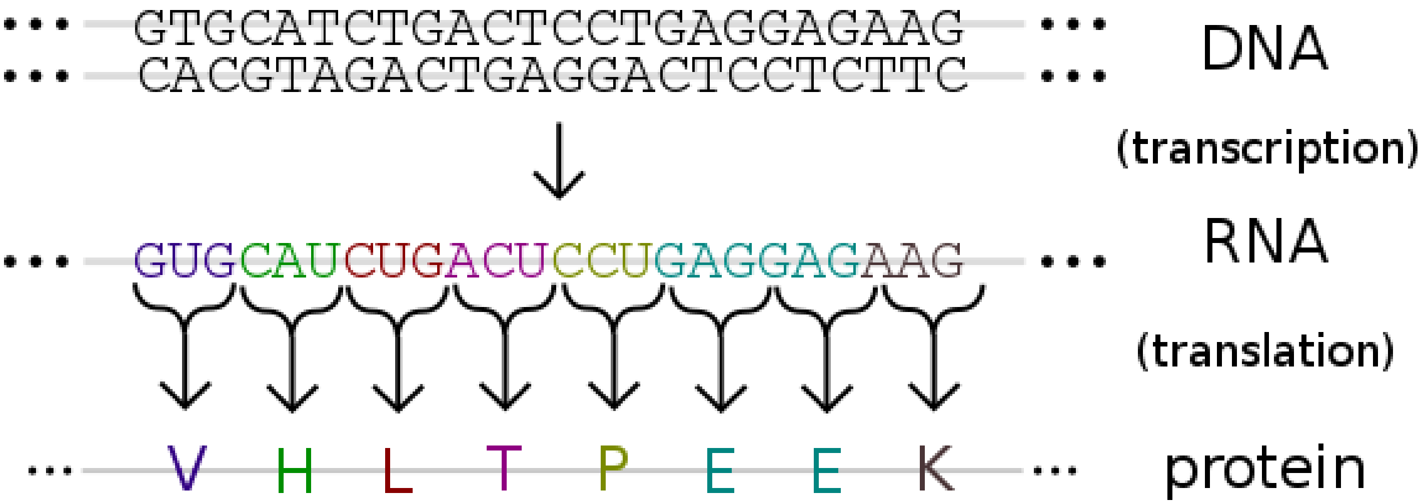
FASTQ format

Internet media type	text/plain, chemical/seq-na-fastq
Developed by	Wellcome Trust Sanger Institute
Initial release	~2000
Type of format	Bioinformatics
Extended from	ASCII and FASTA format
Website	maq.sourceforge.net/fastq.shtml

Contents [hide]

https://en.wikipedia.org/wiki/FASTQ_format

Biology



… GTGCATCTGACTCCTGAGGAGAAG … DNA

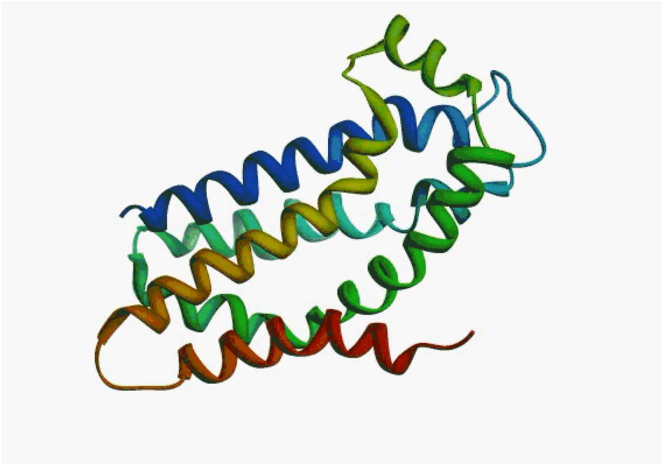
… CACGTAGACTGAGGACTCCTCTTC …

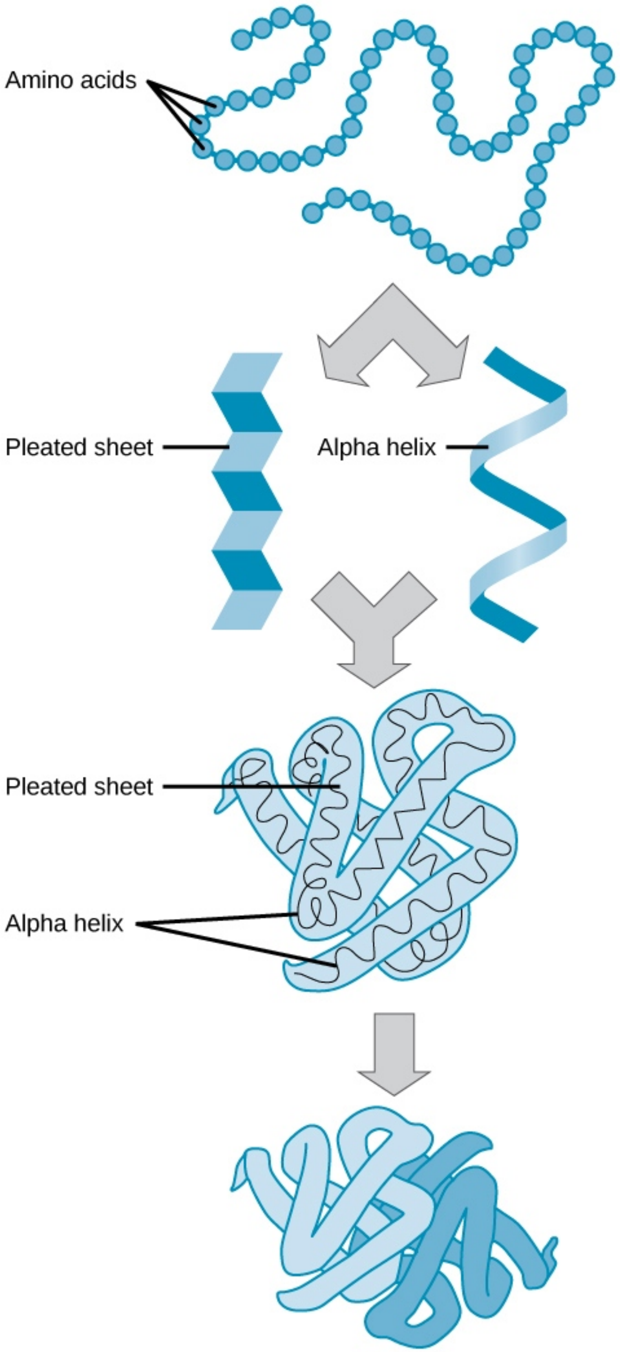
(transcription)

… GUGCAUCUGACUCCUGAGGAGAAG … RNA

(translation)

… V H L T P E E K … protein





Primary Protein structure
sequence of a chain of
animo acids

Secondary Protein structure
hydrogen bonding of the peptide
backbone causes the amino
acids to fold into a repeating
pattern

Tertiary protein structure
three-dimensional folding
pattern of a protein due to side
chain interactions

Quaternary protein structure
protein consisting of more
than one amino acid chain

Few words about proteins

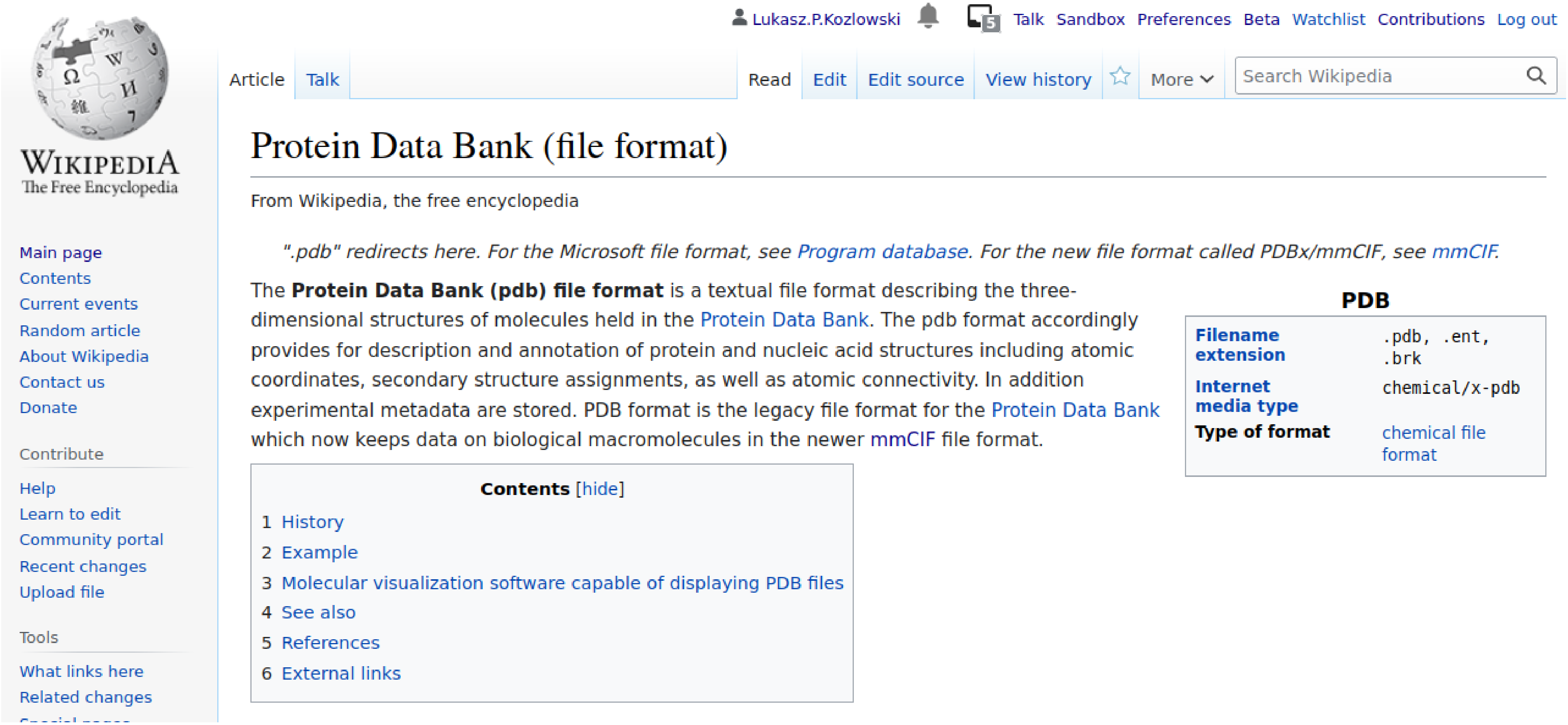
Protein modeling

(USCF Chimera)

Protein disorder

http://iimcb.genesilico.pl/metadisorder/protein_disorder_intrinsically_unstructured_proteins_gallery_images.html

Data formats



Protein Data Bank (file format)

From Wikipedia, the free encyclopedia

".pdb" redirects here. For the Microsoft file format, see Program database. For the new file format called PDBx/mmCIF, see mmCIF.

The **Protein Data Bank (pdb) file format** is a textual file format describing the three-dimensional structures of molecules held in the Protein Data Bank. The pdb format accordingly provides for description and annotation of protein and nucleic acid structures including atomic coordinates, secondary structure assignments, as well as atomic connectivity. In addition experimental metadata are stored. PDB format is the legacy file format for the Protein Data Bank which now keeps data on biological macromolecules in the newer mmCIF file format.

PDB

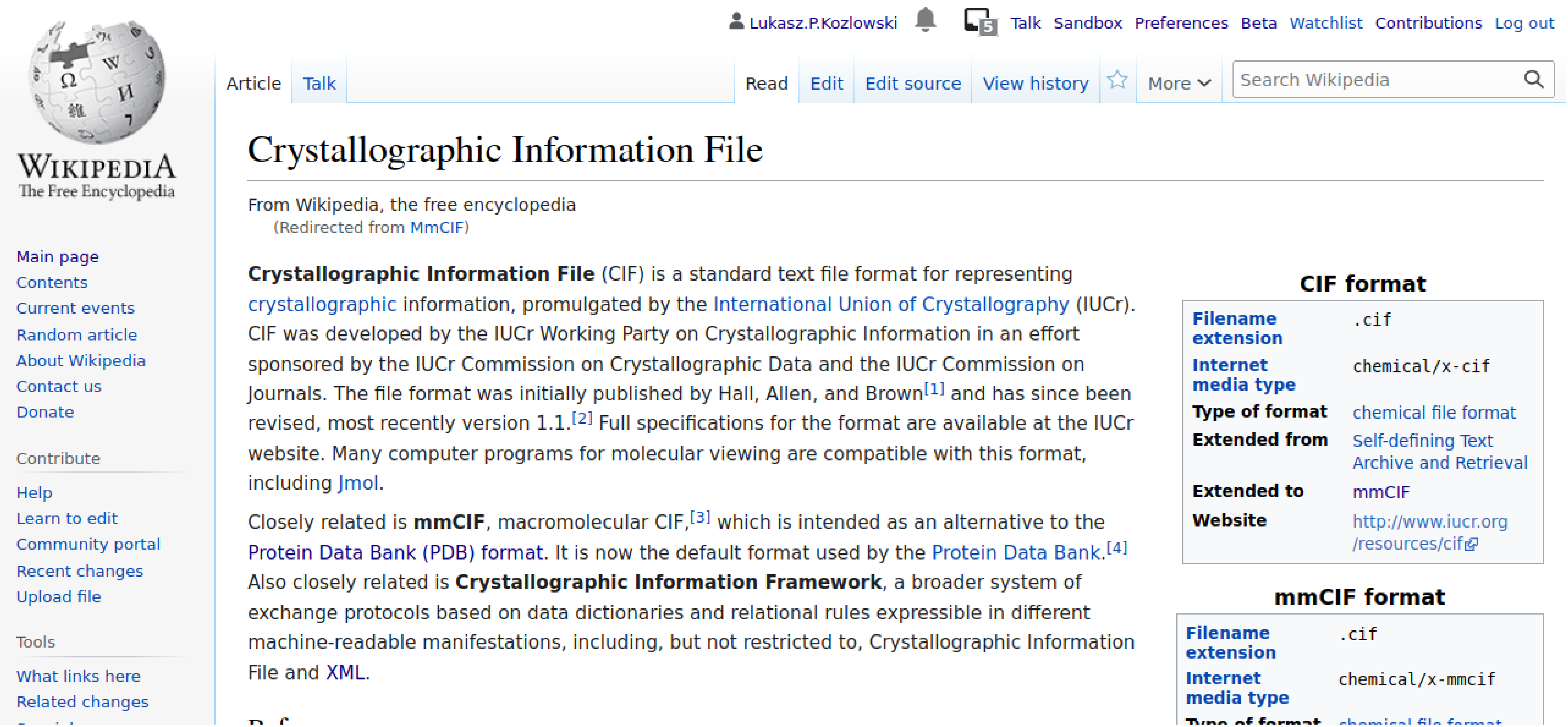
Filename extension	`.pdb`, `.ent`, `.brk`
Internet media type	`chemical/x-pdb`
Type of format	chemical file format

Contents [hide]

https://en.wikipedia.org/wiki/Protein_Data_Bank_(file_format)

Data formats

ADP



Crystallographic Information File

From Wikipedia, the free encyclopedia
(Redirected from MmCIF)

Crystallographic Information File (CIF) is a standard text file format for representing crystallographic information, promulgated by the International Union of Crystallography (IUCr). CIF was developed by the IUCr Working Party on Crystallographic Information in an effort sponsored by the IUCr Commission on Crystallographic Data and the IUCr Commission on Journals. The file format was initially published by Hall, Allen, and Brown[1] and has since been revised, most recently version 1.1.[2] Full specifications for the format are available at the IUCr website. Many computer programs for molecular viewing are compatible with this format, including Jmol.

Closely related is **mmCIF**, macromolecular CIF,[3] which is intended as an alternative to the Protein Data Bank (PDB) format. It is now the default format used by the Protein Data Bank.[4] Also closely related is **Crystallographic Information Framework**, a broader system of exchange protocols based on data dictionaries and relational rules expressible in different machine-readable manifestations, including, but not restricted to, Crystallographic Information File and XML.

CIF format

Filename extension	`.cif`
Internet media type	`chemical/x-cif`
Type of format	chemical file format
Extended from	Self-defining Text Archive and Retrieval
Extended to	mmCIF
Website	http://www.iucr.org/resources/cif

mmCIF format

Filename extension	`.cif`
Internet media type	`chemical/x-mmcif`
Type of format	chemical file format

https://en.wikipedia.org/wiki/Crystallographic_Information_File

Thank you for your time
and
See you at the next lecture

Any other
questions & comments

lukaskoz@mimuw.edu.pl