

# Data Science for Business

## *Data Formats and Collection*

Asst. Prof. Teerapong Leelanupab (Ph.D.)  
Faculty of Information Technology  
King Mongkut's Institute of Technology Ladkrabang (KMITL)



Week 4.1

# Data Formats: Plain Text

---

- Data is generated in different domains (e.g. finance, bioinformatics, engineering), and stored in many different formats.
- **Plain text:** Data is often stored as plain text (.txt) with no specific mark-up or metadata to explain its structure.
- When dealing with raw text, we may often want to tokenize the text to extract words and phrases.

PRIDE AND PREJUDICE  
By Jane Austen

Chapter 1

It is a truth universally acknowledged, that a single man in possession of a good fortune, must be in want of a wife.

However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered the rightful property of some one or other of their daughters.

"My dear Mr. Bennet," said his lady to him one day, "have you heard that Netherfield Park is let at last?"

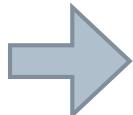
```
f = open("pride.txt", "r")
lines = f.readlines()
f.close()
for line in lines:
    line = line.strip()
    if len(line) > 0:
        words = line.split(" ")
        print(words)
```

```
[ 'PRIDE', 'AND', 'PREJUDICE' ]
[ 'By', 'Jane', 'Austen' ]
[ 'Chapter', '1' ]
...
```

# Data Formats: CSV

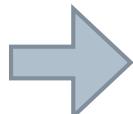
- The CSV ("Comma Separated Values") file format is often used to exchange tabular data between different applications (e.g. Excel).
- Essentially a plain text file where values are split by a comma separator. Alternatively can be tab or space separated. Often includes a first header line indicating the meaning of each column.

| Player      | Team            | Total Goals | Penalties | Home | Away |
|-------------|-----------------|-------------|-----------|------|------|
| J Vardy     | Leicester City  | 19          | 4         | 11   | 8    |
| H Kane      | Tottenham       | 16          | 4         | 7    | 9    |
| R Lukaku    | Everton         | 16          | 1         | 8    | 8    |
| O Ighalo    | Watford         | 15          | 0         | 8    | 7    |
| S Aguero    | Manchester City | 14          | 1         | 10   | 4    |
| R Mahrez    | Leicester City  | 14          | 4         | 4    | 10   |
| O Giroud    | Arsenal         | 12          | 0         | 4    | 8    |
| D Costa     | Chelsea         | 10          | 0         | 7    | 3    |
| J Defoe     | Sunderland      | 10          | 0         | 3    | 7    |
| G Wijnaldum | Newcastle Utd   | 9           | 0         | 9    | 0    |



| Player      | Team            | Total Goals | Penalties | Home | Away |
|-------------|-----------------|-------------|-----------|------|------|
| J Vardy     | Leicester City  | 19          | 4         | 11   | 8    |
| H Kane      | Tottenham       | 16          | 4         | 7    | 9    |
| R Lukaku    | Everton         | 16          | 1         | 8    | 8    |
| O Ighalo    | Watford         | 15          | 0         | 8    | 7    |
| S Aguero    | Manchester City | 14          | 1         | 10   | 4    |
| R Mahrez    | Leicester City  | 14          | 4         | 4    | 10   |
| O Giroud    | Arsenal         | 12          | 0         | 4    | 8    |
| D Costa     | Chelsea         | 10          | 0         | 7    | 3    |
| J Defoe     | Sunderland      | 10          | 0         | 3    | 7    |
| G Wijnaldum | Newcastle Utd   | 9           | 0         | 9    | 0    |

| country | latitude   | longitude  | name       |
|---------|------------|------------|------------|
| AD      | 42.546245  | 1.601554   | Andorra    |
| AL      | 41.153332  | 20.168331  | Albania    |
| AM      | 40.069099  | 45.038189  | Armenia    |
| AR      | -38.416097 | -63.616672 | Argentina  |
| AT      | 47.516231  | 14.550072  | Austria    |
| AU      | -25.274398 | 133.775136 | Australia  |
| AZ      | 40.143105  | 47.576927  | Azerbaijan |
| BB      | 13.193887  | -59.543198 | Barbados   |
| BD      | 23.684994  | 90.356331  | Bangladesh |
| BE      | 50.503887  | 4.469936   | Belgium    |
| BG      | 42.733883  | 25.48583   | Bulgaria   |



| country | latitude   | longitude  | name       |
|---------|------------|------------|------------|
| AD      | 42.546245  | 1.601554   | Andorra    |
| AL      | 41.153332  | 20.168331  | Albania    |
| AM      | 40.069099  | 45.038189  | Armenia    |
| AR      | -38.416097 | -63.616672 | Argentina  |
| AT      | 47.516231  | 14.550072  | Austria    |
| AU      | -25.274398 | 133.775136 | Australia  |
| AZ      | 40.143105  | 47.576927  | Azerbaijan |
| BB      | 13.193887  | -59.543198 | Barbados   |
| BD      | 23.684994  | 90.356331  | Bangladesh |
| BE      | 50.503887  | 4.469936   | Belgium    |
| BG      | 42.733883  | 25.48583   | Bulgaria   |

# Data Formats: Reading CSV

- A simple way to read CSV files is to use `read_csv()` in Pandas to load data into a Data Frame.

```
import pandas as pd  
df = pd.read_csv("goal_scorers.csv")
```

```
Player,Team,Total Goals,Penalties,Home,Away  
J Vardy,Leicester City,19,4,11,8  
H Kane,Tottenham,16,4,7,9  
R Lukaku,Everton,16,1,8,8  
O Ighalo,Watford,15,0,8,7  
S Aguero,Manchester City,14,1,10,4  
R Mahrez,Leicester City,14,4,4,10  
O Giroud,Arsenal,12,0,4,8  
D Costa,Chelsea,10,0,7,3  
J Defoe,Sunderland,10,0,3,7  
G Wijnaldum,Newcastle Utd,9,0,9,0
```

Input: `goal_scorers.csv`

- If the CSV contains a unique identifier column, we can use `set_index()` to set the row index.

```
In [1]: import pandas as pd  
df = pd.read_csv("goal_scorers.csv")
```

```
In [2]: df
```

```
Out[2]:
```

|   | Player      | Team            | Total Goals | Penalties | Home | Away |
|---|-------------|-----------------|-------------|-----------|------|------|
| 0 | J Vardy     | Leicester City  | 19          | 4         | 11   | 8    |
| 1 | H Kane      | Tottenham       | 16          | 4         | 7    | 9    |
| 2 | R Lukaku    | Everton         | 16          | 1         | 8    | 8    |
| 3 | O Ighalo    | Watford         | 15          | 0         | 8    | 7    |
| 4 | S Aguero    | Manchester City | 14          | 1         | 10   | 4    |
| 5 | R Mahrez    | Leicester City  | 14          | 4         | 4    | 10   |
| 6 | O Giroud    | Arsenal         | 12          | 0         | 4    | 8    |
| 7 | D Costa     | Chelsea         | 10          | 0         | 7    | 3    |
| 8 | J Defoe     | Sunderland      | 10          | 0         | 3    | 7    |
| 9 | G Wijnaldum | Newcastle Utd   | 9           | 0         | 9    | 0    |

```
In [3]: df2 = df.set_index("Player")  
df2
```

```
Out[3]:
```

| Player      | Team            | Total Goals | Penalties | Home | Away |
|-------------|-----------------|-------------|-----------|------|------|
| J Vardy     | Leicester City  | 19          | 4         | 11   | 8    |
| H Kane      | Tottenham       | 16          | 4         | 7    | 9    |
| R Lukaku    | Everton         | 16          | 1         | 8    | 8    |
| O Ighalo    | Watford         | 15          | 0         | 8    | 7    |
| S Aguero    | Manchester City | 14          | 1         | 10   | 4    |
| R Mahrez    | Leicester City  | 14          | 4         | 4    | 10   |
| O Giroud    | Arsenal         | 12          | 0         | 4    | 8    |
| D Costa     | Chelsea         | 10          | 0         | 7    | 3    |
| J Defoe     | Sunderland      | 10          | 0         | 3    | 7    |
| G Wijnaldum | Newcastle Utd   | 9           | 0         | 9    | 0    |

# Data Formats: Writing CSV

- After modifying a Data Frame (e.g. applying filtering or cleaning), we might often want to save the modified data as a new CSV file.
- With Pandas, we can use the `to_csv()` function.

```
In [5]: df3 = df2.head(5)  
df3
```

Out[5]:

| Team     |                 | Total | Goals | Penalties | Home | Away |
|----------|-----------------|-------|-------|-----------|------|------|
| Player   |                 |       |       |           |      |      |
| J Vardy  | Leicester City  | 19    | 4     | 11        | 8    |      |
| H Kane   | Tottenham       | 16    | 4     | 7         | 9    |      |
| R Lukaku | Everton         | 16    | 1     | 8         | 8    |      |
| O Ighalo | Watford         | 15    | 0     | 8         | 7    |      |
| S Aguero | Manchester City | 14    | 1     | 10        | 4    |      |

```
In [6]: df3.to_csv("5players.csv")
```



We can also specify an alternative separator character to split fields, instead of a comma.

```
Player,Team,Total  
Goals,Penalties,Home,Away  
J Vardy,Leicester City,19,4,11,8  
H Kane,Tottenham,16,4,7,9  
R Lukaku,Everton,16,1,8,8  
O Ighalo,Watford,15,0,8,7  
S Aguero,Manchester City,14,1,10,4
```

Output: 5players.csv

```
Player;Team;Total  
Goals;Penalties;Home;Away  
J Vardy;Leicester City;19;4;11;8  
H Kane;Tottenham;16;4;7;9  
R Lukaku;Everton;16;1;8;8  
O Ighalo;Watford;15;0;8;7  
S Aguero;Manchester City;14;1;10;4
```

Output: 5players-v2.csv

# Data Formats: JSON

---

- **JSON (JavaScript Object Notation)**: a lightweight format which is becoming increasingly popular for online data exchange. Based originally on the JavaScript language and (relatively) easy for humans to read and write - <http://www.json.org>
- JSON files are built from two different structures:
  1. **Object**: Collection of name/value pairs - like a Python dictionary.  
Begins with `{` and ends with `}`  
Each name is followed by `:` (colon) and the name/value pairs are separated by commas
  2. **Array**: A list of values separated by commas - like a Python list.  
Begins with `[` and ends with `]`
- Values can be of the following types:
  - String in double quotes; a number; true/false; null; object; array.

# Data Formats: JSON

---

- **Object:** Collection of name/value pairs, separated by commas - like a Python dictionary.

```
{ "firstname": "Alison", "age": 30, "car": "BMW" }
```

```
{
    "name": "School of Computer Science",
    "link": "http://www.cs.ucd.ie"
}
```

- **Array:** A list of values separated by commas - like a Python list. The values can have the same or different types.

```
[ "Ford", "BMW", "Fiat", "Mercedes" ]
```

```
[ 1123, 353, 412, 99.0, "Dublin", false, true ]
```

- We can mix Objects and Arrays:

```
{ "codes" : [5,11,31,41] }
```

An array inside an object.

```
[ { "name": "Alison" }, { "name": "John" } ]
```

Two objects in an array.

# Data Formats: JSON

---

- The outmost value in a JSON file can be an array or an object.

Top level value is an array, which itself contains two objects.

```
[  
  {  
    "id" : "978-1933988177",  
    "category" : ["book", "paperback"],  
    "name" : "Lucene in Action",  
    "author" : "Michael McCandless",  
    "genre" : "technology",  
    "price" : 30.50,  
    "pages" : 475  
  },  
  {  
    "id" : "978-1857995879",  
    "category" : ["book", "paperback"],  
    "name" : "Sophie's World",  
    "author" : "Jostein Gaarder",  
    "sequence_i" : 1,  
    "genre" : "fiction",  
    "price" : 3.07,  
    "pages" : 64  
  }  
]
```

Top level value is an object, which contains an array of objects.

```
{  
  "students": [  
    {  
      "name": "John",  
      "age": "23",  
      "city": "Dublin"  
    }, {  
      "name": "Sarah",  
      "age": "28",  
      "city": "Cork"  
    }, {  
      "name": "Peter",  
      "age": "32",  
      "city": "Galway"  
    }, {  
      "name": "Alice",  
      "age": "28",  
      "city": "London"  
    }]  
}
```

# Data Formats: JSON

- Note that whitespace does not matter in JSON, it only makes it data more readable for humans.

```
[  
  {  
    "id" : "978-1933988177",  
    "category" : ["book", "paperback"],  
    "name" : "Lucene in Action",  
    "author" : "Michael McCandless",  
    "genre" : "technology",  
    "price" : 30.50,  
    "pages" : 475  
  },  
  {  
    "id" : "978-1857995879",  
    "category" : ["book", "paperback"],  
    "name" : "Sophie's World",  
    "author" : "Jostein Gaarder",  
    "sequence_i" : 1,  
    "genre" : "fiction",  
    "price" : 3.07,  
    "pages" : 64  
  }  
]
```

```
[ { "id" : "978-1933988177", "category" : ["book", "paperback"],  
  "name" : "Lucene in Action", "author" : "Michael McCandless",  
  "genre" : "technology", "price" : 30.50, "pages" : 475 }, { "id"  
  : "978-1857995879", "category" : ["book", "paperback"], "name" :  
  "Sophie's World", "author" : "Jostein Gaarder", "sequence_i" :  
  1, "genre" : "fiction", "price" : 3.07, "pages" : 64 } ]
```

```
{"students": [{ "name": "John", "age":  
  "23", "city": "Dublin"}, { "name":  
  "Sarah", "age": "28", "city": "Cork"}, { "name":  
  "Peter", "age": "32", "city": "Galway"},  
  { "name": "Alice", "age": "28", "city":  
  "London"}]}
```

```
{  
  "students": [ {  
    "name": "John",  
    "age": "23",  
    "city": "Dublin"  
  }, {  
    "name": "Sarah",  
    "age": "28",  
    "city": "Cork"  
  }, {  
    "name": "Peter",  
    "age": "32",  
    "city": "Galway"  
  }, {  
    "name": "Alice",  
    "age": "28",  
    "city": "London"  
  }]  
}
```

# Using JSON in Python

- JSON is language-agnostic. Many different parsers available.
- The built-in module `json` provides an easy way to encode and decode data in JSON in Python. Two main functions:
  - `json.dumps()`: Turn a Python data structure into a JSON string
  - `json.loads()`: Load a Python data structure from a JSON string

```
import json
names = ["Steve", "Maria", "Jenny"]
s = json.dumps(names)
print(s)

["Steve", "Maria", "Jenny"]
```

Convert simple list into JSON

```
products = []
p1={"id":43,"name":"iPhone 5S","brand":"Apple"}
products.append(p1)
p2={"id":87,"name":"iPhone 6","brand":"Apple"}
products.append(p2)
s = json.dumps(products)
print(s)
```

```
import json
d = {"Steve":25,"Linda":40,"John":33}
s = json.dumps(d)
print(s)

{"Linda": 40, "Steve": 25, "John": 33}
```

Convert simple dictionary into JSON

```
[{"brand": "Apple", "id": 43, "name": "iPhone 5S"}, {"brand": "Apple", "id": 87, "name": "iPhone 6"}]
```

2 dictionaries nested in a list converted to JSON

# Using JSON in Python

- To translate a string containing JSON data into a Python value, pass it to the `json.loads()` function.
- Python automatically converts JSON values into variables of the appropriate type - i.e. arrays become lists, objects become dictionaries, numbers become integers or floats.

File: books.json

```
[  
 {  
   "id" : "978-1933988177",  
   "category" : [ "book", "paperback" ],  
   "name" : "Lucene in Action",  
   "author" : "Michael McCandless",  
   "genre" : "technology",  
   "price" : 30.50,  
   "pages" : 475  
,  
 {  
   "id" : "978-1857995879",  
   "category" : [ "book", "paperback" ],  
   "name" : "Sophie's World",  
   "author" : "Jostein Gaarder",  
   "genre" : "fiction",  
   "price" : 3.07,  
   "pages" : 518  
 }  
 ]
```

```
import json  
fin = open("books.json", "r")  
s = fin.read()  
fin.close()  
data = json.loads(s)  
print(data)
```

```
[{'pages': 475, 'id': '978-1933988177', 'genre':  
 'technology', 'author': 'Michael McCandless', 'price': 30.5,  
 'category': [ 'book', 'paperback' ], 'name': 'Lucene in  
 Action'}, {'pages': 518, 'genre': 'fiction', 'id':  
 '978-1857995879', 'author': 'Jostein Gaarder', 'price':  
 3.07, 'category': [ 'book', 'paperback' ], 'name': 'Sophie's  
 World'}]
```

Variable is now a Python list, containing two dictionaries.

# Data Formats: XML

---

- XML: a **markup** language used to describe data in a structured way using **tags**. Tags are not predefined, but are user defined. Many schemas exist that recommend standardised usage of tags.
- Tags must be opened `<tag>` and closed `</tag>`.
- Tags can contain values as plain text or other nested tags.

```
<note>
  <to>Alice</to>
  <from>John</from>
  <subject>Reminder</subject>
  <body>Remember to buy milk!</body>
</note>
```

Tags are opened and closed  
e.g. `<note>...</note>`

We can "nest" tags inside  
other tags to create a  
hierarchical structure.

```
<notes>
  <note>
    <to>Alice</to>
    <from>John</from>
    <subject>Reminder</subject>
    <body>Remember to buy milk!</body>
  </note>
  <note>
    <to>John</to>
    <from>Alice</from>
    <subject>Shopping</subject>
    <body>I forgot the milk!</body>
  </note>
</notes>
```

# Data Formats: XML

- Tags can also have zero or more name/value **attribute** pairs, which add extra information to a tag.

XML files often (but not always) contain a header line

Tags are opened and closed  
e.g. <book>...</book>

Values can be contained within tags.

Tags can have attributes  
e.g. currency="eur"

File: products.xml

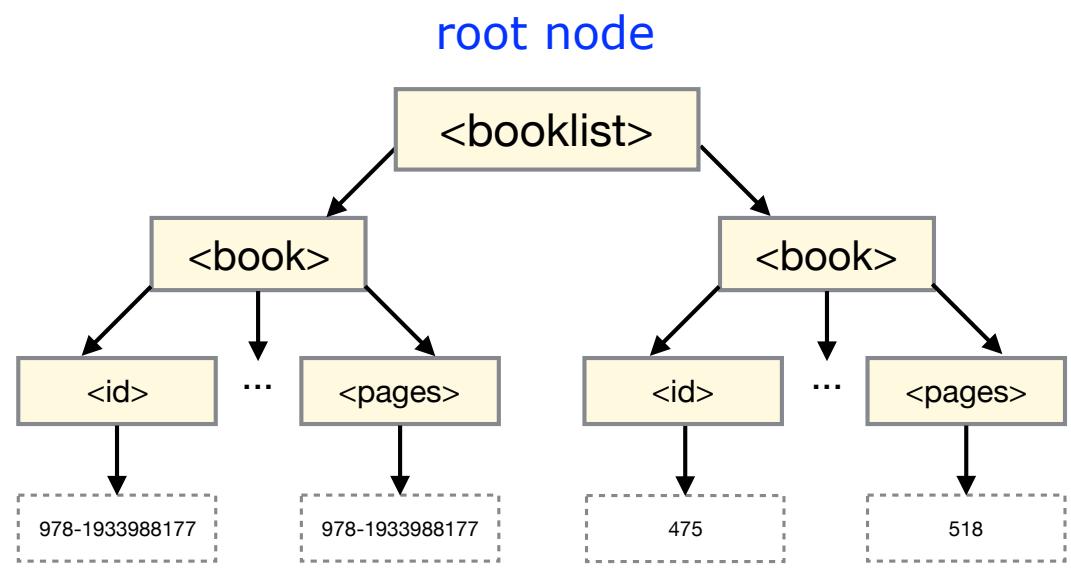
```
<?xml version="1.0" encoding="UTF-8" ?>
<booklist>
  <book>
    <id>978-1933988177</id>
    <category>paperback</category>
    <name>Lucene in Action</name>
    <author>Michael McCandless</author>
    <genre>technology</genre>
    <price currency="eur">30.5</price>
    <pages>475</pages>
  </book>
  <book>
    <id>978-1857995879</id>
    <category>paperback</category>
    <name>Sophie's World</name>
    <author>Jostein Gaarder</author>
    <genre>fiction</genre>
    <price currency="eur">3.07</price>
    <pages>518</pages>
  </book>
</booklist>
```

# Data Formats: XML

- XML is an inherently hierarchical data format, and the most natural way to represent it is with a tree with **nodes** at different levels.
- The document itself is the **root node** of the tree. Other nodes are child nodes. If they contain nodes themselves, they are parent nodes. The lowest level of the tree contains **leaf nodes**.

File: products.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<booklist>
  <book>
    <id>978-1933988177</id>
    <category>paperback</category>
    <name>Lucene in Action</name>
    <author>Michael McCandless</author>
    <genre>technology</genre>
    <price>30.5</price>
    <pages>475</pages>
  </book>
  <book>
    <id>978-1857995879</id>
    <category>paperback</category>
    <name>Sophie's World</name>
    <author>Jostein Gaarder</author>
    <genre>fiction</genre>
    <price>3.07</price>
    <pages>518</pages>
  </book>
</booklist>
```



leaf nodes of this document contain values

# Using XML in Python

- XML is a widely-adopted format. Python includes several built-in modules for parsing XML data.
- The `xml.etree.ElementTree` module can be used to extract data from a simple XML file based on its tree structure.

File: products.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<booklist>
    <book>
        <id>978-1933988177</id>
        <category>paperback</category>
        <name>Lucene in Action</name>
        <author>Michael McCandless</author>
        <genre>technology</genre>
        <price>30.5</price>
        <pages>475</pages>
    </book>
    <book>
        <id>978-1857995879</id>
        <category>paperback</category>
        <name>Sophie's World</name>
        <author>Jostein Gaarder</author>
        <genre>fiction</genre>
        <price>3.07</price>
        <pages>518</pages>
    </book>
</booklist>
```

Import the module and parse the XML file

```
import xml.etree.ElementTree
tree = xml.etree.ElementTree.parse("products.xml")
```

Find all tags that we are interested in (book)  
Then find text in nested child tags (name & author)

```
for book in tree.iterfind("book"):
    n = book.findtext("name")
    a = book.findtext("author")
    print("%s by %s" % (n,a))
```

```
Lucene in Action by Michael McCandless
Sophie's World by Jostein Gaarder
```

# Data Formats: HTML

- HTML is also a markup language similar to XML. Key differences:
  - XML describes data, HTML describes data and presentation.
  - In practice HTML is usually badly-written and invalid.

File: products.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<booklist>
  <book>
    <id>978-1933988177</id>
    <category>paperback</category>
    <name>Lucene in Action</name>
    <author>Michael McCandless</author>
    <genre>technology</genre>
    <price>30.5</price>
    <pages>475</pages>
  </book>
  <book>
    <id>978-1857995879</id>
    <category>paperback</category>
    <name>Sophie's World</name>
    <author>Jostein Gaarder</author>
    <genre>fiction</genre>
    <price>3.07</price>
    <pages>518</pages>
  </book>
</booklist>
```

File: products.html (in browser)

## Book List

### Lucene in Action

- *Author:* Michael McCandless
- *Genre:* Technology
- *Price:* €30.50
- Paperback, 475 pages

### Sophie's World

- *Author:* Sophie's World
- *Genre:* Fiction
- *Price:* €3.07
- Paperback, 518 pages

# Data Formats: HTML

- HTML also consists of opening `<tag>` and closing `</tag>`, with other tags and tested enclosed within tags.
- Basic page structure:

`<html>...</html>`: Root tag

`<head>...</head>`: Page metadata

`<body>...</body>`: Page content

## Book List

### Lucene in Action

- *Author*: Michael McCandless
- *Genre*: Technology
- *Price*: €30.50
- Paperback, 475 pages

### Sophie's World

- *Author*: Sophie's World
- *Genre*: Fiction
- *Price*: €3.07
- Paperback, 518 pages

#### File: products.html

```
<html>
  <head>
    <title>Book List</title>
  </head>
  <body>
    <h2>Book List</h2>
    <hr>
    <h3><font color="red">Lucene in Action</font></h3>
    <ul>
      <li><i>Author:</i> Michael McCandless</li>
      <li><i>Genre:</i> Technology</genre></li>
      <li><i>Price:</i> &euro;30.50</li>
      <li>Paperback, 475 pages</li>
    </ul>
    <hr>
    <h3><font color="blue">Sophie's World</font></h3>
    <ul>
      <li><i>Author:</i> Sophie's World</li>
      <li><i>Genre:</i> Fiction</genre></li>
      <li><i>Price:</i> &euro;3.07</li>
      <li>Paperback, 518 pages</li>
    </ul>
    <hr>
  </body>
</html>
```

# Web Scraping

---

- **Web scraping:** Technique used to extract data from web sites using a tool that acts as a web browser.
- **Basic steps:**
  1. *Data identification:* Identify and study the structure of the web pages containing the required data.
  2. *Data collection:* Download these web pages in the same way as a web browser does. This may even simulate a user logging in to obtain access.
  3. *Data extraction:* Parse web pages to extract data.
  4. *Data cleaning:* Clean and reformat the data to make it usable.
- **The "rules" of web scraping:**
  - Check a site's terms and conditions before you scrape their pages.
  - **Do not** hammer a site with too many automated requests.
  - Sites often change their layout, so scrapers often break and need to be re-written.

# Web Data Identification

**Example:** Want to extract list of UCD college names from the page

[https://www.it.kmitl.ac.th/~teerapong/resources/ds4biz/week4/sample\\_web/sample.html](https://www.it.kmitl.ac.th/~teerapong/resources/ds4biz/week4/sample_web/sample.html)

## Simple.html

```
...  
<div id="content">  
  <h3><a href="http://www.ceir.kmitl.ac.th">College of Educational Innovation Research</a></h3>  
  <h3><a href="http://www.music-engineering.kmitl.ac.th/">Institute of Music Science and Engineering</a></h3>  
  <h3><a href="http://www.kosen.kmitl.ac.th/">KOSEN-KMITL</a></h3>  
  <h3><a href="http://www.chumphon.kmitl.ac.th/">KMITL Prince of Chumphon Campus </a></h3>  
  <h3><a href="http://engineer.kmitl.ac.th/">Faculty of Engineering</a></h3>  
  <h3><a href="http://www.arch.kmitl.ac.th/">Faculty of Architecture</a></h3>  
  <h3><a href="http://www.science.kmitl.ac.th/main.php">Faculty of Science</a></h3>  
  <h3><a href="http://www.ietech.kmitl.ac.th">Faculty of Industrial Education and Technology</a></h3>  
  <h3><a href="http://www.agri.kmitl.ac.th/AgrITH/">Faculty of Agricultural Technology</a></h3>  
  <h3><a href="http://www.it.kmitl.ac.th/">Faculty of Information Technology</a></h3>  
  <h3><a href="http://www.agroind.kmitl.ac.th/th/home-agro">Faculty of Agro-Industry</a></h3>  
  <h3><a href="http://www.fam.kmitl.ac.th/">Faculty of Administration and Management</a></h3>  
  <h3><a href="http://fla.kmitl.ac.th/th/">Faculty of Liberal Arts</a></h3>  
  <h3><a href="http://www.ic.kmitl.ac.th/index.php">International College</a></h3>  
  <h3><a href="http://www.nano.kmitl.ac.th/index.php">College of Nanotechnology</a></h3>  
  <h3><a href="http://www.ami.kmitl.ac.th/">College of Advanced Manufacturing Innovation</a></h3>  
  <h3><a href="http://md.kmitl.ac.th/">Faculty of Medicine</a></h3>  
  <h3><a href="http://aviation.kmitl.ac.th/">International Academy of Aviation Industry</a></h3>  
  <h3><a href="http://www.kmids.ac.th/en/">King Mongkut's International Demonstration School</a></h3>  
</div>  
</div>
```

View HTML source in browser to identify part of page with data of interest



**King Mongkut's Institute of Technology Ladkrabang**  
**KMITL**

**Faculty/College in KMITL**

- College of Educational Innovation Research**
- Institute of Music Science and Engineering**
- KOSEN-KMITL**
- KMITL Prince of Chumphon Campus**
- Faculty of Engineering**
- Faculty of Architecture**
- Faculty of Science**
- Faculty of Industrial Education and Technology**
- Faculty of Agricultural Technology**
- Faculty of Information Technology**
- Faculty of Agro-Industry**
- Faculty of Administration and Management**
- Faculty of Liberal Arts**
- International College**
- College of Nanotechnology**
- College of Advanced Manufacturing Innovation**
- Faculty of Medicine**
- International Academy of Aviation Industry**
- King Mongkut's International Demonstration School**

# Web Data Collection

- The built-in Python `urllib.request` module has functions which help in downloading content from HTTP URLs using minimal code:

```
import urllib.request
link = "https://www.it.kmitl.ac.th/~teerapong/resources/ds4biz/week4/sample_web/sample.html"
response = urllib.request.urlopen(link)
html = response.read().decode()
```

```
lines = html.strip().split("\n")
for l in lines:
    print(l)

<html>
    <head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title> KMITL Faculty/College</title>
        <link href=".css/sample.css" rel="stylesheet" type="text/css">
    </head>
    <body>
        <div class="topbar">
            <div id="top-logo2f">
                
            </div>
            <div id="top-text2f">
                <h1>King Mongkut's Institute of Technology Ladkrabang</h1>
            </div>
        </div>

        <div id="main" style="margin: 20px;">
            <h1>Faculty/College in KMITL</h1>
            <div id="content">
                <h3><a href="http://www.ceir.kmitl.ac.th">College of Educational Innovation Research</a></h3>
                <h3><a href="http://www.music-engineering.kmitl.ac.th/">Institute of Music Science and Engineering</a></h3>
                <h3><a href="http://www.kosen.kmitl.ac.th/">KOSEN-KMITL</a></h3>
                <h3><a href="http://www.chumphon.kmitl.ac.th">KMITL Prince of Chumphon Campus </a></h3>
                <h3><a href="http://engineer.kmitl.ac.th/">Faculty of Engineering</a></h3>
                <h3><a href="http://www.arch.kmitl.ac.th/">Faculty of Architecture</a></h3>
                <h3><a href="http://www.science.kmitl.ac.th/main.php">Faculty of Science</a></h3>
                <h3><a href="http://www.ietech.kmitl.ac.th/">Faculty of Industrial Education and Technology</a></h3>
                <h3><a href="http://www.agri.kmitl.ac.th/AgriTH/">Faculty of Agricultural Technology</a></h3>
                <h3><a href="http://www.it.kmitl.ac.th/">Faculty of Information Technology</a></h3>
                <h3><a href="http://www.agroind.kmitl.ac.th/th/home-agro">Faculty of Agro-Industry</a></h3>
                <h3><a href="http://www.fam.kmitl.ac.th/">Faculty of Administration and Management</a></h3>
                <h3><a href="http://fla.kmitl.ac.th/th/">Faculty of Liberal Arts</a></h3>
                <h3><a href="http://www.ic.kmitl.ac.th/index.php">International College</a></h3>
                <h3><a href="http://www.nano.kmitl.ac.th/index.php">College of Nanotechnology</a></h3>
                <h3><a href="http://www.ami.kmitl.ac.th/">College of Advanced Manufacturing Innovation</a></h3>
                <h3><a href="http://md.kmitl.ac.th/">Faculty of Medicine</a></h3>
                <h3><a href="http://aviation.kmitl.ac.th">International Academy of Aviation Industry</a></h3>
                <h3><a href="http://www.kmids.ac.th/en/">King Mongkut's International Demonstration School</a></h3>
            </div>
        </div>
    </body>
</html>
```

Fetch the HTML code  
from the web page

Split into lines and  
print each line

# Web Data Collection

- When we try to retrieve a URL, we will get a status code indicating if the result was successful.

```
response = urllib.request.urlopen(link)
print(response.code)
```

200

- The most common status codes are:

**200**: The request succeeded, and the resource is returned.

**404**: The requested resource does not exist.

**301/302/303**: The resource has moved to another URL.

**500**: An unexpected error happened on the server side.

- When using **urllib.request**, a 404 response or an invalid address will raise a **HTTPError** / **URLLError** exception. This should be handled using a try/catch block.

```
response = urllib.request.urlopen("https://somemissinglink.it.kmitl.ac.th")
```

Traceback (most recent call last):  
...  
urllib.error.HTTPError: HTTP Error 404: Not Found

# Web Data Extraction

- Many ways to parse HTML pages in Python. The third-party *BeautifulSoup* package is useful for working with badly written HTML pages:  
<http://www.crummy.com/software/BeautifulSoup>

```
<html>
<head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title> KMITL Faculty/College</title>
<link href=".css/sample.css" rel="stylesheet" type="text/css">
</head>
<body>
<div class="topbar">
  <div id="top-logo2f">
    
  </div>
  <div id="top-text2f">
    <h1>King Mongkut's Institute of Technology Ladkrabang</h1>
  </div>
</div>

<div id="main" style="margin: 20px;">
  .
  .
  .
  <h3><a href="http://www.ceir.kmitl.ac.th">College of Educational Innovation Research</a></h3>
  <h3><a href="http://www.music-engineering.kmitl.ac.th/">Institute of Music Science and Engineering</a></h3>
  <h3><a href="http://www.kosen.kmitl.ac.th/">KOSEN-KMITL</a></h3>
  <h3><a href="http://www.chumphon.kmitl.ac.th/">KMITL Prince of Chumphon Campus </a></h3>
  <h3><a href="http://engineer.kmitl.ac.th/">Faculty of Engineering</a></h3>
  <h3><a href="http://www.arch.kmitl.ac.th/">Faculty of Architecture</a></h3>
  <h3><a href="http://www.science.kmitl.ac.th/main.php">Faculty of Science</a></h3>
  <h3><a href="http://www.iitech.kmitl.ac.th">Faculty of Industrial Education and Technology</a></h3>
  <h3><a href="http://www.agri.kmitl.ac.th/AgritH/">Faculty of Agricultural Technology</a></h3>
  <h3><a href="http://www.it.kmitl.ac.th/">Faculty of Information Technology</a></h3>
  <h3><a href="http://www.agroind.kmitl.ac.th/th/home-agro">Faculty of Agro-Industry</a></h3>
  <h3><a href="http://www.fam.kmitl.ac.th/">Faculty of Administration and Management</a></h3>
  <h3><a href="http://fla.kmitl.ac.th/">Faculty of Liberal Arts</a></h3>
  <h3><a href="http://www.ic.kmitl.ac.th/index.php">International College</a></h3>
  <h3><a href="http://www.nano.kmitl.ac.th/index.php">College of Nanotechnology</a></h3>
  <h3><a href="http://www.ami.kmitl.ac.th/">College of Advanced Manufacturing Innovation</a></h3>
  <h3><a href="http://md.kmitl.ac.th/">Faculty of Medicine</a></h3>
  <h3><a href="http://aviation.kmitl.ac.th/">International Academy of Aviation Industry</a></h3>
  <h3><a href="http://www.kmids.ac.th/en/">King Mongkut's International Demonstration School</a></h3>
</div>
</div>
</body>
</html>
```

```
import bs4
parser = bs4.BeautifulSoup(html,"html.parser")
for match in parser.find_all("h3"):
    text = match.get_text()
    print(text)
```

In our example, we want to extract the text in the `<h3>` tags.

We can use BeautifulSoup to find all these tags and get the text between them.

```
...
Faculty of Engineering
Faculty of Architecture
Faculty of Science
Faculty of Industrial Education and Technology
Faculty of Agricultural Technology
Faculty of Information Technology
...
...
```

# Web Scraping: Other Formats

---

- Sometimes web data will be in other non-HTML formats, where an alternative parser may be required to extract data.
- Online CSV data:

e.g. [https://www.it.kmitl.ac.th/~teerapong/resources/ds4biz/week4/goal\\_scorers.csv](https://www.it.kmitl.ac.th/~teerapong/resources/ds4biz/week4/goal_scorers.csv)

Pandas can download and parse CSV data from a remotely hosted file and store it in a data frame:

```
import pandas as pd
link = "https://www.it.kmitl.ac.th/~teerapong/resources/ds4biz/week4/goal_scorers.csv"
df = pd.read_csv( link, index_col="Player" )
df.head(5)
```

|          | Team            | Total Goals | Penalties | Home Goals | Away Goals |
|----------|-----------------|-------------|-----------|------------|------------|
| Player   |                 |             |           |            |            |
| J Vardy  | Leicester City  | 19          | 4         | 11         | 8          |
| H Kane   | Tottenham       | 16          | 4         | 7          | 9          |
| R Lukaku | Everton         | 16          | 1         | 8          | 8          |
| O Ighalo | Watford         | 15          | 0         | 8          | 7          |
| S Aguero | Manchester City | 14          | 1         | 10         | 4          |

# Web Scraping: Other Formats

- Sometimes web data will be in other non-HTML formats, where an alternative parser may be required to extract data.
- Online JSON data:

e.g. <https://www.it.kmitl.ac.th/~teerapong/resources/ds4biz/week4/books.json>

Pandas can also download and parse JSON data from a remotely hosted file and store it in a data frame:

```
import pandas as pd
link = "https://www.it.kmitl.ac.th/~teerapong/resources/ds4biz/week4/books.json"
df = pd.read_json(link, orient="records")
df.head(5)
```

|   | ISBN       | book_id   | language | rating | title   | year |
|---|------------|-----------|----------|--------|---|------|
| 0 |            | 13585350  | eng      | 3      | The World Treasury of Science Fiction         | 1989 |
| 1 | 1936594056 | 124205572 | eng      | 4      | The War of the Worlds                         | 2013 |
| 2 | 1439149038 | 127360065 | eng      | 2      | Under the Dome: A Novel                       | 2013 |
| 3 | 0345453743 | 13908800  | eng      | 5      | The Ultimate Hitchhiker's Guide to the Galaxy | 2002 |
| 4 | 1476764832 | 123734934 | eng      | 5      | The Time Traveler's Wife                      | 2014 |

Tell Pandas we are expecting an Array containing Objects, where each Object will be a row in the data frame.

See [http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read\\_json.html](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_json.html)

# Web Scraping: Other Formats

- A common online use of XML is for syndication, where frequently updated **news feeds** list the latest articles on a news site or forum.  
e.g. <http://www.rte.ie/news/rss/business-headlines.xml>

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet title="XSL_formatting" type="text/xsl" href="/rss/rss_convert.xsl"?>
<rss version="2.0" xmlns:atom="http://www.w3.org/2005/Atom">
    <channel>
        <title>RTÉ News - Business Headlines</title>
        <link>http://www.rte.ie/news/business/</link>
        <item>
            <title>Drumm dismisses legal team</title>
            <link>http://www.rte.ie/news/2016/0223/770341-david-drumm/</link>
            <description>Former Anglo Irish Bank CEO David Drumm has dismissed his legal team in his continued appeal against a failed bid to be declared bankrupt in the US.</description>
            <pubDate>Wed, 24 Feb 2016 07:34:37 +0000</pubDate>
            <guid isPermaLink="true">http://www.rte.ie/news/2016/0223/770341-david-drumm/</guid>
            <enclosure url="http://img.rasset.ie/000b6adf-144.jpg" type="image/jpeg" length="4094" />
        </item>
        <item>
            <title>Oil prices extend losses on Saudi comments</title>
            <link>http://www.rte.ie/news/business/2016/0224/770390-oil-prices/</link>
            <description>Oil prices fell further in Asia trade today after OPEC kingpin Saudi Arabia shut the door on an output cut to ease the global crude supply glut, touting only a freeze in production.</description>
            <pubDate>Wed, 24 Feb 2016 07:41:19 +0000</pubDate>
            <guid isPermaLink="true">http://www.rte.ie/news/business/2016/0224/770390-oil-prices/</guid>
            <enclosure url="http://img.rasset.ie/00036f00-144.jpg" type="image/jpeg" length="4094" />
        </item>
    </channel>
</rss>
```

- We can use the **xml.etree.ElementTree** module to parse this type of web data in XML format.

# Web APIs

---

- Instead of manually scraping HTML data, some web sites provide a convenient "official" way of retrieving their data.
- **Web application programming interface (API)**: A set of HTTP request messages with a definition of the expected structure of the response. API calls can return responses in a variety of formats. Sometimes plain text, but more commonly JSON or XML.
- Web APIs have an **endpoint** - a URL from which we retrieve the data.
- Many Open APIs exist, others are private/paid/proprietary.
- Some Open APIs have third-party Python packages which provide functions that "wrap" the queries and responses. In other cases, HTTP calls will need to be made and responses parsed manually.
- API Restrictions:
  - Many APIs have rate limits. Can only make limited number of calls from a given IP address/account, in a fixed amount of time.
  - Some APIs require authentication - i.e. first need to register a user account, retrieve an authentication token/password.

# Web APIs: Data Collection

- Wikipedia provides a simple Web API for retrieving page content in JSON format, endpoint is <https://en.wikipedia.org/w/api.php>
- Example query URL for the Wikipedia page "King\_Mongkut%27s\_Institute\_of\_Technology\_Ladkrabang"

```
https://en.wikipedia.org/w/api.php?  
format=json&action=query&prop=extracts&exintro=true&  
titles=King_Mongkut%27s_Institute_of_Technology_Ladkrabang
```

Not logged in Talk Contributions Create account Log in

Read Edit View history Search Wikipedia

Main page Article Talk King Mongkut's Institute of Technology Ladkrabang From Wikipedia, the free encyclopedia Coordinates: 13°43'36"N 100°46'48"E

This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed.

Find sources: "King Mongkut's Institute of Technology Ladkrabang" – news · newspapers · books · scholar · JSTOR (May 2012)

King Mongkut's Institute of Technology Ladkrabang (KMITL or KMIT Ladkrabang for short) is a research and educational institution in Thailand. It is approximately 30 km east of the center of Bangkok in Lat Krabang District, Bangkok, Thailand, and has seven faculties: engineering, architecture, technology, science, industrial education, agricultural industry, information technology, and liberal arts.

Contents [hide]

- 1 History
- 2 Faculties and departments
- 3 Centers and services
  - 3.1 Research Center for Communication and Information Technology (ReCCiT)
  - 3.2 Industry/University Cooperative Research Center in Data Storage Technology and Applications
- 4 Campuses
- 5 Notable faculty and alumni
- 6 Sister Universities
- 7 External links

History [edit]

KMITL was founded in 1960 in Nonthaburi province as a telecommunications training center under the technical support of the Japanese government. It later became the Nonthaburi Institute of Telecommunications. After moving to a new location at Lat Krabang near Suvarnabhumi Airport, the name was changed to King Mongkut's Institute of Technology Ladkrabang.

{"batchcomplete": "",  
"warnings": {"extracts": {"\*": "HTML may be malformed and/or unbalanced and may omit inline images. Use at your own risk. Known problems are listed at https://www.mediawiki.org/wiki/Extension:TextExtracts#Caveats."}},  
"query": {"normalized": [{"from": "King\_Mongkut's\_Institute\_of\_Technology\_Ladkrabang",  
"to": "King Mongkut's Institute of Technology Ladkrabang"}]},  
"pages": {"1232312": {"pageid": 1232312,  
"ns": 0,  
"title": "King Mongkut's Institute of Technology Ladkrabang",  
"extract": "

**King Mongkut's Institute of Technology Ladkrabang** (**KMITL** or **KMIT Ladkrabang** for short) is a research and educational institution in Thailand. It is located approximately 30 km east of the center of Bangkok in Lat Krabang District, Bangkok, Thailand, and has seven faculties: engineering, architecture, technology, science, industrial education, agricultural industry, information technology, and liberal arts.

"}}}

# Web APIs: Twitter

---

- Twitter provides a comprehensive Web API for retrieving public tweets, user profiles, follower links etc. in JSON format.  
<https://developer.twitter.com/en/docs.html>
- Access requires a Twitter account and OAuth authentication details:  
<https://dev.twitter.com/oauth>
- Since the JSON data returned is often long and complex, a number of wrapper packages exist for accessing the API through Python.
- Tweepy Package: <http://www.tweepy.org>

```
import tweepy

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)

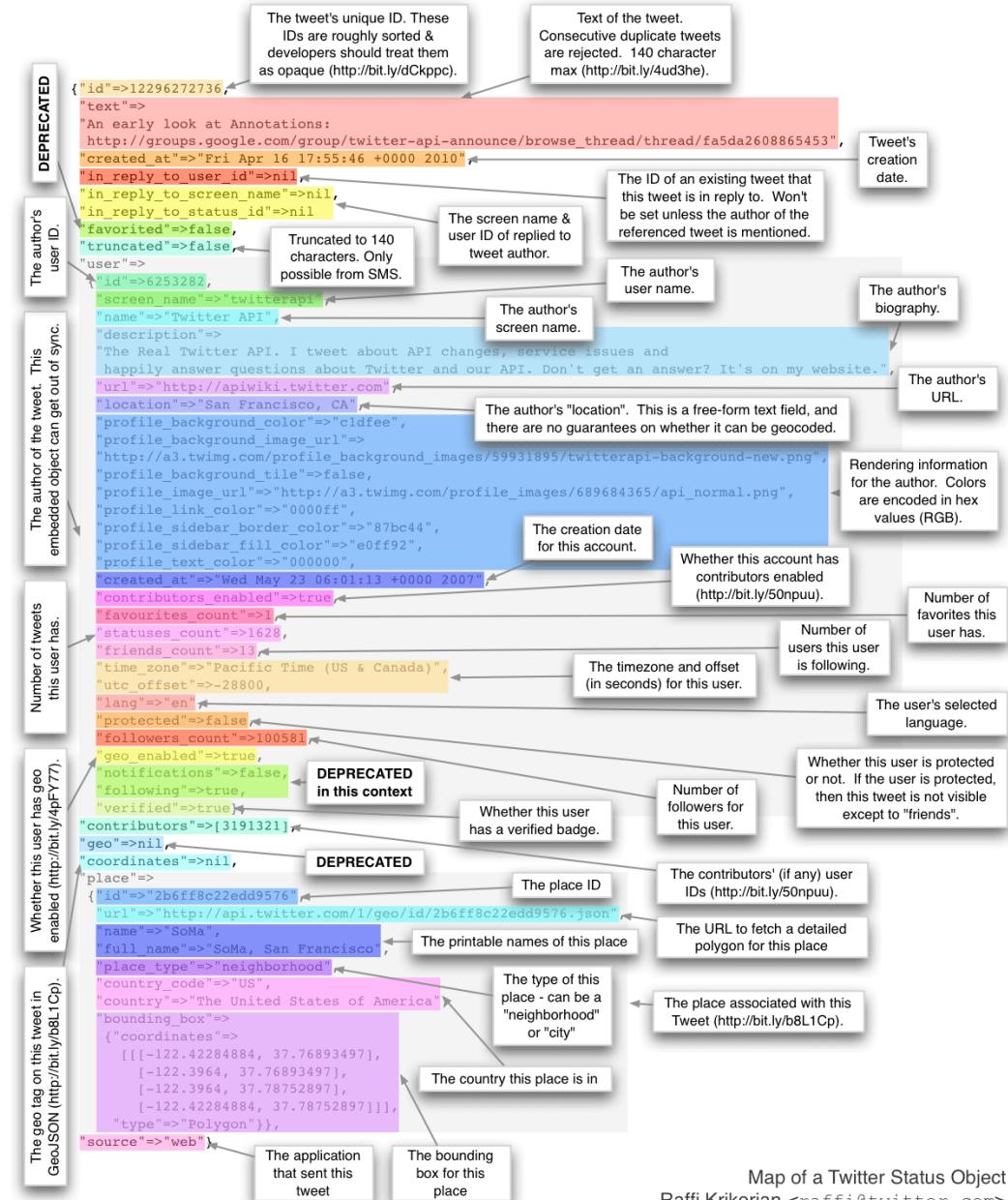
timeline_tweets = api.home_timeline()
for tweet in timeline_tweets:
    print(tweet.text)
```

2 keys and 2 "secret" values are used to authenticate your account

Print all tweets from your home timeline

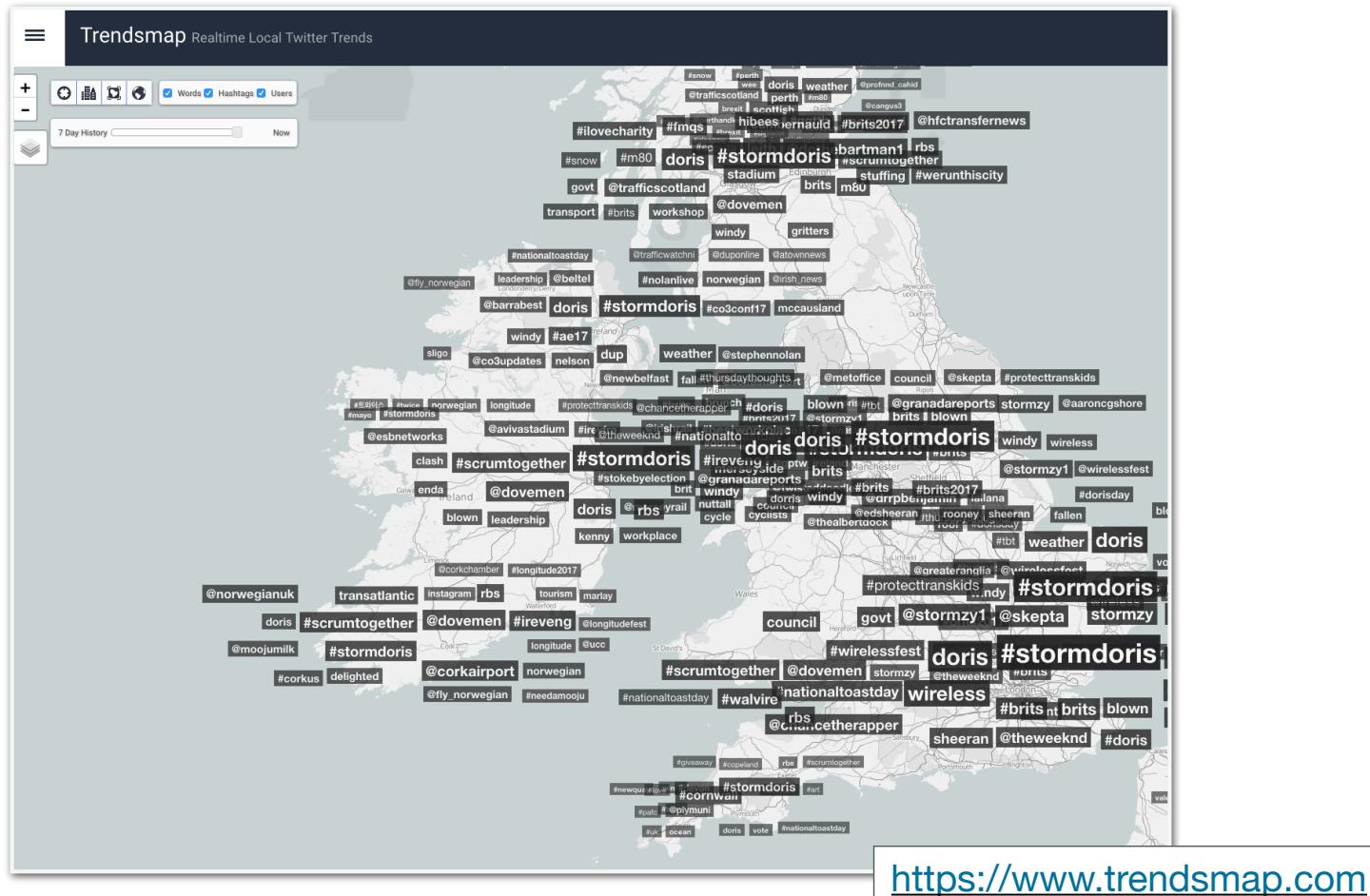
# Web APIs: Twitter

- Twitter API requests are made to end points on [twitter.com](http://twitter.com)
- All results from the API are returned in JSON format.
- Even a result for a single tweet can contain a significant amount of data...



# API Mashups

- Web mashup: web application that uses content from more than one source to create a single new service.
- Often use open APIs and data sources to produce richer results that were not the original reason for producing the raw source data.



# API Resources

---

## Sample Public APIs:

- Google Translate API - <https://developers.google.com/translate/>
- World Bank APIs - <http://data.worldbank.org/developers>
- Guardian Content API - <http://open-platform.theguardian.com/>
- New York Times Developer Network - <http://developer.nytimes.com/>
- Weather Underground API - <https://www.wunderground.com/weather/api/>
- Flickr API - <https://www.flickr.com/services/api/>
- Reddit API - <https://www.reddit.com/dev/api/>
- Foursquare API - <https://developer.foursquare.com/>
- OpenStreetMap API - <https://www.openstreetmap.org>

## Sample Web API Lists:

- [https://en.wikipedia.org/wiki/List\\_of\\_open\\_APIs](https://en.wikipedia.org/wiki/List_of_open_APIs)
- <https://github.com/toddmotto/public-apis>
- <http://www.programmableweb.com/apis/directory>
- <http://www.computersciencezone.org/50-most-useful-apis-for-developers>

# Data Science for Business

## *Learning by Yourself – Extra Pandas Tricks*

Asst. Prof. Teerapong Leelanupab (Ph.D.)  
Faculty of Information Technology  
King Mongkut's Institute of Technology Ladkrabang (KMITL)



Week 4.2

# Extra Pandas Tricks from Data School

---

Remember!!! I cannot teach you all in class. I also still need to keep learning.

There are **5** Jupyter Notebooks to practice and learn much more about Pandas.

The original source is in a Github repos given below:

- Github: <https://github.com/justmarkham/pandas-videos.git>

I already pulled them out and include them into our Week 4 lab set.

# Data Science for Business

## *Assignment 1 – 20 marks*

Asst. Prof. Teerapong Leelanupab (Ph.D.)  
Faculty of Information Technology  
King Mongkut's Institute of Technology Ladkrabang (KMITL)



Week 4.3

# Assignment 1 (20 marks)

---

- Collecting Data from API, Data Preparation and Visualization
- See the details in *the Assignment 1 Sheet*.

# Assignment 1 – Marking Guideline (1)

---

## Marking Guidelines (100%):

### **1) Data collection & storage: 20%**

Marks awarded for:

- Correct use of one or more Web API(s), and storage in an appropriate format. Takes into account the complexity of the API, the use of multiple collection runs (where required), and the use of multiple related APIs.

# Assignment 1 – Marking Guideline (2)

---

## 2) Data pre-processing & analysis: 60%

Marks awarded for:

- Loading the data into an appropriate representation for analysis.
- Application of any necessary pre-processing steps to improve the representation of the data (e.g. missing value imputation, filtering rows, filtering features, merging/ grouping features, deriving new features, integrating data from multiple APIs if required).
- Exploration, summarization, and characterization of the cleaned dataset. This should include the use of data visualization and tables where appropriate.
- Interpretations and insights made about the dataset, based on the above. This includes suggestions for further analysis.

# Assignment 1 – Marking Guideline (3)

---

## 3) Code quality & explanation text: 20%

Marks awarded for:

- Use of Markdown cells to explain each step of the process, from data collection to interpretation.
- Readability and clarity of the code. This includes use of comments, though they should not be used excessively.
- Complexity of the code, and appropriate use of Python packages. Note that code performance/speed will not be taken into account

# Assignment 1 – Penalties for Plagiarism:

---

## Penalties for Plagiarism:

- This work is an individual assignment. Any suspect evidence of plagiarism will initially result in a **O** mark of this assignment.
- If a clear evidence of plagiarism is found, the grade of this course is **F**. In addition, the case will be reported to the Faculty and KMITL Disciplinary committees.

# **Assignment 1 – Penalties for Late Submission:**

---

## **Penalties for Late Submission:**

- 1-5 days late: 20% deduction from overall mark
- 6-10 days late: 40% deduction from overall mark
- Assignments will not be accepted after 10 days without an extenuating circumstances form and/or a medical certificate.