

Introduction to Data Science Course

Data Collection

Le Ngoc Thanh
Inthanh@fit.hcmus.edu.vn
Department of Computer Science

Contents

- ◎ **Review Data Science Process**
- ◎ Data Collection from Website
- ◎ Data Preprocessing
- ◎ Working with Dynamic Webpage
- ◎ API Data Collection

Data Science Process

- ◎ Give the question to answer
- ◎ Collecting data
- ◎ Data Discovery & preprocessing to obtain data that can be analyzed
- ◎ Data analysis (in statistics, visualizations, machine learning)
 - → answers (hypotheses) for the question
- ◎ Evaluation
- ◎ Decision Making

From one step you will probably need to go back to the previous steps to readjust, which will probably need to go to the retreat a number of time.

Required attitude: calm, intuitive.

Tools to know how to use: Python and libraries, Jupyter Notebook.

Collecting data

◎ General notes when collecting data

- Is the data correct and sufficient to answer the question?
Garbage in → garbage out
- Is collecting such data valid? Does it affect others?

◎ Ways to collect Data

- Data is **available** in **company, organization**: ok, use it
- Data is **available** but **out there** (online)
 - ◎ Pre-packaged data (file csv, excel, ...): download
 - ◎ Data provided through the website's API: use API
 - ◎ Data is on the site but no API: parse HTML
- The data is **not yet available**: created by yourself in ways such as conducting surveys, using sensor devices, ...

Scope
of the
course

Ask Question

How is the recruitment situation of the Data Science in Vietnam now?

- Initially, the question was often broad and vague
- At a later time, it will go back to this step a number of times to adjust the question to be more clear and more specific.

Collecting data: Planned

Q: Where to collect data?

A: On recruitment sites in Vietnam

Q: What are the recruitment sites in Vietnam?

A: Ask Google ...

→ <http://www.vietnamworks.com/>, <http://careerbuilder.vn/>, ...

Q: For each job page, looking for recruitment with which keywords?

A: “Khoa học dữ liệu”, “data science”, “data scientist”, ...

Q: For each job page, after searching with a certain keyword, how do I get the recruitment information?

A: On each recruitment, copy-paste information to take into file 😞

Collecting data: Planned

Q: Where to collect data?

A: On recruitment sites in Vietnam

Q: What are the recruitment sites in Vietnam?

A: Ask Google ...

→ <http://www.vietnamworks.com/>, <http://careerbuilder.vn/>, ...

Q: For each job page, looking for recruitment with which keywords?

A: “Khoa học dữ liệu”, “data science”, “data scientist”, ...

Q: For each job page, after searching with a certain keyword, how do I get the recruitment information?

A: Write a program that **automatically** parse HTML, get the information to retrieve and write down the file 😊

Q: After you've got data from different job pages, or from the same page but with different keywords, **how do you merge these data?**

A: ...


Contents

- ◎ Review Data Science Process
- ◎ **Data Collection from Website**
- ◎ Data Preprocessing
- ◎ Working with Dynamic Webpage
- ◎ API Data Collection

Collecting data from the CareerBuilder site with the keyword "data scientist"

data scientist x Tất cả ngành nghề v Tất cả địa điểm v **Q Tìm Ngay**

Tìm việc làm data scientist 10 Chính xác v




Data Scientist
Heineken Vietnam Brewery

📍 Hồ Chí Minh

\$ Lương: Cạnh tranh

Ngày cập nhật: 08/03/2020



Data Scientist
Công Ty Cổ Phần Phần Mềm Citigo

📍 Hà Nội

\$ Lương: 30 Tr - 40 Tr VND

Ngày cập nhật: 17/02/2020

ĐỊA ĐIỂM

- ☐ Hà Nội 6
- ☐ Hồ Chí Minh 4

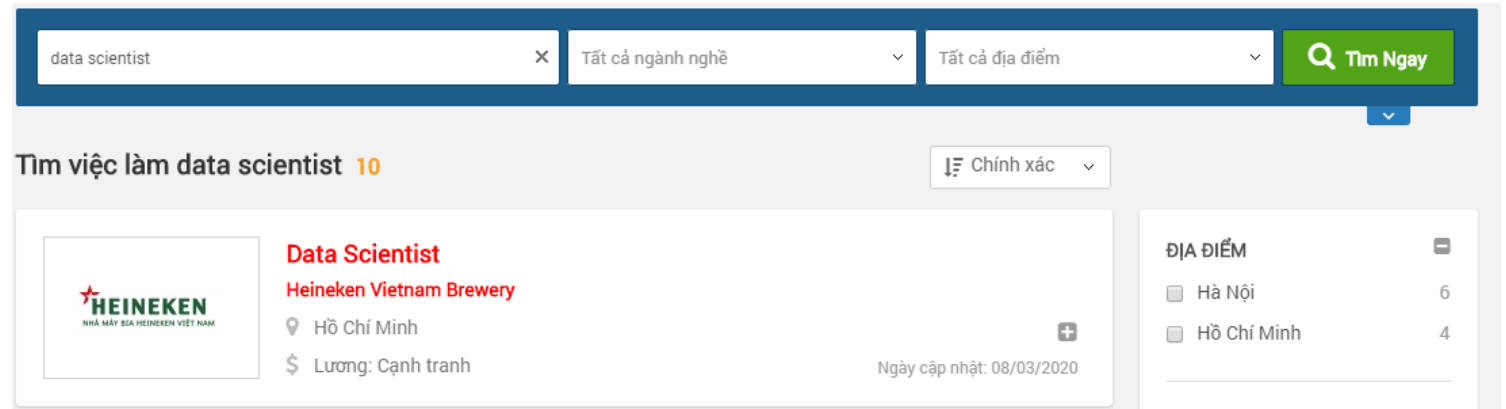
NGÀNH NGHỀ

- ☐ CNTT - Phần mềm 8
- ☐ CNTT - Phần cứng / Mạng 4
- ☐ Tiếp thị / Marketing 2
- ☐ Ngân hàng 1

Collecting data from the CareerBuilder site with the keyword "data scientist"

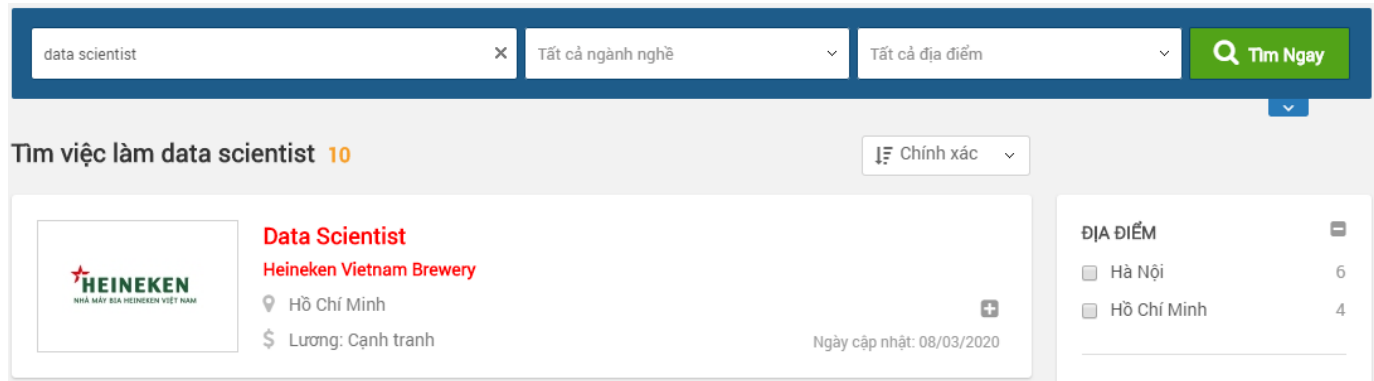
◎ For each recruitment, draw out the information:

- Title
- Recruiters
- Locations
- Wage
- Date Posted Notice
- Link to detailed content
- Detailed content



◎ Save to CSV file (each recruitment is one line)

Collecting data from the CareerBuilder site with the keyword "data scientist"



◎ For each recruitment, draw out the information:

- Title
- Recruiters
- Locations
- Wage
- Date Posted Notice
- Link to detailed content
- Detailed content

The steps taken?

1. Get the website's HTML content
2. Parse HTML to retrieve data needed
3. Write data to CSV file

◎ Save to CSV file (each recruitment is one line)

HTML Code of a Web page

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

This is a Heading

This is a paragraph.

- ◎ HTML code is composed of **tags** and tree form with tag `<html>` as root node
- ◎ Common structure of a tag:
 - `<head>...</head>`: tag contains meta information of the site
 - `<body>...</body>`: tag contains content that will be displayed by the site
 - `<h1>...</h1>`: tag defines the Heading 1
 - `<p>...</p>`: tag defines the paragraphs
 - ...
- ◎ Tags can have the **attribute** to provide more information about the tag
 - `google link`: tag contains links
 - `<h1 id="myHeader">my header</h1>`
 - ...

Retrieving and parse the HTML of your website using Python

© Use library requests-HTML

The screenshot shows the GitHub repository page for `psf/requests-html`. At the top, it displays the repository name, a 'Watch' button with 285 users, a 'Star' button with 10,186 stars, and a 'Fork' button with 631 forks. Below this is a navigation bar with links for 'Code', 'Issues' (72), 'Pull requests' (4), 'Projects' (0), 'Security', and 'Insights'. The main content area features the repository description: 'Pythonic HTML Parsing for Humans™' with a link to <http://html.python-requests.org>. Below the description are tags for 'html', 'scrapping', 'python', 'requests', 'http', 'kennethreitz', 'lxml', 'pyquery', 'css-selectors', and 'beautifulsoup'. A statistics bar shows '438 commits', '3 branches', '30 releases', '52 contributors', and 'MIT' license. At the bottom, there are buttons for 'Branch: master', 'New pull request', 'Find File', and 'Clone or download'. A recent pull request by 'kennethreitz' is highlighted, showing a merge of pull request #317 from 'hugovk/update-org' with the latest commit 'dc82965' on Jul 25. Below this, a list of files is shown: 'docs' (Update links, remove 404s, 2 months ago), 'ext' (sidebars, 2 years ago), and 'tests' (test HTML parser, 7 months ago).

© Install: PowerShell / cmd type `pip install requests-html`

Use basic requests-HTML libraries (document lookups as needed)

◎ Import the library

```
from requests_html import HTMLSession
```

◎ Get the website's HTML code

```
session = HTMLSession()
```

```
r = session.get('web address')
```

```
# r contains all the data sent from the site's server, including the HTML of the website
```

◎ Parse HTML and Tag Search

```
tag = r.html.find(selectors, first=True)
```

```
# selectors are written in the manner of CSS Selector (for example, '#about' means to find the tag with the ID about), how to define the search criteria: using the inspect function of the Web browser
```

```
# first=True this means returning only the first tag found, first=False returns the list containing all the found tags
```

```
# From the found tag, it is possible to call .find(...) to find next in this tag
```

◎ Retrieving tag elements

```
tag.html: tag's HTML string
```

```
tag.text: tag's text string
```

```
tag.attrs: dictionaries containing tag attributes
```

Demo

© Do it by yourself



Note Privacy and Copyright about Data

Note: Avoid doing good things

- Check the "robots. txt" file of the website to see what data is allowed to collect, what data are not allowed
 - For Example: <https://careerbuilder.vn/robots.txt>
- It is not advisable to send too many request to the site in a short time (for example, it is possible to give the program a little sleep between the submitted request)

Note Privacy and Copyright about Data

- ◎ Check file “[robots.txt](https://careerbuilder.vn/robots.txt)” of the site (Example, <https://careerbuilder.vn/robots.txt>)
- ◎ The following Python code can be used to automatically check

```
import urllib.robotparser
rp = urllib.robotparser.RobotFileParser()
rp.set_url('https://careerbuilder.vn/robots.txt')
rp.read()
rp.can_fetch('*', 'https://careerbuilder.vn/viec-lam/data-science-k-vi.html')
# The result will be True or False
```

Contents

- ◎ Review Data Science Process
- ◎ Data Collection from Website
- ◎ **Data Preprocessing**
- ◎ Working with Dynamic Webpage
- ◎ API Data Collection

Ask Question

How is the recruitment situation of the Data Science in Vietnam now?

A more specific question is



What programming languages are often required in the recruitment of DS in Vietnam now?

Assumption: we only demo focus on **careerbuilder with keywords**
“data scientist”

How can we get answers?

- ◎ To the detailed content of each recruitment, see which programming languages are required, and update the corresponding counting variables
- ◎ How do I create program to do that automatically?
 - Need to make a list of programming languages to be counted
 - ◎ Where to get this list?
 - Then, with the detailed content of each recruitment and for each language in the list, check if the language appears in the content, if so, update the corresponding count variable
 - ◎ From the string you can switch to the set of words and then check
 - ◎ Example: 'Proficiency requirements in python, R.'
→ {'Proficiency', 'requirements', 'python', 'R'}



Content [?] → Set of words

- ◎ One way is to use Regular Expression
- ◎ Regular Expression allows to perform complex searches on the string

How to use Regular Expression

Example 1

```
s = 'An has a student ID number 1612345 and email  
an@gmail.com\nHà has a student ID number 1654321 and email  
1654321@hcmus.edu.vn'
```

Request: Find strings 'hcmus' in s

```
import re
```

```
results = re.findall(r'hcmus', s)
```

```
# results: ['hcmus']
```

Raw string

Using strings is also but in some cases will be more troublesome than the raw string

How to use Regular Expression

Example 2

```
s = 'An has a student ID number 1612345 and email  
an@gmail.com\nHà has a student ID number 1654321 and email  
1654321@hcmus.edu.vn'
```

Request: Find the student code (7 digits) in s

```
import re
```

```
results = re.findall(r'\d{7}', s)
```

```
# Results: ['1612345', '1654321', '1654321']
```

```
# Can cast to the set type to remove the duplication
```

Find the string:

- with **numeric characters** (from 0 to 9)
- and **there are 7 characters**

How to use Regular Expression

Example 3

```
s = 'An has a student ID number 1612345 and email  
an@gmail.com\nHà has a student ID number 1654321 and email  
1654321@hcmus.edu.vn'
```

Request: Find the email addresses in S

```
import re
```

```
results = re.findall(r'\w+@[ \w.]+', s)
```

Results:

```
# ['an@gmail.com', '1654321@hcmus.edu.vn']
```

Find the string:

- with **alphabet character**, and there are **one or more such characters**
- then the character **@**
- then the characters in **set** include **word** and character **.**, and there are **one or more such characters**

How to use Regular Expression

Example 4

```
s = 'Required to know c, c++, c#, r, python.'  
# Request: Find the words in S  
import re  
results = re.findall(r'[\w+#]+', s)  
# Results:  
# ['Required', 'to', 'know', 'c', 'c++', 'c#', 'r',  
# 'python']
```



Content ^{re} → set of words
and count the number of occurrences of the languages

Demo..

Contents

- ◎ Review Data Science Process
- ◎ Data Collection from Website
- ◎ Data Preprocessing
- ◎ **Working with Dynamic Webpage**
- ◎ API Data Collection

What is the problem with JavaScript?

- ◎ Example: Get string “Yay! Supports javascript” in <http://avi.im/stuff/js-or-no-js.html>
- ◎ Using the inspect function of the Web browser, you should see the string ID “intro-text”
- ◎ Use Requests-HTML to retrieve ...
- ◎ The result is a string “No javascript support”
 - Cause: HTML content obtained by Requests-HTML is the original content sent from the server, if in this content there is a **JavaScript**, HTML content when using the inspect function of Web browser in the client as HTML content after it has been run JavaScript

How to solve the problem of a website with JavaScript?

- ◎ As [document](#) of Requests-HTML: “Full JavaScript support” 😊

```
session = HTMLSession()  
r = session.get('...')  
r.html.render()
```

- ◎ Function `.render()` will run a browser (without an interface) to fetch HTML content after a JavaScript has been run, and then replace the existing (unjavascript) content with this content (already running JavaScript)
- ◎ Function `.render()` currently **not running at Jupyter Notebook** due to this is somewhat clashed with each other
- ◎ One way to run is **write code in File *.py and run this file in PowerShell/cmd** by typing:
`python file-name.py`

Selenium Library

- ◎ Rather than using the `Render()` method in `Requests-HTML`, we can programmatically control a Web browser and retrieve the HTML content after it has been run by JavaScript.
- ◎ In Python, there are Selenium libraries to do that
 - Selenium **doesn't clash with Jupyter Notebook** 😊
 - Selenium allows programmers to **interactive (fill in information, select, check, Push button,...) with Web browser** 😊 (`Requests-HTML` can't do this)
 - Selenium can be made from A to Z, but will usually run faster if Selenium does not do the `Requests-HTML` jobs and let the rest `Requests-HTML`

Trying with Selenium?

- ◎ Which Vietjet flight from Ho Chi Minh city to Da Nang is the cheapest price in the next 5 days (not include today)?

How to use Selenium?

- ◎ Which Vietjet flight from Ho Chi Minh city to Da Nang is the cheapest price in the next 5 days (not include today)?
- ◎ Steps:
 1. Use Selenium to open web browser and <https://www.vietjetair.com/Sites/Web/vi-VN/Home>
 2. Use Selenium to choose where to go is "Ho Chi Minh City (SGN)", the Destination "Da Nang (DAD)", select "One Way", select the departure date is tomorrow, then press the "Find flights" button
 3. After the results page has been loaded, use Selenium to obtain HTML content, and then give the Requests-HTML for Requests-HTML to handle the rest (parse HTML and search for the data you need)
 4. Repeat step 1 to 3 with the travel date of the next and loop until the full 5 days
 5. From the data collected, find the cheapest flight

Contents

- ◎ Review Data Science Process
- ◎ Data Collection from Website
- ◎ Data Preprocessing
- ◎ Working with Dynamic Webpage
- ◎ **API Data Collection**

Collecting data using Web APIs

- ◎ Some websites offer API (Application Programming Interface) to make external apps retrieve data easier
- ◎ Use the web API "more official" than parse HTML
 - As this is the path that "host" opens to "guests" entering the data
→ If the site has API, use it first.
- ◎ Need to read the host's document to know what data to take, which way to go, ...
- ◎ This is a list (incomplete) of sites providing API
 - <https://github.com/public-apis/public-apis>
 - Large sites like Twitter, Facebook, Google, ... Often provide API
 - Some sites require registration to use the API (charges may apply)

Example: Get information about current weather in Ho Chi Minh City

Parse HTML

The screenshot shows the OpenWeather website for Thanh pho Ho Chi Minh, VN. The current temperature is 29 °C with scattered clouds. A table displays various weather metrics. The developer tools on the right show the HTML structure, with the `tbody` of the weather items table selected. A tooltip indicates the `tbody` has a bounding box of 623.49 x 200. The styles panel shows the `border-box` style applied to the selected element.

OpenWeather

Weather in Thanh pho Ho Chi Minh, VN

29 °C

Scattered clouds

`tbody` 623.49 x 200 data?

Wind	Gentle Breeze, 3.6 m/s, South-southeast (150)
Cloudiness	Scattered clouds
Pressure	1011 hpa
Humidity	83 %
Sunrise	05:42
Sunset	17:45
Geo coords	[10.75, 106.67]

```
<h2 class="weather-widget__city-name">...</h2>
<h3 class="weather-widget__temperature">...</h3>
<p class="weather-widget__main">scattered clouds
</p>
<p>...</p>
<table class="weather-widget__items">
  <tbody> == $0
    <tr class="weather-widget__item">
      <td>Wind</td>
      <td id="weather-widget-wind">...</td>
    </tr>
    <tr class="weather-widget__item">...</tr>
    <tr class="weather-widget__item">...</tr>
    <tr class="weather-widget__item">...</tr>
    <tr class="weather-widget__item">...</tr>
```

... div div span #weather-widget table tbody tr.weather-widget__item td

Styles Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls +

```
element.style {
}

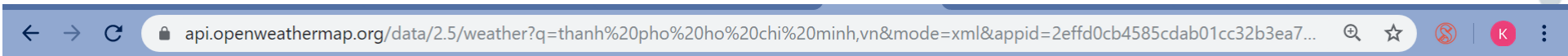
*, :after, bundle_owm ...13644.css:1
:before {
  box-sizing: border-box;
```

border -

- 623.492 x 200 -

Example: Get information about current weather in Ho Chi Minh City

Use API: Almost immediately receive data 😊



This XML file does not appear to have any style information associated with it. The document tree is shown below.

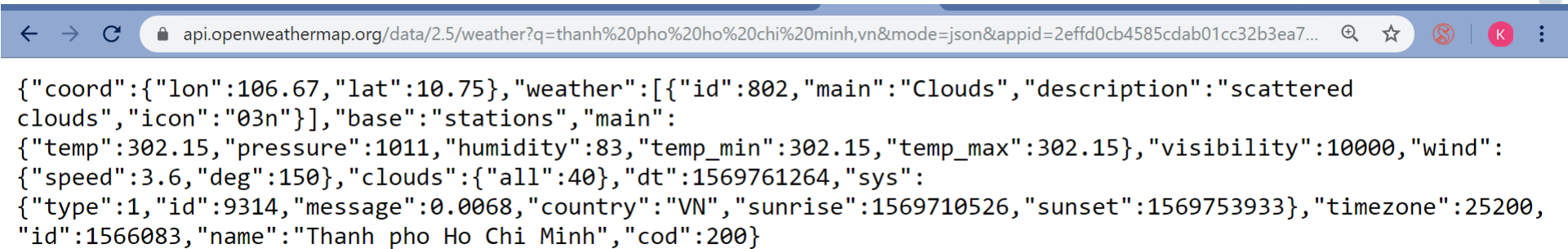
```
<current>
  <city id="1566083" name="Thanh pho Ho Chi Minh">
    <coord lon="106.67" lat="10.75"/>
    <country>VN</country>
    <timezone>25200</timezone>
    <sun rise="2019-09-28T22:42:06" set="2019-09-29T10:45:33"/>
  </city>
  <temperature value="302.15" min="302.15" max="302.15" unit="kelvin"/>
  <humidity value="83" unit="%"/>
  <pressure value="1011" unit="hPa"/>
  <wind>
    <speed value="3.6" unit="m/s" name="Gentle Breeze"/>
    <gusts/>
    <direction value="150" code="SSE" name="South-southeast"/>
  </wind>
  <clouds value="40" name="scattered clouds"/>
  <visibility value="10000"/>
  <precipitation mode="no"/>
  <weather number="802" value="scattered clouds" icon="03n"/>
  <lastupdate value="2019-09-29T12:45:42"/>
</current>
```

This is the XML (eXtensible Markup Language) format, which similar with HTML

- HTML used to display data to viewers
- XML for performing data to exchange between computer applications through a network path
- XML easier parse than HTML

Example: Get information about current weather in Ho Chi Minh City

Use API: Almost immediately receive data 😊



```
{
  "coord": {
    "lon": 106.67,
    "lat": 10.75
  },
  "weather": [
    {
      "id": 802,
      "main": "Clouds",
      "description": "scattered clouds",
      "icon": "03n"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 302.15,
    "pressure": 1011,
    "humidity": 83,
    "temp_min": 302.15,
    "temp_max": 302.15,
    "visibility": 10000,
    "wind": {
      "speed": 3.6,
      "deg": 150
    },
    "clouds": {
      "all": 40
    },
    "dt": 1569761264,
    "sys": {
      "type": 1,
      "id": 9314,
      "message": 0.0068,
      "country": "VN",
      "sunrise": 1569710526,
      "sunset": 1569753933,
      "timezone": 25200,
      "id": 1566083,
      "name": "Thanh pho Ho Chi Minh",
      "cod": 200
    }
  }
}
```

Another format for using API is **JSON** (JavaScript Object Notation)

- JSON is simpler, easier parse than XML (however, the representation is not equal to the XML)
- The simplicity of JSON is sufficient for many cases in practice → JSON is more common than XML
- In the course, we will focus on JSON

JSON

“**JSON** (JavaScript Object Notation) is a **lightweight** data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the [JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999](#). JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- ⦿ A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- ⦿ An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.”

Example: JSON

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

File *.ipynb

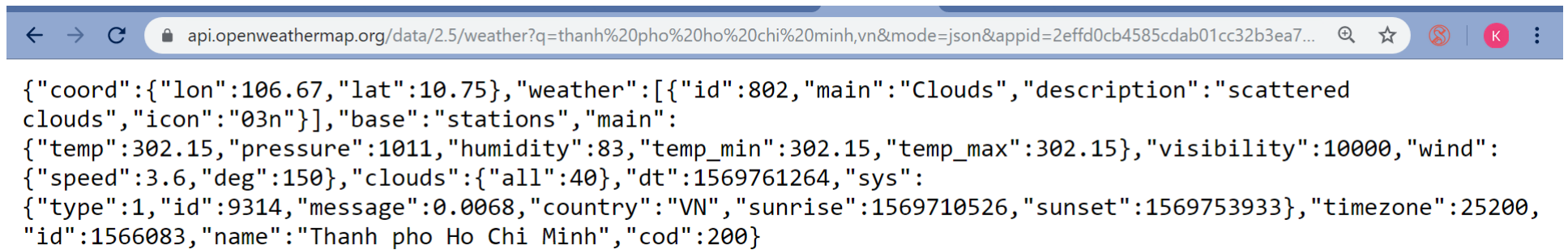
```
{ "cells": [  
  { "cell_type": "markdown",  
    "metadata": {},  
    "source": ["# Continue "]  
  },  
  ...  
]
```

```
[  
  { "id" : 1,  
    "name" : "Hoa"  
    "student": true  
    "email": null  
  },  
  { "id" : 2,  
    "name" : "Mai"  
    "student": true  
    "email": null  
  }  
]
```


How to use Web API in Python?

Q: Get the JSON content that the site returns through the API?

A: Use Requests library



```
{
  "coord": {
    "lon": 106.67,
    "lat": 10.75
  },
  "weather": [
    {
      "id": 802,
      "main": "Clouds",
      "description": "scattered clouds",
      "icon": "03n"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 302.15,
    "pressure": 1011,
    "humidity": 83,
    "temp_min": 302.15,
    "temp_max": 302.15,
    "visibility": 10000,
    "wind": {
      "speed": 3.6,
      "deg": 150
    },
    "clouds": {
      "all": 40
    },
    "dt": 1569761264,
    "sys": {
      "type": 1,
      "id": 9314,
      "message": 0.0068,
      "country": "VN",
      "sunrise": 1569710526,
      "sunset": 1569753933,
      "timezone": 25200,
      "id": 1566083,
      "name": "Thanh pho Ho Chi Minh",
      "cod": 200
    }
  }
}
```

Q: Parse JSON (converting from JSON string to Python data structure)?

A: Use JSON library

Requests Library

- ◎ It is same author with library Requests-HTML
 - if only get site content: use Requests
 - if get site content + parse HTML: use Requests-HTML
- ◎ It is installed when installing Requests-HTML. Otherwise:

pip install requests

- ◎ Basic usage:

```
import requests
```

```
r = requests.get('site path')
```

```
r.text # Content string (HTML/XML/JSON)
```

sent from server

JSON Library

◎ It is built-in library of Python

◎ Basic usage:

```
import json
```

```
# JSON string → data structure of python (parse JSON):
```

```
json_pydata = json.loads(json_str)
```

```
# Data structure of python → JSON string:
```


```
json_str = json.dumps(json_pydata)
```

```
# JSON File → data structure of python:
```

```
json_pydata = json.load(json_fileobj)
```

```
# Data structure of python → JSON file:
```

```
json.dump(json_pydata, json_fileobj)
```



The End

References

© Slides from Tran Trung Kien

