Web scraping

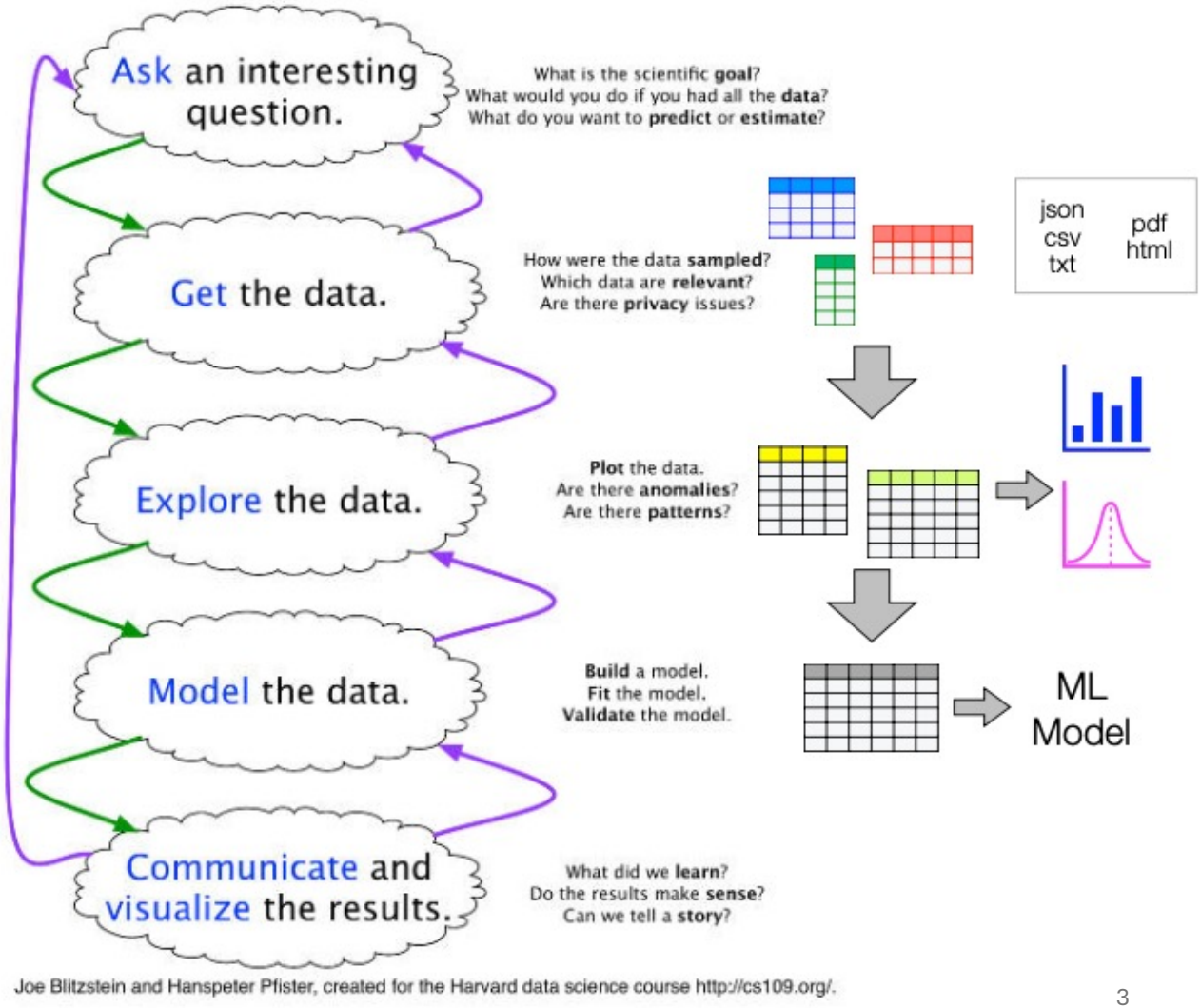
Based on Prof.Natawut's slide

Outline

- Introduction & data extraction steps
- Web scraping
  - Understand web page mechanism
  - HTML
  - CSS & CSS Selector
  - Web page mechanism (DOM & CSSOM)
  - MyWebsite example & DOM tree node structure
  - Parsing a simple webpage: BeautifulSoup
  - Lab1: Basic Web Scraping
  - Lab2: Extract a wiki page
  - Lab3: REST API extraction from Settrade
  - Lab4: Data extraction with Selenium (Google)

Data science process

How to get the data?

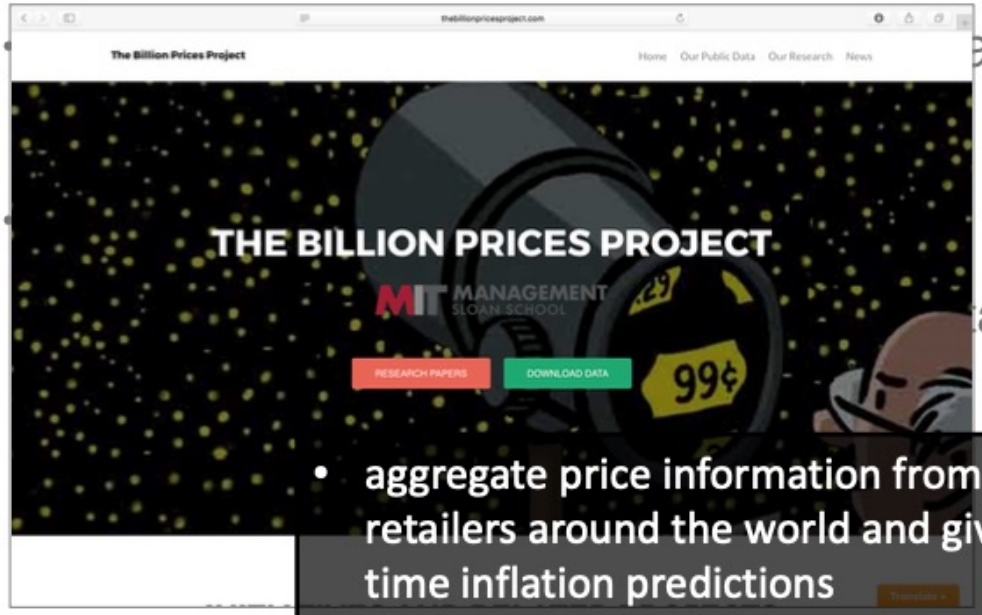


Joe Blitzstein and Hanspeter Pfister, created for the Harvard data science course http://cs109.org/.

Internet data sources

- Internet provides lots of data sources
  - Financial
  - Weather
  - Sports
  - News
- Google creates business by utilizing Internet information
- Most data sources are human-readable, it is also possible to extract information systematically with program. → Web scraping

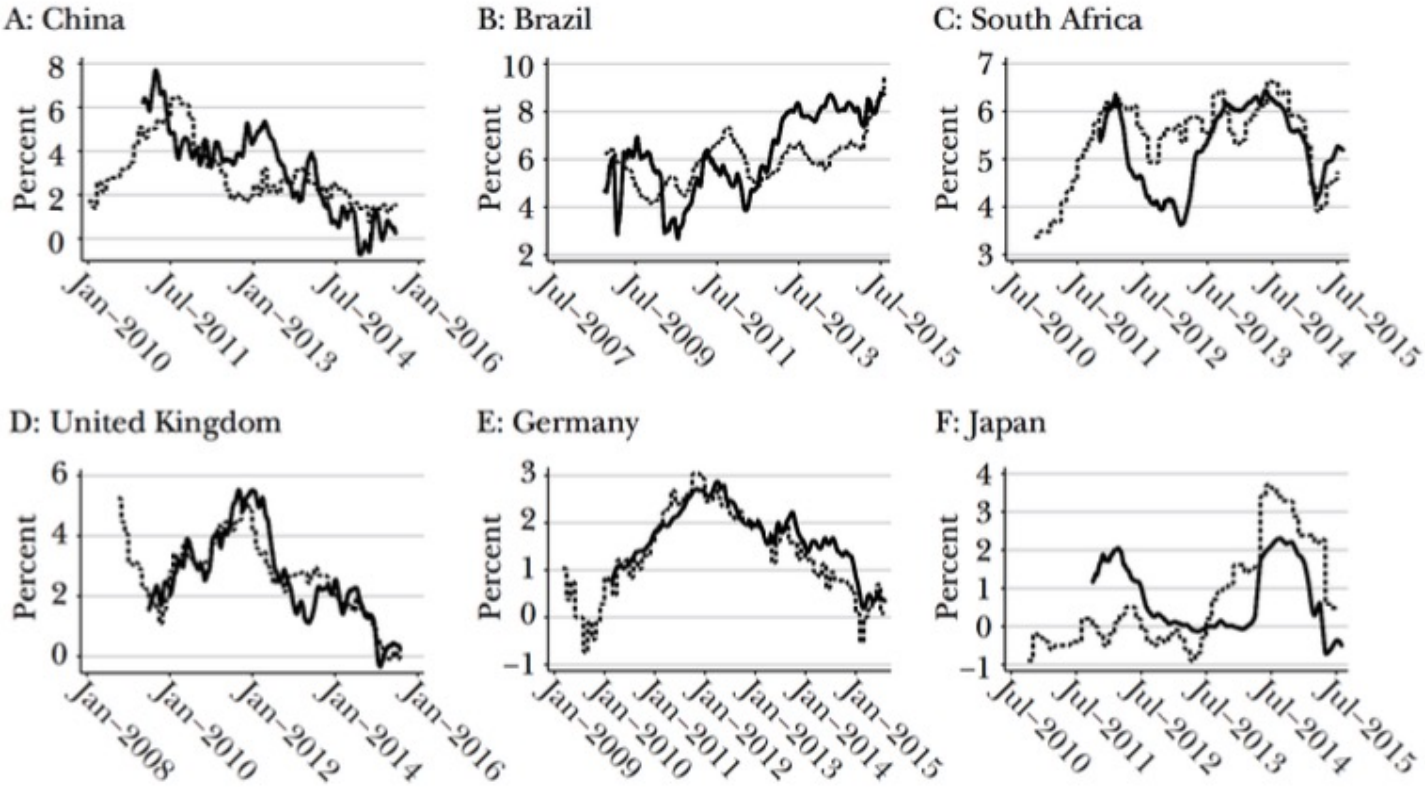
MIT's Billion Prices Project



- aggregate price information from multitude of online retailers around the world and gives real time inflation predictions
- monitor daily price fluctuations of ~5 million items sold by ~300 online retailers in more than 70 countries

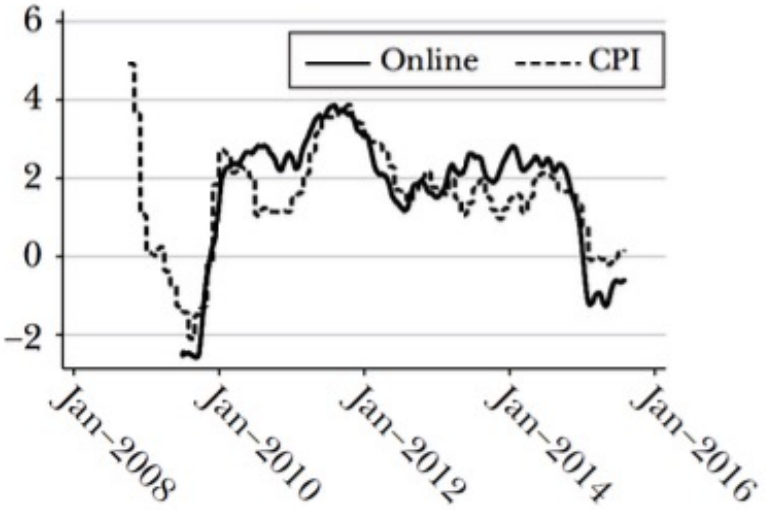
Cavallo, A., & Rigobon, R. (2016). The billion prices project: Using online prices for measurement and research. *The Journal of Economic Perspectives*, *30*(2), 151-178.

Annual inflation rate: online vs. CPI

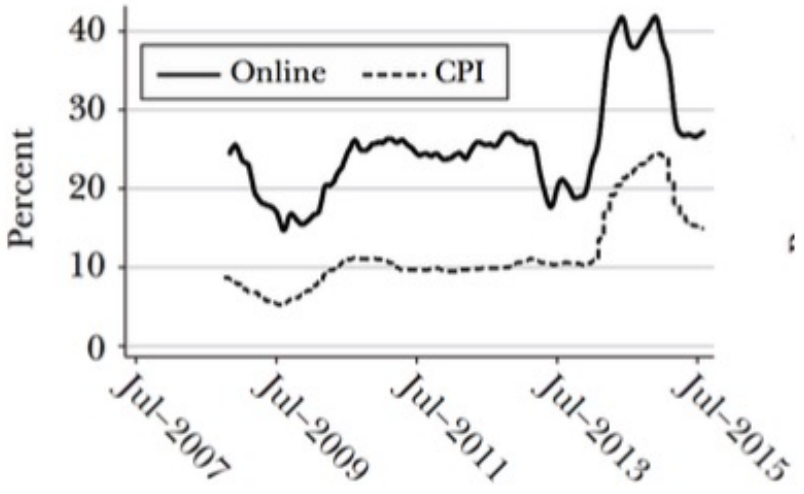


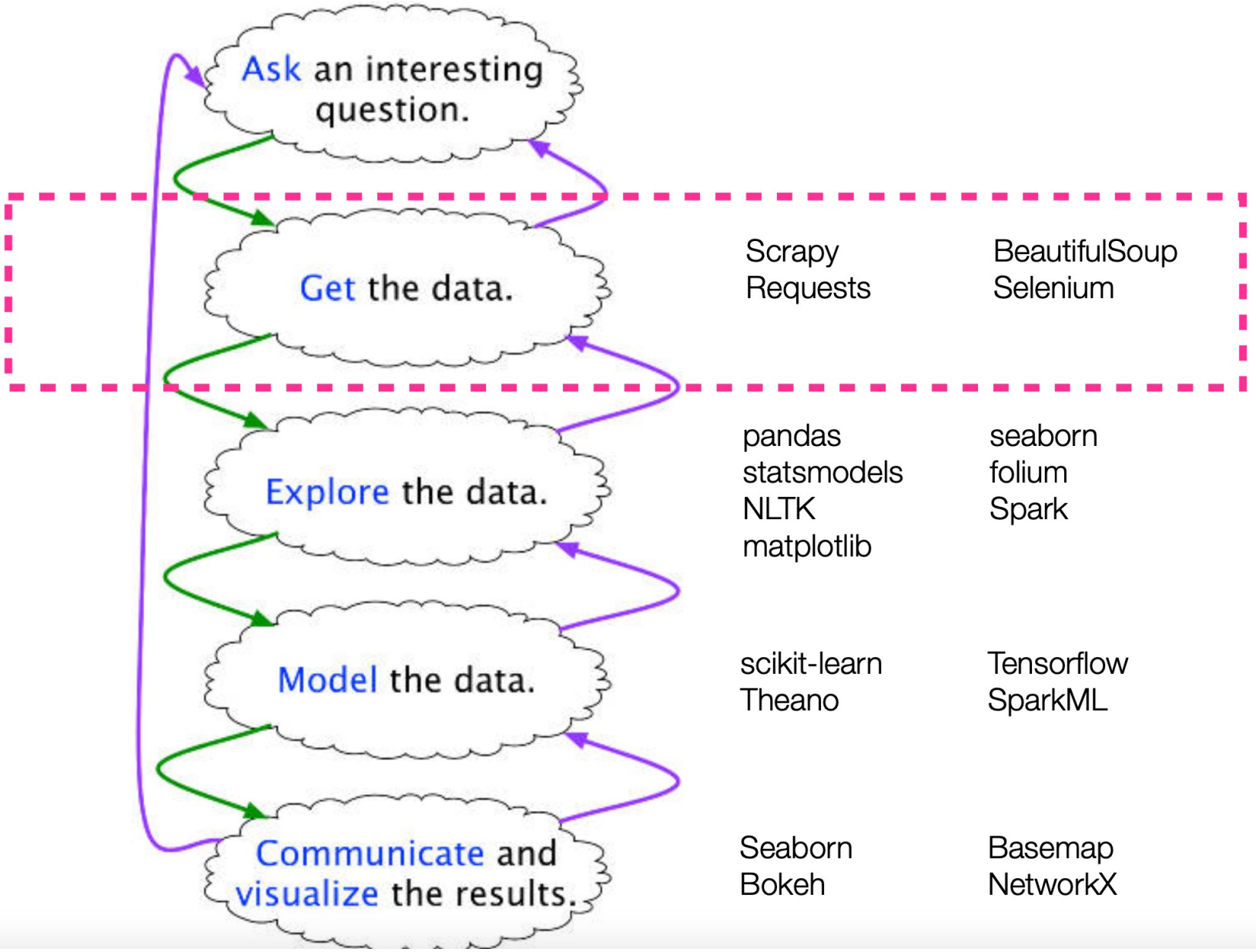
Annual inflation rate: online vs. CPI (cont.)

USA



Argentina




Ask an interesting question.
Get the data.
Explore the data.
Model the data.
Communicate and visualize the results.
Scrapy
Requests
BeautifulSoup
Selenium
pandas
statsmodels
NLTK
matplotlib
seaborn
folium
Spark
scikit-learn
Theano
Tensorflow
SparkML
Seaborn
Bokeh
Basemap
NetworkX

Data extraction steps

- 1) Data Accessing
- 2) Data Parsing
- 3) Data Loading

Data extraction steps (cont.)

- 1) Data Accessing
  - Reading a file [Lab1]
  - Issue a HTTP request [Lab2]
  - Call REST API using Request [Lab3]
  - Call Third-Party API using compatible library
  - Collect complicated web page with Selenium [Lab4]

Data extraction steps (cont.)

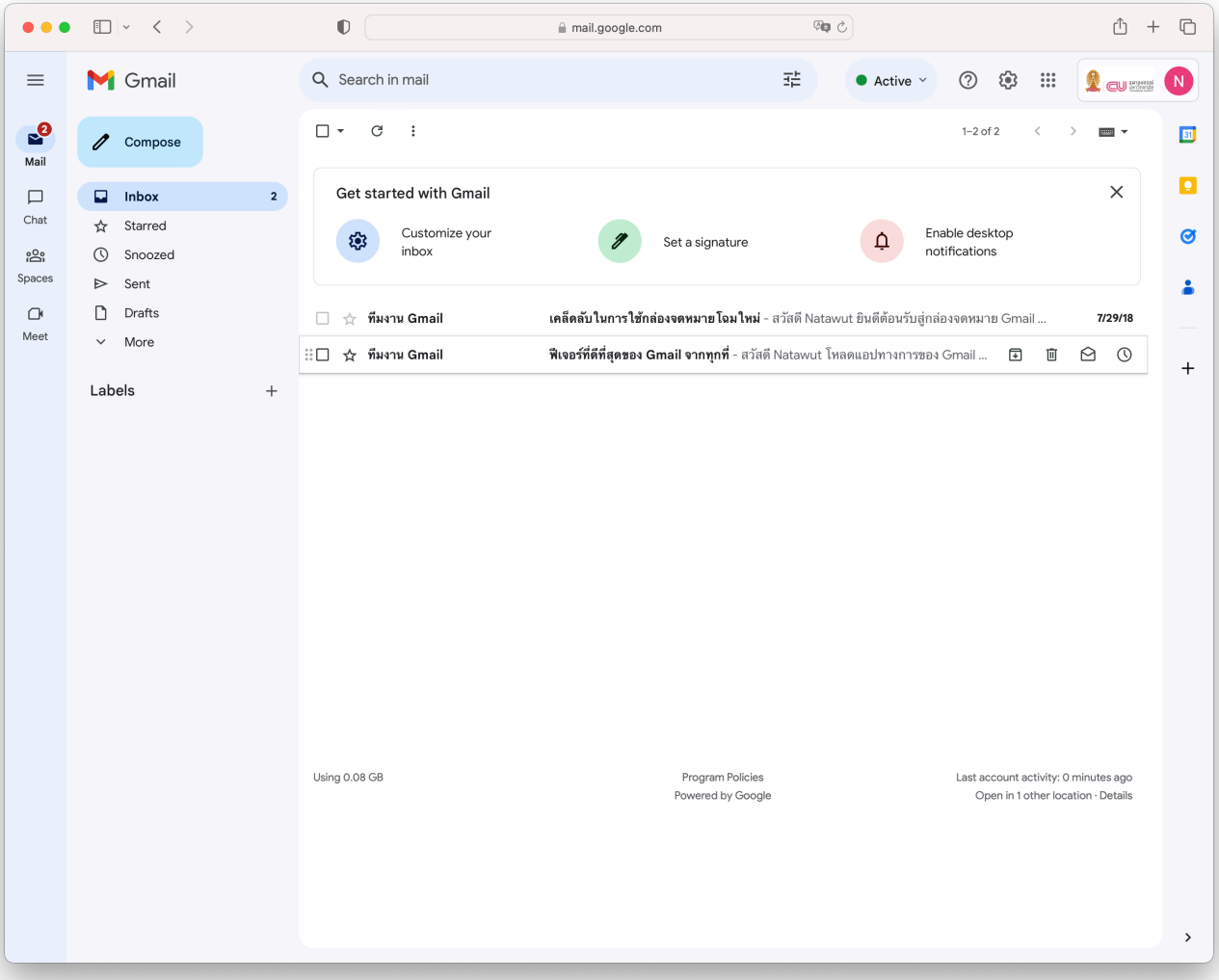
- 2) Data Parsing
  - Parsing HTML page using BeautifulSoup
  - Parsing HTML page using GenAI (Simple but Expensive)
  - Parsing text using regular expressions
  - Parsing docx using python-docx
  - Parsing Excel using Pandas
  - Parsing JSON string using JSON

Data extraction steps (cont.)

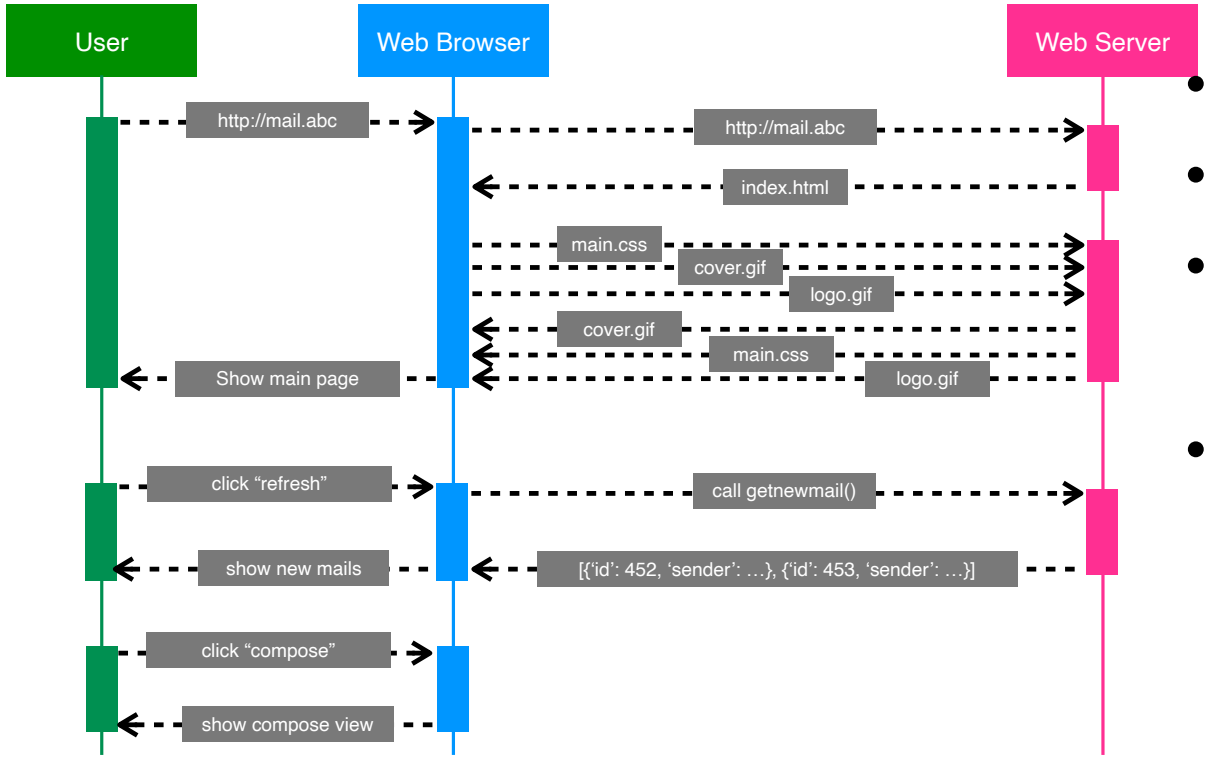
- 3) Data Loading
  - Save to database
  - Save to data file
  - Save to dataframe

Scraping a web page

CHULA ENGINEERING
Foundation toward Innovation



Understand Web Page Mechanism



- 1) index.html
- 2) main.css
- 3) images
- 4) data

HTML: tag

- HTML is a markup language that defines the structure of a web page using series of elements to wrap different parts of the content to make it appear a certain way, or act a certain way
- For example, <p> is a tag to indicate a paragraph

```
<p>Hello, my name is Natawut Nupairoj.  Nice to meet you!</p>
```

- Furthermore, HTML element can be nested

```
<p>Hello, my name is <b>Natawut Nupairoj</b>.  Nice to meet you!</p>
```

- A web browser will show this snippet as followed

Hello, my name is **Natawut Nupairoj**. Nice to meet you!

More HTML element: class & attrs

- HTML element can have an attribute to provide extra information regarding to this element

```
<p class="greeting">Hello, my name is <b>Natawut Nupairoj</b>.  Nice to meet you!</p>
```

Note that "class" is an attribute that can be used in conjunction with CSS

- Some elements have no content and are called void elements

```
<img src="images/firefox-icon.png" alt="My test image" />
```

Anatomy of HTML Page (my_page.html)

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>My test page</title>
  </head>
  <body>
    <p class="greeting">Hello, my name is <b>Natawut Nupairoj</b>.  Nice to meet you!
</p>
    <img src="http://mis.cp.eng.chula.ac.th/view.php?q=instructor/
picture&v=badge&key=00041200" alt="My test image" />
  </body>
</html>
```

Greeting

Hello, my name is **Natawut Nupairoj**. Nice to meet you!

CSS

- CSS (Cascading Style Sheets) is a declarative language that controls how webpages look in the browser
  - Font, size, color, background color, spacing
  - Alignment, number of columns
- The browser applies CSS style declarations to selected elements to display them properly
- A style declaration contains the properties and their values, which determine how a webpage looks

CSS Rules and Selectors

- A CSS rule is a set of properties associated with a selector
- Here is an example that makes every HTML paragraph yellow against a black background:

```
/* The selector "p" indicates that all paragraphs in the document will be
affected by that rule */
p {
  /* The "color" property defines the text color, in this case yellow. */
  color: yellow;

  /* The "background-color" property defines the background color, in this case
black. */
  background-color: black;
}
```

- "Cascading" refers to the rules that govern how selectors are prioritized to change a page's appearance

my_page_with_css.html

```
<!DOCTYPE html>
<html lang="en-US">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <style>
      p {
        color: yellow;
        background-color: black;
      }
    </style>
    <title>My test page</title>
  </head>
  <body>
    <p class="greeting">Hello, my name is <b>Natawut Nupairoj</b>.  Nice to meet you!</p>
    <img src="http://mis.cp.eng.chula.ac.th/view.php?q=instructor/
picture&v=badge&key=00041200" alt="My test image" />
  </body>
</html>
```

<style>...</style>

Note that CSS can be embedded in an HTML page or in a separated file

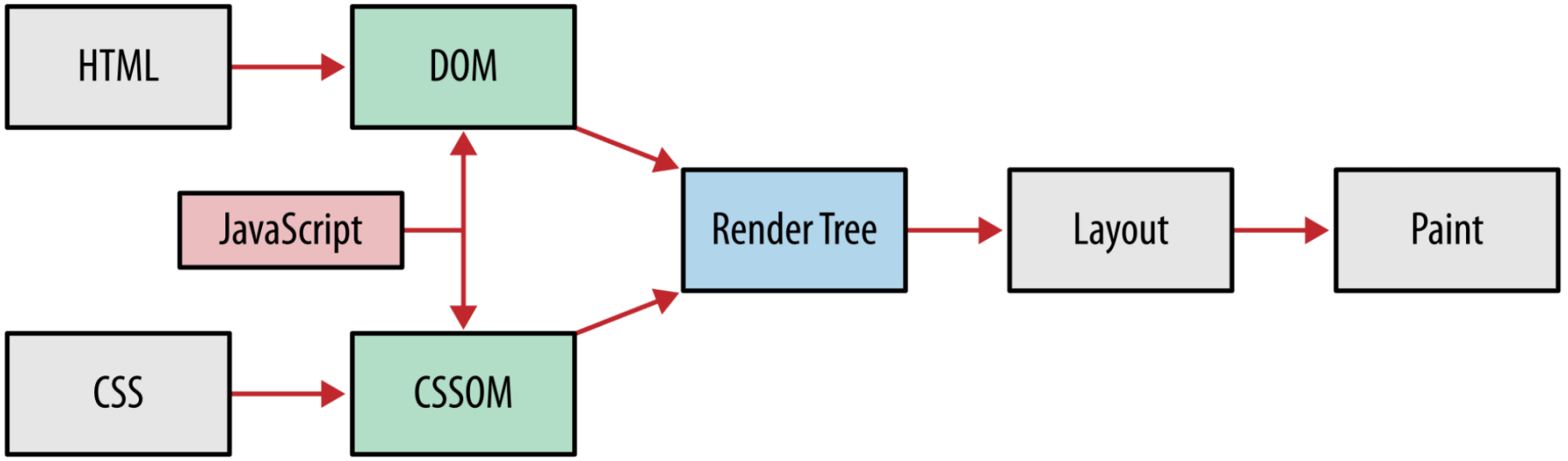
Greeting

Hello, my name is **Natawut Nupairoj**. Nice to meet you!

CSS Selector

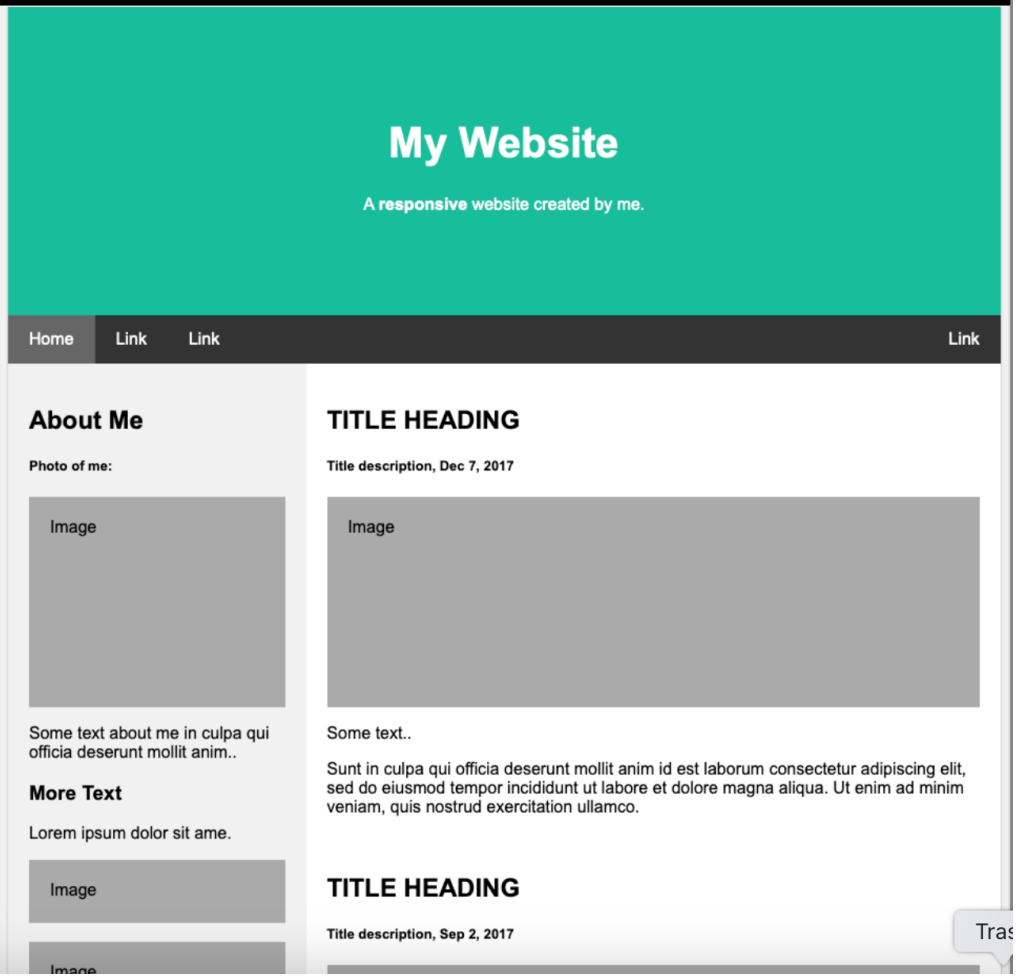
- CSS Selector is a pattern of elements and other terms that tell the browser which HTML elements should be selected to have the CSS property values inside the rule applied to them
  - Elements with a specific tag e.g. all H1 elements
  - Elements with a specific class / id / attribute e.g. all elements with class = “special”
  - Combination of selectors e.g.
    - all H1 elements with class = “special”
    - all H1 elements or elements with class = “special”
- A selector may select multiple elements at once

Web page mechanism (object models)



https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_example_website

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Page Title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}

/* Style the body */
body {
  font-family: Arial, Helvetica, sans-serif;
  margin: 0;
}

/* Header/logo Title */
.header {
  padding: 80px;
  text-align: center;
  background: #1abc9c;
  color: white;
}

/* Increase the font size of the heading */
.header h1 {
  font-size: 40px;
}

/* Sticky navbar - toggles between relative and fixed, depending on the scroll position. It is
positioned relative until a given offset position is met in the viewport - then it "sticks" in place
(like position:fixed). The sticky value is not supported in IE or Edge 15 and earlier versions.
However, for these versions the navbar will inherit default position */
.navbar {
  overflow: hidden;
  background-color: #333;
  position: sticky;
  position: -webkit-sticky;
  top: 0;
}

/* Style the navigation bar links */
.navbar a {
  float: left;
  display: block;
  color: white;
  text-align: center;
  padding: 14px 20px;
  text-decoration: none;
```



simple_page.html

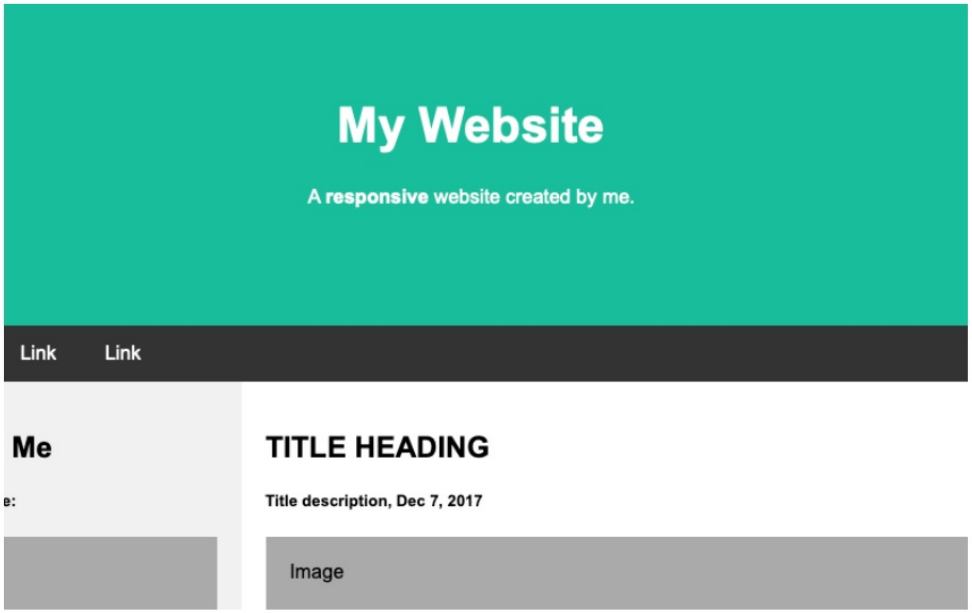
```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Page Title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {
  box-sizing: border-box;
}

/* Style the body */
body {
  font-family: Arial, Helvetica, sans-serif;
  margin: 0;
}

/* Header/logo Title */
.header {
  padding: 80px;
  text-align: center;
  background: #1abc9c;
  color: white;
}

/* Increase the font size of the heading */
.header h1 {
  font-size: 40px;
}

/* Sticky navbar - toggles between relative and fixed, depending on the scroll position. It is positioned relative until
a given offset position is met in the viewport - then it "sticks" in place (like position:fixed). The sticky value is
not supported in IE or Edge 15 and earlier versions. However, for these versions the navbar will inherit default
position */
.navbar {
  overflow: hidden;
  background-color: #333;
  position: sticky;
  position: -webkit-sticky;
  top: 0;
}
```

1) header
2) style (css)
3) body

CHULA ENGINEERING
Foundation toward Innovation

```
<body>

<div class="header">
  <h1>My Website</h1>
  <p>A <b>responsive</b> website created by me.</p>
</div>

<div class="navbar">
  <a href="#" class="active">Home</a>
  <a href="#">Link</a>
  <a href="#">Link</a>
  <a href="#" class="right">Link</a>
</div>

<div class="row">
  <div class="side">
    <h2>About Me</h2>
    <h5>Photo of me:</h5>
    <div class="fakeimg" style="height:200px;">Image</div>
    <p>Some text about me in culpa qui officia deserunt mollit anim..</p>
    <h3>More Text</h3>
    <p>Lorem ipsum dolor sit ame.</p>
    <div class="fakeimg" style="height:60px;">Image</div><br>
    <div class="fakeimg" style="height:60px;">Image</div><br>
    <div class="fakeimg" style="height:60px;">Image</div>
  </div>
  <div class="main">
    <h2>TITLE HEADING</h2>
    <h5>Title description, Dec 7, 2017</h5>
    <div class="fakeimg" style="height:200px;">Image</div>
    <p>Some text..</p>
    <p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco.</p>
    <br>
    <h2>TITLE HEADING</h2>
    <h5>Title description, Sep 2, 2017</h5>
    <div class="fakeimg" style="height:200px;">Image</div>
    <p>Some text..</p>
    <p>Sunt in culpa qui officia deserunt mollit anim id est laborum consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco.</p>
  </div>
</div>

<div class="footer">
  <h2>Footer</h2>
</div>

</body>
```

1) header
2) style (css)
3) body



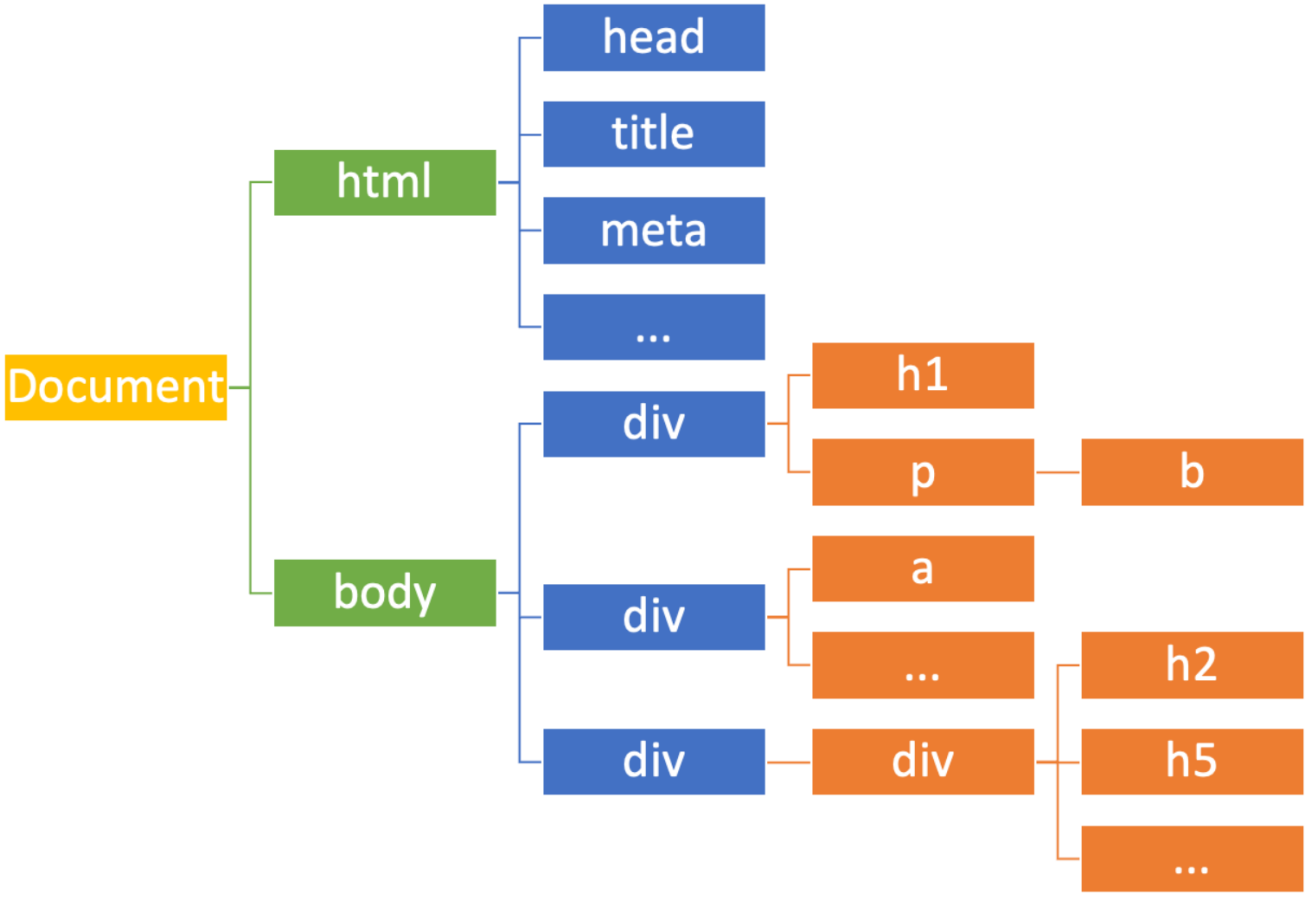
css

```css
/* Header/logo Title */
.header {
  padding: 80px;
  text-align: center;
  background: #1abc9c;
  color: white;
}

/* Increase the font size of the heading */
.header h1 {
  font-size: 40px;
}
```

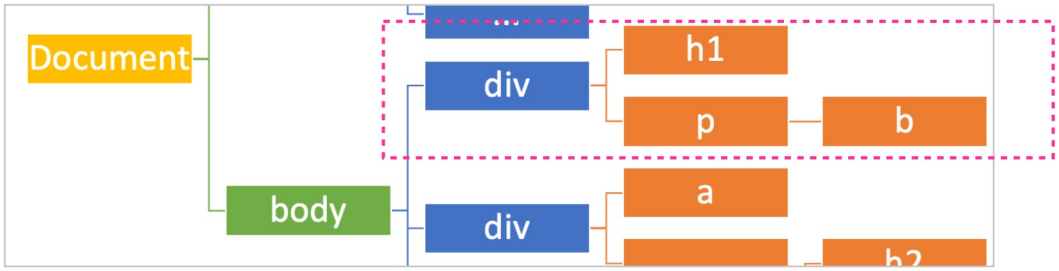
body

```html
<div class="header">
  <h1>My Website</h1>
  <p>A <b>responsive</b> website created by me.</p>
</div>
```

DOM tree structure



DOM node example



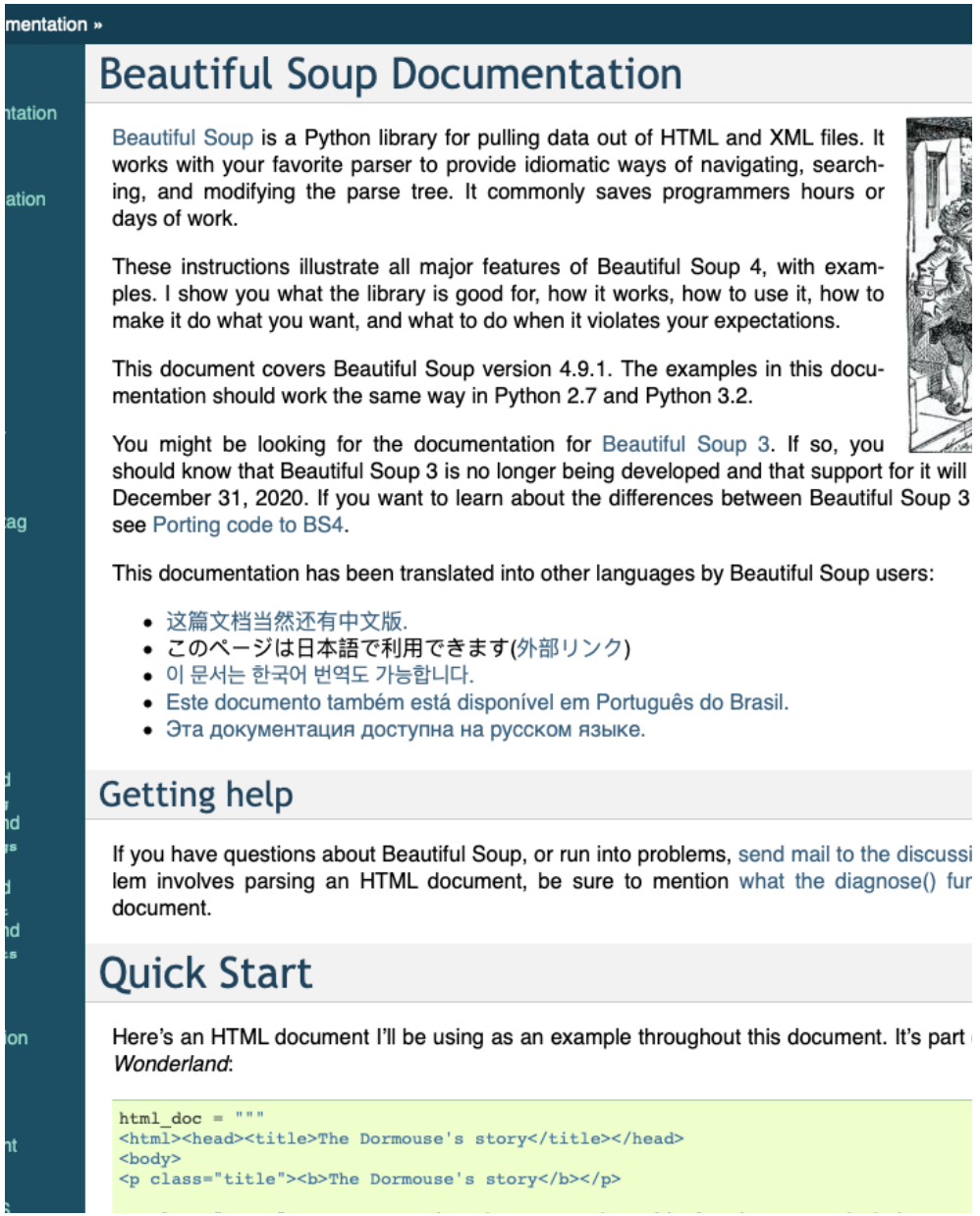

```
<div class="header">
  <h1>My Website</h1>
  <p>A <b>responsive</b> website created by me.</p>
</div>
```

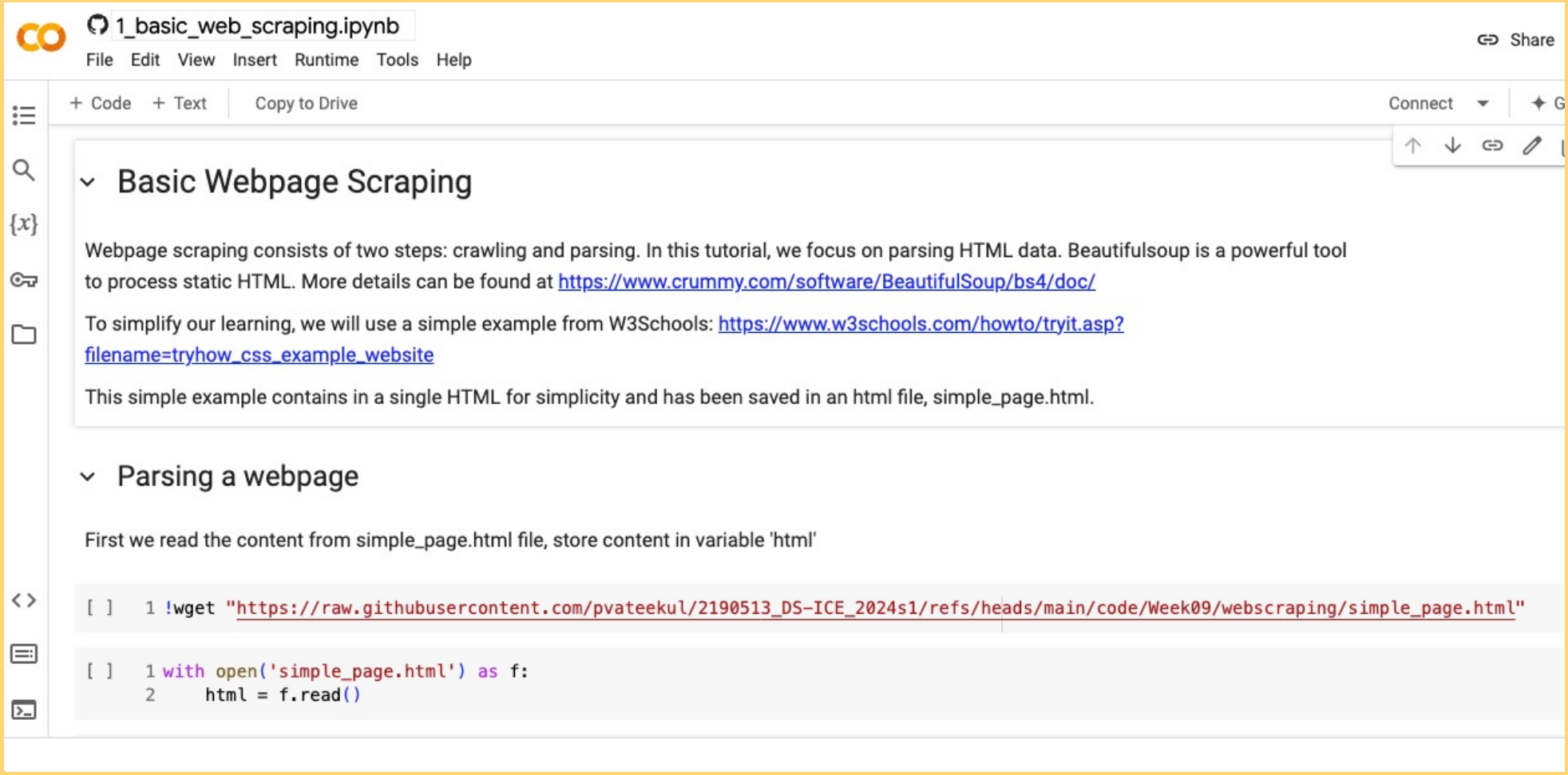
- Node ‘div’ has 2 children (h1 and p) and 1 attribute (key=‘class’, value=‘header’)

CHULA ENGINEERING
Foundation toward Innovation

Parsing A Simple Webpage

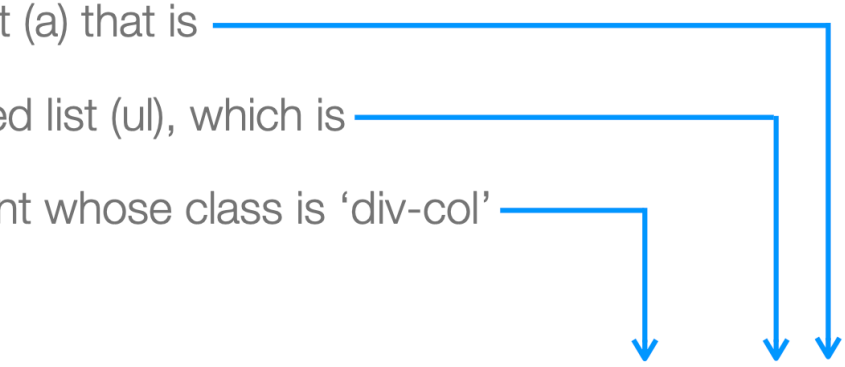
- Applicable for a simple page only
  - No content rendered by javascript
  - Parse HTML with BeautifulSoup
- BeautifulSoup allows
  - Parsing HTML
  - Accessing and Navigating the DOM tree
  - Getting information from nodes




1_basic_web_scraping.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text Copy to Drive
Connect
Basic Webpage Scraping
Webpage scraping consists of two steps: crawling and parsing. In this tutorial, we focus on parsing HTML data. Beautifulsoup is a powerful tool to process static HTML. More details can be found at https://www.crummy.com/software/BeautifulSoup/bs4/doc/
To simplify our learning, we will use a simple example from W3Schools: https://www.w3schools.com/howto/tryit.asp?filename=tryhow_css_example_website
This simple example contains in a single HTML for simplicity and has been saved in an html file, simple_page.html.
Parsing a webpage
First we read the content from simple_page.html file, store content in variable 'html'
[ ] 1 !wget "https://raw.githubusercontent.com/pvateekul/2190513_DS-ICE_2024s1/refs/heads/main/code/Week09/webscraping/simple_page.html"
[ ] 1 with open('simple_page.html') as f:
2 html = f.read()

Understand CSS Selector Syntax

- CSS Selector is a powerful way to navigate DOM tree and locate DOM node(s)
- Find node(s) by tag, id, class, attributes, states (active, focus, hover, etc.), and much more
- Support hierarchical structure e.g. descendant, sibling, order of children, etc.
- In the example, we want to select
  - an anchor element (a) that is
  - inside an unordered list (ul), which is
  - inside a div element whose class is ‘div-col’



```
a_list = soup.select('div.div-col ul a')
```

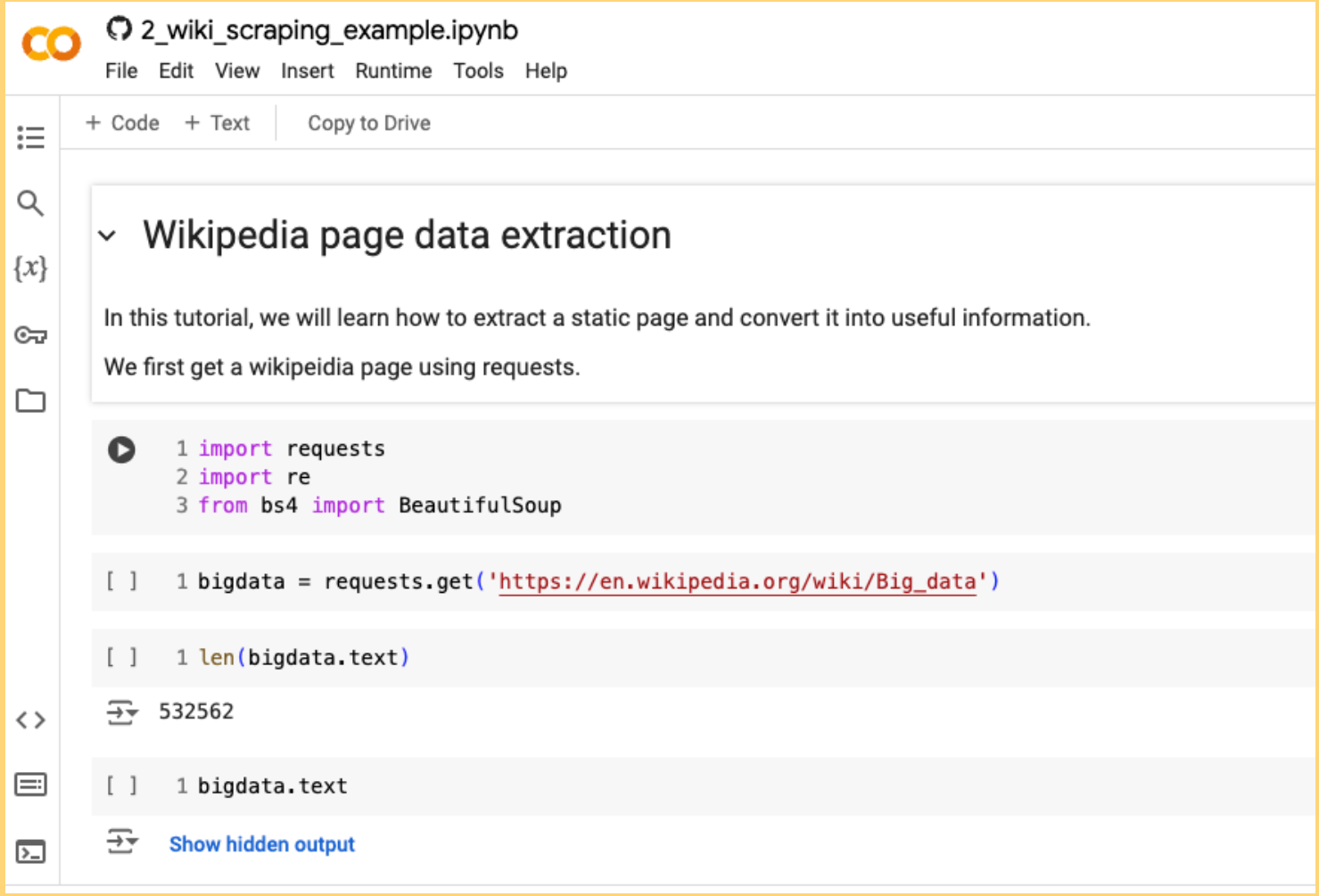
CSS Selector is very powerful for node searching. We can search by tag name, id, class, and combination of criteria. The CSS Selector includes:

- **string**: select node with the specific *tag* e.g. div for node with tag 'div'
- **.class**: select node with the specific *class*
- **#id**: select node with the specific *id*
- **tag[attr]**: select node with the specific *tag* and *attr*

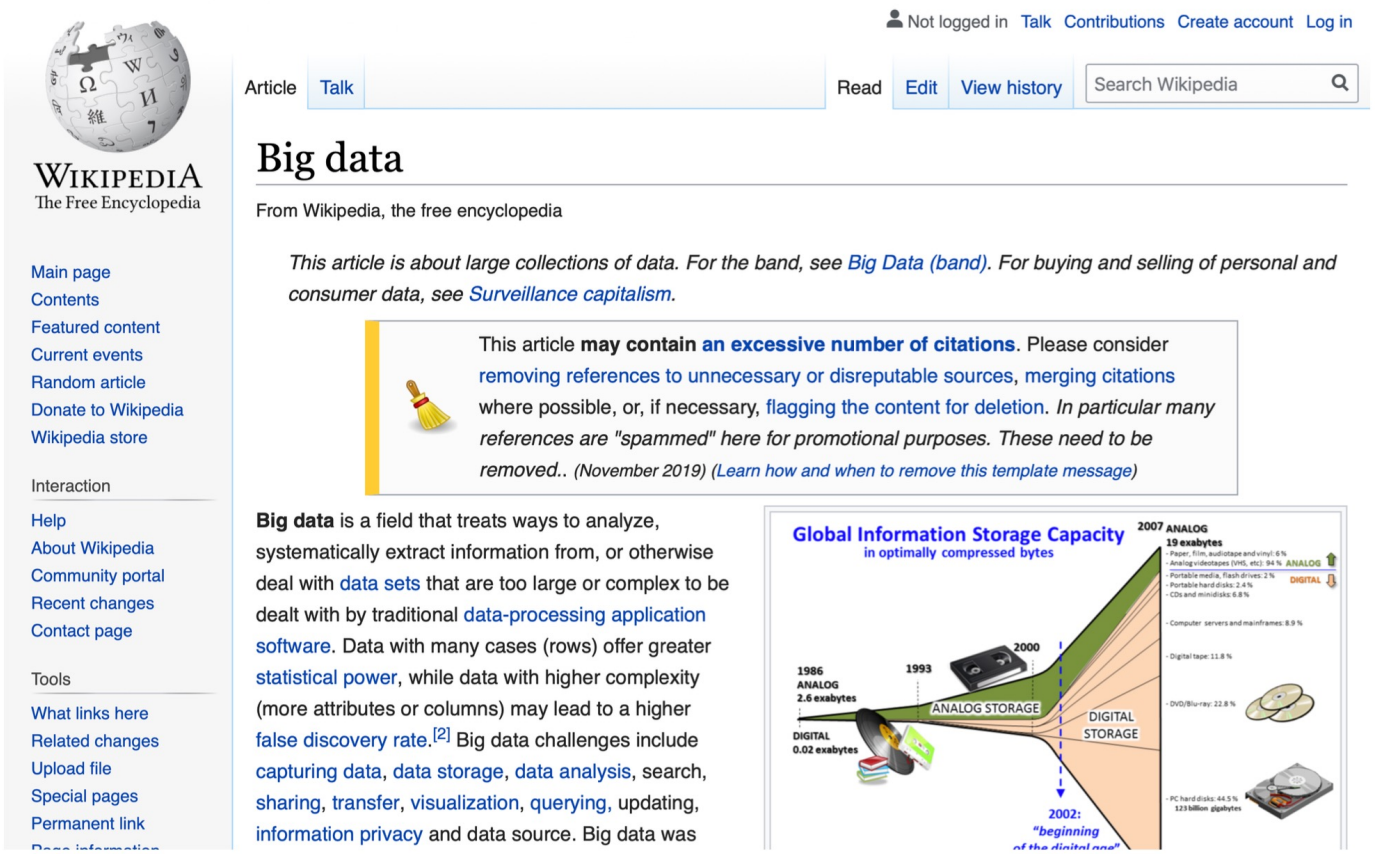
CSS Selector is very powerful for node searching. We can search by tag name, id, class, and combination of criteria. The CSS Selector includes:

- **string**: select node with the specific *tag* e.g. div for node with tag 'div'
- **.class**: select node with the specific *class*
- **#id**: select node with the specific *id*
- **tag[attr]**: select node with the specific *tag* and *attr*

Selector	Example	Example description
.class	.intro	Selects all elements with class="intro"
.class1.class2	.name1.name2	Selects all elements with both *name1* and *name2* set within its class attribute
.class1 .class2	.name1 .name2	Selects all elements with *name2* that is a descendant of an element with *name1*
#id	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
element	p	Selects all <p> elements
element.class	p.intro	Selects all <p> elements with class="intro"
element,element	div, p	Selects all elements and all <p> elements
element element	div p	Selects all <p> elements inside elements
element>element	div > p	Selects all <p> elements where the parent is a element
element+element	div + p	Selects the first <p> element that is placed immediately after elements
element1~element2	p ~ ul	Selects every element that is preceded by a <p> element
[*attribute*]	[target]	Selects all elements with a target attribute
[*attribute=value*]	[target=_blank]	Selects all elements with target="_blank"
[*attribute~=value*]	[title~=flower]	Selects all elements with a title attribute containing the word "flower"
[*attribute|=value*]	[lang|=en]	Selects all elements with a lang attribute value equal to "en" or starting with "en-"
[*attribute^=value*]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"


2_wiki_scraping_example.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text Copy to Drive
Wikipedia page data extraction
In this tutorial, we will learn how to extract a static page and convert it into useful information.
We first get a wikipeidia page using requests.
1 import requests
2 import re
3 from bs4 import BeautifulSoup
[ ] 1 bigdata = requests.get('https://en.wikipedia.org/wiki/Big_data')
[ ] 1 len(bigdata.text)
532562
[ ] 1 bigdata.text
Show hidden output

Example: Extracting references from a wiki page



Not logged in Talk Contributions Create account Log in

Article Talk | Read Edit View history | Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link

Big data

From Wikipedia, the free encyclopedia

This article is about large collections of data. For the band, see Big Data (band). For buying and selling of personal and consumer data, see Surveillance capitalism.

This article **may contain an excessive number of citations**. Please consider removing references to unnecessary or disreputable sources, merging citations where possible, or, if necessary, flagging the content for deletion. *In particular many references are "spammed" here for promotional purposes. These need to be removed.*. *(November 2019) (Learn how and when to remove this template message)*

Big data is a field that treats ways to analyze, systematically extract information from, or otherwise deal with data sets that are too large or complex to be dealt with by traditional data-processing application software. Data with many cases (rows) offer greater statistical power, while data with higher complexity (more attributes or columns) may lead to a higher false discovery rate.[2] Big data challenges include capturing data, data storage, data analysis, search, sharing, transfer, visualization, querying, updating, information privacy and data source. Big data was

- Encouraging members of society to abandon interactions with institutions that would create a digital trace, thus creating obstacles to social inclusion

If these potential problems are not corrected or regulated, the effects of big data policing may continue to shape societal hierarchies. Conscientious usage of big data policing could prevent individual level biases from becoming institutional biases, Brayne also notes.

See also [edit]

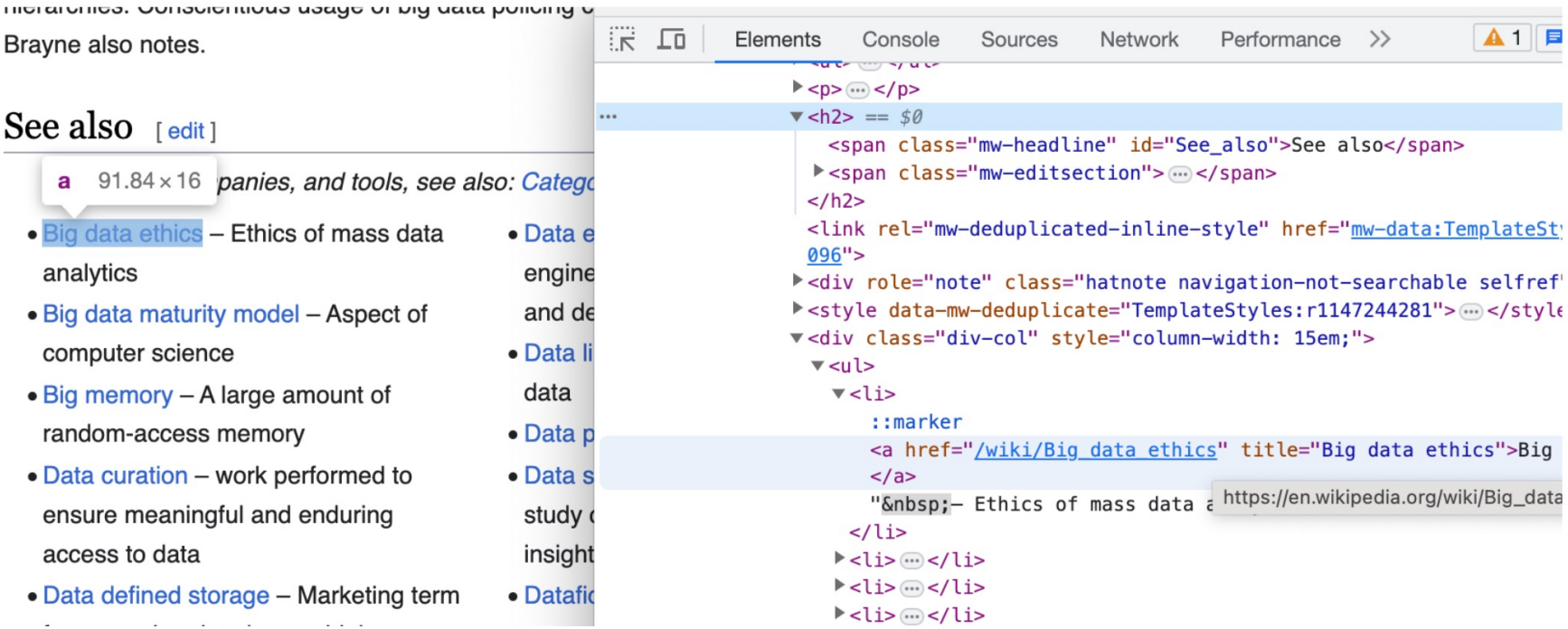
For a list of companies, and tools, see also: Category:Big data

- Big data ethics – Ethics of mass data analytics
- Big data maturity model – Aspect of computer science
- Big memory – A large amount of random-access memory
- Data curation – work performed to ensure meaningful and enduring access to data
- Data defined storage – Marketing term for managing data by combining application, information and storage tiers
- Data engineering – Software engineering approach to designing and developing information systems
- Data lineage – Origins and events of data
- Data philanthropy – Aspect of culture
- Data science – Interdisciplinary field of study on deriving knowledge and insights from data
- Datafication – Technological trend
- Document-oriented database – Type of computer program
- List of big data companies
- Very large database – type of database containing a very large amount of data, so much that it can require specialized architectural, management, processing and maintenance methodologies
- XLDB – annual conference series on databases, data management and analytics

References [edit]

1. ^ Hilbert, Martin; López, Priscila (2011). "The World's Technological Capacity to Store, Communicate, and Compute Information". *Science*. **332** (6025): 60–65. Bibcode:2011Sci...332...60H. doi:10.1126/science.1200970. PMID 21310967. S2CID 206531385. Archived from the original on 14 April 2016. Retrieved 13 April 2016.
2. ^ Breur, Tom (July 2016). "Statistical Power Analysis and the contemporary "crisis" in social sciences". *Journal of Marketing Analytics*. London, England: Palgrave Macmillan. **4** (2–3): 61–65. doi:10.1057/s41270-016-0001-3. ISSN 2050-3318.

104. ^ Josh Rogin (2 August 2018). "Ethnic cleansing makes a comeback – in China". No. Washington Post. Archived from the original on 31 March 2019. Retrieved 4 August 2018. "Add to that the unprecedented security and surveillance state in Xinjiang, which includes all-encompassing monitoring based on identity cards, checkpoints, facial recognition and the collection of DNA from millions of individuals. The authorities feed all this data into an artificial-intelligence machine that rates people's loyalty to the Communist Party in order to control every aspect of their lives."

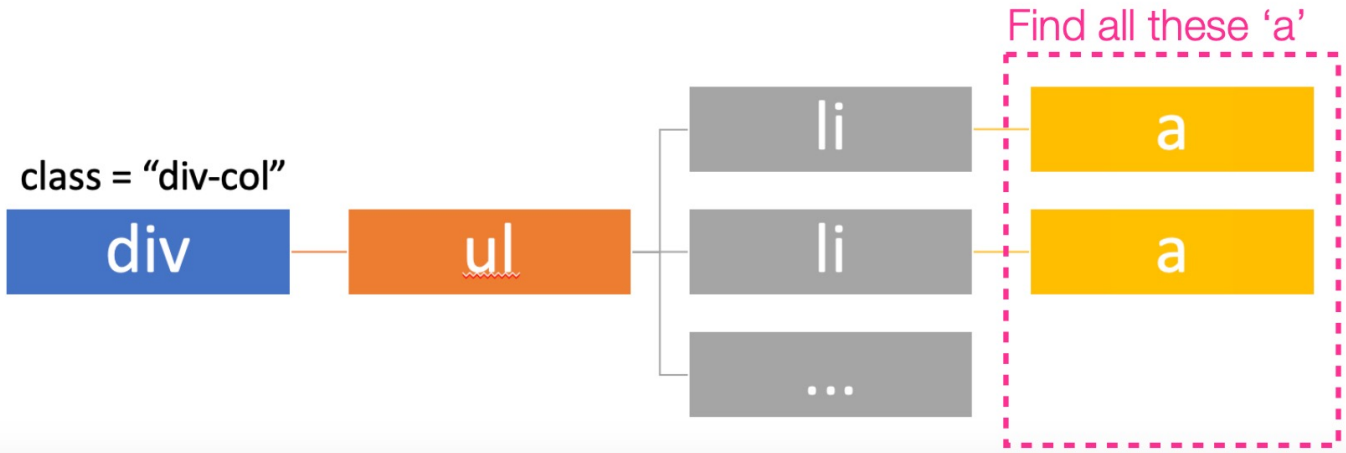

Brayne also notes.
See also [edit]
a 91.84 × 16
panies, and tools, see also: Catego
Big data ethics – Ethics of mass data analytics
Big data maturity model – Aspect of computer science
Big memory – A large amount of random-access memory
Data curation – work performed to ensure meaningful and enduring access to data
Data defined storage – Marketing term
Data e
engine
and de
Data li
data
Data p
Data s
study
insight
Datafic
Elements
Console
Sources
Network
Performance

<h2> == $0
See also

</h2>
<link rel="mw-deduplicated-inline-style" href="mw-data:TemplateSt
096">
<div role="note" class="hatnote navigation-not-searchable selfref
<style data-mw-deduplicate="TemplateStyles:r1147244281"> </style

::marker
Big

" – Ethics of mass data a
https://en.wikipedia.org/wiki/Big_data


Find all these ‘a’
class = “div-col”
div
ul
li
li
...
a
a

3_REST_API_extraction.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

REST API Data Extraction

Gathering data from a REST API is quite typical. Most Single-Page-Application (SPA) and AJAX dynamic pages rely on REST APIs. In addition, most vendor-specific APIs such as Facebook, Twitter, etc., base on REST.

The most important step of extracting data via REST API is to identify the endpoint.

```
[ ]  1 import requests
     2 import json
     3 import pprint
```

ROBOT trade by Settrade Open API | Get Started | API Reference | Use Case | Release Notes | Feedback | FAQ | settrade.com | สมัครเข้าใช้งาน | เข้าสู่ระบบ

Call REST API

After we investigate the main page of settrade.com, we can

```
[ ]  1 api_url = 'http://api.settrade.com/api/mar
```

```
[ ]  1 data_info = requests.get(api_url)
```

ROBOT trade by

Settrade Open API

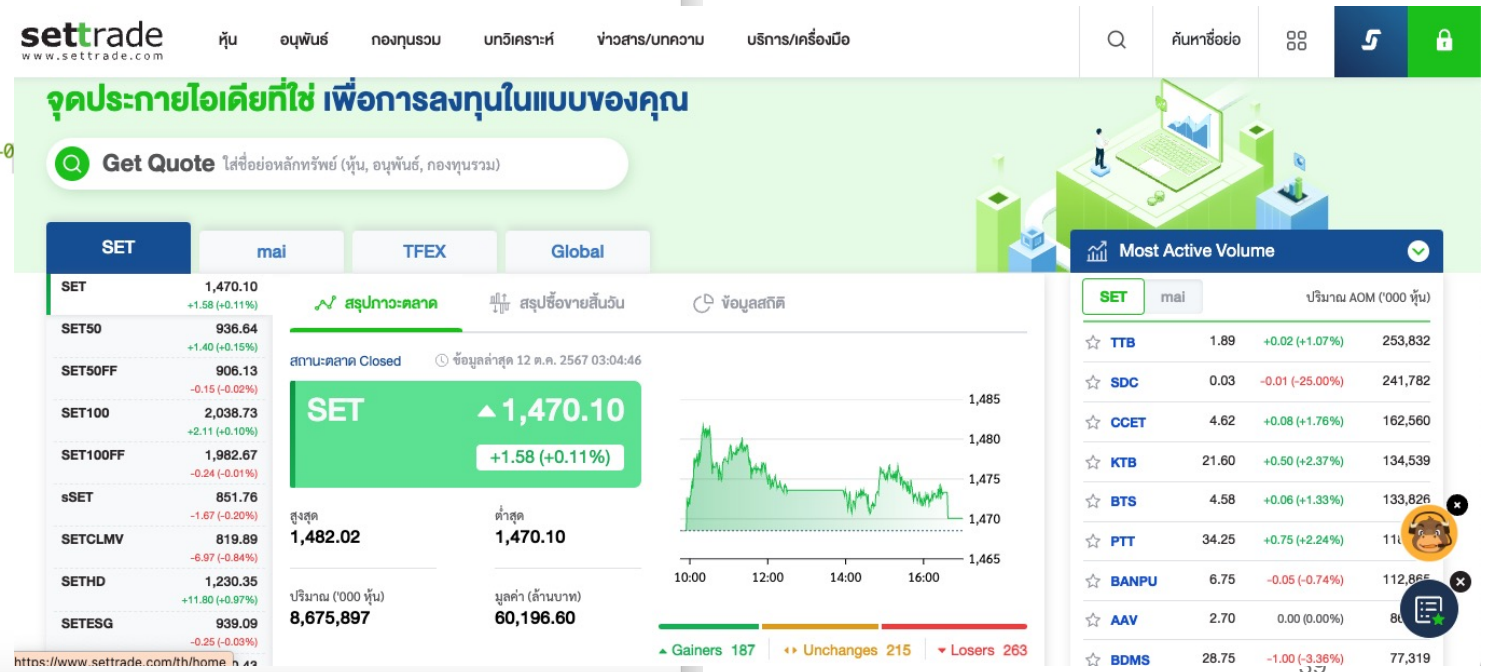
หากคุณเป็นเทรดเดอร์ที่มี Trading Algorithm สุดเจ๋ง
มาเพิ่มขีดความสามารถการเทรดบนตลาด
สร้าง ROBOT trade ของคุณเองด้วย Settrade Open API

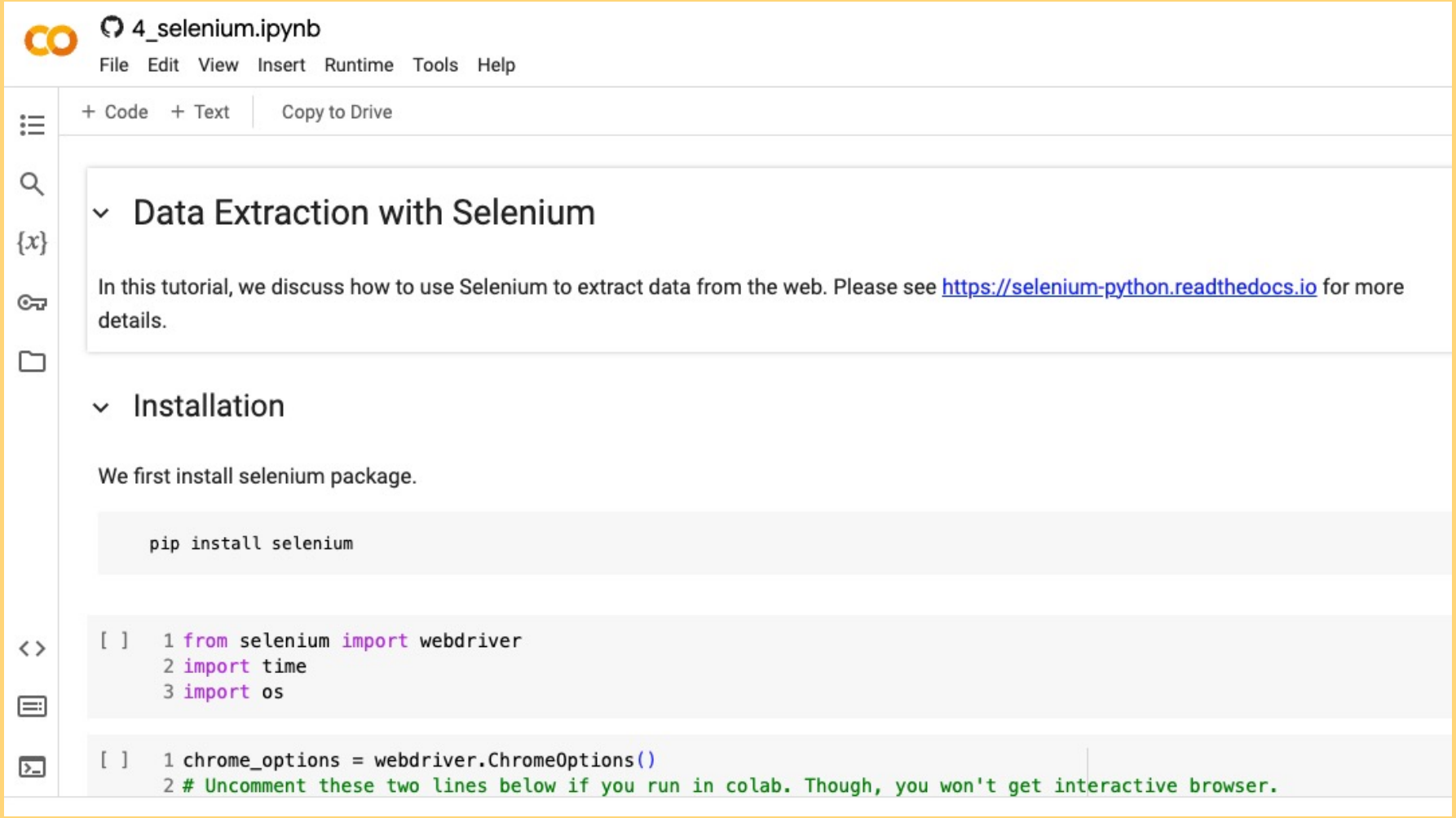
Settrade Open API (Application Programming Interface) เพิ่มโอกาสให้เทรดเดอร์ที่มีความสามารถในการเขียนโปรแกรม สามารถนำแนวคิดในการซื้อขายหลักทรัพย์หรืออนุพันธ์ (Trading Algorithm) ของตนเองมาประยุกต์ใช้กับตลาดหลักทรัพย์แห่งประเทศไทย (SET) เพื่อสร้าง ROBOT trade ทั้งหุ้นและอนุพันธ์

settrade.com/api/set/index/info/list?type=INDEX


```
// 20230611215914
// https://www.settrade.com/api/set/index/info/list?type=INDEX

{
  "indexIndustrySectors": [
    {
      "symbol": "SET",
      "nameEN": "SET",
      "nameTH": "SET",
      "prior": 1559.500000,
      "open": 1558.940000,
      "high": 1559.430000,
      "low": 1549.570000,
      "last": 1555.110000,
      "change": -4.390000,
      "percentChange": -0.280000,
      "volume": 12299648049,
      "value": 43301220472,
      "querySymbol": "SET",
      "marketStatus": "Closed",
      "marketDateTime": "2023-06-10T03:20:11.564264874+0
      "marketName": "SET",
      "industryName": "",
      "sectorName": "",
      "level": "INDEX"
    },
    {
      "symbol": "SET50",
      "nameEN": "SET50",
      "nameTH": "SET50",
      "prior": 945.950000,
      "open": 944.800000,
      "high": 945.280000,
      "low": 938.260000,
      "last": 941.580000,
      "change": -4.370000,
      "percentChange": -0.460000,
      "volume": 1240610800,
      "value": 29122205677.810000,
      "querySymbol": "SET50",
      "marketStatus": "Closed"
```

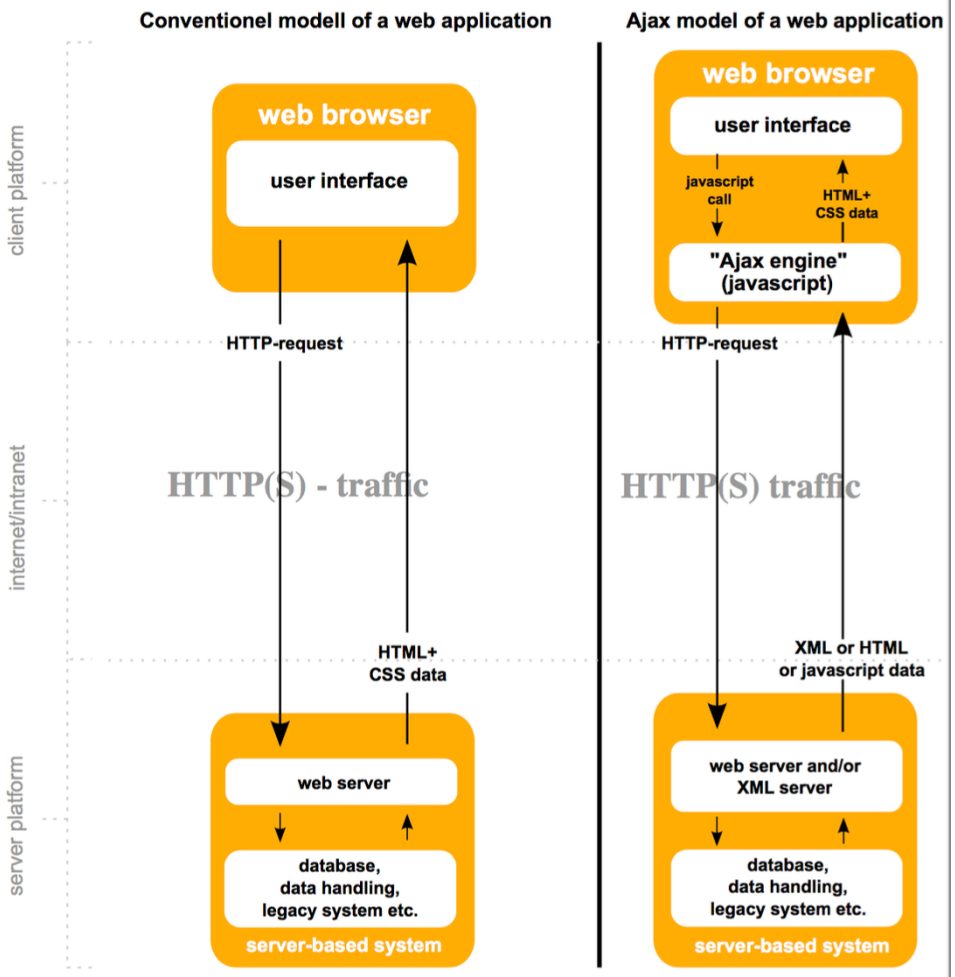

https://www.settrade.com/th/home




4_selenium.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text Copy to Drive
Data Extraction with Selenium
In this tutorial, we discuss how to use Selenium to extract data from the web. Please see https://selenium-python.readthedocs.io for more details.
Installation
We first install selenium package.
pip install selenium
[ ] 1 from selenium import webdriver
2 import time
3 import os
[ ] 1 chrome_options = webdriver.ChromeOptions()
2 # Uncomment these two lines below if you run in colab. Though, you won't get interactive browser.

Extracting AJAX-based Webpage

- Modern web applications are AJAX-based e.g. SPA
- Typical web request approach does not work



Source: https://en.wikipedia.org/wiki/Ajax_(programming)

Extract complicated web page with Selenium

Selenium architecture: Need browser driver!

Selenium capabilities

- Process any website similar to normal web browser
  - Access a web page using normal URL
  - Grab HTML and other resource files in a particular web page
  - Find components based on criteria e.g. CSS selector, etc.
  - Interacting with components in a web page e.g. click, input keyboard
  - Navigate within a web page and across multiple pages
- During operation, Selenium can be normal mode or headless mode
- Checkout https://www.selenium.dev/documentation/ for more information

Example: Selenium on Google

1. Visit https://www.google.com
2. Get HTML of this web page
3. Find a query box
4. Enter “ประเทศไทย” in the query box and press ‘enter’
5. Search result links (‘a’ elements under an element with id=‘search’)
6. Click the first link