Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.

Seoul National University

Jan. 6 – 17, 2020

Python for Data Analytics

Text Processing

Outline

- Introduction to Text Processing
- HTML
- BeautifulSoup

Introduction to Text Processing

Text as Data

- Documents
  - Articles, books and novels
  - E-mails, web pages, blogs
  - Tags, comments
  - Computer programs, logs

- Collections of documents
  - Messages (e-mail, blogs, tags, comments)
  - Social networks (personal profiles)
  - Academic collaborations (publications)

Why Analyze Text?

- Understanding the documents
  - Get the "gist" (요점, 요지) of a document
- Grouping the documents
  - Cluster documents for overview or classification
- Comparison of documents
  - Compare document collections, or inspect evolution of collection over time
- Correlation of documents
  - Compare patterns in text to those in other data, e.g., correlate with social networks

What is Natural Language Processing?

- The study of human languages and how they can be represented computationally and analyzed and generated algorithmically

  The cat is on the mat. → on(mat, cat)

  on(mat, cat) → The cat is on the mat

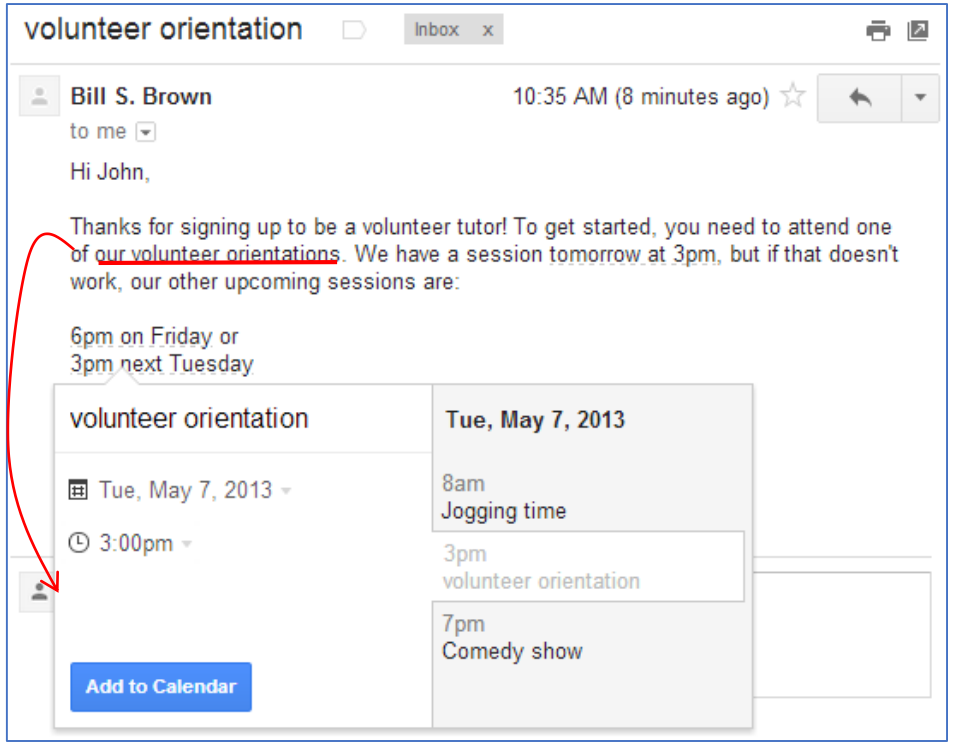
- Building computational models of natural language comprehension and production
- Other names: Computational linguistics
  Human Language Technology
  Natural Language Engineering
  Speech and Text Processing
  (long time ago, Information Retrieval)

NLP is becoming Popular (1)

- IBM Watson won Jeopardy in 2011

  William Wilkinson's "An Account of the Principalities of Wallachia and Moldavia" inspired this author's most famous novel ➡ ***Bram Stoker***

- Information extraction



NLP is becoming Popular (2)

- Information extraction & sentiment analysis

Attributes:

	pos	neg
zoom		
affordability		
size and weight		
flash		
ease of use		

***Size and weight*:**

- Nice and compact to carry! → pos
- Since the camera is small and light, I won't need to carry around those heavy, bulky professional cameras either! → pos
- The camera feels flimsy, is plastic and very light in weight you have to be very delicate in the handling of this camera → neg

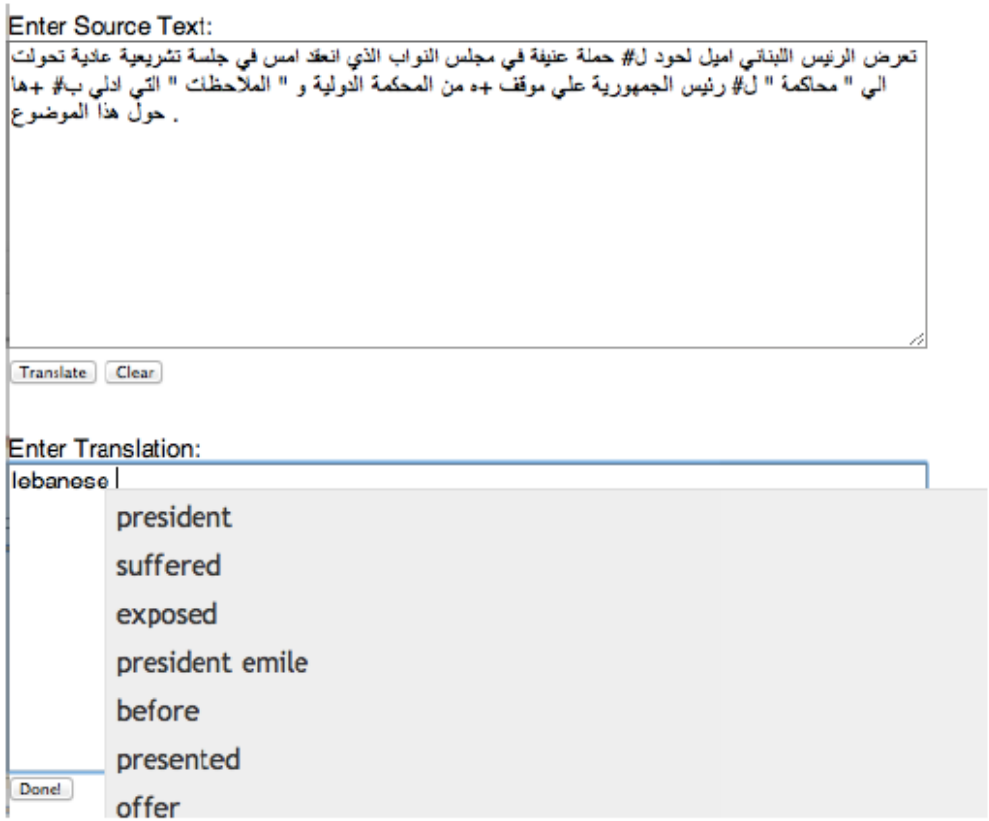
NLP is becoming Popular (3)

- Machine translation
  - Fully automatic

Input 我这心里七上八下的

Output 마음이 조마조마해

  - Helping human translators



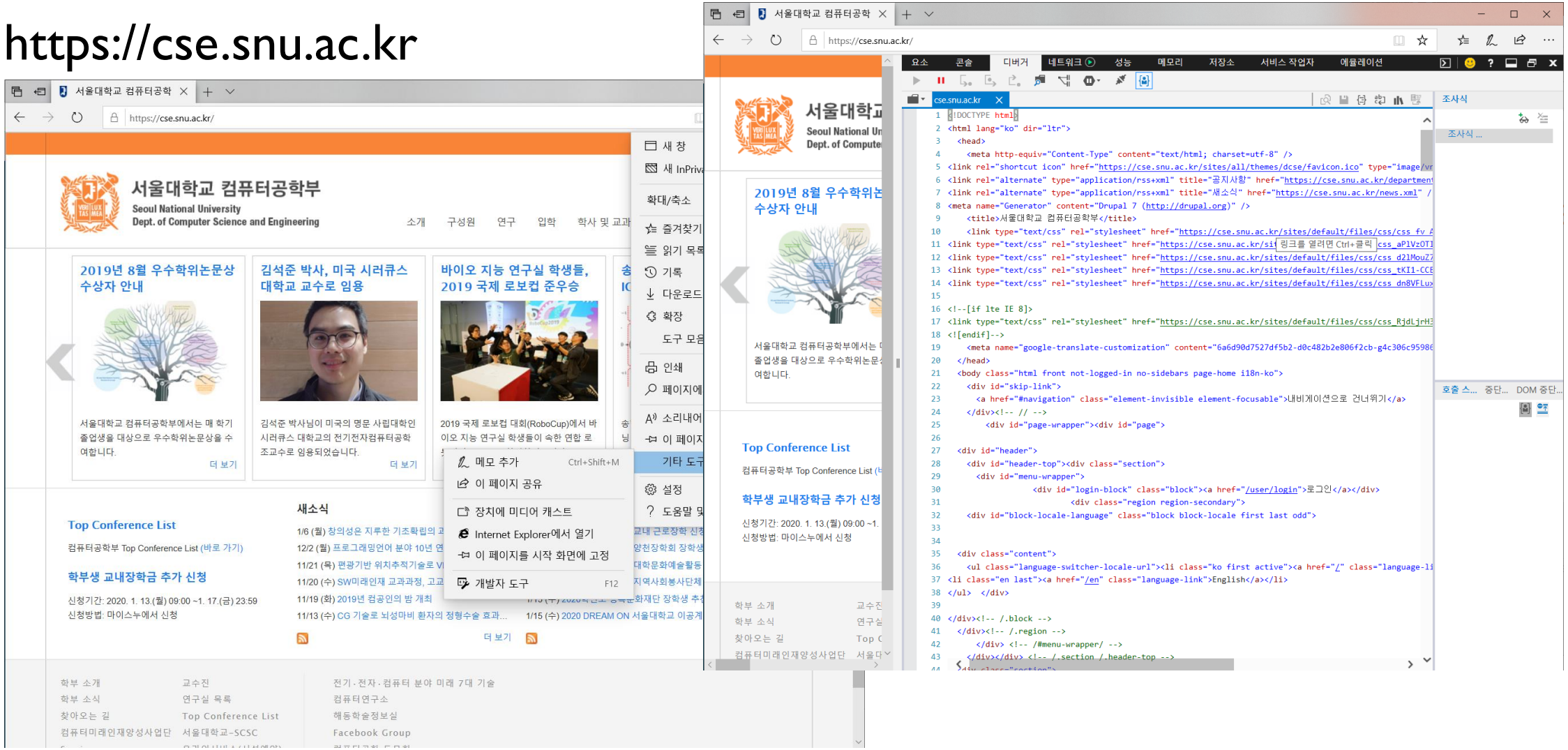
HTML

Web Pages

- Hypertext Markup Language (HTML) is the main language used to define how a Web page should look
- Web pages are created, stored, and sent in HTML encoded form
- A browser converts HTML codes to what we see on the screen
- Features like background color, font, and layout are specified in HTML
  - HTML tags are for basic skeletons of documents
  - CSS rule sets are for styling and embellishing
- HTML 5 is the newest and best WWW standard language

Browsing a Web Page

- https://cse.snu.ac.kr



HTML Page Structure

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

Required Tags

- HTML tags that are required for every Web page:

- <head> tag: 문서전체에 적용되는 정보
  - <title> 문서 제목
  - <script> 클라이언트측 스크립트 정의
  - <style> 문서의 style 정보 정의
  - <meta> HTML 문서에 대한 메타데이터 정의

- <body> tag: 문서의 contents 정보

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <title>Title</title>
  </head>
  <body>
    <p>Content</p>
  </body>
</html>
```

HTML Tags

- Over 118 tags! (including HTML5)
  - Each tag has 0 to 30 attributes
  - http://www.w3schools.com/html/default.asp

<!-->	<!DOCTYPE>	<a>	<abbr>	<acronym>	<address>
<applet>	<area>	<article>	<aside>	<audio>	<b>
<base>	<basefont>	<bdi>	<bdo>	<big>	<blockquote>
<body>	 	<button>	<canvas>	<caption>	
<cite>	<code>	<col>	<colgroup>	<datalist>	<dd>
<del>	<details>	<dfn>	<dialog>	<dir>	
<dl>	<dt>		<embed>	<fieldset>	<figcaption>
…					

Text Formatting

Text Formatting

`<h?> ... </h?>`	Heading (?= 1 for largest to 6 for smallest, eg h1)
`<b> ... </b>`	Bold Text
`<i> ... </i>`	Italic Text
`<u> ... </u>`	Underline Text
`<strike> ... </strike>`	Strikeout
`<sup> ... </sup>`	Superscript - Smaller text placed below normal text
`<sub> ... </sub>`	Subscript - Smaller text placed below normal text
`<small> ... </small>`	Small - Fineprint size text
`<tt> ... </tt>`	Typewriter Text
`<pre> ... </pre>`	Pre-formatted Text
`<blockquote> ... </blockquote>`	Text Block Quote
`<strong> ... </strong>`	Strong - Shown as Bold in most browsers
`<em> ... </em>`	Emphasis - Shown as Italics in most browsers
`<font> ... </font>`	Font tag obsolete, use CSS. (*)

Section Divisions

Section Divisions

`<div> ... </div>`	Division or Section of Page Content
`<span> ... </span>`	Section of text within other content
`<p> ... </p>`	Paragraph of Text
` `	Line Break
`<hr>`	Basic Horizontal Line
`<hr>` Tag Attributes:	
`size="?"`	Line Thickness in pixels
`width="?"`	Line Width in pixels
`width="??%"`	Line Width as a percentage
`color="#??????"`	Line Colour (*)
`align="?"`	Horizontal Alignment: `left, center, right` (*)
`noshade`	No 3D cut-out
`<nobr> ... </nobr>`	Line Break

Images and Linking Tags

Images

`<img src="url" alt="text">`	Basic Image
<img> Tag Attributes:	
`src="url"`	URL or filename of image (required!)
`alt="text"`	Alternate Text (required!)
`align="?"`	Image alignment within surrounding text (*)
`width="??"`	Image width (in pixels or %)
`height="??"`	Image height (in pixels or %)
`border="??"`	Border thickness (in pixels) (*)
`vspace="??"`	Space above and below image (in pixels) (*)
`hspace="??"`	Space on either side of image (in pixels) (*)

Linking Tags

`<a href="url"> link text </a>`	Basic Link
<a> Tag Attributes:	
`href="url"`	Location (url) of page to link to.
`name="??"`	Name of link (name of anchor, or name of bookmark)
`target="?"`	Link target location: `_self`, `_blank`, `_top`, `_parent`.
`href="url#bookmark"`	Link to a bookmark (defined with `name` attribute).
`href="mailto:email"`	Link which initiates an email (dependant on user's email client).

Lists

Lists

`<ol> ... </ol>`	Ordered List
`<ul> ... </ul>`	Un-ordered List
`<li> ... </li>`	List Item (within ordered or unordered)
`<ol type="?">`	Ordered list type: `A, a, I, i, 1`
`<ol start="??">`	Ordered list starting value
`<ul type="?">`	Unordered list bullet type: `disc, circle, square`
`<li value="??">`	List Item Value (changes current and subsequent items)
`<li type="??">`	List Item Type (changes only current item)
`<dl> ... </dl>`	Definition List
`<dt> ... </dt>`	Term or phrase being defined
`<dd> ... </dd>`	Detailed Definition of term

Tables

Tables

`<table> ... </table>`	Define a Table
<table> Tag Attributes:	
`border="?"`	Thickness of outside border
`bordercolor="#??????"`	Border Colour
`cellspacing="?"`	Space between cells (pixels)
`cellpadding="?"`	Space between cell wall and content
`align="??"`	Horizontal Alignment: `left, center, right` (*)
`bgcolor="#??????"`	Background Colour (*)
`width="??"`	Table Width (pixels or %) (*)
`height="??"`	Table Height (pixels or %) (*)
`<tr> ... </tr>`	Table Row within table
`<th> ... </th>`	Header Cell within table row
`<td> ... </td>`	Table Cell within table row
<td> Tag Attributes:	
`colspan="?"`	Number of columns the cell spans across (cell merge)
`rowspan="?"`	Number of row a cell spans across (cell merge)
`width="??"`	Cell Width (pixels or %) (*)
`height="??"`	Cell Height (pixels or %) (*)
`bgcolor="#??????"`	Background Colour (*)
`align="??"`	Horizontal Alignment: `left, center, right` (*)
`valign="??"`	Vertical Alignment: `top, middle, bottom` (*)
`nowrap`	Force no line breaks in a particular cell

```
<table border="1">
  <caption>Country Data</caption>
  <tr>
    <th>Country</th>
    <th>Capital</th>
    <th>Language(s)</th>
  </tr>
  <tr>
    <td>Canada</td>
    <td>Ottawa</td>
    <td>English/French</td>
  </tr>
  <tr>
    <td>Iceland</td>
    <td>Reykjavik</td>
    <td>Icelandic</td>
  </tr>
  <tr>
    <td>Norway</td>
    <td>Oslo</td>
    <td>Norwegian</td>
  </tr>
</table>
```

BeautifulSoup 4

What is bs4 module?

- "bs4": The most famous python package for parsing HTML and XML documents
  - Since 2004 (original author: Leonard Richardson)
  - Useful for web scraping
- BeautifulSoup class
  - Transforming an HTML document into a complex tree of Python objects
  - A parse tree for parsed pages that can be used to extract data from HTML
  - We can easily navigate & search a parse tree
  - Convert incoming documents to Unicode and outgoing documents to UTF-8

```
>>> from bs4 import BeautifulSoup
```

Installing Parsers for BeautifulSoup Class

Parser	Typical usage	Notes
Python's **'html.parser'**	`BeautifulSoup(markup, 'html.parser')`	Not as fast as **lxml**
'lxml' HTML parser	`BeautifulSoup(markup, 'lxml')`	Very fast `$pip install lxml`
'lxml' XML parser	`BeautifulSoup(markup, 'lxml-xml')`	
'html5lib' parser	`BeautifulSoup(markup, 'html5lib')`	`$ pip install html5lib`

file

```
from bs4 import BeautifulSoup

with open('doc.html') as fp:
    soup = BeautifulSoup(fp, 'html.parser')
```

string

```
from bs4 import BeautifulSoup

html_doc = "<html><head> .... </html>"
soup = BeautifulSoup(html_doc, 'html.parser')
```

network

```
import requests
from bs4 import BeautifulSoup

response = requests.get('http://cse.snu.ac.kr')
soup = BeautifulSoup(response.text, 'html.parser')
```

BeautifulSoup Class: Attribute List

Functions		
get()	find_all_next()	insert_after()
get_text()	find_next()	clear()
prettify()	find_all_previous()	extract()
find()	find_previous()	decompose()
find_all()	select()	replace_with()
find_parents()	append()	wrap()
find_parent()	new_tag()	unwrap()
find_next_siblings()	insert()	
find_next_sibling()	insert_before()	

Parsing

```
html_doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""
```

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')
soup
```

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>
<p class="story">Once upon a time there were three little sisters; and their names were
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>
<p class="story">...</p>
</body></html>
```

Using Attributes

```
soup.title
```

```
<title>The Dormouse's story</title>
```

```
soup.title.name
```

```
'title'
```

```
soup.title.string
```

```
"The Dormouse's story"
```

```
soup.title.parent.name
```

```
'head'
```

```
soup.p
```

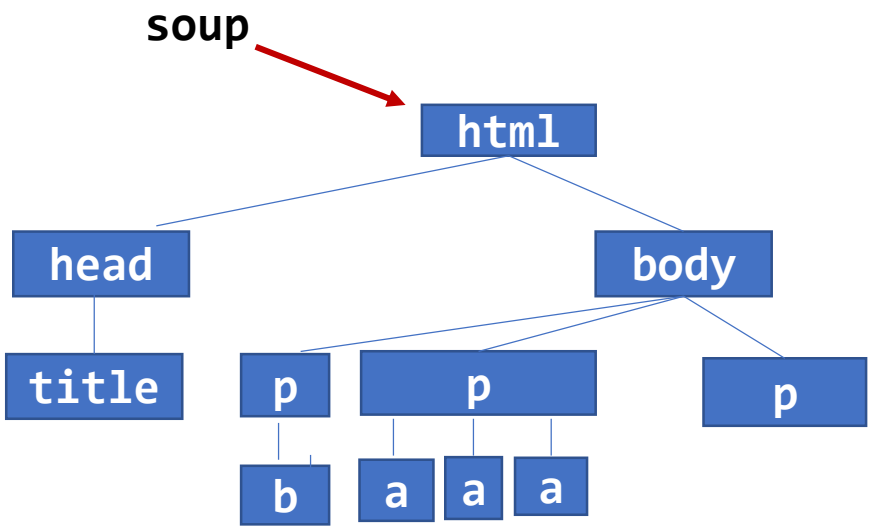
```
<p class="title"><b>The Dormouse's story</b></p>
```

```
soup.p['class']
```

```
['title']
```

```
soup.a
```

```
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
```



htmlparser가 만든 parse tree

Using Functions

```
soup.find_all('a')
```

```
[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
 <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
 <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

Find all hyperlinks

```
soup.find(id='link3')
```

```
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
```

Find all the matching attributes and values

```
for link in soup.find_all('a'):
    print(link.get('href'))
```

```
http://example.com/elsie
http://example.com/lacie
http://example.com/tillie
```

Extract all the URLs found within a page 'a' tags

```
print(soup.get_text())
```

```
The Dormouse's story

The Dormouse's story
Once upon a time there were three little sisters; and their names were
Elsie,
Lacie and
Tillie;
and they lived at the bottom of a well.
...
```

Extract all the text from a page

find_all()

- `find_all(`*name, attrs, recursive, string, limit, ...*`)`
  - Looking through a tag's descendants and retrieves all descendants that match your filters

```
soup.find_all('title')
```

```
[<title>The Dormouse's story</title>]
```

Find the tag 'title'

```
soup.find_all(id='link2')
```

```
[<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]
```

Find the attribute 'id' which has the value 'link2'

```
soup.find_all(id=True)
```

```
[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
 <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
 <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

Find the attribute 'id' which has a value

```
soup.find_all('a', class_='sister')
```

```
[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
 <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
 <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

Find a CSS class named 'sister' in the tag 'a'

find_all() (cont'd)

```
soup.find_all(string='Elsie')
```

```
['Elsie']
```

Find the string 'Elsie'

```
soup.find_all('a', string='Lacie')
```

```
[<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]
```

Find the string 'Lacie' in the tag 'a'

```
soup.find_all('a', limit=2)
```

```
[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
 <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]
```

Just return the maximum 2 results

```
soup.find_all('title', recursive=False)
```

```
[]
```

Just find among direct children

```
soup.head.find_all('title', recursive=False)
```

```
[<title>The Dormouse's story</title>]
```

Find the tag in the children of 'head'

```
soup('a')
```

```
[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
 <a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>,
 <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
```

= `soup.find_all('a')`

```
<html>
  <head>
    <title>
        ...
    </title>
  </head>
</html>
```

find()

- Equivalent to `find_all(..., limit=1)`
- `find_all()` returns a list containing the single result, while `find()` just returns the result

```
soup.find('title')
```

```
<title>The Dormouse's story</title>
```

```
soup.find('p')
```

```
<p class="title"><b>The Dormouse's story</b></p>
```

```
soup.find('a')
```

```
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
```

Retrieving Contents

```
soup.find('title').get_text()
```

```
"The Dormouse's story"
```

Get the text of the tag 'title'

```
soup.find('title').text
```

```
"The Dormouse's story"
```

`.text == .get_text()`

```
for a in soup.find_all('a'):
    print(a.get('href'))
```

```
http://example.com/elsie
http://example.com/lacie
http://example.com/tillie
```

get the URL in the 'href' attribute

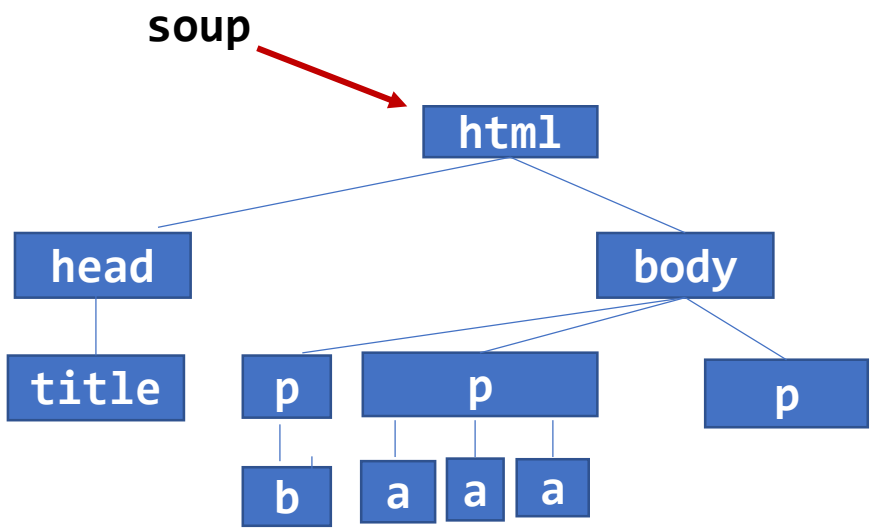
```
for a in soup.find_all('a', id=True):
    print(a.get('id'))
```

```
link1
link2
link3
```

get the text in the 'id' attribute

Pretty Printing

- `prettify()`



htmlparser가 만든 parse tree

```
print(soup.prettify())
```

```
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    The Dormouse's story
   </b>
  </p>
  <p class="story">
   Once upon a time there were three little sisters; and their names were
   <a class="sister" href="http://example.com/elsie" id="link1">
    Elsie
   </a>
   ,
   <a class="sister" href="http://example.com/lacie" id="link2">
    Lacie
   </a>
   and
   <a class="sister" href="http://example.com/tillie" id="link3">
    Tillie
   </a>
   ;
and they lived at the bottom of a well.
  </p>
  <p class="story">
   ...
  </p>
 </body>
</html>
```

Saving HTML File from a Web Site

Simple writing

```
import requests

url = 'https://cse.snu.ac.kr'
result = requests.get(url)

fp = open('cse.html', 'w')
fp.write(result.text)
fp.close()
```

Pretty Printing

```
import requests
from bs4 import BeautifulSoup

url = 'https://cse.snu.ac.kr'
result = requests.get(url)

soup = BeautifulSoup(result.text, 'html.parser')
fp = open('cse.html', 'w')
fp.write(soup.prettify())
fp.close()
```

Example

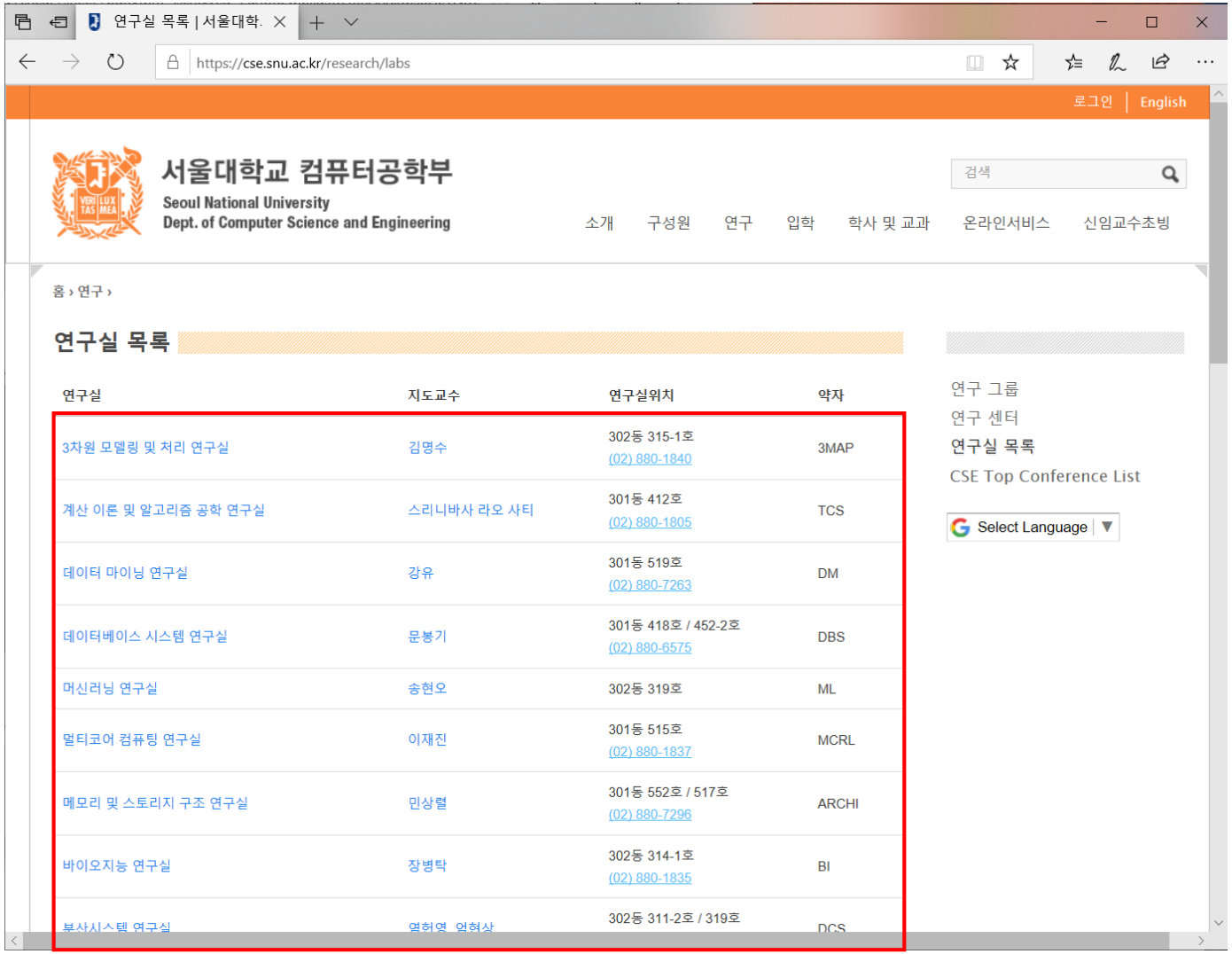
- 서울대학교 컴퓨터공학부 사이트에서 연구실 정보가 담긴 page를 scraping 한다.

- https://cse.snu.ac.kr/research/labs

- 위 페이지에서 "연구실, 지도교수, 연구실위치, 약자" 정보를 추출한다.

- 301동, 302동에 속한 연구실 목록으로 분류한다.

Example: Target Page



Example: Table in HTML

```
<div class="content">
  <div class="view view-research-labs view-id-research_labs view-display-id-page view-dom-id-3686693d59000d614c0214a4bac5eb34">



    <div class="view-content">
    <table class="views-table cols-4" >
      <thead>
    <tr>
                <th class="views-field views-field-title" >
          연구실          </th>
                <th class="views-field views-field-field-faculty" >
          지도교수          </th>
                <th class="views-field views-field-field-office" >
          연구실위치          </th>
                <th class="views-field views-field-field-abbreviation" >
          약자          </th>
            </tr>
  </thead>
  <tbody>
        <tr class="odd views-row-first">
                <td class="views-field views-field-title" >
          <a href="/lab/3%EC%B0%A8%EC%9B%90-%EB%AA%A8%EB%8D%B8%EB%A7%81-%EB%B0%8F-%EC%B2%98%EB%A6%AC-%EC%97%B0%EA%B5%AC%EC%8B%A4">3차원 모델링 및 처리 연구실</a>          </td>
                <td class="views-field views-field-field-faculty" >
          <a href="/professor/%EA%B9%80%EB%AA%85%EC%88%98">김명수</a>          </td>
                <td class="views-field views-field-field-office" >
          302동 315-1호<br />(02) 880-1840          </td>
                <td class="views-field views-field-field-abbreviation" >
          3MAP          </td>
            </tr>
```

Example: Extracting Information

```
1  import requests
2  import pandas as pd
3  from bs4 import BeautifulSoup
4
5  def get_cse_labs(url):
6      response = requests.get(url)
7      soup = BeautifulSoup(response.text, 'html.parser')
8      content = soup.find(id='content')
9      headers = [ header.text.strip() for header in content.find_all('th') ]
10     rows = content.find('tbody').find_all('tr')
11     items = [{k:v.get_text().strip() for k, v in zip(headers, row.find_all('td'))}
12              for row in rows]
13     return items
```

headers `['연구실', '지도교수', '연구실위치', '약자']`

items *(list of dict)*


```
[{'연구실': '3차원 모델링 및 처리 연구실', '지도교수': '김명수', '연구실위치': '302동 315-1호(02) 880-1840', '약자': '3MAP'}, {'연구실': '계산 이론 및 알고리즘 공학 연구실', '지도교수': '스리니바사 라오 사티', '연구실위치': '301동 412호(02) 880-1805', '약자': 'TCS'}, {'연구실': '데이터 마이닝 연구실', '지도교수': '강유', '연구실위치': '301동 519호(02) 880-7263', '약자': 'DM'}, {'연구실': '데
```

Example: To Pandas

```
url = 'https://cse.snu.ac.kr/research/labs'
table = get_cse_labs(url)
df = pd.DataFrame(table, columns=table[0].keys())
df['연구동'] = df['연구실위치'].map(lambda s: s.split()[0])
df1 = df[df['연구동']=='301동']
df2 = df[df['연구동']=='302동']
df
```

	연구실	지도교수	연구실위치	약자	연구동
0	3차원 모델링 및 처리 연구실	김명수	302동 315-1호(02) 880-1840	3MAP	302동
1	계산 이론 및 알고리즘 공학 연구실	스리니바사 라오 사티	301동 412호(02) 880-1805	TCS	301동
2	데이터 마이닝 연구실	강유	301동 519호(02) 880-7263	DM	301동
3	데이터베이스 시스템 연구실	문봉기	301동 418호 / 452-2호(02) 880-6575	DBS	301동
4	머신러닝 연구실	송현오	302동 319호	ML	302동
5	멀티코어 컴퓨팅 연구실	이재진	301동 515호(02) 880-1837	MCRL	301동
6	메모리 및 스토리지 구조 연구실	민상렬	301동 552호 / 517호(02) 880-7296	ARCHI	301동
7	바이오지능 연구실	장병탁	302동 314-1호(02) 880-1835	BI	302동
8	분산시스템 연구실	염헌영, 엄현상	302동 311-2호 / 319호(02) 880-1856	DCS	302동
9	생물정보 및 생명정보 연구실	김선	301동 516호(02) 880-1784	BHI	301동

Example: Results

```
df1
```

	연구실	지도교수	연구실위치	약자	연구동
1	계산 이론 및 알고리즘 공학 연구실	스리니바사 라오 사티	301동 412호(02) 880-1805	TCS	301동
2	데이터 마이닝 연구실	강유	301동 519호(02) 880-7263	DM	301동
3	데이터베이스 시스템 연구실	문봉기	301동 418호 / 452-2호(02) 880-6575	DBS	301동
5	멀티코어 컴퓨팅 연구실	이재진	301동 515호(02) 880-1837	MCRL	301동
6	메모리 및 스토리지 구조 연구실	민상렬	301동 552호 / 517호(02) 880-7296	ARCHI	301동
9	생물정보 및 생명정보 연구실	김선	301동 516호(02) 880-1784	BHI	301동
14	시스템 소프트웨어 및 구조 연구실	김진수	301동 517호(02) 880-7296	CSL	301동
15	실시간 유비쿼터스 시스템 연구실	이창건	301동 415호(02) 880-2562	RUBIS	301동
16	아키텍처 및 코드 최적화 연구실	이재욱	301동 554-1호(02) 880-1836	ARC	301동
17	양자정보 및 양자컴퓨팅 연구실	김태현	301동 416호(02) 880-4165	QUIQCL	301동
20	인간 중심 컴퓨터 시스템 연구실	이영기	301동 416호(02) 880-4165	HCS	301동
21	인터넷 데이터베이스 연구실	김형주	301동 453호(02) 880-1830	IDB	301동
22	인터넷 융합 및 보안 연구실	최양희, 권태경	301동 518호(02) 880-9147	NCSL	301동
24	지능형 데이터 시스템 연구실	이상구	301동 420호(02) 880-1859	IDS	301동
27	컴퓨터 시스템 및 플랫폼 연구실	버나드 에거	301동 419호(02) 880-1819	CSAP	301동
28	컴퓨터이론 및 응용 연구실	박근수	301동 414호(02) 880-1828	CTA	301동
30	통합설계 및 병렬 처리 연구실	하순회	301동 455-1호(02) 880-7292	CAP	301동

```
df2
```

	연구실	지도교수	연구실위치	약자	연구동
0	3차원 모델링 및 처리 연구실	김명수	302동 315-1호(02) 880-1840	3MAP	302동
4	머신러닝 연구실	송현오	302동 319호	ML	302동
7	바이오지능 연구실	장병탁	302동 314-1호(02) 880-1835	BI	302동
8	분산시스템 연구실	염헌영, 엄현상	302동 311-2호 / 319호(02) 880-1856	DCS	302동
10	소셜정보망 연구실	김종권	302동 310-1호(02) 880-1858	SCONE	302동
11	소프트웨어 원리 연구실	허충길	302동 312-2호(02) 880-1865	SF	302동
12	소프트웨어 플랫폼 연구실	전병곤	302동 420호02) 880-1611	SPL	302동
13	시각 및 학습 연구실	김건희	302동 317호(02) 880-7289	VL	302동
18	운동 연구실	이제희	302동 312-1호(02) 880-1864	MRL	302동
19	이동 컴퓨팅 및 통신 연구실	전화숙	302동 313-1호(02) 880-1841	MCCL	302동
23	임베디드 시스템 연구실	김지홍	302동 315-2호(02) 880-1861	CARES	302동
25	최적화 및 금융공학 연구실	문병로	302동 313-2호(02) 880-1851	OPT	302동
26	컴퓨터 그래픽스 및 이미지 처리 연구실	신영길	302동 320호(02) 880-1860	CGIP	302동
31	프로그래밍 연구실	이광근	302동 312-2호(02) 880-1865	ROPAS	302동
32	휴먼-컴퓨터 인터액션 연구실	서진욱	302동 314-2호(02) 880-7044	HCI	302동

Example: Pandas to HTML

```
df1.to_html()
```

```
'<table border="1" class="dataframe">\n  <thead>\n    <tr style="text-align: right;">\n      <
th></th>\n      <th>연구실</th>\n      <th>지도교수</th>\n      <th>연구실위치</th>\n      <th>
약자</th>\n      <th>연구동</th>\n    </tr>\n  </thead>\n  <tbody>\n    <tr>\n      <th>1</th>
\n      <td>계산 이론 및 알고리즘 공학 연구실</td>\n      <td>스리니바사 라오 사티</td>\n      <td
>301동 412호(02) 880-1805</td>\n      <td>TCS</td>\n      <td>301동</td>\n    </tr>\n    <tr>
\n      <th>2</th>\n      <td>데이터 마이닝 연구실</td>\n      <td>강유</td>\n      <td>301동 51
9호(02) 880-7263</td>\n      <td>DM</td>\n      <td>301동</td>\n    </tr>\n    <tr>\n      <th
>3</th>\n      <td>데이터베이스 시스템 연구실</td>\n      <td>문봉기</td>\n      <td>301동 418호
/ 452-2호(02) 880-6575</td>\n      <td>DBS</td>\n      <td>301동</td>\n    </tr>\n    <tr>\n
<th>5</th>\n      <td>멀티코어 컴퓨팅 연구실</td>\n      <td>이재진</td>\n      <td>301동 515호
(02) 880-1837</td>\n      <td>MCRL</td>\n      <td>301동</td>\n    </tr>\n    <tr>\n      <th>
6</th>\n      <td>메모리 및 스토리지 구조 연구실</td>\n      <td>민상렬</td>\n      <td>301동 552
호 / 517호(02) 880-7296</td>\n      <td>ARCHI</td>\n      <td>301동</td>\n    </tr>\n    <tr>
\n      <th>9</th>\n      <td>생물정보 및 생명정보 연구실</td>\n      <td>김선</td>\n      <td>3
01동 516호(02) 880-1784</td>\n      <td>BHI</td>\n      <td>301동</td>\n    </tr>\n    <tr>\n
<th>14</th>\n      <td>시스템 소프트웨어 및 구조 연구실</td>\n      <td>김진수</td>\n      <td>30
1동 517호(02) 880-7296</td>\n      <td>CSL</td>\n      <td>301동</td>\n    </tr>\n    <tr>\n
<th>15</th>\n      <td>실시간 유비쿼터스 시스템 연구실</td>\n      <td>이창건</td>\n      <td>301
동 415호(02) 880-2562</td>\n      <td>RUBIS</td>\n      <td>301동</td>\n    </tr>\n    <tr>\n
<th>16</th>\n      <td>아키텍처 및 코드 최적화 연구실</td>\n      <td>이재욱</td>\n      <td>301
```