



DATA ANALYSIS STUDY

TEAM 3

유지원, 송지원, 김관엽





TOPIC

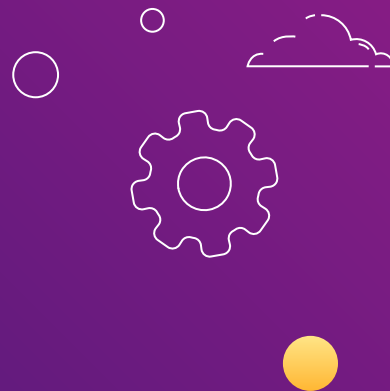
시카고 샌드위치 맛집 분석하기





01

데이터 수집



Beautiful Soup를 통한 Wep Scrapping

데이터 수집

In [1]:

```
from bs4 import BeautifulSoup
```

-import



-html 읽기



In [2]:

```
page = open("../data/03. test_first.html",  
soup = BeautifulSoup(page, 'html.parser')  
print(soup.prettify())
```

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>  
      Very Simple HTML Code by PinkWink  
    </title>  
  </head>  
  <body>  
    <div>  
      <p class="inner-text first-item" i  
        Happy PinkWink.  
        <a href="http://www.pinkwink.kr"  
          PinkWink  
        </a>  
      </p>  
      <p class="inner-text second-item">  
        Happy Data Science.  
        <a href="https://www.python.org"  
          Python  
        </a>  
      </p>  
    </div>  
    <p class="outer-text first-item" ic  
      <b>  
        Data Science is funny.  
      </b>  
    </p>  
    <p class="outer-text">  
      <b>  
        All I need is Love.  
      </b>  
    </p>  
  </body>
```



Beautiful Soup를 통한 Web Scraping

데이터 수집

In [3]:

```
list(soup.children)
```

Out[3]:

```
['html',
 '\n',
 <html>
 <head>
 <title>Very Simple HTML Code by PinkV
 </head>
 <body>
 <div>
 <p class="inner-text first-item" id=
   Happy PinkWink.
   <a href="http://www.

</p>
<p class="inner-text second-item">
   Happy Data Science.
   <a href="https://www

</p>
</div>
<p class="outer-text first-item" id=
<b>
   Data Science is funr
</b>
</p>
<p class="outer-text">
<b>
   All I need is Love.
</b>
</p>
</body>
```

In [4]:

```
html = list(soup.children)[2]
html
```

Out[4]:

```
<html>
<head>
<title>Very Simple HTML Code by PinkV
</head>
<body>
<div>
<p class="inner-text first-item" id='
   Happy PinkWink.
   <a href="http://www.p

</p>
<p class="inner-text second-item">
   Happy Data Science.
   <a href="https://www.

</p>
</div>
<p class="outer-text first-item" id='
<b>
   Data Science is funny
</b>
</p>
<p class="outer-text">
<b>
   All I need is Love.
</b>
</p>
</body>
</html>
```

soup이라는 변수에서 한 단계 아래에서 포함된 태그들을 알고 싶은 경우, children 속성을 사용

soup는 문서 전체를 저장한 변수이므로 그 안에서 html 태그에 접속하고 싶으면 위와 같이 접근!

Beautiful Soup를 통한 Wep Scrapping

데이터 수집

In [5]:

```
list(html.children)
```

Out[5]:

```
['\n',
<head>
<title>Very Simple HTML Code by Pink
</head>,
'\n',
<body>
<div>
<p class="inner-text first-item" id=
    Happy PinkWink.
    <a href="http://www.

</p>
<p class="inner-text second-item">
    Happy Data Science.
    <a href="https://www

</p>
</div>
<p class="outer-text first-item" id=
<b>
    Data Science is funn
</b>

</p>
<p class="outer-text">
<b>
    All I need is Love.
</b>

</p>
</body>,
'\n']
```

In [6]:

```
body = list(html.children)[3]
body
```

Out[6]:

```
<body>
<div>
<p class="inner-text first-item" id='
    Happy PinkWink.
    <a href="http://www.

</p>
<p class="inner-text second-item">
    Happy Data Science.
    <a href="https://www.

</p>
</div>
<p class="outer-text first-item" id='
<b>
    Data Science is funny
</b>

</p>
<p class="outer-text">
<b>
    All I need is Love.
</b>

</p>
</body>
```



children과 parent를 이용해서 태그 조사



Beautiful Soup를 통한 Web Scraping

데이터 수집

But, 한 번에 나타내어 바로 찾을 수도
있음.



In [7]:

```
soup.body
```

Out[7]:

```
<body>
<div>
<p class="inner-text first-item" id='
    Happy PinkWink.
    <a href="http://www.p

</p>
<p class="inner-text second-item">
    Happy Data Science.
    <a href="https://www.

</p>
</div>
<p class="outer-text first-item" id='
<b>
    Data Science is funny
</b>
</p>
<p class="outer-text">
<b>
    All I need is Love.
</b>
</p>
</body>
```



Beautiful Soup를 통한 Wep Scrapping

데이터 수집

-body 태그 안에 children의 리스트 확인 가능

In [8]:

```
list(body.children)
```

Out[8]:

```
['\n',  
 <div>  
   <p class="inner-text first-item" id=  
       Happy PinkWink.  
       <a href="http://www.  
   </p>  
   <p class="inner-text second-item">  
       Happy Data Science.  
       <a href="https://ww  
   </p>  
 </div>,  
 '\n',  
 <p class="outer-text first-item" id=  
 <b>  
       Data Science is funr  
   </b>  
 </p>,  
 '\n',  
 <p class="outer-text">  
 <b>  
       All I need is Love.  
   </b>  
 </p>,  
 '\n']
```



Beautiful Soup를 통한 Wep Scrapping

데이터 수집

-모든 p 태그 찾기(find_all)

-한 태그만 찾기(find)

In [10]:

```
soup.find_all('p')
```

Out[10]:

```
[<p class="inner-text first-item" id=
    Happy PinkWink.
    <a href="http://www.
  </p>,
  <p class="inner-text second-item">
    Happy Data Science.
    <a href="https://ww
  </p>,
  <p class="outer-text first-item" id=
  <b>
    Data Science is funr
  </b>
  </p>,
  <p class="outer-text">
  <b>
    All I need is Love.
  </b>
  </p>]
```

In [14]:

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

Out[14]:

```
<p class="inner-text first-item" id='
    Happy PinkWink.
    <a href="http://www.f
  </p>
```



Beautiful Soup를 통한 Wep Scrapping

데이터 수집

-class 이름으로만 outer_text 찾기

-p태그의 class가 outer_text인 것 찾기

-id가 first인 태그들 찾기

In [12]:

```
soup.find_all(class_ = 'outer-text')
```

Out[12]:

```
[<p class="outer-text first-item" id=
<b>
    Data Science is funr
</b>
</p>,
<p class="outer-text">
<b>
    All I need is Love.
</b>
</p>]
```

In [11]:

```
soup.find_all('p', class_ = 'outer-text')
```

Out[11]:

```
[<p class="outer-text first-item" id=
<b>
    Data Science is funr
</b>
</p>,
<p class="outer-text">
<b>
    All I need is Love.
</b>
</p>]
```

In [13]:

```
soup.find_all(id = "first")
```

Out[13]:

```
[<p class="inner-text first-item" id=
    Happy PinkWink.
    <a href="http://www.
</p>]
```



Beautiful Soup를 통한 Web Scraping

데이터 수집

-next_sibling

In [18]:

```
soup.head.next_sibling.next_sibling
```

Out[18]:

```
<body>
<div>
<p class="inner-text first-item" id='
    Happy PinkWink.
    <a href="http://www.p
  </p>
<p class="inner-text second-item">
    Happy Data Science.
    <a href="https://www.

</p>
</div>
<p class="outer-text first-item" id='
<b>
    Data Science is funny
  </b>
</p>
<p class="outer-text">
<b>
    All I need is Love.
  </b>
</p>
</body>
```

-get_text()

In [21]:

```
for each_tag in soup.find_all('p'):
    print(each_tag.get_text())
```

Happy PinkWink.
PinkWink

Happy Data Science.
Python

Data Science is funny

All I need is Love.

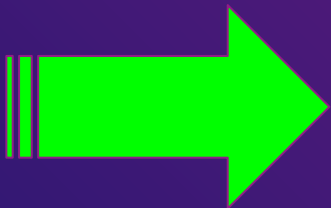


Beautiful Soup를 통한 Wep Scrapping

데이터 수집

-a 태그 찾기(클릭 가능한 링크)

-링크 주소 얻기



In [23]:

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

Out[23]:

```
[<a href="http://www.pinkwink.kr" id=  
  <a href="https://www.python.org" id=
```

In [24]:

```
for each in links:  
    href = each['href']  
    text = each.string  
    print(text + ' -> ' + href)
```

PinkWink -> http://www.pinkwink.kr

Python -> https://www.python.org



크롬 개발자 도구를 이용해서 원하는 태그 찾기

데이터 수집

finance.naver.com/marketindex/

NAVER 금융 종목명·지수명·펀드명·환율명·원자재명 입력 통합검색 로그인

금융 홈 국내증시 해외증시 **시장지표** 펀드 리서치 뉴스 MY

블로그 이웃 태그 경품 받아주세요!
BESPOKE 가전부터 백화점 상품권까지! [더 알아보기 >](#)

환전 고시 환율	국제 시장 환율	유가·금시세
미국 USD 1,185.50원 ▲3.50 [圖] 2021.11.04 20:01 하나은행 기준 고시회차223회	달러/일본 엔 114.0800엔 ▲0.2600 [圖] 2021.11.03 모닝스타 기준	유가·금시세 페이지를 다른 이름으로 저장... Ctrl+S 바로가기 만들기... 창 이름 지정... 인터넷 사용 기록 삭제... Ctrl+Shift+Del 확장 프로그램 작업 관리자 Shift+Esc 개발자 도구 Ctrl+Shift+I
일본 JPY(100엔) 1,040.41원 ▲2.34	유로/달러 1.1578달러 ▼0.0012	휘발유 1794.95원 ▲3.66
유럽연합 EUR 1,368.78원 ▼1.22	영국파운드/달러 1.3660달러 ▲0.0040	국제 금 1763.6달러 ▼25.10
중국 CNY 185.23원 ▲0.44	달러인덱스 93.8500 ▼0.2200	국내 금 67558.53원 ▼83.95

주요뉴스 더보기

- 원자재 물류비騰 원가 폭탄에...침대 매트 11.04 17:51
- 원자재 폭등해도 '어닝서프라이즈'...쌍용C&S 11.04 17:00
- 국고채 금리 대체로 하락...3년물은 0.4b 11.04 17:23
- [외환마감]美 테이퍼링 '도비시'의 재해석 11.04 16:04

국내 시장 급리 CD(91일) 1.13 ▲0.01

크롬 개발자 도구를 이용해서 원하는 태그 찾기

데이터 수집

NAVER 금융

종목명 지수명 펀드명 환율명 원자재명 입력

통합검색

로그인

금융 홈

국내증시

해외증시

시장지표

펀드

리서치

뉴스

MY

블로그 이웃 맺고 경품 받아보세요!

BESPOKE 가전부터 백화점 상품권까지!

더 알아보기 >

환전 고시 환율

국제 시장 환율

유가·금시세

미국USD

1,185.50원 ▲3.50

span.value 70.08 × 20

달러/일본엔

114.0800엔 ▲0.2600

WTI

80.86달러 ▼3.05

일본JPY(100엔)

1,040.41원 ▲2.34

유로/달러

1.1578달러 ▼0.0012

휘발유

1794.95원 ▲3.66

유럽연합EUR

1,368.78원 ▼1.22

영국파운드/달러

1.3660달러 ▲0.0040

국제 금

1763.6달러 ▼25.10

중국CNY

185.23원 ▲0.44

달러인덱스

93.8500 ▼0.2200

국내 금

67558.53원 ▼83.95

주요뉴스

더보기 >

원자재 물류비값 원가 폭탄에...침대 매트

11.04 17:51

국고채 금리 대체로 하락...3년물은 0.4b

11.04 17:23

원자재 폭등에도 '어닝스트라이즈'...쌍용C&S

11.04 17:00

[외환마감]美 테이퍼링 '도비시'의 재해석

11.04 16:04

국내 시장 정리

CD (91일)

1.13

▲ 0.01

DevTools is now available in Korean!

Always match Chrome's language Switch DevTools to Korean Don't show again

Elements Console Sources Network >> 6 6

<!-- data -->

<div class="data">

<ul class="data_lst" id="exchangelist">

<li class="on">

<h3 class="h_lst"></h3>

<div class="head_info">

1,185.50 == \$0

3.50

상승

::after

... liv.data ul#exchangelist.data_lst li.on a.head.usd div.head_info.point_up span.value ...

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Filter

element.style {

}

.market_data .data_lst li.on .value {

height: 20px;

margin: 1px -4px 0 0;

}

.market_data .head_info span {

display: inline-block;

vertical-align: middle;

}

.market_data .value {

margin-right: -3px;

}

Inherited from div.head_info.point_up

-urlopen

In [25]:

```
from urllib.request import urlopen
```

In [26]:

```
url = "https://finance.naver.com/marketindex"
page = urlopen(url)

soup = BeautifulSoup(page, "html.parser")
print(soup.prettify())
```

In [27]:

```
soup.find_all('span', 'value')[0].string
```

Out[27]:

```
'1,237.70'
```



CHICAGO

NEWS & POLITICS

DINING & DRINKING

CITY LIFE

CULTURE

REAL ESTATE

STYLE

TOP DOCS

The 50 Best Sandwiches in Chicago

Our list of Chicago's 50 best sandwiches, ranked in order of deliciousness

OCTOBER 9, 2012, 6:12 PM



시카고 샌드위치 맛집 소개 사이트에 접근하여 데이터 수집하기

데이터 수집

The screenshot shows a web browser displaying the Chicago Magazine website. The URL is `chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/`. The page title is "CHICAGO". A red box highlights a list item in the "div.sammyListing" container, which contains the text "BLT", "Old Oak Tap", and "Read more". A blue arrow points from this list item to the DevTools Elements panel on the right. In the Elements panel, the corresponding HTML structure is highlighted, showing a `div class="sammyRank">1</div>` and a `div class="sammyListing">...</div>` element. The Styles panel shows the default user agent styles for the `div` element.

CHICAGO

div.sammyListing 506.51 x 72

BLT
Old Oak Tap
Read more

2
Fried Bologna
Au Cheval
Read more

3
Woodland Mushroom
Xoco
Read more

4
Roast Beef
Al's Deli
Read more

5
PB&L
Publian Quality Meats
Read more

6
Belgian Chicken Curry Salad
Hendrickx Belgian Bread Crafter
Read more

7
Lobster Roll
Acadia
Read more

8

DevTools is now available in Korean!

Always match Chrome's language Switch DevTools to Korean Don't show again

Elements Console Sources Network >> 18

```
<p>...</p>
<section class="related-content pull-right">...</section>
<p>...</p>
<p>...</p>
<p>...</p>
<p>...</p>
<p>...</p>
<div class="sammy" style="position: relative;">
  <div class="sammyRank">1</div>
  <div class="sammyListing">...</div> == $0
</div>
<div class="sammy" style="position: relative;">...</div>
<div class="sammy" style="position: relative;">...</div>
<div class="sammy" style="position: relative;">...</div>
<div class="sammy" style="position: relative;">...</div>
<div class="sammy" style="position: relative;">...</div>
<div class="sammy" style="position: relative;">...</div>
```

... -grid-100.mobile-prefix-0.mobile-suffix-0 div.article-body div.sammy div.sammyListing ...

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls + [4]

```
element.style {
}

a, abbr, acronym, address, applet, big, blockquote, style.min.c...ver=3.0.2:1
body, caption, cite, code, dd, del, dfn, div, dl, dt, em, fieldset, font, form,
h1, h2, h3, h4, h5, h6, html, iframe, ins, kbd, label, legend, li, object, ol,
p, pre, q, s, samp, small, span, strike, strong, sub, sup, table, tbody, td,
tfoot, th, thead, tr, tt, ul, var {
  border: 0;
  margin: 0;
  padding: 0;
}

div {
  display: block;
}

Inherited from div#content.site-content
```

```
In [28]: from bs4 import BeautifulSoup
         from urllib.request import urlopen

         # url_base와 url_sub으로 나뉘놓은 이유는 코드가 너무 길어짐을 방지하기 위해서!
         url_base = 'http://www.chicagomag.com'
         url_sub = '/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/'
         url = url_base + url_sub

         html = urlopen(url)
         soup = BeautifulSoup(html, "html.parser")

         soup
```

```
In [29]: print(soup.find_all('div', 'sammy'))
```



```
In [30]: len(soup.find_all('div', 'sammy'))
```

```
Out[30]: 50
```

맛집 50개이므로 길이 일치

```
In [31]: print(soup.find_all('div', 'sammy')[0])
```

```
<div class="sammy" style="position: relative;">
```

```
<div class="sammyRank">1</div>
```

```
<div class="sammyListing"><a href="/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/"><b>BLT</b><br>Old Oak Tap<br>
```

```
<em>Read more</em> </br></br></a></div>
```

```
</div>
```

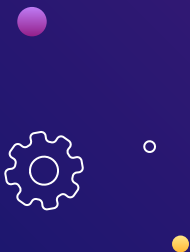
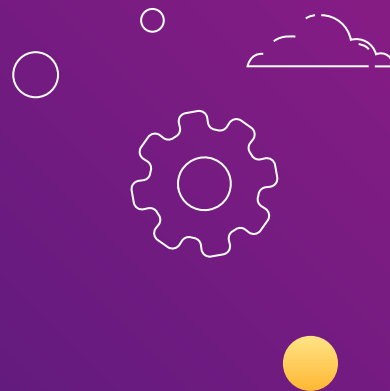
원하는 정보 추출 완료





02

데이터 가공



원하는 데이터 추출하기

- 태그 찾기

```
[ ] tmp_one = soup.find_all('div', 'sammy')[0]  
    type(tmp_one)
```

```
bs4.element.Tag
```

```
[ ] tmp_one.find(class_='sammyRank')
```

```
<div class="sammyRank">1</div>
```

- get_text()

```
[ ] tmp_one.find(class_='sammyRank').get_text()
```

```
'1'
```

```
[ ] tmp_one.find(class_='sammyListing').get_text()
```

```
'BLT#n0ld Oak Tap#nRead more '
```



원하는 데이터 추출하기

- 접근 주소 추출

```
[ ] tmp_one.find('a')['href']
```

```
    '/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/'
```

- 정규식(Regular Express) - split

```
[ ] import re

    tmp_string = tmp_one.find(class_='sammyListing').get_text()

    re.split(('\\n|\\r\\n'), tmp_string)

    print(re.split(('\\n|\\r\\n'), tmp_string)[0])
    print(re.split(('\\n|\\r\\n'), tmp_string)[1])
```

```
BLT
Old Oak Tap
```



리스트에 데이터 저장하기

- .append 명령어로 리스트에 넣기

```
[ ] from urllib.parse import urljoin
```

```
▶ rank = []  
main_menu = []  
cafe_name = []  
url_add = []  
  
list_soup = soup.find_all('div', 'sammy')  
  
for item in list_soup:  
    rank.append(item.find(class_='sammyRank').get_text())  
  
    tmp_string = item.find(class_='sammyListing').get_text()  
  
    main_menu.append(re.split(('#\n|#r#\n'), tmp_string)[0])  
    cafe_name.append(re.split(('#\n|#r#\n'), tmp_string)[1])  
  
    url_add.append(urljoin(url_base, item.find('a')['href']))
```



리스트에 데이터 저장하기

- 리스트 출력해보고 잘 들어갔는지 확인하기

```
rank[:5]
```

```
[ ] main_menu[:5]
```

```
[ ] cafe_name[:5]
```

```
[ ] url_add[:5]
```

```
[ ] len(rank), len(main_menu), len(cafe_name), len(url_add)
```

```
['1', '2', '3', '4', '5']
```

```
['BLT', 'Fried Bologna', 'Woodland Mushroom', 'Roast Beef', 'PB&L']
```

```
['Old Oak Tap', 'Au Cheval', 'Xoco', 'Al's Deli', 'Publican Quality Meats']
```

```
['http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-In-Chicago-Old-Oak-Tap-BLT/',  
'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-In-Chicago-Au-Cheval-Fried-Bologna/',  
'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-In-Chicago-Xoco-Woodland-Mushroom/',  
'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-In-Chicago-Als-Deli-Roast-Beef/',  
'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-In-Chicago-Publican-Quality-Meats-PB-L/']
```

```
(50, 50, 50, 50)
```



pandas-DataFrame으로 데이터 추출하기

- 각 칼럼의 이름과 순서 정리

```
[ ]
```

```
import pandas as pd
```

```
data = {'Rank':rank, 'Menu':main_menu, 'Cafe':cafe_name, 'URL':url_add}
```

```
df = pd.DataFrame(data)
```

```
df.head()
```

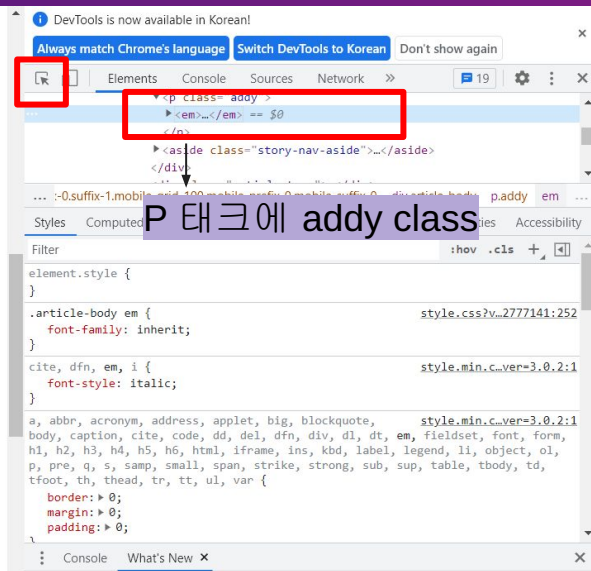
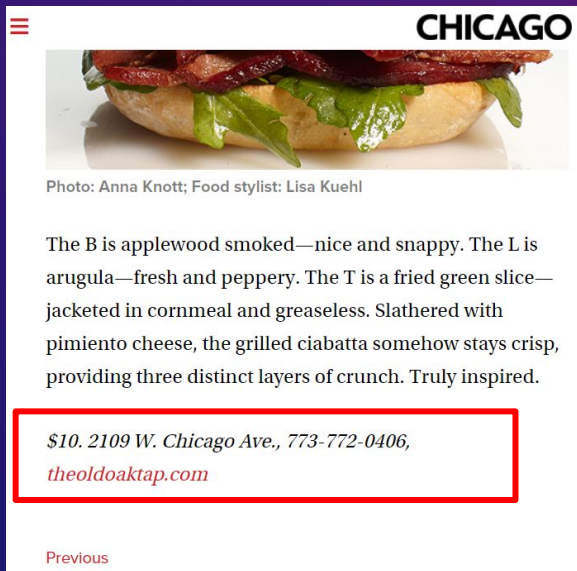
	Rank	Menu	Cafe	URL
0	1	BLT	Old Oak Tap	http://www.chicagomag.com/Chicago-Magazine/Nov...
1	2	Fried Bologna	Au Cheval	http://www.chicagomag.com/Chicago-Magazine/Nov...
2	3	Woodland Mushroom	Xoco	http://www.chicagomag.com/Chicago-Magazine/Nov...
3	4	Roast Beef	Al's Deli	http://www.chicagomag.com/Chicago-Magazine/Nov...
4	5	PB&L	Publican Quality Meats	http://www.chicagomag.com/Chicago-Magazine/Nov...



세부 메뉴 정보 가져오기

·가게 주소, 대표 샌드위치 가격

- 해당 태그 찾기: 우측 상단 ... >도구 더보기>개발자 도구>



세부 메뉴 정보 가져오기

- 가격, 주소, 전화번호 추출하기
- 사이트에서 사용자를 bot으로 인식하여 차단 -> header정보 추가해서 해결!

```
[61] #html = urlopen(df['URL'][0])-> 오류 발생
      req = Request(df['URL'][0], headers={'User-Agent': 'Mozilla/5.0'})#header 정보 추가하기
      html = urlopen(req).read()
      soup_tmp = BeautifulSoup(html, "html.parser")
      print(soup_tmp.find('p', 'addy'))
```

```
<p class="addy">
<em>$10. 2109 W. Chicago Ave., 773-772-0406, <a href="http://www.theoldoaktap.com/">theoldoaktap.com</a></em></p>
```

- get_text(), split() 으로 정리하기

```
▶ price_tmp = soup_tmp.find('p', 'addy').get_text()
   price_tmp.split()
```

```
['$10.', '2109', 'W.', 'Chicago', 'Ave.', '773-772-0406', 'theoldoaktap.com']
```



세부 메뉴 정보 가공하기

- 개별 세부 정보 추출하기
- .split() : 대표 샌드위치 가격 추출

```
['$10.', '2109', 'W.', 'Chicago', 'Ave.', ' ', '773-772-0406', ' ', 'theoldoaktap.com']
```

```
[64] price_tmp.split()[0][:~1]

'$10'
```

- Join 명령어 : 가게 주소 합치기

```
['$10.', '2109', 'W.', 'Chicago', 'Ave.', ' ', '773-772-0406', ' ', 'theoldoaktap.com']
```

```
[65] ' '.join(price_tmp.split()[1:-2])

'2109 W. Chicago Ave.,'
```



상태 진행바 적용 후, 페이지에서 세부 정보 추출하기


- Tqdm 모듈: 50개의 정보를 추출하는 동안 진행 상태를 알 수 있음

```
from tqdm import tqdm_notebook
```

```
price = []  
address = []
```

```
for n in tqdm_notebook(df.index):  
    req = Request(df['URL'][n], headers={'User-Agent': 'Mozilla/5.0'})  
    html = urlopen(req)  
    soup_tmp = BeautifulSoup(html, 'lxml')  
  
    gettings = soup_tmp.find('p', 'addy').get_text()  
  
    price.append(gettings.split()[0][:1])  
    address.append(' '.join(gettings.split()[1:-2]))
```

```
... /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: TqdmDeprecationWarning: This function will be removed in tqdm==5.0.0  
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
```

4%  2/50 [00:00<00:12, 3.70it/s]



100%  50/50 [00:29<00:00, 3.15it/s]

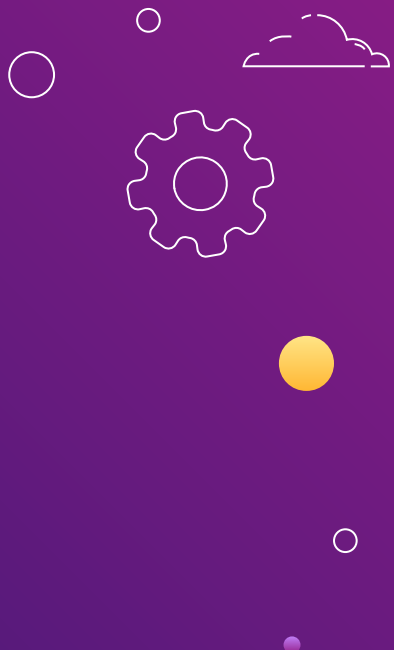




03

데이터 시각화

: 맛집의 위치를 지도에 표시하기



1. 모듈 import하기

필요한 모듈 import하기

```
!pip install googlemaps  
  
import folium  
import pandas as pd  
import googlemaps  
import numpy as np
```

- folium
- Googlemaps
- numpy



2. (pandas) 가공해둔 데이터 가져오기

df에 저장해둔 데이터 읽어오기

```
df = pd.read_csv('../data/03. best_sandwiches_list_chicago2.csv', index_col=0)  
df.head(10)
```

	Cafe	Menu	Price	Address
Rank				
1	Old Oak Tap	BLT	\$10	2109 W. Chicago Ave.,
2	Au Cheval	Fried Bologna	\$9	800 W. Randolph St.,
3	Xoco	Woodland Mushroom	\$9.50	445 N. Clark St.,
4	Al's Deli	Roast Beef	\$9.40	914 Noyes St., Evanston,
5	Publican Quality Meats	PB&L	\$10	825 W. Fulton Mkt.,
6	Hendrickx Belgian Bread Crafter	Belgian Chicken Curry Salad	\$7.25	100 E. Walton
7	Acadia	Lobster Roll	\$16	1639 S. Wabash Ave.,
8	Birchwood Kitchen	Smoked Salmon Salad	\$10	2211 W. North Ave.,
9	Cemitas Puebla	Atomica Cemitas	\$9	3619 W. North Ave.,
10	Nana	Grilled Laughing Bird Shrimp and Fried Po' Boy	\$17	3267 S. Halsted St.,



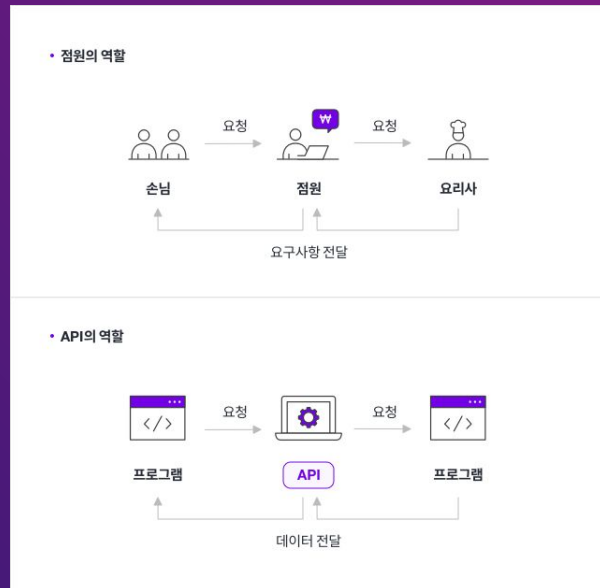
3. (googlemaps) API 키 발급 받기

API란?

- Application Programming Interface
- 프로그램들이 서로 상호작용하는 것을 도와주는 매개체

googlemaps

- 통합 Google Cloud Platform에서 서비스
- 개발 API KEY 받고 이용 가능



Google Cloud Platform My First Project

제품 및 리소스 검색

Google Maps Platform 사용자 인증 정보

모든 Google Maps Platform API + 사용자 인증 정보 만들기

알아보기

개요
API
측정항목
활동량
사용자 인증 정보
지원
Locator Plus 솔루션
자동 완성 솔루션 신규
지도 관리
지도 스타일 신규

모든 사용자 인증 정보를 보려면 [API 및 서비스의 사용자 인증 정보로 이동](#)하세요.

⚠ 애플리케이션에 대한 정보를 포함하여 OAuth 동의 화면을 구성해야 합니다. [동의 화면 구성](#)

API 키

이름	생성일	제한 사항	키	작업
✓ API 키 2개	2021. 11. 4.	Geocoding API	AIzaSyBiA...UQT7B8HuZo	
⚠ Maps API Key	2021. 11. 4.	제한 없음	AIzaSyBRc1...cQIuPNwj7A	

OAuth 2.0 클라이언트 ID

이름	생성일 ↓	유형	클라이언트 ID	작업
표시할 OAuth 클라이언트가 없습니다.				

서비스 계정

[서비스 계정 관리](#)

이메일	이름	이 서비스의 사용 현황(지난 30일) ? ↓	모든 서비스의 사용량(지난 30일) ?	작업
표시할 서비스 계정이 없습니다.				



4. API 키로 googlemaps 읽어오기

API KEY 값

```
gmaps_key='AlzaJQT7BOHuZo'  
gmaps = googlemaps.Client(key=gmaps_key)
```

Server에서 googlemaps API를 요청하면,
client에서 보낸 값을 사람이 읽을 수 있는 값으로 변환한 후 다시 client로 보내줌



5. (googlemaps) 위도, 경도 정보 받아오기

```
lat = []
lng = []

for n in tqdm_notebook(df.index):
    if df['Address'][n] != 'Multiple':
        target_name = df['Address'][n] + ', ' + 'Chicago'
        gmaps_output = gmaps.geocode(target_name)
        location_output = gmaps_output[0].get('geometry')
        lat.append(location_output['location']['lat'])
        lng.append(location_output['location']['lng'])
    else:
        lat.append(np.nan)
        lng.append(np.nan)

df['lat'] = lat
df['lng'] = lng
df.head()
```

예외 처리

- 주소가 제대로 표기되어 있는 경우만 주소를 검색하도록 처리
- 'Multiple'이라고 적혀있는 경우 제외

지오코딩(geocoding)

- 주소를 지리적 좌표로 변환하는 것
- 2109 W. Chicago Ave. → lat: 41.9, lng: -87.7

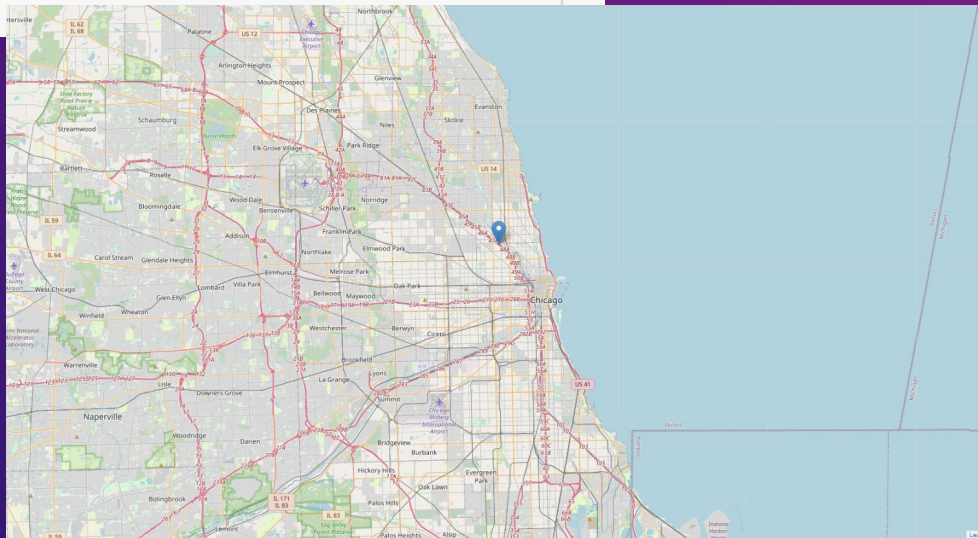
예외 처리 후 배열에 저장



6. (folium) 지도 그리기

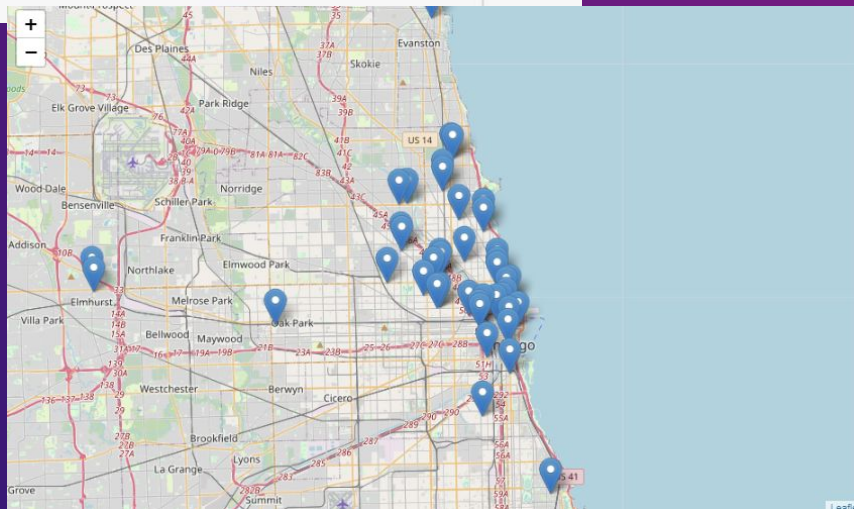
50개 위도, 경도의 평균값이 중앙에 오도록 설정

```
mapping = folium.Map(location=[df['lat'].mean(),df['lng'].mean()], zoom_start=11)  
folium.Marker(df['lat'].mean(), df['lng'].mean()),popup='center').add_to(mapping)  
mapping
```



7. (folium) Marker로 맛집 위치 표시하기

```
mapping = folium.Map(location=[df['lat'].mean(),df['lng'].mean()],zoom_start=11)
for n in df.index:
    if df['Address'][n]!='Multiple':
        folium.Marker([df['lat'][n],df['lng'][n]],popup=df['Cafe'][n]).add_to(mapping)
mapping
```



● 느낀 점



유지원

beautifulSoup로 데이터를 가져오고 가공하는 과정을 경험해볼 수 있어서 유익했다. 또, 아나콘다 가상환경과 주피터 노트북, 콜랩 등으로 작업하는 경험을 늘릴 수 있어서 좋았다. Folium, googlemaps, pandas 모듈을 사용한 것이 새로웠고 여러 메서드/함수들을 더 사용해보고 싶었다. 각 모듈을 사용하며 발생하는 오류들을 해결한 것도 재미있었다. 특히 urlopen() 버그에 대해 알아봤던 것이 기억에 남는다.



송지원

전체적인 공부를 한 뒤에 실습 하는 것이 아닌 실습을 하면서 필요한 부분을 공부하고 사용하는 형태가 다양한 모듈들을 이해하는데 적합했다. 처음에는 파일경로를 설정하는 점에서도 버벅거렸지만 이를 해결하면서 절대경로와 상대경로 개념도 알게 되고 기억에 잘 남았다. 특히 urlopen 명령어를 사용하면서 어려움을 겪은게 가장 기억에 남는다. 에러를 구글링 하는게 제일 빠르다는 점을 몸소 느끼며 많은 시간을 소비했지만 다음 장에서는 이리저리 말아야지라고 생각하며 재밌게 실습했다.



김관엽

파이썬이라는 언어를 보다 심도 있게 다뤄보면서 제대로 사용해서 구현해 본 느낌이다. 활동 이전까지는 데이터 분석과 같은 것에 진입 장벽을 느꼈었는데, 책 설명을 따라 차근차근 해결해 보고 나름대로 시행착오를 겪으면서 조금씩 그 경계를 허물 수 있었다. 이번 경험은 앞으로의 성장을 위한 중요한 자산이 될 것 같다.





감사합니다.

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon** and infographics & images by **Freepik**

