[📝필사 과제 안내]

필사를 진행하며, 주요 코드나 작업에 대해 손과 눈에 최대한 익으실 수 있도록 함입니다 :)

수업을 진행하시며 이해가 잘 되지 않았던 부분을 천천히 써보시거나,

단순히 다시 한 번 따라서 진행해보시는 것만으로도 좋습니다.

특히,

지금 작성하시는 웹크롤링 부분은 태그가 변경되는 등 코드가 동작하지 않을 수 있습니다.

그럴 경우 다시 한 번, 태그를 직접 찾아서 수정해보시면 도움이 될거라 생각합니다.

너무 어렵거나, 힘든 경우 같이 공부하시는 분들은 해결을 어떻게 하셨는지 의견을 한 번 들어보시는 것도 좋습니다.

1️⃣ 문제가 생겼을 경우, **해결 방법에 대한 스스로 고민**

2️⃣ 문제가 생겼을 경우, **같이 공부하는 다른 사람들과의 의견 공유**

위 두 가지도 함께 필사과제를 통해 연습이 되셨으면 좋겠습니다! 나중에 프로젝트 하실 때 분명 큰 도움이 될거라 생각합니다.

새롭게 노트북파일을 만드신 뒤, 써주세요 😀

---

1. BeautifulSoup for web data

03. Web Data

BeautifulSoup Basic

- https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- install

```
- conda install -c anaconda beautifulsoup4
- pip install beautifulsoup4
```

- data
  - 03.test_first.html

```
# import
from bs4 import BeautifulSoup
```

```
page = open("../data/03. zerobase.html", "r").read()
soup = BeautifulSoup(page, "html.parser")
print(soup.prettify())
```

숨겨진 출력 표시

```
# head 태그 확인
soup.head
```

```
<head>
<title>ZeroBase</title>
</head>
```

```
# body 태그 확인
soup.body
```

```
<body>
<div>
<p class="inner-text first-item" id="first">
                Happy ZeroBase.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
</p>
<p class="innter-text second-item">
                Happy Data Science.
                <a href="https://www.python.org" id="py-link"
target="_blink">Python</a>
</p>
</div>
<p class="outer-text first-item" id="second">
<b>Data Science is funny.</b>
</p>
<p class="outer-text">
<i>All I need is Love.</i>
</p>
</body>
```

```
# p 태그 확인
# 처음 발견한 p 태그만 출력
# find()
soup.p
```

```
<p class="inner-text first-item" id="first">
                Happy ZeroBase.
                <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
</p>
```

```
soup.find("p")
```

```
<p class="inner-text first-item" id="first">
                Happy ZeroBase.
```

```
                    <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
    </p>
```

```
1 # 파이썬 예약어
2 # class, id, def, list, str, int, tuple...
```

```
1 soup.find("p", class_="innter-text second-item")
```

```
    <p class="innter-text second-item">
                    Happy Data Science.
                    <a href="https://www.python.org" id="py-link"
    target="_blink">Python</a>
    </p>
```

```
1 soup.find("p", {"class":"outer-text first-item"}).text.strip()
```

```
    'Data Science is funny.'
```

```
1 # 다중 조건
2 soup.find("p", {"class":"inner-text first-item", "id":"first"})
```

```
    <p class="inner-text first-item" id="first">
                    Happy ZeroBase.
                    <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
    </p>
```

```
1 # find_all(): 여러 개의 태그를 반환
2 # list 형태로 반환
3
4 soup.find_all("p")
```

```
    [<p class="inner-text first-item" id="first">
                     Happy ZeroBase.
                     <a href="http://www.pinkwink.kr" id="pw-link">PinkWink</a>
     </p>,
     <p class="innter-text second-item">
                     Happy Data Science.
                     <a href="https://www.python.org" id="py-link"
    target="_blink">Python</a>
     </p>,
     <p class="outer-text first-item" id="second">
     <b>Data Science is funny.</b>
     </p>,
     <p class="outer-text">
     <i>All I need is Love.</i>
     </p>]
```

```
1 # 특정 태그 확인
2 soup.find_all(id="pw-link")[0].text
```

```
    'PinkWink'
```

```
1 soup.find_all("p", class_="innter-text second-item")
```

```
[<p class="innter-text second-item">
                 Happy Data Science.
                 <a href="https://www.python.org" id="py-link"
target="_blink">Python</a>
 </p>]
```

```
1 len(soup.find_all("p"))
```

```
4
```

```
1 print(soup.find_all("p")[0].text)
2 print(soup.find_all("p")[1].string)
3 print(soup.find_all("p")[1].get_text())
```

```

                 Happy ZeroBase.
                 PinkWink

None

                 Happy Data Science.
                 Python
```

```
1 # p 태그 리스트에서 텍스트 속성만 출력
2
3 for each_tag in soup.find_all("p"):
4     print("=" * 50)
5     print(each_tag.text)
```

```
==================================================

                 Happy ZeroBase.
                 PinkWink

==================================================

                 Happy Data Science.
                 Python

==================================================

Data Science is funny.

==================================================

All I need is Love.
```

```
1 # a 태그에서 href 속성값에 있는 값 추출
2 links = soup.find_all("a")
3 links[0].get("href"), links[1]["href"]
```

```
('http://www.pinkwink.kr', 'https://www.python.org')
```

```
for each in links:
    href = each.get("href") # each["href"]
    text = each.get_text()
    print(text + "=>" + href)
```

```
PinkWink=>http://www.pinkwink.kr
Python=>https://www.python.org
```

BeautifulSoup 예제 1-1 - 네이버 금융

```
# import
from urllib.request import urlopen
from bs4 import BeautifulSoup
```

```
url = "https://finance.naver.com/marketindex/"
# page = urlopen(url)
response = urlopen(url)
response
soup = BeautifulSoup(page, "html.parser")
print(soup.prettify())
```

```
# 1
soup.find_all("span", "value"), len(soup.find_all("span", "value"))
```

```
([<span class="value">1,171.10</span>,
  <span class="value">1,064.01</span>,
  <span class="value">1,383.48</span>,
  <span class="value">181.73</span>,
  <span class="value">109.9200</span>,
  <span class="value">1.1810</span>,
  <span class="value">1.3849</span>,
  <span class="value">92.6500</span>,
  <span class="value">70.45</span>,
  <span class="value">1641.73</span>,
  <span class="value">1792.0</span>,
  <span class="value">67506.13</span>],
 12)
```

```
# 2
soup.find_all("span", class_="value"), len(soup.find_all("span", "value"))
```

```
([<span class="value">1,171.10</span>,
  <span class="value">1,064.01</span>,
  <span class="value">1,383.48</span>,
  <span class="value">181.73</span>,
  <span class="value">109.9200</span>,
  <span class="value">1.1810</span>,
```

```
 <span class="value">1.3849</span>,
 <span class="value">92.6500</span>,
 <span class="value">70.45</span>,
 <span class="value">1641.73</span>,
 <span class="value">1792.0</span>,
 <span class="value">67506.13</span>],
12)
```

```
1 # 3
2 soup.find_all("span", {"class":"value"}), len(soup.find_all("span", {"class":"va
```

```
([<span class="value">1,171.10</span>,
  <span class="value">1,064.01</span>,
  <span class="value">1,383.48</span>,
  <span class="value">181.73</span>,
  <span class="value">109.9200</span>,
  <span class="value">1.1810</span>,
  <span class="value">1.3849</span>,
  <span class="value">92.6500</span>,
  <span class="value">70.45</span>,
  <span class="value">1641.73</span>,
  <span class="value">1792.0</span>,
  <span class="value">67506.13</span>],
 12)
```

```
1 soup.find_all("span", {"class":"value"})[0].text, soup.find_all("span", {"class"
```

```
('1,171.10', '1,171.10', '1,171.10')
```

BeautifulSoup 예제1-2 - 네이버 금융

- !pip install requests
- find, find_all
- select, select_one
- find, select_one : 단일 선택
- select, find_all : 다중 선택

```
1 import requests
2 # from urllib.request.Request
3 from bs4 import BeautifulSoup
```

```
1 url = "https://finance.naver.com/marketindex/"
2 response = requests.get(url)
3 # requests.get(), requests.post()
4 # response.text
5 soup = BeautifulSoup(response.text, "html.parser")
6 print(soup.prettify())
```

숨겨진 출력 표시

```
# soup.find_all("li", "on")
# id => #
# class => .
exchangeList = soup.select("#exchangeList > li")
len(exchangeList), # exchangeList
```

숨겨진 출력 표시

```
title = exchangeList[0].select_one(".h_lst").text
exchange = exchangeList[0].select_one(".value").text
change = exchangeList[0].select_one(".change").text
updown = exchangeList[0].select_one(".head_info.point_dn > .blind").text
# link

title, exchange, change, updown
```

```
('미국 USD', '1,203.20', ' 1.30', '하락')
```

```
# findmethod = soup.find_all("ul", id="exchangeList")
# findmethod[0].find_all("span", "value")
```

```
[<span class="value">1,171.20</span>,
 <span class="value">1,063.81</span>,
 <span class="value">1,382.95</span>,
 <span class="value">181.70</span>]
```

```
baseUrl = "https://finance.naver.com"
baseUrl + exchangeList[0].select_one("a").get("href")
```

```
'https://finance.naver.com/marketindex/exchangeDetail.naver?marketindexCd=FX_
USDKRW'
```

```
# 4개 데이터 수집

exchange_datas = []
baseUrl = "https://finance.naver.com"

for item in exchangeList:
    data = {
        "title": item.select_one(".h_lst").text,
        "exchnage": item.select_one(".value").text,
        "change": item.select_one(".change").text,
        "updown": item.select_one(".head_info.point_dn > .blind").text,
        "link": baseUrl + item.select_one("a").get("href")
    }
    exchange_datas.append(data)
df = pd.DataFrame(exchange_datas)
df.to_excel("./naverfinance.xlsx", encoding="utf-8")
```

숨겨진 출력 표시

BeautifulSoup 예제2 - 위키백과 문서 정보 가져오기

```
import urllib
from urllib.request import urlopen, Request

html = "https://ko.wikipedia.org/wiki/{search_words}"
# https://ko.wikipedia.org/wiki/여명의_눈동자
req = Request(html.format(search_words=urllib.parse.quote("여명의_눈동자"))) # 글자를
response = urlopen(req)
soup = BeautifulSoup(response, "html.parser")
print(soup.prettify())
```

숨겨진 출력 표시

```
n = 0

for each in soup.find_all("ul"):
    print("=>" + str(n) + "========================")
    print(each.get_text())
    n += 1
```

숨겨진 출력 표시

```
soup.find_all("ul")[15].text.strip().replace("\xa0", "").replace("\n", "")
```

```
'채시라: 윤여옥 역 (아역: 김민정)박상원: 장하림(하리모토 나츠오) 역 (아역: 김태진)최재성: 최대치
(사카이) 역 (아역: 장덕수)'
```

Python List 데이터형

- list형은 대괄호로 생성

```
colors = ["red", "blue", "green"]

colors[0], colors[1], colors[2]
```

```
('red', 'blue', 'green')
```

```
b = colors
b
```

```
['red', 'blue', 'green']
```

```
b[1] = "black"
b
```

```
['red', 'black', 'green']
```

```
colors
```

```
['red', 'black', 'green']
```

```
c = colors.copy()
c
```

```
['red', 'black', 'green']
```

```
c[1] = "yellow"
c
```

```
['red', 'yellow', 'green']
```

```
colors
```

```
['red', 'black', 'green']
```

- list형을 반복문에(for) 적용

```
for color in colors:
    print(color)
```

```
red
black
green
```

- in명령으로 조건문(if)에 적용

```
if "white" in colors:
    print("True")
```

```
movies = ["라라랜드", "먼 훗날 우리", "어벤저스", "다크나이트"]
print(movies)
```

```
['라라랜드', '먼 훗날 우리', '어벤저스', '다크나이트']
```

- append: list 제일 뒤에 추가

```
movies.append("타이타닉")
movies
```

```
['라라랜드', '먼 훗날 우리', '어벤저스', '다크나이트', '타이타닉']
```

- pop: 리스트 제일 뒤부터 자료를 하나씩 삭제

```
movies.pop()
movies
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/var/folders/8c/jb57288j6xlb3s_tkmys1kj00000gn/T/ipykernel_3807/662002098.py
in <module>
----> 1 movies.pop()
      2 movies

IndexError: pop from empty list
```

SEARCH STACK OVERFLOW

- extend: 제일 뒤에 자료 추가

```
1 movies.extend(["위대한쇼맨", "인셉션", "터미네이터"])
2 movies
```

```
['라라랜드', '먼 훗날 우리', '어벤저스', '다크나이트', '위대한쇼맨', '인셉션', '터미네이터']
```

- remove: 자료를 삭제

```
1 movies.remove("어벤저스")
2 movies
```

```
['라라랜드', '먼 훗날 우리', '다크나이트', '위대한쇼맨', '인셉션', '터미네이터']
```

- 슬라이싱: [n:m] n번재 부터 m-1까지

```
1 movies[3:5]
```

```
['위대한쇼맨', '인셉션']
```

```
1 favorite_movies = movies[3:5]
2 favorite_movies
```

```
['위대한쇼맨', '인셉션']
```

- insert: 원하는 위치에 자료를 삽입

```
1 favorite_movies.insert(1, 9.60)
2 favorite_movies
```

```
['위대한쇼맨', 9.6, '인셉션']
```

```
1 favorite_movies.insert(3, 9.50)
2 favorite_movies
```

```
['위대한쇼맨', 9.6, '인셉션', 9.5]
```

- list안에 list

```
1 favorite_movies.insert(5, ["레오나르도 디카프리오", "조용하"])
2 favorite_movies
```

```
['위대한쇼맨', 9.6, '인셉션', 9.5, ['레오나르도 디카프리오', '조용하']]
```

- isinstance: 자료형 True/False

```
1 isinstance(favorite_movies, list)
```

```
True
```

```
1 favorite_movies
```

```
['위대한쇼맨', 9.6, '인셉션', 9.5, ['레오나르도 디카프리오', '조용하']]
```

```
1 for each_item in favorite_movies:
2     if isinstance(each_item, list):
3         for nested_item in each_item:
4             print("nested_item", nested_item)
5     else:
6         print("each_item", each_item)
```

```
each_item 위대한쇼맨
each_item 9.6
each_item 인셉션
each_item 9.5
nested_item 레오나르도 디카프리오
nested_item 조용하
```

---

2. 시카고 맛집 데이터 분석 - 개요

- https://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/
- chicago magazine the 50 best sandwiches

최종목표

총 51개 페이지에서 각 가게의 정보를 가져온다

- 가게이름
- 대표메뉴
- 대표메뉴의 가격
- 가게주소

3. 시카고 맛집 데이터 분석 - 메인페이지

```
!pip install fake_useragent
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-w
Collecting fake_useragent
  Downloading fake_useragent-1.0.1-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 2.5 MB/s
Requirement already satisfied: importlib-resources>=5.0 in /usr/local/lib/pyth
Requirement already satisfied: importlib-metadata~=4.0 in /usr/local/lib/pytho
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/pyth
Installing collected packages: fake-useragent
Successfully installed fake-useragent-1.0.1
```

```
# !pip install fake-useragent
from urllib.request import Request, urlopen
from fake_useragent import UserAgent
from bs4 import BeautifulSoup

url_base = "https://www.chicagomag.com/"
url_sub = "Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/"
url = url_base + url_sub
ua = UserAgent()
req = Request(url, headers={"user-agent": ua.ie})
html = urlopen(req)
soup = BeautifulSoup(html, "html.parser")
# print(soup.prettify())

```

```
soup.find_all("div", "sammy"), len(soup.find_all("div", "sammy"))
# soup.select(".sammy"), len(soup.select(".sammy"))
```

숨겨진 출력 표시

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

```
bs4.element.Tag
```

```
tmp_one.find(class_="sammyRank").get_text()
# tmp_one.select_one(".sammyRank").text
```

```
'1'
```

```
tmp_one
```

```
<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> </a></div>
</div>
```

```
tmp_one.find("div", {"class":"sammyListing"}).get_text()
# tmp_one.select_one(".sammyListing").text
```

```
'BLT\nOld Oak Tap\nRead more '
```

```
tmp_one.find("a")["href"]
# tmp_one.select_one("a").get("href")
```

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

```
import re

tmp_string = tmp_one.find(class_="sammyListing").get_text()
re.split(("\n|\r\n"), tmp_string)
```

```
['BLT', 'Old Oak Tap', 'Read more ']
```

```
print(re.split(("\n|\r\n"), tmp_string)[0]) # menu
print(re.split(("\n|\r\n"), tmp_string)[1]) # cafe
```

```
BLT
Old Oak Tap
```

```
from urllib.parse import urljoin

url_base = "http://www.chicagomag.com"

# 필요한 내용을 담을 빈 리스트
# 리스트로 하나씩 컬럼을 만들고, DataFrame으로 합칠 예정
rank = []
main_menu = []
cafe_name = []
url_add = []

list_soup = soup.find_all("div", "sammy") # soup.select(".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"]))
```

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

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

```
1 rank[:5]
```

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

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

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

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

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

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

```
['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/']
```

```
1 import pandas as pd
2
3 data = {
4     "Rank": rank,
5     "Menu": main_menu,
6     "Cafe": cafe_name,
7     "URL": url_add,
8 }
9
10 df = pd.DataFrame(data)
11 df.tail(2)
```

	Rank	Menu	Cafe	URL
48	49	Le Végétarien	Toni Patisserie	https://www.chicagomag.com/Chicago-Magazine/No...
49	50	The Gatsby	Phoebe's Bakery	https://www.chicagomag.com/Chicago-Magazine/No...

```
1 # 컬럼 순서 변경
2 df = pd.DataFrame(data, columns=["Rank", "Cafe", "Menu", "URL"])
3 df.tail()
```

	Rank	Cafe	Menu	URL
45	46	Chickpea	Kufta	https://www.chicagomag.com/Chicago-Magazine/No...
		The Goddess and	Debbie's Egg	https://www.chicagomag.com/Chicago-

```
# 데이터 저장
df.to_csv(
    "../data/03. best_sandwiches_list_chicago.csv", sep=",", encoding="utf-8"
)
```

4. 시카고 맛집 데이터 분석 - 하위페이지

```
# requirements
import pandas as pd
from urllib.request import urlopen, Request
from fake_useragent import UserAgent
from bs4 import BeautifulSoup
```

```
df = pd.read_csv("../data/03. best_sandwiches_list_chicago.csv", index_col=0)
df.tail()
```

	Rank	Cafe	Menu	URL
45	46	Chickpea	Kufta	https://www.chicagomag.com/Chicago-Magazine/No...
46	47	The Goddess and Grocer	Debbie's Egg Salad	https://www.chicagomag.com/Chicago-Magazine/No...
47	48	Zenwich	Beef Curry	https://www.chicagomag.com/Chicago-Magazine/No...
				https://www.chicagomag.com/Chicago-

```
df["URL"][0]
```

```
'http://www.chicagomag.com/Chicago-Magazine/November-2012/Best-Sandwiches-in-Chicago-Old-Oak-Tap-BLT/'
```

```
import requests

response = requests.get(df['URL'][0], headers={'user-agent': ua.ie})
soup_tmp = BeautifulSoup(response.content, 'html.parser')
soup_tmp.select_one('.addy')
```

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

```
# 22.11.23. 기준 403 코드 에러
req = Request(df["URL"][0], headers={"user-agent":ua.ie})
html = urlopen(req).read()
```

```
soup_tmp = BeautifulSoup(html, "html.parser")
soup_tmp.find("p", "addy") # soup_find.select_one(".addy")
```

```
---------------------------------------------------------------------------
HTTPError                                 Traceback (most recent call last)
<ipython-input-22-832beb3a0d1c> in <module>
      1 req = Request(df["URL"][0], headers={"user-agent":ua.ie})
----> 2 html = urlopen(req).read()
      3 soup_tmp = BeautifulSoup(html, "html.parser")
      4 soup_tmp.find("p", "addy") # soup_find.select_one(".addy")

                                5 frames
/usr/lib/python3.7/urllib/request.py in http_error_default(self, req, fp,
code, msg, hdrs)
    647 class HTTPDefaultErrorHandler(BaseHandler):
    648     def http_error_default(self, req, fp, code, msg, hdrs):
--> 649         raise HTTPError(req.full_url, code, msg, hdrs, fp)
    650
    651 class HTTPRedirectHandler(BaseHandler):

HTTPError: HTTP Error 403: Forbidden
```

SEARCH STACK OVERFLOW

```
# 22.11.23. 기준 아래 코드로 변경
import requests

req = requests.get(df['URL'][49], headers={'user-agent': ua.ie})
soup_tmp = BeautifulSoup(req.content, 'html.parser')
soup_tmp.select_one('.addy')
```

```
<p class="addy">
<em>$6.85. 3351 N. Broadway, 773-868-4000, <a
href="http://phoebesbakery.com/">phoebesbakery.com</a></em></p>
```

```
# regular expression
price_tmp = soup_tmp.find("p", "addy").text
price_tmp
```

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

```
import re
re.split(".,", price_tmp)
```

```
['\n$10. 2109 W. Chicago Ave', ' 773-772-040', ' theoldoaktap.com']
```

```
price_tmp = re.split(".,", price_tmp)[0]
price_tmp
```

```
'\n$10. 2109 W. Chicago Ave'
```

```
tmp = re.search("\$\d+\.(\d+)?", price_tmp).group()
price_tmp[len(tmp) + 2:]
```

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

```
from tqdm import tqdm

price = []
address = []

for idx, row in tqdm(df.iterrows()):
    req = requests.get(row["URL"], headers={"user-agent": ua.ie})
    # html = urlopen(req).read()
    soup_tmp = BeautifulSoup(req.content, "html.parser")
    gettings = soup_tmp.find("p", "addy").get_text()
    price_tmp = re.split(".,", gettings)[0]
    tmp = re.search("\$\d+\.(\d+)?", price_tmp).group()
    price.append(tmp)
    address.append(price_tmp[len(tmp)+2:])
    print(idx)
```

숨겨진 출력 표시

```
len(price), len(address)
```

```
(2, 2)
```

```
price[:5]
```

```
['$10.', '$9.']
```

```
address[:5]
```

```
['2109 W. Chicago Ave',
 '800 W. Randolph St',
 ' 445 N. Clark St',
 ' 914 Noyes St',
 '825 W. Fulton Mkt']
```

```
df.tail(2)
```

	Rank	Cafe	Menu	URL
48	49	Toni Patisserie	Le Végétarien	https://www.chicagomag.com/Chicago-Magazine/No...
49	50	Phoebe’s Bakery	The Gatsby	https://www.chicagomag.com/Chicago-Magazine/No...

```
df["Price"] = price
df["Address"] = address
df = df.loc[:, ["Rank", "Cafe", "Menu", "Price", "Address"]]
df.set_index("Rank", inplace=True)
df.head()
```

숨겨진 출력 표시

```
df.to_csv(
    "../data/03. best_sandwiches_list_chicago2.csv", sep=",", encoding="UTF-8"
)
```

```
pd.read_csv("../data/03. best_sandwiches_list_chicago2.csv", index_col=0)
```

11	Lula Cafe	Ham and Raclette Panino	$11.	2537 N. Kedzie Blvd
12	Ricobene's	Breaded Steak	$5.49	Multiple location
13	Frog n Snail	The Hawkeye	$14.	3124 N. Broadwa
14	Crosby's Kitchen	Chicken Dip	$10.	3455 N. Southport Ave
15	Longman & Eagle	Wild Boar Sloppy Joe	$13.	2657 N. Kedzie Ave
16	Bari	Meatball Sub	$4.50	1120 W. Grand Ave
17	Manny's	Corned Beef	$11.95	1141 S. Jefferson St
18	Eggy's	Turkey Club	$11.50	333 E. Benton Pl
19	Old Jerusalem	Falafel	$6.25	1411 N. Wells St
20	Mindy's HotChocolate	Crab Cake	$15.	1747 N. Damen Ave
21	Olga's Delicatessen	Chicken Schnitzel	$5.	3209 W. Irving Park Rd
22	Dawali Mediterranean Kitchen	Shawarma	$6.	Multiple location
23	Big Jones	Toasted Pimiento Cheese	$8.	5347 N. Clark St
24	La Pane	Vegetarian Panino	$5.99	2954 W. Irving Park Rd
25	Pastoral	Cali Chèvre	$7.52	Multiple location
26	Max's Deli	Pastrami	$11.95	191 Skokie Valley Rd
27	Lucky's Sandwich Co.	The Fredo	$7.50	Multiple location
28	City Provisions	Smoked Ham	$12.95	1818 W. Wilson Ave

5. 시카고 맛집 데이터 지도 시각화

```
# requirements

import folium
import pandas as pd
import numpy as np
import googlemaps
from tqdm import tqdm
```

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

Rank	Cafe	Menu	Price	Address
41	Z&H MarketCafe	The Marty	$7.25	1323 E. 57th St
42	Market House on the Square	Whitefish	$11.	655 Forest Ave
43	Elaine’s Coffee Call	Oat Bread, Pecan Butter, and Fruit Jam	$6.	Hotel Lincol
44	Marion Street Cheese Market	Cauliflower Melt	$9.	100 S. Marion St
45	Cafecito	Cubana	$5.49	26 E. Congress Pkwy
46	Chickpea	Kufta	$8.	2018 W. Chicago Ave
47	The Goddess and Grocer	Debbie’s Egg Salad	$6.50	25 E. Delaware Pl
48	Zaytinya	Beef Curry	$7.50	410 N. York St, Pkwy

```
gmaps_key = ""
gmaps = googlemaps.Client(key=gmaps_key)
```

47	The Goddess and Grocer	Debbie’s Egg Salad	$6.50	25 E. Delaware Pl

```
lat = []
lng = []

for idx, row in tqdm(df.iterrows()):
    if not row["Address"] == "Multiple location":
        target_name = row["Address"] + ", " + "Chicago"
        # print(target_name)
        gmaps_output = gmaps.geocode(target_name)
        location_ouput = gmaps_output[0].get("geometry")
        lat.append(location_ouput["location"]["lat"])
        lng.append(location_ouput["location"]["lng"])
        # location_output = gmaps_output[0]
    else:
        lat.append(np.nan)
        lng.append(np.nan)
```

```
50it [00:17,  2.85it/s]
```

```
len(lat), len(lng)
```

```
(50, 50)
```

```
df.tail()
```

Rank	Cafe	Menu	Price	Address
46	Chickpea	Kufta	$8.	2018 W. Chicago Ave
47	The Goddess and Grocer	Debbie’s Egg Salad	$6.50	25 E. Delaware Pl

```
df["lat"] = lat
df["lng"] = lng
df.tail()
```

Rank	Cafe	Menu	Price	Address	lat	lng
46	Chickpea	Kufta	$8.	2018 W. Chicago Ave	41.896113	-87.677857
47	The Goddess and Grocer	Debbie’s Egg Salad	$6.50	25 E. Delaware Pl	41.898979	-87.627393
48	Zenwich	Beef Curry	$7.50	416 N. York St	41.910583	-87.940488
49	Toni Patisserie	Le Végétarien	$8.75	65 E. Washington St	41.883106	-87.625438

```
mapping = folium.Map(location=[41.8781136, -87.6297982], zoom_start=11)

for idx, row in df.iterrows():
    if not row["Address"] == "Multiple location":
        folium.Marker(
            location=[row["lat"], row["lng"]],
            popup=row["Cafe"],
            tooltip=row["Menu"],
            icon=folium.Icon(
                icon="coffee",
                prefix="fa"
            )
        ).add_to(mapping)

mapping
```

숨겨진 출력 표시