python

Crawing / Scraping
(Beautiful Soup 4 / Selenium)

- Scraping 시작하기
- BeautifulSoup 4
  - BS4 의 특징
  - BS4의 주요 함수
- Selenium
  - Selenium 특징
  - Selenium 주요 함수

❑ 용어

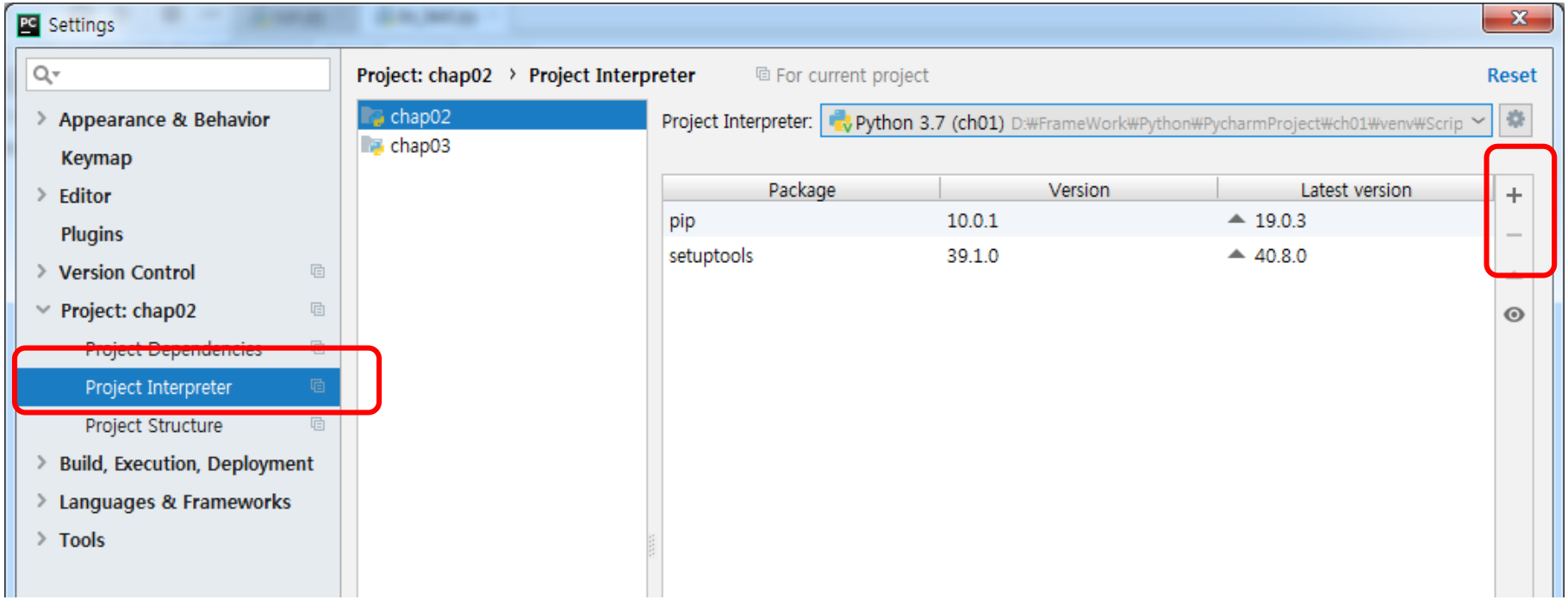
- 웹 크롤링(Web Crawling)
  - 크롤링이란 단어는 웹 크롤러(crawler)라는 단어에서 시작한 말이다.
  - 크롤러란 조직적, 자동화된 방법으로 월드와이드 웹을 탐색하는 컴퓨터 프로그램이다.(출처: 위키백과)
  - 크롤링은 크롤러가 하는 작업을 부르는 말로, 여러 인터넷 사이트의 페이지(문서, html 등)를 수집해서 분류하는 것이다.
  - 대체로 찾아낸 데이터를 저장한 후 쉽게 찾을 수 있게 인덱싱한다.

- 파싱(parsing)
  - 파싱이란 어떤 페이지(문서, html 등)에서 내가 원하는 데이터를 특정 패턴이나 순서로 추출하여 정보를 가공하는 것이다.
  - 위 문장만 보면 굉장히 간단해 보이지만 컴퓨터 과학적 정의를 보면 파싱이란 일련의 문자열을 의미있는 토큰(token)으로 분해하고 이들로 이루어진 파스 트리(parse tree)를 만드는 과정을 말한다.(출처: 위키백과)

- 스크래핑(scraping)
  - 스크래핑이란 HTTP를 통해 웹 사이트의 내용을 긁어다 원하는 형태로 가공하는 것이다.
  - 쉽게 말해 웹 사이트의 데이터를 수집하는 모든 작업을 뜻한다.
  - 크롤링도 일종의 스크래핑 기술이라고 할 수 있다.

❑ PyCharm 에서 가상환경 라이브러리 추가

- File 메뉴의 Settings 선택
- Project -> Project Interpreter 메뉴 선택.
- Project Interpreter 화면의 우측의 +, - 버튼을 사용하여 라이브러리를 추가 및 제거할 수 있다.
- + 버튼을 선택하여 필요한 라이브러리를 검색하고 Install Package 버튼을 클릭하면 된다.



❑ Beautiful Soup 4는

- Beautiful Soup은 스크래핑 프로젝트를 위한 Python 라이브러리이며
  주요 특징은 아래와 같다.
  - HTML 및 XML 파일에서 데이터를 추출하기 위한 Python 라이브러리이다.
  - 구문 분석 트리를 탐색, 검색 및 수정하기 위한 심플한 함수 제공
  - 대상 문서의 인코딩에 상관없이 내부적으로 UTF-8로 자동 변환 처리한다.
  - 기본 파서 이외에 다양한 파서(lxml, html5lib)를 설치하여 파싱전략 또는 성능을 개선할 수 있다.
- 프로그래머가 문서 파싱 및 분석을 위해 BS4를 사용한다면 몇 시간 또는 며칠을 절약할 수 있을 것이다.

- 라이브러리 설치
  - 사이트 : https://www.crummy.com/software/BeautifulSoup/
  - 설치 : pip install beautifulsoup4
  - 아나콘다 설치 :
    - conda activate py37
    - conda install beautifulsoup4
    - conda list

❑ 파서 종류

파서	파서명	특징
Python's html.parser	html.parser	기본 내장, 일반적인 성능
lxml's HTML parser	lxml	매우 빠름, 외부 C 의존
lxml's XML parser	lxml-xml, xml	매우 빠름, 외부 C 의존
html5lib	html5lib	웹브라우저 방식 파싱, 유효한 html5 생성, 매우 느림

- 기본 파서 이외의 파서를 사용하고 싶다면 설치한다.
  - lxml 파서 설치 : pip install lxml
  - html5lib 파서 설치 : pip install html5lib

❑ Beautiful Soup 실습 1

- 이상한 나라의 엘리스에서
  - 겨울잠쥐의 이야기
  - 옛날 옛날에 어린 세자매가 살았습니다. 그들의 이름은
  - 엘시, 레이시, 틸리 였습니다.
  - 그들은 우물 바닥에 살았습니다.

```
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>
"""
```

- 위 샘플은 구조적으로 완변한 문서가 아니다.

❑ 분석 대상 가져오기

- 단순 문자열 처리

```
doc = "<html> ... </html>"

soup = BeautifulSoup(doc,"html.parser")
```

- html 문서 파일 열기
  - 현재 경로에 있는 "example.html"

```
with open("example.html") as fp:

    soup = BeautifulSoup(fp, 'html.parser')
```

❑ 분석 대상 가져오기

- urllib 모듈을 사용하여 웹에 있는 소스 가져오기

```
import urllib.request
import urllib.parse

url = "https://www.google.com/"

with urllib.request.urlopen(url) as response:
    html = response.read()
    soup = BeautifulSoup(html, 'html.parser')
```

- requests 모듈을 사용하여 웹에 있는 소스 가져오기
  - 만약 requests 모듈이 없다면 설치 : pip install requests

```
import requests

# https://movie.naver.com/milkis 존재하지 않는 페이지
url = "https://movie.naver.com/milkis"

resp = requests.get(url)
print("status_code", resp.status_code)

html = resp.text
soup = BeautifulSoup(html, 'html.parser')
```

❑ Beautiful Soup 실습 1

```
from bs4 import BeautifulSoup

html_doc = ...

# 문서 파싱
soup = BeautifulSoup(html_doc,"html.parser")

# 구조화 하여 출력
print(soup.prettify())

print("> soup.title", soup.title , sep="₩n", end="₩n₩n")
print("> soup.title.name", soup.title.name,  sep="₩n", end="₩n₩n")
print("> soup.title.string", soup.title.string,  sep="₩n", end="₩n₩n")
print("> soup.title.parent.name", soup.title.parent.name, sep="₩n", end="₩n₩n")
print("> soup.p", soup.p ,  sep="₩n", end="₩n₩n")
print("> soup.p['class']", soup.p['class'],  sep="₩n", end="₩n₩n")
print("> soup.a", soup.a,  sep="₩n", end="₩n₩n")
print("> soup.find_all('a')", soup.find_all('a'),  sep="₩n", end="₩n₩n")
print("> soup.find(id='link3')", soup.find(id='link3'),  sep="₩n", end="₩n₩n")
```

❑ find() 및 find_all()함수

- 태그와 속성을 이용할 때 주로 사용
- 함수 인자로는 찾고자 하는 태그의 이름, 속성 기타 등등이 들어간다.

- find_all(name, attrs, recursive, string, limit, **kwargs)
  - 해당 조건에 맞는 모든 태그들을 가져온다.

- find(name, attrs, recursive, string, **kwargs)
  - 해당 조건에 맞는 하나의 태그를 가져온다. 중복이면 가장 첫 번째 태그를 가져온다.

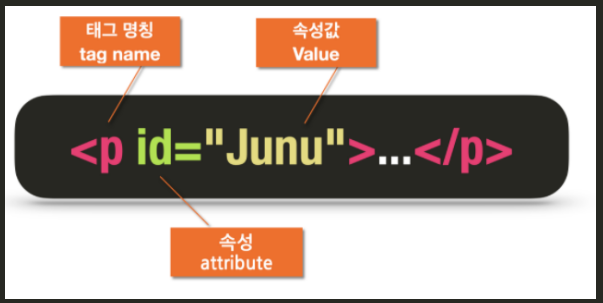
❑ select() 및 select_one()함수

- Beautiful Soup 4.7.0 부터 CSS Selector 를 통한 요소 검색 지원
- CSS Selector를 이용할 때 주로 사용

- select : 해당 선택자에 맞는 모든 요소정보 가져온다.
- select_one : 해당 선택자에 맞는 요소정보 중 첫 번째 요소를 가져온다.

```
tag = "<p class='youngone' id='junu'> Hello World! </p>"
soup = BeautifulSoup(tag)

# 태그 이름만 특정
soup.find('p')
# 태그 속성만 특정
soup.find(class_='youngone')
soup.find(attrs = {'class':'youngone'})
# 태그 이름과 속성 모두 특정
soup.find('p', class_='youngone')
```




```
tag = "<p class='youngone' id='junu'> Hello World! </p>"
soup = BeautifulSoup(tag)
object_tag = soup.find('p')

#태그의 이름
object_tag.name
#결과: 'p'

#태그에 담긴 텍스트
object_tag.text
#결과: ' Hello World! '

#태그의 속성과 속성값
object_tag.attrs
#결과: {'class': ['youngone'], 'id': 'junu'}
```

```
# 태그 이름만 특정
soup.select_one('p')
# 태그 class 특정
soup.select_one('.youngone')
# 태그 이름과 class 모두 특정
soup.select_one('p.youngone')
# 태그 id 특정
soup.select_one('#junu')
# 태그 이름과 id 모두 특정
soup.select_one('p#junu')
# 태그 이름과 class, id 모두 특정
soup.select_one('p.youngone#junu')
```

```
#find
soup.find('div').find('p')
#select
soup.select_one('div > p')
```

❑ find_all() 함수 사용

```
soup.find_all("title")

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

soup.find_all("p", "title")
# [<p class="title"><b>The Dormouse's story</b></p>]

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>]

# id argument
soup.find_all(id="link2")
# [<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a>]

# string argument
soup.find_all(string="Elsie")
# ['Elsie']

soup.find_all(string=["Tillie", "Elsie", "Lacie"])
# ['Elsie', 'Lacie', 'Tillie']

soup.find_all("a", string=["Tillie", "Elsie", "Lacie"])

import re
soup.find(string=re.compile("sisters"))
# 'Once upon a time there were three little sisters; and their names were₩n'
```

❑ Beautiful Soup 실습 2

- 네이버의 영화랭킹 사이트에서 영화제목, 링크, 평점 추출해보기

```
from bs4 import BeautifulSoup
import urllib.request

# 분석을 위한 사이트
url = "https://movie.naver.com/movie/sdb/rank/rmovie.nhn?sel=pnt"
html = urllib.request.urlopen(url).read()
soup = BeautifulSoup(html, "html.parser")

# 영화제목:링크: 평점
tb = soup.select_one(".list_ranking")
trs  = tb.find_all("tr", limit=10)

# 해당 tr 에 class ="tit5", "point"
for tr in trs :
    print(tr)
    # a =  tr.select_one(".tit5 a")
    # if a :
    #     title = a.get_text()
    #     href = a.get("href")
    #
    #     p = tr.select_one(".point")
    #     point = p.get_text()
    #     print("₩n", title, href , point, sep="---")
    #
    # print("-" * 50)
```

❑ Beautiful Soup 문제

- 다음 영화의 박스오피스 에서 영화정보 조회하여
  - 영화제목 , 링크정보 , 점수 , 관객수 , 이미지 경로 가져오기
  - 사이트 : https://movie.daum.net/boxoffice/weekly

1. 각각의 영화정보를 둘러싸는 태그 또는 선택자 파악
   - .main_detail .list_boxthumb li
2. 영화의 링크 정보 및 제목을 가지는 태그 또는 선택자
   - .tit_join a
3. 평점 정보를 갖고 있는 태그 또는 선택자
   - .emph_grade
4. 주말관객 정보가 있는 태그 또는 선택자
   - dl dd

❑ Selenium 는

- Selenium은 다양한 플랫폼에서 웹 브라우저를 자동화 하는 도구
- 다양한 운영체체, 프로그래밍 언어에서 브라우저기반의 시나리오 테스트 프레임 워크

- Selenium 설치
  - 설치 : pip install selenium
  - 사이트 : https://www.seleniumhq.org/
  - API 관련 : https://selenium-python.readthedocs.io/ (with Baiju Muthukadan)

- BeautilfulSoup 등 다른 파서와 차이
  - 현재 보여지는 페이지는 페이지와 실제 소스와는 차이가 있다.
  - JavaScript 등으로 다양한 event, ajax 통신으로 인해 동적으로 DOM이 변화한다.

- Selenium은
  - 실제 웹 브라우저가 동작하기 때문에 JS로 렌더링이 완료된 후의 DOM결과물에 접근이 가능
  - 이전의 작업(로그인, 이전 페이지에서 이동)이 필요한 경우 등을 처리할 수 있다.

❑ WebDriver 다운받기

- Selenium은 webdriver라는 것을 통해 디바이스에 설치된 브라우저들을 제어할 수 있다.
- 다양한 서드파티 드라이버 지원
  - 사이트 : https://www.seleniumhq.org/download/

Third Party Browser Drivers NOT DEVELOPED by seleniumhq

Browser					
Mozilla GeckoDriver	latest	change log	issue tracker	Implementation Status	
Google Chrome Driver	latest	change log	issue tracker	selenium wiki page	
Opera	latest		issue tracker	selenium wiki page	
Microsoft Edge Driver			issue tracker	Implementation Status	
GhostDriver	(PhantomJS)		issue tracker	SeConf talk	
HtmlUnitDriver	latest		issue tracker		
SafariDriver			issue tracker		
Windows Phone			issue tracker		
Windows Phone	4.14.028.10		issue tracker		Released 2013-11-23
Selendroid - Selenium for Android			issue tracker		
ios-driver			issue tracker		
BlackBerry 10			issue tracker		Released 2014-01-28

❑ WebDriver 다운받기

- Google Chrome 드라이버 다운로드
  - 사이트 : https://sites.google.com/a/chromium.org/chromedriver/downloads

- FantomJS 드라이버 다운로드
  - 백엔드 기반의 브라우저 (Command line 기반)
  - 사이트 : http://phantomjs.org/download.html

- 위 드라이버를 PATH 경로상에 두거나 또는 현재 프로젝트 경로에 위치시킨다.

- 단, 2018년+ 기준 PhantomJS는 더이상 개발되지 않고 있기 때문에 앞으로는 크롬의 headless모드를 사용하는 것을 추천합니다.
- PhantomJS는 기본적으로 WebTesting을 위해 나온 Headless Browser다.(즉, 화면이 존재하지 않는다)
- 하지만 JS등의 처리를 온전하게 해주며 CLI환경에서도 사용이 가능하기 때문에, 만약 CLI서버 환경에서 돌아가는 크롤러라면 PhantomJS를 사용하는 것도 방법이다.

❑ 주요 메서드

- find_element_xxx : 단일 요소 선택
- find_elements_xxx : 여러 요서 요청 list
- 참조 : https://selenium-python.readthedocs.io/locating-elements.html

❑ 이벤트기반 메서드

- click : 클릭이벤트 호출
- send_keys : 키 보내기
- submit : 서브밋 이벤트 호출
- clear : 입력내용(input) 비우기
- execute_script : 스크립트 호출

❑ Selenium 실습 1

- 인터파크 여행사이트에서 여행지 검색하기
  - 오키나와 관련 정보를 구하기

```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
import time

url = "http://tour.interpark.com/"
driver = webdriver.Chrome('./chromedriver')

# 명시적 대기
# WebDriverWait(driver, 3)

# 암묵적으로 웹 자원 로드를 위해 3초까지 기다려 준다.
driver.implicitly_wait(3)

driver.get(url)

# 수정할경우 => 뒤에 내용이 붙어버림 => .clear() -> send_keys('내용')
driver.find_element_by_id("SearchGNBText").send_keys("오키나와")
driver.find_element_by_css_selector("button.search-btn").click()

# 또는 해당 JavaScript 호출
# driver.execute_script("searchBarModule.ClickForSearch();")
```

❑ Selenium 실습 1

- 오키나와 관련 정보를 구하기 계속

```
# 너무 빠르게 코드가 동작할수 있죠... time.sleep(3)
time.sleep(3)

lis = driver.find_elements_by_css_selector("div.oTravelBox li.boxItem")

for li in lis:
    print(li.text)
    # print("제목=", li.find_element_by_css_selector("h5.proTit").text)
    # print("가격=", li.find_element_by_css_selector("strong.proPrice").text)
    # print("정보=", li.find_element_by_css_selector("div.info-row .proInfo").text)
    print("-" * 60)

# 제목    h5.proTit .text
# 가격    strong.proPrice .text
# 여행정보 div.info-row .proInfo   .text

# driver.stop_client()
# driver.close()
```

❑ Selenium 실습 1

- 오키나와 관련 정보를 구하기 계속

```
import sqlite3
class DBManager:
    """
    DB Connection 을 관리하는 클래스.
    """

    def __init__(self):
        self.conn = None

    def __del__(self):
        self.db_free()

    def get_connection(self):
        self.conn=sqlite3.connect("../tour.db")
        # cur = self.conn.cursor()
        # cur.execute("""CREATE TABLE tours
        #     (title text, price text, area text, link text, img text, contents blob, reg_date text)""")
        # print("-" * 30)
        return self.conn

    def db_free(self):
        if self.conn:
            self.conn.close()
```

❑ Selenium 실습 1

- 오키나와 관련 정보를 구하기 계속

```
class Tour:
    """
    여행 관련 사이트에서 필용한 공통 정보 모델.
    """

    def __init__(self, title, price, area, link, img, contents=None):
        """ 제목, 가격, 지역, 링크정보, 이미지정보, 내용 등
        기간, 평점 필요한 경우 정보 추가 필요
        """
        self.title = title
        self.price = price
        self.area  = area
        self.link = link
        self.img  = img
        self.contents = contents
```

❑ Selenium 실습 1

- 오키나와 관련 정보를 구하기 계속

```
class TourDao:
    """
    여행정보를 처리하는 Dao
    """

    def insert_tour(self, conn, tour):
        cur = conn.cursor()
        cur.execute(
            """insert into tours (
                    title, price, area , link, img, contents, reg_date
                )values(
                    :title, :price, :area , :link, :img, :contents, datetime('now') ) """
        , tour.__dict__
        # , (tour.title, tour.price, tour.area, tour.link, tour.img, tour.contents)
        )
        conn.commit()
        return cur.rowcount
```

❑ Selenium 실습 1

- 오키나와 관련 정보를 구하기 계속

```
if __name__ == '__main__':
    # DBManager , Tour, TourDao 테스트
    # title, price, area, link, img, contents=None)
    t = Tour("오키나와로 출발해요", "325,555", "홍공->여수", "/test.do", "http://www.interpark.com/img")
    # title, price, area, link, img, contents=None
    print(t.__dict__)
    db = DBManager()
    conn = db.get_connection()
    dao = TourDao()
    cnt =  dao.insert_tour(conn, t)
    print("cnt", cnt)
    db.db_free()
```

❑ Selenium 사용시 주의 사항

- 크롤링을 자동화하여 사용하게 되면 임시 파일들이 쌓인다
  - 임시 파일을 삭제하는 방법을 생각해야 한다.

❑ Miscellaneous

- 이미지 저장하기
  - urllib.request 로 저장

```
# 문서를 파싱하여 이미지 경로를 구한다

img_url = "https://mblogthumb-phinf.pstatic.net/20100404_119/" ₩
          "choibokyu_1270372975416EGyEC_jpg/%C1%D6%C0%B1%B9%DF_choibokyu.jpg?type=w2"
save_name = os.environ.get("HOME") + os.sep + "milkis.jpg"
print(save_name)

urllib.request.urlretrieve(img_url, save_name)
```

- 세레니움의 현재화면 스크린 샷 (default PNG)
  - driver.save_screenshot('/path/to/file')
  - driver..get_screenshot_as_file('/path/to/file')

- Chrome headless 모드로 실행

```
options = webdriver.ChromeOptions()
options.add_argument('headless')
options.add_argument('window-size=1920x1080')
options.add_argument("disable-gpu")  # or ("--disable-gpu")
options.add_argument("lang=ko_KR")

driver = webdriver.Chrome('chromedriver', options=options)
```

❑ Python을 활용한 웹 크롤러 만들기

❑ 모듈 설치

```
pip install selenium
pip install bs4
pip install pymysql
```

❑ BeautifulSoup 모듈설치

- pip install bs4
- 크롤링의정의
- 개발환경구축
- 웹드라이버이해
- Selenium이해
- 웹드라이버를이용한selenium 주요api습득
- 크롤링타겟사이트분석및데이터접근실습
- Beautiful Soup 이해및api습득
- 수집데이터전처리및DB.처리

❑ 태그와 속성을 이용해서 가져오기

- 태그와 속성을 이용할 때 함수의 인자로 원하는 태그를 첫번째 인자로 그 다음에 속성:값의 형태로 dictionary 형태로 만들어서 넣어주면 된다.

- find_all('태그명', {'속성명' : '값' ...})
- find('태그명', {'속성명' : '값' ...})

- 예제 : 태그에서 id속성의 값이 ex_id인거 불러오기
- with open("example.html") as fp:
- soup = BeautifulSoup(fp, 'html.parser')
- ex_id_divs = soup.find('div', {'id' : 'ex_id'})
- print(ex_id_divs)
- # 출력 결과
-
- g
- h
- i
-

❑ HTML 구조를 이용해 부분부분 가져오기

- 예제 : id속성의 값이 ex_id인 태그에서 태그들만 가져오기
- with open("example.html") as fp:
- soup = BeautifulSoup(fp, 'html.parser')
- # id=ex_id인 div 태그를 가져와서
- ex_id_divs = soup.find("div", {"id":"ex_id"})
- # 그 태그들 안에서 p 태그를 가져온다.
- all_ps_in_ex_id_divs = ex_id_divs.find_all("p")
- print(all_ps_in_ex_id_divs)
- # 출력 결과
- [g , h , i]