

# Artificial Intelligence

## Lecture 4. Data Acquisition & Preprocessing

### II. Data Acquisition

Spring 2022

Prof. Jonghoon Chun, Ph.D.

E-mail : [jchun@mju.ac.kr](mailto:jchun@mju.ac.kr)  
Lecture Note: <http://lms.mju.ac.kr>

# Data Acquisition & Preprocessing

---

- Data (Know your data)
- Data Acquisition
- Data Preprocessing

# Scraping and Crawling

---

- Scraping
  - extracting data (from websites) in an automated manner
  - pulling content from a page
  - (웹사이트에서) 특정 정보만을 추출해서 수집하는 기술
  - 특정 정보만을 추출 하기 위해 HTML 구조 분석 및 가공 기술 필요
- Crawling
  - 웹사이트에서 (주기적으로) 정보를 추출해서 수집하는 기술
  - 웹검색엔진의 인덱스를 생성하기 위한 용도로 사용
  - following links to reach numerous pages
  - Crawler도 scraping을 해야 됨
  - Crawler, Spider, Bot
- Data Storage (조직 내 / website)
  - Text file, Database, NoSql, Excel 등
  - CSV, JSON, XML, YAML 형식으로 저장
  - 활용 가능한 머신러닝 라이브러리에 적합한 형태로 가공이 필요
  - 다양한 형식의 데이터를 처리하는 기능을 제공하는 라이브러리 존재

# URL Library

- Python urllib (Url Library)
  - a package that collects several modules for working with URLs
  - for opening and reading URLs
  - for parsing URLs
- urllib.request
  - Extensible library for opening URLs

```
In [2]: # 라이브러리 읽어 들이기
import urllib.request

# Url과 저장 경로 지정하기
url = "http://uta.pw/shodou/img/28/214.png"
savename = "test.png"

# 파일 다운로드
urllib.request.urlretrieve(url, savename)
print("저장 되었습니다.")
```

저장 되었습니다.



# 파일을 메인메모리에 저장

```
In [1]: # 라이브러리 읽어 들이기
import urllib.request

# Url과 저장 경로 지정하기
url = "http://uta.pw/shodou/img/28/214.png"
savename = "test2.png"

# Main memory에 저장
mem = urllib.request.urlopen(url).read()

# 파일로 저장하기 (with block이 끝나면 f를 자동으로 close함)
# wb: write binary mode
with open(savename, mode = "wb") as f:
    f.write(mem)
    print("저장 되었습니다.")
```

저장 되었습니다.



# Decode 주의

```
In [7]: import urllib.request  
  
# 데이터 읽어 들이기  
url = "https://www.google.com/"  
mem = urllib.request.urlopen(url).read()  
  
# binary를 문자열로 변환하기  
print(mem.decode("utf-8"))
```

```
<!doctype html><html itemscope="" itemtype="http:  
set=UTF-8" http-equiv="Content-Type"><meta conten  
mprop="image"><title>Google</title><script nonce=  
oSl8QWtJDYDA'  
33519,329527,1  
5,1236,4323,49  
2,1000,2,1000
```

UTF-8: Unicode (or Universal  
Coded Character  
Set) Transformation Format – 8-bit

```
In [9]: import urllib.request  
  
# 데이터 읽어 들이기  
url = "https://www.google.co.kr/"  
mem = urllib.request.urlopen(url).read()  
  
# binary를 문자열로 변환하기  
print(mem.decode("euc-kr"))
```

```
<!doctype html><html itemscope="" itemtype="http:  
set=UTF-8" http-equiv="Content-Type"><meta conten  
mprop="image"><title>Google</title><script nonce=  
YmN8wWIiL74Cw',kEXPI:'0,18167,1335580,57,1957,242  
33570,303171,26305,1294,12383,4855,32692,15247,86
```

Extended Unix Code (EUC)  
is a multibyte character  
encoding system (한, 중, 일  
에서 주로 사용)

# 예제 사이트

```
In [5]: # IP 확인 API로 접근해서 결과 출력하기
import urllib.request

# 데이터 읽어 들이기
url = "http://api.aoikujira.com/ip/ini"
res = urllib.request.urlopen(url)
data = res.read()

# binary를 문자열로 변환하기
text = data.decode("utf-8")
print(text)

[ip]
API_URI=http://api.aoikujira.com/ip/get.php
REMOTE_ADDR=125.132.56.119
REMOTE_HOST=125.132.56.119
REMOTE_PORT=60190
HTTP_HOST=api.aoikujira.com
HTTP_USER_AGENT=Python-urllib/3.7
HTTP_ACCEPT_LANGUAGE=
HTTP_ACCEPT_CHARSET=
SERVER_PORT=80
FORMAT=ini
```



# Request parameter

- Url encoding
  - **GET** carries request parameter appended in **URL** string
  - E.g., <http://naver.com?name1=value1&name2=value2....>
  - 한글, 특수문자 등을 자동으로 encoding

```
In [20]: import urllib.request
import urllib.parse
API = "https://search.naver.com/search.naver"
values = {
    "sm": "top_hty",
    "fbm": "1",
    "ie": "utf8",
    "query": "초콜릿"
}
params = urllib.parse.urlencode(values)
# Request Url을 생성
url = API + "?" + params
print("url =", url)

url = https://search.naver.com/search.naver?sm=top_hty&fbm=1&ie=utf8&query=%EC%B4%88%EC%BD%9C%EB%A6%BF
```

# Request parameter

# BeautifulSoup

---

- Python library for pulling data out of HTML and XML files
- 다운로드 기능은 없음
- BeautifulSoup 설치(pip 또는 pip3 이용)
  - pip3 install beautifulsoup4
- Beautiful Soup 4.9.0 documentation
  - <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

# BeautifulSoup 기본 사용

```
In [3]: # 라이브러리 읽어 들이기
from bs4 import BeautifulSoup

# 분석 대상 HTML
html = """
<html>
  <body>
    <h1> 스크래핑이란?</h1>
    <p> 웹 페이지를 분석해서 </p>
    <p> 원하는 부분을 추출하는 기능 </p>
  </body>
</html>
"""

# HTML 분석하기 (beautifulsoup 인스턴스 생성)
# BeautifulSoup(html, 분석기) 사용하고자 하는 파서 지정
soup = BeautifulSoup(html, 'html.parser')

# 원하는 부분 추출하기
h1 = soup.html.body.h1 #<html><body><h1> 요소 추출
p1 = soup.html.body.p

# next_sibling을 활용 (한번 사용하면 </p> 다음의 공백이나 줄바꿈을 출력하므로 한번 더 사용)
p2 = p1.next_sibling.next_sibling

# 출력하기
print("h1 = " + h1.string)
print("p1 = " + p1.string)
print("p2 = " + p2.string)

h1 = 스크래핑이란?
p1 = 웹 페이지를 분석해서
p2 = 원하는 부분을 추출하는 기능
```

# Finding an element

- Find()
  - 태그 명칭으로 검색하여 추출

```
In [8]: # 라이브러리 읽어 들이기
from bs4 import BeautifulSoup

# 분석 대상 HTML
html = """
<html>
    <body>
        <h1> 스크래핑이란?</h1>
        <p> 웹 페이지를 분석해서 </p>
        <p> 원하는 부분을 추출하는 기능 </p>
    </body>
</html>
"""

# HTML 분석하기 (beautifulsoup 인스턴스 생성)
# BeautifulSoup(html, 분석기) 사용하고자 하는 파서 지정
soup = BeautifulSoup(html, 'html.parser')

# find 메소드로 원하는 부분 출력하기
title = soup.find("h1")
body = soup.find("p")

# text 출력하기
print("title = " + title.string)
print("body = " + body.string)

title = 스크래핑이란?
body = 웹 페이지를 분석해서
```

# Finding an element

```
In [9]: # 라이브러리 읽어 들이기
from bs4 import BeautifulSoup

# 분석 대상 HTML
html = """
<html>
  <body>
    <h1> 스크래핑이란?</h1>
    <p> 웹 페이지를 분석해서 </p>
    <p> 원하는 부분을 추출하는 기능 </p>
  </body>
</html>
"""

# HTML 분석하기 (beautifulsoup 인스턴스 생성)
# BeautifulSoup(html, 분석기) 사용하고자 하는 파서 지정
soup = BeautifulSoup(html, 'html.parser')

# find 메소드로 원하는 부분 추출하기
title = soup.find("h1")

# find_all 메소드로 추출하기
body = soup.find_all("p")

# title 출력하기
print("title = " + title.string)

# body 출력하기
for p in body:
    print("body = " + p.string)

title = 스크래핑이란?
body = 웹 페이지를 분석해서
body = 원하는 부분을 추출하는 기능
```

# Finding an element

```
In [11]: from bs4 import BeautifulSoup

html = """
<html>
    <ul>
        <li><a href="http://www.naver.com">naver</a></li>
        <li><a href="http://www.daum.net">daum</a></li>
    </ul>
</html>
"""

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

# find_all 메소드로 추출하기
links = soup.find_all("a")

for a in links:
    href = a.attrs['href'] # attrs로 tag의 속성값 추출
    text = a.string
    print(text, ">", href)

naver > http://www.naver.com
daum > http://www.daum.net
```

# BeautifulSoup 사용 웹페이지 추출 및 출력

```
from bs4 import BeautifulSoup
import urllib.request as req
url = "http://www.weather.go.kr/weather/forecast/mid-term-rss3.jsp"

# urlopen()으로 데이터 가져오기
# 데이터 로드는 원래 req.urlopen(url).read()로 해야 하나 open만 해도
# BeautifulSoup이 자동으로 load함
res = req.urlopen(url)

# 분석하기
soup = BeautifulSoup(res, "html.parser")

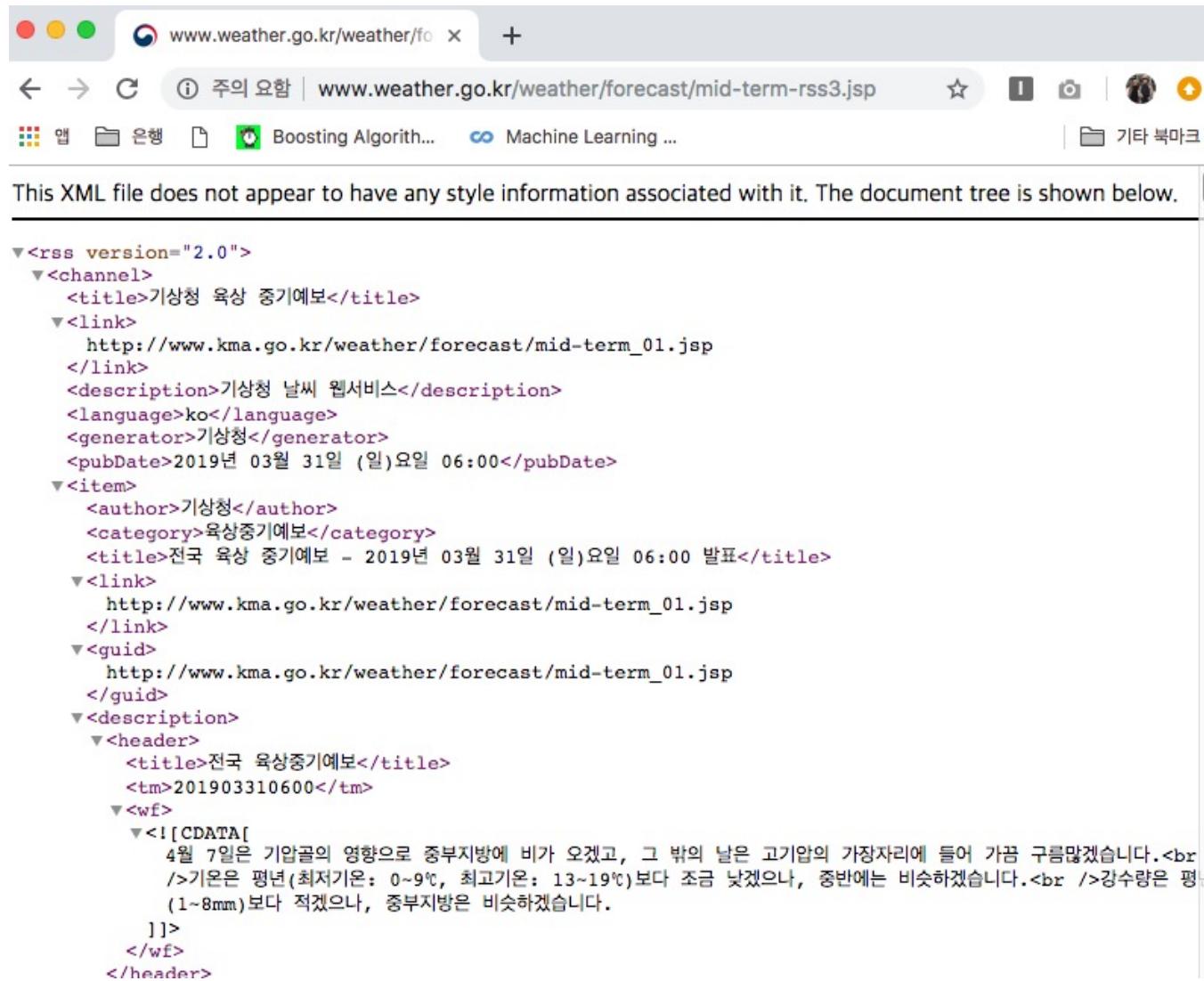
# 데이터 추출 및 출력
title = soup.find("title").string
wf = soup.find("wf").string

print(title)
print(wf)
```

기상청 육상 중기예보

4월 7일은 기압골의 영향으로 중부지방에 비가 오겠고, 그 밖의 날은 고기압의 가장자리에 들어 가끔 구름많겠습니다.<br />기온은 평년 기온: 13~19℃)보다 조금 낮겠으나, 중반에는 비슷하겠습니다.<br />강수량은 평년(1~8mm)보다 적겠으나, 중부지방은 비슷하겠습니다.

# BeautifulSoup 사용 웹페이지 추출 및 출력



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<rss version="2.0">
  <channel>
    <title>기상청 육상 중기예보</title>
    <link>http://www.kma.go.kr/weather/forecast/mid-term_01.jsp</link>
    <description>기상청 날씨 웹서비스</description>
    <language>ko</language>
    <generator>기상청</generator>
    <pubDate>2019년 03월 31일 (일)요일 06:00</pubDate>
    <item>
      <author>기상청</author>
      <category>육상중기예보</category>
      <title>전국 육상 중기예보 - 2019년 03월 31일 (일)요일 06:00 발표</title>
      <link>http://www.kma.go.kr/weather/forecast/mid-term_01.jsp</link>
      <guid>http://www.kma.go.kr/weather/forecast/mid-term_01.jsp</guid>
      <description>
        <header>
          <title>전국 육상중기예보</title>
          <tm>201903310600</tm>
          <wf><![CDATA[
            4월 7일은 기압골의 영향으로 중부지방에 비가 오겠고, 그 밖의 날은 고기압의 가장자리에 들어 가끔 구름 많겠습니다.<br/>
            />기온은 평년(최저기온: 0~9%, 최고기온: 13~19%)보다 조금 낮겠으나, 중반에는 비슷하겠습니다.<br />강수량은 평년(1~8mm)보다 적겠으나, 중부지방은 비슷하겠습니다.
          ]]>
        </wf>
      </header>
    </description>
  </item>
</channel>
</rss>
```

# CSS Selector

---

- Methods
  - soup.select\_one() : 1개의 element 추출
  - soup.select(): n개의 element들을 추출
- Element 지정
  - 태그에 id나 class가 설정 되어 있는 경우
  - E.g., <div id="meigen">의 경우 div#meigen  
<div class="meigen">의 경우 div.meigen

# CSS Selector 예제

```
from bs4 import BeautifulSoup

# 분석 대상 HTML
html = """
<html>
  <body>
    <div id = "meigen">
      <h1>위키북스 도서</h1>
      <ul class = "items">
        <li>유니티 게임 이펙트 입문 </li>
        <li>스위프트로 시작하는 아이폰 앱 개발 교과서 </li>
        <li>모던 웹사이트 디자인의 정석 </li>
      </ul>
    </div>
  </body>
</html>
"""

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

h1 = soup.select_one("div#meigen > h1").string
print("h1 = ", h1)

li_list = soup.select("div#meigen > ul.items > li")

for li in li_list:
    print("li = ", li.string)
```

```
h1 = 위키북스 도서
li = 유니티 게임 이펙트 입문
li = 스위프트로 시작하는 아이폰 앱 개발 교과서
li = 모던 웹사이트 디자인의 정석
```

# 예제: Naver에서 환율 정보 추출

- https://finance.naver.com/marketindex/

The screenshot shows the Naver Finance Market Index page. At the top, there is a search bar and a menu bar with options like NAVER 금융, 종목명·펀드명·환율명·원자재명 입력, 통합검색, 로그인, and more. Below the menu, there are tabs for 금융 홍보, 국내증시, 해외증시, 시장지표 (highlighted in green), 펀드, 투자전략, 뉴스, MY, and 추천종목. A note says '(중요) MY 메마네역 및 MY 펀드서비스가 종료되었습니다.' and another note says '네이버 증권 200% 활용하자! 확장액 출시!'.

The main content area displays exchange rates for four currencies:

- 미국USD**: 1,137.00원 ▼0.50 (01/02 ~ 2019.03.29 KEB하나은행 기준)
- 일본JPY(100엔)**: 1,026.40원 ▼4.74 (01/01 ~ 2019.03.30 모닝스타 기준)
- 유럽연합EUR**: 1,277.48원 ▼2.15 (01/02 ~ 2019.03.29 KEB하나은행 기준)
- 중국CNY**: 169.13원 ▲0.37 (01/02 ~ 2019.03.29 KEB하나은행 기준)

On the right side, a detailed view of the USD exchange rate is shown with a callout box highlighting the current value of 1,137.00 and its change of 0.50. The chart shows the USD/KRW exchange rate from January 2nd to March 29th, 2019, with a green line indicating an upward trend.

Below the charts, there is a section for '주요뉴스' (Major News) and '국내 시장 금리' (Domestic Market Interest Rates). The news section lists several articles with dates and times. The interest rate section lists various rates with their current values and changes.

날짜	제목	날짜	제목
03.30 14:23	국제유가, 1분기 32% 올라…60.14달러	03.30 10:01	국제유가, 상승 마감…WTI, 1-4분
03.30 09:51	(원자재시장) 국제유가, 무역협상 기대로…	03.30 06:20	국제유가, 미·중 무역협상 진전 기대에 올…
03.30 05:14	국제유가, 상승 마감…WTI, 1분기 32…	03.29 17:42	(채권 마감)국고채 소폭 조정…
03.29 17:36	'외환시장 개입' 첫 공개…환율조작국…	03.29 16:47	국고채 금리 일제히 반등…3년물 연 1.6…
03.29 16:45	[외환마감]미·중 무역협상 긍정론에 '무게…'	03.29 16:27	(채권마감)안전 자산 멀리 진정…미·중 무…

금리 종류	금리	변동
CD (91일)	1.90	- 0.00
콜 금리	1.76	▲ 0.02
국고채 (3년)	1.69	▲ 0.01
회사채 (3년)	2.16	▲ 0.01
COFIX 잔액	2.02	▲ 0.01
COFIX 신규취급액	1.92	▼ 0.07

At the bottom, there are tabs for 환전고시 환율, 국제시장 환율, 유가·금시세, and 원자재.

# 예제: Naver에서 환율 정보 추출

```
In [27]: import urllib.request  
from bs4 import BeautifulSoup  
  
url = "https://finance.naver.com/marketindex/"  
response = urllib.request.urlopen(url)  
  
soup = BeautifulSoup(response, "html.parser")  
  
results = soup.select("span.value")  
  
for result in results:  
    print(result.string)
```

1,137.00  
1,026.40  
1,277.48  
169.13  
110.8700  
1.1231  
1.3044  
96.8400  
60.14  
1394.19  
1293.0  
47130.71



span.value에  
해당하는 모든  
환율이 다 표시  
됨!

# 예제: Naver에서 환율 정보 추출

```
In [29]: import urllib.request
from bs4 import BeautifulSoup

url = "https://finance.naver.com/marketindex/"
response = urllib.request.urlopen(url)

soup = BeautifulSoup(response, "html.parser")

result = soup.select_one("div.head_info > span.value")
print("usd/krw =", result.string)
```

```
usd/krw = 1,137.00
```

```
In [30]: import urllib.request
from bs4 import BeautifulSoup

url = "https://finance.naver.com/marketindex/"
response = urllib.request.urlopen(url)

soup = BeautifulSoup(response, "html.parser")

result = soup.select_one("div.head_info.point_dn > span.value")
print("usd/krw =", result.string)
```

```
usd/krw = 1,137.00
```

# CSS Selector (Chrome)

The screenshot shows a Korean Wikipedia page for Yun Dong-ju (윤동주). The page includes a sidebar with user preferences and a main content area with a biography, a portrait photo, and a list of poems. A red arrow points from the bottom of the page to a screenshot of the Chrome DevTools context menu, specifically highlighting the 'Copy selector' option.

Red arrow pointing to the 'Copy selector' option in the DevTools context menu.

#mw-content-text > div.mw-parser-output > ul:nth-child(6) > li > b > a

# CSS Selector

---

- 하늘과 바람과 별과 시
  - #mw-content-text > div.mw-parser-output > ul:nth-child(6) > li > b > a
- 서시
  - #mw-content-text > div.mw-parser-output > ul:nth-child(6) > li > ul > li:nth-child(1) > a
- 리스트 아이템(li tag)의 모든 후손을 선택
  - #mw-content-text > div.mw-parser-output > ul:nth-child(6) > li a

'>' 는 자식

빈칸은 후손

# CSS Selector

```
In [22]: import urllib.request as req
from bs4 import BeautifulSoup

# 저자: 윤동주 %EC%A0%80%EC%9E%90:%EC%9C%A4%EB%8F%99%EC%A3%BC
url = "https://ko.wikisource.org/wiki/%EC%A0%80%EC%9E%90:%EC%9C%A4%EB%8F%99%EC%A3%BC"
res = req.urlopen(url)
soup = BeautifulSoup(res, "html.parser")

# 하늘과 바람과 별과 시 선택
name = soup.select_one("#mw-content-text > div.mw-parser-output > ul:nth-child(6) > li > b > a")

print("-", name.string)

- 하늘과 바람과 별과 시
```

```
In [25]: import urllib.request as req
from bs4 import BeautifulSoup

# 저자: 윤동주 %EC%A0%80%EC%9E%90:%EC%9C%A4%EB%8F%99%EC%A3%BC
url = "https://ko.wikisource.org/wiki/%EC%A0%80%EC%9E%90:%EC%9C%A4%EB%8F%99%EC%A3%BC"
res = req.urlopen(url)
soup = BeautifulSoup(res, "html.parser")

# 서시 선택
name = soup.select_one("#mw-content-text > div.mw-parser-output > ul:nth-child(6) > li > ul > li:nth-child(1) > a")

print("-", name.string)

- 서시
```

# CSS Selector

```
In [26]: import urllib.request as req
from bs4 import BeautifulSoup

# 저자: 운동주 %EC%A0%80%EC%9E%90:%EC%9C%A4%EB%8F%99%EC%A3%BC
url = "https://ko.wikisource.org/wiki/%EC%A0%80%EC%9E%90:%EC%9C%A4%EB%8F%99%EC%A3%BC"
res = req.urlopen(url)
soup = BeautifulSoup(res, "html.parser")

# li의 후손 a tag를 모두 선택
a_list = soup.select("#mw-content-text > div.mw-parser-output > ul:nth-child(6) > li a")

for a in a_list:
    name = a.string
    print("-", name)
```

- 하늘과 바람과 별과 시
- 증보판
- 서시
- 자화상
- 소년
- 눈 오는 지도
- 돌아와 보는 밤
- 병원
- 새로운 길
- 간판 없는 거리
- 태초의 아침
- 또 태초의 아침
- 새벽이 올 때까지
- 무서운 시간
- 십자가
- 바람이 불어
- 슬픈 족속
- 눈감고 간다
- 또 다른 고향
- 길
- 별 해는 밤

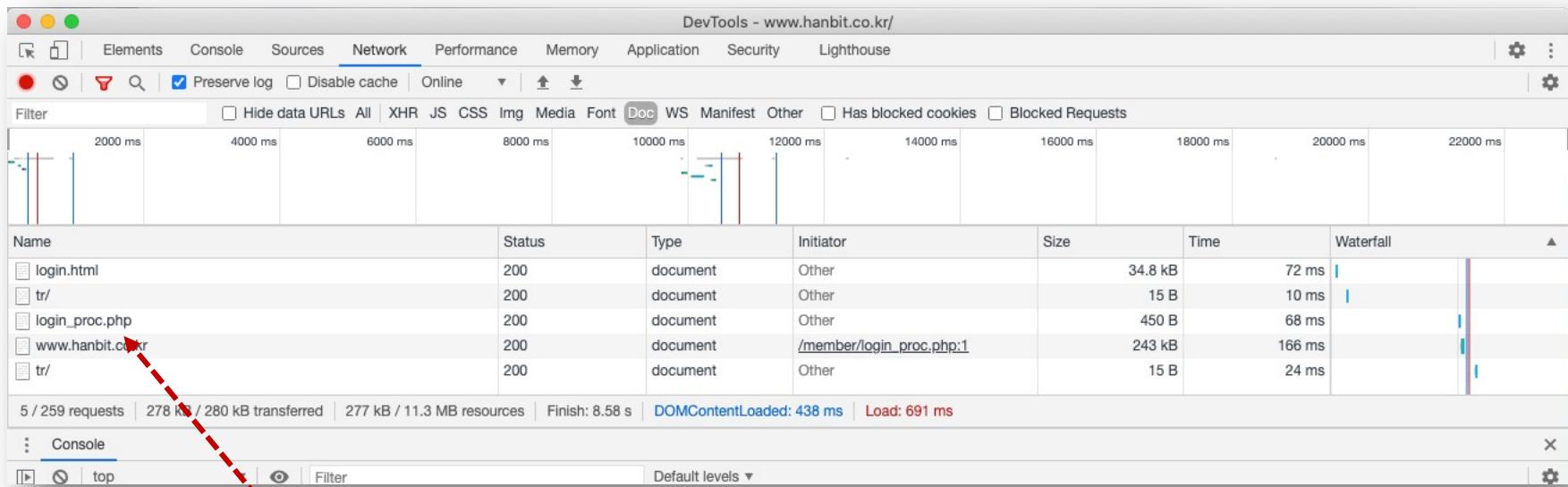
# 로그인이 필요한 웹페이지

## ■ 한빛출판네트워크

The screenshot shows the homepage of hanbit.co.kr. At the top, there's a navigation bar with links to various websites like Wells Fargo, Bank of America, TJX Rewards, KIAT, etc. Below that is a secondary navigation bar with links to 'HOME', '한빛미디어', '한빛아카데미', '한빛비즈', '한빛라이프', '한빛매거진', '리얼타임', '한빛정보교과서', and '한빛대관서비스'. On the right side of the header, there are links for '로그인', '회원가입', 'マイハビット', '장바구니', and 'ENGLISH'. The main content area features a large banner with the text '부의 흐름을 짚어내는 빠송의 입지분석 바이블' (A guide to analyzing the flow of wealth by Baesung). Below the banner, there's a photo of three people wearing face masks looking at a computer screen. To the right of the banner are three circular icons labeled '기획/원고 모집' (Planning/稿征), '교수전용' (For professors), and '자료실' (Resource room). A green button at the bottom right says '자료실' with a download icon.

# 로그인이 필요한 웹페이지

- 개발자 도구(f12)를 활용하여 로그인 방식 확인
- Google Chrome에서 다음 순으로 진행
  - [검사] -> [network]탭 선택
  - [Doc]탭 선택
  - Preserve log 체크
  - 로그인 진행



login\_proc.php에 의해서 로그인 과정이 처리되는 것을 확인할 수 있음

# 로그인 과정 확인

The screenshot shows two separate network requests in the Chrome DevTools Network tab.

**Request 1: login\_proc.php**

- Request URL: [https://www.hanbit.co.kr/member/login\\_proc.php](https://www.hanbit.co.kr/member/login_proc.php)
- Request Method: POST
- Status Code: 200 OK
- Remote Address: 218.38.58.195:443
- Referrer Policy: strict-origin-when-cross-origin

**Request 2: index.html**

- Request URL: <https://www.hanbit.co.kr/index.html>
- Request Method: GET
- Status Code: 200 OK
- Remote Address: 218.38.58.195:443
- Referrer Policy: strict-origin-when-cross-origin

Both requests show a "Form Data" section with the following data:

- return\_url: <https://www.hanbit.co.kr/index.html>
- m\_id: [REDACTED]
- m\_passwd: [REDACTED]

# 로그인 과정 + 데이터 수집

```
In [27]: import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin

USER = "*****"
PASS = "*****"

session = requests.session()

login_info = {
    "m_id": USER,
    "m_passwd" : PASS
}

url_login = "https://www.hanbit.co.kr/member/login_proc.php"
res = session.post(url_login, data = login_info)
res.raise_for_status()

url_mypage = "https://www.hanbit.co.kr/myhanbit/myhanbit.html"
res = session.get(url_mypage)
res.raise_for_status()

soup = BeautifulSoup(res.text, "html.parser")
mileage = soup.select_one(".mileage_section1 span").get_text()
ecoin = soup.select_one(".mileage_section2 span").get_text()

print("마일리지: " + mileage)
print("이코인: " + ecoin)
```

```
마일리지: 3,000
이코인: 0
```

# GET vs. POST

---

- GET 방식
  - `r = requests.get("https://www.google.com")`
  - `r = requests.get("https://www.google.com?n1=v1&n2=v2")`
- POST 방식
  - `form_data = {"key1": "value1", "key2": "value2"}`
  - `r = requests.post("https://www.google.com", data=form_data)`
- Request 객체를 사용할 수 없는 경우도 있음!
  - 보안이 강화된 사이트의 경우 request 객체를 통한 로그인/크롤링 불가
  - 브라우저를 원격으로 조작하여 접근 (e.g., Selenium 활용)

# GET 예제

```
In [38]: # 데이터 가져오기
import requests
res = requests.get("http://api.aoikujira.com/time/get.php")

# 텍스트 형식으로 데이터 추출하기
text = res.text
print(text)

# 바이너리 형식으로 데이터 추출하기
bin = res.content
print(bin)
```

```
2019/03/31 21:42:30
b'2019/03/31 21:42:30'
```

```
In [43]: # 이미지 데이터 추출하기
import requests
res = requests.get("https://wikibook.co.kr/images/cover/l/9791158391478.jpg")

# 바이너리 형식으로 데이터 저장하기
with open("test.png", "wb") as f:
    f.write(res.content)
print("saved")
```

```
saved
```

# WEB API를 활용한 데이터 수집

# Web API (Open API)

---

- API를 통해 특정 데이터를 제공 (공개)
  - JSON, XML 형태로 API Call을 return
  - 무료나 유료
  - 데이터 공개를 통한 정보 제공
  - 임의의 대규모 스크래핑을 사전에 방지함으로써 서버 부하 감소
- Web API 제공 주체
  - Naver, Daum 등 포털 사이트
  - Auction, Amazon, 11번가 등 온라인 쇼핑 사이트
  - data.go.kr, 서울시 열린 광장, 기상청 등 정부부처 및 공공기관
  - 기타 해외 사이트

# API 사용 예제 (OpenWeatherMap)

---

- <https://openweathermap.org>
  - 절차
    - 개발자 사용자 등록
    - API 키 발급 받음
    - 주어진 API를 통해 데이터 다운로드
  - 현재와 5일까지의 날씨만 무료로 제공됨
- 계정 생성

# 웹 API 사용 예(OpenWeatherMap)

- [API keys] 탭 선택
  - Key를 복사하여 기억해 놓아야 함(key 발급 후 10분 후에 사용가능)

The screenshot shows the 'API keys' section of the OpenWeatherMap account management interface. At the top, there are several navigation tabs: 'Setup', 'API keys' (which is highlighted with a red dashed box), 'My Services', 'My Payments', 'Billing plans', 'Map editor', 'Block logs', and 'History bulk'. To the right of these tabs is a 'Logout' button. Below the tabs, a message states: 'Activation of an API key for **Free** and **Startup accounts** takes **10 minutes**. For **other accounts** it takes from **10 to 60 minutes**. You can generate as many API keys as needed for your subscription. We accumulate the total load from all of them.' On the left, under the heading 'Key', is a large text input field containing the API key 'cb87576d7bd2601e3997973249420850', which is also highlighted with a red dashed box. To the right of the key is a 'Name' field set to 'Default' with edit and delete icons. On the far right, there is a 'Create key' section with a 'Name' input field and a 'Generate' button.

# 웹 API 사용 예(OpenWeatherMap)

```
In [48]: import requests
import json

# API 키를 지정
apikey = "71ae3abd7fb3642aabc18be712c44b38"

# 날씨 확인 도시 지정
cities = ["Seoul,KR", "Tokyo,JP"]

# API 지정
api = "http://api.openweathermap.org/data/2.5/weather?q={city}&APPID={key}"

# 캘빈 온도를 섭씨로 변환
k2c = lambda k: k - 273.15

# 도시 정보 추출
for name in cities:
    # API url 구성
    url = api.format(city=name, key=apikey)
    # API 요청을 통해 데이터 추출
    r = requests.get(url) # r.text는 JSON형식의 문자열
    # 문자열을 파이썬 자료형(리스트나 딕셔너리)으로 변환
    data = json.loads(r.text)
    # print(data) 결과 출력하기
    print("+" + " 도시 =", data["name"])
    print("+" + " 날씨 =", data["weather"][0]["description"])
    print("+" + " 최저 기온 =", k2c(data["main"]["temp_min"]))
    print("+" + " 최고 기온 =", k2c(data["main"]["temp_max"]))
    print("+" + " 습도 =", data["main"]["humidity"])
    print("+" + " 기압 =", data["main"]["pressure"])
    print("+" + " 풍향 =", data["wind"]["deg"])
    print("+" + " 풍속 =", data["wind"]["speed"])
    print("")
```

+ 도시 = Seoul	날씨 = haze
최저 기온 = 0.0	최고 기온 = 5.0
습도 = 27	기압 = 1022
풍향 = 240	풍속 = 1.5
+ 도시 = Tokyo	
날씨 = scattered clouds	최저 기온 = 7.220000000000027
최고 기온 = 12.220000000000027	습도 = 40
기압 = 1010	풍향 = 330
풍속 = 6.7	

# JSON(JavaScript Object Notation) 객체 데이터 형식

```
In [50]: d":5509,"message":0.006,"country":"KR","sunrise":1553980804,"sunset":1554025955},"id":1835848,"name":"Seoul","cod":200}
```

```
Out[50]: {'coord': {'lon': 126.98, 'lat': 37.57},
  'weather': [{'id': 721,
    'main': 'Haze',
    'description': 'haze',
    'icon': '50n'}],
  'base': 'stations',
  'main': {'temp': 277.16,
    'pressure': 1022,
    'humidity': 38,
    'temp_min': 276.15,
    'temp_max': 278.15},
  'visibility': 10000,
  'wind': {'speed': 1.5, 'deg': 340},
  'clouds': {'all': 1},
  'dt': 1554039600,
  'sys': {'type': 1,
    'id': 5509,
    'message': 0.006,
    'country': 'KR',
    'sunrise': 1553980804,
    'sunset': 1554025955},
  'id': 1835848,
  'name': 'Seoul',
  'cod': 200}
```

<https://api.openweathermap.org/data/2.5/weather?q=Seoul,KR&APPID=71ae3abd7fb3642aabc18be712c44b38>

```
{"coord": {"lon": 126.98, "lat": 37.57}, "weather": [{"id": 721, "main": "Haze", "description": "haze", "icon": "50n"}], "base": "stations", "main": {"temp": 277.16, "pressure": 1022, "humidity": 38, "temp_min": 276.15, "temp_max": 278.15}, "visibility": 10000, "wind": {"speed": 1.5, "deg": 340}, "clouds": {"all": 1}, "dt": 1554039600, "sys": {"type": 1, "id": 5509, "message": 0.006, "country": "KR", "sunrise": 1553980804, "sunset": 1554025955}, "id": 1835848, "name": "Seoul", "cod": 200}
```

# JSON 처리 (Python)

- <https://api.github.com/repositories>

```
[  
 {  
   "id": 1,  
   "node_id": "MDEwOlJlcG9zaXRvcnkx",  
   "name": "grit",  
   "full_name": "mojombo/grit",  
   "private": false,  
   "owner": {  
     "login": "mojombo",  
     "id": 1,  
     "node_id": "MDQ6VXNlcjE=",  
     "avatar_url": "https://avatars0.githubusercontent.com/u/1?v=4",  
     "gravatar_id": "",  
     "url": "https://api.github.com/users/mojombo",  
     "html_url": "https://github.com/mojombo",  
     "followers_url": "https://api.github.com/users/mojombo/followers",  
     "following_url": "https://api.github.com/users/mojombo/following{/other_user}",  
     "gists_url": "https://api.github.com/users/mojombo/gists{/gist_id}",  
     "starred_url": "https://api.github.com/users/mojombo/starred{/owner}{/repo}",  
     "subscriptions_url": "https://api.github.com/users/mojombo/subscriptions",  
     "organizations_url": "https://api.github.com/users/mojombo/orgs",  
     "repos_url": "https://api.github.com/users/mojombo/repos",  
     "events_url": "https://api.github.com/users/mojombo/events{/privacy}",  
     "received_events_url": "https://api.github.com/users/mojombo/received_events",  
     "type": "User",  
     "site_admin": false  
 },  
 ...  
 ]
```

# JSON 처리 (Python)

```
In [52]: import urllib.request as request
import json

json_str = request.urlopen("https://api.github.com/repositories").read()
output = json.loads(json_str)

for item in output:
    print(item["name"])
    print(item["full_name"])
    print(item["owner"]["login"])
    print()

grit
mojombo/grit
mojombo

merb-core
wycats/merb-core
wycats

rubinius
rubinius/rubinius
rubinius

god
mojombo/god
mojombo

jsawesome
vanpelt/jsawesome
vanpelt
```

# JSON형식으로 출력

```
In [57]: import json
price = {
    "date": "2019-04-02",
    "price": {
        "apple": 80,
        "orange": 20,
        "banana": 40
    }
}

# json.dumps는 파이썬 객체(dictionary)를
# json 형식의 문자열로 변환, 즉, JSON.loads와 역 개념임
json_str = json.dumps(price)
print(json_str)

# 다시 파이썬 객체로 변환(dictionary)
data = json.loads(json_str)
print(data["date"])
print(data["price"]["apple"])

{"date": "2019-04-02", "price": {"apple": 80, "orange": 20, "banana": 40}}
2019-04-02
80
```

# CSV(Comma Separated Values)

- Comma(,)로 구분하여 저장한 데이터 형식

```
ID, 이름, 가격  
1000, 비누, 300  
1001, 장갑, 150  
1002, 마스크, 230
```

```
ID, 이름, 가격  
"1000", "비누", "300"  
"1001", "장갑", "150"  
"1002", "마스크", "230"
```

- ""를 사용하는 것도 가능
  - 공백이 있는 경우, 반드시 "" 사용 (예, "상품 번호")
- TSV(Tab)
  - SSV(Space)

# CSV 파일 읽기

- 파이썬 파일 입출력 함수를 이용하여 CSV파일을 입력 받아 리스트 생성

```
In [4]: import codecs  
filename = "euckr.csv"  
  
# 일반 파일 입력함수 사용  
csv = open(filename, "r").read()  
  
# 줄 단위로 리스트 생성  
rows = csv.split("\n")  
data = []  
for row in rows:  
    if row == "": continue  
    # comma로 서브 리스트 생성  
    cells = row.split(",")  
    data.append(cells)  
# 결과 출력하기  
for c in data:  
    print(c[1], c[2])
```

이름	가격
비누	300
장갑	150
마스크	230

# CSV 파일 읽기

- Euc-kr로 저장된 파일의 경우 codecs 라이브러리를 이용

```
In [20]: import codecs
# EUC_KR로 저장된 CSV 파일 읽기
filename = "euckr.csv"
csv = codecs.open(filename, "r", "euc_kr").read()

# CSV를 파이썬 리스트로 변환하기
data = []
rows = csv.split("\r\n")
for row in rows:
    if row == "": continue
    cells = row.split(",")
    data.append(cells)
# 결과 출력하기
for c in data:
    print(c[1], c[2])
```

이름 가격  
비누 300  
장갑 150  
마스크 230

# CSV 모듈을 이용한 CSV파일 읽기

```
In [17]: # CSV 라이브러리를 이용한 csv 파일 읽기
import csv

#CSV 파일 읽기
filename = "euckr.csv"
with open(filename, "r") as csvfile:
    reader = csv.reader(csvfile, delimiter = ", ", quotechar = "'")
    print(reader) # CSV object 출력

    for cells in reader:
        print(cells[1], cells[2])

<_csv.reader object at 0x1046ef828>
이름  가격
비누  300
장갑  150
마스크  230
```

# CSV로 출력

- codecs를 이용하여 파일을 열고, csv.writer 객체를 생성
  - 아래의 예는 euc-kr로 저장하는 예
  - utf-8로 저장할 경우 excel에서 읽지 못할 수 있음
- 한 라인씩 출력하기 위해서는 writer.writerow를 이용

In [19]: `import csv, codecs`

```
with codecs.open("test.csv", "w", "euc_kr") as csvfile:  
    writer = csv.writer(csvfile, delimiter = ",", quotechar = "'")  
  
    # write method를 사용하여 한줄씩 출력  
    writer.writerow(["ID", "이름", "가격"])  
    writer.writerow(["1000", "SD카드", 30000])  
    writer.writerow(["1001", "키보드", 21000])  
    writer.writerow(["1002", "마우스", 15000])
```

ID, 이름, 가격  
1000, SD카드, 30000  
1001, 키보드, 21000  
1002, 마우스, 15000

# CSV로 출력

- utf-8로 저장할 경우 excel에서 읽지 못할 수 있음

```
In [1]: import csv, codecs

with codecs.open("test1.csv", "w", "utf_8") as csvfile:
    writer = csv.writer(csvfile, delimiter = ",", quotechar = '"')

    writer.writerow(["ID", "이름", "가격"])
    writer.writerow(["1000", "SD카드", 30000])
    writer.writerow(["1001", "키보드", 21000])
    writer.writerow(["1002", "마우스", 15000])
```

	A	B	C
1	ID	대팻 媛寃	
2	1000	SD移대牢固树立	30000
3	1001	사낫	21000
4	1002	留ձ슉	15000
5			

# 기타

---

- Periodic Crawling
  - Cron : Linux, MacOS 에서 사용
  - Task Scheduler: Windows 에서 사용
- Excel이나 데이터베이스를 이용한 입출력 기능 제공

END