

ML&DL

WITH PYTHON

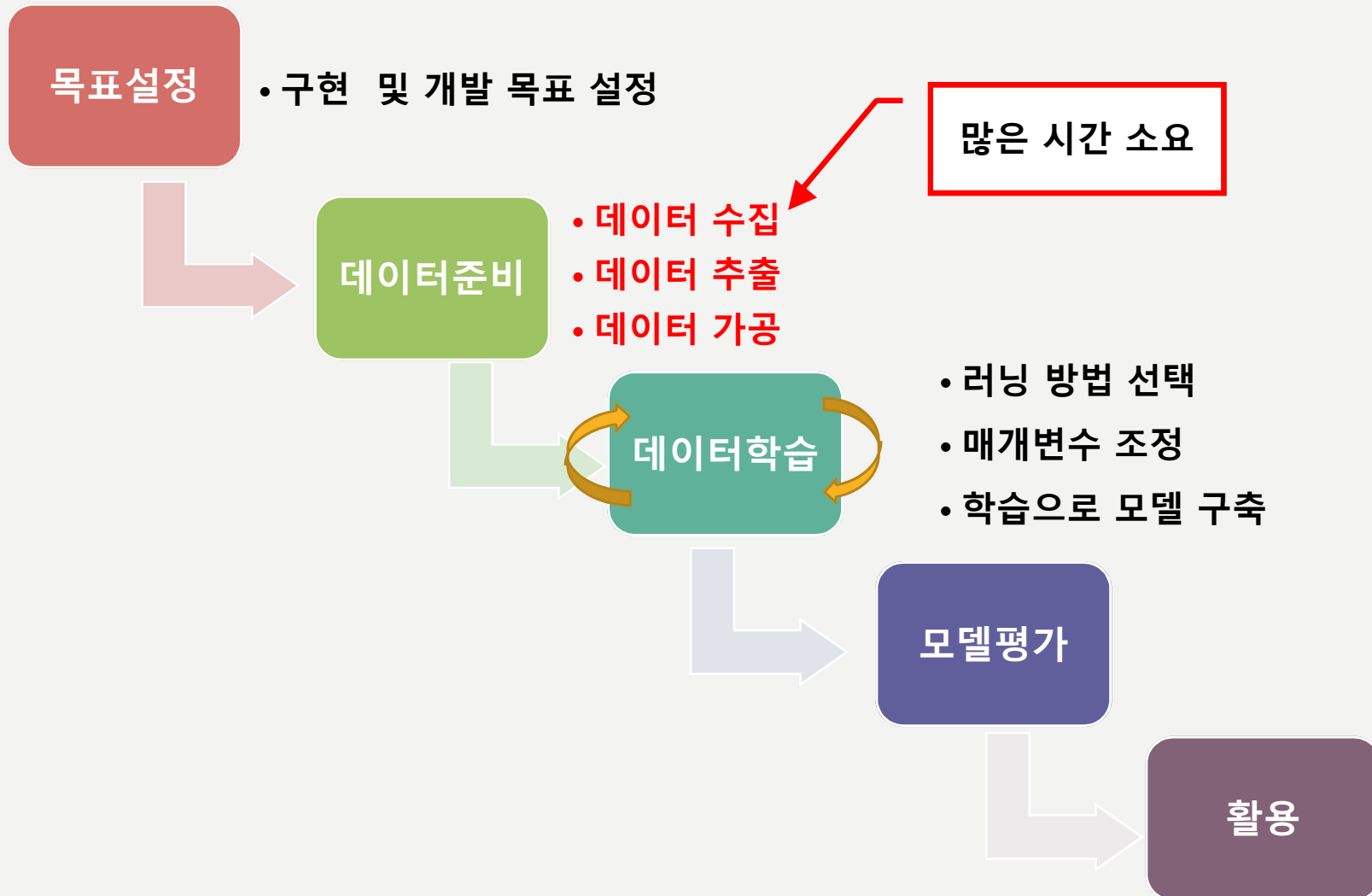
PART II

DATA 수집 및 가공

CH01. DATA 수집 및 가공

DATA 수집 및 가공

◆ 러닝 과정



DATA 수집 및 가공

◆ 데이터 소스

- 검색 엔진 데이터
- 웹 사이트 데이터
- 행정 공개 데이터
- 공개 러닝 데이터
- 공개 이미지 데이터

DATA 수집 및 가공

◆ 데이터 추출

데이터 소스	추출 데이터
SNS, BLOG	트렌드 정보 수집
아마존, 알리바바, 네이버쇼핑	인터넷 쇼핑 상품 데이터
금융정보	환율, 주식, 경제 흐름
오픈 데이터	기상 정보, 소비 통계, 지역별 출산율...
미디어 공유 사이트	이미지, 동영상, 음성 데이터
위키피디아	사진 데이터

DATA 수집 및 가공

◆ 데이터 가공

- 수집된 데이터 사용 가능 형태로 저장

```
<div id="whale_promotion_banner" class="banner_area">
  <div class="area_flex">
    <a href="https://blog.naver.com/whaleteam/22
    
      <p class="ba_t"><em class="ba_em">현
      <button id="whale_promotion_download
      <button id="whale_promotion_close_bt
      class="ico_btn_close"></i><span class="blind">닫기</s
    </div>
  </a>
</div>
</div>
</div>
<a id="whale_promotion_download_file" style="display
href="http://update.whale.naver.net/downloads/in
```

HTML

데이터
추출

데이터 파일 형식

CSV 파일 형식

INI 파일 형식

XML 파일 형식

JSON 파일 형식

YAML 파일 형식

데이터베이스 저장

MySQL / Maria

Oracle

PostgreSQL

SQLite

SQLServer

CH02. WEB 데이터 수집

WEB DATA 수집

◆ 데이터 수집 방법

➤ WEB 데이터 수집 방법

- 크롤링 (Crawling)

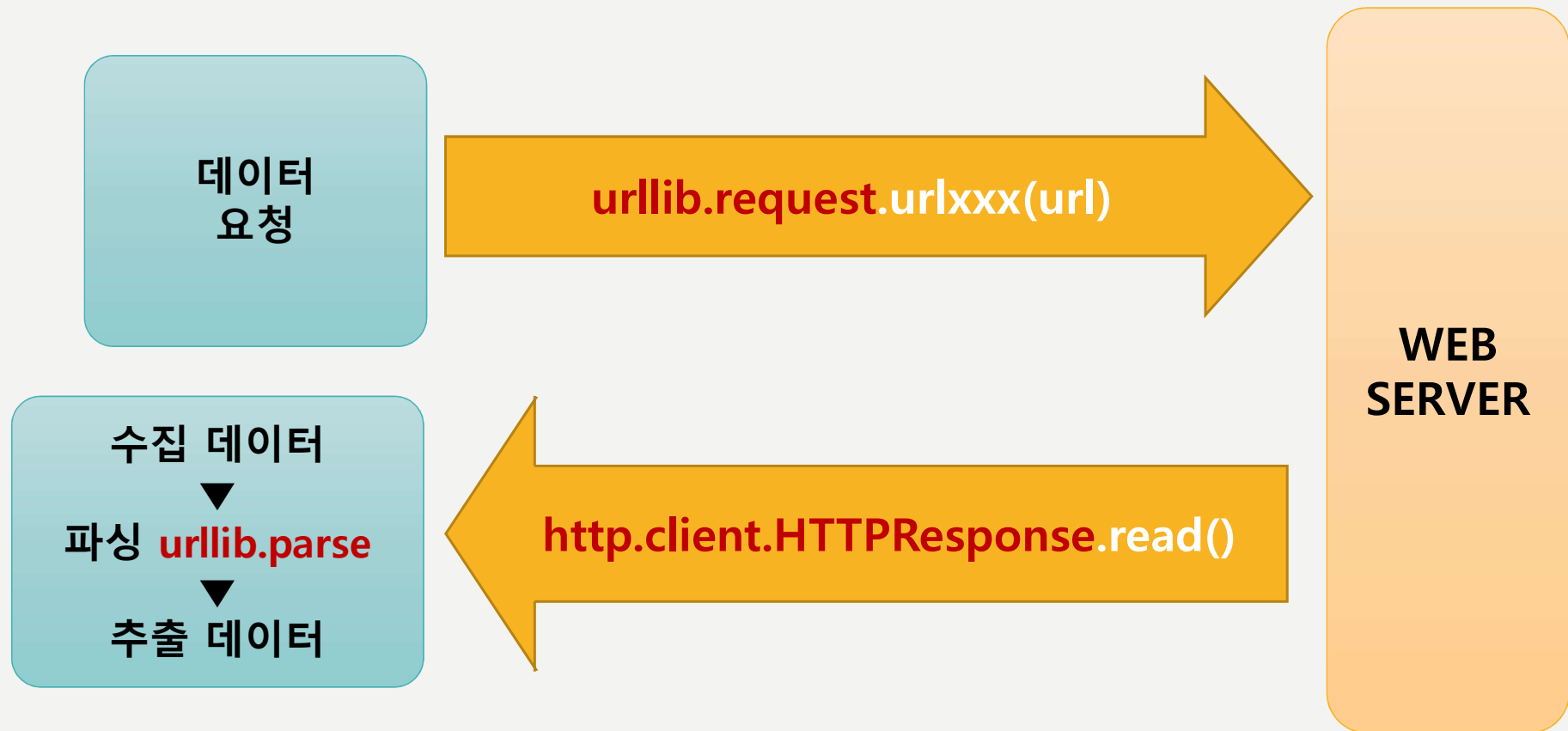
- ✓ WEB 페이지의 하이퍼링크를 정기적인 순회하며 다운로드
- ✓ 최신 정보 유지
- ✓ 크롤링 SW를 크롤러(Crawler), 스파이더(Spider)라 함

- 스크레이핑 (Scraping)

- ✓ WEB 페이지의 특정 정보 추출 및 구조 분석 기술
- ✓ 스크레이핑 SW를 스크레이퍼(Scraper)

WEB DATA 수집

◆ 데이터 수집 과정



WEB DATA 수집

◆ 데이터 다운로드 라이브러리

- 데이터 요청 라이브러리

[urllib](#) — URL handling modules ¶

Source code: [Lib/urllib/](#)

`urllib` is a package that collects several modules for working with URLs:

- `urllib.request` for opening and reading URLs
- `urllib.error` containing the exceptions raised by `urllib.request`
- `urllib.parse` for parsing URLs
- `urllib.robotparser` for parsing `robots.txt` files

- <https://docs.python.org/3/library/urllib.html>

WEB DATA 수집

◆ 데이터 다운로드 라이브러리

- 응답 처리 라이브러리

`http.client` — HTTP protocol client

Source code: [Lib/http/client.py](#)

This module defines classes which implement the client side of the HTTP and HTTPS protocols. It is normally not used directly — the module `urllib.request` uses it to handle URLs that use HTTP and HTTPS.

▪ HTTPConnection Objects ▪ HTTPResponse Objects ▪ HTTPMessage Objects

- <https://docs.python.org/3/library/http.client.html#httpresponse-objects>

WEB DATA 수집

◆ 데이터 다운로드 라이브러리

```
# File 읽기 & 출력 -----  
  
try:  
    # 읽기모드, utf-8 인코딩으로 파일 open & read  
    # 파일 객체 리턴  
    file = open( ' test.html', mode='r', encoding='utf-8')  
  
    # 1라인씩 읽어서 출력  
    while True:  
        line = file.readline()      #줄바꿈 단위로 읽기  
        if not line: break  
        print(line)  
  
finally:  
    file.close()
```

WEB DATA 수집

◆ 데이터 다운로드 라이브러리

```
# File 쓰기 -----  
  
try:  
    # 쓰기모드로 파일 open & write  
    # 파일이 존재하지 않을 경우 생성 & 덮어쓰기  
    file = open('data.txt', mode= 'w', encoding='utf-8')  
  
    # 1~5까지 파일에 쓰기  
    for i in range(1, 6):  
        data = '%d번째 줄\n' % i  
        file.write(data)  
  
finally:  
    file.close()
```

WEB DATA 수집

◆ 데이터 다운로드 라이브러리

with ~ as 구문

- python2.5 도입
- 파일 입출력을 간단하게 하는 구문
- 파일 open 후 close 및 exception 처리 생략

쓰기모드로 파일 open & write

```
with open('data.txt', mode= 'w', encoding='utf-8') as file :
```

1~5까지 파일에 쓰기

```
for i in range(1, 6):
```

```
    data = '%d번째 줄\n' % i
```

```
    file.write(data)
```

WEB DATA 수집

◆ 데이터 다운로드 라이브러리

```
# 이미지 파일 다운로드하기 -----  
import urllib.request  
  
# URL과 저장 경로 지정  
url = "http://uta.pw/shodou/img/28/214.png"  
savename = "test.png"  
  
# 파일 형태로 다운로드  
urllib.request.urlretrieve( url, savename )  
print("저장되었습니다...!")
```


WEB DATA 수집

◆ 데이터 다운로드 라이브러리

```
# 이미지 파일 다운로드하기 -----  
import urllib.request  
  
# URL과 저장 경로 지정하기  
url = "http://uta.pw/shodou/img/28/214.png"  
savename = "test.png"  
  
# 다운로드  
mem = urllib.request.urlopen(url).read()  
  
# 바이너리 모드로 파일 open & write  
with open(savename, mode="wb") as f:  
    f.write(mem)  
    print("저장되었습니다...!")
```

WEB DATA 수집

◆ 데이터 다운로드 라이브러리

```
# 텍스트기반 파일 다운로드하기 -----  
# 라이브러리 읽어들이기  
import urllib.request  
  
# 데이터 읽어 들이기  
url = 'http://api.aoikujira.com/ip/ini'  
res = urllib.request.urlopen(url)  
data = res.read() # 바이트 데이터 읽어오기  
print('type(data) =', type(data)) # 타입 체크  
  
# 바이너리를 문자열로 변환하기  
text = data.decode("utf-8")  
print(text)
```

WEB DATA 수집

◆ 데이터 다운로드 라이브러리

텍스트기반 파일 다운로드하기 -----

```
import urllib.request
import urllib.parse
```

API = <http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp>

매개변수 URL 인코딩
values = { 'stnId': '108' }

지역번호

params = urllib.parse.urlencode(values) # 키=값 형태로 인코딩

요청 전용 URL을 생성
url = API + "?" + params
print("url=", url)

다운로드
data = urllib.request.urlopen(url).read()
text = data.decode("utf-8")
print(text)

<http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108>

1개 이상이면
[?stnId=103&name=seoul](http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=103&name=seoul)

WEB DATA 수집

◆ 데이터 다운로드 라이브러리

```
# 명령줄 지정 텍스트기반 파일 다운로드하기 -----  
import sys  
import urllib.request as req  
import urllib.parse as parse  
  
# 명령줄 매개변수 추출  
if len(sys.argv) <= 1:  
    print("USAGE: download-forecast-argv <Region Number>")  
    sys.exit()  
regionNumber = sys.argv[1]  
  
# 매개변수를 URL 인코딩  
API = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"  
values = { 'stnId': regionNumber }  
params = parse.urlencode(values)  
url = API + "?" + params  
print("url=", url)
```

WEB DATA 수집

◆ 데이터 다운로드 라이브러리

```
# 다운로드
#res = req.urlopen(url)
#data = res.read()
data = req.urlopen(url).read()
text = data.decode("utf-8")

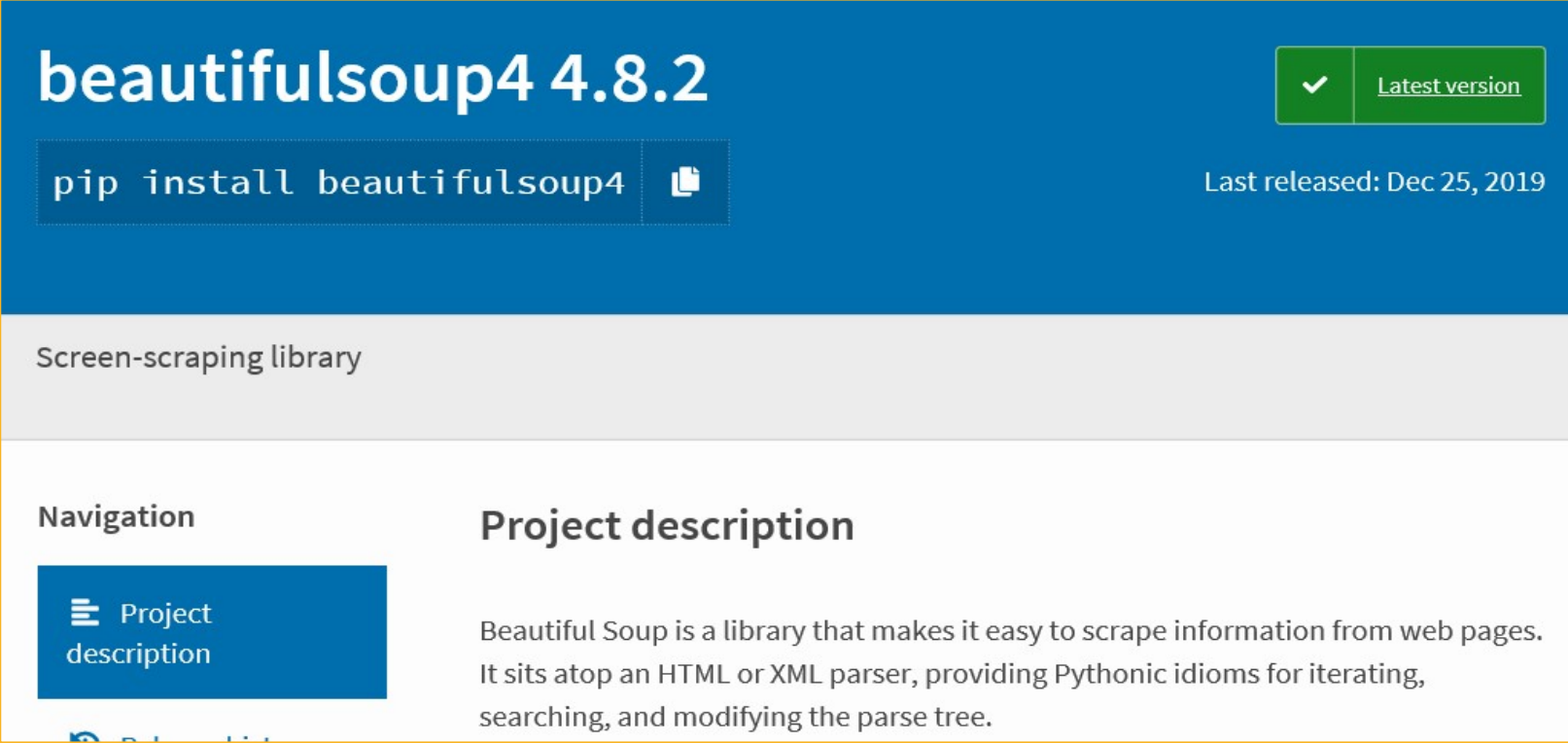
print(text)
```

CH03. BeautifulSoup

BEAUTIFULSOUP

◆ BeautifulSoup 라이브러리

- 스크레이핑 라이브러리



The screenshot shows the PyPI page for BeautifulSoup4 4.8.2. The header is blue with the text 'beautifulsoup4 4.8.2' in white. To the right, there is a green checkmark icon and a green button labeled 'Latest version'. Below the header, there is a dark blue box with the text 'pip install beautifulsoup4' and a copy icon. To the right of this box, it says 'Last released: Dec 25, 2019'. Below the header, there is a light gray box with the text 'Screen-scraping library'. Below this, there is a white box with a 'Navigation' section on the left and a 'Project description' section on the right. The 'Navigation' section has a blue button labeled 'Project description'. The 'Project description' section has the text: 'Beautiful Soup is a library that makes it easy to scrape information from web pages. It sits atop an HTML or XML parser, providing Pythonic idioms for iterating, searching, and modifying the parse tree.'

beautifulsoup4 4.8.2

✓ Latest version

pip install beautifulsoup4

Last released: Dec 25, 2019

Screen-scraping library

Navigation

Project description

Project description

Beautiful Soup is a library that makes it easy to scrape information from web pages. It sits atop an HTML or XML parser, providing Pythonic idioms for iterating, searching, and modifying the parse tree.

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

BEAUTIFULSOUP

◆ BeautifulSoup 라이브러리

[[Download](#) | [Documentation](#) | [Hall of Fame](#) | [For enterprise](#) | [Source](#) | [Changelog](#) | [Discussion group](#) | [Zine](#)]

Beautiful Soup

You didn't write that awful page. You're just trying to get some data out of it. BeautifulSoup is here to help. Since 2004, it's been saving programmers hours or days of work on quick-turnaround screen scraping projects.

Beautiful Soup is a Python library designed for quick turnaround projects like screen-scraping. Three features make it powerful:

1. BeautifulSoup provides a few simple methods and Pythonic idioms for navigating, searching, and modifying a parse tree: a toolkit for dissecting a document and extracting what you need. It doesn't take much code to write an application
2. BeautifulSoup automatically converts incoming documents to Unicode and outgoing documents to UTF-8. You don't have to think about encodings, unless the document doesn't specify an encoding and BeautifulSoup can't detect one. Then you just have to specify the original encoding.
3. BeautifulSoup sits on top of popular Python parsers like [lxml](#) and [html5lib](#), allowing you to try out different parsing strategies or trade speed for flexibility.



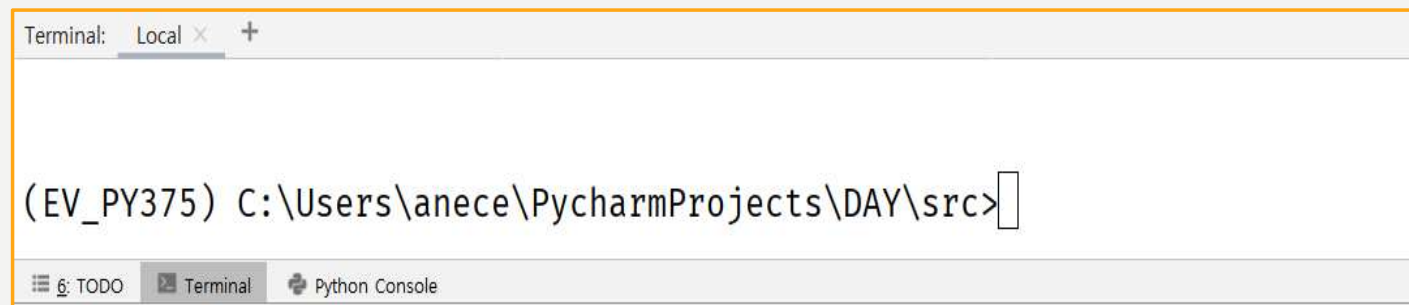
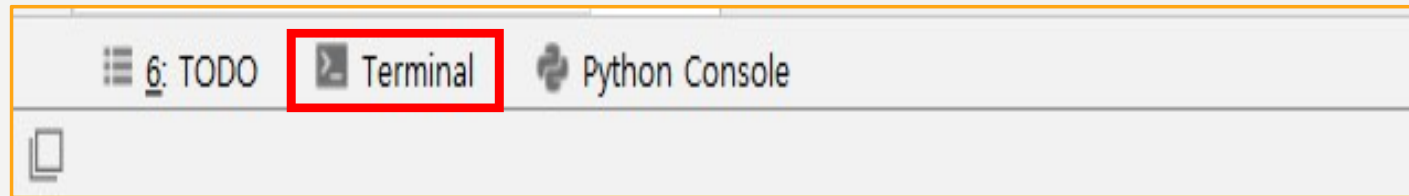
- <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

BEAUTIFULSOUP

◆ BeautifulSoup 설치

➤ 설치 방법 1

- Pycharm 창 아래 > Terminal 선택



BEAUTIFULSOUP

◆ BeautifulSoup 설치

➤ 설치 방법 1

- 라이브러리 존재 여부 체크

`conda list | grep 라이브러리명`

```
Terminal: Local x +  
  
(EV_PY375) C:\Users\anece\PycharmProjects\DAY\src>conda list | grep numpy  
numpy                1.17.4                py37h4320e6b_0  
numpy-base           1.17.4                py37hc3f5095_0  
numpydoc              0.9.1                  py_0  
  
(EV_PY375) C:\Users\anece\PycharmProjects\DAY\src>conda list | grep beautiful  
  
(EV_PY375) C:\Users\anece\PycharmProjects\DAY\src>
```

BEAUTIFULSOUP

◆ BeautifulSoup 설치

➤ 설치 방법 1

- 라이브러리 설치

```
conda install beautifulSoup4
```

```
-----|-----
beautifulsoup4-4.8.2      | py37_0      163 KB
-----|-----
                                Total:      163 KB
```

The following NEW packages will be INSTALLED:

```
beautifulsoup4      pkgs/main/win-64::beautifulsoup4-4.8.2-py37_0
soupsieve            pkgs/main/win-64::soupsieve-1.9.5-py37_0
```

Proceed ([y]/n)? █

BEAUTIFULSOUP

◆ BeautifulSoup 설치

➤ 설치 방법 1

- 라이브러리 설치 확인

```
Downloading and Extracting Packages
```

```
beautifulsoup4-4.8.2 | 163 KB | ##### | 100%
```

```
Preparing transaction: done
```

```
Verifying transaction: done
```

```
Executing transaction: done
```

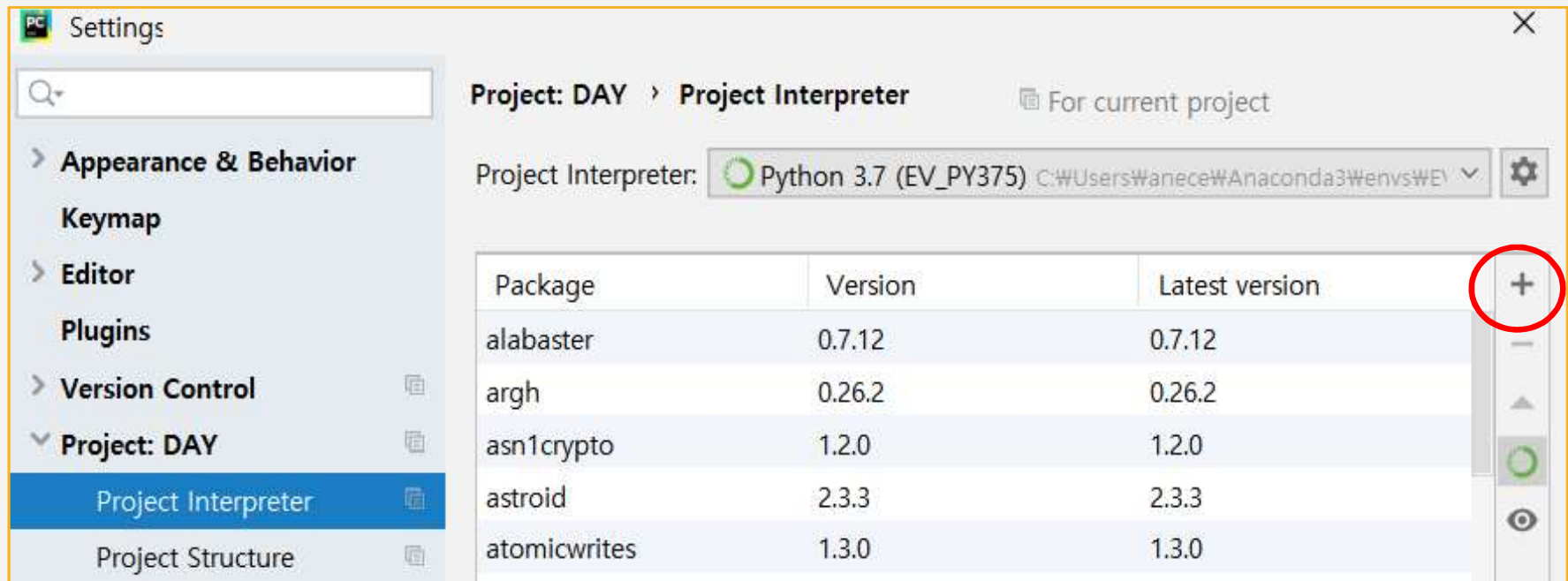
```
(EV_PY375) C:\Users\anece\PycharmProjects\DAY>conda list | grep beautiful  
beautifulsoup4          4.8.2                py37_0
```

BEAUTIFULSOUP

◆ BeautifulSoup 설치

➤ 설치 방법 2

- PyCham > File > Settings... > Project > Project Interpreter
> 오른쪽 + 클릭

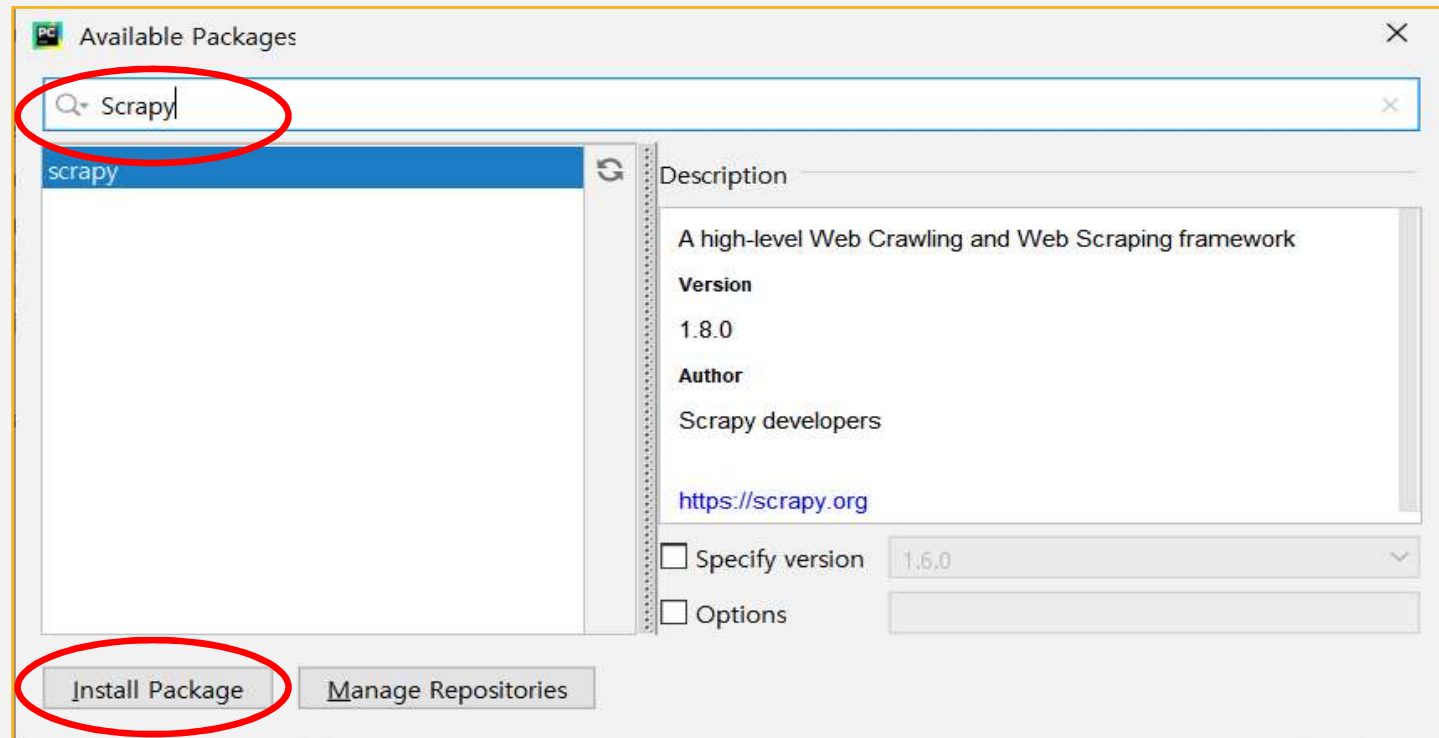


BEAUTIFULSOUP

◆ BeautifulSoup 설치

➤ 설치 방법 2

- Available Packages > 설치 라이브러리명 입력
> 왼쪽 아래 Install Package 클릭

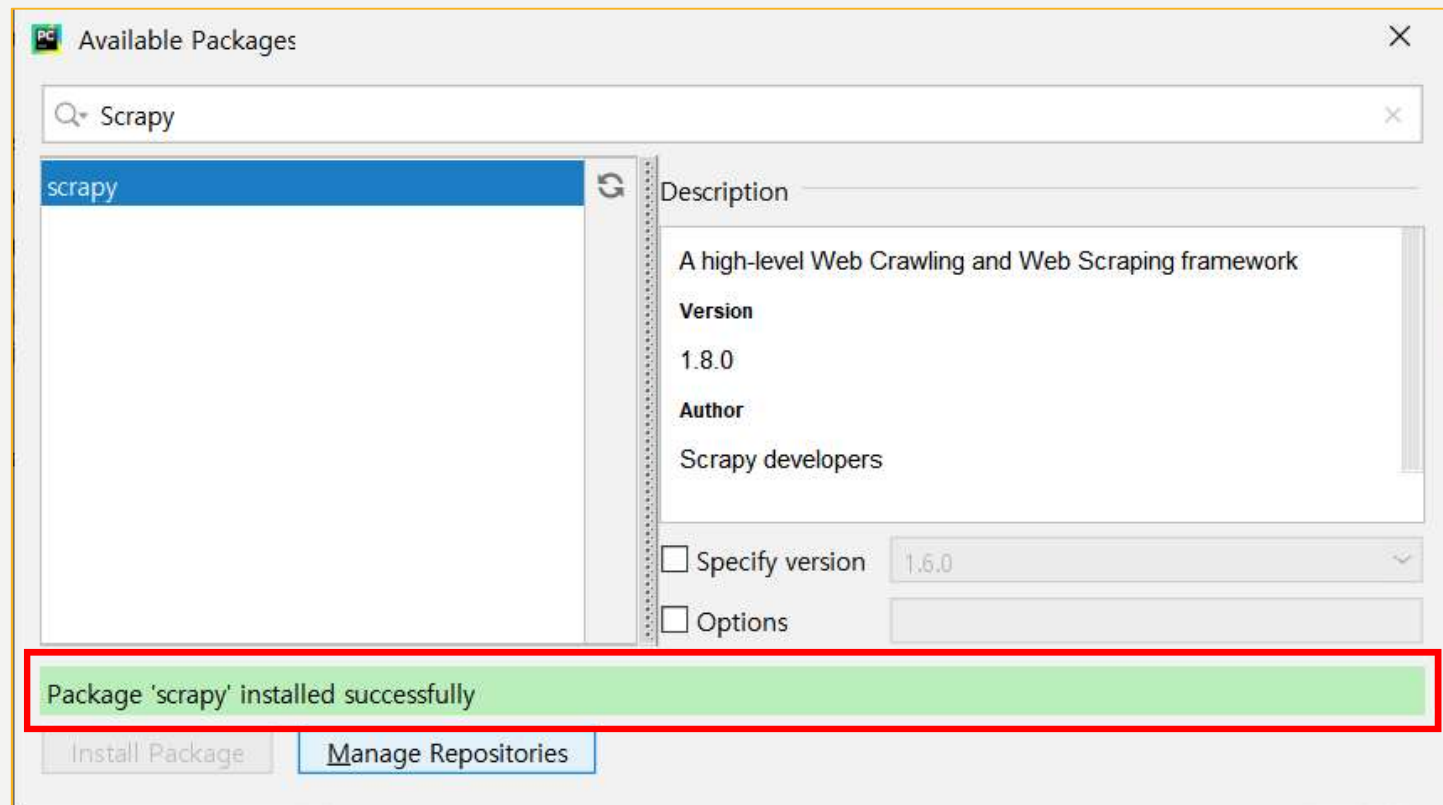


BEAUTIFULSOUP

◆ BeautifulSoup 설치

➤ 설치 방법 2

- 설치 완료

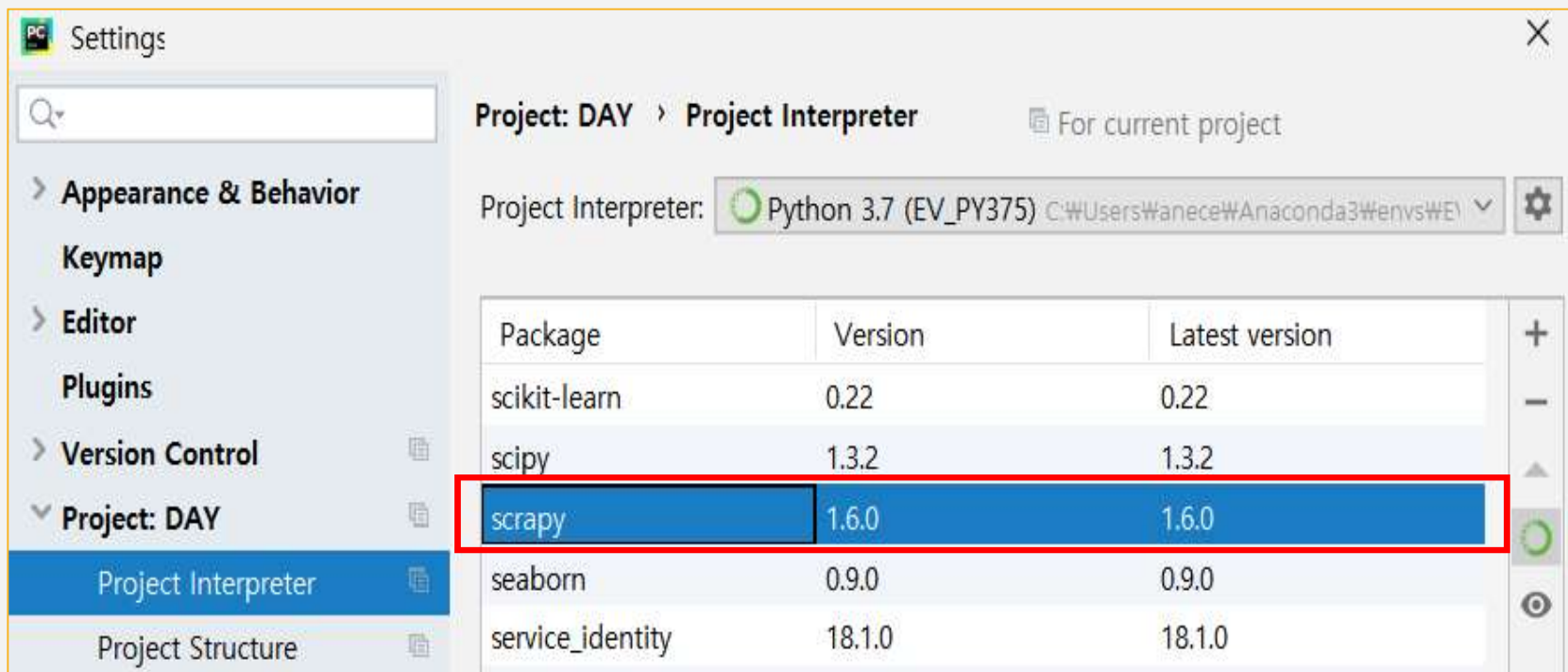


BEAUTIFULSOUP

◆ BeautifulSoup 설치

➤ 설치 방법 2

- 설치 완료



BEAUTIFULSOUP

◆ BeautifulSoup 사용법

① BeautifulSoup 분석 객체 생성

⇒ BeautifulSoup(마크업문자열 / HTTPResponse OBJ, 파서)
⇒ BeautifulSoup.prettify()

② Web 페이지 원하는 부분 추출

⇒ find('요소명', id='id명', class='class명')
⇒ find(name, attrs, recursive, string, **kwargs)
⇒ find_all('요소명', id='id명', class='class명')
⇒ find_all(name, attrs, recursive, string, limit, **kwargs)

③ 추출된 데이터 출력 및 활용

=> DOM 요소의 각 속성에 따른 검출

BEAUTIFULSOUP

◆ Markup Parser

- HTML, XML 등 마크업언어를 분석하는 것

파서명	사용 설정	장점	단점
html.parser	BeautifulSoup(markup, 'html.parser')	바로 사용 가능	
lxml HTML parser	BeautifulSoup(markup, 'lxml')	매우빠른	lxml 추가 설치
lxml XML parser	BeautifulSoup(markup, 'lxml-xml') BeautifulSoup(markup, 'xml')	매우빠름	lxml 추가 설치
html5lib	BeautifulSoup(markup, 'html5lib')	웹브라우저방식 파싱 유효한 HTML5 생서	html5lib 추가 설치 매우 느림

BEAUTIFULSOUP

◆ HTML 구조

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE> New Document </TITLE>
    <META NAME="Generator" CONTENT="EditPlus">
    <META NAME="Author" CONTENT="">
    <META NAME="Keywords" CONTENT="">
    <META NAME="Description" CONTENT="">
  </HEAD>

  <BODY bgcolor='yellow'>
    <h1> HEADER </h1>
    <a href="http://www.naver.com">GO~! NAVER</a>

    <div>
      SPACE 1
    </div>

    <div>
      SPACE 2
    </div>
  </BODY>
</HTML>
```

태그(Tag)

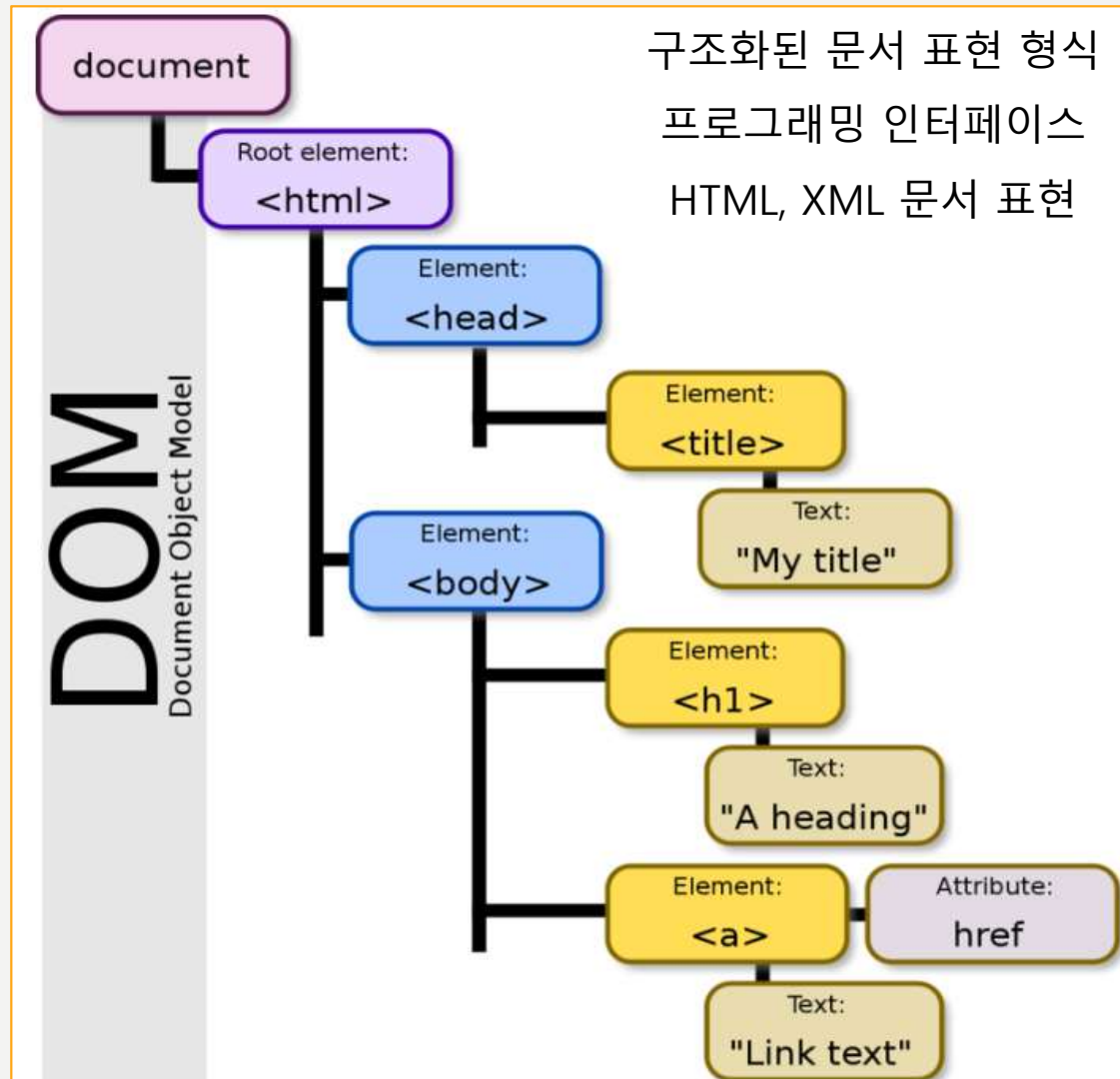
- HTML 구성 요소
- 사용자에게 보여줄 여러가지 요소
- 형식 : < 태그명 속성> </태그명>
- 역할에 따라 다양한 속성 존재

BEAUTIFULSOUP

◆ DOM 구조

출처 : 위치백과

구조화된 문서 표현 형식
프로그래밍 인터페이스
HTML, XML 문서 표현



BEAUTIFULSOUP

◆ 요소 검출 - html 태그명

```
# HTML 파일 분석 및 출력 01 -----
from bs4 import BeautifulSoup

# 분석하고 싶은 HTML
html = """
<html>
  <body>
    <h1>스크레이핑이란?</h1>
    <p>웹 페이지를 분석하는 것</p>
    <p>원하는 부분을 추출하는 것</p>
  </body>
</html>
"""

soup = BeautifulSoup(html, 'html.parser')      # HTML 분석 객체 생성

# 원하는 부분 추출하기
h1 = soup.html.body.h1                        # HTML 태그 추출
p1 = soup.html.body.p
p2 = p1.next_sibling.next_sibling
```

BEAUTIFULSOUP

◆ 요소 검출 - HTML 태그명

```
# HTML 파일 분석 및 출력 01-----
```

```
# 요소의 글자 출력하기
```

```
print("h1 = " + h1.string)
```

```
print("p = " + p1.string)
```

```
print("p = " + p2.string)
```

BEAUTIFULSOUP

◆ 요소 검출 - ID속성 활용

웹 페이지에서 HTML 요소 찾기

```
find('요소명', id='id명', class='class명')
```

```
⇒ find(name, attrs, recursive, string, **kwargs)
```

```
find_all ('요소명', id='id명', class='class명')
```

```
⇒ find_all(name, attrs, recursive, string, limit, **kwargs)
```

***단! HTML 요소에 존재하는 속성 범위**

BEAUTIFULSOUP

◆ 요소 검출 - ID속성 활용 find()

```
# HTML 파일 분석 및 출력 02-----
from bs4 import BeautifulSoup

html = """
<html>
  <body>
    <h1 id="title">스크레이핑이란?</h1>
    <p id="body">웹 페이지를 분석하는 것</p>
    <p>원하는 부분을 추출하는 것</p>
  </body>
</html>
"""

# HTML 분석기 객체 생성
soup = BeautifulSoup(html, 'html.parser')

# find() 메서드 - 원하는 부분 추출
title = soup.find(id="title")
body = soup.find(id="body")
```


BEAUTIFULSOUP

◆ 요소 검출 - ID속성 활용 find()

HTML 파일 분석 및 출력 02 -----

텍스트 부분 출력하기

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

```
print("#body=" + body.string)
```

BEAUTIFULSOUP

◆ 요소 검출 – ID속성 활용 find_all()

```
# HTML 파일 분석 및 출력 03 -----
from bs4 import BeautifulSoup
html = """
<html> <body>
  <ul>
    <li> <a href="http://www.naver.com">naver</a> </li>
    <li> <a href="http://www.daum.net">daum</a> </li>
  </ul>
</body> </html>
"""

# HTML 분석
soup = BeautifulSoup(html, 'html.parser')
links = soup.find_all("a") # find_all() 여러 개 요소 검출

# 링크 목록 출력
for a in links:
    href = a.attrs['href']
    text = a.string
    print(text, ">", href)
```

BEAUTIFULSOUP

◆ 요소 검출 - ID속성 활용

```
# HTML 파일 다운 & 분석 & 추출 -----
from bs4 import BeautifulSoup
import urllib.request as req

# urlopen()으로 데이터 수집
url = "http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp"
res = req.urlopen(url)

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

# 원하는 데이터 추출
title = soup.find("title").string
wf = soup.find("wf").string
print(title)
print(wf)
```

BEAUTIFULSOUP

◆ 요소 검출 – CSS 선택자

```
from bs4 import BeautifulSoup

fp = open("books.html", encoding="utf-8")
soup = BeautifulSoup(fp, "html.parser")

# CSS 선택자로 검색하는 방법
sel = lambda q : print(soup.select_one(q).string)
sel("#nu")
sel("li#nu")
sel("ul > li#nu")
sel("#bible #nu")
sel("#bible > #nu")
sel("ul#bible > li#nu")
sel("li[id='nu']")
sel("li:nth-of-type(4)")

# 그 밖의 방법
print(soup.select("li")[3].string)
print(soup.find_all("li")[3].string)
```

BEAUTIFULSOUP

◆ 요소 검출 – CSS 선택자 02

```
from bs4 import BeautifulSoup

fp = open("fruits-vegetables.html", encoding="utf-8")
soup = BeautifulSoup(fp, "html.parser")

# CSS 선택자로 추출하기
# print(soup.select_one("li:nth-of-type(8)").string)      # bs4.5.1 이후 미지원
print(soup.select_one("#ve-list > li:nth-of-type(4)").string)
print(soup.select("#ve-list > li[data-lo='us']")[1].string)
print(soup.select("#ve-list > li.black")[1].string)

# 요소명 & 조건으로 검출
cond = {"data-lo": "us", "class": "black"}
print(soup.find("li", cond).string)

# find 메서드를 연속적으로 사용
print(soup.find(id="ve-list").find("li", cond).string)
```

BEAUTIFULSOUP

◆ re 모듈

- 파이썬 기본 제공 regular expression 모듈

함수명	기능
compiler(패턴)	패턴 컴파일 / 패턴 객체 리턴
match(패턴, 확인 문자열)	확인 문자열이 전부 패턴에 일치하는 지 여부, match 객체 리턴
search(패턴, 확인 문자열)	확인 문자열에 패턴이 존재하는지 여부, match 객체 리턴
findall(확인 문자열)	패턴과 일치되는 문자열 리스트 리턴
finditer(확인 문자열)	정규식과 매치되는 모든 문자열을 반복 가능한 객체로 돌려준다.

BEAUTIFULSOUP

◆ re 모듈

- 자주 사용되는 표현식

- `\d` - 숫자와 매치, `[0-9]`와 동일한 표현식이다.
- `\D` - 숫자가 아닌 것과 매치, `[^0-9]`와 동일한 표현식이다.
- `\s` - whitespace 문자와 매치, `[\t\n\r\f\v]`와 동일한 표현식이다. 맨 앞의 빈 칸은 공백문자(space)를 의미한다.
- `\S` - whitespace 문자가 아닌 것과 매치, `[^\t\n\r\f\v]`와 동일한 표현식이다.
- `\w` - 문자+숫자(alphanumeric)와 매치, `[a-zA-Z0-9_]`와 동일한 표현식이다.
- `\W` - 문자+숫자(alphanumeric)가 아닌 문자와 매치, `[^a-zA-Z0-9_]`와 동일한 표현식이다.

BEAUTIFULSOUP

◆ 요소 검출 – CSS & 정규식

```
from bs4 import BeautifulSoup
import re # 파이썬 regular expression 모듈

html = """
<ul>
  <li> <a href="hoge.html">hoge</li>
  <li> <a href="https://example.com/fuga">fuga*</li>
  <li> <a href="https://example.com/foo">foo*</li>
  <li> <a href="http://example.com/aaa">aaa</li>
</ul>
"""

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

# 정규 표현식으로 href에서 https인 것 추출
li = soup.find_all( href=re.compile( r"^https://" ) ) # 문자열 그대로 패턴

for e in li: print( e.attrs['href'] )
```


BEAUTIFULSOUP

◆ 요소 검출 - 다운 & 분석

HTML 파일 분석 및 출력 : 네이버 금융 -----

```
from bs4 import BeautifulSoup
import urllib.request as req
```

HTML 가져오기

```
url = "https://finance.naver.com/marketindex/?tabSel=exchange#tab_section"
res = req.urlopen(url)
```

HTML 분석하기

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

원하는 데이터 추출

```
price =
soup.select_one("ul#exchangeList>li.on>a.head>div.head_info>span.value").string
print("usd/krw =", price)
```

BEAUTIFULSOUP

◆ BeautifulSoup – 데이터 경로

웹 상 다운로드 데이터 경로 변환

```
urllib.request.urljoin( base url, 상대path )
```

BEAUTIFULSOUP

◆ BeautifulSoup – 데이터 경로 예제

데이터 절대경로 설정 -----

```
from urllib.parse import urljoin
```

```
base = "http://example.com/html/a.html"
```

<pre>print(urljoin(base, "b.html"))</pre>	# 현재 경로 즉, html 아래 존재
<pre>print(urljoin(base, "sub/c.html"))</pre>	# 현재 경로 즉, html 아래 존재
<pre>print(urljoin(base, "../index.html"))</pre>	# .. => 한 단계 위 의미
<pre>print(urljoin(base, "../img/hoge.png"))</pre>	# 한 단계 위 즉, example.com
<pre>print(urljoin(base, "../css/hoge.css"))</pre>	# 한 단계 위 즉, example.com

BEAUTIFULSOUP

◆ BeautifulSoup – 데이터 경로 예제

```
# 데이터 절대경로 설정 -----  
from urllib.parse import urljoin  
  
base = "http://example.com/html/a.html"  
  
print( urljoin(base, "/hoge.html") )           # 제일 상위 경로 아래  
print( urljoin(base, http://otherExample.com/wiki) ) # 절대 경로  
print( urljoin(base, "//anotherExample.org/test") ) # 절대 경로
```

BEAUTIFULSOUP

◆ BeautifulSoup – url 파싱

`urllib.parse.urlparse(url)`

- url 구성 요소 구문 분석, ParseResult 객체 리턴

- <https://netloc/path;params?query=value#fragment>

Attribute	Index	Value	Value if not present
scheme	0	URL scheme specifier	<i>scheme</i> parameter
net loc	1	Network location part	empty string
path	2	Hierarchical path	empty string
params	3	Parameters for last path element	empty string
query	4	Query component	empty string
fragment	5	Fragment identifier	empty string
username		User name	None
password		Password	None
hostname		Host name (lower case)	None
port		Port number as integer, if present	None