인터넷 응용

- 인터넷 응용
  - 웹브라우저를 사용하여 우리는 매일 필요한 정보를 인터넷에서 접하고 이를 사용하고 있다.
  - 이러한 정보는 그 양이 대단히 많을 수 있고, 시간이 경과함에 따라 다이나믹하게 변할 수 있다.
  - 따라서, 웹브라우저만으로 원하는 정보를 추출하는 것은 오랜 검색 시간이 필요하는 등 여러 한계가 있을 수 있다.
  - 그러나 컴퓨터 프로그램을 사용할 경우, 이러한 정보를 인터넷에서 손쉽게 자동으로 얻을 수 있다.
  - 본 강의에서는 Python 프로그래밍을 통하여 인터넷 정보를 얻을 수 있는 기초적인 방법을 알아보고, 간단한 프로그램을 작성하여 본다.

Example

- 아래 보인 그림은 다이닝 코드라는 맛집 추천 서비스를 제공하는 사이트의 처음 화면의 일부이다.
- 이 사이트에서는 국내 여러 블로그에서 해당 음식점이 얼마나 등장했고, 해당 리뷰에서 어떤 키워드가 자주 사용되었는지를 추출하여 분석한 정보를 제공한다(*).
  http://www.diningcode.com/

(*) 여기서, 웹을 체계적으로 검색하여 원하는 정보만을 추출하는 행위를 웹 크롤링(web crawling)이라고 한다.

HTML(HyperText Markup Language)

- 확장자는 html, htm이며 인터넷 익스플로러와 같은 웹 브라우저에서 읽을 수 있는 웹 문서를 만들기 위한 언어이다.
- HTML의 특성
  - 하이퍼텍스트(Hypertext) : 참조(하이퍼링크)를 통해 독자가 한 문서에서 다른 문서로 즉시 접근할 수 있는 텍스트.
  - 마크업 언어(Markup Language) : 문서의 내용 뿐만 아니라, 태그 등을 이용하여 글자 크기, 모양 등 문서의 출력 형태까지 명기하는 언어의 한 가지이다.
  - HTML에서는 태그(tag)라고 하는 심볼을 사용하여 문서의 구조, 출력 형태 등을 조정한다.
  - 태그는 < >를 사용하여 표시하며, <태그명>으로 시작하고 </태그명>으로 끝맺는데(*), 이 사이에 내용을 넣는다.
  - 태그에는 태그의 성격을 구체화 하는 속성(attribute)를 부여할 수 있다.

(*) 종료태그라고 하며, 일부 태그는 종료태그가 없을 수도 있다.

✦HTML 문서의 예

head : 브라우저 정보
title : 툴바의 제목
a (anchor) : 웹문서(or 문서내 다른 곳)를 연결
style="color:red;" :
속성 속성값 (*)

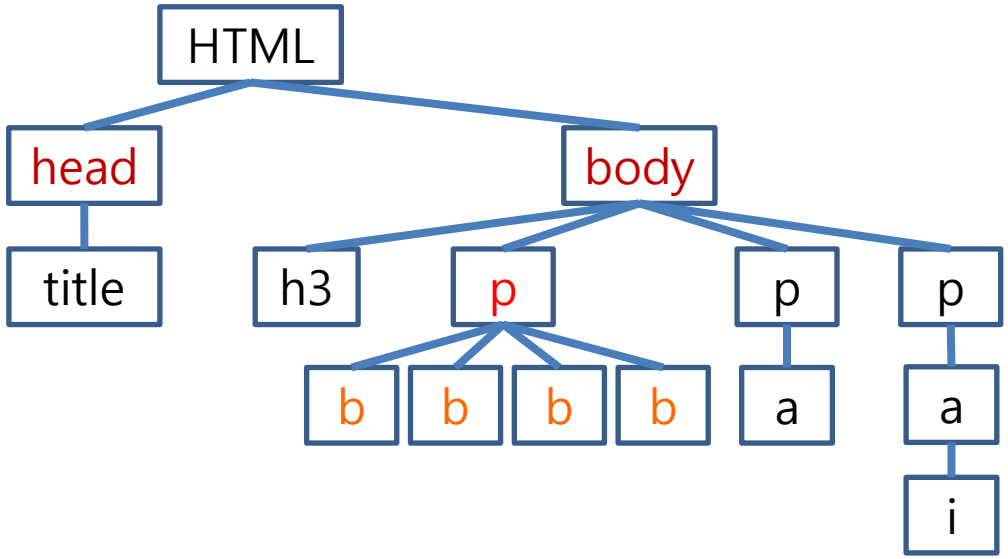
```
<!doctype html>
<head><title>An Example</title></head>
<body>
  <h3>The second heading(small font).</h3>
  <p>A paragraph(number list):
    <b style="color:red;">123</b>, <b style="color:red;">356</b>,
    <b>641</b>, <b>387</b> </p>
  <p>Hyperlink: <a href="http://www.google.com">google</a></p>
  <p>Link with italic: <a href="http://www.naver.com"><i>naver</i></a></p>
</body>
```

위 문서가 저장된 파일을 웹브라우저에서 읽으면 다음과 같이 보인다.

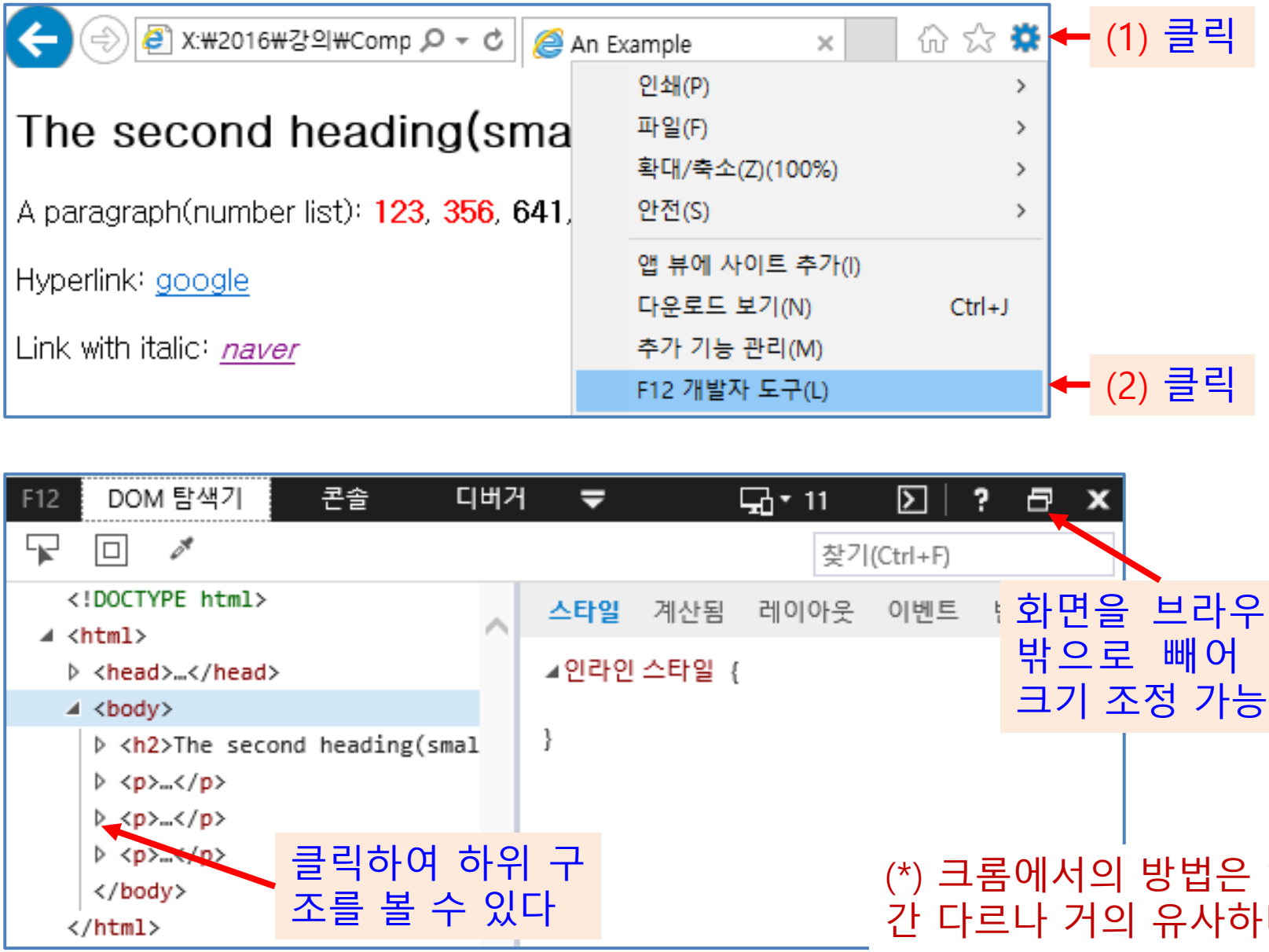
(*) 본 강의에서 태그와 속성의 의미를 알 필요는 없다

✦HTML 문서는 태그를 노드로 하는 트리 구조를 갖는다.

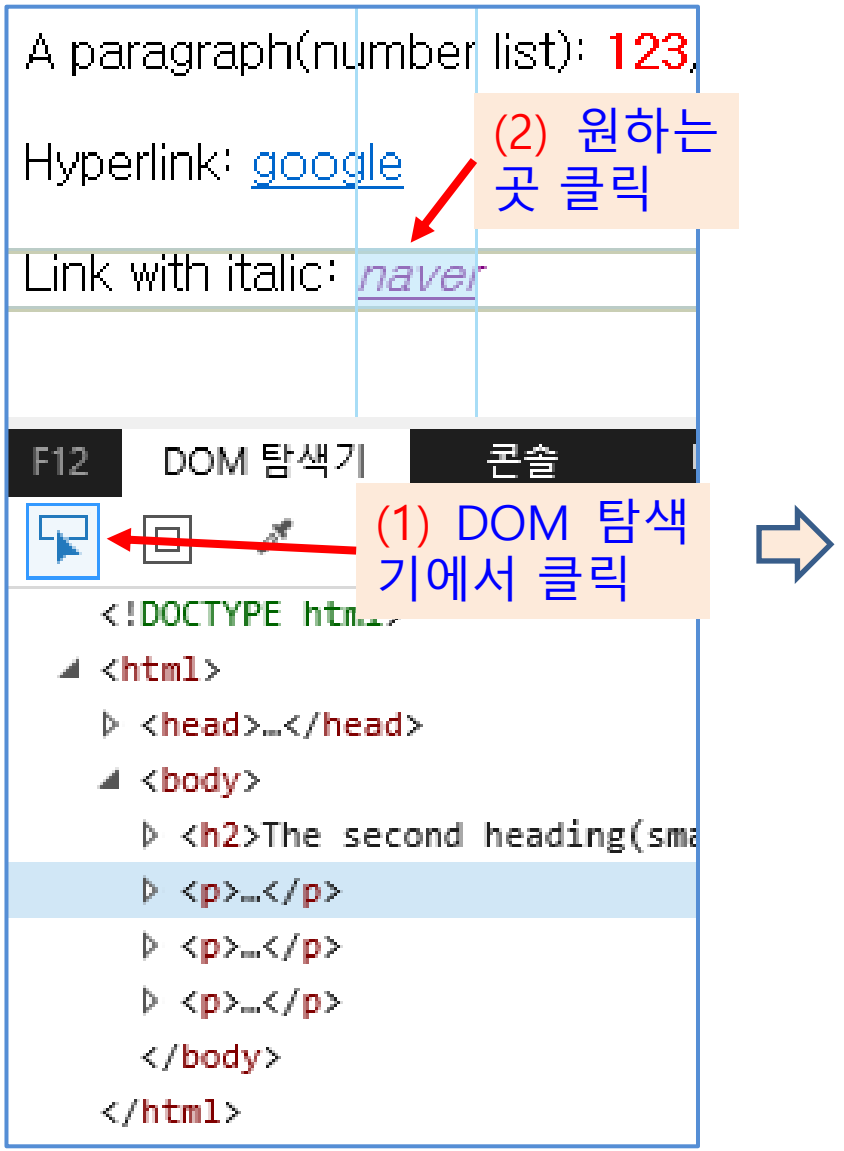
```
<!doctype html>
<head><title>An Example</title></head>
<body>
  <h3>The second heading(small font).</h3>
  <p>A paragraph(number list):
    <b style="color:red;">123</b>, <b style="color:red;">356</b>,
    <b>641</b>, <b>387</b> </p>
  <p>Hyperlink: <a href="http://www.google.com">google</a></p>
  <p>Link with italic: <a href="http://www.naver.com"><i>naver</i></a></p>
</body>
```



✦웹브라우저에서 HTML 문서 보기(*)



✦웹브라우저에서 HTML 문서내 원하는 부분의 코드 보기



웹 데이터 추출 (Web Scraping)

- 웹 문서에서 원하는 데이터를 정제해서 추출할 수 있다.
- 이를 위해서는 인터넷에서 문서를 읽어오고 이로부터 필요한 데이터를 추출하는 방법이 필요하다.
- 본 강의에서는 이를 위한 가장 기본적인 방법을 소개한다.

인터넷에서 웹 문서를 읽는 방법

- 모듈 urllib.request의 함수 urlopen( )을 사용한다(1).
  - 인수는 url(2) 즉, 인터넷 주소이다(문자열).
  - HTTPResponse라는 object를 반환한다.
  - 이 object는 모듈 BeautifulSoup의 메소드 prettify( )를 통하여 HTML 문서로 출력할 수 있다(다음 쪽에서 설명).
- Example

```
from urllib.request import *
wp = urlopen('http://mail.sogang.ac.kr')
print(type(wp)) #<class 'http.client.HTTPResponse'>
```

(1) 서로 연관된 모듈을 모아둔 것을 패키지(package)라고 한다. urllib는 패키지이고 request는 urllib에 포함된 모듈이다.

(2) url(uniform resource locator) : 네트워크 상에서 자원이 어디 있는지 알려주는 규약. 웹 사이트 주소가 이에 속한다.

데이터 추출

- 모듈 Beautiful Soup를 사용한다(외부 모듈로 설치 필요).
- Beautiful Soup는 HTML 코드를 입력 받아, 구문 분석(*)을 통하여 데이터 추출에 용이한 구조로 변환한다.
- Example

```
from urllib.request import *
from bs4 import *  # import Beautiful Soup

wp = urlopen('http://mail.sogang.ac.kr')

soup = BeautifulSoup(wp, 'html.parser') # parsing
print(type(soup))  # <class 'bs4.BeautifulSoup'>
print(soup.prettify()) # HTML 코드 출력
```

이런 형태로 출력된다

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "h
<html>
 <head>
  <title>
  </title>
  <meta content="IE=edge" http-equiv="X-UA-Compatible"> ...
```

(*) 파싱(parsing)이라고 하며, 일반적으로 트리 구조(parse tree)를 만든다.

Method .find_all( )을 통한 자료 추출

- 다음과 같은 HTML 문서를 읽었다고 하자

```
<p>A paragraph(number list):
 <b style="color:red;">123</b>, <b style="color:red;">356</b>,
 <b>641</b>, <b>387</b> </p>
<p>Hyperlink: <a href="http://www.google.com">google</a></p>
```

```
# soup : BeautifulSoup()의 반환 값이라고 가정
bList = soup.find_all('b') # 태그가 b인 원소들의 리스트
for b in bList :
  print(b.get_text()) # 123 356 641 387 (문자열 출력)
        # Method .get_text()는 bList의 원소에서 화면에 보이는
        # 것만을 추출하여 문자열로 반환.
```

- bList의 실제 내용

```
bList = [<b style="color:red;">123</b>,
 <b style="color:red;">356</b>, <b>641</b>, <b>387</b>]
```

```
<p>A paragraph(number list):
  <b style="color:red;">123</b>, <b style="color:red;">356</b>,
  <b>641</b>, <b>387</b> </p>
<p>Hyperlink: <a href="http://www.google.com">google</a></p>
```

✦ 태그가 b이고 속성이 style="color:red;"인 정수 356은 다음과 같이 태그와 속성 및 속성 값을 포함시켜 얻을 수 있다.

```
bList = soup.find_all('b', {'style':'color:red;'} )

print(bList[1].get_text()) # 356 (bList의 두번째)

  # bList = [<b style="color:red;">123</b>,
             <b style="color:red;">356</b>]
```

✦ Methods .find_all( )과 .get_text( )는 아래와 같은 유형의 Beautiful Soup object들에 적용할 수 있다.

```
BeautifulSoup()의 반환 유형: bs4.BeautifulSoup
find_all()의 반환 유형: bs4.element.ResultSet
```

```
<p>A paragraph(number list):
  <b style="color:red;">123</b>, <b style="color:red;">356</b>,
  <b>641</b>, <b>387</b> </p>
<p>Hyperlink: <a href="http://www.google.com">google</a></p>
```

✦ 두 단계 이상을 거쳐 태그가 b인 정수를 추출할 수도 있다(1).

```
pList = soup.find_all('p') # 태그가 p인 요소들의 리스트

# pList[0]에 태그가 b인 요소들이 있다.
bList = pList[0].find_all('b') # 태그가 b인 요소 리스트
for b in bList :
  print(b.get_text()) # 123 356 641 387 차례로 출력
```

✦ 태그, 속성 (및 속성값) 등을 조합하여 원하는 자료를 다양하게 추출할 수 있다(2).

(1) 사실 이 예에서는 불필요하다.
(2) 여기서는 종가 추출에 꼭 필요한 것만 보였다.

```
<p>A paragraph(number list):
  <b style="color:red;">123</b>, <b style="color:red;">356</b>,
  <b>641</b>, <b>387</b> </p>
<p>Hyperlink: <a href="http://www.google.com">google</a></p>
```

Method find( )

- find_all( )은 인수가 일치하는 모든 항목을 찾으나, find( )는 인수가 일치하는 첫 번째 항목만을 찾아 반환한다.
- find( ) 를 사용하면 속성 값도 얻을 수 있다.

```
# 태그가 b이고 style 속성이 없는 첫 번째 정수
b = soup.find('b', {'style':''})
print(b.get_text())  # 641

# 태그가 a인 첫 번째 요소의 속성 href의 값
url = soup.find('a')['href']
print(url) # http://www.google.com
```