

# Web Scraping (1)

---

Jehyuk Lee

Department of AI, Big Data & Management

Kookmin University

# Contents

---

- What is Web Scraping?
- Introduction to HTML
- 정적 웹페이지(Static Web Page) 스크래핑

# 1. What is Web Scraping?

---

# Web scraping

---

- 정의

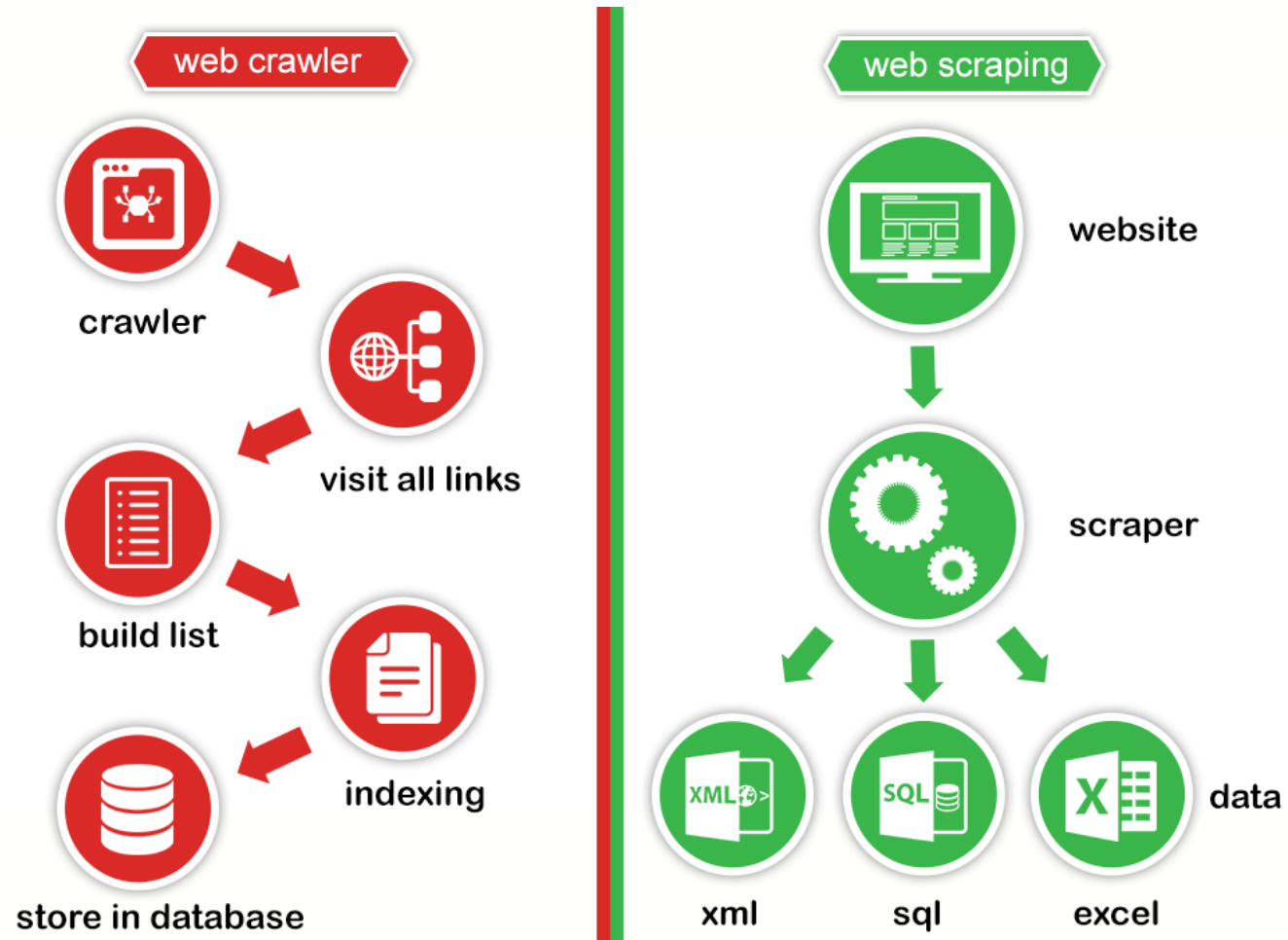
- 단일 웹페이지에서 원하는 정보를 추출하는 행위

- **Web Crawling vs Web Scraping**

- Web Crawling
    - URL을 탐색해 반복적으로 링크를 찾고 가져오는 과정으로, 특정 웹페이지를 목표로 하지 않음
  - Web Scraping
    - 우리가 정한 특정 웹 페이지에서 데이터를 추출

# Web scraping

- Web Crawling vs Web Scraping



(Source: <http://prowebscraping.com/web-scraping-vs-web-crawling/>)

# Web scraping

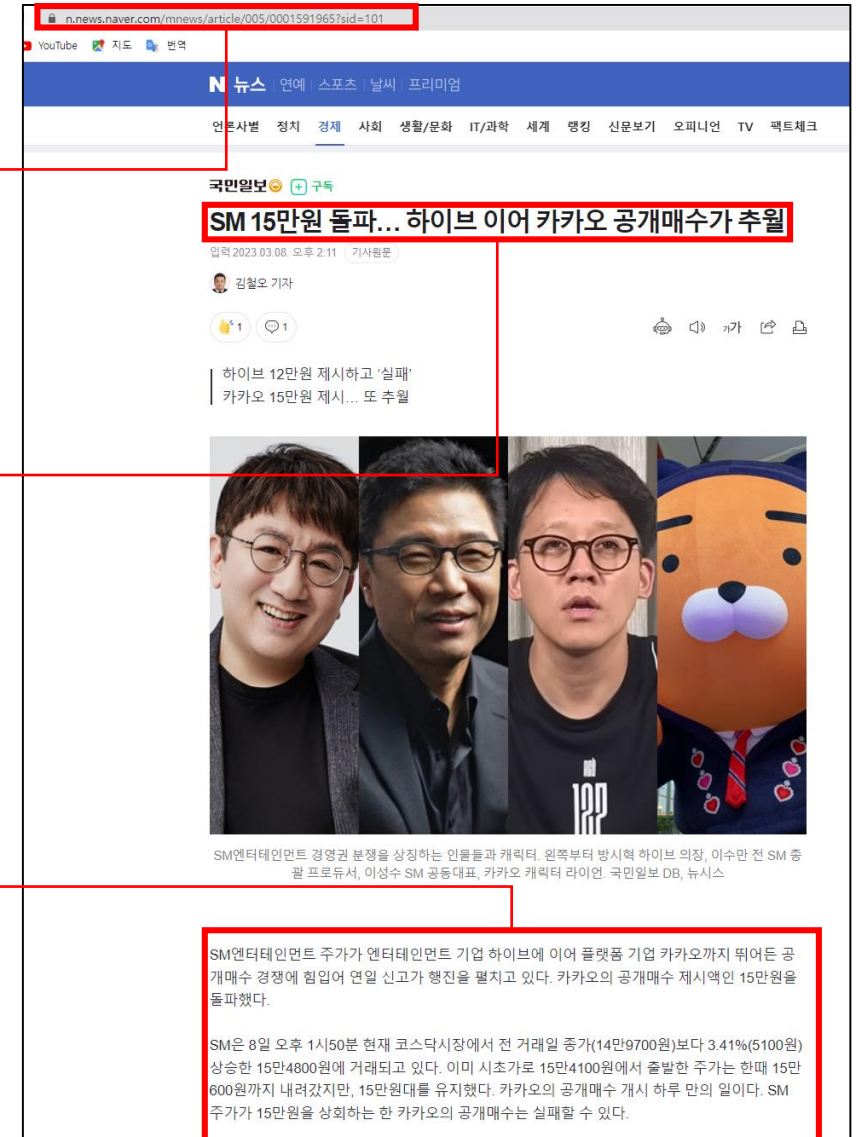
## • 우리는 인터넷에서 다양한 정보들을 얻습니다.

- 홈페이지 '열기'
- 필요한 정보의 위치를 '찾기'
- 필요한 정보를 '보기'

홈페이지 '열기'

정보위치 '찾기'  
필요정보 '보기'

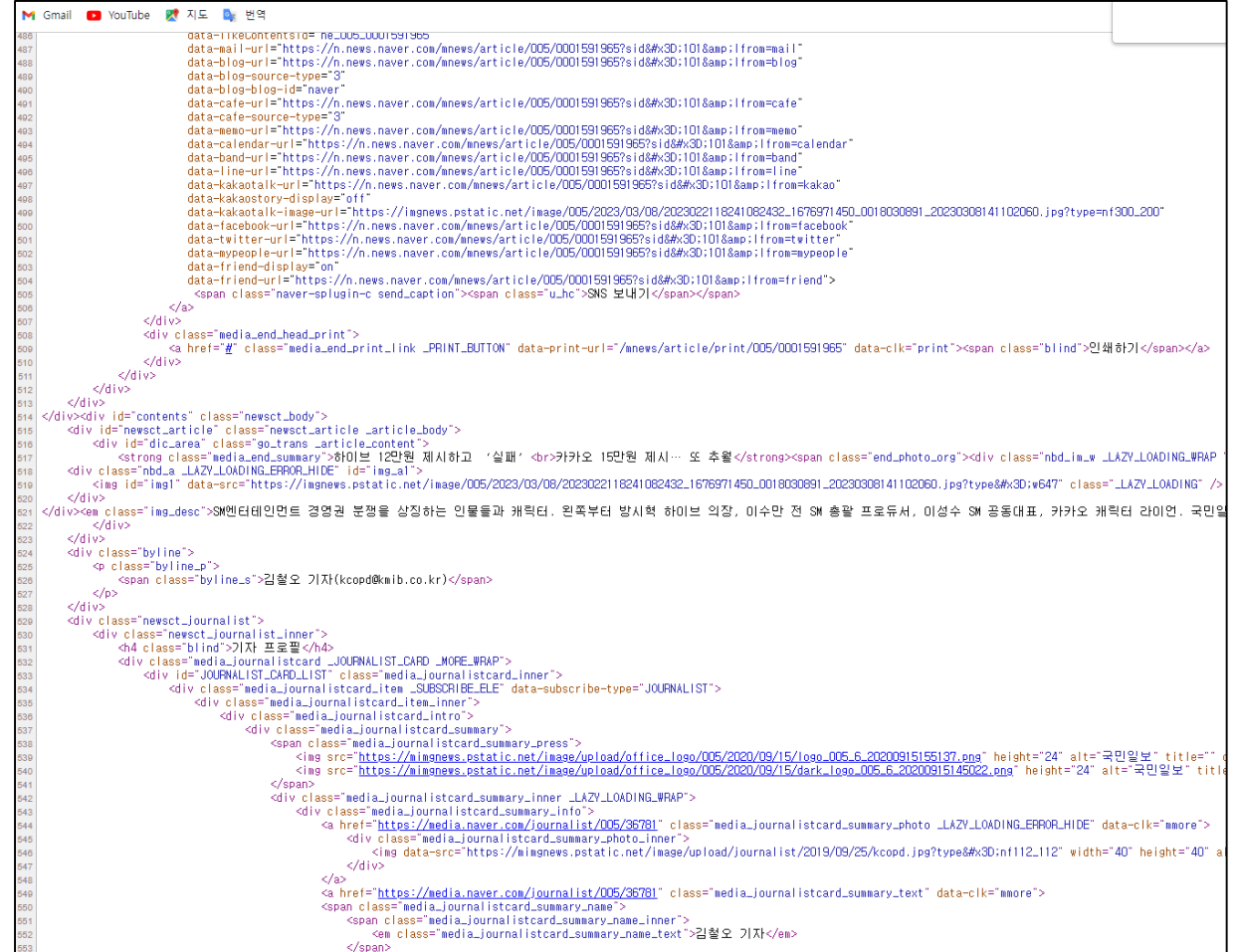
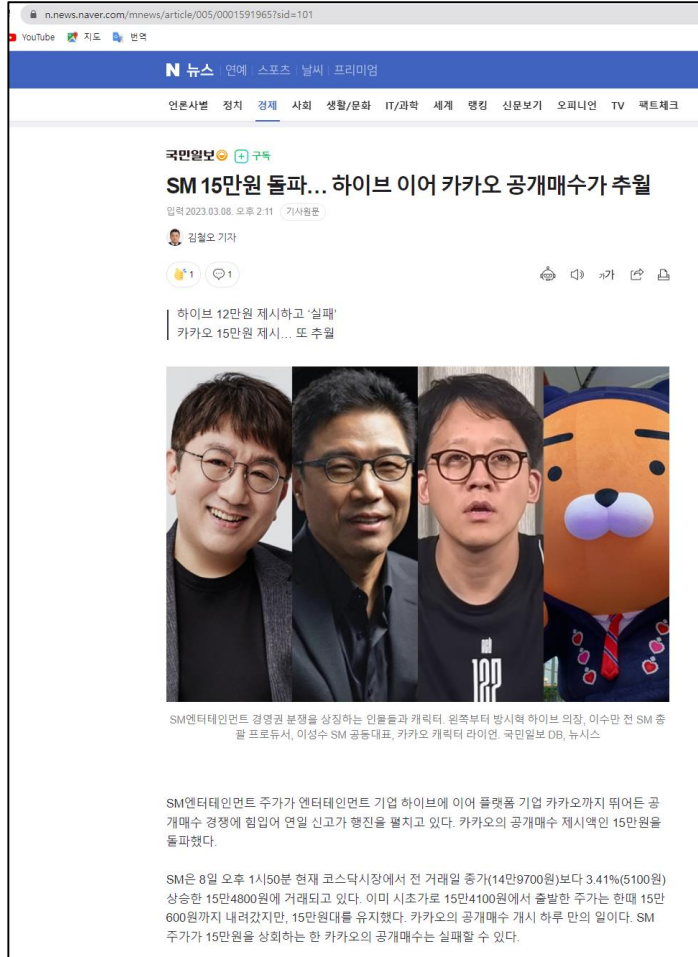
- 사실 여러분은 이미 (수동으로) Web Scraping을 하고 있었습니다.
- 이러한 행위를 파이썬을 활용해 자동으로 수행하도록 합시다.
- 어떻게?



(Source: <https://n.news.naver.com/mnews/article/005/0001591965?sid=101>)

# Web scraping

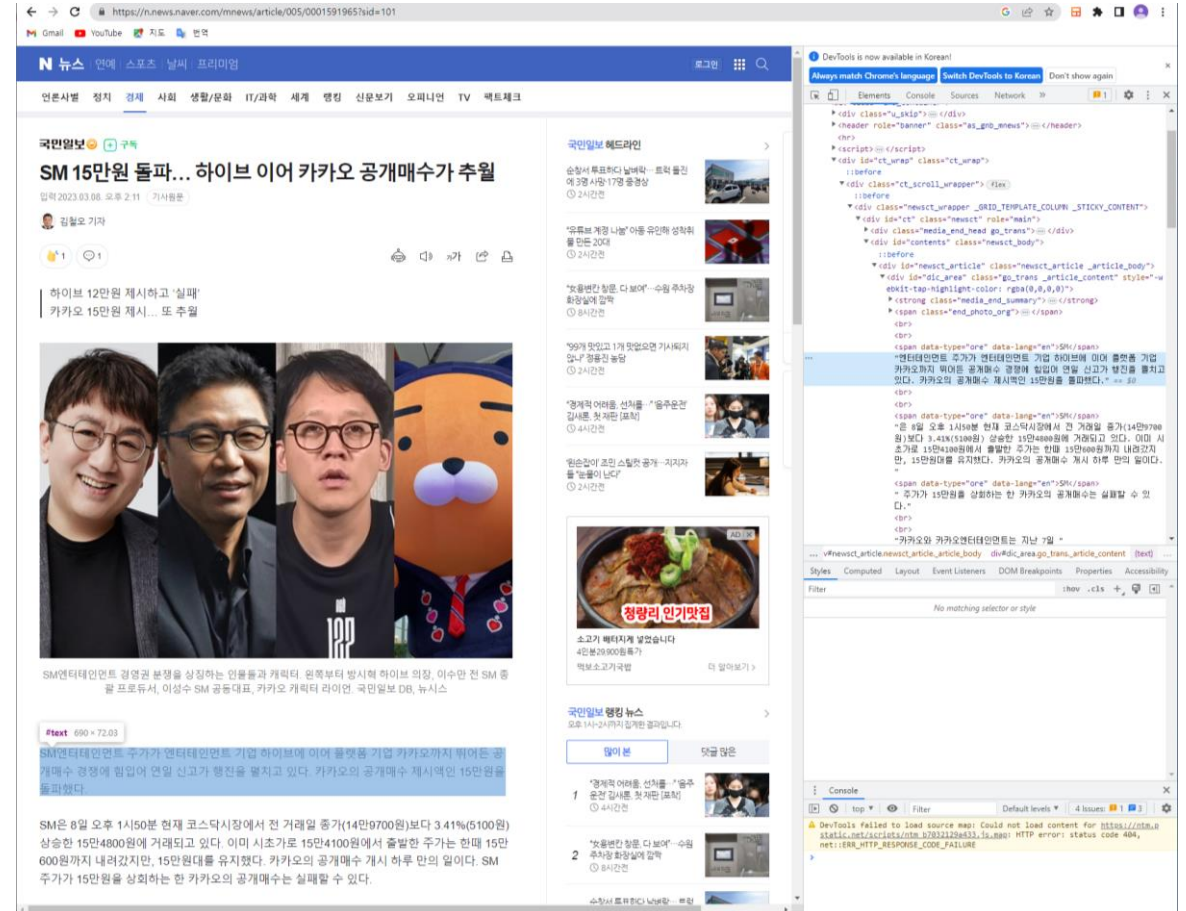
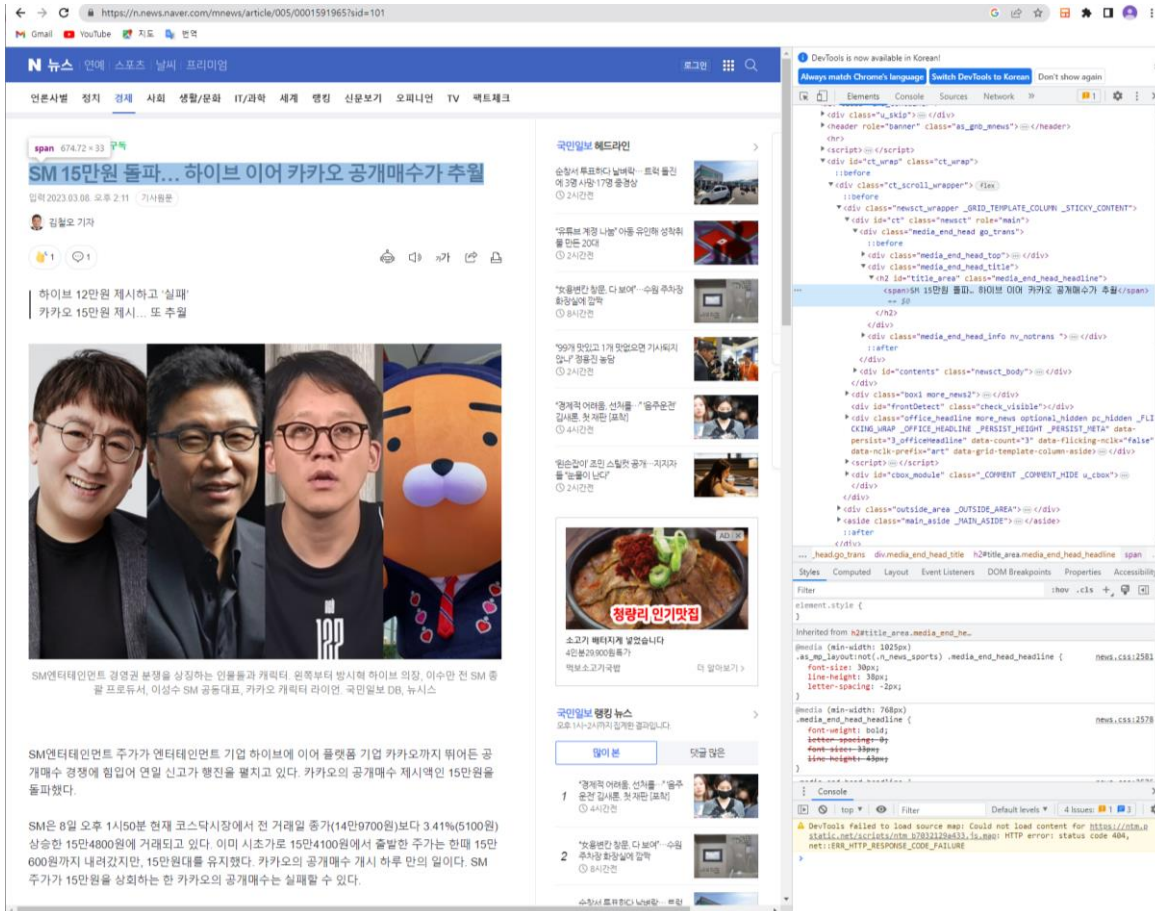
- 우리가 보는 홈페이지는 사실 이런 소스코드로 구성되었습니다.



(Source: <https://n.news.naver.com/mnews/article/005/0001591965?sid=101>)

# Web scraping

- 우리가 보는 홈페이지는 사실 이런 소스코드로 구성되었습니다.



(Source: <https://n.news.naver.com/mnews/article/005/0001591965?sid=101>)



# Web scraping

---

- 이러한 특성을 활용하여 실제로 필요한 정보를 추출합니다.
  - 이를 위해서는 이 소스코드(HTML)에 대한 이해가 필요합니다.
  - 다음 section에서 HTML에 대해서 먼저 배울 예정입니다.
- 그 전에 다음 내용들을 좀 더 다루고 넘어가겠습니다.
  - Web scraping 전체 프로세스
  - Client-Server간 통신 체계

(Source: <https://n.news.naver.com/mnews/article/005/0001591965?sid=101>)

# Web scraping

## • Web Scraping Process

- 웹페이지 열기: (1), (2)
- 정보위치 찾기: (3)
- 필요정보 저장: (4)



(Source: <https://topwebscrapingservice.wordpress.com/>)

# Web scraping

## • Web Scraping Process

### – 웹페이지 '열기'

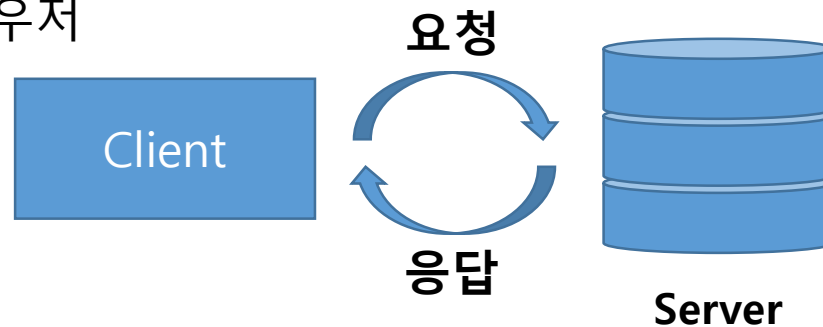
- 요청(Request): 클라이언트(혹은 사용자)가 서버에게 리소스나 서비스를 요청
- 응답(Response): 요청한 사항에 대해 서버가 클라이언트에게 리소스나 서비스를 제공

### Client

- 서버에서 제공하는 서비스를 받는 입장
- 데스크탑, 태블릿과 같은 장비
- Chrome, firefox등의 웹 브라우저

### Server

- 서비스를 제공하는 입장
- 웹서버, 메일서버, 파일 서버 등
- 여러 client를 대상으로 서비스 제공



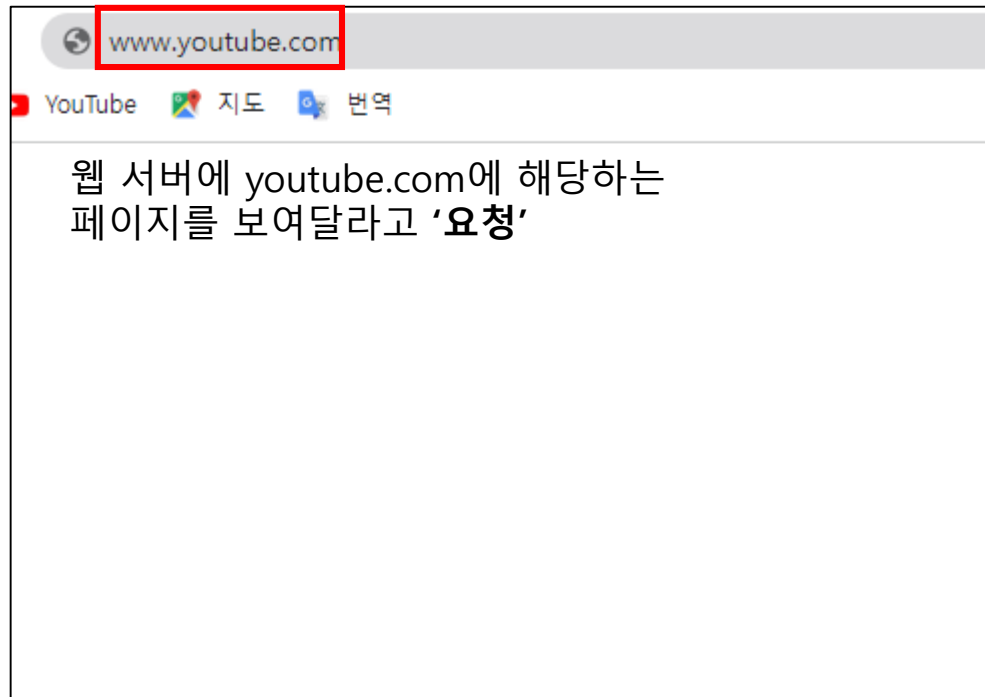
(Source: <https://topwebscrapingservice.wordpress.com/>)

# Web scraping

## • Web Scraping Process

### – 웹페이지 '열기'

- 요청(Request): 클라이언트(혹은 사용자)가 서버에게 리소스나 서비스를 요청
- 응답(Response): 요청한 사항에 대해 서버가 클라이언트에게 리소스나 서비스를 제공



(Source: <https://www.youtube.com>)

# Web scraping

## • Web Scraping Process

- 웹페이지 열기: (1), (2)
- 정보위치 찾기: (3)
- 필요정보 저장: (4)

결국 Web scraper에서도

- 파이썬 프로그램(Client)가 웹서버(Server)에 '요청'
- 서버가 해당 사항에 대해 서비스를 제공하는 '응답'



(Source: <https://topwebscrapingservice.wordpress.com/>)

## 2. Introduction to HTML

---

# 웹페이지를 구성하는 기본 요소

---

- **HTML**

- 웹페이지 contents(텍스트, 이미지 등)를 정의

- **CSS**

- HTML로 입력한 contents들에 디자인 스타일을 적용

- **Javascript**

- 웹페이지의 동작과 상호작용을 정의

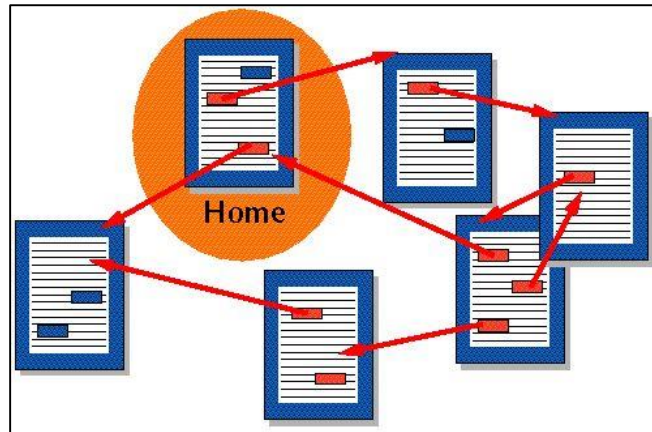
(Source: <https://nonipc.com/entry/%EC%9E%90%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8-%EC%97%AD%ED%95%A0%EA%B3%BC-%EC%BD%94%EB%93%9C-%EC%B6%94%EA%B0%80-%EB%B0%A9%EB%B2%95>)

# HTML

- What is HTML

- HyperText Markup Language

- 하이퍼텍스트와 마크업 언어를 합친 언어
      - 하이퍼텍스트: 하이퍼링크를 통해 한 문서에서 다른 문서로 즉시 접근할 수 있는 텍스트
      - 마크업 언어: 태그 등을 이용하여 문서나 데이터의 구조를 명기하는 언어
    - 확장자: .html



```
Document: Bungler OED      At: "<entry>"

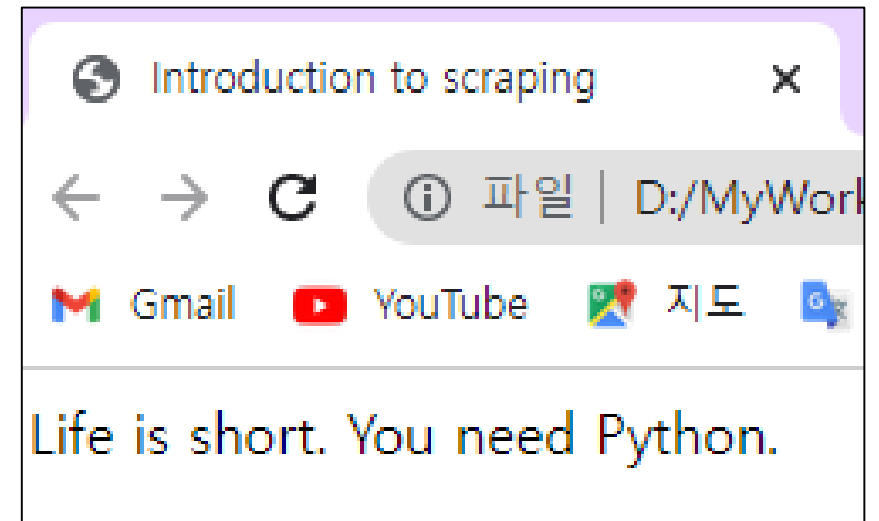
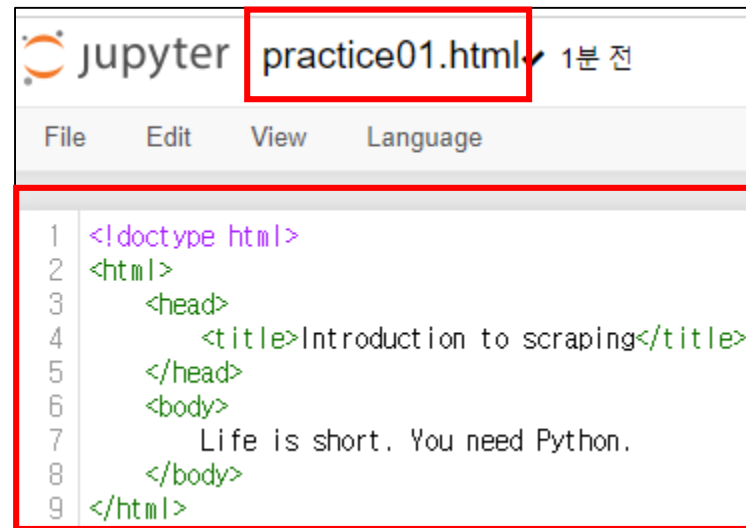
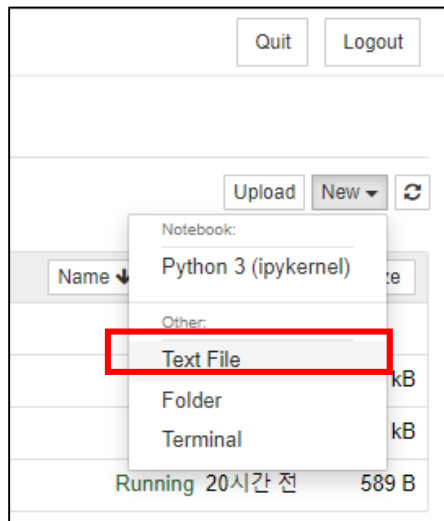
<entry>
  <hwsec>
    <hwgp>
      <hwlem>bungler</hwlem>
      <pron>b<I>ŋ</I>ˈɡlɜː</pron>. </hwgp>
      <vfl>Also <vd>b</vd> <vf>bongler</vf>
      </vfl>
      <etym>f. as prec. + <xra><xlem>-ER</xle
    </etym>
    <sen>One who bungles; a clumsy unskilful
    <quot>
      <qdat>1533 </qdat>
      <auth>MORE </auth>
      <wk>Answ. Poyson. 3k. </wk>Wks. (1557
      <qtxt>He is euen but a very bungler.
    </qtxt>
  </sen>
</entry>
```

(Source: <https://ko.wikipedia.org/wiki/%ED%95%98%EC%9D%B4%ED%8D%BC%ED%85%8D%EC%8A%A4%ED%8A%B8,> )



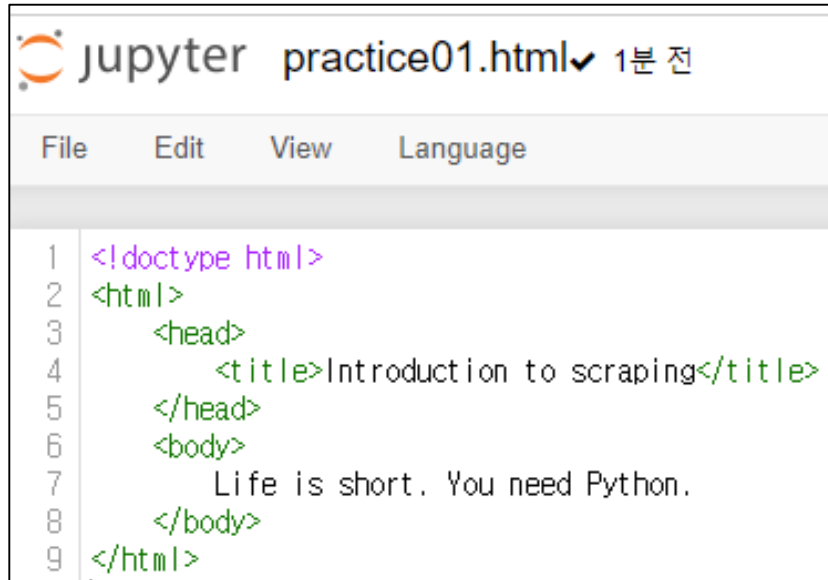
# HTML

- HTML 연습하기



# HTML

- HTML 연습하기



```
1 <!doctype html>
2 <html>
3     <head>
4         <title>Introduction to scraping</title>
5     </head>
6     <body>
7         Life is short. You need Python.
8     </body>
9 </html>
```

**<!doctype html>**: HTML문서임을 의미

**<html>, </html>**: HTML문서의 시작과 끝을 의미

**<head>, </head>**: 브라우저가 문서를 해석하는데 필요한 정보

**<title>, </title>**: 브라우저의 제목 표시줄에 들어갈 내용

**<body>, </body>**: 웹페이지에서 보게 될 주요 정보

# HTML

## • HTML 연습하기

### – 속성 추가하기

- <태그 속성 = "값">의 형태로 사용.
- 태그마다 여러 속성을 부여할 수 있음

```
1  <!-- <!doctype html> -->
2  <html>
3    <head>
4      <title>Introduction to scraping</title>
5    </head>
6    <body>
7      <font color = 'red' face='Dotum'>Life is short. You need Python.</font><br>
8      <img src = "GPT.gif" width="500".height="400">
9    </body>
10 </html>
```

- (1) font라는 tag에 color, face 속성을 부여하고 값을 지정
- (2) img라는 tag에 src, width, height 속성을 부여하고 값을 지정

# HTML

## • HTML 연습하기

### – 표(table)

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>초간단 테이블</title>
5   </head>
6   <body>
7     <table border="1">
8       <caption>첫번째 표</caption>
9       <tr>
10        <th>상품</th>
11        <th>가격</th>
12      </tr>
13      <tr>
14        <td>item 01</td>
15        <td>1000</td>
16      </tr>
17      <tr>
18        <td>item 02</td>
19        <td>2000</td>
20      </tr>
21    </table>
22    <table border="2">
23      <caption>두번째 표</caption>
24      <tr>
25        <th>상품</th>
26        <th>가격</th>
27      </tr>
28      <tr>
29        <td>item 03</td>
30        <td>3000</td>
31      </tr>
32      <tr>
33        <td>item 04</td>
34        <td>4000</td>
35      </tr>
36    </table>
37  </body>
38 </html>
```

초간단 테이블

← → ↻ ⓘ 파일 | D:/MyWorkspace

Gmail YouTube 지도 번역

첫번째 표

상품	가격
item 01	1000
item 02	2000

두번째 표

상품	가격
item 03	3000
item 04	4000

태그	비고
<table>	테이블을 만드는 태그
<th>	테이블의 헤더부분을 만드는 태그
<tr>	테이블의 행을 만드는 태그
<td>	테이블의 열을 만드는 태그

(Source: <https://coding-factory.tistory.com/184>)

# HTML

- HTML 연습하기

- 기타 여러가지 태그들

Tag	역할
<h1> </h1>, <h2> </h2>, ...	크기가 다른 텍스트. 숫자가 클수록 글씨 크기는 작음
<p> </p>	텍스트 문단을 지정
<li> </li>	목록을 만들 때 사용
<div> </div>	블록단위로 페이지의 부분공간을 정의(레이아웃)
<span> </span>	줄단위로 페이지의 부분공간을 정의
 	줄바꿈

# HTML

## • HTML 연습하기

### – 기타 여러가지 태그들

```
1 <!-- <!doctype html> -->
2 <html>
3   <head>
4     <title>Practice HTML</title>
5   </head>
6   <body>
7     <div style = 'width:600px; height:100p'>
8       <h1>핸드폰 분실로 인한 전화번호 새로 등록하기</h1>
9       <h2>번거로운 그 자체</h2>
10      <p>
11        저는 최근에 새로운 핸드폰을 구입해서, 주변 사람들의 연락처를 다시 입력해야 했습니다.
12        가족들의 전화번호는 이미 기억하고 있어서 문제 없었지만, 다음 친구들의 전화번호는 다시
13        입력해야 했습니다.<br></p>
14        <li>011-1111-2222</li>
15        <li>010-3333-4444</li>
16        <li>02-555-1234</li>
17        <li>010-7777-8888</li>
18      <p>핸드폰의 주소록에 연락처를 등록하는 것은 꽤 번거로운 작업이지만, 한 번 등록해 놓으면 훨
19      씩 편리하게 사용할 수 있습니다.</p>
20    </div>
21
22    <div style = 'width:600px;height:100p;margin:50px'>
23      <font color = 'red' face='Dotum'>Life is short. You need Python.</font><br>
24      <img src = "GPT.gif" width="500",height="400">
25    </div>
26  </body>
27</html>
```



## 핸드폰 분실로 인한 전화번호 새로 등록하기

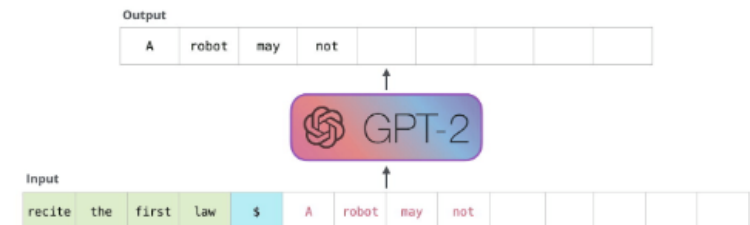
### 번거로운 그 자체

저는 최근에 새로운 핸드폰을 구입해서, 주변 사람들의 연락처를 다시 입력해야 했습니다. 가족들의 전화번호는 이미 기억하고 있어서 문제 없었지만, 다음 친구들의 전화번호는 다시 입력해야 했습니다.

- 011-1111-2222
- 010-3333-4444
- 02-555-1234
- 010-7777-8888

핸드폰의 주소록에 연락처를 등록하는 것은 꽤 번거로운 작업이지만, 한 번 등록해 놓으면 훨씬 편리하게 사용할 수 있습니다.

Life is short. You need Python.

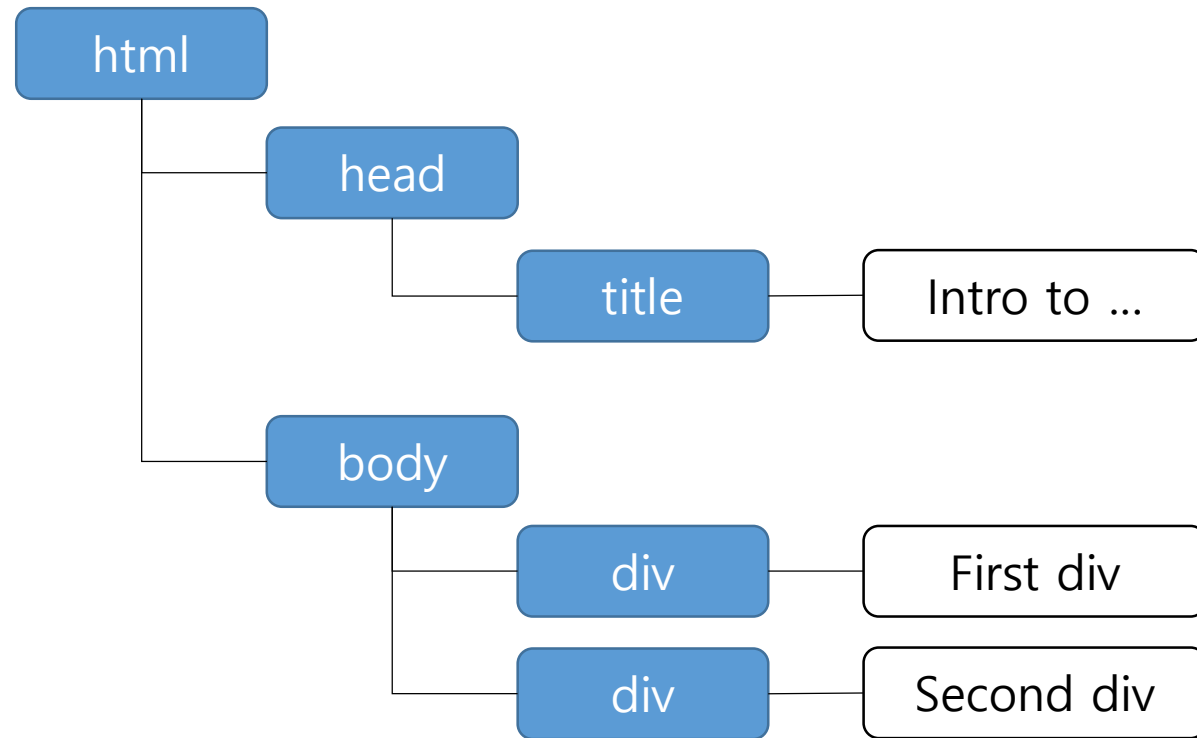


# HTML

- HTML 구조

- HTML은 트리 구조(계층 구조)로 되어있음

```
1 <html>
2   <head>
3     <title>Intro to Web Scraping</title>
4   </head>
5   <body>
6     <div>First div</div>
7     <div>Second div</div>
8   </body>
9 </html>
```



# CSS

---

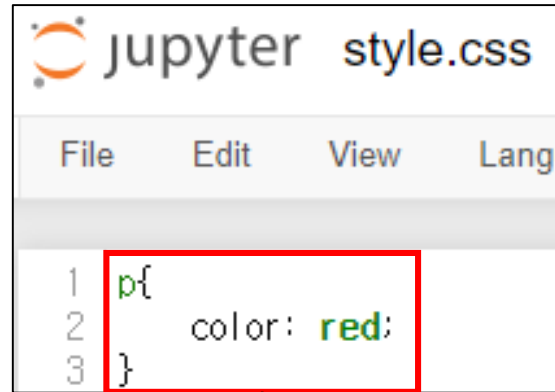
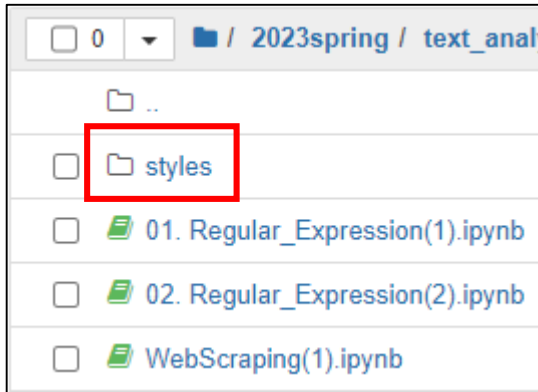
- **What is CSS**

- CSS(Cascading Style Sheets)
- HTML로 입력한 요소에 디자인 스타일을 적용할 때 사용하는 언어
  - 웹 페이지 구성 요소의 크기, 색상 변경 등
- 확장자: .css

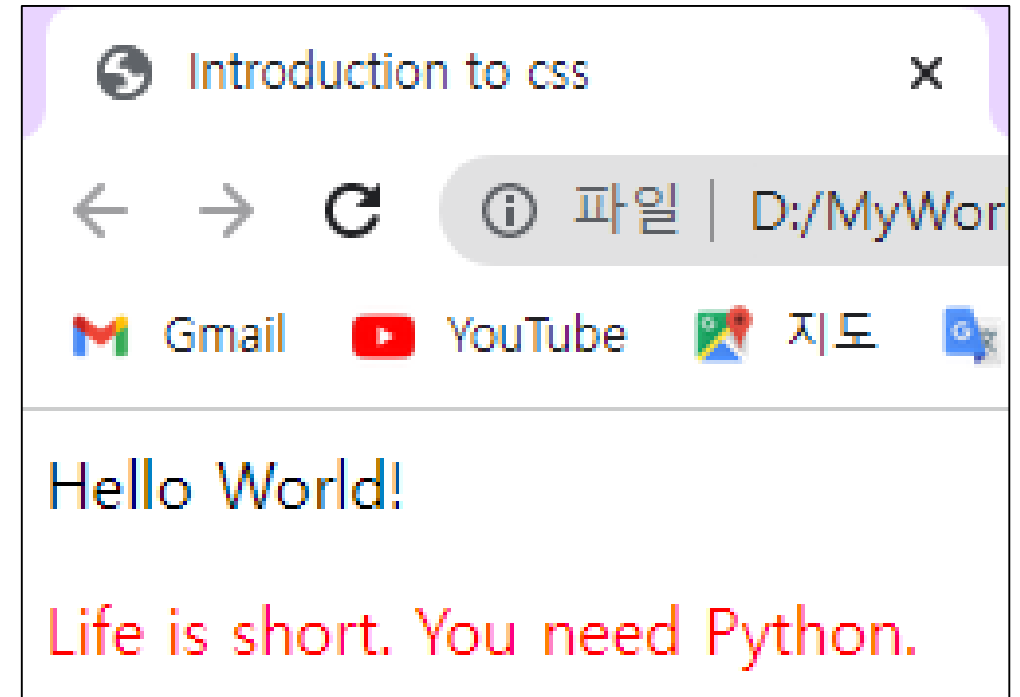
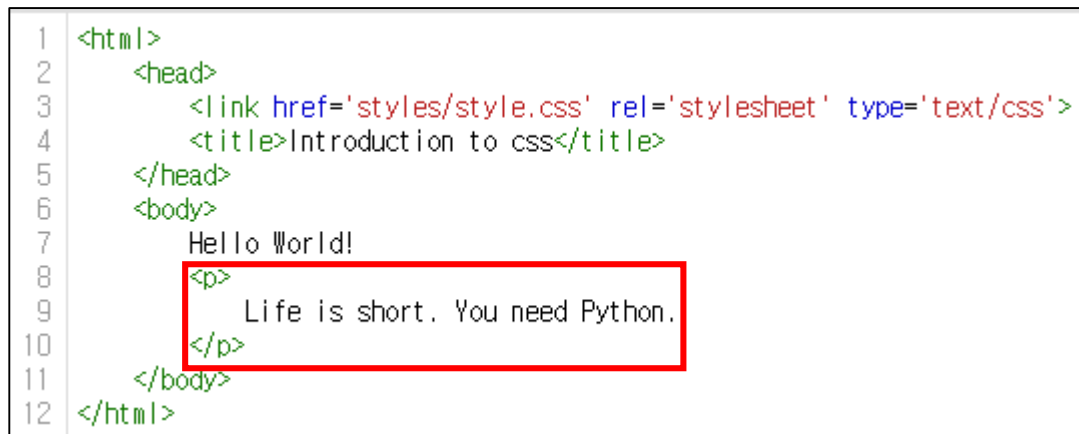


# CSS

- What is CSS



<p>태그를 적용할 때 텍스트 색상을 빨간색으로 지정



# Javascript

---

- **What is Javascript**

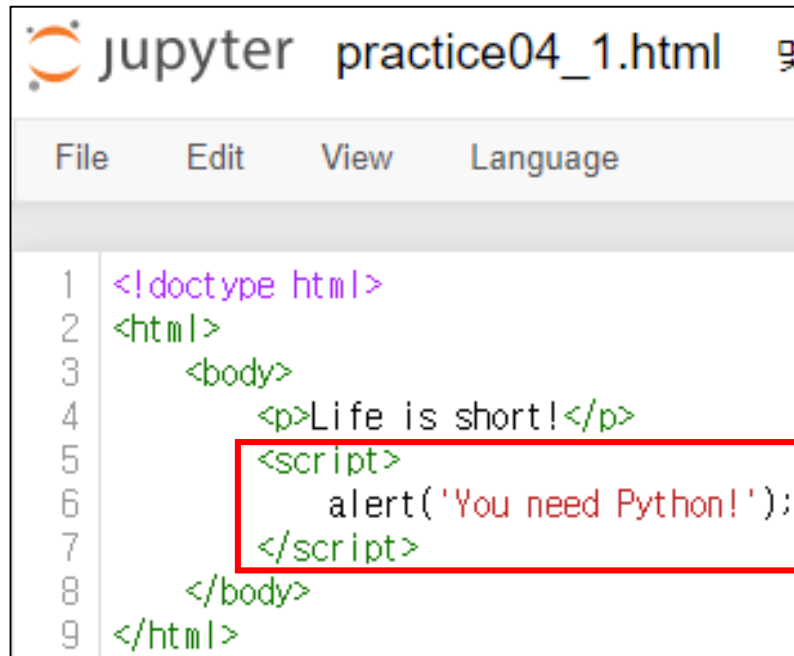
- 웹페이지의 동작과 상호작용을 정의
  - 동작 예시: 움직이는 요소, 이미지 슬라이더 등
  - 상호작용: 사용자가 마우스를 올리면 메뉴 보여주기 등

# Javascript

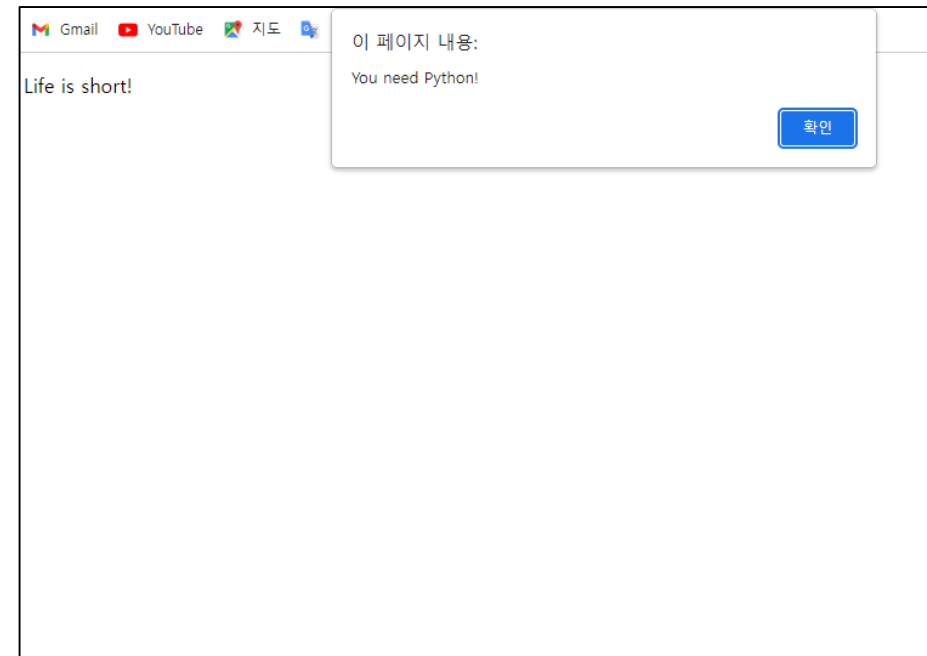
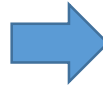
- What is Javascript

- HTML에서 Javascript사용하기 (1)

- <script> </script> 태그를 사용하여 javascript사용하기



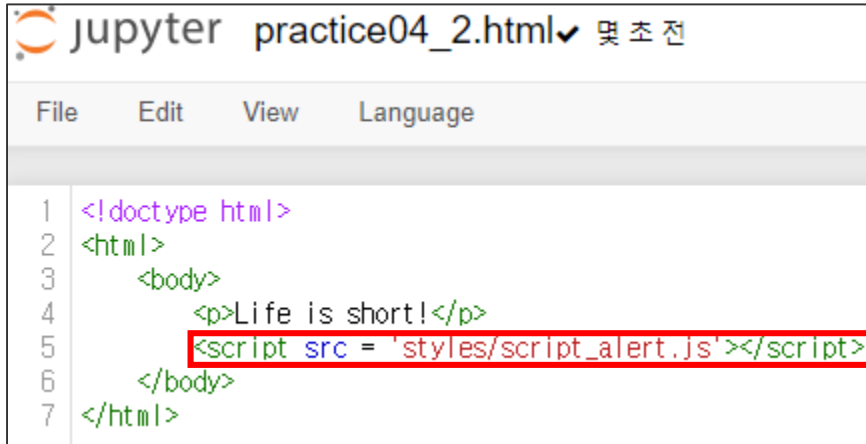
```
1 <!doctype html>
2 <html>
3   <body>
4     <p>Life is short!</p>
5     <script>
6       alert('You need Python!');
7     </script>
8   </body>
9 </html>
```



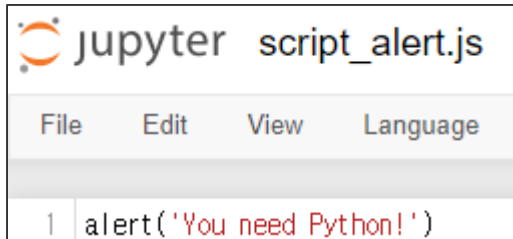
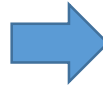
# Javascript

## • What is Javascript

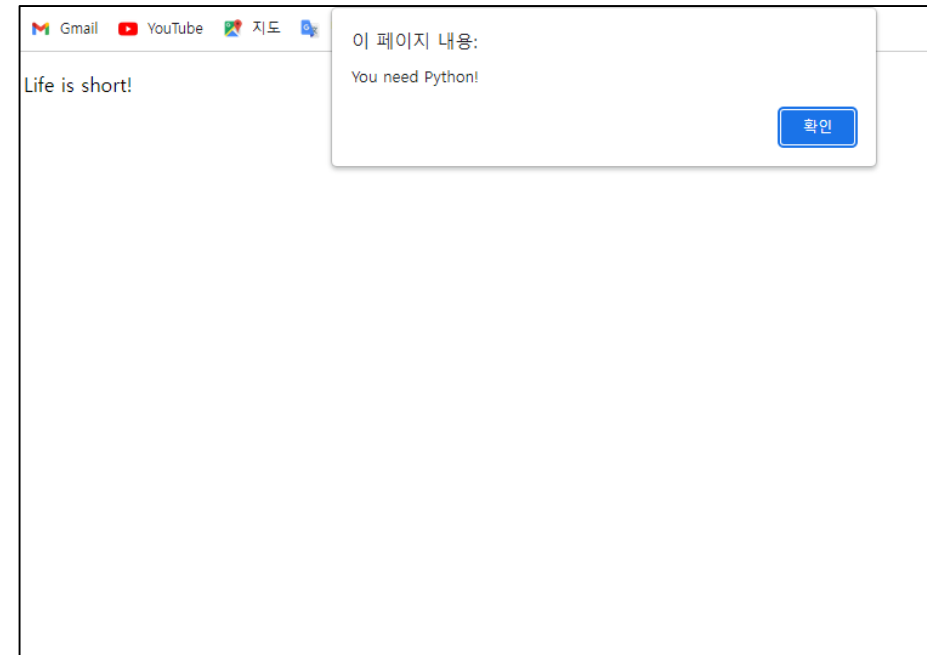
- HTML에서 Javascript사용하기 (2)
  - 별도의 javascript파일(.js)을 생성하고 HTML파일에서 불러오기



```
jupyter practice04_2.html ✓ 몇 초 전
File Edit View Language
1 <!doctype html>
2 <html>
3   <body>
4     <p>Life is short!</p>
5     <script src = 'styles/script_alert.js'></script>
6   </body>
7 </html>
```



```
jupyter script_alert.js
File Edit View Language
1 alert('You need Python!')
```



### 3. 정적 웹페이지(Static Web Page) 스크래핑

---

# 실제로 Web Scraping을 해봅시다!

---

- HTML문서에서 원하는 정보를 추출하는 것

- 단순히 말하자면, HTML 파일(긴 문자열)에서 원하는 정보(짧은 문자열)를 추출

- 다음과 같은 순서로 진행됩니다.

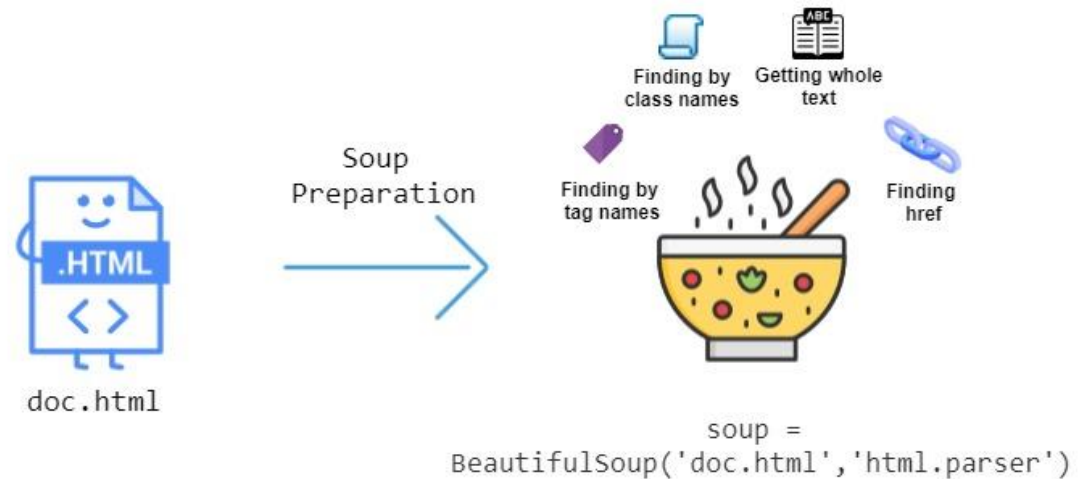
- 단순한 HTML형식의 문자열에서 scraping
  - 로컬의 HTML파일에서 scraping
  - 실제 웹사이트에서 scraping

# 실제로 Web Scraping을 해봅시다!

- BeautifulSoup

- Python에서 scraping하는데 필요한 함수를 모아놓은 라이브러리

# BeautifulSoup



(Source: <https://newstellar.tistory.com/1>, <https://stackabuse.com/guide-to-parsing-html-with-beautifulsoup-in-python/>)

# Local에 있는 HTML파일 Scraping

- 예제 3: Table이 있는 HTML파일

```
jupyter practice05.html ✓ 몇 초 전
File Edit View Language

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>초간단 테이블</title>
5   </head>
6   <body>
7     <table border="1">
8       <caption>첫번째 표</caption>
9       <tr>
10        <th>상품</th>
11        <th>가격</th>
12      </tr>
13      <tr>
14        <td>item 01</td>
15        <td>1000</td>
16      </tr>
17      <tr>
18        <td>item 02</td>
19        <td>2000</td>
20      </tr>
21    </table>
22    <table border="2">
23      <caption>두번째 표</caption>
24      <tr>
25        <th>상품</th>
26        <th>가격</th>
27      </tr>
28      <tr>
29        <td>item 03</td>
30        <td>3000</td>
31      </tr>
32      <tr>
33        <td>item 04</td>
34        <td>4000</td>
35      </tr>
36    </table>
37  </body>
38 </html>
```

초간단 테이블

← → ↺ ⓘ 파일 | D:/MyWorkspace

Gmail YouTube 지도 번역

첫번째 표

상품	가격
item 01	1000
item 02	2000

두번째 표

상품	가격
item 03	3000
item 04	4000

(Source: <https://newstellar.tistory.com/1>, <https://stackabuse.com/guide-to-parsing-html-with-beautifulsoup-in-python/>)



# Local에 있는 HTML파일 Scraping

## • 예제 3

### – 로컬 HTML파일 열기

```
bs_obj = BeautifulSoup(open("practice05.html", encoding="utf8"), "html.parser")
```

### – <table> 태그에 해당하는 내용 추출

```
table = bs_obj.find_all('table')
```

```
print(f'table 수: {len(table)}')  
print()  
print(table)
```

table 수: 2

```
[<table border="1">  
<caption>첫번째 표</caption>  
<tr>  
<th>상품</th>  
<th>가격</th>  
</tr>  
<tr>  
<td>item 01</td>  
<td>1000</td>  
</tr>  
<tr>  
<td>item 02</td>  
<td>2000</td>  
</tr>  
</table>, <table border="2">
```

```
<caption>두번째 표</caption>  
<tr>  
<th>상품</th>  
<th>가격</th>  
</tr>  
<tr>  
<td>item 03</td>  
<td>3000</td>  
</tr>  
<tr>  
<td>item 04</td>  
<td>4000</td>  
</tr>  
</table>]
```

# Local에 있는 HTML파일 Scraping

## • 예제 3

### – 로컬 HTML파일 열기

```
bs_obj = BeautifulSoup(open("practice05.html", encoding="utf8"), "html.parser")
```

### – <table> 태그에 해당하는 내용 추출

```
table = bs_obj.find_all('table')
```

```
print(f'table 수: {len(table)}')  
print()  
print(table)
```

table 수: 2

```
[<table border="1">  
<caption>첫번째 표</caption>  
<tr>  
<th>상품</th>  
<th>가격</th>  
</tr>  
<tr>  
<td>item 01</td>  
<td>1000</td>  
</tr>  
<tr>  
<td>item 02</td>  
<td>2000</td>  
</tr>  
</table>, <table border="2">
```

```
<caption>두번째 표</caption>  
<tr>  
<th>상품</th>  
<th>가격</th>  
</tr>  
<tr>  
<td>item 03</td>  
<td>3000</td>  
</tr>  
<tr>  
<td>item 04</td>  
<td>4000</td>  
</tr>  
</table>]
```

# Local에 있는 HTML파일 Scraping

## • 예제 3

- <table>태그의 속성값을 특정하여 내용 추출

```
table = bs_obj.find_all('table', {'border':'2'})
```

```
table
```

```
[<table border="2">
  <caption>두번째 표</caption>
  <tr>
    <th>상품</th>
    <th>가격</th>
  </tr>
  <tr>
    <td>item 03</td>
    <td>3000</td>
  </tr>
  <tr>
    <td>item 04</td>
    <td>4000</td>
  </tr>
</table>]
```

# 실제 웹페이지 Scraping하기

- 예제 4

- <https://ai-dev.tistory.com/1>

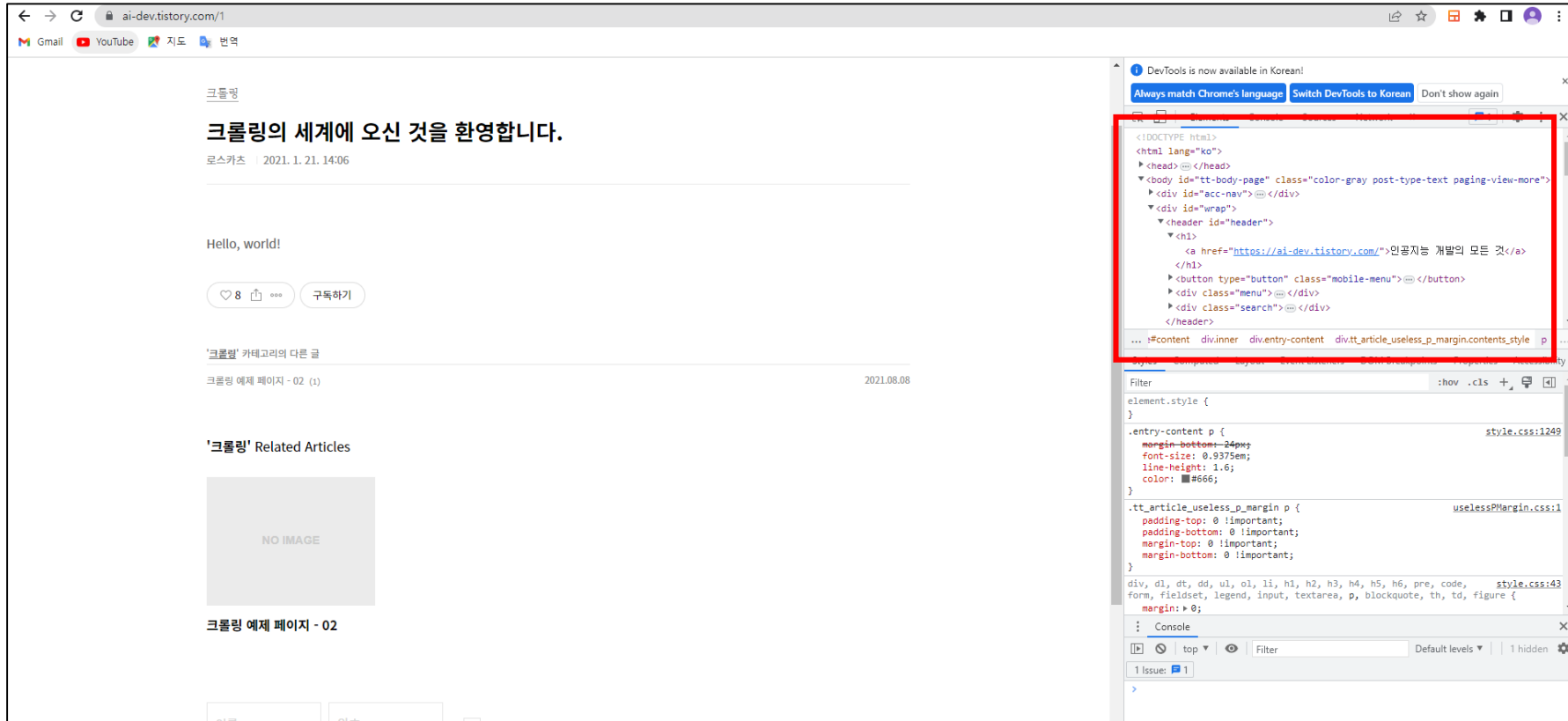


(Source: <https://ai-dev.tistory.com/1>)

# 실제 웹사이트 Scraping하기

## • 예제 4

– F12를 눌러서 다음과 같은 화면이 나타나게 해보세요.

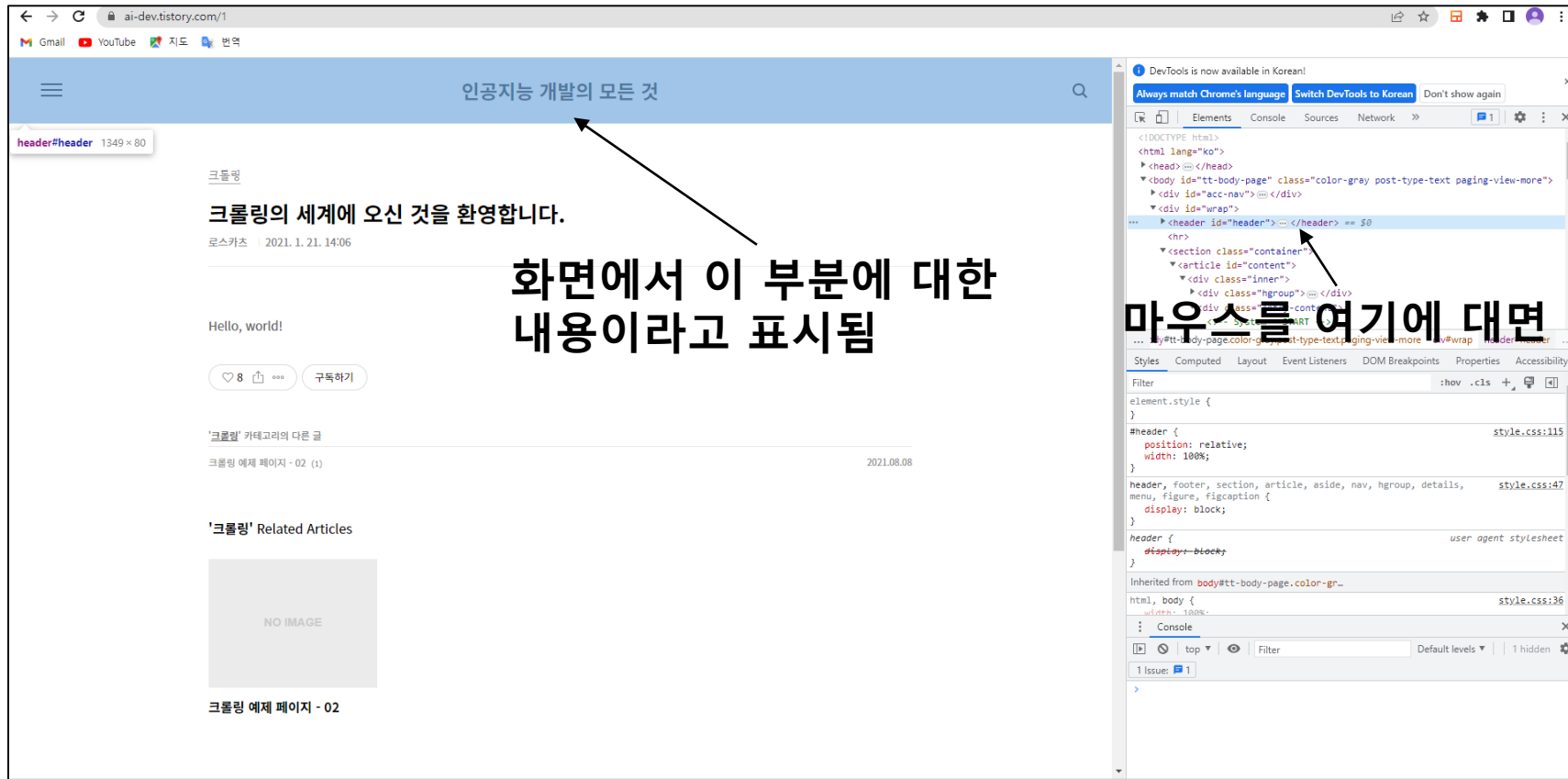


(Source: <https://ai-dev.tistory.com/1>)

# 실제 웹페이지 Scraping하기

## • 예제 4

- HTML의 어느 부분이 화면과 연관되어 있는지 파악이 용이합니다.

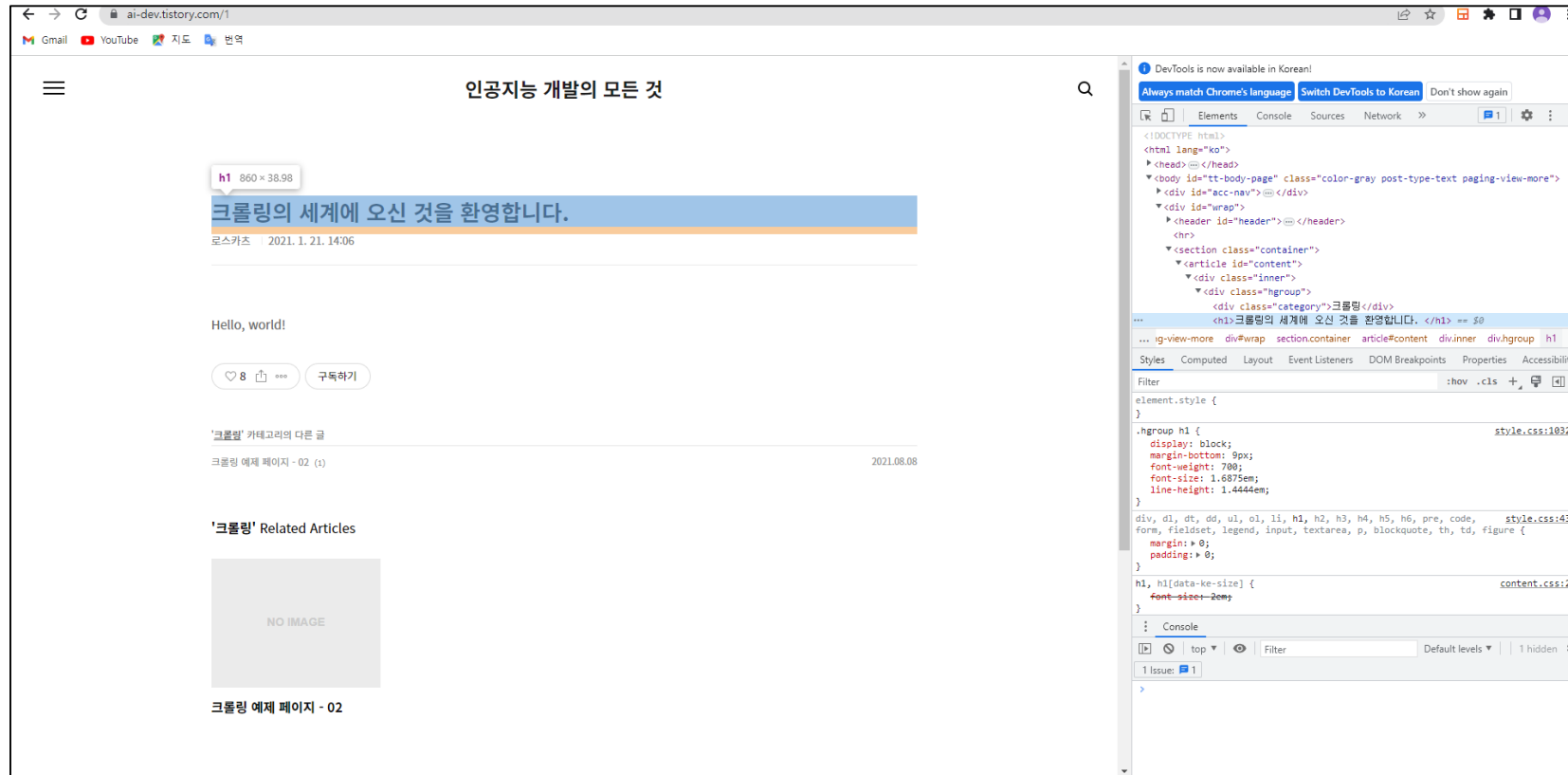


(Source: <https://ai-dev.tistory.com/1>)

# 실제 웹페이지 Scraping하기

## • 예제 4.1

– 제목을 추출해봅시다.



<h1>태그에  
해당합니다.

(Source: <https://ai-dev.tistory.com/1>)

# 실제 웹페이지 Scraping하기

## • 예제 4.1

- 웹페이지 열기 (urllib 라이브러리 사용)

```
from urllib.request import urlopen
```

```
url = 'https://ai-dev.tistory.com/1'
```

```
html = urlopen(url)
```

- BeautifulSoup로 태그 기준 parsing하기

```
bs_obj = BeautifulSoup(html, 'html.parser')
```

- <h1>태그에 해당하는 내용 모두 찾기

```
title = bs_obj.find_all('h1')  
print(title)
```

[<h1><a href="https://ai-dev.tistory.com/">인공지능 개발의 모든 것</a></h1>, <h1>크롤링의 세계에 오신 것을 환영합니다. </h1>]

- 글 제목에 해당하는 내용 찾기

```
print(title[1].text)
```

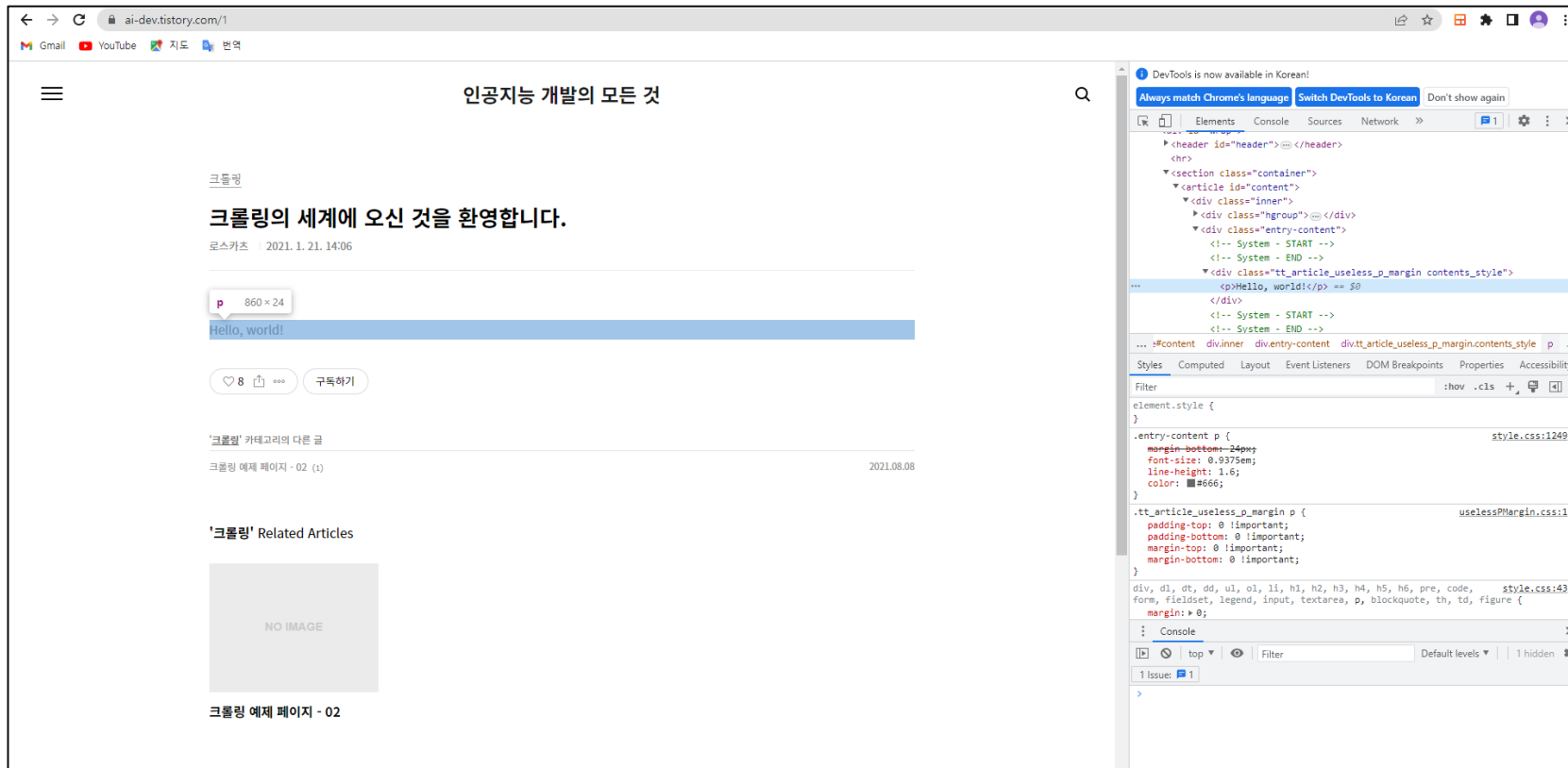
크롤링의 세계에 오신 것을 환영합니다.



# 실제 웹페이지 Scraping하기

## • 예제 4.2

– 본문을 추출해봅시다.



<p> 태그에  
해당합니다.

(Source: <https://ai-dev.tistory.com/1>)

# 실제 웹사이트 Scraping하기

## • 예제 4.2

- 웹사이트 열기 (urllib 라이브러리 사용)

```
from urllib.request import urlopen
```

```
url = 'https://ai-dev.tistory.com/1'
```

```
html = urlopen(url)
```

- BeautifulSoup로 태그 기준 parsing하기

```
bs_obj = BeautifulSoup(html, 'html.parser')
```

- <p>태그에 해당하는 내용 모두 찾기

```
contents = bs_obj.find_all('p')  
print(contents)
```

```
[<p>POWERED BY TISTORY</p>, <p>Hello, world!</p>, <p class="copyright">DESIGN BY <a href="#">TISTORY</a> <a class="admin" href="https://ai-dev.tistory.com/manage">관리자</a></p>]
```

- 글 본문에 해당하는 내용 찾기

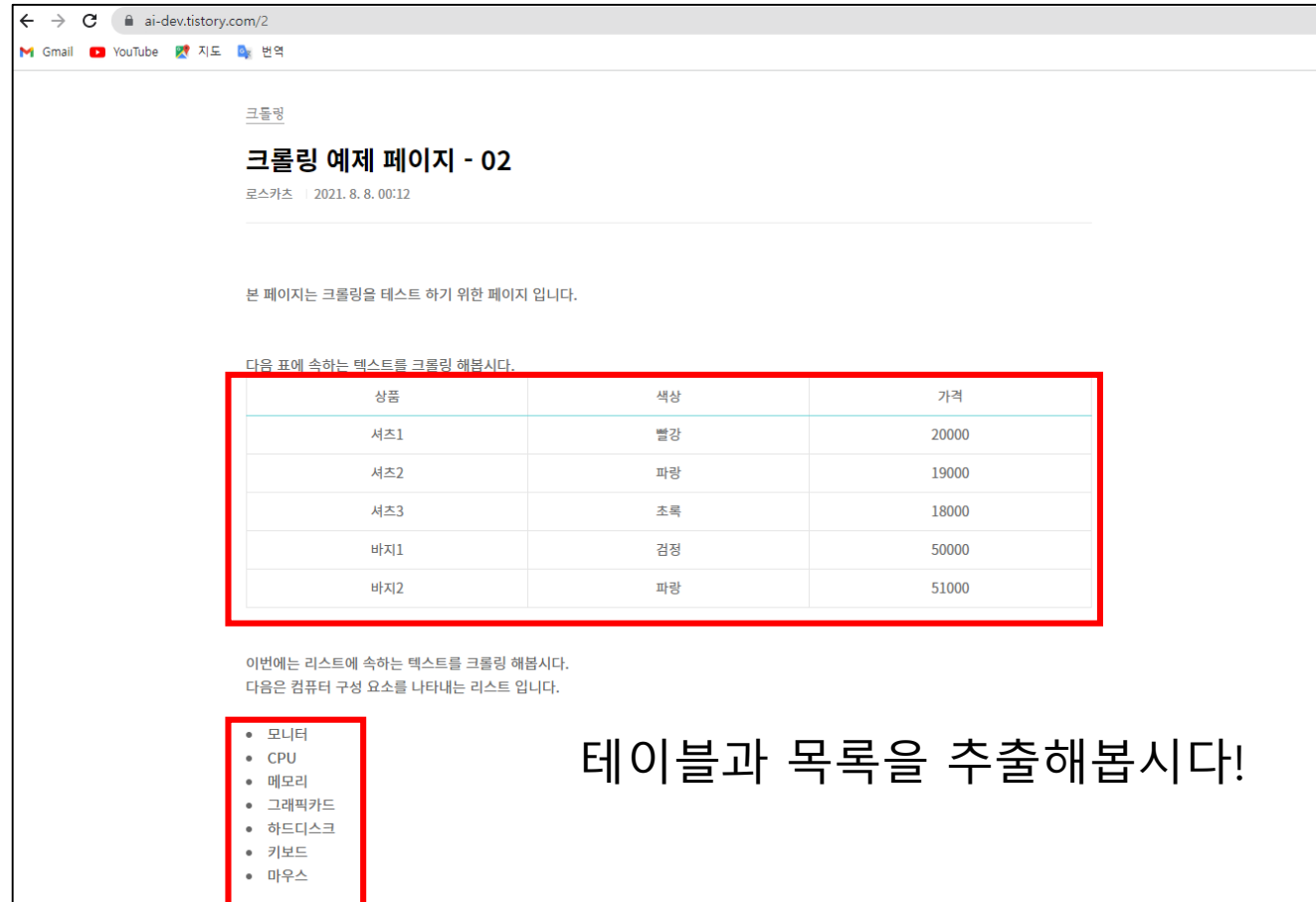
```
print(contents[1].text)
```

```
Hello, world!
```

# 실제 웹페이지 Scraping하기

## • 예제 5

– <https://ai-dev.tistory.com/2>



The screenshot shows a web browser window with the URL [ai-dev.tistory.com/2](https://ai-dev.tistory.com/2). The page title is "크롤링 예제 페이지 - 02" and the author is "로스카츠" with a timestamp of "2021. 8. 8. 00:12". The page content includes a paragraph stating it is for testing crawling, followed by a table of products. The table has three columns: "상품명" (Product Name), "색상" (Color), and "가격" (Price). The table contains six rows of data. Below the table, there is another paragraph and a list of computer components. The table and the list are highlighted with red boxes in the original image.

상품명	색상	가격
셔츠1	빨강	20000
셔츠2	파랑	19000
셔츠3	초록	18000
바지1	검정	50000
바지2	파랑	51000

이번에는 리스트에 속하는 텍스트를 크롤링 해봅시다.  
다음은 컴퓨터 구성 요소를 나타내는 리스트입니다.

- 모니터
- CPU
- 메모리
- 그래픽카드
- 하드디스크
- 키보드
- 마우스

테이블과 목록을 추출해봅시다!

(Source: <https://ai-dev.tistory.com/2>)

# 실제 웹페이지 Scraping하기

## • 예제 5.1

– 테이블내의 텍스트 정보를 추출해봅시다.

- 방법1: <table>태그를 활용한 접근

The screenshot shows a web browser at the URL `ai-dev.tistory.com/2`. The page content includes a title '크롤링 예제 페이지 - 02', a date '로스카츠 | 2021. 8. 8. 00:12', and a paragraph stating '본 페이지는 크롤링을 테스트 하기 위한 페이지 입니다.' Below this is a table with 3 columns: '상품명', '색상', and '가격'. The table contains 5 rows of data. A tooltip over the table indicates its dimensions as 'table 860 x 234.91'. Below the table, there is another paragraph and a list of computer components.

상품명	색상	가격
서조1	빨강	20000
서조2	파랑	19000
서조3	초록	18000
바지1	검정	50000
바지2	파랑	51000

이번에는 리스트에 속하는 텍스트를 크롤링 해봅시다.  
다음은 컴퓨터 구성 요소를 나타내는 리스트 입니다.

- 모니터
- CPU
- 메모리
- 그래픽카드
- 하드디스크
- 키보드
- 마우스

The right side of the image shows the Chrome DevTools console with the HTML source code of the table. The table element is highlighted, and its HTML structure is visible:

```
<table style="border-collapse: collapse; width: 100%; border="
  "1" data-ke-align="alignLeft" data-ke-style="style1"> <tbody>
  <tr>
  <tr>
  <tr>
  <tr>
  <tr>
  </tbody>
</table>
```

<table>태그에  
해당합니다.

(Source: <https://ai-dev.tistory.com/2>)

# 실제 웹사이트 Scraping하기

## • 예제 5.1 – 방법(1)

- 웹사이트 열기 (urllib 라이브러리 사용)

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

```
url = 'https://ai-dev.tistory.com/2'  
html = urlopen(url)
```

- BeautifulSoup로 태그 기준 parsing하기

```
bs_obj = BeautifulSoup(html, 'html.parser')
```

- <table>태그에 해당하는 내용 모두 찾기

```
table = bs_obj.find_all('table')
```

- 원하는 내용 찾기

```
table[0]
```

```
<table border="1" data-ke-align="alignLeft" data-ke-style="style1" style="border-collapse: collapse; width: 100%;">  
<tbody>  
<tr>  
<td style="width: 33.3333%; text-align: center;">상품</td>  
<td style="width: 33.3333%; text-align: center;">색상</td>  
<td style="width: 33.3333%; text-align: center;">가격</td>  
</tr>
```

# 실제 웹페이지 Scraping하기

## • 예제 5.1 – 방법(1)

– 원하는 내용에서 <td>태그 내용 찾기

```
td = table[0].find_all('td')

td

[<td style="width: 33.3333%; text-align: center;">상품</td>,
 <td style="width: 33.3333%; text-align: center;">색상</td>,
 <td style="width: 33.3333%; text-align: center;">가격</td>,
 <td style="width: 33.3333%; text-align: center;">셔츠1</td>,
 <td style="width: 33.3333%; text-align: center;">빨강</td>,
 <td style="width: 33.3333%; text-align: center;">20000</td>,
 <td style="width: 33.3333%; text-align: center;">셔츠2</td>,
 <td style="width: 33.3333%; text-align: center;">파랑</td>,
 <td style="width: 33.3333%; text-align: center;">19000</td>,
 <td style="width: 33.3333%; text-align: center;">셔츠3</td>,
 <td style="width: 33.3333%; text-align: center;">초록</td>,
 <td style="width: 33.3333%; text-align: center;">18000</td>,
 <td style="width: 33.3333%; text-align: center;">바지1</td>,
 <td style="width: 33.3333%; text-align: center;">검정</td>,
 <td style="width: 33.3333%; text-align: center;">50000</td>,
 <td style="width: 33.3333%; text-align: center;">바지2</td>,
 <td style="width: 33.3333%; text-align: center;">파랑</td>,
 <td style="width: 33.3333%; text-align: center;">51000</td>]

for x in td:
    print(x.text)

상품
색상
가격
셔츠1
빨강
20000
셔츠2
파랑
19000
셔츠3
초록
18000
바지1
검정
50000
바지2
파랑
51000
```

# 실제 웹페이지 Scraping하기

## • 예제 5.1

– 테이블내의 텍스트 정보를 추출해봅시다.

- 방법2: <td>태그를 활용한 접근

The screenshot shows a web browser window with the URL [ai-dev.tistory.com/2](https://ai-dev.tistory.com/2). The page content includes a title "크롤링 예제 페이지 - 02", a date "2021. 8. 8. 00:12", and a table with 3 columns: "상품명", "색상", and "가격". The table contains 6 rows of data. Below the table, there is a list of computer components: 모니터, CPU, 메모리, 그래픽카드, 하드디스크, 키보드, and 마우스. The Chrome DevTools Elements panel is open on the right, showing the HTML structure of the page. A table element is selected, and its attributes and styles are visible in the right-hand pane. The table has a border-collapse: collapse and width: 100%. The tbody contains 6 rows, each with 3 td elements. The first row is highlighted in blue.

상품명	색상	가격
서즈1	빨강	20000
서즈2	파랑	19000
서즈3	초록	18000
바지1	검정	50000
바지2	파랑	51000

<td>태그에  
해당합니다.

(Source: <https://ai-dev.tistory.com/2>)

# 실제 웹사이트 Scraping하기

## • 예제 5.1 – 방법(2)

- 웹사이트 열기 (urllib 라이브러리 사용)

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

```
url = 'https://ai-dev.tistory.com/2'
html = urlopen(url)
```

- BeautifulSoup로 태그 기준 parsing하기

```
bs_obj = BeautifulSoup(html, 'html.parser')
```

- <td>태그에 해당하는 내용 모두 찾기

```
td = bs_obj.find_all('td')
```

td

```
[<td style="width: 33.3333%; text-align: center;">상품</td>,
<td style="width: 33.3333%; text-align: center;">색상</td>,
<td style="width: 33.3333%; text-align: center;">가격</td>,
<td style="width: 33.3333%; text-align: center;">셔츠1</td>,
<td style="width: 33.3333%; text-align: center;">빨강</td>,
<td style="width: 33.3333%; text-align: center;">20000</td>,
<td style="width: 33.3333%; text-align: center;">셔츠2</td>,
<td style="width: 33.3333%; text-align: center;">파랑</td>]
```

```
<td style="width: 33.3333%; text-align: center;">파랑</td>,
<td style="width: 33.3333%; text-align: center;">19000</td>,
<td style="width: 33.3333%; text-align: center;">셔츠3</td>,
<td style="width: 33.3333%; text-align: center;">초록</td>,
<td style="width: 33.3333%; text-align: center;">18000</td>,
<td style="width: 33.3333%; text-align: center;">바지1</td>,
<td style="width: 33.3333%; text-align: center;">검정</td>,
<td style="width: 33.3333%; text-align: center;">50000</td>,
<td style="width: 33.3333%; text-align: center;">바지2</td>,
<td style="width: 33.3333%; text-align: center;">파랑</td>,
<td style="width: 33.3333%; text-align: center;">51000</td>,
<td>2021.01.21</td>]
```



# 실제 웹페이지 Scraping하기

## • 예제 5.1 – 방법(2)

– <td>태그의 속성까지 지정하여 자세히 찾기

```
td = bs_obj.find_all('td', {"style": "width: 33.3333%; text-align: center;"})
```

```
td
```

```
[<td style="width: 33.3333%; text-align: center;">상품</td>,
<td style="width: 33.3333%; text-align: center;">색상</td>,
<td style="width: 33.3333%; text-align: center;">가격</td>,
<td style="width: 33.3333%; text-align: center;">셔츠1</td>,
<td style="width: 33.3333%; text-align: center;">빨강</td>,
<td style="width: 33.3333%; text-align: center;">20000</td>,
<td style="width: 33.3333%; text-align: center;">셔츠2</td>,
<td style="width: 33.3333%; text-align: center;">파랑</td>,
<td style="width: 33.3333%; text-align: center;">19000</td>,
<td style="width: 33.3333%; text-align: center;">셔츠3</td>,
<td style="width: 33.3333%; text-align: center;">초록</td>,
<td style="width: 33.3333%; text-align: center;">18000</td>,
<td style="width: 33.3333%; text-align: center;">바지1</td>,
<td style="width: 33.3333%; text-align: center;">검정</td>,
<td style="width: 33.3333%; text-align: center;">50000</td>,
<td style="width: 33.3333%; text-align: center;">바지2</td>,
<td style="width: 33.3333%; text-align: center;">파랑</td>,
<td style="width: 33.3333%; text-align: center;">51000</td>]
```

```
for x in td:
    print(x.text)
```

```
상품
색상
가격
셔츠1
빨강
20000
셔츠2
파랑
19000
셔츠3
초록
18000
바지1
검정
50000
바지2
파랑
51000
```

# 실제 웹페이지 Scraping하기

## • 예제 5.2

– 목록 정보를 추출해봅시다.

- <ul>태그, <li>태그 를 활용한 접근

크롤링

### 크롤링 예제 페이지 - 02

로스카츠 2021. 8. 8. 00:12

본 페이지는 크롤링을 테스트 하기 위한 페이지 입니다.

다음 표에 속하는 텍스트를 크롤링 해봅시다.

상품명	색상	가격
서초1	빨강	20000
서초2	파랑	19000
서초3	초록	18000
바지1	검정	50000
바지2	파랑	51000

이번에는 리스트에 속하는 텍스트를 크롤링 해봅시다.

다음은 컴퓨터 구성 요소를 나타내는 리스트 입니다.

- 모니터
- CPU
- 메모리
- 그래픽카드
- 하드디스크
- 키보드
- 마우스

DevTools is now available in Korean!

Always match Chrome's language Switch DevTools to Korean Don't show again

Elements Console Sources Network

```
<p data-ke-size="size16">다음은 컴퓨터 구성 요소를 나타내는 리스트  
입니다.</p>  
<p data-ke-size="size16">&nbsp;</p>  
<ul style="list-style-type: disc;" data-ke-list-type="disc">  
  <li>  
    ::marker  
    "모니터"  
  </li>  
  <li></li>  
  <li></li>  
  <li></li>  
  <li></li>  
  <li></li>  
  <li></li>  
</ul>
```

Filter

element.style {  
 border-collapse: collapse;  
 width: 100%;  
}

table[data-ke-style], #tt-body-page table[data-ke-style] {  
 margin-bottom: 20px;  
}

.entry-content table {  
 width: 100%;  
 margin-bottom: 20px;  
 border: 1px solid #e6e6e6;  
 border-collapse: collapse;  
 font-size: 0.875em;  
 line-height: 1.5714;  
 color: #666;  
}

table[Attributes Style] {

Console

1 issue

<ul>태그에  
해당합니다.

(Source: <https://ai-dev.tistory.com/2>)

# 실제 웹페이지 Scraping하기

## • 예제 5.2

- 웹페이지 열기 (urllib 라이브러리 사용)

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

```
url = 'https://ai-dev.tistory.com/2'  
html = urlopen(url)
```

- BeautifulSoup로 태그 기준 parsing하기

```
bs_obj = BeautifulSoup(html, 'html.parser')
```

# 실제 웹페이지 Scraping하기

## • 예제 5.2

– 목록 정보를 추출해봅시다.

- <ul>태그, <li>태그 를 활용한 접근

The screenshot shows a web browser window with the URL `ai-dev.tistory.com/2`. The page content includes a title '크롤링 예제 페이지 - 02', a date '2021. 8. 8. 00:12', and a table of products. Below the table, there is a list of computer components. The right side of the image shows the Chrome DevTools console with the following HTML and CSS code:

```
<p data-ke-size="size16">다음은 컴퓨터 구성 요소를 나타내는 리스트입니다.</p>
<p data-ke-size="size16">&nbsp;</p>
<ul style="list-style-type: disc;" data-ke-list-type="disc">
  <li>::marker<br>모니터</li>
  <li>::marker<br>CPU</li>
  <li>::marker<br>하드디스크</li>
  <li>::marker<br>키보드</li>
  <li>::marker<br>마우스</li>
</ul>
```

The CSS code shows the styling for the list items, including padding, font size, line height, color, and list style type.

<li>태그에  
해당합니다.

(Source: <https://ai-dev.tistory.com/2>)

# 실제 웹페이지 Scraping하기

## • 예제 5.2

- <ul>태그에 속성을 부여하여 자세히 찾기

```
ul = bs_obj.find_all('ul', {'data-ke-list-type': 'disc',  
                             'style': 'list-style-type: disc;'})  
  
ul  
  
[<ul data-ke-list-type="disc" style="list-style-type: disc;">  
  <li>모니터</li>  
  <li>CPU</li>  
  <li>메모리</li>  
  <li>그래픽카드</li>  
  <li>하드디스크</li>  
  <li>키보드</li>  
  <li>마우스</li>  
</ul>]
```

- <li>태그로 텍스트만 추출하기

```
li = ul[0].find_all('li')  
for x in li:  
    print(x.text)  
  
모니터  
CPU  
메모리  
그래픽카드  
하드디스크  
키보드  
마우스
```

# End of the documents

---