Web Crawling

웹 크롤링(Web Crawling)

- 스크레이핑(Scraping)
  - 웹 사이트의 특정 정보를 추출하는 것으로 웹 데이터의 구조 분석이 필요
  - 로그인이 필요한 경우가 많다

- 크롤링(Crawling)
  - 프로그램으로 자동으로 (보통 정기적으로) 웹 사이트를 돌며 정보를 추출
  - 이러한 작업을 수행하는 프로그램을 크롤러, 스파이더 라고 한다.

Web의 특징

- **Web의 특징**
  - ✓ 전 세계의 컴퓨터를 연결
  - ✓ HTTP(Hyper Text Transfer Protocol) 프로토콜 사용
  - ✓ 텍스트, 그래픽, 오디오, 비디오, 프로그램 파일 등 멀티미디어 서비스 제공

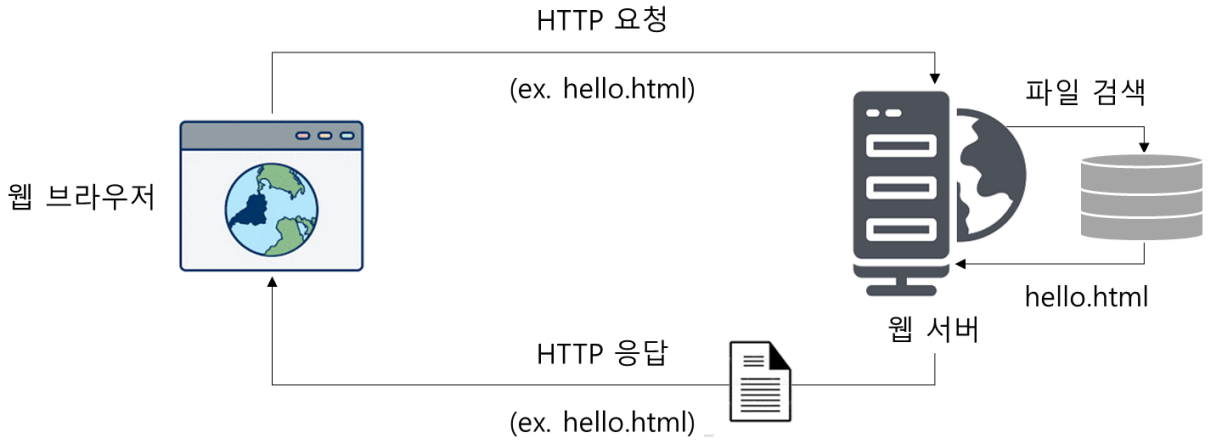
Web 표준

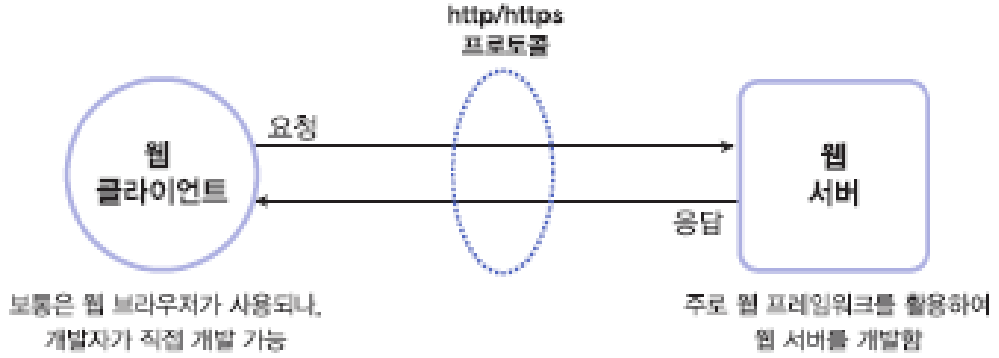
- Web **표준**
  - ✓ **개발자 간 일종의 약속**
  - ✓ **누가 개발하든 정해진 규칙을 준수하면 모두 호환되어 편리하게 사용**

- W3C (http://www.w3.org)
  - ✓ Web**에 관련된 기술과** Web browser **사용을 위한 표준안 제정**
  - ✓ Web **개발자나 사용자 간의 정보 공유 및 신기술 개발 등에 기여**

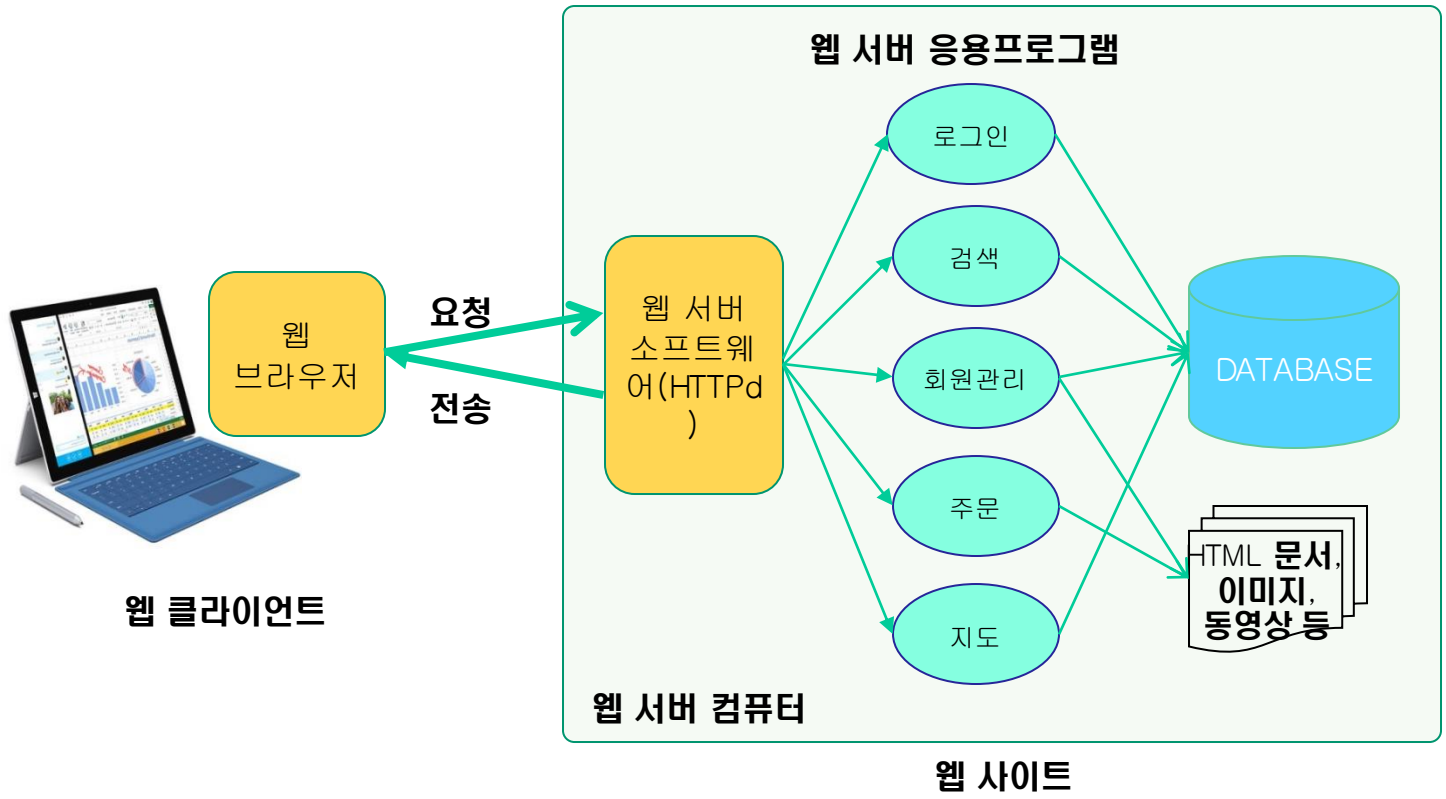
웹 통신의 이해

- Client-Server **방식**
  - ✓ HTTP(S) **프로토콜**



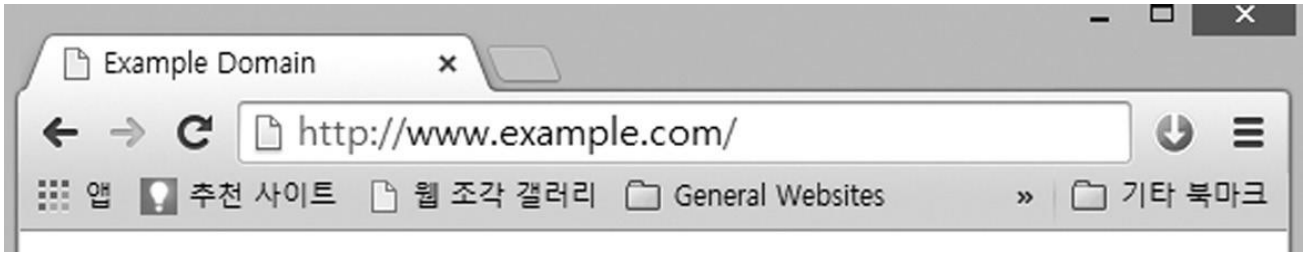


웹 통신의 이해


웹 서버 응용프로그램
로그인
검색
회원관리
주문
지도
웹 서버 소프트웨어(HTTPd)
DATABASE
HTML 문서, 이미지, 동영상 등
웹 서버 컴퓨터
웹 사이트
요청
전송
웹 브라우저
웹 클라이언트

다양한 웹 Client

- Web Browser



- **리눅스** curl **명령어**
  - ✓ HTTP/HTTPS/FTP **등 여러 가지의 프로토콜을 사용하여 데이터를 송수신**

```
$ curl  http://www.example.com
```

다양한 웹 Client

- Telnet **명령어**
  - ✓ Terminal **창에서 입력하는 내용을 그대로 웹 서버에 전송**

```
[shkim@localhost ch1]$ telnet www.example.com 80
Trying 93.184.216.119...
Connected to www.example.com.
Escape character is '^]'.
GET / HTTP/1.1  Enter 입력
Host: www.example.com  Enter 입력
Enter 입력
```

- **직접 프로그램으로 작성**

```
C:\RedBook\ch1>notepad example.py

import urllib.request
print(urllib.request.urlopen("http://www.example.com").read().decode('utf-8'))
```

HTTP 처리 방식

메소드명	의미	CRUD와 매핑되는 역할
GET	리소스 취득	Read(조회)
POST	리소스 생성, 리소스 데이터 추가	Create(생성)
PUT	리소스 변경	Update(변경)
DELETE	리소스 삭제	Delete(삭제)
HEAD	리소스의 헤더(메타데이터) 취득	
OPTIONS	리소스가 서포트하는 메소드 취득	
TRACE	루프백 시험에 사용	
CONNECT	프록시 동작의 터널 접속으로 변경	

- GET **방식**
  - ✓ **지정한** URL**의 정보를 가져오는 메소드, 가장 많이 사용**
- POST **방식**
  - ✓ **리소스를 생성**(ex, **블로그에 글을 등록 등**)
- PUT **방식**
  - ✓ **리소스를 변경**

HTTP 처리 방식

- **상태 코드**(State code)
  - ✓ Server**에서의 처리 결과**

메소드명	의미	CRUD와 매핑되는 역할
1xx	Informational (정보 제공)	임시적인 응답으로, 현재 클라이언트의 요청까지 처리되었으니 계속 진행하라는 의미입니다. HTTP 1.1 버전부터 추가되었습니다.
2xx	Success(성공)	클라이언트의 요청이 서버에서 성공적으로 처리되었다는 의미입니다.
3xx	Redirection (리다이렉션)	완전한 처리를 위해서 추가적인 동작을 필요로 하는 경우입니다. 주로 서버의 주소 또는 요청한 URI의 웹 문서가 이동되었으니, 그 주소로 다시 시도해보라는 의미입니다.
4xx	Client Error (클라이언트 에러)	없는 페이지를 요청하는 것처럼 클라이언트의 요청 메시지 내용이 잘못된 경우입니다.
5xx	Server Error (서버 에러)	서버 측 사정에 의해서 메시지 처리에 문제가 발생한 경우입니다. 서버의 부하, DB 처리 과정 오류, 서버에서 익셉션이 발생하는 경우가 이에 해당합니다.

HTTP 처리 방식

- **자주 사용되는 상태 코드**(State code)

상태 코드	상태 텍스트	응답 문구	서버 측면에서의 의미
2xx	Success	성공	**클라이언트가 요청한 동작을 수신하여 이해했고, 승낙했으며 성공적으로 처리했다.**
200	OK	성공	**서버가 요청을 성공적으로 처리했다.**
201	Created	생성됨	**요청이 처리되어서 새로운 리소스가 생성되었다.** 응답 헤더 Location에 새로운 리소스의 절대 URI를 기록합니다.
202	Accepted	허용됨	**요청은 접수했지만 처리가 완료되지 않았다.** 클라이언트는 응답 헤더의 Location, Retry-After를 참고하여 다시 요청을 보냅니다.
3xx	Redirection	리다이렉션	**클라이언트는 요청을 마치기 위해 추가적인 동작을 취해야 한다.**
301	Moved Permanently	영구 이동	**지정한 리소스가 새로운 URI로 이동했다.** 이동할 곳의 새로운 URI는 응답 헤더 Location에 기록합니다.

HTTP 처리 방식

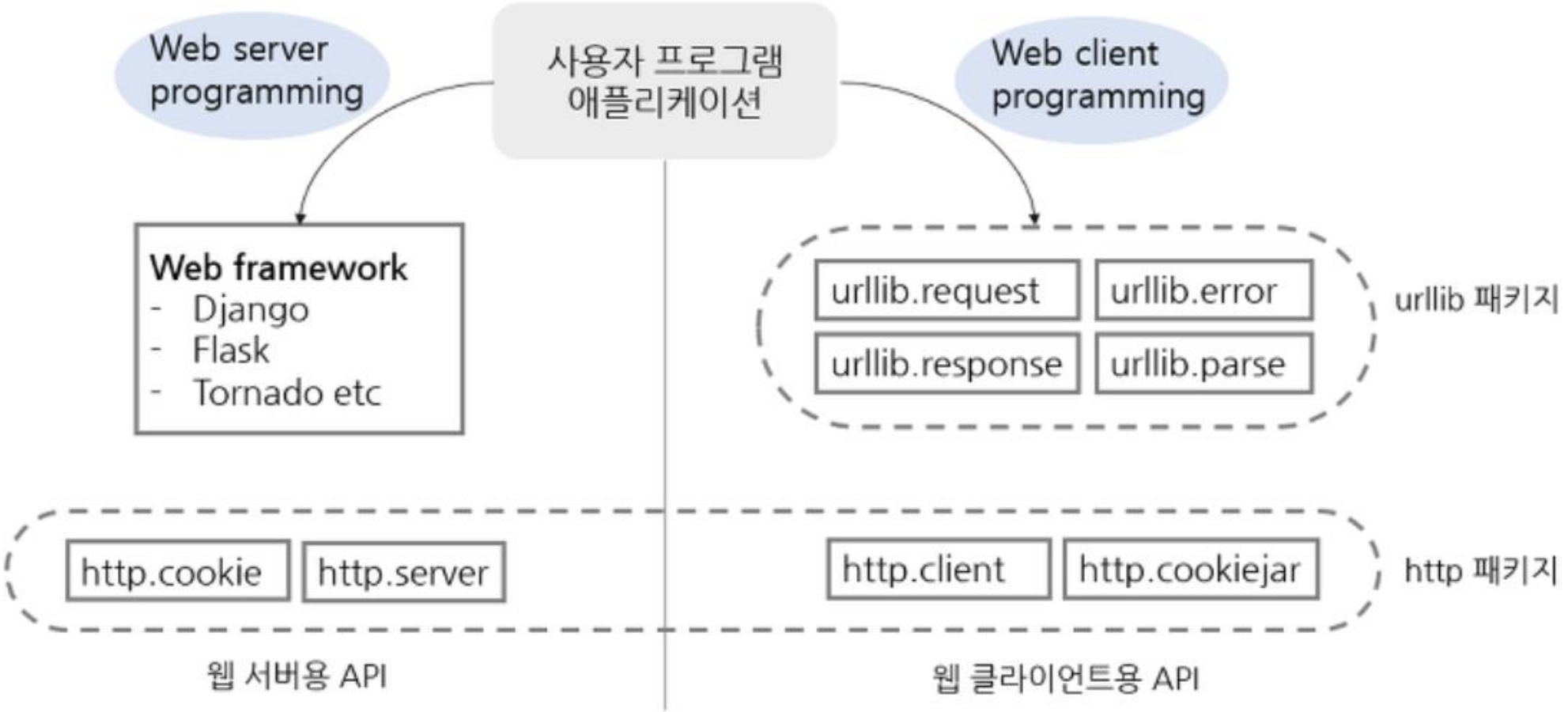
상태 코드	상태 텍스트	응답 문구	서버 측면에서의 의미
303	See Other	다른 위치 보기	**다른 위치로 요청하라.** 요청에 대한 처리 결과를 응답 헤더 Location에 표시된 URI에서 GET으로 취득할 수 있습니다. 브라우저의 폼 요청을 POST로 처리하고 그 결과 화면으로 리다이렉트시킬 때, 자주 사용하는 응답 코드입니다.
307	Temporary Redirect	임시 리다이렉션	**임시로 리다이렉션 요청이 필요하다.** 요청한 URI가 없으므로, 클라이언트는 메소드를 그대로 유지한 채 응답 헤더 Location에 표시된 다른 URI로 요청을 재송신할 필요가 있습니다. 클라이언트는 향후 요청 시 원래 위치를 계속 사용해야 합니다. 302의 의미를 정확하게 재정의해서 HTTP/1.1의 307 응답으로 추가되었습니다.
4xx	Client Error	클라이언트 에러	**클라이언트의 요청에 오류가 있다.**
400	Bad Request	잘못된 요청	**요청의 구문이 잘못되었다.** 클라이언트가 모르는 4xx 계열의 응답 코드가 반환된 경우에도 클라이언트는 400과 동일하게 처리하도록 규정하고 있습니다.
401	Unauthorized	권한 없음	**지정한 리소스에 대한 액세스 권한이 없다.** 응답 헤더 WWW-Authenticate에 필요한 인증 방식을 지정합니다.
403	Forbidden	금지됨	**지정한 리소스에 대한 액세스가 금지되었다.** 401 인증 처리 이외의 사유로 리소스에 대한 액세스가 금지되었음을 의미합니다. 리소스의 존재 자체를 은폐하고 싶은 경우는 404 응답 코드를 사용할 수 있습니다.
404	Not Found	찾을 수 없음	**지정한 리소스를 찾을 수 없다.**
5xx	Server Error	서버 에러	**클라이언트의 요청은 유효한데, 서버가 처리에 실패했다.**
500	Internal Server Error	내부 서버 오류	**서버쪽에서 에러가 발생했다.** 클라이언트가 모르는 5xx 계열의 응답 코드가 반환된 경우에도 클라이언트는 500과 동일하게 처리하도록 규정하고 있습니다.
502	Bad Gateway	불량 게이트웨이	**게이트웨이 또는 프록시 역할을 하는 서버가 그 뒷단의 서버로부터 잘못된 응답을 받았다.**
503	Service Unavailable	서비스 제공불가	**현재 서버에서 서비스를 제공할 수 없다.** 보통은 서버의 과부하나 서비스 점검 등 일시적인 상태입니다.

URL(Uniform resource locator)

- **URL**
  - **네트워크 상에서 자원이 어디 있는 지 알려주기 위한 규약**

- URL 스킴 : URL**에 사용된 프로토콜**
- **호스트명** : **웹 서버의 호스트명으로, 도메인명 또는** IP **주소로 표현**
- **포트번호** : **웹 서버 내의 서비스 포트번호**.
  - **생략 시에는 디폴트 포트번호로**, http**는** 80**을**, https**는** 443**을 사용**
- **경로** : **파일이나 애플리케이션 경로를 의미**
- **쿼리스트링** : **질의 문자열로, 앰퍼샌드**(&)**로 구분된 이름=값 쌍 형식으로 표현**
- **프라그먼트** : **문서 내의 앵커 등 조각을 지정**

Web 표준 library



urlib.request 모듈

- urlib.request 모듈
  - 주어진 URL에서 데이터를 가져오는 기본 기능을 제공
  - 대표적인 함수, urlopen()
  - urlopen() 함수
    - Default 요청 방식 : Get 방식, (서버에 전달할 parameter, URL에 포함)
    - 요청 방식을 Post로 보내고 싶으면 data 인자에 질의 문자열을 지정
    - timeout 옵션 : 응답을 기다리는 time out 시간(초 단위)

```
urlopen(url,data=None,[timeout])
```

urlib.request 모듈

```
>>> from urllib.request import urlopen
>>> f = urlopen("https://www.example.com")
>>> print(f.read(500).decode('utf-8'))
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica,
 Arial, sans-serif;
```

urlib.parse 모듈

- URL parse 모듈
  - URL의 분해 조립 변경 및 URL 문자 인코딩 디코딩 등을 처리하는 함수를 제공
  - 대표적인 함수, urlparse()
  - urlparse() 함수는 URL 파싱한 결과로 ParseResult 인스턴스를 반환
  - ParseResult 클래스 각 속성
    - scheme : URL에 사용된 프로토콜
    - netloc : 네트워크 위치, user:password@host:port 형식으로 표현
    - path : 파일이나 애플리케이션 경로
    - params : 애플리케이션에 전달될 매개변수 (현재는 사용 X)
    - query : 질의 문자열 또는 매개변수로 key-value 로 구성
    - fragment : 문서 내의 특정 부분을 지정

urlib.parse 모듈

```
>>> python
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'python' is not defined
>>> from urllib.parse import urlparse
>>> result = urlparse("http://www.python.org:80/guido/python.html;philosophy?overall=3#n10")
>>> result
ParseResult(scheme='http', netloc='www.python.org:80', path='/guido/python.html', params='philosop
hy', query='overall=3', fragment='n10')
>>>
```

Markup Language

- **Markup Language**
  - Web에서 사용되는 문서가 어떻게 구조화되는가를 나타내는 언어

- **유형**
  - ✓ 구조적 마크업 : 문서의 구성 방식을 표현한 것
  - ✓ 유형적 마크업 : 문서를 시각적으로 표현하는 방법
  - ✓ 의미적 마크업 : 데이터 내용 자체에 관한 것

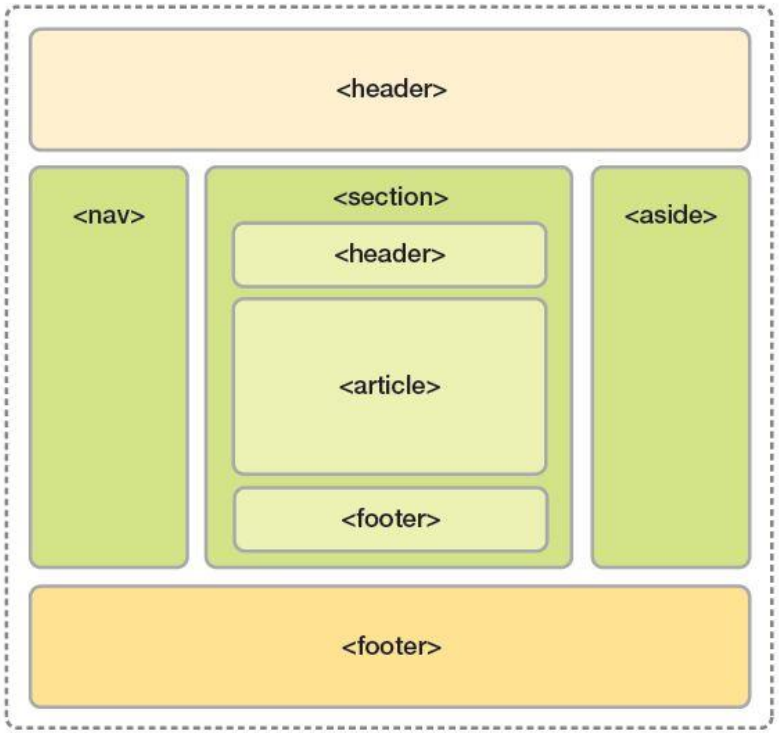
HTML(HyperText Markup Language)

- HTML
  - web page & web application 들을 생성하기 위한 표준 Markup 언어
  - WWW 를 위한 주요 기술(with CSS & Javascript)
    * CSS : Cascading STYLE sheets)
  - Web page의 구조를 semantically 설명 & 문서의 외형에 대한 표현을 포함
  - W3C(World Wide Web Consortium)은 1997년 이후 CSS의 사용을 권장

HTML5

- HTML5 특징
  - ✓ 구조적 설계 지원
  - ✓ 그래픽 및 멀티미디어 기능 강화
  - ✓ CSS3 지원
  - ✓ 자바스크립트 지원
  - ✓ 다양한 API 제공
  - ✓ 모바일 웹 지원

HTML5의 구조적 설계



<header>
- HTML5 문서의 머리말 영역, 중요한 정보를 상단에 표시
  (예 : 사이트의 제목, 로고 등)

<nav>
- 내비게이션(navigation) 영역
- 웹 사이트 내에 분류된 다른 영역으로 이동할 때 사용

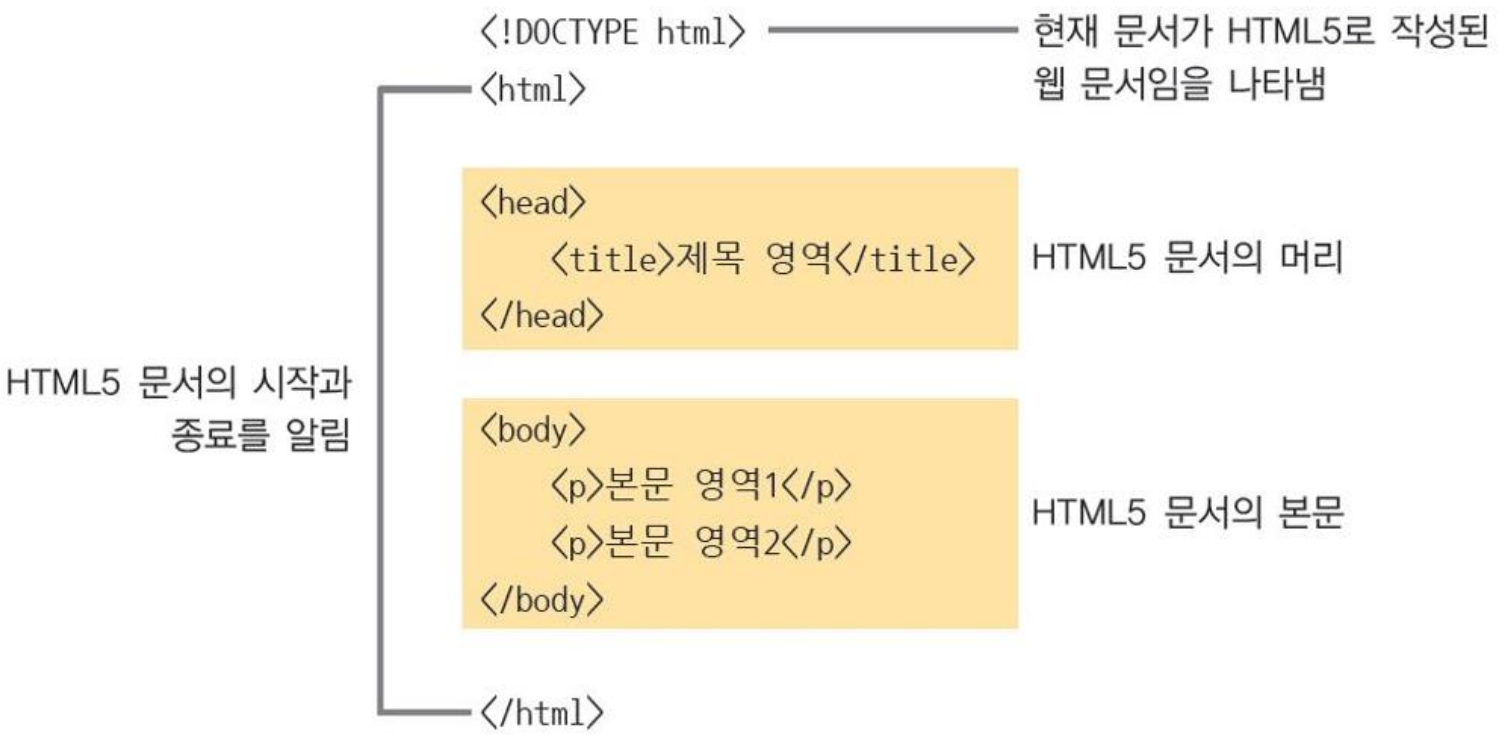
<section>
- 문서의 영역을 구성
- <header>, <article> 태그 등을 포함할 수 있음

<article>
- 독립된 주요 콘텐츠 영역을 정의
- 하나의 <section> 태그 내에 여러 개의 <article> 태그를 구성

<aside>
- 주요 콘텐츠 이외에 남은 콘텐츠를 표시
  (예 : 사이드 바(sidebar) 등).

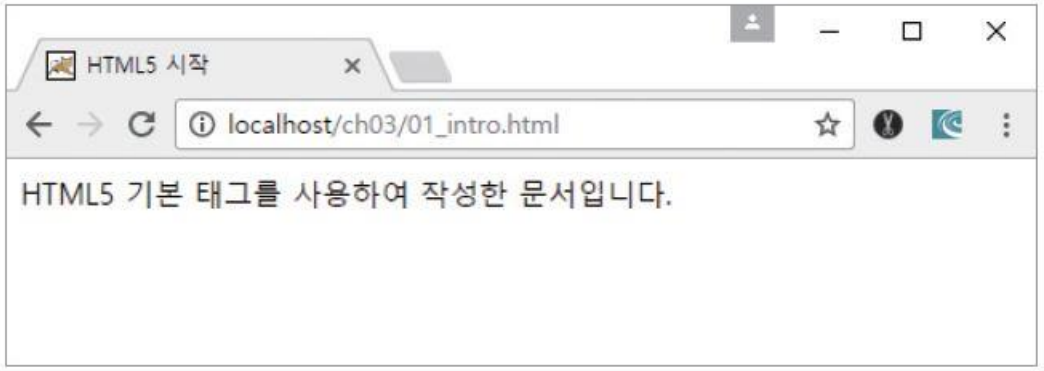
<footer>
- 사이트의 자세한 정보를 표시
  (예 : 저작권 정보, 관리자 정보, 회사 정보 등)

HTML5의 문서 구조


<!DOCTYPE html>
현재 문서가 HTML5로 작성된 웹 문서임을 나타냄
<html>
<head>
<title>제목 영역</title>
</head>
HTML5 문서의 머리
HTML5 문서의 시작과 종료를 알림
<body>
본문 영역1
본문 영역2
</body>
HTML5 문서의 본문
</html>

HTML5의 문서 구조

```
<!DOCTYPE html>
<html>
<head>
    <title>HTML5 시작</title>
</head>
<body>
    HTML5 기본 태그를 사용하여 작성한 문서입니다.
</body>
</html>
```



HTML

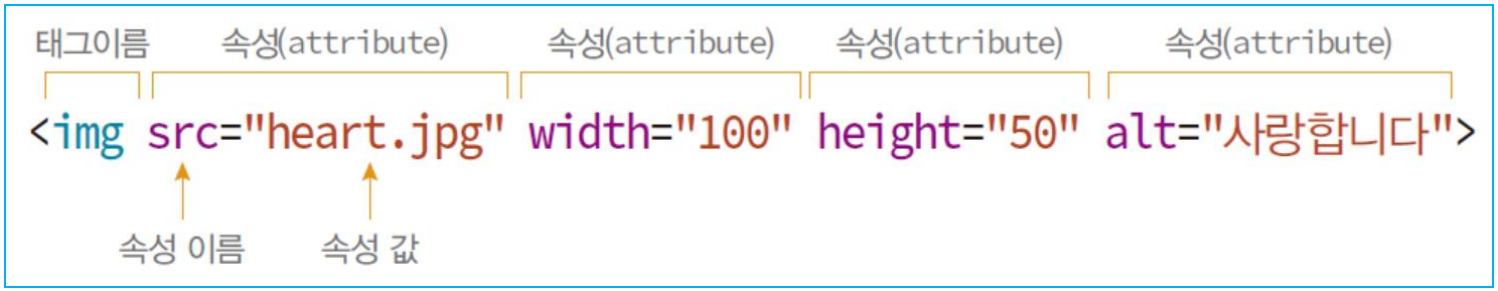
❖ Elements

- Building blocks of HTML pages
- (ex) <head></head>: 시작을 알리는 태그, <title> 은 <head> 안에서 제목 부여.
- <body></body>: 실제 웹검색기 화면에 출력되는 부분
- <h1></h1> : 제목 텍스트의 크기 (<h1> ~ <h6>)
-   : paragraphs (단락)
- Tags such as <img /> and <input /> directly introduce content into the page.
- A link to Wikipedia! : create a link

❖ **Attributes** of an element

- id: document-wide unique identifier for an element
- class: classifying similar elements
- style: assign presentational properties
- title: attach subtextual explanation to an element
- lang: identifies a natural language to an element

HTML Tag의 구성


태그이름
속성(attribute)
속성(attribute)
속성(attribute)
속성(attribute)
<img src="heart.jpg" width="100" height="50" alt="사랑합니다">
속성 이름
속성 값

HTML Tags

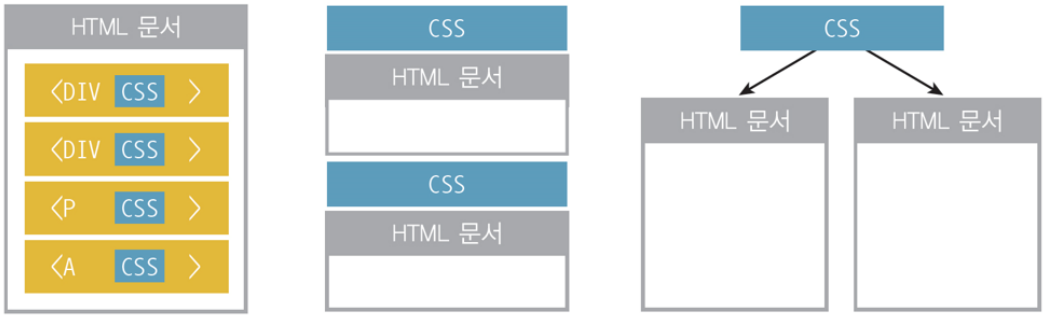
- **Some of the Basic Tags**
  - Headings: <h1><h2><h3>
  - Paragraphs: <p>
  - Images: <img src="logo.jpg" alt="site_logo" width="100" height="100"> ; alternative text, size (width and height)
  - Links: Here
  - Tables: <table border="1"cellpadding="5" cellspacing="5">
  - <tr> <td>One</td><td>Two</td> </tr> ; row and data </table>
  - Divisions: This is a DIV container ; define a section in a document (you can think of it like a container or a building block.)
  - Lists: ; defines unordered list, ordered list, a list item
  - Line breaks:
  - Bold text: …
  - Italic text: …

CSS (Cascading Style Sheets)

- **CSS**
  - HTML element들이 screen, paper 또는 다른 media 상에서 어떻게 display 되는 지를 설명
  - Presentation & content를 분리 가능하게 함(layout, color, font 포함)
  - Content accessibility를 개선
  - More flexibility and control 제공
  - Multiple web page들이 formatting을 공유할 수 있게 해 준다 (별도의 .css 파일 내에 해당 CSS 들을 명시함으로써)

CSS (Cascading Style Sheets)

- **HTML 내에서의 CSS를 추가하는 3가지 방법**
  - Inline
    - HTML element들 내에 style attribute을 사용
    - (ex) <h1 style="color:blue;">This is a Blue Heading</h1>
  - Internal
    - <head> section 내에 <style> element를 사용
  - External
    - 외부 CSS 파일을 사용 (가장 보편적인 방법)

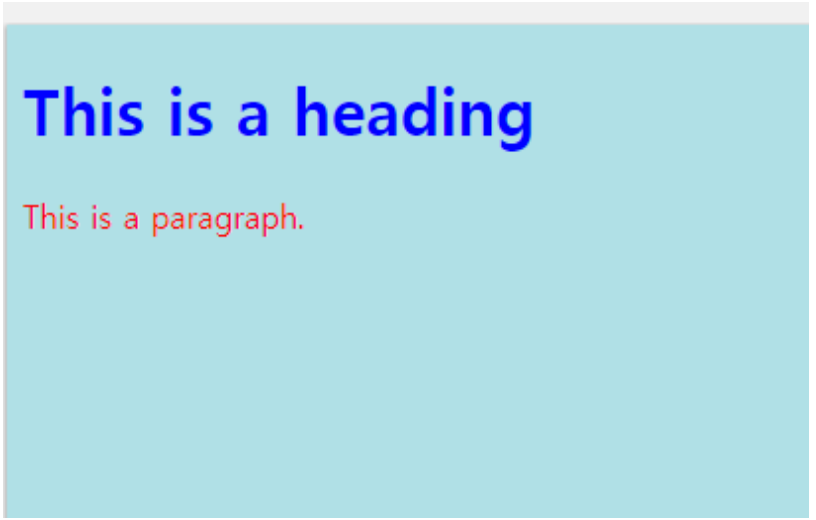


인라인 스타일 시트 내부 스타일 시트 외부 스타일 시트

CSS 정의 문법

Internal CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



External CSS

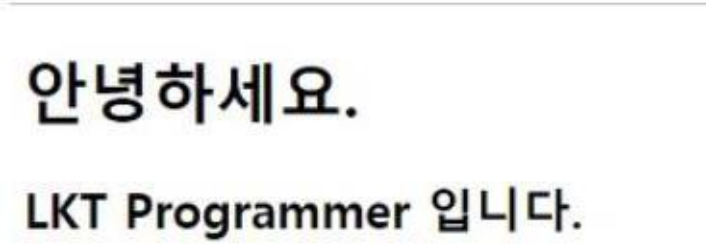
- 하나의 외부 style sheet로 많은 HTML page들에 대한 style을 정의하는데 사용
- 하나의 외부 style sheet로 전체 web site의 look을 변화시킬 수 있다 (단 하나의 파일만을 변경함으로써)
- 외부 style sheet을 사용 : HTML page의 <body> section 내에 추가

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

```css
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

Example(HTML only)

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h1>
        안녕하세요.
    </h1>
    <h2>
        LKT Programmer 입니다.
    </h2>
</body>
</html>
```

Example(CSS)

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>

    <style>
        h1 {
            background-color : red;
        }
        h2 {
            background-color : green;
        }
    </style>

</head>
<body>
    <h1>
        안녕하세요.
    </h1>
    <h2>
        LKT Programmer 입니다.
    </h2>
</body>
</html>
```


CSS Tags (attributes)

- CSS Fonts : color, font-family, font-size
- CSS Border : bordor
- CSS Padding : padding
  - text와 border 사이의 padding(space) 정의
- CSS Margin : margin
  - border 외부의 마진 정의
- id attribute : 특정 id와 관련된 style 정의
  - I am different
- Class attribute : 특정 class에 대한 style 정의
  - I am different
- External reference:

<link rel="stylesheet" href=https://www.w3schools.com/html/styles.css>

JavaScript (JS)

- **JavaScript**
  - ✓ Web 문서를 동적으로 제어하기 위해 고안된 프로그래밍 언어

- **Web page 구성 3 요소**
  - ✓ HTML: 모델 담당
  - ✓ CSS: 뷰(view) 담당
  - ✓ 자바스크립트: 동적 제어 담당

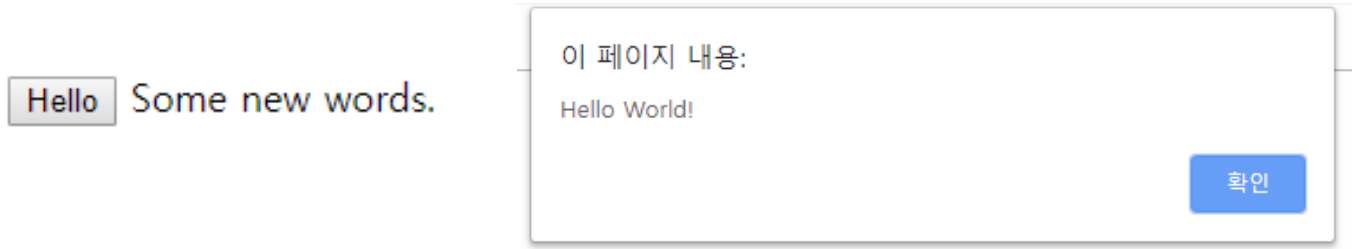
JavaScript의 역할

- 요소의 추가 및 삭제
- CSS 및 HTML 요소의 스타일 변경
- 사용자와의 상호작용
- 폼의 유효성 검증
- 마우스와 키보드 이벤트에 대한 스크립트 실행
- 웹 브라우저 제어 및 쿠키 등의 설정과 조회
- AJAX 기술을 이용한 웹 서버와의 통신

JavaScript (JS)

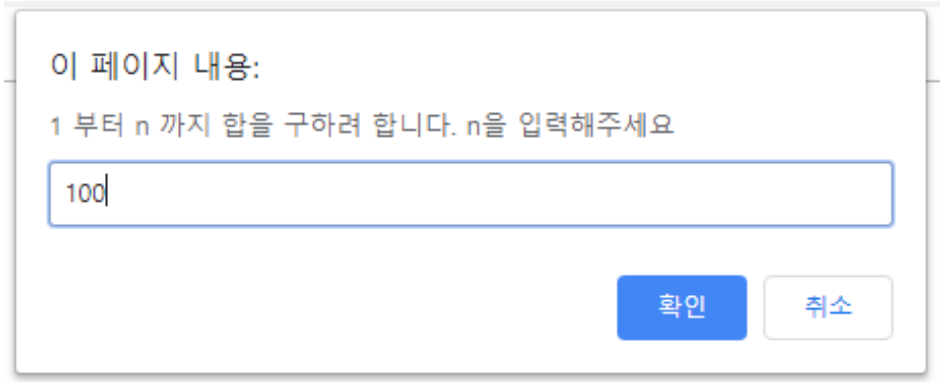
- **JavaScript(HTML5 syntax) & DOM 을 포함하는 Web page의 예**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <button id="hellobutton">Hello</button>
    <script>
        document.getElementById('hellobutton').onclick = function() {
            alert('Hello world!');                    // Show a dialog
            var myTextNode = document.createTextNode('Some new words.');
            document.body.appendChild(myTextNode);    // Append "Some new words" to the page
        };
    </script>
  </body>
</html>
```



JavaScript (JS) – Example(2)

```
<!DOCTYPE html>
<html>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <body>
    <script>
        var sum=0;
        n=parseInt(prompt("1 부터 n 까지 합을 구하려 합니다. n을 입력해주세요",""));
        for(var i=1; i<=n; i++){
        sum+=i;
        }
        document.write("1부터 " + n + "까지의 합은 " + sum + "입니다.");
    </script>
    </body>
</html>
```



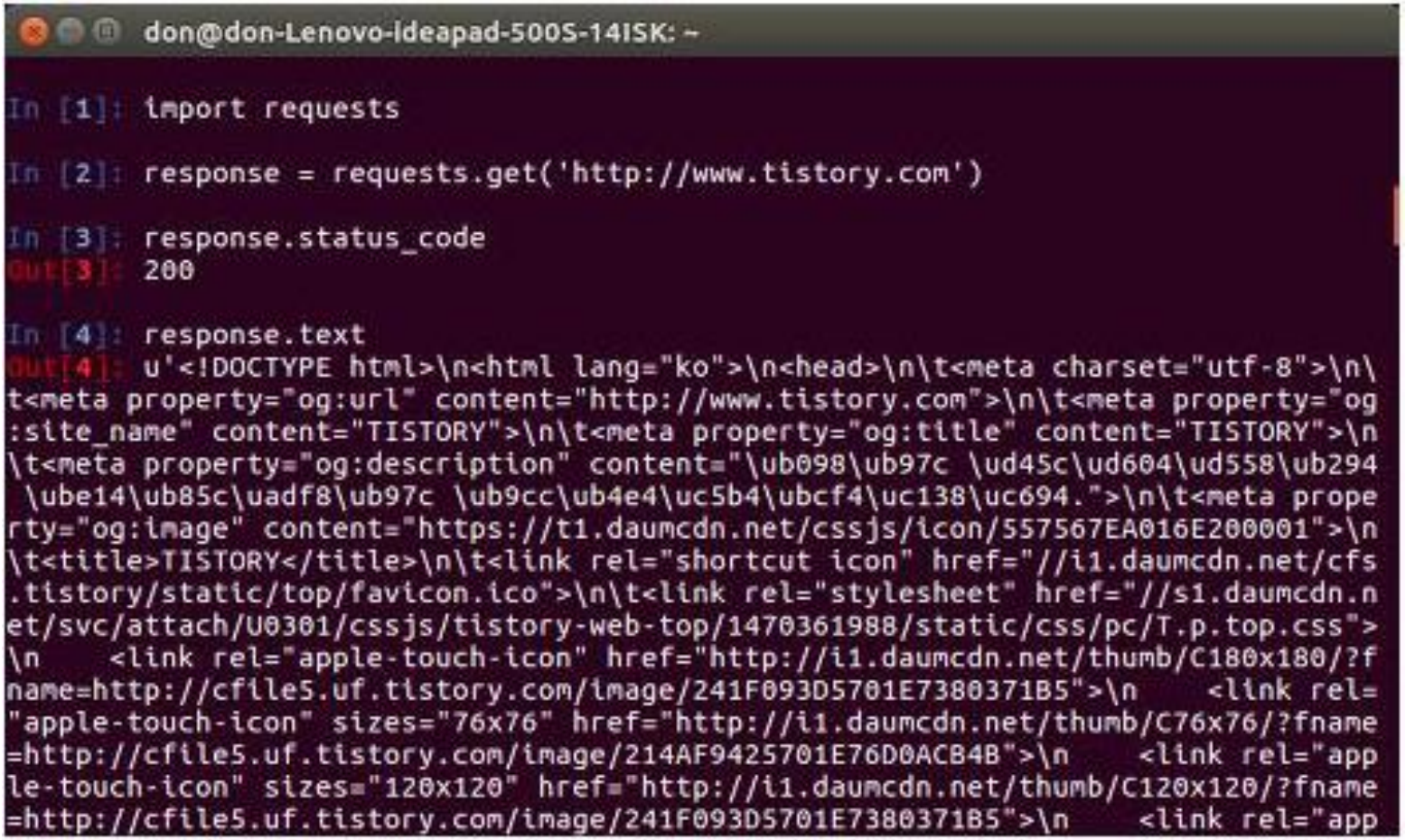
1부터 100까지의 합은 5050입니다.

BeautifulSoup parser

- Python 내장 html.parser or lxml's HTML parser
  - `BeautifulSoup(html,"html.parser")` or `BeautifulSoup(html,"lxml")`
  - 일반적으로 lxml 이 빠르고 flexible 하다고 알려짐 (정확하게 html 로 마크업이 안되어 있을 경우에는 lxml 사용)
  - (ex) Google Finance 의 예 - </tr>, </td> 가 없음. 이때 "lxml" parser 사용

```
<div id=prices class="gf-tablewrapper sfe-break-bottom-16">
<table class="gf-table historical_price">
    <tr class=bb>
        <th class="bb lm lft">Date
        <th class="rgt bb">Open
        <th class="rgt bb">High
        <th class="rgt bb">Low
        <th class="rgt bb">Close
        <th class="rgt bb">Volume
    <tr>
        <td class="lm">Feb 28, 2014
        <td class="rgt">100.71
        <td class="rgt">100.71
        <td class="rgt">100.71
        <td class="rgt rm">0
```

Requests module



Requests module

- url에 access하기 위해 사용하는데 유용
  - request.get() ; 특정 source로부터 data를 얻거나 검색하기 위해 사용
  - request.post()
  - request.put()
  - request.delete()
- GET/POST 상관없이 parameter를 다시 encoding할 필요없다. 단순히 dictionary 형태로 argument를 취하면 된다
  - userdata = {firstname:"John", "lastname":"Doe", "passwd":"jdoe123"}
  - resp = request.post('http://www.mywebsite.com/user', data=userdata)
- Built-in JSON decoder
  - resp.json()
  - resp.text ; if response data is just text

Urllib module

- BeautifulSoup ; parsing html
  - find()
  - find_all()
  - prettify()
  - select_one() ; query by CSS
  - select(id / tags / class / tag.class / id>tag.class ..)
- Urllib.request
  - urlopen() ; open the url
- Requests
  - get()

JSON

- JSON Encoding: Python Object (dict, list, tuple 등) 를 JSON 문자열로 변경
  - ✓ json.dumps(result)
- JSON Decoding: JSON 문자열 -> Python type (dict, list, tuple, 등)
  - ✓ json.loads(obj)
- json_normalize()
  - ✓ semi-structured JSON 데이터를 손쉽게 DataFrame으로 전환

```
import json, requests
from pandas.io.json import json_normalize

json_normalize(json.loads(r.text), 'data')
```

urllib.parse

- urllib.parse
  - ✓ 이 모듈은 URL(Uniform Resource Locator) 문자열을 구성 요소(주소 지정 체계, 네트워크 위치, 경로 등)로 분리하고, 구성 요소를 다시 URL 문자열로 결합하고, 《상대 URL》 을 주어진 《기본 URL》 에 따라 절대 URL로 변환하는 표준 인터페이스를 정의

- urllib.parse.urlencode(query)
  - ✓ 인자로 받은 매핑 객체 또는 2개의 요소로 이루어진 tuple을 POST 방식으로 전송하기 위한 데이터로 변환
  - ✓ query :
    - ➢ 변환할 데이터 문자열
    - ➢ 구성 : { key : value }

parameter를 전송하여 data를 읽는 경우

- URL 끝 부분에 ?를 입력하고, key = value 형식으로 매개변수를 추가
- 여러개의 parameter를 넣는 경우 &를 사용하여 구분
- 한글 등이 parameter로 사용될 때는 반드시 이러한 코딩을 해주어야 한다

```
import urllib.parse
Import urllib.request

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

# 파라미터 코딩, 딕셔너리 사용, 109: 서울 경기지역
values = {
    'stdId': '109'
}

params = urllib.parse.urlencode(values)
url = API + "?" + params
data = urllib.request.urlopen(url).read()  # 바이너리 데이터를 문자열로 변환
text = data.decode("utf-8")
print(text)
```

Web에서 file 다운로드하기

- urllib.request.urlretrieve(url, savename)

  - url**에서** download**한 데이터를 파일에 바로 저장**

- mem = urllib.request.urlopen(url).read()

  - urlopen()**을 사용하면 데이터를 파이썬에서 읽을 수 있다**.

문자 encoding/decoding

- 문자를 인코딩할 때는 출력 장치(콘솔, 주피터 노트북)가 어떤 인코딩을 지원하는지를 미리 알고 있어야 한다.
- 만약 출력 장치가 지원하지 않는 방식으로 인코딩하면 화면에는 이상한 글자만 보이게 된다.

- 인코딩(encoding)
  - ✓ 어떤 글자를 어떤 숫자로 바꿀지에 대한 규칙
  - ✓ 가장 기본이 되는 인코딩 방식은 아스키(ASCII) 코드
    - ➢ http://www.asciitable.com/

- 세계 표준 : 최대 4바이트의 숫자로 전 세계 모든 글자를 대응시킨 것
  - ✓ 유니코드(unicode)
  - ✓ UTF-8, UTF-16, UTF-32
  - ✓ encode() : 유니코드를 바이트 열로 변환할 때
  - ✓ decode() : 바이트 열을 유니코드로 변환할 때

정규식 표현(Regular Expression(RE))

- **Regular Expression**
  - 특정한 규칙의 문자열의 집합을 표현하기 위한 형식 언어
- **Meta characters**

표현식	의미
^x	문자열의 시작을 표현하며 x 문자로 시작됨을 의미한다.
x$	문자열의 종료를 표현하며 x 문자로 종료됨을 의미한다.
.x	임의의 한 문자의 자리수를 표현하며 문자열이 x 로 끝난다는 것을 의미한다.
x+	반복을 표현하며 x 문자가 한번 이상 반복됨을 의미한다.
x?	존재여부를 표현하며 x 문자가 존재할 수도, 존재하지 않을 수도 있음을 의미한다.
x*	반복여부를 표현하며 x 문자가 0번 또는 그 이상 반복됨을 의미한다.
x|y	or 를 표현하며 x 또는 y 문자가 존재함을 의미한다.
(x)	그룹을 표현하며 x 를 그룹으로 처리함을 의미한다.
(x)(y)	그룹들의 집합을 표현하며 앞에서 부터 순서대로 번호를 부여하여 관리하고 x, y 는 각 그룹의 데이터로 관리된다.
(x)(?:y)	그룹들의 집합에 대한 예외를 표현하며 그룹 집합으로 관리되지 않음을 의미한다.
x{n}	반복을 표현하며 x 문자가 n번 반복됨을 의미한다.
x{n,}	반복을 표현하며 x 문자가 n번 이상 반복됨을 의미한다.
x{n,m}	반복을 표현하며 x 문자가 최소 n번 이상 최대 m 번 이하로 반복됨을 의미한다.

Regular Expression(RE)

표현식	의미
[xy]	문자 선택을 표현하며 x 와 y 중에 하나를 의미한다.
[^xy]	not 을 표현하며 x 및 y 를 제외한 문자를 의미한다.
[x-z]	range를 표현하며 x ~ z 사이의 문자를 의미한다.
₩^	escape 를 표현하며 ^ 를 문자로 사용함을 의미한다.
₩b	word boundary를 표현하며 문자와 공백사이의 문자를 의미한다.
₩B	non word boundary를 표현하며 문자와 공백사이가 아닌 문자를 의미한다.
₩d	digit 를 표현하며 숫자를 의미한다.
₩D	non digit 를 표현하며 숫자가 아닌 것을 의미한다.
₩s	space 를 표현하며 공백 문자를 의미한다.
₩S	non space를 표현하며 공백 문자가 아닌 것을 의미한다.
₩t	tab 을 표현하며 탭 문자를 의미한다.
₩v	vertical tab을 표현하며 수직 탭(?) 문자를 의미한다.
₩w	word 를 표현하며 알파벳 + 숫자 + _ 중의 한 문자임을 의미한다.
₩W	non word를 표현하며 알파벳 + 숫자 + _ 가 아닌 문자를 의미한다.

Flag	의미
g	Global 의 표현하며 대상 문자열내에 모든 패턴들을 검색하는 것을 의미한다.
i	Ignore case 를 표현하며 대상 문자열에 대해서 대/소문자를 식별하지 않는 것을 의미한다.
m	Multi line을 표현하며 대상 문자열이 다중 라인의 문자열인 경우에도 검색하는 것을 의미한다.

Regular Expression(RE)

- **Examples**
  - ^http : 문자열의 맨 처음에 http가 온 경우에 매치
  - them$: 문자열이 them으로 끝난 경우에 them에 매치
  - \bplay\b : play 의 양 끝에 단어 경계가 오는 경우에만 play에 매치
    - ("playground"의 play에는 매치하지 않음)
  - \bplay\B : play뒤에 단어 경계가 아닌 것이 왔을 때 play에 매치
    - (play는 No, "playground", "playball" 은 Yes)
  - /[0-9]/g : 전체에서 0~9 사이의 아무 숫자 '한 개' 찾음
  - /[to]/g : 전체에서 t 혹은 o 를 찾음
  - /filter/g : 전체에서 'filter' 라는 단어에 매칭되는 것을 찾음
  - \b(?!\b**to**\b)\w+\b : 'to'라는 단어 빼고 다른 단어 매칭
    - (\w+ 는 match one or more word charqcters: '[a-zA-Z0-9_]'
  - /^\d{3}-\d{3,4}-\d{4}$/ : 전화번호
  - /^01([0|1|6|7|8|9]?)-?([0-9]{3,4})-?([0-9]{4})$/ 휴대폰번호

Regular Expression(RE)

- findall()
  - ✓ **정규식과 매치되는 모든 문자열을 list형식으로** return

```
import re
import requests

re.findall('<span class="ah_k">(.*?)</span>', requests.get('http://naver.com').text)[:20]
```

Beautifulsoup

- HTML**과** XML **문서에서 정보를 추출할 수 있다**
  - ✓ BeautifulSoup(html, 'html.parser')
    - ➢ html **형식의 파일을 읽어 들여서 깔끔하게 정리하여 준다**
  - ✓ BeautifulSoup(markup, 'xlml')
- https://twpower.github.io/84-how-to-use-beautiful-soup
- https://victorydntmd.tistory.com/245

Web에서 원하는 정보 추출

- HTML **파일을 직접 열기**

```
Import request
Import urllib.parse

with open("example.html")  as fp:
    soup = BeautifulSoup(fp,  'html.parser')

soup
```

Web에서 원하는 정보 추출

- urllib를 통해 웹에 있는 소스 가져오기

```
web_url = 'http://www.naver.com'

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

soup
```

- Requests를 통해 web에 있는 소스 가져오기

```
Import requests

r = requests.get(web_url)
r.status_code

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

Beautifulsoup – find, findall

- soup.find ('title')
  - tag값을 기준으로 내용 불러오기
- soup.find_all ('p' , limit = 2)
  - 해당 모든 tag 중 2개 까지만
- soup.find (align = "center")
  - 속성값을 기준으로 tag를 불러온다
- soup.find ('p' , align = "right")
  - p 태그 중 align = "right" 을 포함한 tag
- body_tag = soup.find ('body');

  list1 = body_tag.find_all (['p','img'])
  - [] 리스트 , 'p' 또는 'img' 를 포함하는 태그(or)
- head_tag = soup.find ('head');

  head_tag.find ('title')
  - head 태그 내부의 title 태그의 내용(and)

Beautifulsoup – find, findall

- id 를 사용하는 방법
  - ✓ 위와 같이 내부 구조를 일일이 파악하고 코딩하는 것은 복잡하다
  - ✓ 속성 값을 이용해서 가져 오기
  - ✓ find()를 사용하여 간단히 원하는 항목을 찾을 수 있다
  - find()
    - ➢ 해당 조건에 맞는 하나의 태그를 가져온다
    - ➢ 중복이면 가장 첫 번째 태그를 가져온다
  - find_all()
    - ➢ 해당 조건에 맞는 모든 태그들을 가져온다.

Beautifulsoup – find, findall

```
html = """
<html><body>
 <h1 id="title"> 파이선으로 웹문서 읽기 </h1>
 <p id="body"> 페이지 분석기능 </p>
 <p> 페이지 정렬 </p>
</body></html>
"""

soup = BeautifulSoup(html,  'html.parser')
title = soup.find(id="title")
body = soup.find(id="body")
```

DOM 요소 파악하기

- Document Object Model:
  - ✓ XML이나 HTML 요소에 접근하는 구조를 나타낸다
  - ✓ 요소(Element) :
    - ➢ 구성 : 시작 tag, 요소 내용, 종료 tag
  - ✓ 태그(tag) :
    - ➢ <, >로 둘러싸인 것으로 짝을 이룬다
    - ➢ 시작 태그, 종료 태크
  - ✓ 요소 속성(attributes of Element) : id, class, style, title, lang, href 등
    - ➢ 속성 = 속성 값 : href = \"http://www.naver.com"
  - ✓ 요소의 내용(string) : naver

DOM 요소 파악하기

```
from bs4 import BeautifulSoup
html = """
<html><body>
 <ul>
   <li><a href = "http://www.naver.com">naver</a></li>
   <li><a href = "http://www.daum.com">daum</a></li>
 </ul>
</body></html>
"""

soup = BeautifulSoup(html,  'html.parser')
links = soup.find_all("a")

for aa in links:
    href = aa.attrs["href"]
    text = aa.string
    print(text, "-->", href)
```

Beautifulsoup – CSS 선택자 사용하기

- CSS 선택자를 사용해서 원하는 요소를 추출할 수 있다.
- h1 과 li 태그를 추출하는 코드

Beautifulsoup – CSS 선택자 사용하기

- CSS 선택자를 사용해서 원하는 요소를 추출할 수 있다.
- h1 과 li 태그를 추출하는 코드

원하는 정보가 있는 위치 찾기

- soup.select('원하는 정보') : 단 하나만 있더라도, 복수 가능한 형태로 되어있음
  - soup.select('태그명')
  - soup.select('.클래스명')
  - soup.select('상위태그명 > 하위태그명 > 하위태그명')
    - 바로 아래의(자식) 태그를 선택시에는 > 기호를 사용
  - soup.select('상위태그명.클래스명 > 하위태그명.클래스명')
  - soup.select('상위태그명.클래스명 하위태그명')
    - 아래의(자손) 태그를 선택시에는 띄어쓰기 사용
  - soup.select('상위태그명 > 바로아래태그명 하위태그명')
  - soup.select('.클래스명')
  - soup.select('#id명')
    - 태그는 여러개에 사용 가능하나 id는 한번만 사용 가능함! ==> 선택하기 좋음
  - soup.select('태그명.클래스명)
  - soup.select('#id명 > 태그명.클래스명)
  - soup.select('태그명[속성1=값1]')

Beautifulsoup – 정규식 이용

- re (정규표현식)을 사용하여 필요한 데이터를 추출할 수 있다

```
from bs4 import BeautifulSoup
import re

html = """
<ul>
   <li><a href="https://sample.com/foo">fuga  </li>
   <li><a href="http://sample.com/okay">okay  </li>
   <li><a href="https://sample.com/fuga">  fuga* </li>
   <li><a href="https://example.com/sample">  aaa </li>
</ul>
"""
soup = BeautifulSoup(html,  'html.parser’)
li=soup.find_all(href=re.compile(r"^https://"))

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

베이즈(Bayes) 알고리즘

베이즈(Bayes) 이론

- Bayes 알고리즘
  - Bayes 이론을 이용한 machine learning 알고리즘

- Bayes 이론
  - 조건부 확률 이론을 이용하여 새로운 사건의 조건부 확률을 예측
  - 단순하지만 성능이 우수

베이즈(Bayes) 이론

- 독립적인 두 사건 A와 B가 있을 때
- 사건 A가 발생했다는 조건에서 사건 B가 발생할 확률

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} = \frac{P(A,B)}{P(A)}$$

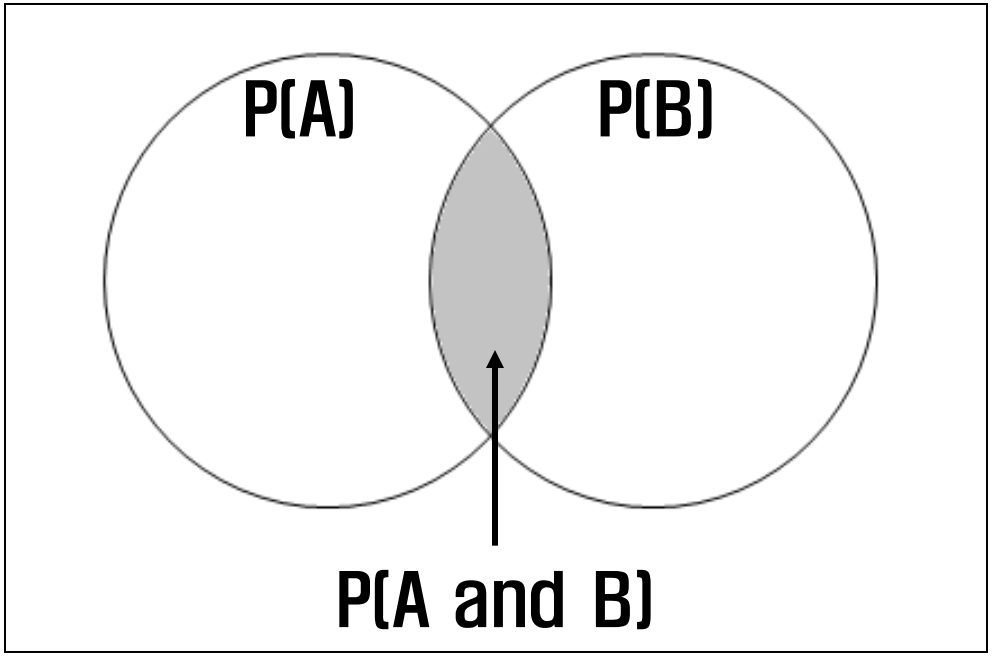
- 사건 B가 발생했다는 조건에서 사건 A가 발생할 확률

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(A,B)}{P(B)}$$

- P(A,B)로 정리하면 P(B|A)P(A) = P(A|B)P(B)

  수식에서 3개의 확률을 알면 나머지 하나는 구할 수 있음

베이즈(Bayes) 이론



$$P(B|A) = \frac{P(A,B)}{P(A)}$$

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

$$P(B|A)P(A) = P(A|B)P(B)$$

- 수식에서 3개의 확률을 알면 나머지 하나는 구할 수 있음

베이즈(Bayes) 이론 예

- 피부암 적용 예
  - 사건 A : 피부암에 걸릴 확률
  - 사건 B : 붉은 반점이 생길 확률
  - 피부암에 걸리면 붉은 반점이 생길 확률 : P(B|A) = 0.99

- 사전 확률
  - 피부암이 발생 확률 : P(A) =0.0001 (1만명 중에 1 명)
  - 붉은 반점이 나타난 확률 : P(B) = 0.01 (1만명 중에 100명)

- 붉은 반점이 나타났다는 조건 하에 내가 피부암일 확률

$$P(A|B) = P(B|A)P(A)/P(B) = (0.99)*(0.0001)/0.01 = 0.01$$

베이즈(Bayes) 이론 예 – spam 메일 예측

- 수신한 메일이 spam인 사건: A
  - 통계적으로 100통 중 하나가 spam이었다면
  - 메일이 스팸일 확률은

$$P(A) = 0.01$$

- 메일이 spam인지 여부를 판단하는 알고리즘
  - "쿠폰"이라는 단어 하나만 보고 스팸인지 아닌지를 판단
  - 과거 통계로부터 메일에 "쿠폰" 단어가 들어 있을 확률은 평균 0.003
- 전체 메일중에 '쿠폰' 단어가 들어있는 사건 : B

$$P(B) = 0.003$$

- 전체 spam 중에 '쿠폰'을 포함한 메일이 10%였다고 하면

$$P(B|A) = 0.1$$

베이즈(Bayes) 이론 예 – spam 메일 예측

- 이제 새로운 메일이 하나 도착했는데, 여기에 '쿠폰'이라는 단어가 포함되어 있다면
  - 이 메일이 spam일 확률 ?
  - Bayes 이론에 따라

$$\begin{aligned} P(A|B) &= P(B|A)P(A)/P(B) \\ &= (0.1)(0.01)/(0.003) \\ &= 0.333 \end{aligned}$$

베이즈(Bayes) 이론 예 – spam 메일 예측

쿠폰의 존재 여부와 spam 메일과 정상 메일의 분포 표

	"쿠폰"		
	있음	없음	합
스팸	1	9	10
정상 메일	2	988	990
합	3	997	1000

$P(A) = 0.01$
$P(B) = 0.003$
$P(B|A) = 0.1$
$P(A|B) = 0.333$

- 전체 1000 개의 메일 중에 스팸의 총 수가 10개, 따라서 스팸 메일이 발생할 확률은 P(A)=0.01
- 모든 메일 중에 "쿠폰" 단어가 들어간 메일은 총 3개이므로 P(B)=0.003
- 스팸 메일 10개만을 놓고 볼 때, 쿠폰 단어가 들어간 것은 1개, 9개에는 쿠폰 단어가 없었다.
  즉, 10%의 스팸 메일에 쿠폰이 들어 있었다. P(B|A)=0.1
- 새로운 메일에 쿠폰이라는 단어가 들어 있었다.
  이 메일의 스팸일 확률은?

나이브 베이즈 알고리즘

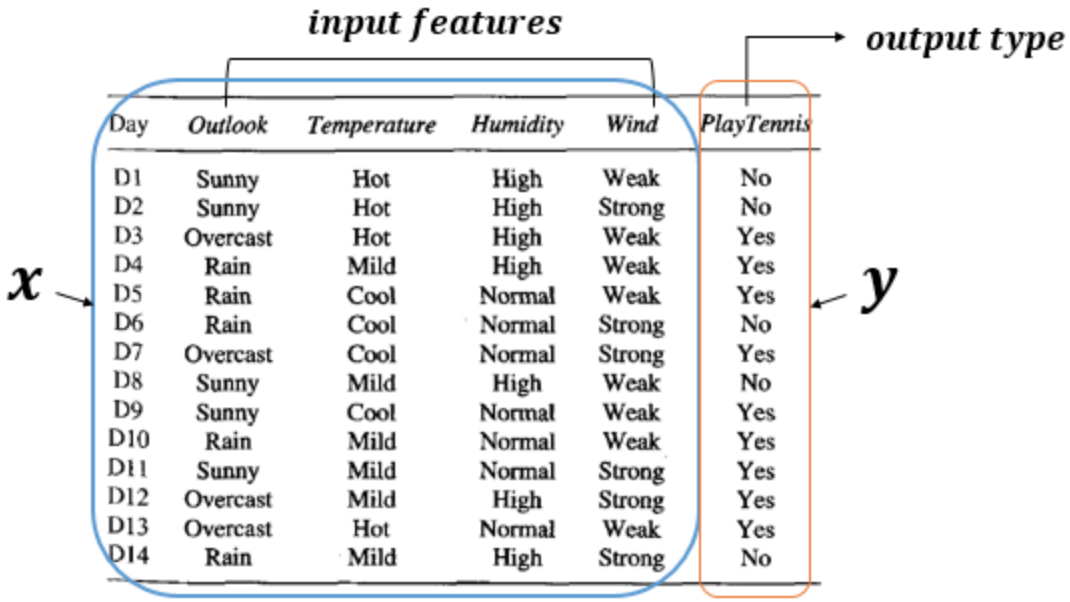
- Spam 메일 여부의 판단을 **여러 개의 단어**를 고려하면
  - 한 단어만 고려하는 것보다 더 정교하게 spam 메일을 찾아낼 수 있을 것

- 여러 개의 사건이 결합된 경우
  - 조건부 확률 식이 복잡 : **각 단어**의 발생 간에 **서로 조건부 확률** 존재
  - 이를 모두 고려하면 알고리즘 구현이 매우 복잡
  - Ex) '쿠폰', '할인' : 서로 독립적인 사건 아니다. 동시 발생 확률 높다

- **나이브 베이즈**(Naïve Bayes, NB) 알고리즘
  - 여러 특성변수들이 **서로 독립적**이라고 가정(Naïve) 하에 단순화
  - 서로 독립이라 가정해도 상대적인 확률은 구할 수 있다
  - 나이브 베이즈로 추정한 값은 0~1 사이의 값을 값지만, 이는 정확한 확률값을 구한 것이 아니라 상대적인 점수(score)를 구한 것
  - 일정한 점수 이상의 메일을 spam으로 처리, 오차 발생하면 조정 통한 학습
  - 평소 통계를 기반으로 확률값만 준비하면 됨. 동작이 매우 단순, 속도 빠름
  - 특정 단어가 들어있는 지를 파악해야 하는 블로그 분석, 트위터 분석 등 텍스트 분석에 널리 사용되는 등 거의 모든 머신 러닝을 이용한 데이터 분석에 사용

나이브 베이즈 알고리즘

- Data set 예



Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

나이브 베이즈 알고리즘

- 다시 Bayes 정리

$$P(y|X) = \frac{P(X|y)p(y)}{P(X)}$$

- n-차원 특징 벡터에 대해 풀어 쓰면
- X : input의 특징(feature) 벡터(n-차원)

$$X = (x_1, x_2, x_3, \cdots, x_n)$$

- 각각의 특징들이 어떤 상관관계가 있을 수 있지만, conditionally independent 가정
- 임의의 두 feature에 대해 joint probability 구하고, 독립을 적용

$$\begin{aligned} P(x_1, x_2|y) &= P(x_1|x_2, y)P(x_2|y) \\ &= P(x_1|y)P(x_2|y) \end{aligned}$$

- n-차원으로 확장

$$P(x_1, \cdots, x_n|y) = \prod_{i=1}^{n} P(x_i|y)$$

나이브 베이즈 알고리즘

- 다시 Bayes 정리

$$P(y|X) = \frac{P(X|y)p(y)}{P(X)}$$

- n-차원 feature에 대해 joint probability 구하면

$$P(y|x_1, \cdots, x_n) = \frac{P(x_1, \cdots, x_n|y)p(y)}{P(x_1, \cdots, x_n)}$$

- conditional independence를 적용

$$P(y|x_1, \cdots, x_n) = \frac{P(x_1|y)P(x_2|y)\cdots P(x_n|y)p(y)}{P(x_1)P(x_2)\cdots P(x_n)}$$

- argmax 문제에서는 각각의 정확한 값이 중요하지 않고, 또한 X에 대한 확률은 상수 값을 가지므로 무시 가능

$$\tilde{y} = h_{NB}(X) = \underset{y}{\operatorname{argmax}}\, p(y) \prod_{i=1}^{n} P(x_i|y)$$

수고하셨습니다.

Q & A