

생성형AI 활용_웹 크롤링

2024.05.23.

강사: 부성순 (bakpak@empas.com)

Part1. 웹데이터 구조 이해 및 파이썬 기초

- (1) 웹데이터 구조의 이해
- (2) 웹크롤링을 위한 Html 구조 이해
- (3) GPT를 이용한 html 구조의 이해 및 생성
- (4) 크롤링에 필요한 파이썬 필수 문법

강사: 부성순 (bakpak@empas.com)

웹 크롤링 (Web crawling)

- 봇이 많은 웹사이트들을 돌아다니면서 URL, 키워드 등을 수집하는것
(보통 검색 엔진에서 웹사이트를 수집하기 위해 사용됨)

웹 스크래핑 (Web scraping)

- 웹사이트에서 필요한 데이터를 긁어오는것

웹 프로그래밍 언어

위키백과, 우리 모두의 백과사전.

웹 프로그래밍 언어는 WWW에서 사용되는 프로그래밍 언어들을 통틀어 말한다. 일부는 웹 프로그래밍 언어에 마크업 언어를 텍스트 언어라는 이유로 포함시키지 않지만, 일부는 TeX같이 프로그래밍이 가능한 마크업 언어가 있기 때문에, 마크업 언어를 포함시키기도 한다.

목차 [\[보이기\]](#)

웹 프로그래밍 언어 [\[편집\]](#)

웹 프로그래밍 언어로는 PHP, JSP, ASP, .NET 등이 있다.

마크업 언어 [\[편집\]](#)

마크업 언어로는 SGML 계열과 TeX 계열이 있다.

SGML 계열 [\[편집\]](#)

SGML 계열의 마크업 언어로, HTML, XML이 있다.

TeX 계열 [\[편집\]](#)

TeX 계열 마크업 언어로, LaTeX와 pTeX가 있다.

같이 보기 [\[편집\]](#)

- 웹 애플리케이션 프레임워크

파일명: test.html

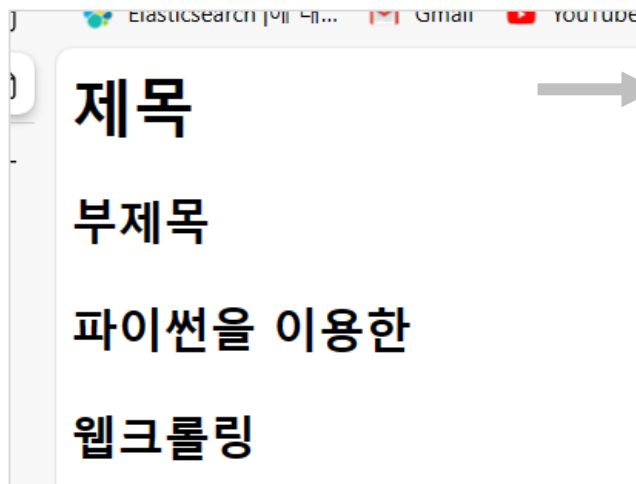
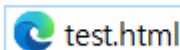
```
<h1>제목
<h2>부제목
<p> 파이썬을 이용한 </p> <p> 웹크롤링 </p>
```

Ln 2, Col 8

100%

Windows (CRLF)

UTF-8



test.html
실행파일

▶다음 용어에 대하여 정리합니다.

→ 통신프로토

→ http

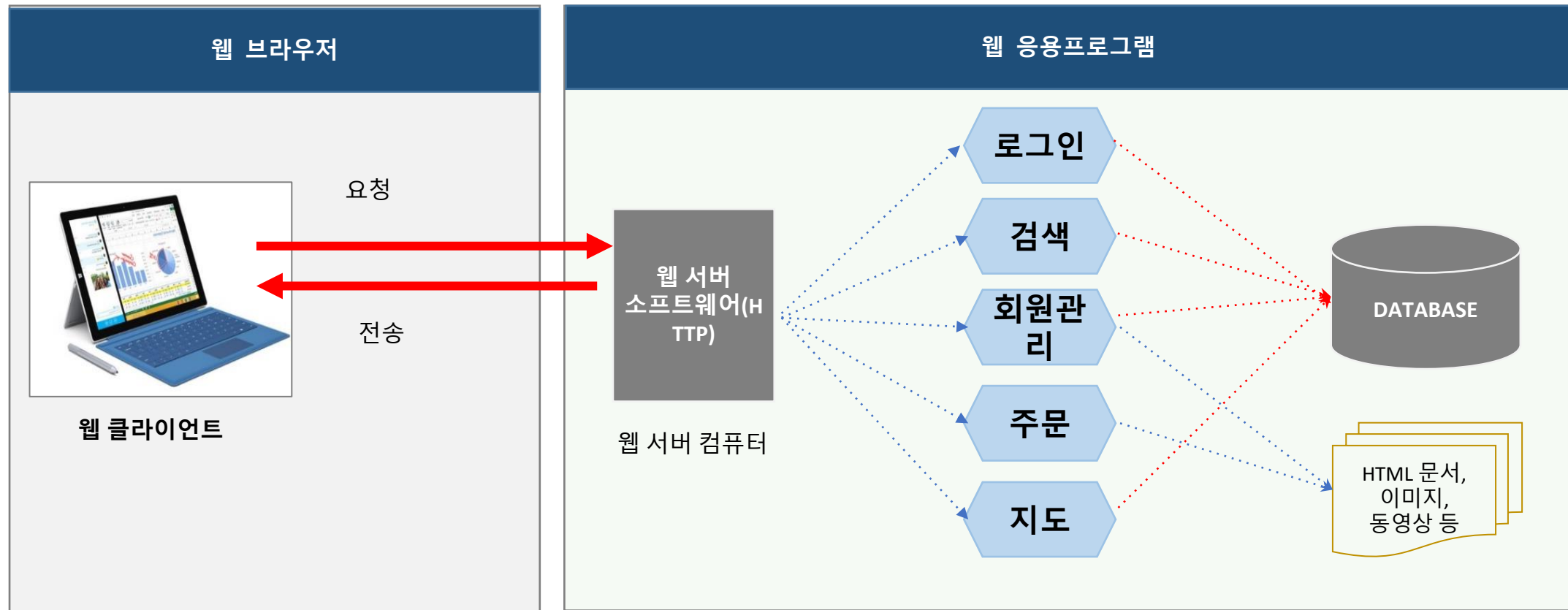
→ URL

→ 웹서버와 클라이언트

→ 웹퍼블리싱

--> 웹 사이트 구축

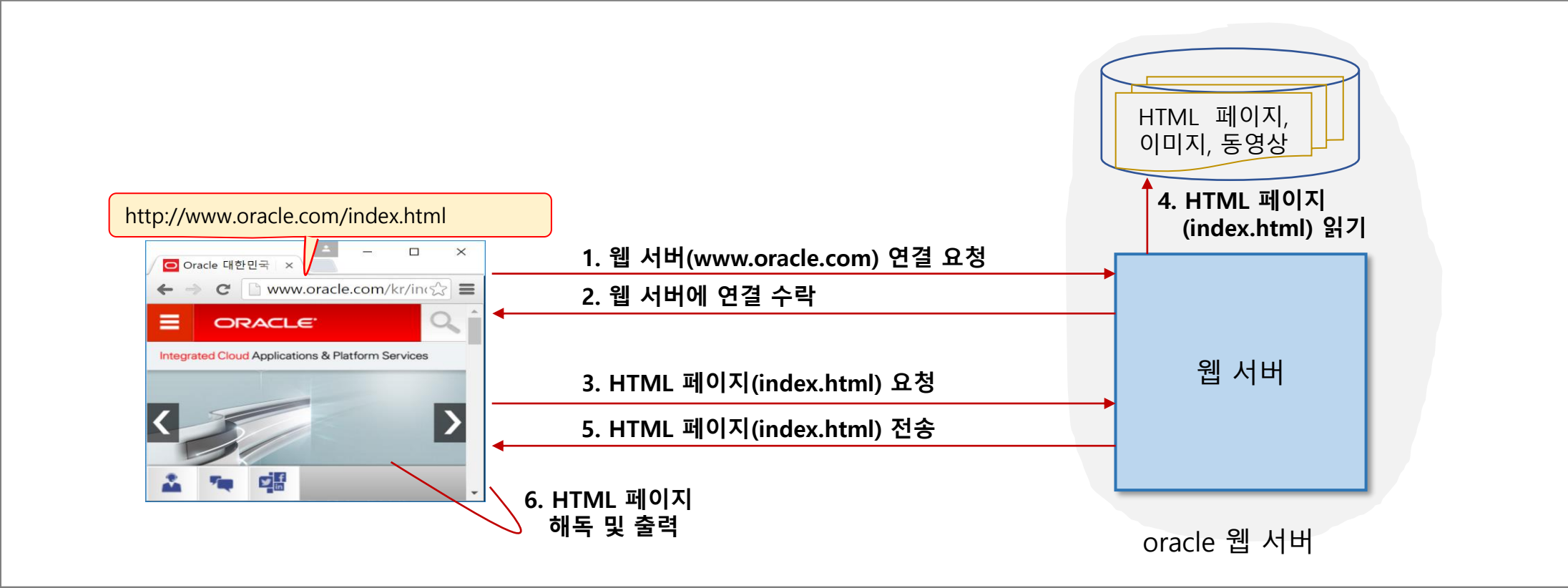
- 웹 서버로 사용할 컴퓨터에 웹 서버 소프트웨어 설치
- 웹 페이지, 동영상, 이미지 저장, 데이터베이스 설치
- 웹 서버 응용프로그램 개발 및 설치



웹 페이지 주소(URL)

1. 웹데이터 구조 이해 및 파이썬 기초 - (1) 웹데이터 구조의 이해

--> 웹 브라우저와 웹 서버 통신 HTTP



파싱	인터넷에 주어진 정보를 내가 원하는대로 가공하여 서버에서 원하는 때에 불러올 수 있도록 하는 것	XML 을 파싱하는 방법과 JSON 을 파싱하는 방법으로 크게 나눌 수 있음
태그	어떤 정보에 메타데이터로 부여된 키워드 또는 분류이다.	HTML태그란? HTML을 기술하기 위하여 사용하는 명령어의 집합을 태그(Tag)라함.

html

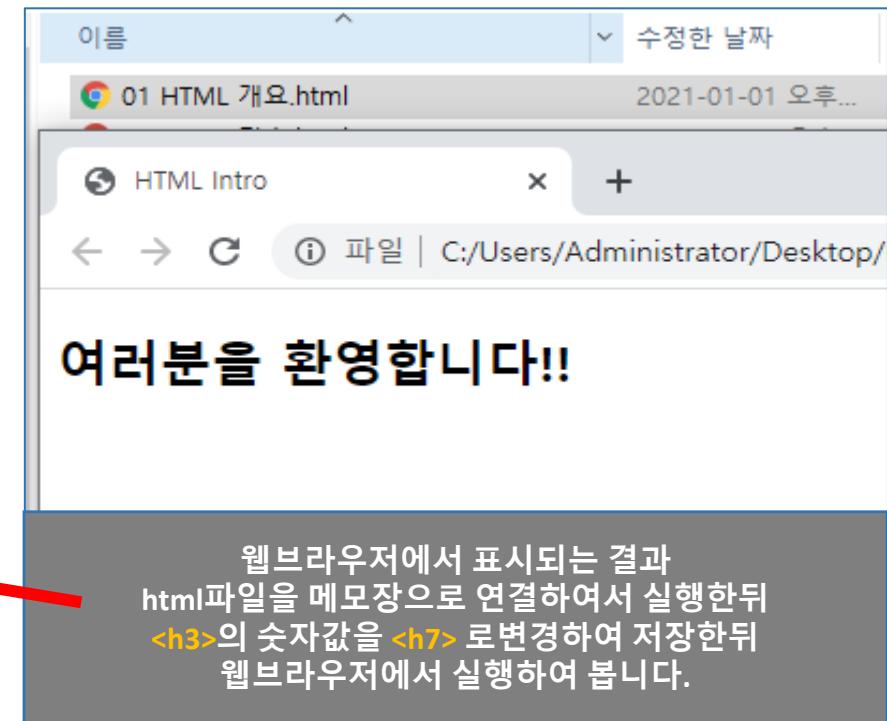
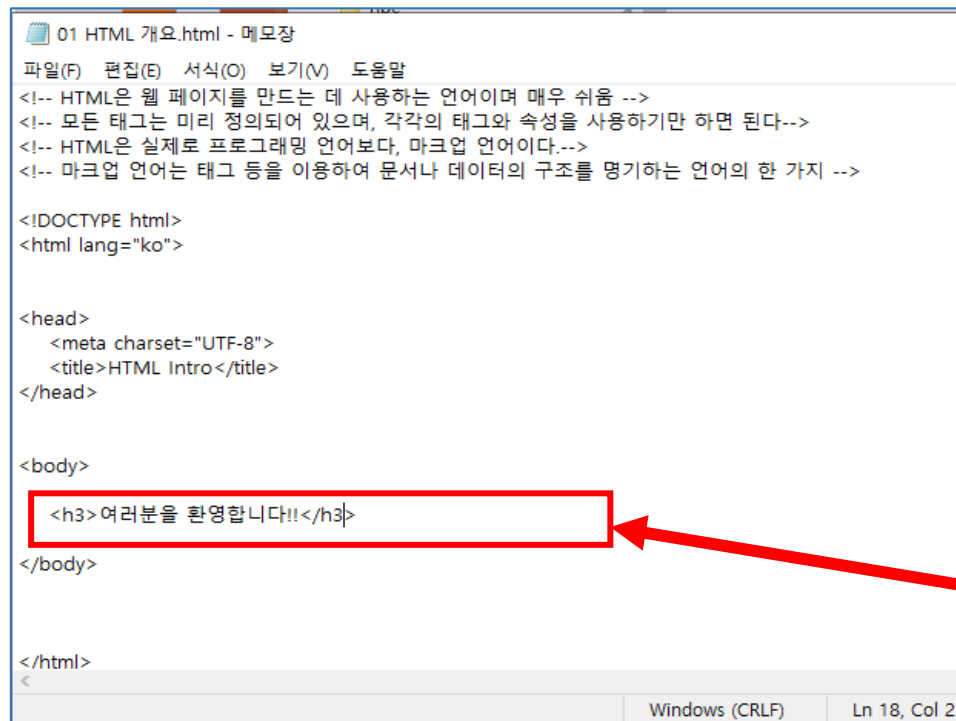


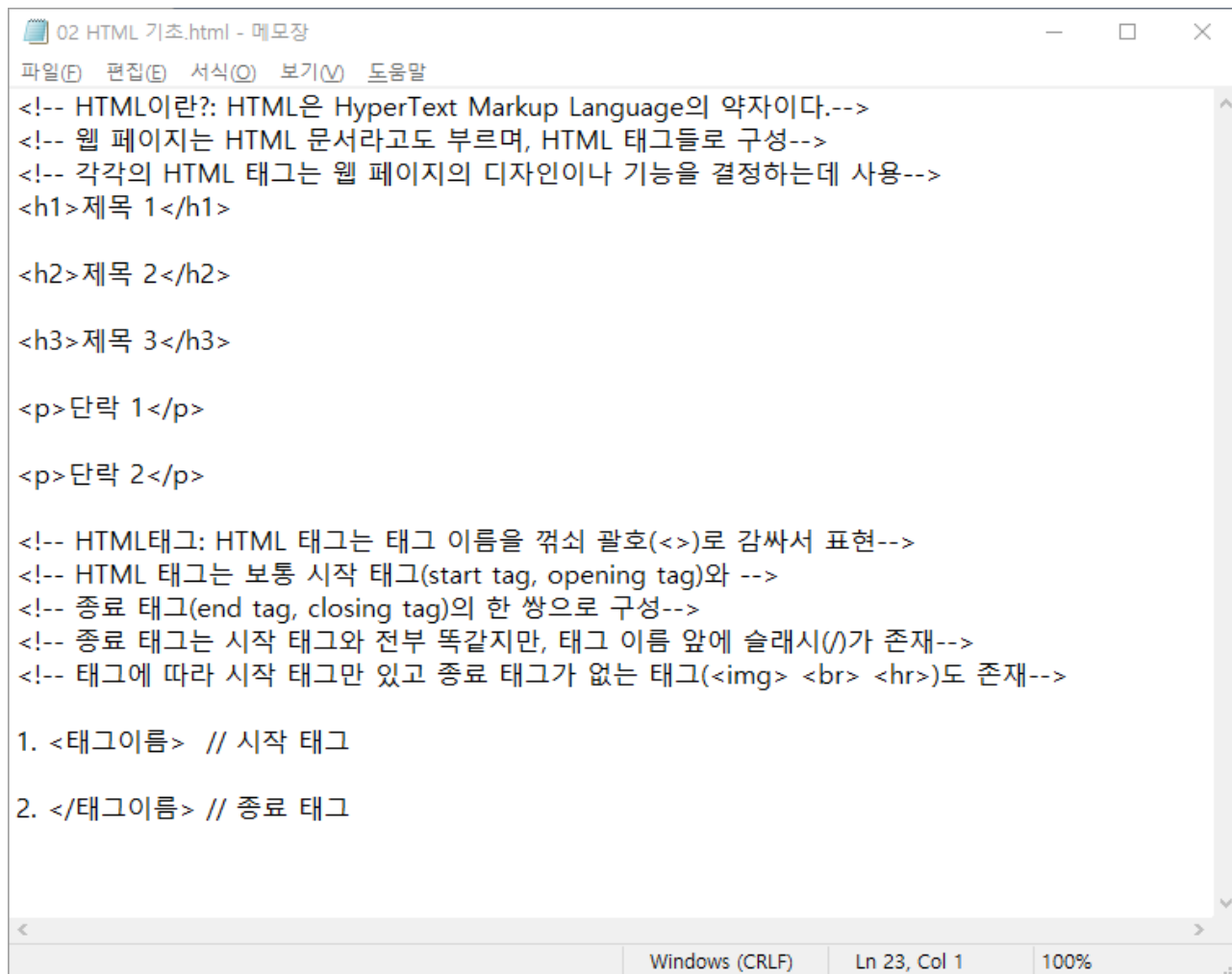
하이퍼텍스트 마크업 언어(HyperText Markup Language, **HTML**, 문화어: 초본문표식달기언어, 하이퍼본문표식달기언어)는 웹 페이지를 위한 지배적인 마크업 언어다.

```
<!DOCTYPE html>
<!-- 주석문 -->
<html>

  <head>
    문서의 제목, CSS, 자바스크립트, 메타데이터 정의
  </head>

  <body>
    문서의 본문 텍스트, 이미지, 동영상, 테이블,
    자바스크립트 코드 등
  </body>
</html>
```





```
02 HTML 기초.html - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말
<!-- HTML이란?: HTML은 HyperText Markup Language의 약자이다.-->
<!-- 웹 페이지는 HTML 문서라고도 부르며, HTML 태그들로 구성-->
<!-- 각각의 HTML 태그는 웹 페이지의 디자인이나 기능을 결정하는데 사용-->
<h1>제목 1</h1>

<h2>제목 2</h2>

<h3>제목 3</h3>

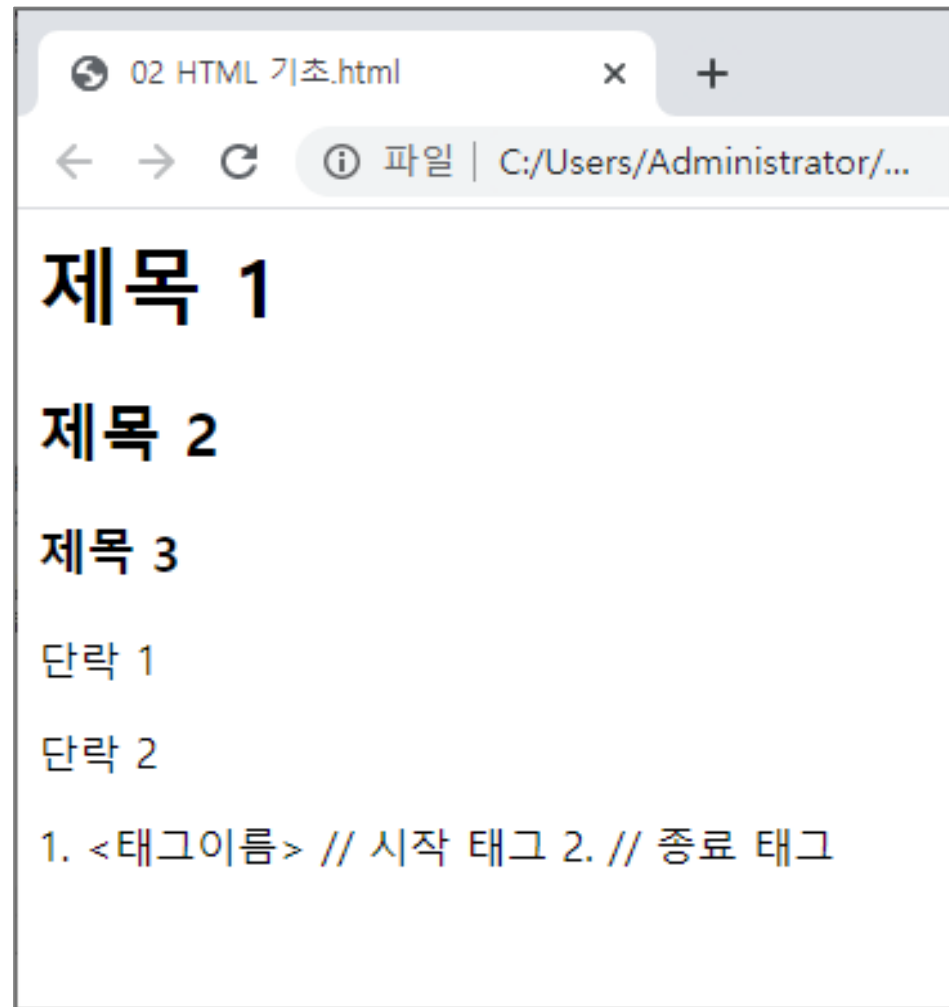
<p>단락 1</p>

<p>단락 2</p>

<!-- HTML태그: HTML 태그는 태그 이름을 꺾쇠 괄호(<>)로 감싸서 표현-->
<!-- HTML 태그는 보통 시작 태그(start tag, opening tag)와 -->
<!-- 종료 태그(end tag, closing tag)의 한 쌍으로 구성-->
<!-- 종료 태그는 시작 태그와 전부 똑같지만, 태그 이름 앞에 슬래시(/)가 존재-->
<!-- 태그에 따라 시작 태그만 있고 종료 태그가 없는 태그(<img> <br> <hr>)도 존재-->

1. <태그이름> // 시작 태그
2. </태그이름> // 종료 태그

Windows (CRLF) Ln 23, Col 1 100%
```



태그 규칙

- 태그는 대소문자를 구분하지 않습니다. 그러나 일반적으로 가독성을 위해 소문자를 사용
- <> 괄호를 둘러 사용 예: <p>
- 태그는 일반적으로 시작 태그와 종료 태그의 짝으로 구성
예:<p>콘텐츠</p>
- 단일 태그는 속성을 포함할 수 있으며, 콘텐츠를 가지지 않음
- 태그는 중첩하여 사용 가능 예:<div><h1>콘텐츠</h1></div>
- 태그는 가독성을 위해 일반적으로 들여쓰기를 하여 작성
예)
<div>
 <h1>콘텐츠</h1>
</div>

자주 사용되는 태그

- <html>: 전체 HTML 문서의 루트 요소를 정의
- <head>: 웹페이지의 메타 정보, 제목, 외부 스크립트 및 스타일시트를 포함하는 데 사용
- <title>: 웹페이지의 제목을 정의하며, 브라우저 탭에 표시
- <body>: 웹페이지의 내용을 담는 요소로, 웹페이지에 표시되는 모든 요소가 포함
- <h1>~<h6>: 제목 또는 부제목을 나타내는 데 사용. <h1>이 가장 크고, <h6>이 가장 작음.
- <p>: 문단(paragraph)을 정의하는 태그
- <a>: 하이퍼링크(anchor)를 정의. 다른 웹페이지로 이동하거나 문서 내 특정 부분으로 이동하는 데 사용
- : 이미지를 삽입하는 데 사용. src 속성을 사용하여 이미지 파일의 경로를 지정
- : 순서 없는 목록(unordered list)을 정의. 태그를 사용하여 목록 항목을 만듦
- : 순서 있는 목록(ordered list)을 정의. 태그를 사용하여 목록 항목을 만듦
- : 목록 항목(list item)을 정의. 또는 태그 안에 사용됨

- <table>: 표(table)를 정의하는 데 사용되며, <tr>(table row), <th>(table header), <td>(table data) 등의 태그와 함께 사용하여 표의 구조를 만듦
- <form>: 사용자 입력을 수집하는 양식(form)을 정의하며, <input>, <textarea>, <button> 등의 태그를 포함하여 사용자로부터 데이터를 받음
- <div>: 구획(division)을 정의하는 태그로, 웹페이지의 레이아웃과 구조를 만드는 데 사용. CSS와 함께 사용되어 스타일 및 위치를 지정할 수 있음
- : 인라인 요소를 그룹화하는 데 사용되는 태그로, 주로 텍스트 내에서 특정 부분에 스타일을 적용할 때 사용
- <header>: 웹페이지 또는 구획의 헤더를 정의하는 태그로, 주로 로고, 내비게이션, 제목 등의 요소를 포함
- <footer>: 웹페이지 또는 구획의 바닥글(footer)을 정의하는 태그로, 저작권 정보, 연락처 정보, 링크 등의 요소를 포함할 수 있음
- <nav>: 네비게이션(navigation)을 정의하는 태그로, 웹페이지의 주요 메뉴나 링크를 포함
- <aside>: 웹페이지의 사이드바(sidebar)를 정의하는 태그로, 주로 추가 정보, 링크, 광고 등의 요소를 포함

HTML의 속성(attribute)은 요소(element)에 추가 정보를 입력하거나 요소의 동작, 모양을 변경하는데 사용
속성은 태그 내부에 위치하며 이름과 값의 쌍으로 구성태그에 추가되는 속성의 개수는 제한이 없음
띄어쓰기를 사용해 여러 속성을 구분할 수 있음

```

```

속성의 값은 따옴표 없이 사용할 수 있지만, 특수문자가 사용될 경우 의도치 않은 오류가 생길 수 있기 때문에
일반적으로는 따옴표로 값을 묶어 사용

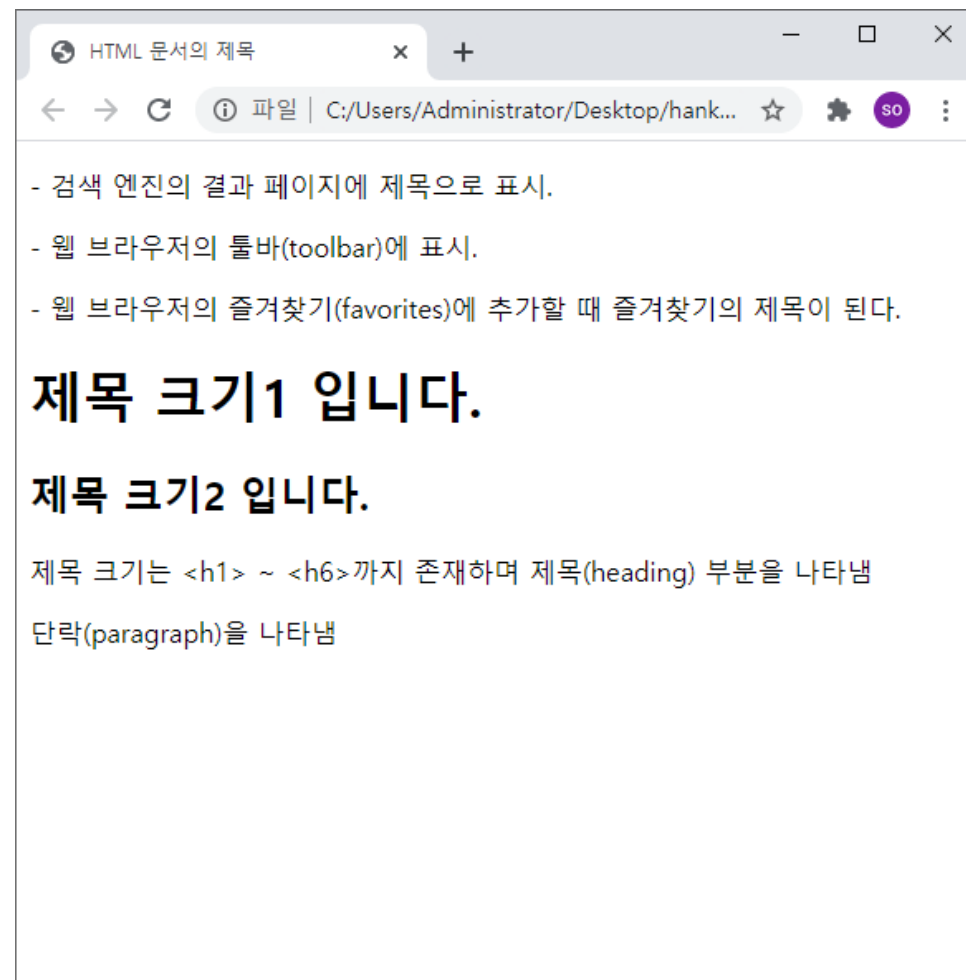
```
<!-- 잘못된 예시(공백) -->  
<img src=image file.png width=300 height=200>
```

./HTML이해/ 03 HTML 기초.html

```
03 HTML 기본 구조.html - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말
<!-- HTML 기본 구조 -->
<!DOCTYPE html>
<html>
  <head>
    <!-- *HTML 문서의 메타데이터(metadata)를 정의-->
    <!-- - 메타데이터(metadata)란 HTML 문서에 대한 정보(data)로 웹 브라우저에는 직접적으로 -->
    <!-- 표현되지 않는 정보를 의미-->
    <!-- - 이러한 메타데이터는 <title>, <style>, <meta>, <link>, <script>, <base> -->
    <!-- 태그 등을 이용하여 표현-->

    <title> HTML 문서의 제목 </title>
    <p>- 검색 엔진의 결과 페이지에 제목으로 표시.</p>
    <p>- 웹 브라우저의 툴바(toolbar)에 표시.</p>
    <p>- 웹 브라우저의 즐겨찾기(favorites)에 추가할 때 즐겨찾기의 제목이 된다.</p>
  </head>

  <body> <!-- 웹 브라우저를 통해 보이는 내용 -->
    <h1> 제목 크기1 입니다.</h1>
    <h2> 제목 크기2 입니다.</h2>
    <p> 제목 크기는 &lt;h1> ~ &lt;h6>까지 존재하며 제목(heading) 부분을 나타냄 </p>
    <p> 단락(paragraph)을 나타냄 </p>
  </body>
</html>
```



```

04 HTML 요소 구조.html - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말
<!-- HTML 요소 구조-->
<!-- HTML 요소(element)는 여러 속성을 가질 수 있으며, 이러한 속성(attribute)은 해당 요소에 대한 추가적인 정보를 제공-->
<!-- HTML 요소는 시작 태그로 시작해서 종료 태그로 끝남 -->


<p> 속성은 HTML 요소 중에서도 언제나 시작 태그 내에서만 정의되며, 속성 이름과 속성값(value)으로 표현 </p>
<h5> 문법 구조 </h5>
<p> <태그이름 속성이름 = "속성값"> </p>

<!-- 속성 이름은 언제나 소문자로 작성 -->
<!-- HTML5 표준에서는 속성 이름에 대소문자를 구분하지 않지만, W3C에서는 속성 이름을 사용할 때 될 수 있으면 소문자로 작성하도록 권장-->

<!-- 속성값은 언제나 따옴표로 감싸자 -->
<p>  </p>


    
```

< > 안에 속성값이 여러개나올수도 있음.

```

태그이름  속성(attribute)  속성(attribute)  속성(attribute)  속성(attribute)

      ↑           ↑
    속성 이름   속성 값
    
```

HTML 문서의 제목

04 HTML 요소 구조.html

+

← → ↺

① 파일 | C:/Users/Administrator/Desktop/hankyung/craw/http기초/04%20H...

☆ ⚙ 80

HTML 요소

태그 이름

속성명

속성값

내용

<p class="para">

한경아카데미빅데이터

</p>

시작 태그

종료 태그

속성은 HTML 요소 중에서도 언제나 시작 태그 내에서만 정의되며, 속성 이름과 속성값(value)으로 표현

문법 구조

<태그이름 속성이름 = "속성값">

1. 아래의 내용을 메모장에 복사 > 파일-다른이름으로저장 -> 표.html

```
<!DOCTYPE html PUBLIC>
<html>
<head>
    <meta charset="EUC-KR">
    <title>초간단 테이블</title>

</head>

<body>
    <p><h1> html표는 tr/td 등의 태그로 작업 </h1> </p>
    <table border="1">
        <tr><!-- 첫번째 줄 시작 -->
            <td>이름</td>
            <td>주소</td>
        </tr>
        <tr><!-- 두번째줄 시작 -->
            <td>
                <br>
            </td>
            <td>
                <br>
            </td>
        </tr>
        <tr>
            <td>김가나</td>
            <td>서울시</td>
        </tr>
        <tr><!-- 두번째 줄 끝 -->
    </table>

</body>
</html>
```

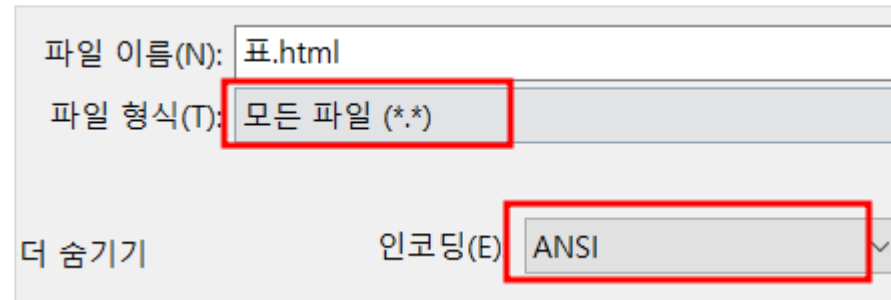


표.html 결과화면

html표는 tr/td 등의 태그로 작업

이름	주소
김가나	서울시

1. 웹데이터 구조 이해 및 파이썬 기초 - (2) 웹크롤링을 위한 Html 구조 이해

CSS(Cascading Style Sheet)

- CSS는 HTML 요소들이 각종 미디어에서 어떻게 보이게 하는지를 정의하는 데 사용되는 스타일 시트 언어
- HTML4 부터는 이러한 모든 서식 설정을 HTML 문서로부터 따로 분리하는 것이 가능해짐
- 오늘날 대부분의 웹 브라우저들은 모두 CSS를 지원하고 있음.
- HTML만으로 웹 페이지를 제작할 경우 HTML 요소의 세부 스타일을 일일이 따로 지정해 주어야만 하며 이 작업은 매우 많은 시간이 걸리며, 완성한 후에도 스타일의 변경 및 유지 보수가 매우 힘들어짐
- 이러한 문제점을 해소하기 위해 W3C(World Wide Web Consortium)에서 만든 스타일 시트 언어가 바로 CSS임
- CSS는 웹 페이지의 스타일을 별도의 파일로 저장할 수 있게 해주므로 사이트의 전체 스타일을 손쉽게 제어할 수 있으며 웹 사이트의 스타일을 일관성 있게 유지할 수 있게 해주며, 그에 따른 유지 보수 또한 쉬워짐
- 이러한 외부 스타일 시트는 보통 확장자를 .css 파일로 저장함.

1

• Inline Style Sheet

HTML 태그의 style 속성에 CSS 코드를 넣는 방법입니다.

2

• Internal Style Sheet

HTML 문서 안의 <style>과 </style> 안에 CSS 코드를 넣는 방법입니다.

• Linking Style Sheet

별도의 CSS 파일을 만들고 HTML 문서와 연결하는 방법입니다.

./HTML이해/표.htm 에 아래의 스타일을 복사하여 추가

```
<!DOCTYPE html PUBLIC>
<html>
<head>
    <meta charset="EUC-KR">
    <title>초간단 테이블</title>

    <style>
        body {background-color: purple;}
        h1, h3 {color:white;}
    </style>

</head>

<body>
    <p><h1> html표는 tr/td 등의 태그로 작업 </h1>

    <p>
        <table border="1", style="color: blue">
            <tr><!-- 첫번째 줄 시작 -->
                <td class="s1">이름
            <td>주소</td>
            </tr>
            <tr><!-- 두번째줄 시작 -->
                <td>김가나</td>
                <td>서울시</td>
            </tr>
            <tr><!-- 두번째 줄 끝 -->
            </tr>
        </table>

</body>
</html>
```

1

2

아래코드를 해석해줘

```
<!DOCTYPE html PUBLIC>
```

```
<html>
```

```
<head>
```

```
<meta charset="EUC-KR">
```

```
<title>초간단 테이블</title>
```

```
</head>
```

```
<body>
```

```
<p><h1> html표는 tr/td 등의 태그로 작업 </h1> </p>
```

```
<table border="1">
```

```
<tr><!-- 첫번째 줄 시작 -->
```

```
<td>이름</td>
```

```
<td>주소</td>
```

```
</tr>
```

1 아래코드를 해석해줘, 입력후 shift+enter

2 html코드 복사붙여넣기

GPT에게
질문하기

3 GPT 응답



ChatGPT

이 HTML 코드는 간단한 표를 생성합니다. 각 부분을 자세히 해석하면 다음과 같습니다:

1. `<!DOCTYPE html PUBLIC>`: 이 문서는 HTML 문서임을 선언합니다.
2. `<html>`: HTML 문서의 시작을 나타냅니다.
3. `<head>`: 문서의 머리 부분을 나타냅니다. 여기에는 메타데이터나 문서 제목 등이 포함됩니다.
4. `<meta charset="EUC-KR">`: 문서의 문자 인코딩을 EUC-KR로 설정합니다. 이는 한국어 인코딩 방식 중 하나입니다.
5. `<title>초간단 테이블</title>`: 브라우저의 제목 표시줄에 표시될 문서의 제목을 설정합니다. 여기서는 "초간단 테이블"입니다.
6. `</head>`: 머리 부분의 끝을 나타냅니다.
7. `<body>`: 문서의 본문 시작을 나타냅니다.
8. `<p><h1> html표는 tr/td 등의 태그로 작업 </h1> </p>`: 큰 제목을 포함하는 단락을 만듭니다. 제목 내용은 "html표는 tr/td 등의 태그로 작업"입니다.
9. `<table border="1">`: 테이블을 생성하며, 테두리의 두께를 1로 설정합니다.
10. `<tr><!-- 첫번째 줄 시작 -->`: 테이블의 첫 번째 행을 시작합니다. HTML 주석 `<!-- 첫번째 줄 시작 -->`는 개발자에게 첫 번째 행의 시작임을 알립니다.

html표는 tr/td 등의 태그로 작업

이름	주소
김가나	서울시

→ ② html 결과물 복사 붙이기

아래결과와 같은 html 작성

→ ① 아래결과와 같은 html작성, 입력후
결과물을 복사 붙여넣기

GPT에게
질문하기

③ GPT 응답



ChatGPT

아래와 같이 HTML 코드를 작성하면, 제공된 이미지와 동일한 결과를 얻을 수 있습니다:

```
html
<!DOCTYPE html PUBLIC>
<html>
<head>
  <meta charset="EUC-KR">
  <title>초간단 테이블</title>
</head>

<body>
  <h1> html표는 tr/td 등의 태그로 작업 </h1>
  <table border="1">
    <tr><!-- 첫번째 줄 시작 -->
      <td>이름</td>
      <td>주소</td>
    </tr>
  </table>
</body>
</html>
```



나
같은 파일내에 css



1 CSS 작성 명령

GPT에게
질문하기

2 GPT 응답

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="EUC-KR">
  <title>초간단 테이블</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }

    h1 {
      font-size: 2em;
    }

    table {
      border-collapse: collapse;
      width: 50%;
      margin-top: 20px;
    }

    table, th, td {
      border: 1px solid black;
    }

    th, td {
      padding: 8px;
      text-align: left;
    }
  </style>
</head>
```



2 GPT 응답

```
<body>
  <h1> html표는 tr/td 등의 태그로 작업 </h1>
  <table>
    <tr>
      <td>이름</td>
      <td>주소</td>
    </tr>
    <tr>
      <td>김가나</td>
      <td>서울시</td>
    </tr>
  </table>
</body>
</html>
```

so 나 → 1 html 작성명령

다음내용의 html 작성

1. h1 태그로 제품가격
2. 4행 3열의 표태그 작성

제품1, a herf태그로 site1, a1.jpg이미지링크
 제품2, a herf태그로 site2 a2.jpg이미지링크
 제품3, a herf태그로 site3 ,a3.jpg이미지링크
 제품4, a herf태그로 site14,a4.jpg이미지링크

2 gpt 응답으로 작성된 html 결과

제품가격

제품	사이트	이미지
제품1	site1	
제품2	site2	
제품3	site3	
제품4	site4	

so 나 → 1 html 작성명령

div로 작성

2 gpt 응답으로 작성된 html 결과

제품가격

제품1 [site1](#) 

제품2 [site2](#) 

제품3 [site3](#) 

제품4 [site4](#) 

1

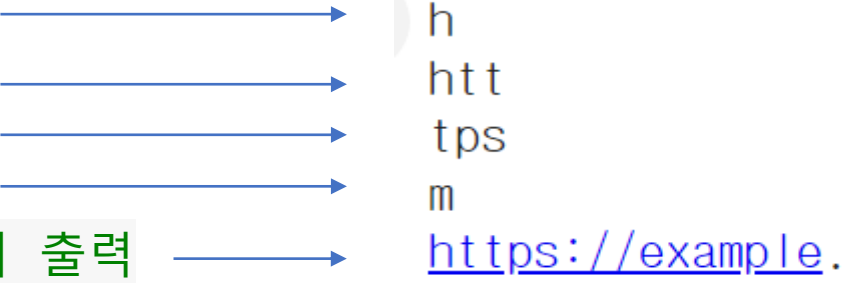
문자열 / 인덱싱과 슬라이싱

웹 페이지의 URL을 처리하거나 HTML 요소에서 텍스트를 추출할 때 문자열 자료형을 자주 사용합니다.

```
url = 'https://example.com'  
title = 'Example Domain'
```

인덱싱: url[0] / 슬라이싱 : url[2:5]

```
print(url[0])  
print(url[:3]) # 0,1,2 인덱싱 출력  
print(url[2:5]) # 2,3,4 출력  
print(url[-1]) # 마지막 자리 출력  
print(url[:-3]) # 0부터 마지막 3자리 전까지 출력
```



h
htt
tps
m
[https://example.](https://example.com)

웹 크롤링을 할 때 문자열 연결은 URL을 구성하거나 데이터를 포매팅하는 데 자주 사용됩니다. 파이썬에서 문자열을 연결하는 방법에는 여러 가지가 있으며, 각 방법은 상황에 따라 적절하게 사용될 수 있습니다.

2

문자열 연결

웹 크롤링을 할 때 문자열 연결은 URL을 구성하거나 데이터를 포맷팅하는 데 자주 사용됩니다. 파이썬에서 문자열을 연결하는 방법에는 여러 가지가 있으며, 각 방법은 상황에 따라 적절하게 사용될 수 있습니다.

```
base_url = 'https://example.com/page'
page_number = '1'
full_url = base_url + page_number
print(full_url)    # Output: https://example.com/page1
```

```
base_url = 'https://example.com/page'
for page_number in range(5):
    full_url = base_url + str(page_number)
    print(full_url)
```

출력물

<https://example.com/page0>
<https://example.com/page1>
<https://example.com/page2>
<https://example.com/page3>
<https://example.com/page4>

GPT에게
질문하기



크롤링에 필요한 문자열 연결 샘플

*

복잡한 URL 생성 샘플 예

크롤링 시 여러 매개변수를 포함하여 URL을 생성해야 할 경우, `urljoin` 및 `urllib.parse` 라이브러리를 사용할 수 있습니다.

```
from urllib.parse import urljoin, urlencode

base_url = 'https://example.com/search'
params = {
    'query': 'python',
    'page': 1,
    'sort': 'relevance'
}
url_with_params = f'{base_url}?{urlencode(params)}'
print(url_with_params)
```

출력물

<https://example.com/search?query=python&page=1&sort=relevance>

4

리스트

여러 개의 데이터를 순차적으로 저장하고 처리할 때 사용되며 여러 개의 링크나 텍스트 데이터를 저장할 때 유용합니다.

```
links = ['https://example.com/page1', 'https://example.com/page2']
for link in links:
    print(link)
```

5

튜플(Tuple)

튜플은 리스트와 비슷하지만, 변경할 수 없는 자료형으로 ()로 표시되며 변경되지 않아야 하는 데이터를 저장할 때 사용됩니다.

```
page_info = ('Example Domain', 'https://example.com')
print(page_info[0])
page_info[0]=1 # 튜플은 값변경이 불가능함
```

Example Domain

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-10-15fe97ae4716> in <cell line: 3>()
      1 page_info = ('Example Domain', 'https://example.com')
      2 print(page_info[0])
----> 3 page_info[0]=1 # 튜플은 값변경이 불가능함
```

TypeError: 'tuple' object does not support item assignment

6

딕셔너리

딕셔너리는 키-값 쌍을 저장할 때 유용하며 크롤링한 데이터를 구조화된 형태로 저장할 때 자주 사용됩니다.

```
page_data = {  
    'title': 'Example Domain',  
    'url': 'https://example.com',  
    'description': 'This domain is for use in illustrative examples in  
documents.'  
}  
print(page_data['title'])
```

GPT에게
질문하기



이름, 나이, 주소 샘플 딕셔너리

GPT에게
질문하기



여러개

GPT에게
질문하기



복잡하게

GPT에게
질문하기



json 으로 저장

7

Set

셋은 중복되지 않은 고유한 값들의 모임을 저장할 때 유용하며 중복된 URL을 제거하거나 유일한 항목을 추출할 때 사용할 수 있습니다.

```
page_data = {  
    'title': 'Example Domain',  
    'url': 'https://example.com',  
    'description': 'This domain is for use in illustrative examples in documents.'  
}  
print(page_data['title'])
```

8

날짜와 시간

크롤링한 데이터에 타임스탬프를 추가하거나 시간과 관련된 데이터를 처리할 때 사용됩니다.

```
from datetime import datetime  
  
now = datetime.now()  
print(now.strftime('%Y-%m-%d %H:%M:%S'))
```

9

정규 표현식 (Regular Expressions)

정규 표현식은 문자열 검색과 치환을 효율적으로 수행할 수 있게 해줍니다. 특히 특정 패턴의 텍스트를 추출할 때 유용합니다.

```
import re
```

```
text = 'Contact us at contact@example.com'
```

```
email_pattern = r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}'
```

```
email = re.findall(email_pattern, text)
```

```
print(email)
```

GPT에게
질문하기



한글을 제외한 모든 문자 제거하는 정규표현식 코드

GPT에게
질문하기



파이썬 정규표현식 대표 패턴

```
import requests
from bs4 import BeautifulSoup
from datetime import datetime
```

```
url = 'https://example-blog.com'
response = requests.get(url)
```

```
if response.status_code == 200:
    html = response.text
    soup = BeautifulSoup(html, 'html.parser')
```

```
posts = []
for post in soup.find_all('div', class_='post'):
    title = post.find('h2').text
    link = post.find('a').get('href')
```

```
    post_data = {
        'title': title,
        'url': link,
        'crawled_at': datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    }
    posts.append(post_data)
```

```
print(posts)
else:
    print("Failed to retrieve the page")
```

리스트, 딕셔너리, 문자열, 그리고 datetime 모듈을 사용하여 크롤링한 데이터를 구조화하고 출력하는 샘플

Part2.

GhatGPT를 활용한 API & 웹 데이터 수집

- (1) 파이썬 http호출 라이브러리
- (2) 판다스의 read_html을 이용한 웹의 표자료 수집
- (3) API를 활용한 데이터 수집
- (4) 네이버 API를 활용한 데이터 수집
- (5) 웹사이트 크롤링
- (6) 동적 웹크롤링 selenium

강사: 부성순 (bakpak@empas.com)

request 모듈

- 파이썬에서 HTTP호출할때 사용되는 대표적인 라이브러리
- 설치방법 : pip install requests
- GET 방식 : requests.get(웹페이지 주소)
- POST 방식 : requests.post(웹페이지 주소)
- PUT 방식 : requests.put(웹페이지 주소)
- DELETE 방식 : requests.delete(웹페이지 주소)

beautifulsoup

- requests모듈을 통하여 가져온 html 데이터에서 원하는 html
- 태그를 추출하게 해주는 라이브러리
- html 태그 이름과 속성을 활용하여 검색
- 설치방법 : pip install beautifulsoup4
- 주로 사용되는 함수 : find(), find_all(), string, get_text(), select()

```
1 from bs4 import BeautifulSoup as bs
2
3 html = """
4 <!DOCTYPE html>
5 <html>
6 <head>
7 <title>BeautifulSoup</title>
8 </head>
9 <body>
10 <div>
11 <p>사과1</p>
12 <p>사과2</p>
13 </div>
14 <div class="ex_class">
15 <p>딸기1</p>
16 <p>딸기2</p>
17 </div>
18 <div id="ex_id">
19 <p>포도1</p>
20 </div>
21 </body>
22 </html>
23 """
24 soup = bs(html, "html.parser")
25 result = soup.find('p')
26 print(result)
27 print(result.text)
28 result_all = soup.find_all('p')
29 print(result_all)
30 result_div_all = soup.find_all('div', class_='ex_class')
31 result_div_all
32
```

<p>사과1</p>

사과1

[<p>사과1</p>, <p>사과2</p>, <p>딸기1</p>, <p>딸기2</p>, <p>포도1</p>]

[<div class="ex_class">

<p>딸기1</p>

<p>딸기2</p>

</div>]

beautifulsoup 라이브러리

주로 사용되는 함수와 속성

find()	가장 먼저 검색되는 html 태그를 찾습니다.
find_all()	해당되는 html 태그를 모두 찾아 리스트로 반환합니다.
select_one()	CSS 선택자로 또는 XPath로 요소를 찾습니다.
select()	CSS 선택자로 또는 XPath로 모든 요소를 찾아 리스트로 반환합니다.
string	태그 객체의 문자열을 추출합니다. 하위에 태그가 있으면 None을 반환합니다.
text	태그 객체의 문자열을 추출합니다. 하위 태그의 문자열도 추출합니다.
get_text()	태그 객체의 문자열을 추출합니다. separator, strip과 같은 파라미터를 제공합니다.
prettify()	파싱한 객체에 들여쓰기를 사용하여 보기 좋게 변환 후 출력합니다.
next_sibling	다음 요소 하나를 추출합니다.
next_siblings	다음 요소 전부를 추출해 generator로 반환합니다.
parent	한 단계 위의 부모 요소만 추출합니다.
parents	모든 부모 요소를 generator로 반환합니다.
find_parent()	조건에 맞는 한 단계 위의 부모 요소만 추출합니다.
find_parents()	조건에 맞는 부모 요소 전부를 추출합니다.

beautifulsoup 라이브러리

`find(name, attrs, recursive=True, text, **kwargs)`

- `name`: 태그 이름
- `attrs`: 적어 넣은 속성과 값이 포함된 요소 탐색. 딕셔너리로 작성
- `recursive`: 하위 태그에서 탐색할지 여부 지정
- `text`: 해당 텍스트가 포함된 요소 탐색
- `**kwargs`: 적어 넣은 속성과 값이 포함된 요소 탐색

```
soup.find('div', {"class":"my-scope", "id"="myid"})  
soup.find('div', class_="my-scope", id="myid")
```

`find_all(name, attrs, recursive=True, text, limit=None, **kwargs)`

- `limit`: 탐색 개수 지정

```
soup.find_all('div', {"class":"my-scope"}, limit=2)  
soup.find_all('div', limit=3, class_="my-scope")
```

beautifulsoup 라이브러리

select_one(selector)

select(selector)

- selector: CSS 선택자 지정 또는 XPATH 지정

```
soup.select_one('div > table > tbody')  
soup.select('//div[@id="header"]')
```

get_text(separator="", strip=False)

- separator: 구분자 지정
- strip: 공백 제거 유무

```
soup.select_one('div > table > tbody').get_text(separator='\n', strip=True)
```

beautifulsoup 라이브러리

find_parent(name, attrs, **kwargs)

find_parents(name, attrs, **kwargs)

- name: 태그 이름
- attrs: 적어 넣은 속성과 값이 포함된 요소 탐색. 딕셔너리로 작성
- **kwargs: 적어 넣은 속성과 값이 포함된 요소 탐색

```
soup.find_parent('div', {"class":"my-scope", "id"="myid"})  
soup.find_parents('div', class_="my-scope", id="myid")
```

parents

- 모든 부모 요소를 generator로 반환
- 한 단계 위의 부모 요소를 포함한 자기 자신을 generator의 첫 번째에 저장 후 또 한 단계 위의 부모 요소를 포함한 자기 자신을 generator의 다음에 저장을 반복

beautifulsoup 라이브러리

HTML 문서에서 요소를 선택하고 처리하기 위해서는 약간의 CSS Selector 또는 XPath 지식을 알아야 합니다.

CSS Selector

- 태그 이름 탐색 : div, p, a
- 아이디 탐색 : #main, #contents
- 클래스 탐색 : .box, .container
- 자식 탐색 : div > p, ul > li > abc
- 속성 지정 탐색 : [id="title"]
- * : 모든 태그를 선택하는 와일드카드

XPath

- // : 문서 내의 모든 요소 탐색
- / : 자식 요소 탐색
- . : 현재 요소
- .. : 현재 요소의 부모 요소 접근
- @ : 요소의 속성 선택
- [] : 조건 지정
- * : 모든 태그를 선택하는 와일드카드

CSS Selector 예시

- 'li', 'p', 'div'
- 'div#contents'
- 'div.section.multiclass'
- '#frm > div > table > tbody'
- 'a[id="title"]', '*[href="link"]'

XPath 예시

- '/html/body/div[1]/div'
- '//div[@class="content"]/p'
- '//*[@id="header"]'
- '//*[@id="header" and @class="content"]'
- '//*[@id="header"][@class="content"]'
- '//*[@class="multiclass1 multiclass2"]'

GPT에게
질문하기

beautifulsoup 모듈을 학습할 txt에 html 샘플 자료

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>BeautifulSoup Example</title>
</head>
<body>
  <div id="container">
    <h1>Welcome to BeautifulSoup Example</h1>
    <p class="description">BeautifulSoup is a Python library for parsing HTML and XML documents.</p>
    <ul>
      <li class="item">Easy to use</li>
      <li class="item">Great for web scraping</li>
      <li class="item">Useful for data extraction</li>
    </ul>
    <div id="nested">
      <h2>Nested Div</h2>
      <p>This is a nested div element.</p>
    </div>
    <a href="https://www.crummy.com/software/BeautifulSoup/bs4/doc/" class="link">Official BeautifulSoup Documentation</a>
  </div>
</body>
</html>
```

결과물

Welcome to BeautifulSoup Example

BeautifulSoup is a Python library for parsing HTML and XML documents.

- Easy to use
- Great for web scraping
- Useful for data extraction

Nested Div

This is a nested div element.

[Official BeautifulSoup Documentation](https://www.crummy.com/software/BeautifulSoup/bs4/doc/)

GPT에게
질문하기

find 태그 샘플

```
from bs4 import BeautifulSoup
# BeautifulSoup 객체 생성
soup = BeautifulSoup(html_doc, 'html.parser')

# <h1> 태그 찾기
h1_tag = soup.find('h1')
print("Found h1 tag:", h1_tag)

# <p> 태그 중 class가 'description'인 것 찾기
p_tag_with_class = soup.find('p', class_='description')
print("Found p tag with class 'description':", p_tag_with_class)

# <div> 태그 중 id가 'nested'인 것 찾기
div_tag_with_id = soup.find('div', id='nested')
print("Found div tag with id 'nested':", div_tag_with_id)

# <a> 태그 찾기
a_tag = soup.find('a')
print("Found a tag:", a_tag)
```

결과물

Found h1 tag: <h1>Welcome to BeautifulSoup Tutorial</h1>

Found p tag with class 'description': <p class="description">BeautifulSoup is a Python library for pulling data out of HTML and XML files.</p>

Found div tag with id 'nested': None

Found a tag: Learn more about BeautifulSoup

**GPT에게
질문하기**

selector 로 변경

```
# BeautifulSoup 객체 생성
soup = BeautifulSoup(html_doc, 'html.parser')

# <h1> 태그 찾기
h1_tag = soup.select('h1')
print("Found h1 tag:", h1_tag)

# <p> 태그 중 class가 'description'인 것 찾기
p_tag_with_class = soup.select('p.description')
print("Found p tag with class 'description':", p_tag_with_class)

# <div> 태그 중 id가 'nested'인 것 찾기
div_tag_with_id = soup.select('div#nested')
print("Found div tag with id 'nested':", div_tag_with_id)

# <a> 태그 찾기
a_tag = soup.select('a')
print("Found a tag:", a_tag)
```

GPT에게
질문하기



class tag find

GPT에게
질문하기



2개 이상의 class

GPT에게
질문하기



내부 css를 작성하고 find