Week 2: Web Application & HTML

Quiz!

다음 함수가 어떤 행동을 하는지 맞춰보세요!

```
def function_a(x, y):
  num = 1
  for i in range(y):
    num = num*x
  return num
```

이번에는 조금 어려울 수 있습니다!

```
def function_b(y, epsilon = 10**(-5)):
    # 각각의 값들을 초기화합니다.
    num = y
    upper_band = y
    lower_band = 0

    # uppder_band와 lower_band의 차이가 매우 작은 수보다 작아질 때까지 반복합니다.
    while upper_band - lower_band >= epsilon:
        if num**2 > y:
            # num의 제곱이 y보다 크다면 upper_band를 num으로 갱신합니다.
            upper_band = num
        elif num**2 < y:
            # num의 제곱이 y보다 작다면 lower_band를 num으로 갱신합니다.
            lower_band = num

        # num을 upper_band와 lower_band의 평균으로 갱신합니다.
        num = (upper_band + lower_band)/2

    return num
```

Web Application

웹 애플리케이션은 기본적으로 Client가 Server에 요청함으로써 이루어집니다.

- 브라우저(Client)가 특정 주소를 가진 컴퓨터(Server)에 웹 페이지를 요청합니다.
- Server 컴퓨터에서는 관련 데이터를 처리하여 웹 페이지를 만들어 보냅니다.
- 우리의 목적이 이러한 역할을 하는 Server를 만드는 것입니다!
- 보여지는 페이지 따로, 관련 데이터 따로 요청할 수도 있습니다. 보여지는 페이지를 다루는 서버의 경우 Frontend, 보이지 않는 데이터 처리를 다루는 경우 Backend라고 합니다.

HTTP

HTTP(HyperText Transfer Protocol)는 client와 server가 정보를 교환하는(요청하고 보내주는) 규약입니다.
Meta 정보를 담은 Header, 요청한 정보를 담은 Body 등의 component로 구성되어 있습니다.

대표적으로 통신의 상태를 나타내는 StatusCode가 있습니다. 자주 보게 될 code는 다음과 같습니다.

- 200: OK. 통신이 성공적입니다.
- 201: CREATED. 데이터가 성공적으로 생성되었습니다.
- 301: PERMANENTLY MOVED. 페이지가 이전되었습니다.
- 400: BAD REQUEST. 주소를 요청하는 형식이 잘못되었습니다.
- 401: UNAUTHORIZED. 페이지에 접근할 권한이 없습니다.
- 404: NOT FOUND. 요청한 주소를 찾을 수 없거나, 요청한 정보를 찾을 수 없거나...
- 500: INTERNAL SERVER ERROR. 요청한 주소의 컴퓨터가 제대로 작동하지 않습니다.
- ...

HTML

HTML(HyperText Markup Language)은 웹 페이지를 구성하는 언어입니다. 브라우저에서 보여주는 모든 웹 페이지는 이러한 HTML로 구성된 문서라고 볼 수 있겠습니다.

- HyperText: 연결된 텍스트를 의미합니다.
- Markup Language: 마크업 언어는 태그 등을 이용해 문서의 구조를 명기하는 언어입니다.

즉 HTML은 HyperText로 구성된 웹 페이지를 마크업하는 언어라고 보시면 되겠습니다. 용도가 한정적이기 때문에 programming language로 취급받지는 않습니다.

HTML – Tag

HTML은 미리 약속된 태그들을 통해 웹 페이지를 구성합니다. 예를 들어 `h1` 태그는 "첫 번째 제목"을 표현하며, `a` 태그는 linked text를 표현하며, `ol` 는 순서가 있는 목록(ordered list)을 표현합니다.

- 태그는 `<tag>` 로 열고, `</tag>` 로 닫습니다. 그 사이에는 표현하고자 하는 컨텐츠가 들어갑니다.

```
<h1>This is Title!</h1>
```

- 태그는 중첩될 수 있습니다.

```
<div>
  <h1>This is Title!</h1>
  <div>
    <h2>Ordered List</h2>
    <ol>
      <li>List Item 1</li>
      <li>List Item 2</li>
      <li>List Item 3</li>
    </ol>
  </div>
</div>
```

- 스스로 닫는 태그도 있습니다.

```
<br/> 한 칸을 띄우는 태그입니다.
<image source="https://address.com/img_address.png" height="30"
width="30" /> 이미지를 표현하는 태그입니다.
```

이러한 HTML 언어는 크게 `Head` , `Body` , `Script` 세 부분으로 이루어져 있습니다.

```
<html>
  <head>
    웹 페이지의 메타 데이터가 들어갈 곳입니다.
    <style>웹 페이지의 스타일이 들어갈 곳입니다.</style>
  </head>
  <body>
    웹 페이지의 내용이 들어갈 곳입니다.
    <script>
      자바스크립트로 쓰여진, 웹 페이지를 동적으로 다루는 스크립트 코드가 들어갈 곳입니다.
    </script>
  </body>
</html>
```

HTML - Head

Head 태그 안에는 웹 페이지의 메타 데이터가 태그들을 통해 표현됩니다. 메타 데이터는 "데이터에 대한 데이터"라고 볼 수 있습니다. 즉 웹 페이지의

- 제목: `<title>This is Title</title>`
- 설명: `<meta name="description" content="This is Description" />`
- 인코딩 규칙: `<meta charset="utf-8" />`
- 스타일: `<style>styles...</style>`

등이 그 예시가 되겠습니다.

HTML - Body

웹 페이지의 내용이 태그들을 통해 표현됩니다. 모든 태그를 외우는 것은 불가능합니다. 자주 쓰는 것의 태그 이름만 외워둔 후, 사용법을 구글에 찾아 사용하는 것이 일반적입니다. 다음은 자주 사용하는 태그들의 목록입니다.

- `div` : 태그들을 하나로 묶기 위한 태그입니다. 가장 많이 쓰이고, 가장 기본적인 태그입니다.
- `h1, h2, ..., h6` : 제목을 위한 태그입니다. 폰트가 크고 진하게 표현됩니다. `h1` 이 가장 크고, `h6` 이 가장 작습니다.
- `ol` : 순서를 가진 목록을 표현하기 위한 태그입니다. 자동으로 숫자를 붙여줍니다.
- `ul` : 순서가 없는 목록을 표현하기 위한 태그입니다.
- `li` : 목록의 하나 하나를 표현하기 위한 태그입니다. 주로 `ol` 이나 `ul` 의 자식(하위 태그)로 사용됩니다.
- `a` : 링크된 텍스트를 표현하기 위한 태그입니다. 흔히 보는 링크입니다.
- `p` : 문단을 표현하기 위한 태그입니다. Paragraph의 약자입니다. 문단 간격을 살짝 띄워 줍니다.
- `button` : 버튼을 표현하기 위한 태그입니다. `onclick` parameter로 JavaScript function을 제공하여 해당 함수를 실행시킬 수 있습니다.

```html
<!-- introduction.html -->
<html>
    <head>
        <title>Introduction</title>
    </head>
    <body>
        <div id="intro-header">
            <h1>Introduction</h1>
            <h3>Introduction to Econometrics</h3>
        </div>
        <div id="intro-body">
            <ul>
                <li>
                    <a href="https://blog.jesse.kim">
                        이 링크를 클릭하면 블로그로 이동합니다.
                    </a>
                </li>
                <li>
                    <a
href="https://github.com/weekendkim/coding_kindergarten">
                        이 링크를 클릭하면 깃헙 페이지로 이동합니다.
                    </a>
                </li>
            </ul>
            <textarea
                cols="100"
                rows="10"
                placeholder="이곳에 글을 입력할 수 있습니다."
            >
            </textarea>
        </div>
        <div>
            <button onclick="alert('Hello!')">버튼을 클릭하세요!</button>
        </div>
    </body>
</html>
```

과제 3

서로 다른 태그를 5개 이상 사용해 자기 소개 페이지를 하나 만들어보세요!

- *.html 파일을 하나 만들어 속을 채웁니다.
- `html` , `head` , `body` , `script` 태그는 제외합니다.
- `h#` 태그는 하나로 칩니다.

HTML - Script

HTML 언어로는 정적인 웹 페이지만을 만들어낼 수 있습니다. 정적인 웹 페이지는 동적이지 않은 웹 페이지라고 정의할 수 있겠습니다. 동적인 웹 페이지는 유저의 action에 따라 dynamic하게 변할 수 있는 웹 페이지를 의미합니다. 예컨대 버튼을 클릭했을 때 페이지의 일부분의 바뀐다던가, 시간이 지남에 따라 페이지가 바뀐다던가 하는 요소들이 웹 페이지를 동적으로 만듭니다.

JavaScript는 브라우저 위에서 작동할 수 있는 프로그래밍 언어입니다. 이러한 JavaScript의 특성을 이용해 웹 페이지를 동적으로 바꿀 수 있습니다. 예를 들어, `button` 태그의 "onclick" parameter는 JavaScript function을 인수로 받아, 버튼이 클릭될 때마다 해당 함수를 실행시킵니다.

```
<html>
  <body>
    <button onclick="alert('hello!')">
      이 버튼을 클릭하면 알람을 줍니다!
    </button>
  </body>
</html>
```

혹은 미리 정의해 놓은 함수를 불러와 실행할 수도 있습니다. 이러한 JavaScript 코드는 `body` 태그의 `script` 태그 안에 작성합니다.

웹 페이지를 렌더링할 때, 브라우저는 `script` 태그 안의 코드를 먼저 쭉 읽어 놓습니다!

```
<html>
  <body>
    <button onclick="increase_value()">
      버튼을 클릭하세요!
```

```
        </button>

        <script>
          let i = 0;

          function increase_value() {
            i++;
            alert(i);
          }
        </script>
      </body>
    </html>
```

Document

웹 페이지에는 JavaScript의 `document` 라는 object가 정의되어 있습니다. 이 `document` 오브젝트에는 웹 페이지를 제어하기 위한 method들이 구현되어 있습니다.

- `document.getElementById("element_id")` : id 값을 통해 element를 불러옵니다.
- `document.createElement("tag_name")` : 입력된 태그로 이루어진 element를 하나 생성합니다. 이렇게 생성된 element를 변수에 저장하여 다른 element에 추가할 수도 있습니다.
- `document.createTextNode("text")` : 텍스트로 이루어진 element를 하나 생성합니다.
- ...

Element

또 HTML의 tag로 구성된 요소 하나하나를 element라고 합니다. 위의 `document.getElementById()` 메서드가 `id` 로 element를 불러오는 것이었습니다. 이러한 element는 하나의 JavaScript object이고, 역시 스스로를 제어하기 위한 method들이 구현되어 있습니다.

- `element.innerHTML` : element의 컨텐츠를 지칭합니다.
- `element.parentNode` : element가 속한 상위 element를 지칭합니다.
- `element.appendChild("child_element")` : element에 하위 element를 추가합니다.
- `element.setAttribute("param", "param_value")` : element에 parameter를 추가/변경합니다.
- `element.removeChild("child_element")` : element의 하위 element를 삭제합니다.
- ...

이제 다음과 같이 리스트에 항목을 추가하는 script를 작성할 수 있습니다.

```
<html>
    <body>
        <div>
            <button onclick="add_item()">
                버튼을 누르면 항목이 추가됩니다.
            </button>
        </div>

        <div>
            <ol id="my-list">
                <li>이 밑에 항목을 추가해봅시다!</li>
            </ol>
        </div>

        <script>
            // 리스트에 항목을 추가해주는 함수입니다.
            function add_item() {

                // 리스트 element를 불러와 변수에 저장합니다.
                const list = document.getElementById("my-list");

                // 새로운 li element를 생성해 변수에 저장합니다.
                const new_item = document.createElement("li");

                // 유저의 입력을 받아 변수에 저장합니다.
                const input_text = prompt("텍스트를 입력하세요!");

                // 입력받은 텍스트를 이용해 새로운 text element를 생성해 변수에 저장합니
다.
                const new_item_text = document.createTextNode(input_text);

                // 생성된 li element에 text node를 삽입합니다.
                new_item.appendChild(new_item_text);

                // li element를 리스트 element에 삽입합니다.
                list.appendChild(new_item);
            }
        </script>
    </body>
</html>
```

This

JavaScript에서 `this` 는 자기 자신(오브젝트)를 지칭합니다.

```
<html>
    <body>
        <div>
            <button onclick="hello(this)">버튼1</button>
            <button onclick="hello(this)">버튼2</button>
            <button onclick="hello(this)">버튼3</button>
        </div>

        <script>
            function hello(element) {
                const button_text = element.innerText;
                alert(button_text);
            }
        </script>
    </body>
</html>
```

과제 4: Todo Application

이제 Todo 앱을 한 번 만들어봅니다! 우리의 Todo 앱에서는 세 가지의 기능이 있습니다.

- 새로운 할 일을 등록할 수 있습니다.
  - `id` 가 `"todo_list"` 인 element에 새로운 `li` element를 추가함으로써 가능합니다.
  - 이 때, 새로 생성된 `li` element에는 "Done" 버튼과 "Delete" 버튼이 포함되어 있어야 합니다.
  - `button` 의 `onclick` parameter는 다음과 같이 추가할 수 있습니다.

    ```
    element.setAttribute("onclick", "FUNCTION_TO_EXECUTE(this)")
    ```

- 할 일을 삭제할 수 있습니다. (Hint: element.parentNode를 이용하면 편합니다.)
- 할 일을 완료할 수 있습니다. 완료된 할 일에는 ~~strike 표시~~가 됩니다. 이 함수는 `css` 를 알아야 하기 때문에 구현해 놓았습니다.

다음의 빈 부분들을 채워 Todo 앱을 완성해보세요.

```
<html>
    <head>
        <title>TODO</title>
    </head>
    <body>
        <h1>My first TODO Application</h1>
        <ol id="todo_list">
            <li>
                Breathe
                <button onclick="complete_todo(this)">Done</button>
                <button onclick="delete_todo(this)">Delete</button>
            </li>
        </ol>
        <button onclick="add_todo()">Add TODO</button>

        <script>

            // Todo 항목을 추가하는 함수입니다.
            function add_todo() {

                // 새로운 li element를 생성해 변수에 저장합니다.
                const new_todo_item = document.createElement("li");

                // 유저의 입력을 받아 변수에 저장합니다.
                const new_todo_text = prompt("What to do?");

                // 입력이 없을 경우 함수를 종료합니다.
                if (!new_todo_text) {
                    return false;
                }

                // 텍스트 노드를 생성하여 li element(new_todo_item)에 삽입합니다.
                // 이 부분을 채워주세요!

                // "Done" 버튼을 생성하여 li element(new_todo_item)에 삽입합니다.
                // 이 부분을 채워주세요!

                // "Delete" 버튼을 생성하여 li element(new_todo_item)에 삽입합니다.
                // 이 부분을 채워주세요!

                // todo_list element의 하위 element로 li element를 추가합니다.

document.getElementById("todo_list").appendChild(new_todo_item);
            }
```

```
            // Todo 항목을 삭제하는 함수입니다.
            const delete_todo = function(element) {
                // 선택된 element를 parentNode에서 삭제합니다.
                // 이 부분을 채워주세요!
            }

            // Todo 항목을 완료하는 함수입니다. 항목에 strike 표시를 합니다.
            const complete_todo = function(element) {
                const parent_list = element.parentNode;
                parent_list.setAttribute("style", "text-decoration: line-
through")
            }
        </script>
    </body>
</html>
```

HTML: Style

`style` 태그를 통해 각 element의 style을 지정할 수 있습니다. 즉 예쁘게 만들 수 있습니다. 이러한 style을 지정하는 언어를 CSS(Cascading Style Sheet)라 합니다.

```
<html>
    <head>
        <style>
            .XL {
                font-size: 36px;
            }

            #box {
                border: 1px solid black;
            }

            .underlined {
                text-decoration: underline;
            }

            .red {
```

```
                color: red;
            }

            #box .blue {
                color: blue;
            }
        </style>
    </head>
    <body>
        <div class="XL">XL size text</div>
        <div id="box">
            <div class="underlined">Underlined text</div>
            <div class="red">Red text</div>
            <div class="italic">Italic text</div>
            <div class="blue">This text is blue only in the box</div>
        </div>
        <div class="blue">This text is blue only in the box</div>
        <div style="margin-top: 24px;">Text with top margin</div>
    </body>
</html>
```

- `.CLASS_NAME {}` 은 "CLASS_NAME" 라는 `class` 로 지정된 element에 대한 style을 지정합니다.
- `#ID {}` 은 "ID"라는 `id` 로 지정된 element에 대한 style을 지정합니다.
- `#ID .CLASS_NAME {}` 은 "id" element 하위에 있는 "CLASS_NAME" class element에만 적용됩니다.
- style은 하위의 element에게도 물려집니다.
- 하지만 하위의 style이 상위의 style을 이깁니다. (중복될 경우)
- 주로 사용하는(?) style들은 다음과 같습니다.
  - `text-decoration` : underline, strike 등 텍스트를 꾸밉니다.
  - `font-size` : 폰트 사이즈를 지정합니다.
  - `color` : 폰트의 색상을 지정합니다.
  - `text-align` : 텍스트를 좌/우/중앙으로 정렬합니다.
  - `margin` : element의 margin을 지정합니다.
  - `padding` : element의 padding을 지정합니다.
  - ...

CSS는 굉장히 많고, 복잡하고, 빡치기 쉬운 분야입니다. 인내심을 가지고 임하셔야 합니다!