

# Flask

```
0 response = requests.get(url)
1
2 # checking response.status_code (if you get
3 if response.status_code != 200:
4     print(f"Status: {response.status_code}")
5 else:
6     print(f"Status: {response.status_code}")
7
8 # using BeautifulSoup to parse the response
9 soup = BeautifulSoup(response.content, "html.parser")
10
11 # finding Post images in the soup
12 images = soup.find_all("img", attrs={"alt": "Post image"})
13
14 # downloading images
15 counter = 0
16 for image in images:
17     url = image.get("src")
18     response = requests.get(url)
19     with open(f"image_{counter}.jpg", "wb") as f:
20         f.write(response.content)
21     counter += 1
22 
```

**1. Flask 초기 세팅**

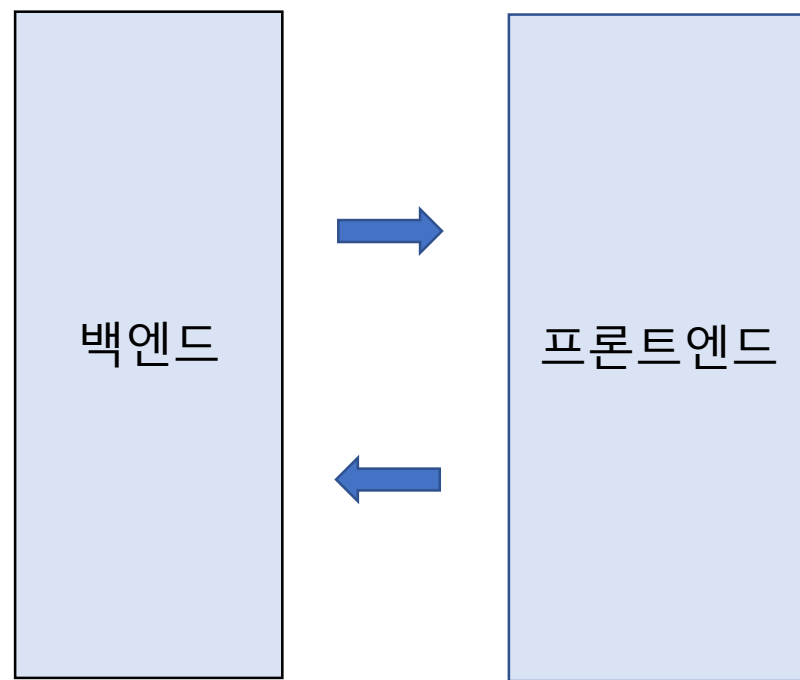
**2. Flask와 템플릿(HTML), CSS 확장**

**3. DataBase 연결**

**4. 로그인 기능 구현**

**5. 게시판 페이지 확장**

# 웹 프로그래밍이란?



**1. 프론트엔드: 웹 사용자에게 보여지는 화면을 디자인 및 기능을 작성**

1) 디자인: css, html

2) 기능담당: javascript, ajax 등등

# 웹 프로그래밍이란?

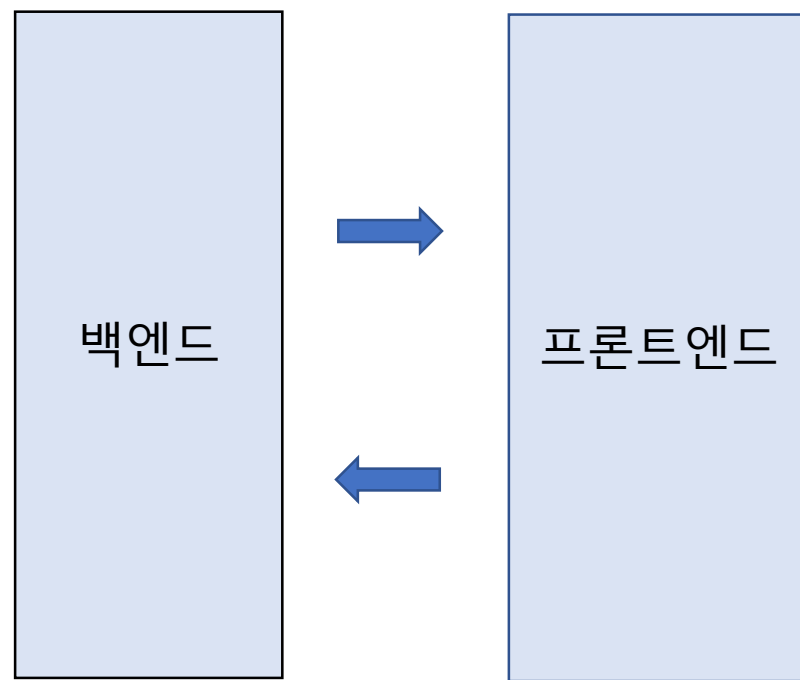


1. 프론트엔드: 웹 사용자에게 보여지는 화면을 디자인 및 기능을 작성

1) 디자인: css, html

2) 기능담당: javascript, ajax 등등

# 웹 프로그래밍이란?



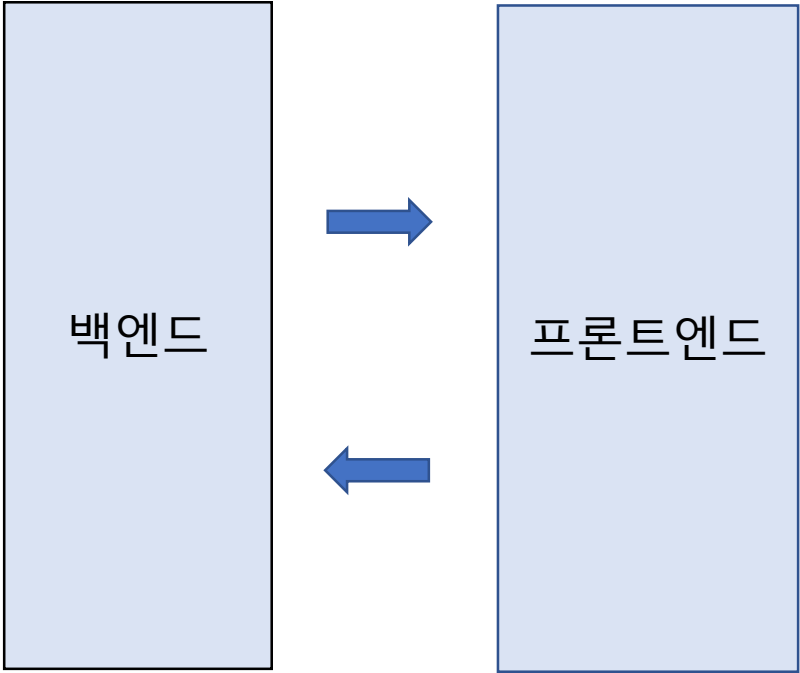
## 2. 백엔드: 웹에서 필요한 정보들을 공급, 처리하는 부분

첫번째 예: 로그인을 하려면 백엔드에 기존 회원 정보를 가지고 있어야 함 (DB 조회)

두번째 예: 회원가입을 할때 내 정보를 백엔드로 보내야함 (DB 삽입)

# 웹 프로그래밍이란?

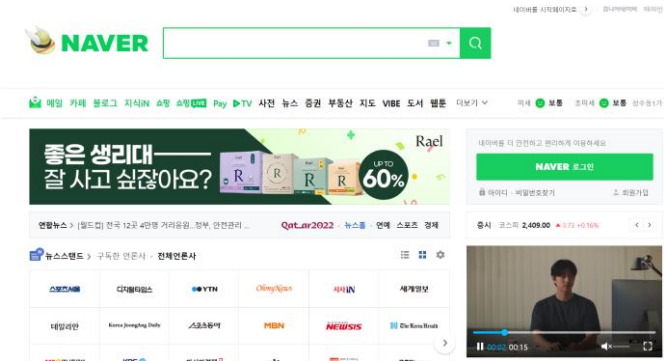
서울 성수



연결수단: IP

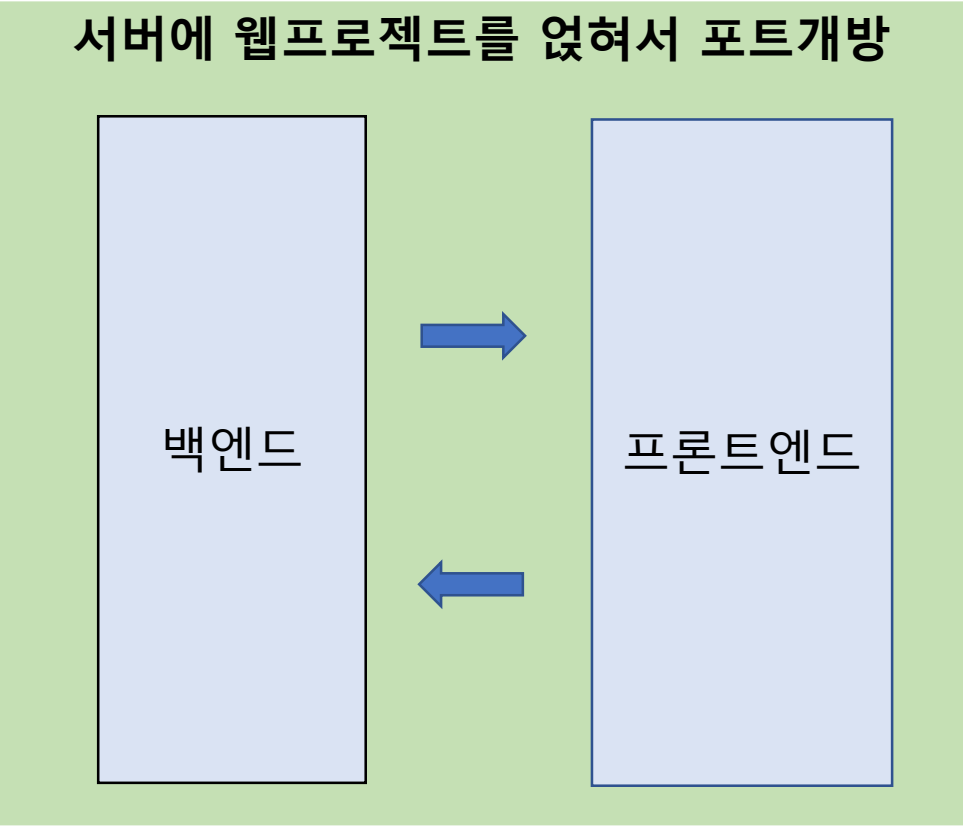


부산



# 웹 프로그래밍이란?

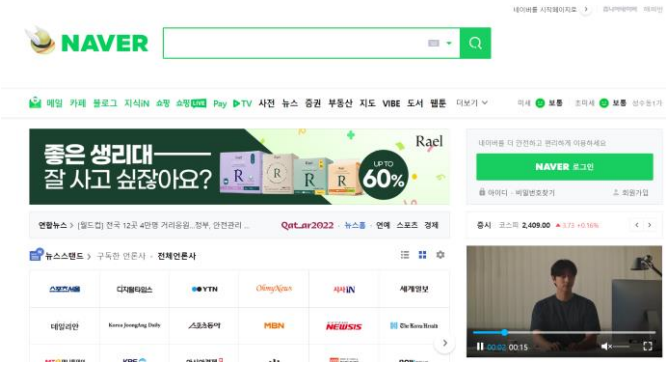
서울 성수



연결수단: IP와 포트



부산



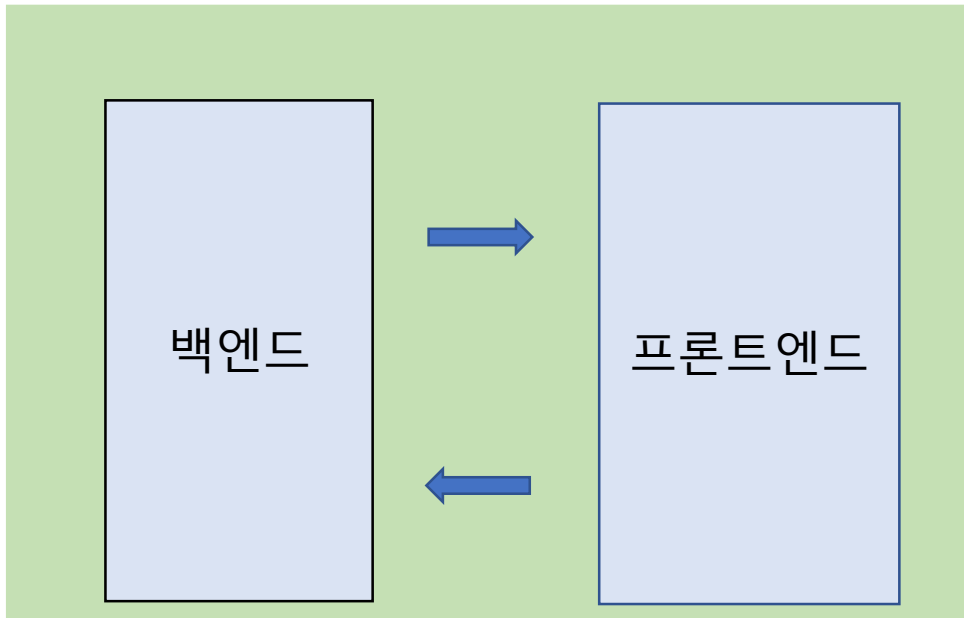
3. 서버: 웹 프로젝트를 외부에 공개하려면 '포트와 아이피'를 열어야함 → 서버를 이용

# 웹 프로그래밍이란?

사용자 PC 영역: IP는 localhost 또는 123.123.123 일 경우

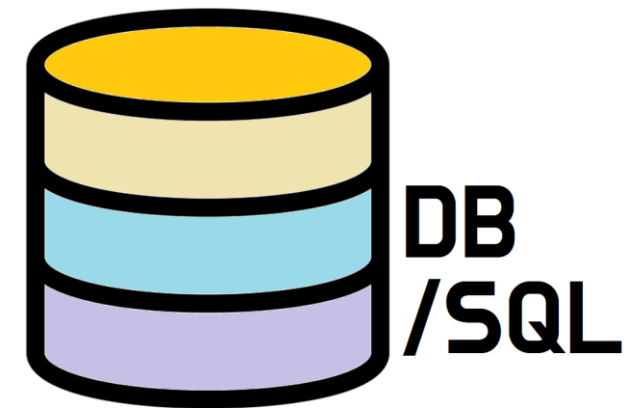
내부접근: localhost:80  
외부접근: 123.123.123:80

서버 (아파치 톰캣 서버일 경우: 80포트)



내부접근: localhost:3306  
외부접근: 123.123.123:3306

Database (Mysql인 경우: 3306포트)







# Flask?

파이썬으로 작성된 마이크로 웹 프레임워크  
Micro Web Server Framework



**쓰는 이유:** 1. 타 프레임워크에 비해 간결하다.  
2. 자체 Back-End WAS 서버가 있다.

# 실습

```
0 response = requests.get(url)
1
2 # checking response.status_code (if you get
3 if response.status_code != 200:
4     print(f"Status: {response.status_code}")
5 else:
6     print(f"Status: {response.status_code}")
7
8 # using BeautifulSoup to parse the response
9 soup = BeautifulSoup(response.content, "html.parser")
10
11 # finding Post images in the soup
12 images = soup.find_all("img", attrs={"alt": "Post Image"})
13
14 # downloading images
15 for image in images:
16     url = image.get("src")
17     if url:
18         response = requests.get(url)
19         with open(f"image_{url}.jpg", "wb") as f:
20             f.write(response.content)
21
22
```

# Flask 초기 세팅

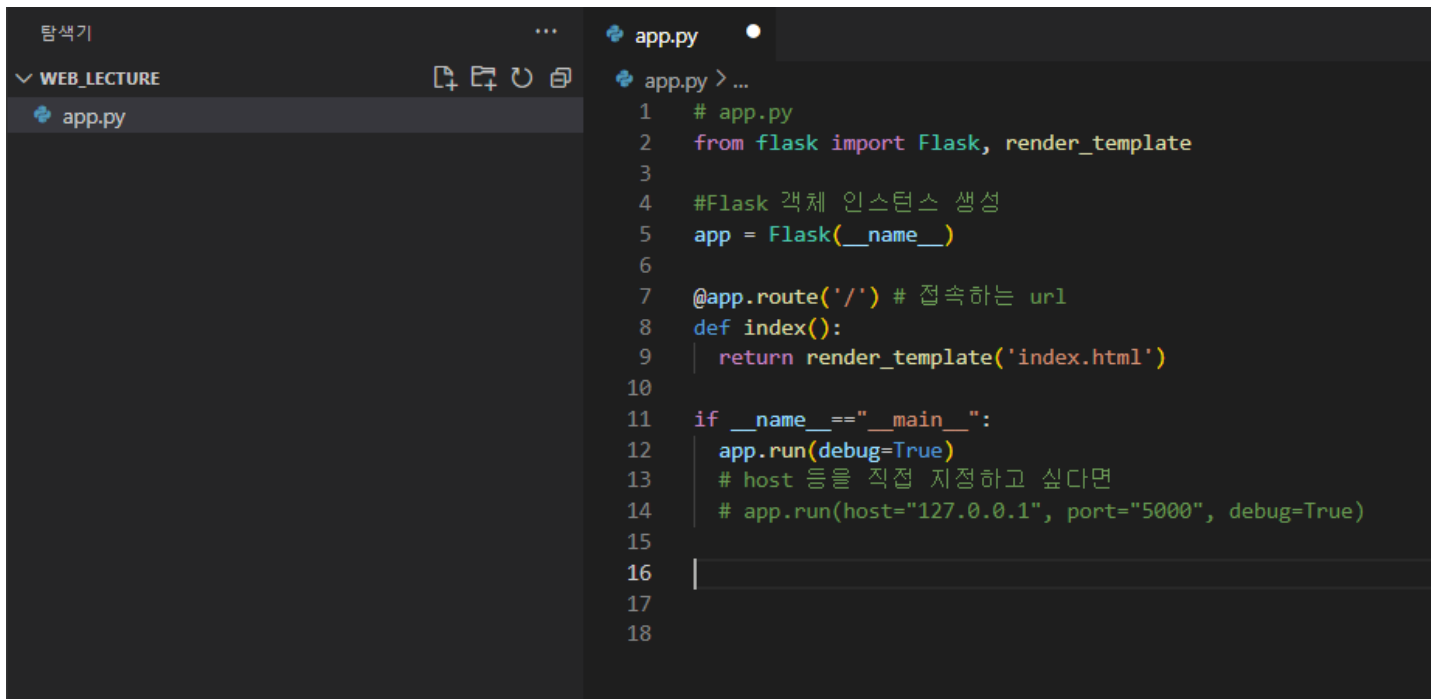
## 1. 웹 환경 세팅 (아나콘다 가상환경 이용):

conda create -n 가상환경이름 python=3.8.3

## 2. Flask 설치 (가상환경 **활성** 후):

pip install flask

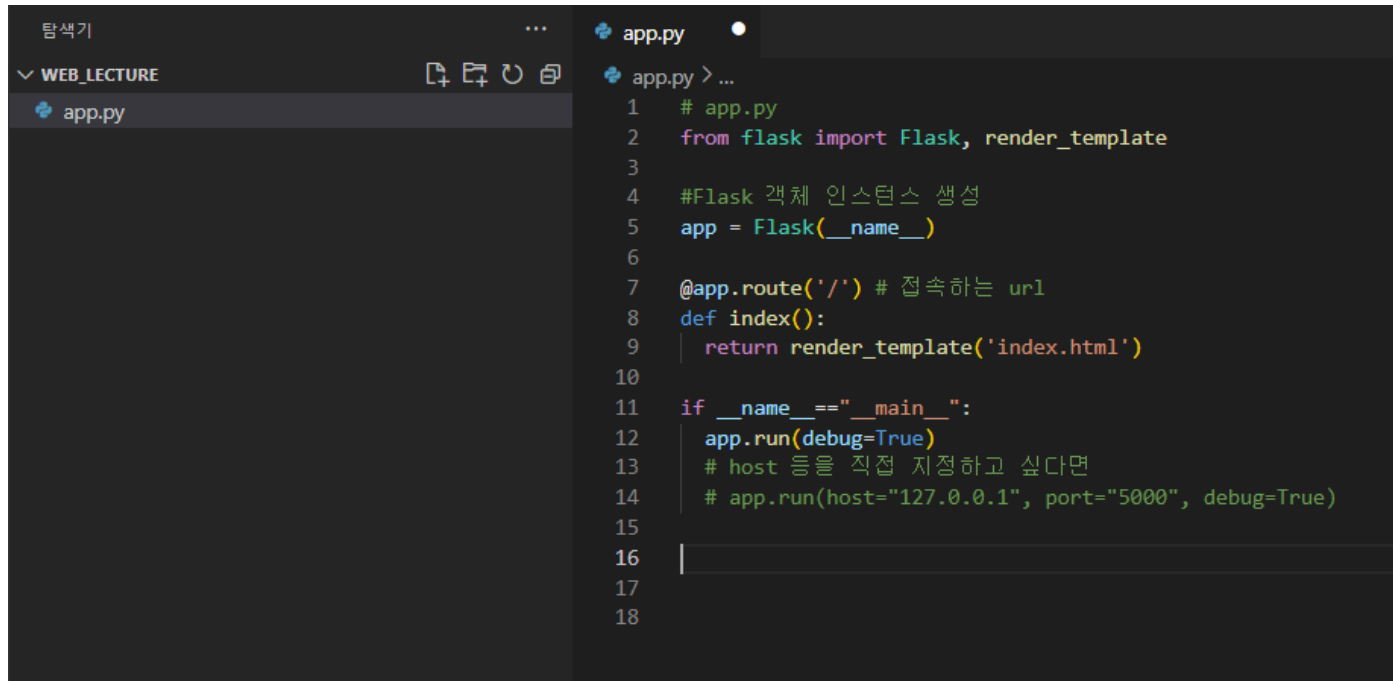
## 3. 프로젝트 생성 및 py파일 생성 및 작성:



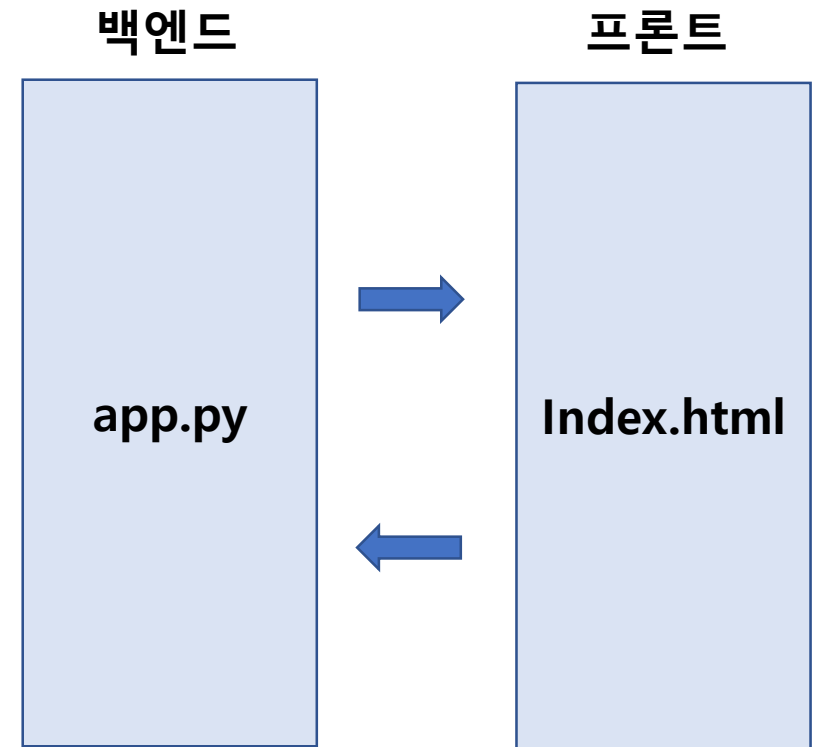
```
1  # app.py
2  from flask import Flask, render_template
3
4  #Flask 객체 인스턴스 생성
5  app = Flask(__name__)
6
7  @app.route('/') # 접속하는 url
8  def index():
9      return render_template('index.html')
10
11 if __name__ == "__main__":
12     app.run(debug=True)
13     # host 등을 직접 지정하고 싶다면
14     # app.run(host="127.0.0.1", port="5000", debug=True)
15
16
17
18
```

# Flask 초기 세팅

## 1. app.py파일 분석:



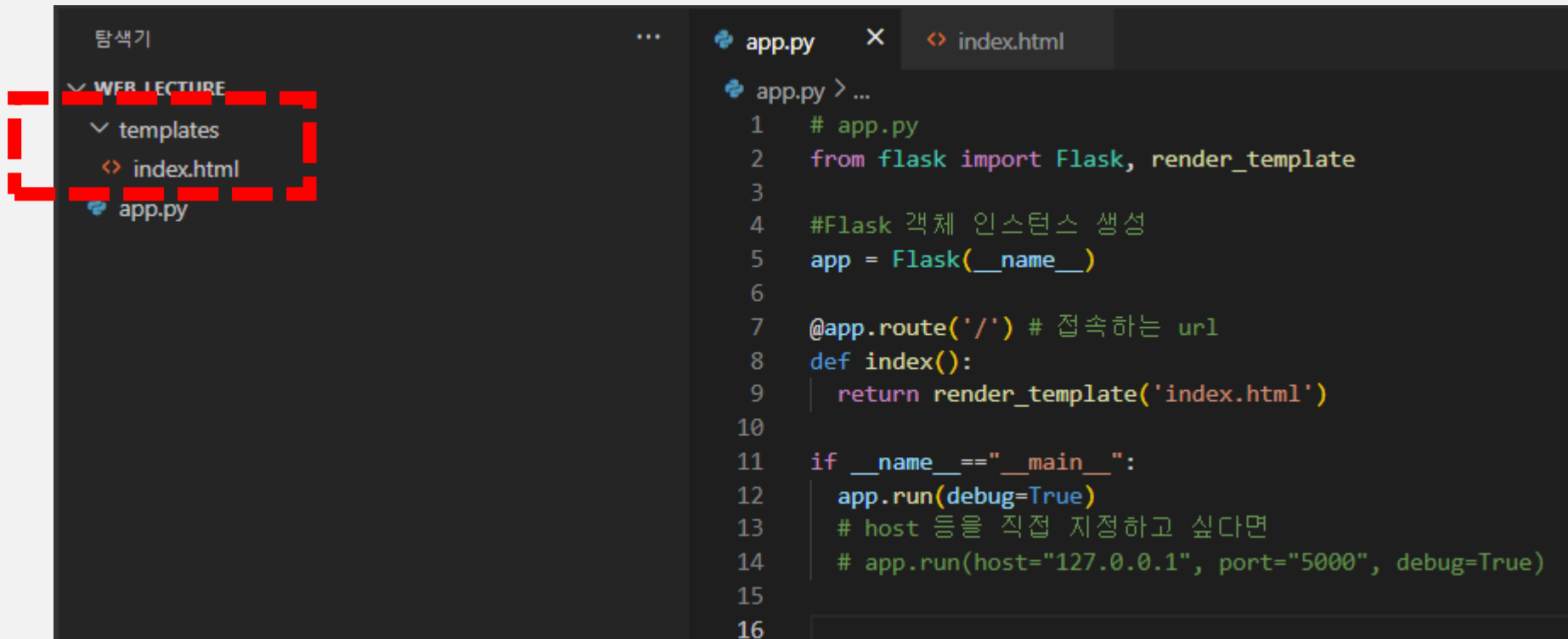
```
1 # app.py
2 from flask import Flask, render_template
3
4 # Flask 객체 인스턴스 생성
5 app = Flask(__name__)
6
7 @app.route('/') # 접속하는 url
8 def index():
9     return render_template('index.html')
10
11 if __name__ == "__main__":
12     app.run(debug=True)
13     # host 등을 직접 지정하고 싶다면
14     # app.run(host="127.0.0.1", port="5000", debug=True)
15
16
17
18
```



# Flask 초기 세팅

## 1.2 app.py파일과 연동할 템플릿 만들기:

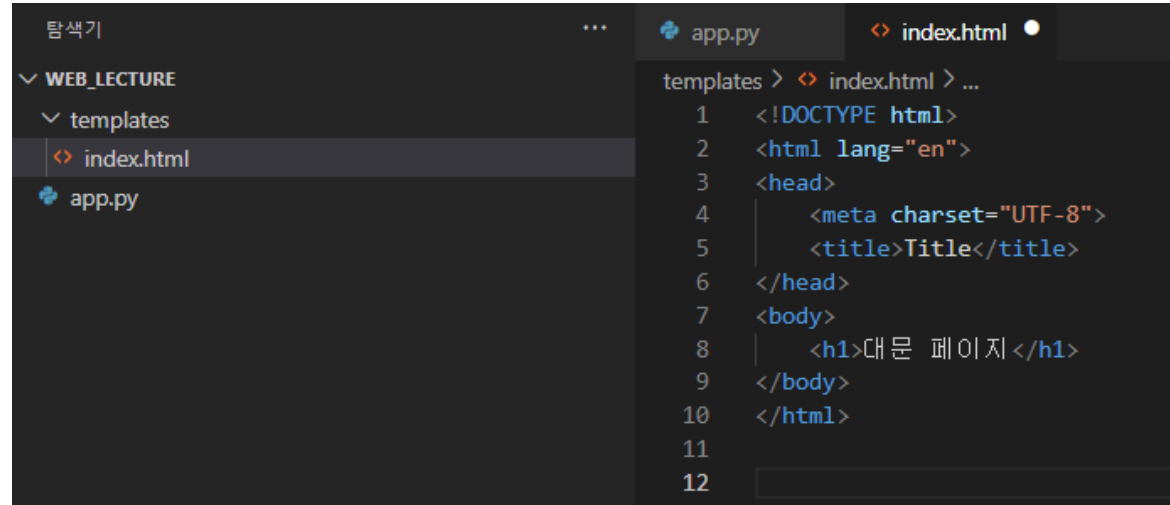
### Templates 폴더 생성 및 index.html 파일 생성



```
app.py > ...
1  # app.py
2  from flask import Flask, render_template
3
4  # Flask 객체 인스턴스 생성
5  app = Flask(__name__)
6
7  @app.route('/') # 접속하는 url
8  def index():
9      return render_template('index.html')
10
11 if __name__ == "__main__":
12     app.run(debug=True)
13     # host 등을 직접 지정하고 싶다면
14     # app.run(host="127.0.0.1", port="5000", debug=True)
15
16
```

# Flask 초기 세팅

## 1.3 html 작성 및 브라우저 화면 띄워보기:



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'WEB\_Lecture' with a 'templates' folder containing 'index.html' and a file named 'app.py'. The code editor shows the content of 'index.html' with the following HTML code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Title</title>
6 </head>
7 <body>
8   <h1>대문 페이지</h1>
9 </body>
10 </html>
11
12
```



http://127.0.0.1:5000



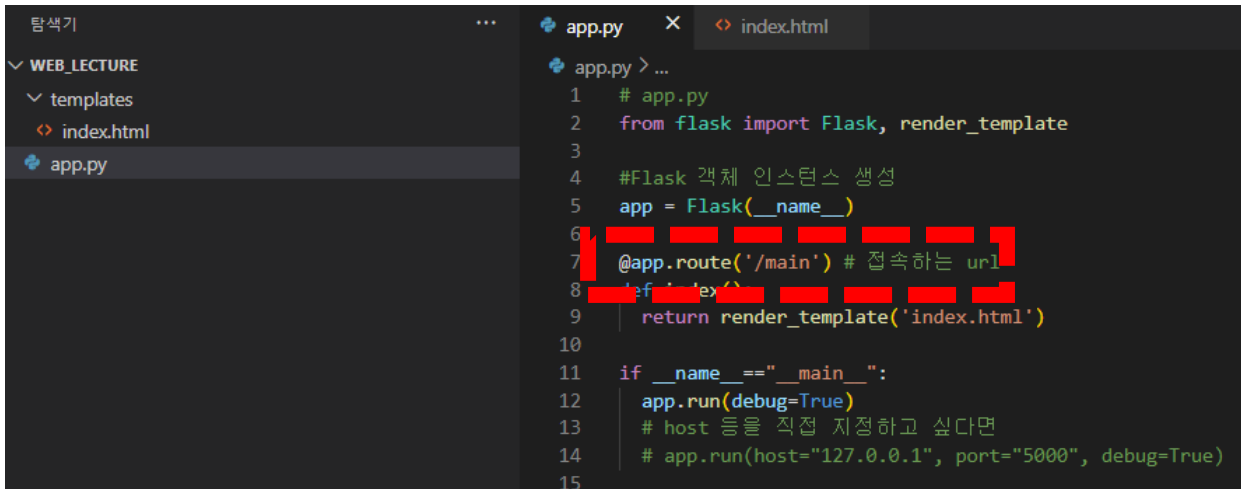
대문 페이지

# Flask 초기 세팅

## 1.4 경로 한번 바꿔보기:

`App.route('/') → App.route('/main')`

`http://127.0.0.1:5000/main`



```
1 # app.py
2 from flask import Flask, render_template
3
4 # Flask 객체 인스턴스 생성
5 app = Flask(__name__)
6
7 @app.route('/main') # 접속하는 url
8 def index():
9     return render_template('index.html')
10
11 if __name__ == "__main__":
12     app.run(debug=True)
13     # host 등을 직접 지정하고 싶다면
14     # app.run(host="127.0.0.1", port="5000", debug=True)
15
```



대문 페이지



POST, GET, 값 넘기기

```
0 response = requests.get(url)
1
2 # checking response.status_code (if you get
3 if response.status_code != 200:
4     print(f"Status: {response.status_code}")
5 else:
6     print(f"Status: {response.status_code}")
7
8 # using BeautifulSoup to parse the response
9 soup = BeautifulSoup(response.content, "html.parser")
10
11 # finding Post images in the soup
12 images = soup.find_all("img", attrs={"alt": "Post Image"})
13
14 # downloading images
15 for image in images:
16     url = image.get("src")
17     response = requests.get(url)
18     with open(f"image_{url}.jpg", "wb") as f:
19         f.write(response.content)
20
21 # printing the downloaded images
22 for image in images:
23     url = image.get("src")
24     print(f"Downloaded image: {url}")
```

# POST, GET 값 넘기기

## 1. GET:

- GET은 서버의 리소스에서 데이터를 요청할 때, 데이터를 읽거나(Read), 검색(Retrieve)할 때
- GET 방식 경우에는 브라우저마다 글자 수 제한이 있기 때문에 **게시판의 게시물, 목록 조회와 같은 간단한 데이터 요청**할 때 적합

※ 예시 스토리: 특정 게시물 화면으로 이동한다면?

사용자 입력 또는 링크 이동

http://127.0.0.1:5000/**Content?uid=osh&cid=30**  
?파라미터=값&파라미터=값

해석: osh가 쓴 게시물번호 30번의 게시물을 가져와라

백엔드 (app.py)

**DB에서  
게시판  
내용  
가져오기**

프론트 뷰

**게시판.html**

# POST, GET 값 넘기기

## 1. GET 실습:

코드를 아래와 같이 작성

http://127.0.0.1:5000/main?uid=osh&cid=30



```
app.py > ...
1 # app.py
2 from flask import Flask, render_template
3 from flask import request
4
5 #Flask 객체 인스턴스 생성
6 app = Flask(__name__)
7
8 @app.route('/main') # 접속하는 url
9 def index():
10     temp = request.args.get('uid')
11     temp1 = request.args.get('cid')
12
13     print(temp, temp1)
14
15     return render_template('index.html')
```



```
* Debugger is active!
* Debugger PIN: 516-527-692
osh 30
127.0.0.1 - - [07/Dec/2022 14:52:48] "GET /main?uid=osh&cid=30 HTTP/1.1" 200 -
```

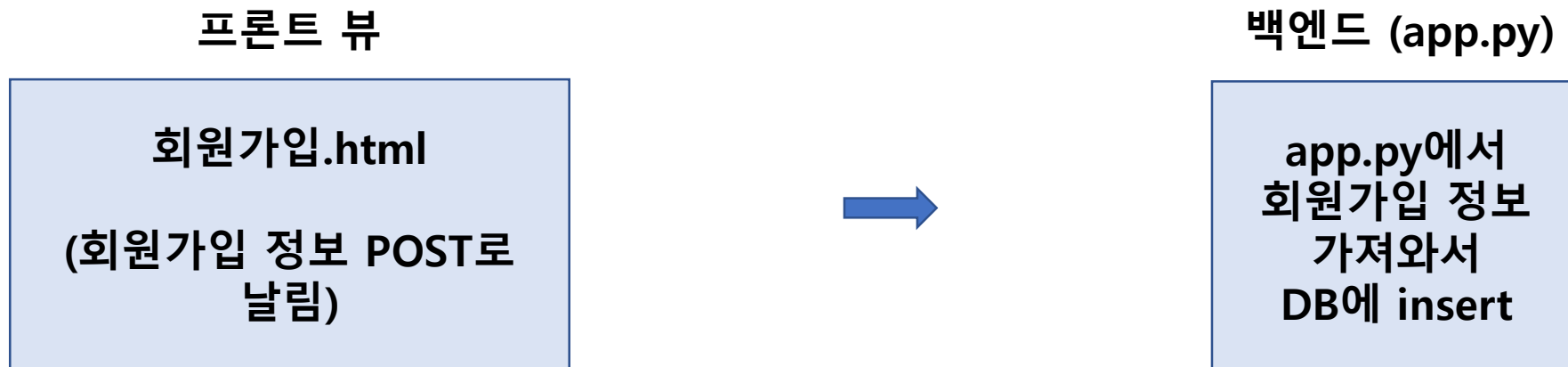
# POST, GET 값 넘기기

## 2. POST:

POST방식은 클라이언트가 서버로 데이터를 전송해 리소스를 추가, 생성하기 위해 사용되는 Method.

데이터의 양에 제한이 없어 대용량 데이터를 전송할 때는 **POST** 방식이 적합.

※ 예시 스토리: 회원가입을 한다면?



# POST, GET 값 넘기기

## 1. POST 실습 posttest.html 만들기:

```
templates > <> posttest.html > ...
 1  <!DOCTYPE html>
 2  <html>
 3  <head>
 4      <meta charset="UTF-8">
 5      <title>POST Test</title>
 6  </head>
 7
 8  <body>
 9      <form action="/test" method="post">
10          <p>이름 : <input type="text" id="input" name="input"></p>
11          <p>이름을 입력하고 제출버튼을 누르세요. <input type="submit" value="제출" onclick="alert('제출 완료!')" /></p>
12      </form>
13  </body>
14  </html>
15
```

# POST, GET 값 넘기기

## 1.1 POST 실습:

http://127.0.0.1:5000/test



```
app.py x index.html posttest.html
app.py > testpost
11 temp1 = request.args.get('adress', "seoul")
12
13 print(temp ,temp1)
14
15 return render_template('index.html')
16
17
18
19 @app.route('/test')
20 def testget():
21
22     return render_template('posttest.html')
23
```



이름 :

이름을 입력하고 제출버튼을 누르세요.



```
25
26 @app.route('/test',methods=['POST'])
27 def testpost():
28     value = request.form['input']
29
30     print(value)
31
32     return render_template('posttest.html')
33
34
35 if __name__=="__main__":
36     app.run(debug=True)
37     # host 등을 직접 지정하고 싶다면
38     # app.run(host="127.0.0.1", port="5000", debug=True)
39
```

```
오승환
27 - [07/Dec/2022 13:45:43] "POST /test HTTP/1.1" 200 -
[]
```



## Flask 데이터베이스 연동

```
0 response = requests.get(url)
1
2 # checking response.status_code (if you get
3 if response.status_code != 200:
4     print(f"Status: {response.status_code}")
5 else:
6     print(f"Status: {response.status_code}")
7
8 # using BeautifulSoup to parse the response
9 soup = BeautifulSoup(response.content, "html.parser")
10
11 # finding Post images in the soup
12 images = soup.find_all("img", attrs={"alt": "Post Image"})
13
14 # downloading images
15 for image in images:
16     url = image.get("src")
17     response = requests.get(url)
18     with open(f"image_{url}.jpg", "wb") as f:
19         f.write(response.content)
20
21 # saving image urls to database
22 db.session.add(image_urls)
```

# Flask DB 연동

## 1. Flask DB 연동 실습:

1) pip install pymysql 설치

2) mysql workbench 설치      설치법:(<https://goddaehee.tistory.com/277>)

3) app.py에 아래 코드 추가

```
app.py 1 x index.html posttest.html
app.py > ...
1 # app.py
2 from flask import Flask, render_template
3 from flask import request
4
5 import pymysql
6
7 # DB 연동
8 db_conn = pymysql.connect(
9     host = 'localhost',
10    port = 3306,
11    user = 'root',
12    passwd = '1234',
13    db = 'test',
14    charset= 'utf8'
15 )
16
17 print(db_conn)
18
19 #Flask 객체 인스턴스 생성
20 app = Flask(__name__)
21
22 @app.route('/main') # 접속하는 url
23 def index():
24     temp = request.args.get('name', "osb")
```



python app.py



```
(web_lecture) C:\Users\User\Desktop\web_lecture>python app.py
<pymysql.connections.Connection object at 0x0000026C9B7B7AC0>
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
<pymysql.connections.Connection object at 0x0000028A2DA57850>
* Debugger is active!
* Debugger PIN: 516-527-692
```



# Flask DB 연동

## 1.1 app.py에서 진짜 데이터를 불러오자:

```
app.py 1 x index.html posttest.html
app.py > ...
1 # app.py
2 from flask import Flask, render_template
3 from flask import request
4
5 import pymysql
6
7 # DB 연동
8 db_conn = pymysql.connect(
9     host = 'localhost',
10    port = 3306,
11    user = 'root',
12    passwd = '1234',
13    db = 'test',
14    charset= 'utf8'
15 )
16
17 print(db_conn)
18
19 # 커서 객체 생성
20 cursor = db_conn.cursor()
21
22 query = "select * from player"
23
24 cursor.execute(query)
25
26 for i in cursor:
27     print(i)
```



python

```
문제 1 출력 디버그 콘솔 터미널 JUPYTER: VARIABLES
('2012125', '에드밀손', 'K05', 'Edmilson', 'EDY', '2012', 'FW', 20, '', datetime.date(1978, 5, 29), '1', 1
('2012126', '다오', 'K04', '', '', '', 'DF', 61, '', datetime.date(1992, 9, 25), '2', 190, 80)
('2012127', '디디', 'K06', 'Sebastiao Pereira do Nascimento', '', '2012', 'FW', 8, '브라질', datetime.date
('2012128', '자스민', 'K08', 'Jasmin Mujidza', '', '', 'MF', 33, '크로아티아', datetime.date(1984, 3, 2),
('2012129', '알리송', 'K01', 'Alison Barros Moraes', '', '', 'FW', 14, '브라질', datetime.date(1992, 6, 30
('2012130', '메디', 'K01', 'Edmilson Alves', '', '2012', 'MF', 7, '브라질', datetime.date(1986, 2, 17), '1
('2012131', '무스타파', 'K04', '', '', '', 'MF', 77, '', datetime.date(1985, 1, 8), '1', 180, 73)
('2012132', '실바', 'K07', '', '', '', 'MF', 45, '', datetime.date(1987, 6, 20), '1', 173, 67)
('2012133', '레오', 'K03', '', '', '', 'MF', 45, '', datetime.date(1984, 10, 22), '1', 179, 74)
('2012134', '페르난도', 'K04', '', '', '', 'DF', 44, '', datetime.date(1988, 2, 24), '1', 178, 74)
('2012135', '윤원철', 'K02', '', '', '', 'MF', 45, '', datetime.date(1993, 3, 31), '1', 176, 70)
('2012136', '오비나', 'K10', '', '', '', 'MF', 26, '', datetime.date(1990, 6, 3), '1', 169, 70)
('2012137', '이고르', 'K06', '이골 실바 데 페레이따스', '이골', '2012', 'MF', 21, '브라질', datetime.date(
* Debugger is active!
* Debugger PIN: 516-527-692
```

# Flask DB 연동

## 1.2 데이터를 웹 화면에 뿌려보자: sqltest.html 생성

templates > sqltest.html > ...

```
3 <head>
4 <style>
5 table, th, td{
6     border : 1px solid black;
7 }
8 </style>
9 <meta charset="UTF-8">
10 <title>회원관리</title>
11 </head>
12 <body>
13 <table>
14 <tr>
15     <th>선수id</th>
16     <th>선수이름</th>
17 </tr>
18 {% for row in result_table%}
19 <tr>
20     <td>{{row.player_id}}</td>
21     <td>{{row.player_name}}</td>
22 </tr>
23 {% endfor %}
24 </table>
25 </body>
26 </html>
```

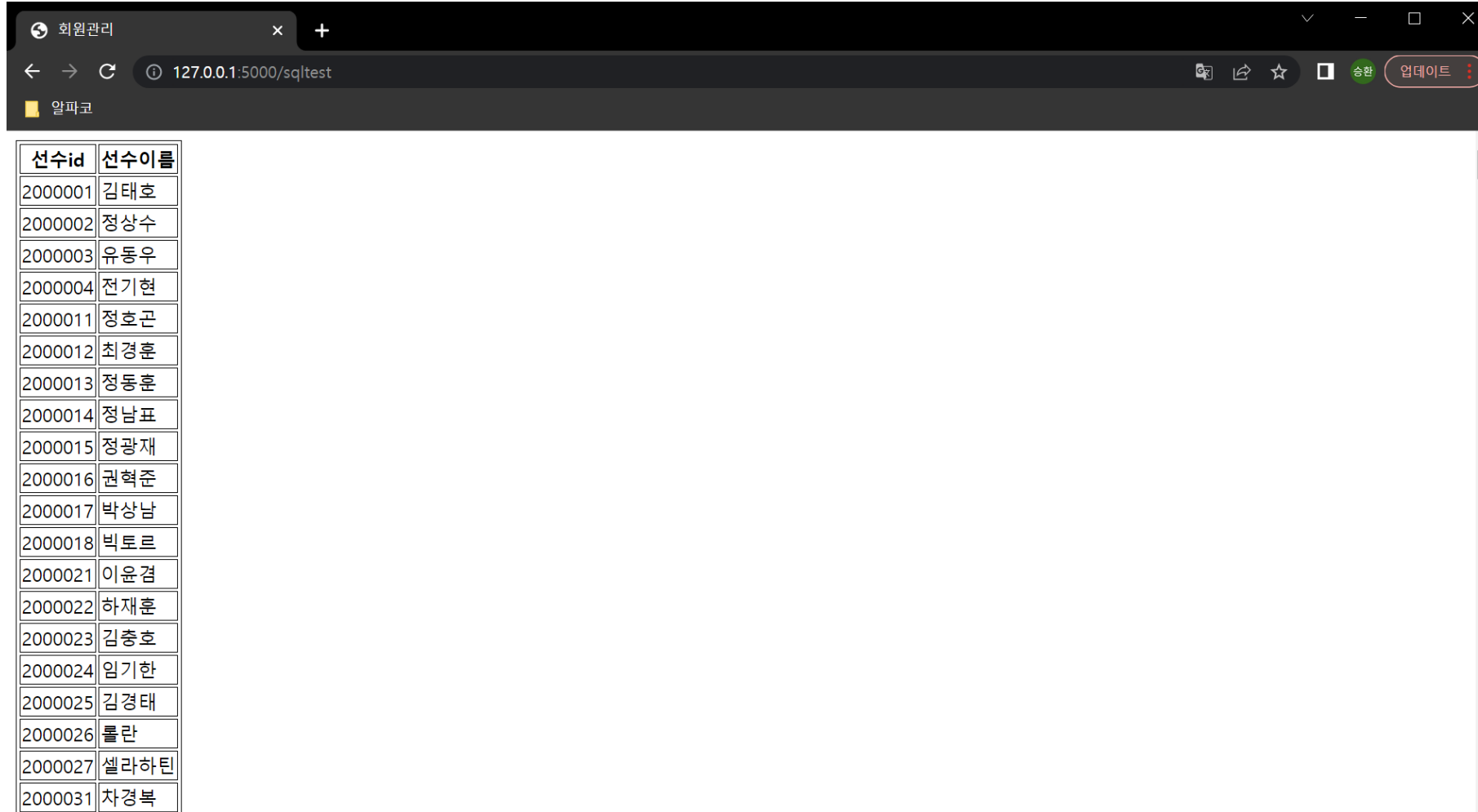
# Flask DB 연동

## 1.3 데이터를 웹 화면에 뿌려보자: app.py에 코드추가

```
app.py 1 x  sqltest.html  index.html  posttest.html
app.py > ...
41 value = request.form['input']
42
43 print(value)
44
45 return render_template('posttest.html')
46
47 @app.route('/sqltest')
48 def sqltest():
49     # 커서 객체 생성
50     cursor = db_conn.cursor()
51
52     query = "select * from player"
53
54     cursor.execute(query)
55
56     result = []
57
58     for i in cursor: # i는 ('2012136', '오비나', 'K10', '', '', 'MF', 26, '', datetime.date(1990, 6, 3), '1', 169, 70) 들어옴
59         temp = {'player_id':i[0], 'player_name':i[1]}
60         result.append(temp)
61
62     return render_template('sqltest.html', result_table = result)
63
64
65 if __name__ == "__main__":
66     app.run(debug=True)
67     # host 등을 직접 지정하고 싶다면
68     # app.run(host="127.0.0.1", port="5000", debug=True)
69
```

# Flask DB 연동

## 1.3 데이터를 웹 화면에 뿌려보자:



A screenshot of a web browser window. The address bar shows the URL `127.0.0.1:5000/sqltest`. The browser's developer tools are open, showing a console message with a green checkmark and the text "업데이트". The main content area displays a table with two columns: "선수id" (Player ID) and "선수이름" (Player Name). The table contains 27 rows of data, listing players with their IDs and names.

선수id	선수이름
2000001	김태호
2000002	정상수
2000003	유동우
2000004	전기현
2000011	정호곤
2000012	최경훈
2000013	정동훈
2000014	정남표
2000015	정광재
2000016	권혁준
2000017	박상남
2000018	빅토르
2000021	이윤겸
2000022	하재훈
2000023	김충호
2000024	임기한
2000025	김경태
2000026	롤란
2000027	셀라하틴
2000031	차경복

# Flask DB 연동

## 1.3 데이터를 웹 화면에 뿌려보자 : <응용> 버튼을 누르면 선수의 정보를 보여주는 화면을 만들자

```
templates > < sqlite.html > ...
3  <head>
4  <style>
5  table, th, td{
6      border : 1px solid black;
7  }
8  </style>
9  <meta charset="UTF-8">
10 <title>회원 관리 </title>
11 </head>
12 <body>
13 <table>
14     <tr>
15         <th>선수id</th>
16         <th>선수이름</th>
17     </tr>
18     {% for row in result_table%}
19     <tr>
20         <td>{{row.player_id}}</td>
21         <td>{{row.player_name}}</td>
22         <td><button onClick="location.href='/detail?id={{row.player_id}}&name={{row.player_name}}'" >상세정보 </button></td>
23     </tr>
24     {% endfor %}
25 </table>
26 </body>
27 </html>
28
```

선수id	선수이름	상세정보
2000001	김태환	상세정보
2000002	정상수	상세정보
2000003	유동수	상세정보
2000004	전기환	상세정보
2000011	정호준	상세정보
2000012	최경환	상세정보
2000013	정동훈	상세정보
2000014	정남준	상세정보
2000015	정광기	상세정보
2000016	권혁준	상세정보
2000017	박상현	상세정보
2000018	빅토리	상세정보
2000021	이윤준	상세정보
2000022	하재환	상세정보
2000023	김충호	상세정보
2000024	임기환	상세정보

버튼 만들어짐

# Flask DB 연동

## 1.3 데이터를 웹 화면에 뿌려보자 : <응용> 버튼을 누르면 선수의 정보를 보여주는 화면을 만들자

회원관리 x +

← → ↺ 127.0.0.1:5000/sqltest

알파코

선수id	선수이름	
2000001	김태호	상세정보
2000002	정상수	상세정보
2000003	유동우	상세정보
2000004	전기현	상세정보
2000011	정호곤	상세정보
2000012	최경훈	상세정보
2000013	정동훈	상세정보
2000014	정남표	상세정보
2000015	정광재	상세정보
2000016	권혁준	상세정보
2000017	박상남	상세정보
2000018	빅토르	상세정보
2000021	이윤겸	상세정보
2000022	하재훈	상세정보
2000023	김충호	상세정보
2000024	임기한	상세정보

```
<td>{{row.player_name}}</td>  
<td><button onClick="location.href='/detail?id={{row.player_id}}&name={{row.player_name}}'" >상세정보</button></td>  
</tr>  
</tbody>  
</table>
```

<http://127.0.0.1:5000/detail?id=2000001&name=김태호>

# Flask DB 연동

## 1.3 데이터를 웹 화면에 뿌려보자 : <응용> 버튼을 누르면 선수의 정보를 보여주는 화면을 만들자

app.py 에서 get 받는 함수 추가 및 detail.html 생성

```
@app.route('/detail')
def detailtest():
    temp = request.args.get('id')
    temp1 = request.args.get('name')

    cursor = db_conn.cursor()
    # sql 쿼리에서 작은따옴표 쿼리문에 넣으니까 넣어줘야 한다!
    query = "select * from player where player_id = {} and player_name like '{}'.format(temp,temp1)

    cursor.execute(query)

    result = []
    for i in cursor: # i는 ('2012136', '오비나', 'K10', '', '', '', 'MF', 26, '', datetime.date(1990, 6, 3), '1', 169, 70) 들어옴
        temp = {'player_id':i[0], 'player_name':i[1], 'team_name':i[2], 'height':i[-2], 'weight':i[-1] }
        result.append(temp)

    return render_template('detail.html', result_table = result)

if __name__=="__main__":
    app.run(debug=True)
    # host 등을 직접 지정하고 싶다면
    # app.run(host="127.0.0.1", port="5000", debug=True)
```

# Flask DB 연동

## 1.3 데이터를 웹 화면에 뿌려보자 : <응용> 버튼을 누르면 선수의 정보를 보여주는 화면을 만들자

### detail.html 생성

```
templates > detail.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style>
5  table, th, td, tr{
6  |   border : 1px solid red;
7  }
8  </style>
9  <meta charset="UTF-8">
10 <title>회원관리</title>
11 </head>
12 <body>
13 <table>
14 |   <tr>
15 |     <th>선수id</th>
16 |     <th>선수이름</th>
17 |     <th>팀이름</th>
18 |     <th>weight</th>
19 |     <th>height</th>
20 |   </tr>
21 |   {% for row in result_table%}
22 |     <tr>
23 |       <td>{{row.player_id}}</td>
24 |       <td>{{row.player_name}}</td>
25 |       <td>{{row.team_name}}</td>
26 |       <td>{{row.weight}}</td>
27 |       <td>{{row.height}}</td>
28 |     </tr>
29 |   {% endfor %}
30 </table>
31 </body>
32 </html>
```



선수id	선수이름	팀이름	weight	height
2000001	김태호	K10	72	180