




5주차 Python 심화 (1)



서강대 김동윤



목차

1. 주소 크롤링
2. Geocoding
3. 지도, 마커 띄우기



**이삭 토스트 매장의
분포를 html에
지도와 마커로 보기 쉽게 띄워봅시다!**

주소 크롤링

이삭토스트 전체 매장 안내 페이지 구조

isaacs.co.kr/bbs/board.php?bo_table=branches&page=1		isaacs.co.kr/bbs/board.php?bo_table=branches&page=2		isaacs.co.kr/bbs/board.php?bo_table=branches&page=3	
보훈병원점	서울 강동구 둔촌동 545 하이츠 아파트 상가 106호 (명일로 117)	서울 관악구 난곡로34길 7 (신림동)	서울 관악구 자양동 8-22 (아차산로 15마길 34)	서울 관악구 자양로 93 (자양동)	서울 관악구 자양로3길 59 (자양동)
강동구청역점	서울 강동구 천호옛길 14 (성내동)	서울 관악구 남부순환로 1949 (봉천동)	서울 관악구 보라매로 32 (봉천동)	서울 관악구 봉천동 858-9 (관악로 184)	
서울삼양점	서울 강북구 미아동 838-62 (솔샘로 213-2)				
	서울대입구역점				

http://www.isaacs.co.kr/bbs/board.php?bo_table=branches&page={원하는 페이지}

1. 만들어 둔 파이썬 파일로 가서 필요한 모듈을 import 해옵시다.

```
map.py > ...  
1 from bs4 import BeautifulSoup  
2 import urllib.request
```

1 : 크롤링을 위한 파이썬 라이브러리
2 : URL을 가져오기 위한 파이썬 모듈

라이브러리 vs 패키지 vs 모듈

모듈 : 프로그램을 구성하는 작은 부품

패키지 : 특정 기능과 관련된 여러 모듈들을 하나의 상위 폴더에 넣어 놓은 것

라이브러리 : 여러 모듈과 패키지를 묶어놓은 집합

주소 크롤링

2. 이삭토스트 매장 정보를 담아올 리스트를 만들어주는 함수를 만들어봅시다.

```
6 def make_issac_list():
7     url = 'http://www.isaacs.co.kr/bbs/board.php?bo_table=branches&page='
8     page_num = 2
9     issac_list = []
10
11     for j in range(1, page_num):
12         sourcecode = urllib.request.urlopen(url+str(j)).read()
13         soup = BeautifulSoup(sourcecode, 'html.parser')
14
15         for i in soup.find_all('td', 'td_subject'):
16             temp_text = i.get_text()
17
18             issac_list.append(temp_text)
19     return issac_list
20
```

7 : 우리의 이삭토스트 페이지 url

8 : 내가 불러오고 싶은 마지막 페이지+1 의 수를 담는 변수
(우선 1페이지만 불러올 예정이라 2로 설정했어요!)

9 : 이삭 매장 주소를 담아올 리스트

11 : j가 1부터 1씩 증가해 page_num-1 이 될때까지 돌게 해줄 for 문 (이것이 아까 8에서 page_num을 불러오고 싶은 마지막 페이지에다가 1을 더하는 이유입니다!)

12 : 우리는 페이지 (변수 j) 를 하나씩 늘려가며 데이터를 데려올 것이라서 아까 미리 지정해 둔 url 에다가 page를 나타내는 변수 j를 붙여줘서 읽어올 것입니다.

13: soup라는 아이에 j번째 페이지의 모든 정보를 html.parser을 이용해 읽어 온 결과물을 담아줄 거예요

15 : soup에 있는 아이들 중 주소를 감싸는 태그와 클래스는 td, td_subject 이므로 해당하는 아이들을 데려오게 합니다.

16 : 데려온 아이들에서 우리는 주소 텍스트만 필요하니, text 만 빼오게 합니다.

18 : 아까 만들어진 리스트에 주소들을 append 함으로써 추가시켜줍시다.

19 : 1페이지부터 page_num에서 1을 뺀 만큼 페이지를 다 돌고 나면 주소들을 모아둔 issac_list를 돌려주게 됩니다.

2. 이삭토스트 매장 정보를 담아올 리스트를 만들어주는 함수를 만들어봅시다.

1) soup 만 찍으면 ?

```
<tr>
<th scope="col">지역</th>
<th scope="col">지점명</th>
<th scope="col">주소</th>
<th scope="col">연락처</th>
<!-- <th scope="col">단체주문</th> -->
<th scope="col">상세보기</th>
</tr>
</thead>
<tbody>
<tr class="">
<td class="td_date">서울특별시</td>
<td class="td_date">개포동역점</td>
<td class="td_subject">서울 강남구 개포로 512 개포종합빌딩 109호 </td>
<td class="td_date">02-451-5421</td>
<!-- <td class="td_date"></td> -->
<td class="td_date"><a href="http://www.isaacs.co.kr/bbs/board.php?bo_
</tr>
<tr class="">
<td class="td_date">서울특별시</td>
```

2) get_text를 하기 전의 모습은 ?

```
<td class="td_subject">서울 강남구 개포로 512 개포종합빌딩 109호 </td>
<td class="td_subject">서울 강남구 대치동 950-10 (삼강로 85길 10)</td>
<td class="td_subject">서울 강남구 도곡동 959-21 1층 101호 (남부순환로 361길 7)</td>
<td class="td_subject">서울 강남구 삼성로 212 (대치동, 은마아파트)</td>
<td class="td_subject">서울 강남구 수서동 724 로즈데일빌딩 지하2층 125호 (광평로 280)</td>
<td class="td_subject">서울 강남구 일원동 682-7 (양재대로 33길 25)</td>
<td class="td_subject">서울 강동구 동남로 71길 38 (명일동, 동양타워)</td>
<td class="td_subject">서울 강동구 둔촌동 545 하이츠아파트 상가 106호 (명일로 117)</td>
<td class="td_subject">서울 강동구 천호옛길 14 (성내동)</td>
<td class="td_subject">서울 강북구 미아동 838-62 (솔샘로 213-2)</td>
```


Geocoding

Geocoding 이란?

지오코딩(geocoding)은 주소를 위도, 경도로 바꾸는 작업입니다!

도로명주소를 순서대로 입력하면
해당 주소의 정보를 json형태로 받아서
정보 내에서 **위도 & 경도 데이터**만 뽑아주는 코드예요!

우리는 사전 과제로 여러분들이 가입해주신
네이버 Map API 를 활용해서 Geocoding을 진행해볼 거예요.

Python 심화

Geocoding



서울 강남구 개포로512 개포종합빌딩 109호

Geocoding

```

{
  "status": "OK",
  "meta": {
    "totalCount": 1,
    "page": 1,
    "count": 1
  },
  "addresses": [
    {
      "roadAddress": [
        {
          "roadAddress": "서울특별시 강남구 개포로 512 개포종합상가",
          "jibunAddr": "강남구 개포동 186-ess",
          "englishAddress": "epublic of Korea",
          "512, Gaepo-ro, Gangnam-gu, Seoul, Republic of Korea",
          "addressElements": {
            "e": "서울특별시",
            "c": [
              {
                "types": [
                  "SIDO"
                ]
              }
            ]
          }
        }
      ],
      "code": "",
      "typecode": "",
      "types": [
        "SIGUGUN"
      ],
      "longName": "강남구",
      "shortName": "강남구",
      "types": [
        "RI"
      ],
      "lon": "",
      "code": "",
      "types": [
        "DONGMYUN"
      ],
      "ongName": "개포동",
      "shortName": "개",
      "개포로",
      "shortNam",
      "code": ""
    },
    {
      "types": [
        "RI"
      ],
      "longName": "",
      "X": "127.0688664",
      "Y": "37.4893190",
      "distance": 0.0
    }
  ],
  "errorMessage": ""
}

```

우리는 네이버에서 제공해주는 Geocoding API 를 활용할 거예요!
사전 과제에서 발급받은 아이디와 키를 가지고, 작업을 할 것입니다

[Classic](#) / [AI·NAVER API](#) / [Application](#)

Application ¹

등록한 Application 정보를 확인하고 관리합니다.

+ Application 등록





개발 가이드 [↗](#)

상품 더 알아보기 [↗](#)

↻ 새로 고침



삭제

<input type="checkbox"/>	App 이름	서비스구분	당일 사용량
<input type="checkbox"/>	likelion-application <div>인증 정보  변경</div>	<div>Web Dynamic Map </div> <div> Geocoding </div>	<div>0% <div></div> 0/10,000,000 회</div> <div>0% <div></div> 2,804/3,000,000 회</div>

잠깐!

외부 API와 연동하기 위해서 사용되는 접근키(access key)나
보안키(secret key)와 같은 인증 정보는
보안 측면에서 코드 상에 저장을 하면 안 됩니다.

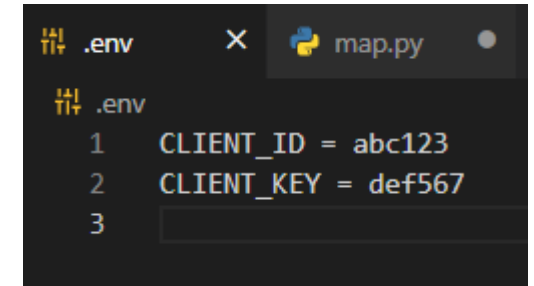
그래서 우리는 미리 설치한 python-dotenv 모듈을 활용해
환경 변수를 이용한 API Key관리를 통해서
안전하게 아이디와 키 값을 관리할 것이예요.

환경 변수를 이용한 API Key관리?

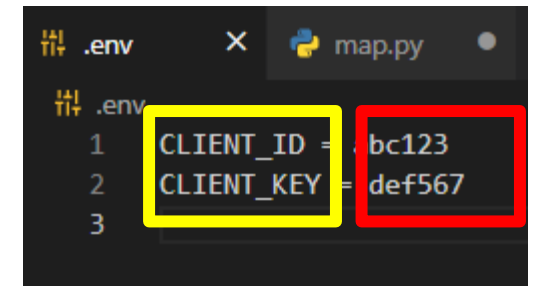
민감한 정보들을 코드가 아닌
컴퓨터 내부에 변수 형태로 저장하는 방법

.env 파일

- 애플리케이션에 필요한 모든 환경 변수에 대해 키 값을 포함하는 개별 파일
- 로컬에 저장되므로 잠재적으로 민감한 정보를 위험에 빠뜨리지 않습니다.
- python dictionary 형태로 저장합니다.



```
.env
1 CLIENT_ID = abc123
2 CLIENT_KEY = def567
3
```



```
.env
1 CLIENT_ID = bc123
2 CLIENT_KEY = def567
3
```

키(key)

값(value)

1. 환경변수를 위한 모듈을 import 해줍니다.

```
4 from dotenv import load_dotenv
5 import os
```

4 : dotenv는 .env 파일에서 키와 값 쌍을 읽고 환경 변수에 추가하는 데 사용됩니다.

5 : 운영 체제에 등록되어 있는 모든 환경 변수는 os 모듈의 environ 이라는 속성을 통해서 접근이 가능하므로 os 모듈도 import !

2. .env에 있는 환경변수 데려오기

```
7 load_dotenv()
8 client_id = os.environ.get("CLIENT_ID")
9 client_key = os.environ.get("CLIENT_KEY")
10
```

7 : .env에 우리가 저장한 환경변수를 불러와 줍니다.

8-9 : 환경 변수는 os 모듈의 environ 이라는 속성으로 접근해줍니다.
사전에서 키를 이용해 값을 가져오듯
우리가 .env에 저장했던 값들을 CLIENT_ID와 CLIENT_KEY라는 키를 통해 읽어와
각각 client_id 와 client_key 라는 변수에 저장해줍니다.

Geocoding

3. 도로명 주소를 geocoding해서 위도와 경도 정보를 포함한 json 파일을 받아오는 함수를 만들어줍니다.

```
27 def search_map(search_text):
28     encText = urllib.parse.quote(str(search_text))
29
30     url = 'https://naveropenapi.apigw.ntruss.com/map-geocode/v2/geocode?query='+encText
31     request = urllib.request.Request(url)
32     request.add_header('X-NCP-APIGW-API-KEY-ID', client_id)
33     request.add_header('X-NCP-APIGW-API-KEY', client_key)
34
35     response = urllib.request.urlopen(request)
36     rescode = response.getcode()
37     if(rescode==200):
38         response_body = response.read()
39         return response_body.decode('utf-8')
40     else:
41         print("Error Code:" + rescode)
```

28 : search_text에는 도로명 주소 하나씩이 담기게 됩니다. 이 도로명 주소 문자열에서 특수문자를 문자열로 변환해서 반환해줍니다.

```
data = '/hi/hi*hi#'
```

```
base_quote = urllib.parse.quote(data)
```

```
print(base_quote)
```

결과 : /hi/hi%2Ahi%23

30 : geocode를 부탁해~ 라고 request를 보낼 url로 geocode를 수행해줄 네이버 api를 설정해줍니다.

31 : request를 보낼 타겟을 위에서 지정한 url로 설정해줍니다.

32 -33 : request를 보내며 geocode 해주길 요청할 때 request의 헤더에 우리는 너희 api 사용할 자격이 있음을 증명할 인증값을 같이 주어야 해요! 그래서 헤더에 우리가 아까 환경변수에서 불러온 아이들을 알맞은 자리에 넣어줍니다.

35 : 네이버 geocode url 에 요청을 보냈으면 응답이 오는게 인지상정~! 응답으로 온 url을 urllib라는 url을 가져와주는 아이를 이용해 열어준 뒤 그 값을 response라는 변수에 담아줍니다.

36. 돌려주는 응답에는 응답이 잘 왔으면 잘 왔다고 나타내는 코드인 200 을, 잘 오지 않았다면 그 외의 코드를 돌려줍니다. 그 코드를 rescode로 데려옵니다.

37-39 : 응답코드가 정상적인 200이라면 잘 왔구나 하고 response_body라는 아이에 response를 돌려준 뒤, 이를 utf-8로 복호화, 즉 사람이 인식할 수 있는 형태로 복호화 해서 반환해줍니다.

40-41 : 200 코드가 아니라면 좋지 않은 일이 일어난 것이니, 해당 에러 코드를 프린트 해서 에러를 알아보려고 하기 위한 요청입니다.

Geocoding

4. Geocoding한 결과 json 에서 위도와 경도만 추출해서 위도리스트, 경도리스트에 담아줍니다.

```
import json
```

```
43 def make_location(issac_list):
44     x = []
45     y = []
46     for issac_location in issac_list:
47         temp_map = search_map(issac_location)
48         temp_map = json.loads(temp_map)
49         try:
50             temp_map = temp_map['addresses'][0]
51             x.append(float(temp_map['x']))
52             y.append(float(temp_map['y']))
53         except IndexError:
54             pass
55     return x, y
```

우선 json 모듈을 추가적으로 import 해줍니다.

43 : 함수가 가지고 조작할 아이는 우리가 처음에 만들었던 make_issac_list() 에서 만들어 준 issac_list입니다.

44-45 : x와 y는 각각 경도, 위도를 담을 리스트 입니다.

46-48 : issac_list에 담긴 도로명 주소들을 issac_location이라는 아이로 하나씩 빼옵니다. 각각의 도로명 주소를 위에서 정의했었던 search_map에 넣어주면 search_map은 geocoding을 수행해 돌려주게 됩니다. 그리고 이 값을 json으로 변환해 temp_map에 저장 합니다.

49 -52 : temp_map을 geocoding된 결과 값들 중에서 address라는 키값을 가진 배열의 0번째 인덱스 값으로 갱신해줍니다. 이렇게 geocoding 된 temp_map은 딱 위도, 경도만 가지는 게 아닌 address라는 배열의 0번째 인덱스에서 x, y 값으로 경도와 위도를 가지고 있습니다. 이를 추출해준 뒤 x 경도 값은 우리가 아까 만든 x 리스트에, 추출한 위도값 y는 y 리스트에 담아줍니다.

53-54 : 만약 x,y가 존재하지 않을 경우엔 IndexError가 발생합니다. 이 경우는 skip하고 pass문으로 이동해 다시 for문으로 돌아가 수행하도록 합니다.

55 : 최종적으로 함수가 반환하는 것은 경도값을 모아놓은 x 리스트와 위도들을 모아놓은 y리스트가 됩니다.

Python 심화

Geocoding (리마인드!)

서울 강남구 개포로512 개포종합빌딩 109호

Geocoding

```
{
  "status": "OK",
  "meta": {
    "totalCount": 1,
    "page": 1,
    "count": 1
  },
  "addresses": [
    {
      "roadAddress": "[{"roadAddress": "서울특별시 강남구 개포로 512 개포종합상가", "jibunAddr": "강남구 개포동 186-ess", "englishAddress": "epublic of Korea", "512, Gaepo-ro, Gangnam-gu, Seoul, Republic of Korea",
      "addressElements": {
        "e": "서울특별시",
        "c": [
          {
            "types": [
              "SIDO"
            ]
          },
          {
            "code": "",
            "typecode": "",
            "types": [
              "SIGUGUN"
            ],
            "longName": "강남구",
            "shortName": "강남구",
            "types": [
              "RI"
            ],
            "lon": "",
            "code": "",
            "types": [
              "DONGMYUN"
            ],
            "longName": "개포동",
            "shortName": "개",
            "code": ""
          },
          {
            "types": [
              "RI"
            ],
            "longName": "",
            "shortName": ""
          }
        ]
      },
      "x": "127.0688664",
      "y": "37.4893190",
      "distance": 0.0
    }
  ],
  "errorMessage": ""
}
```

지도, 마커 띄우기

Python 심화

지도, 마커 띄우기

1. 지도를 띄우고 (map_osm) 아까 함수들을 통해 만들어진 경도, 위도 리스트를 돌면서 마커를 생성해줍니다.

```
import folium
```

```
60 def make_marker(map_osm, x, y):  
61     for i in range(len(x)):  
62         folium.Marker([y[i], x[i]]).add_to(map_osm)
```

Folium을 추가적으로 import 해줍니다.

60 : map_osm과 make_location에서 만든 x리스트(경도), y리스트(위도)를 전달해줄 것입니다.

61-62 :

x의 리스트와 y리스트는 길이가 동일할 것이니 len(y)로 하셔도 됩니다.
X안에 들어있는 경도의 개수만큼 for문을 돌면서 y에 있는 위도와 x에 있는 경도를 map_osm에 마커로 찍히게 더해줍니다.

Marker은 위도, 경도 순으로 넣어줘야 해요!! X는 경도, Y는 위도입니다!
그래서 y[i], x[i]순으로 넣어줘야 해요!

```
map_osm = folium.Map()
```

```
map_osm = folium.Map(location=[37.4729081, 127.039306])
```

```
folium.Marker([37.4729081, 127.039306]).add_to(map_osm)
```

이 방식은 전 세계 지도를 띄워줍니다.

서울 기준으로 지도를 띄워줍니다.

마커를 추가하는 방식입니다.

folium.Marker([위도, 경도]).add_to(추가할 folium 지도)

실행 코드 작성

1. `make_issac_list` 로 이삭 토스트 매장의 도로명 주소들을 가지는 리스트인 `issac_list`를 받아온다.
2. `make_issac_list`로 받아온 도로명 주소를 하나씩 돌면서 `search_map`으로 각 도로명 주소들을 geocoding한 값을 돌려준다.
3. `make_location`을 이용해 도로명 주소를 `search map`으로 geocoding 한 정보에서 각 주소의 위도, 경도에 해당하는 `y,x` 를 `y,x` 리스트에 담아주면서 도로명 주소 위도, 경도 리스트를 만든다.
4. 담아온 위도 경도를 `map` 에 마커로 하나씩 `add` 해주면서 지도에 해당하는 위도 경도 위치에 마커를 만들어준다.

실행 코드 작성

```
65 if __name__ == "__main__" :
66
67     issac_list = make_issac_list()
68     x_list, y_list = make_location(issac_list)
69
70     map_osm = folium.Map()
71     map_osm = folium.Map(location=[37.4729081, 127.039306])
72
73     make_marker(map_osm, x_list, y_list) #순서 주의해 주세요!
74
75     map_osm.save('issac.html')
```

65 : 메인 함수의 선언, 시작을 의미합니다. 해당 코드 밑에 main 등의 함수 호출 코드를 작성해서 함수의 기능을 수행합니다.

67 : 이삭토스트 매장의 도로명 주소 리스트들을 담아와줍니다.

68. make_location에 issac_list를 담아주면, issac_list안에 있는 각 도로명 주소들은 search_map으로 geocoding 된 후, x, y 값 각각을 x 리스트, y 리스트에 더해주고 최종적으로 두 개의 리스트를 반환해줍니다.

그래서 돌려주는 두 개의 리스트 값을 x_list , y_list로 받아주었습니다.

70 : folium map을 띄워줍니다. 이렇게 디폴트 값으로 만들면 지도가 전 세계를 띄워줍니다. (단순 시험용입니다! 작성하지 않으셔도 문제 없어요.)

71 : 우리나라 위도, 경도로 folium map 을 띄워주면 우리나라 기준으로 지도가 뜹니다.

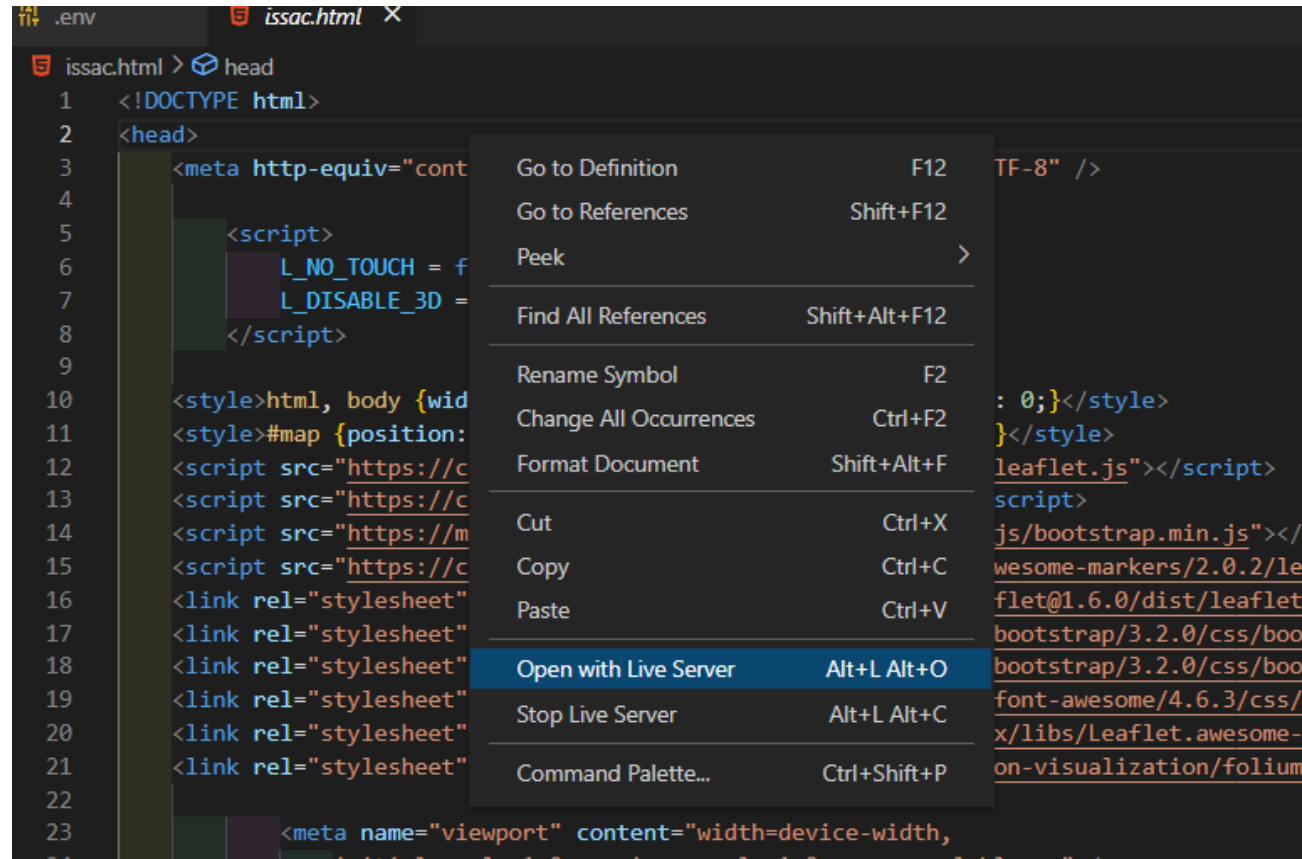
73 : 우리가 정한 folium 지도, make_location으로 반환한 경도, 위도 리스트를 make_marker로 전달해 map_osm이라는 지도에 마커가 찍히게 해줍니다.

75 : 우리가 만든 map을 issac.html이라는 파일에 저장해줍니다. 그리고 실행을 시켜주면 됩니다!

Python 심화

지도, 마커 띄우기

issac.html 로 우리가 생성한 folium map 이 생성됐을 것입니다.
이를 라이브 서버로 열어주세요!

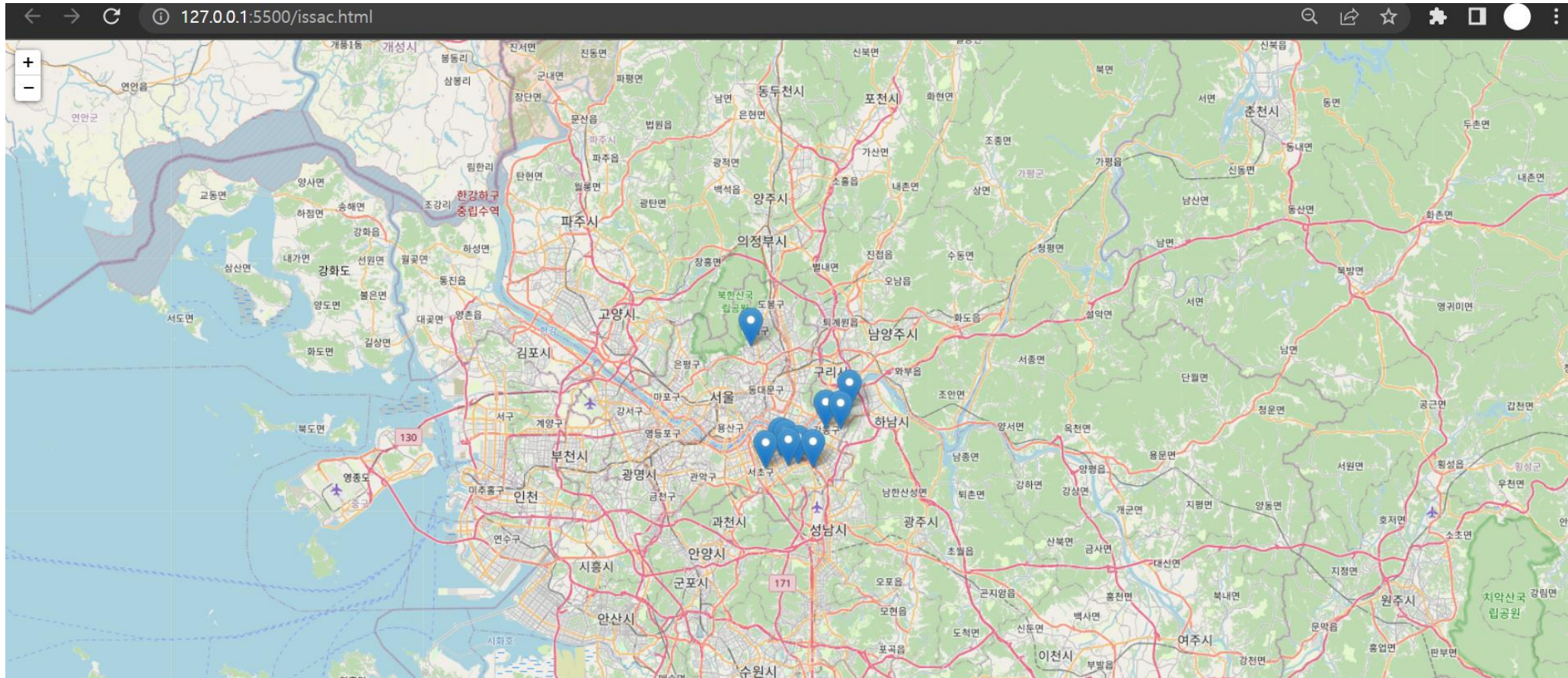


```
issac.html > head
1 <!DOCTYPE html>
2 <head>
3   <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
4
5   <script>
6     L_NO_TOUCH = false;
7     L_DISABLE_3D = false;
8   </script>
9
10  <style>html, body {width: 100%; height: 100%;}</style>
11  <style>#map {position: absolute; top: 0; left: 0; width: 100%; height: 100%;}</style>
12  <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>
13  <script src="https://unpkg.com/bootstrap@3.2.0/css/bootstrap.min.js"></script>
14  <script src="https://unpkg.com/wesome-markers/2.0.2/leaflet.awesome-markers.js"></script>
15  <script src="https://unpkg.com/leaflet@1.6.0/dist/leaflet.js"></script>
16  <link rel="stylesheet" href="https://unpkg.com/bootstrap@3.2.0/css/bootstrap.min.css">
17  <link rel="stylesheet" href="https://unpkg.com/wesome-markers/2.0.2/leaflet.awesome-markers.css">
18  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css">
19  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css">
20  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css">
21  <link rel="stylesheet" href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css">
22
23  <meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
```


Python 심화

지도, 마커 띄우기

이런 식으로 뜨게 된다면 성공입니다!





수고하셨습니다

