제목 : 국내 여행 일정 자동 생성기

정보융합학부, 2학년, 2021204045, 이성민

1. 과제 개요 :

**국내 여행 일정 자동 생성기**는 사용자가 입력한 여행지와 여행 일수, 선호 활동 정보를 바탕으로 웹에서 관련 데이터를 수집하고, 이에 기반해 추천 여행 일정을 자동으로 생성하는 프로그램이다. 사용자는 이를 통해 여행 계획에 필요한 정보를 간편하게 얻을 수 있으며, 일정 파일로 저장하여 나중에 참고할 수 있다.

2. 문제 정의와 과제의 필요성

**문제 정의**: 여행 계획을 세우는 과정에서 각 명소와 일정을 일일이 검색하고 정리하는 일은 시간이 많이 걸리며 번거로운 작업이다. 특히 처음 방문하는 여행지라면, 어떤 명소가 인기 있는지, 일정 중 어떻게 배치해야 효율적인지 알기 어렵다.

**과제의 필요성**: 이 프로그램은 사용자의 선호 활동에 맞춘 맞춤형 여행 일정을 자동으로 생성해 주므로, 일정 계획에 소요되는 시간과 노력을 줄여줄 수 있다. 또한 웹에서 자동으로 여행지 정보를 수집해 제공함으로써 사용자에게 최적의 여행 일정을 추천할 수 있다.

**장점**: 이 프로그램을 통해 사용자는 쉽게 여행 일정을 작성할 수 있으며, 파일로 저장하여 여행 중이나 계획을 재확인할 때 편리하게 사용할 수 있다. 이는 계획에 소요되는 시간을 줄여 줄 뿐만 아니라, 여행 일정을 시각화하여 동선을 파악하기에도 좋다.

3. 개발 개요

**사용자 입력**: 여행지, 여행 일수, 선호 활동을 입력받고, 이 데이터를 바탕으로 일정 생성을 진행한다. 사용자가 입력한 선호 활동을 바탕으로 웹에서 관련 명소를 검색하고, 일정에 반영할 수 있도록 설계한다.

**웹 자료 수집**: **Selenium** 혹은 **requests & BeautifulSoup** 라이브러리를 사용하여 여행지 정보를 크롤링한다. 특정 여행지의 인기 명소나 활동을 제공하는 사이트(TripAdvisor나 Google 여행 정보 등)에서 주요 데이터(여행지와 관련된 추천 명소와 활동 정보 등)를 수집해 명소 리스트를 구성한다.

**일정 생성 로직**: 수집한 명소 정보를 바탕으로 사용자 여행 일수와 하루 일정 수에 맞춰 추천 일정을 자동으로 생성한다. 수집한 명소 리스트에서 하루 일정에 포함될 활동을 가까운 루트대로 선택하고, 날짜별로 정리해 반환한다.

**파일 입출력**: 생성된 일정을 텍스트 파일로 저장하고, 사용자가 나중에 파일을 열어 일정을 확인할 수 있도록 기능을 구현한다.

**터틀 그래픽을 통한 일정 시각화**: 터틀 그래픽을 이용해 각 명소의 위치(구글 지도를 이용한 위경도 등)를 간단한 지도로 표현하고, 일정 동선을 점과 선으로 나타내어 시각적으로 여행 계획을 이해할 수 있도록 하는 시각적 가이드 기능을 추가한다.

**기타 기능**: 사용자가 일정을 커스터마이징할 수 있도록 선택한 명소를 추가하거나 삭제할 수 있는 기능을 추가하고, 일정 내 활동 통계를 제공하여 일정의 구성 비율을 쉽게 파악할 수 있게 한다.

4. 개발 내용(지금은 채우지 않습니다. 기말 과제 제출시 실제 개발 내용을 자세히 설명합니다.)

제출 과제 폴더는 해당 파일들로 이루어져 있다.

1. 관광지 데이터 DB 제작

1-1. DB Crawling - **DB1Selenium.py**

해당 파일로 제작된 **tourDB.xls** (이름, 주소, 이미지)

1-2. 상세주소, 위도/경도 추가 - **DB2Geocoding.py**

해당 파일로 제작된 **tourDB2.xls** (이름, 주소, 이미지, **상세주소, 위도, 경도**)

1-3. 상세주소, 위도/경도 추가 - **DB3Classification.py**

해당 파일로 제작된 **tourDB3.xls** (이름, 주소, 이미지, 상세주소, 위도, 경도, **종류**)

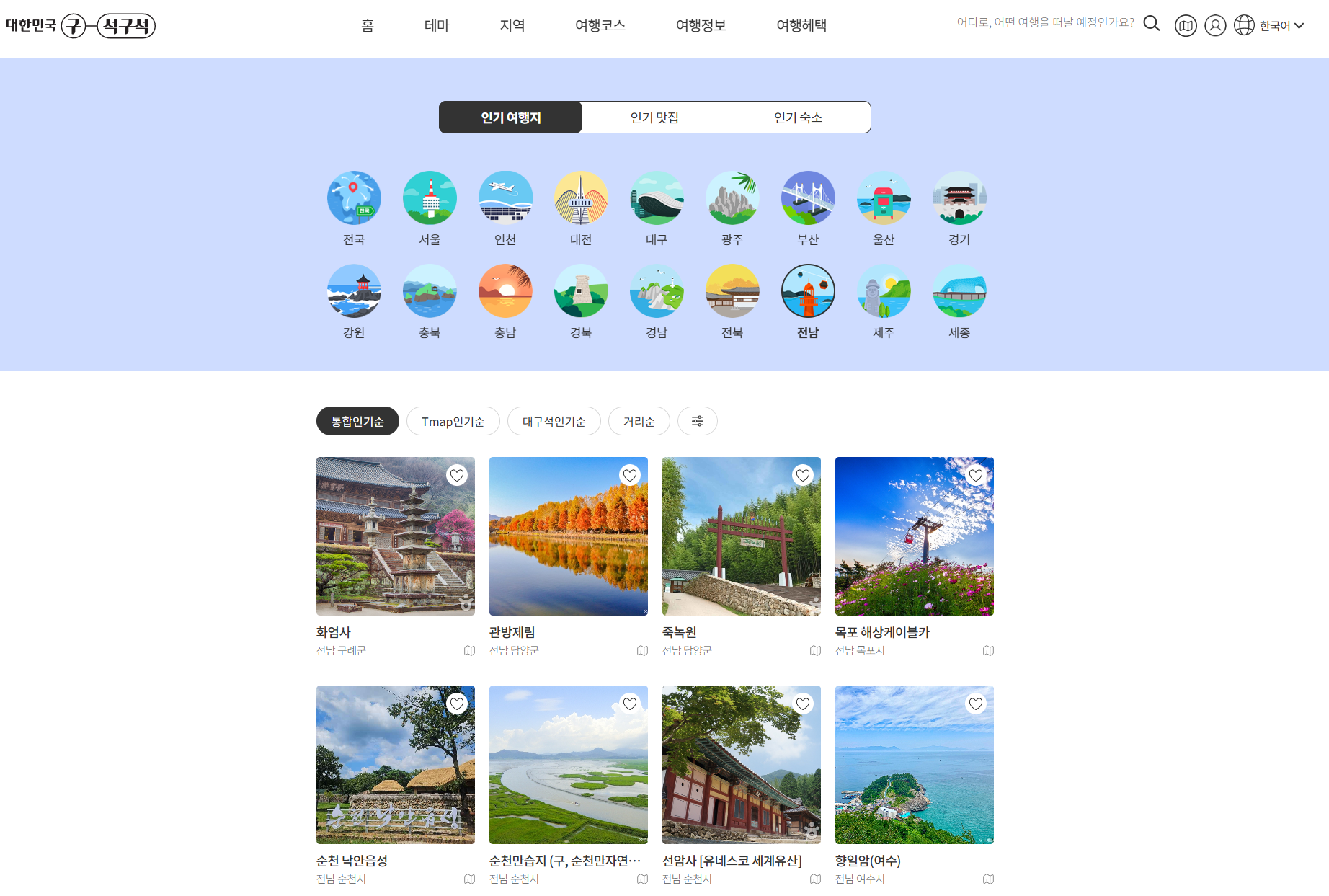
2. 실행

2-1. 실행 - **Execution.py**

2-2. 실행에 사용할 사진 모음 폴더 - **pic**

1-1. **DB1Selenium.py**

**주요 개발 내용: 반복적 데이터 수집(6312개)(Selenium), 엑셀 저장(OpenPyXL)**

본 python 파일은 한국 관광공사 - 대한민국 구석구석(https:// korean.visitkorea.or.kr/main/main.do) 사이트의 모든 관광지 정보를 크롤링하고, 이를 엑셀 파일로 저장하는 코드를 담았다. 코드 속 url은 사이트에서도 모든 여행지를 나열한 세부 페이지로 이동한다. 이 곳에서 얻을 수 있는 정보는 이미지(src), 장소 이름, 대략적인 주소로, 이는 엑셀에서 이름, 주소, 이미지라는 이름으로 저장했다. 코드는 크게 세 부분으로 나뉘어진다.

**환경 설정**: 웹 드라이버 설정 및 크롤링 대상 URL 지정

**크롤링 로직**: 지역별 관광지 정보 수집 및 데이터 정리

**엑셀 저장**: 수집한 데이터를 엑셀 파일로 저장

먼저, delay(minimum, maximum) 함수는 매크로 감지 방지를 위해 랜덤 지연시간을 생성

하는 함수이다. 최소/최대 지연시간을 입력받아 그 시간 내에서 랜덤으로 얼마나 time.sleep()를 할지 정하는 방식이다. 그 뒤에, Selenium을 사용해 Chrome WebDriver를 설정한다. **service = Service('C:/Program Files/Google/Chrome/chromedriver-win64/chromedriver.exe')** 해당 부분은 작성자의 컴퓨터에서 chromedriver가 설치된 위치를 가르키고 있으므로, 해당 파일을 실행하고자 한다면 해당 부분을 본인의 chromedriver가 설치된 위치로 조정할 필요가 있다. 또한 **options.add\_argument("--log-level=3")** 하여 레벨이 낮은 로그는 출력되지 않게 하였다.

**tour\_name = [], tour\_add = [], tour\_img = []**를 통해 각각 여행지의 이름, 여행지의 대략적 주소, 이미지 src(url)를 수집한다. try-except블록으로 로딩 및 클릭 오류 처리를 시행하였으며, webdriverwait를 사용하여 특정 Xpath가 로딩될 때까지 기다리는 로직을 사용하였다.

관광지 정보 수집은, **먼저 "더보기" 버튼을 반복적으로 클릭하여 모든 관광지를 로드한 뒤, 각 관광지의 이름, 주소, 이미지를 XPath로 탐색하여 리스트에 저장**하는 방식이다. 사이트에 “전국” 칸이 있음에도 굳이 지역별로 나눠서 조사한 이유는 전국을 기준으론 메모리 자원 한계 때문에 더보기 버튼을 반복적으로 클릭하여 모든 관광지를 로드할 수 없기 때문이다. 작성자의 컴퓨터 기준 70페이지가 넘어갔을 때 메모리 자원 부족으로 chrome 창이 오류를 일으켰다. 또한 아무리 지역 별로 나누어 조사하였다고 해도 한 지역 당 많으면 40페이지가 넘는 경우가 있을 정도로 내용이 많아, 총 실행에 **20분** 조금 넘는 시간이 걸렸다. 이는 환경에 따라 차이가 발생할 수 있다.

모든 정보를 배열에 저장한 뒤, 해당 데이터를 Pandas를 이용해 엑셀 파일 **tourDB.xls**로 저장한다.

|  |  |  |
| --- | --- | --- |
| **이름** | **주소** | **이미지** |
| 국립항공박물관 | 서울 강서구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=db57ed01-0281-4826-8bba-d9526eeabf83 |
| 하늘공원 | 서울 마포구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=1da5aec8-9e48-4012-8e32-52d5e1895a3f |
| 서대문형무소역사관 | 서울 서대문구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=6f83fb09-79a9-4697-9cf8-b95efa67fcc7 |
| 서울숲 | 서울 성동구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=1d19d1be-1096-4794-9389-63815d9e51ae |
| 길상사(서울) | 서울 성북구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=a65ec764-79b3-4203-a98a-7e4dd01e1733 |
| 올림픽공원 | 서울 송파구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=a024e46a-f30a-47a7-902d-dca0db007403 |
| 롯데월드 어드벤처 | 서울 송파구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=e7882449-3fb8-48d5-a99b-9ac6d4a0a772 |
| 남산서울타워 | 서울 용산구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=dd6daf06-28ff-48ae-b90e-472b6ecfa1db |

옆의 표는 tourDB.xls 파일의 일부이다. 이름 열에는 여행지의 이름이, 주소 열에는 대략적 주소가, 이미지 열에는 이미지를 띄울 수 있는 url이 적혀있다.

1-2. **DB2Geocoding.py**

**주요 개발 내용: Geocoding(Goggle Maps API)**

해당 코드는 DB1Selenium.py로 완성된 tourDB.xls에 저장된 장소 정보를 기반으로 Google Maps API의 **Geocoding서비스를 호출**하여 상세 주소, 위도, 경도를 가져온 뒤, 이를 다시 엑셀 파일로 저장하는 프로그램이다. **API\_KEY = "AIzaSyBEKIxwQ05ddRKKSOs-ImlpTAeZRxVxd3M"gmaps = googlemaps.Client(key=API\_KEY)** 해당 키는 작성자의 구글 계정에 발급한 API\_KEY이다. 무료버전이다보니, 너무 많이 실행하게 되면 **할당량에 막힐 수 있다는 점**을 유념하길 바란다. df['상세주소'] = ""df['위도'] = ""df['경도'] = ""를 이용해 데이터프레임에 새로운 열(상세주소, 위도, 경도)를 추가하여 Geocoding 결과를 저장할 준비를 하게 된다. 그 뒤에, Google Maps API의 geocode메서드를 호출하여 해당 장소의 Geocoding 정보를 요청한다. formatted\_address: Google Maps에서 제공하는 정식 주소이고, latitude, longitude: 해당 장소의 위도와 경도를 추출한 값이다. 이름 열과

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **이름** | **주소** | **이미지** | **상세주소** | **위도** | **경도** |
| 국립항공박물관 | 서울 강서구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=db57ed01-0281-4826-8bba-d9526eeabf83 | 대한민국 서울특별시 강서구 하늘길 177 | 37.5567157 | 126.8089333 |
| 하늘공원 | 서울 마포구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=1da5aec8-9e48-4012-8e32-52d5e1895a3f | 대한민국 서울특별시 마포구 상암동 482 하늘공원 | 37.5677554 | 126.8856819 |
| 서대문형무소역사관 | 서울 서대문구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=6f83fb09-79a9-4697-9cf8-b95efa67fcc7 | 대한민국 서울특별시 서대문구 통일로 251 | 37.574271 | 126.956071 |
| 서울숲 | 서울 성동구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=1d19d1be-1096-4794-9389-63815d9e51ae | 대한민국 서울특별시 성동구 뚝섬로 273 서울숲공원 | 37.5443878 | 127.0374424 |
| 길상사(서울) | 서울 성북구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=a65ec764-79b3-4203-a98a-7e4dd01e1733 | 대한민국 서울특별시 성북구 선잠로5길 68 | 37.5988548 | 126.9943511 |
| 올림픽공원 | 서울 송파구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=a024e46a-f30a-47a7-902d-dca0db007403 | 대한민국 서울특별시 송파구 올림픽로 424 올림픽공원 | 37.5206868 | 127.1214941 |
| 롯데월드 어드벤처 | 서울 송파구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=e7882449-3fb8-48d5-a99b-9ac6d4a0a772 | 대한민국 서울특별시 송파구 | 37.5056205 | 127.1152992 |
| 남산서울타워 | 서울 용산구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=dd6daf06-28ff-48ae-b90e-472b6ecfa1db | 대한민국 서울특별시 용산구 남산공원길 105 | 37.5512167 | 126.9881884 |

주소 열의 내용을 조합하여 검색어로 만든 뒤, 이를 검색하여 상세 주소, 위도, 경도를 가져와 결과를 저장하는 과정을 반복하여 tourDB2.xlsx로 저장한다. xls에서 xlsx가 된 이유는, pandas는 Excel 2007 이후의 파일 형식인 .xlsx로의 저장을 기본으로 사용하기 때문이다.

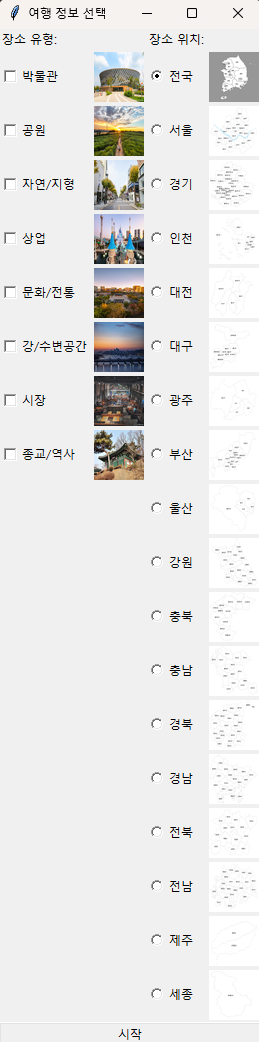
옆의 표는 tourDB2.xlxs 파일의 일부이다. 기존의 이름, 주소, 이미지 열과 새로 추가된 상세주소, 위도, 경도 열을 볼 수 있다.

1-3. **DB3Classification.py**

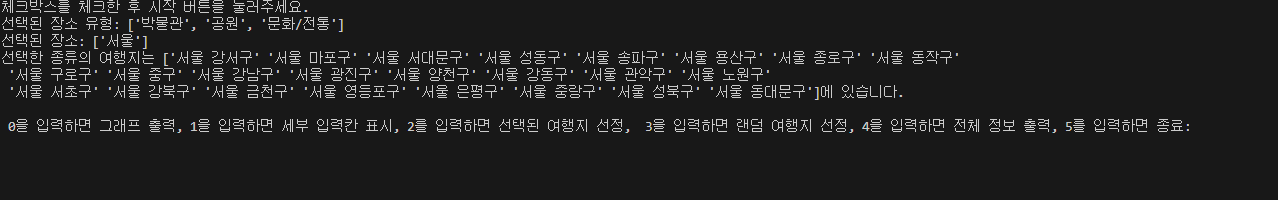
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **이름** | **주소** | **이미지** | **상세주소** | **위도** | **경도** | **종류** |
| 국립항공박물관 | 서울 강서구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=db57ed01-0281-4826-8bba-d9526eeabf83 | 대한민국 서울특별시 강서구 하늘길 177 | 37.5567157 | 126.8089333 | 박물관 |
| 하늘공원 | 서울 마포구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=1da5aec8-9e48-4012-8e32-52d5e1895a3f | 대한민국 서울특별시 마포구 상암동 482 하늘공원 | 37.5677554 | 126.8856819 | 공원 |
| 서대문형무소역사관 | 서울 서대문구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=6f83fb09-79a9-4697-9cf8-b95efa67fcc7 | 대한민국 서울특별시 서대문구 통일로 251 | 37.574271 | 126.956071 | 박물관 |
| 서울숲 | 서울 성동구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=1d19d1be-1096-4794-9389-63815d9e51ae | 대한민국 서울특별시 성동구 뚝섬로 273 서울숲공원 | 37.5443878 | 127.0374424 | 공원 |
| 길상사(서울) | 서울 성북구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=a65ec764-79b3-4203-a98a-7e4dd01e1733 | 대한민국 서울특별시 성북구 선잠로5길 68 | 37.5988548 | 126.9943511 | 자연/지형 |
| 올림픽공원 | 서울 송파구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=a024e46a-f30a-47a7-902d-dca0db007403 | 대한민국 서울특별시 송파구 올림픽로 424 올림픽공원 | 37.5206868 | 127.1214941 | 공원 |
| 롯데월드 어드벤처 | 서울 송파구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=e7882449-3fb8-48d5-a99b-9ac6d4a0a772 | 대한민국 서울특별시 송파구 | 37.5056205 | 127.1152992 | 상업 |
| 남산서울타워 | 서울 용산구 | https://cdn.visitkorea.or.kr/img/call?cmd=VIEW&id=dd6daf06-28ff-48ae-b90e-472b6ecfa1db | 대한민국 서울특별시 용산구 남산공원길 105 | 37.5512167 | 126.9881884 | 공원 |

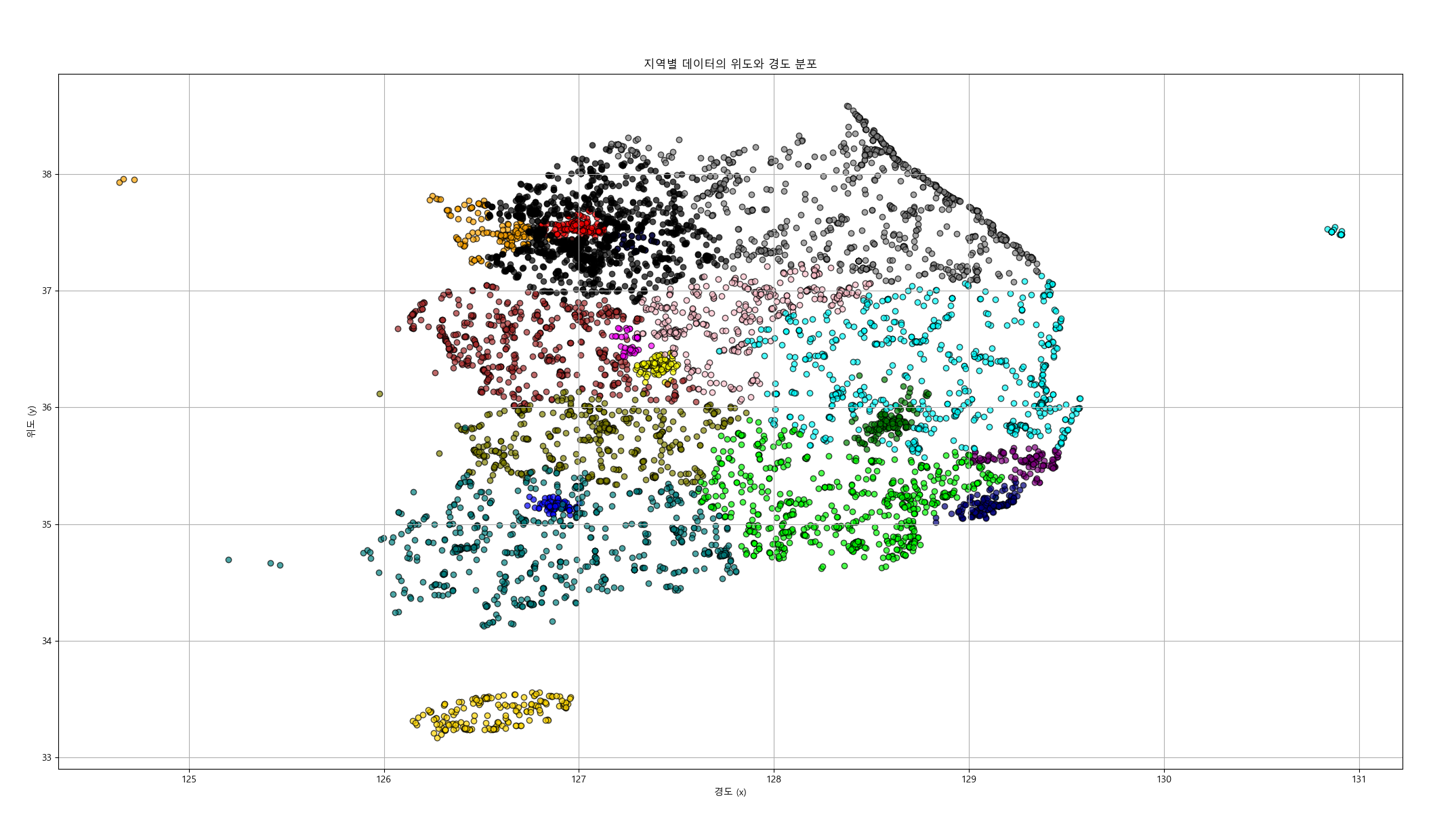
해당 py 파일은 설명할 내용이 많지 않다. 여행지의 분류를 위해 ‘종류’ 열을 새로 만든 뒤 이름에 있는 특정 단어를 이용해 분류한다. 특정 상호 명까지 정리를 끝마쳐, 기타로 분류된 행은 존재하지 않는다. 식물관, 생태관 등 박물관과 유사한 것들은 모두 박물관으로 분류하였고, 섬, 크루즈, 강변 등 물가와 관련된, 혹은 근처인 장소는 모두 강/수변공간으로 분류하였다. 종교 혹은 종교 관련 역사에 속하는 것들은 종교/역사에 분류하였고, 그 밖에 역사적인 것과 문화를 문화/전통으로 분류하였다.

**2-1. Execution.py**

① 시작 화면.

처음 시작하면 나오는 선택 화면이다. grid를 이용해 정렬하였으며, 장소 유형은 체크 박스(중복 선택 가능), 장소 위치는 라디오 버튼(중복 선택 불가능)으로 구현하였다. 장소 유형에서 아무것도 체크하지 않았을 때는 모든 요소를 체크한 것과 동일한 취급을 한다. 체크 여부에 따른 출력 결과의 차이는 결과 탭에서 다루도록 하고 본 단계에서는 개발 내용에 대해서 중점적으로 다뤄보겠다.



 체크된 정보를 토대로 새로운 df를 제작한 후로, 사용자에게 입력을 요구한다. 0번을 누를 시 새로운 df의 경도를 x, 위도를 y로 하여 장소의 위치를 표시한다. 아래의 위경도 분포 그래프는 임시로 위치에 따라 색을 다르게 하여 위경도를 표시한 그래프이다. 실제 프로그램에서 전국과 모든 장소 유형을 체크하고(혹은 아무 장소 유형도 체크하지 않고) 0번을 누르면 해당 그래프에서 색만 통일된 그래프를 볼 수 있다.

1번은 ‘세부 입력칸 표시’이다. 새로운 df의 ‘주소’ 열의 unique를 출력하는데, 만약 여기서 특정하게 고르고 싶은 주소가 있다면 1번을 누른 후 해당 주소를 입력하게 된다. 해당 주소의 문자열을 입력할 때 문자열의 일부만 입력해도 그 일부 문자열에 해당하는 내용이 입력된다(노원 -> 서울 노원구). 만약 문자열의 일부에 해당하는 내용이 2개 이상이라면 다시 입력받는다(강 -> 서울 강동구, 서울 강서구, 서울 강북구, 서울 강남구...) 또한. 중복해서 넣을 수 없다([서울 노원구, 서울 노원구]). 0을 입력하면 다시 체크 박스를 제출한 직후의 단계로 돌아오게 된다. 그러나 참고할 DB의 내용은 달라진 상태이다.

4번을 누르게 되면 참고하고 있는 DB의 전체 내용을 출력한다. 사실 전체 내용은 아니고 head(10)개 만 출력하고 나머지는 등 {db.shape[0] - 10}개 있다고 출력된다. 몇백~몇천개를 모두 출력할 수 없어 이렇게 구현하였다.

5번을 누르면 quit()을 이용해 프로그램을 종료한다. 2~3번은 결과 탭에서 결과 사진과 함께 자세히 설명하겠다.

5. 중간 계획과 최종 과제 사이의 차이점 구체적 기술(변경/수정 내용이 있다면 그 이유도 함께 기술)

**5-1. 사용자가 직접 여행 계획 수정**

**중간 계획**: 사용자가 자동으로 선정된 여행 코스에서 특정 부분을 빼고 넣을 수 있도록 구현한다.

**최종 과제**: 여행 코스의 순서를 조정하는 기능이 구현되지 않았다. 순서 구현 문제로 인해 사용자가 직접 여행 코스를 수정하는 기능을 추가하지 못하였다.

**수정 이유**: 순서 조정이 복잡하게 구현되었고, 이를 구현하기 위한 충분한 시간과 리소스를 확보할 수 없었기 때문에 최종적으로 수정 기능을 제외하였다.

**5-2. 텍스트 파일 저장/불러오기**

**중간 계획**: 여행 계획을 텍스트 파일로 저장하고 불러오는 기능을 추가한다.

**최종 과제**: 텍스트 파일 저장/불러오기 기능을 구현하지 않았다. 대신, 사용자가 반복적으로 여행 코스를 작성할 수 있도록 while True:를 이용한 무한 반복을 구현하였다.

**수정 이유**: 여행 계획을 반복적으로 수정하고 싶을 때, while True:를 사용하여 코드 실행 후 다시 처음부터 선택을 할 수 있게 되어, 텍스트 파일의 필요성을 느끼지 못하였다.

**5-3. TripAdvisor나 Google 여행 정보 사용**

**중간 계획**: TripAdvisor와 Google 여행 정보를 크롤링하여 여행 코스를 선정하는 방식으로 구현한다.

**최종 과제**: TripAdvisor 대신, '대한민국 구석구석' 사이트를 사용하여 크롤링을 실행하였다.

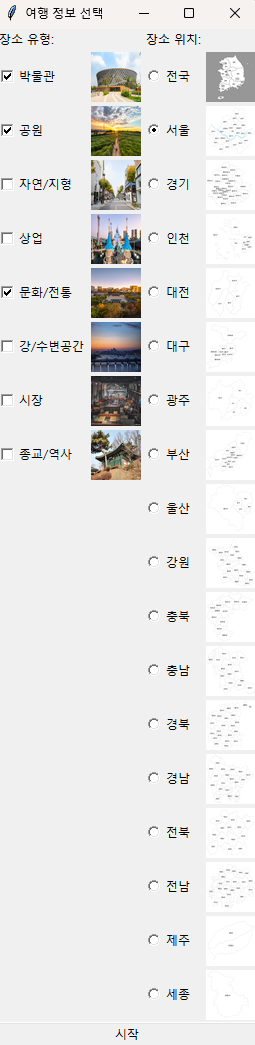
**수정 이유**: TripAdvisor는 자동 크롤링을 차단하고 있어 크롤링 기능을 사용할 수 없었다. 이로 인해 대신 '대한민국 구석구석' 사이트로 대체하여 크롤링을 실행하게 되었다. 그러나 '대한민국 구석구석' 사이트는 ‘TripAdvisor’와 다르게 장소의 종류가 미리 정의되어 있지 않아서, 장소 분류 작업이 추가로 필요해졌다(DB3Classification.py).

**5-4. 여행 일수 입력받기**

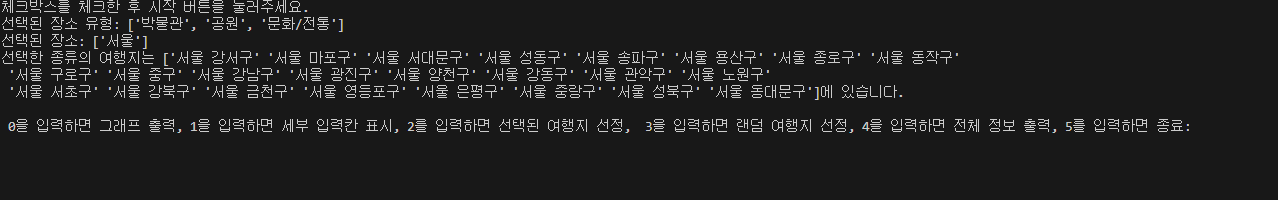
**중간 계획**: 여행 일수를 입력받아 그만큼의 활동을 진행하도록 설정한다.

**최종 과제**: 여행 일수를 입력받는 기능을 완전히 구현하지 않았다. 대신, while True:루프를 통해 여행 계획을 반복적으로 받을 수 있게 되어, 여러 날의 여행 계획을 한 번에 받을 수 있게끔 처리하였다.

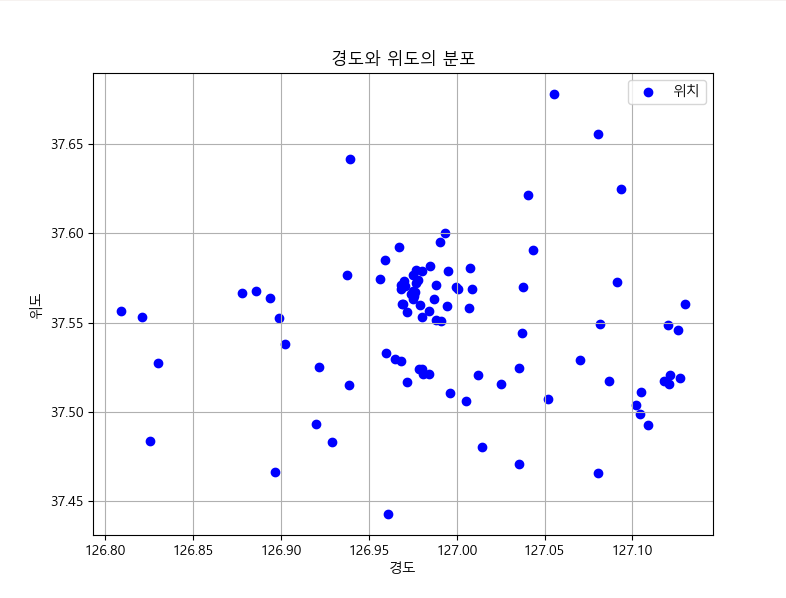
**수정 이유**: 여행 일수에 따른 활동 수를 동적으로 설정할 필요성을 느끼지 못했고, while True:구조로 여러 날의 계획을 반복적으로 받을 수 있기 때문에 일수에 따른 활동 수를 별도로 관리할 필요가 없다고 판단하였다.

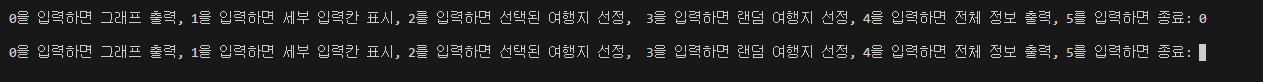
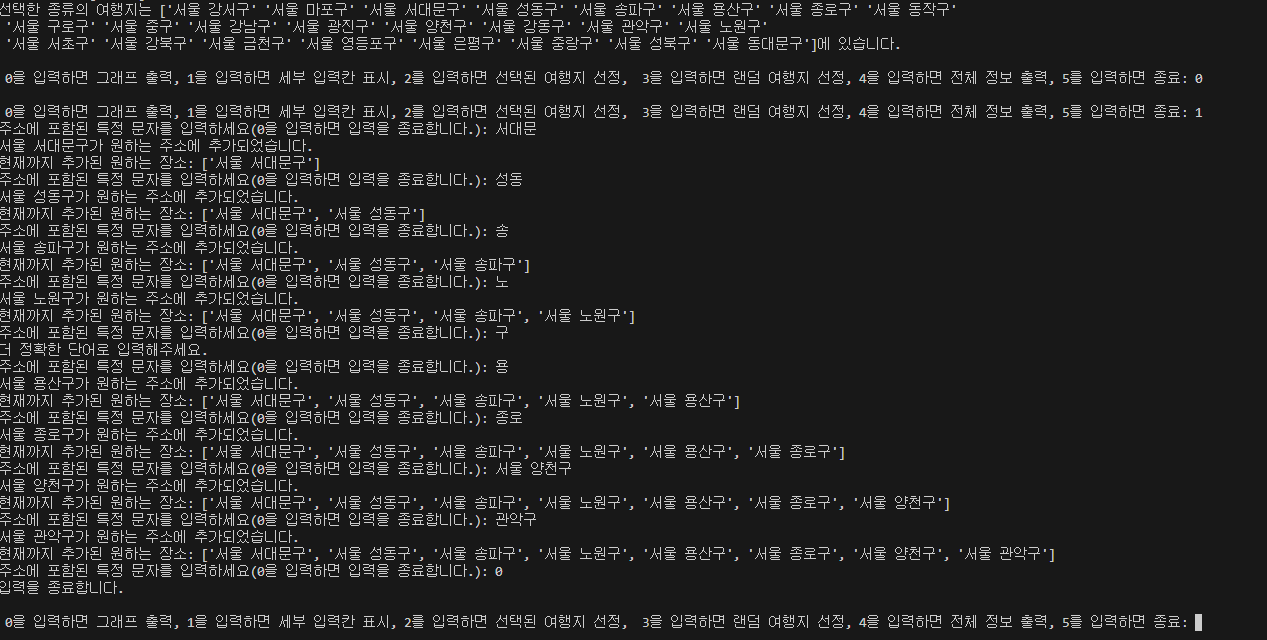
6.결과

시작 화면에선 [박물관, 공원, 문화/전통]을 고르고 장소 위치는 서울로 골랐다.



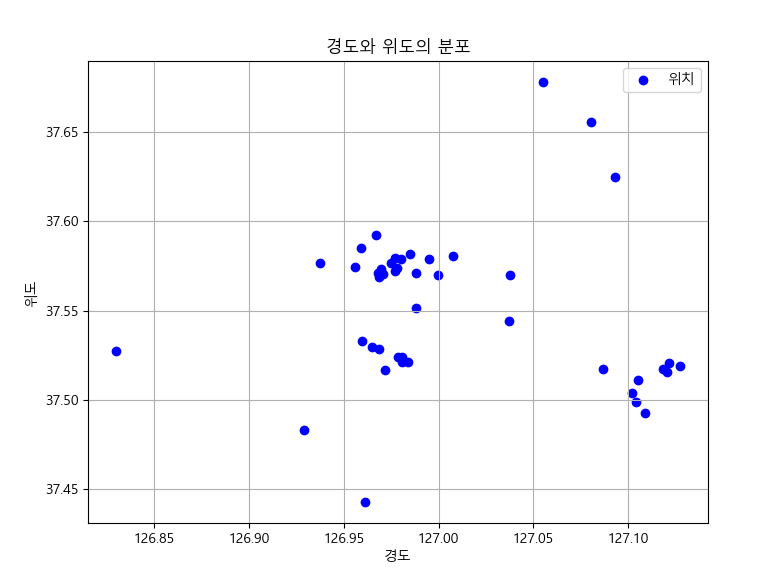
그렇게 하면 종류가 박물관, 공원, 문화/전통 중 하나이면서 주소에 ‘서울’이 포함된 행만 남는 새로운 DB가 만들어진다. 그 DB의 ‘주소’ 열의 unique를 출력하여 1번을 눌렀을 때 무엇 중에서 입력해야 할지 볼 수 있다.

여기서 0번을 누르면 서울의 박물관, 공원, 문화/전통의 위치를 볼 수 있다.



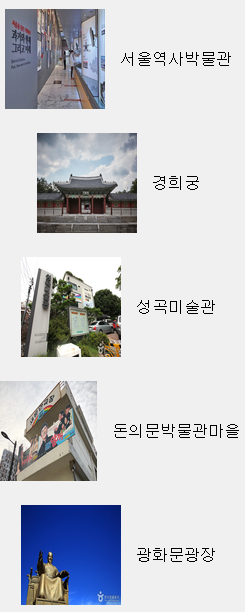
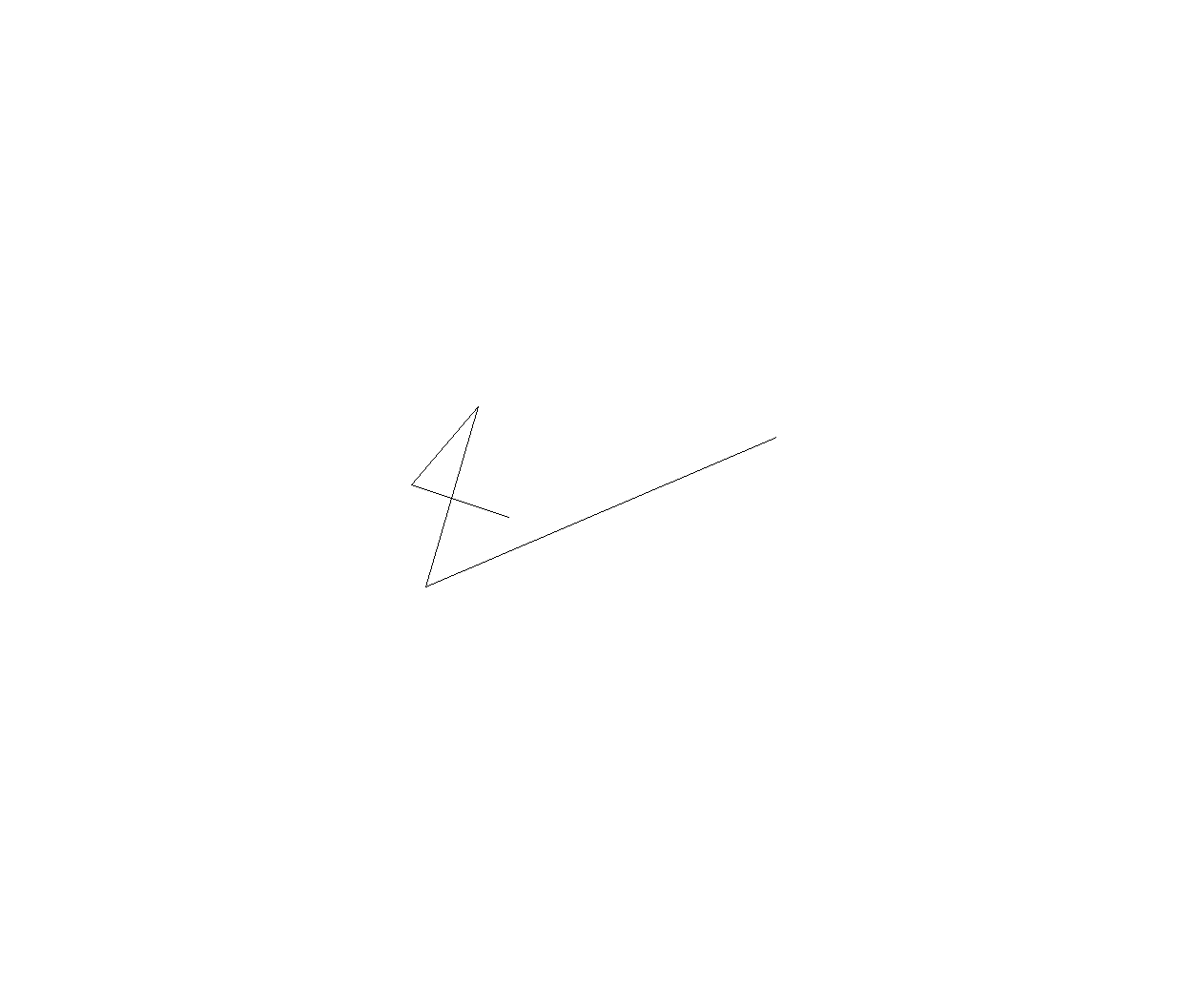
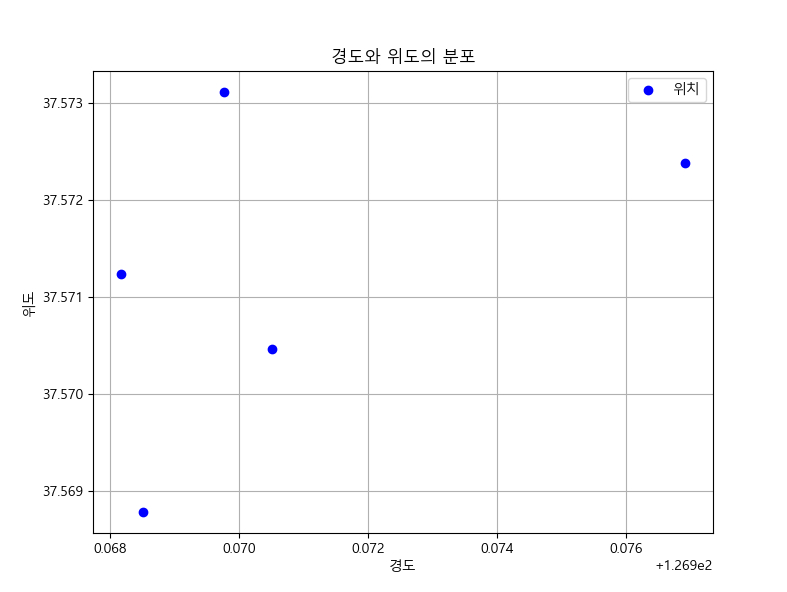
그래프를 종료하면 DB의 정보는 저장된 채로 다시 숫자 입력 단계로 돌아오게 된다.

1번을 누르면 세부 주소 입력 단계로 들어오게 된다. 사진처럼 이름의 일부만 입력해도 겹치는 글자가 있지 않는 선에서 일부를 인식해 배열에 들어오게 된다. ‘구’를 입력하면 글자가 겹쳐 ‘더 자세히 입력해주세요’가 나오고, 겹치지 않으면 한글자만 입력해도 정상적으로 배열에 들어오는 것을 볼 수 있다.

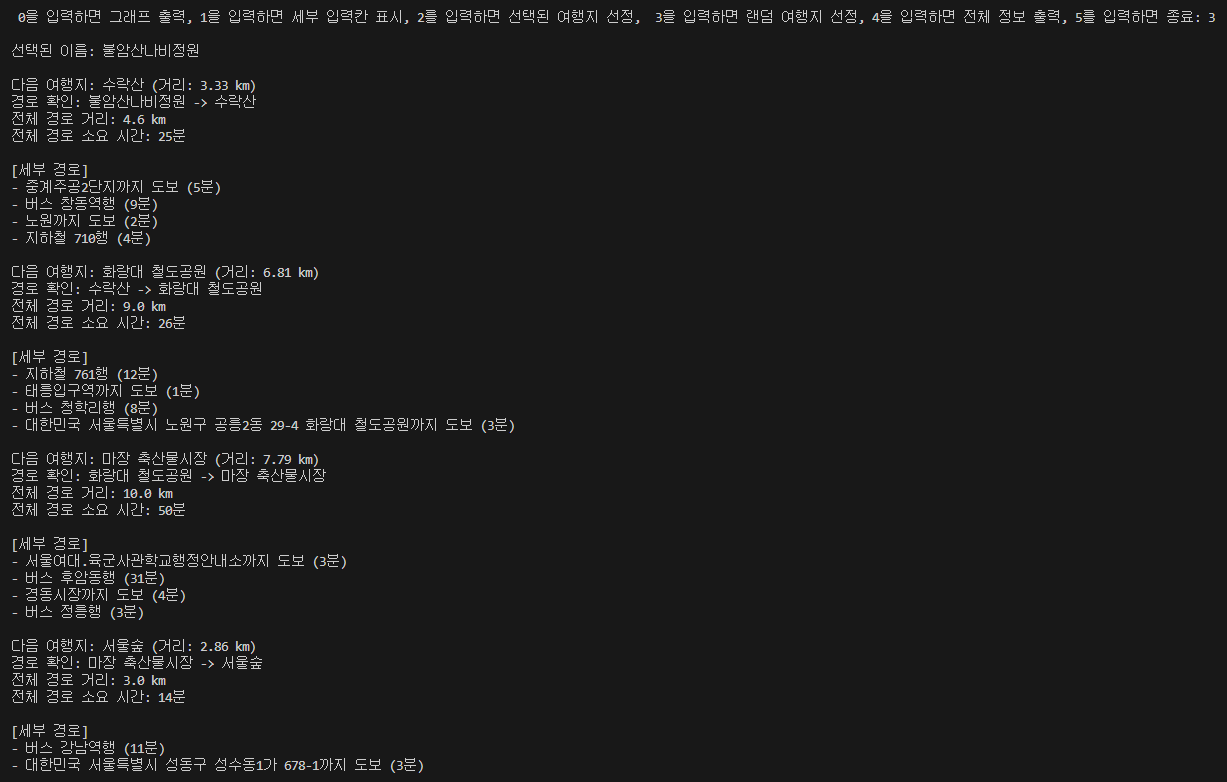
 0을 눌러 입력을 종료한 뒤 다시 0을 눌러 그래프를 출력하였다. 위의 그래프와 비교했을 때 성동구, 송파구, 노원구, 용산구, 종로구, 양천구, 관악구 위치의 장소만 남은 것을 볼 수 있다.



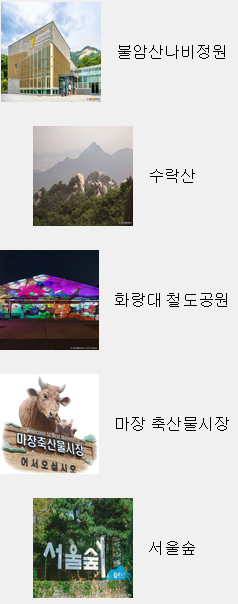
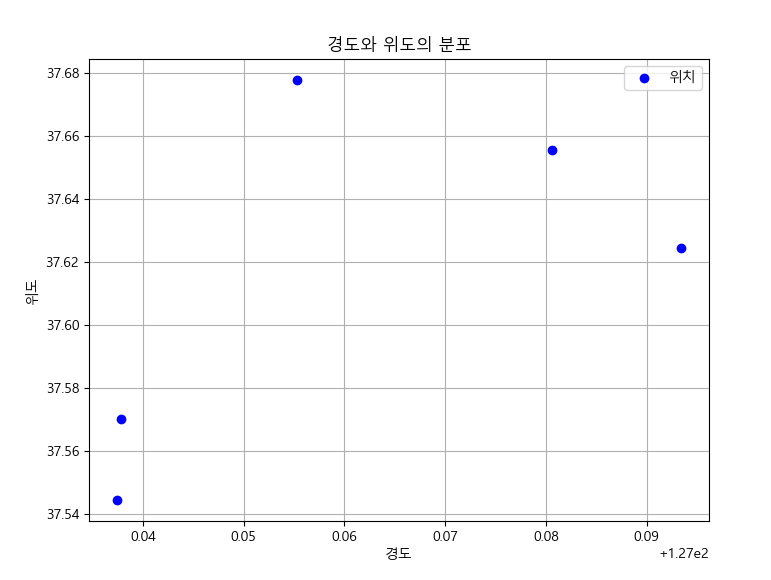
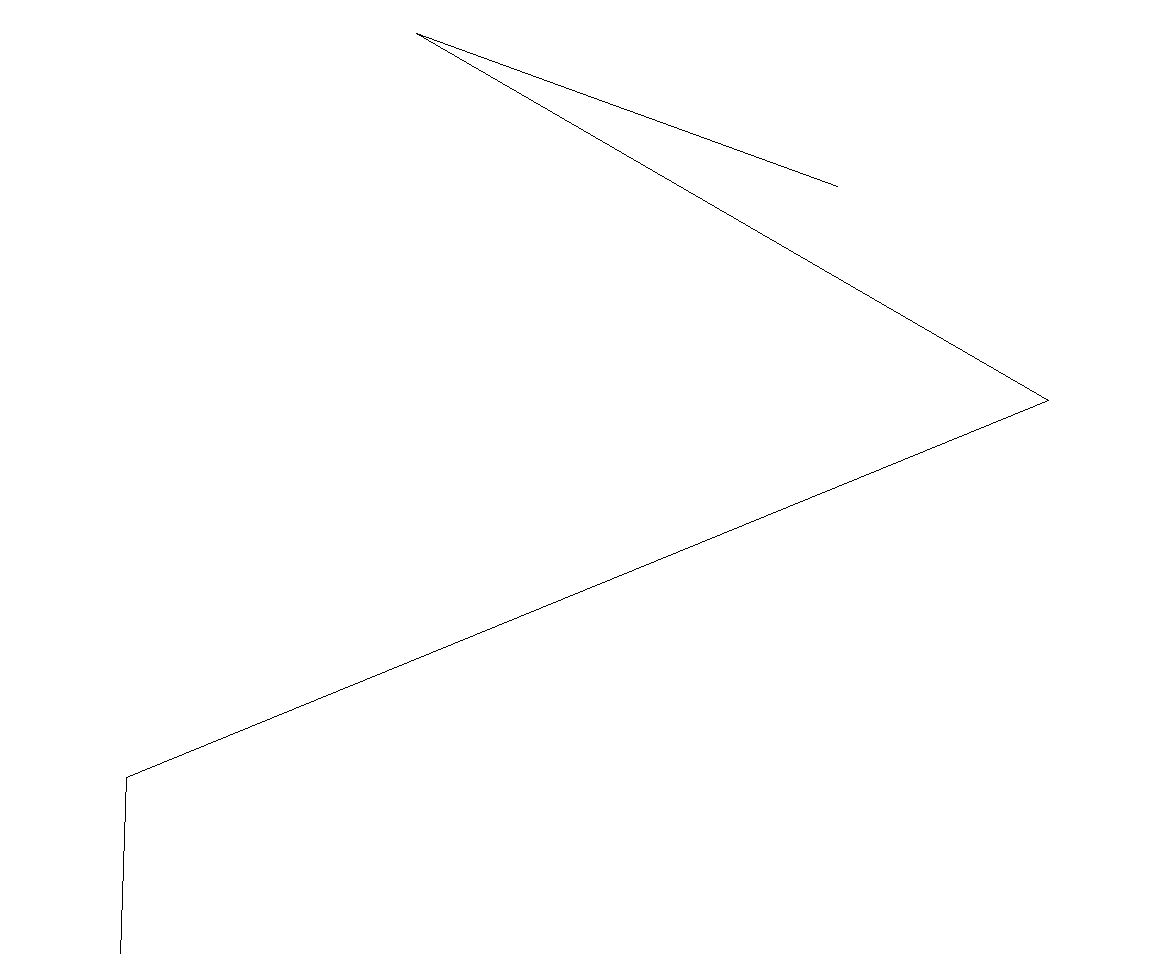
2를 입력했을 때의 화면이다. 2는 원하는 여행지를 선정한 뒤, 이와 가장 가까운 장소를 출력, 거리를 출력한 뒤 Geocoding을 이용해 구글맵과 연동하여 대중교통 기준 가는 길을 표시한다. 이를 4번 반복한다. 대중교통이 운행하지 않는 **새벽** 등의 시간이거나 거리가 너무 가까울 경우 길찾기 정보를 찾지 못할 수 있다.



출력된 뒤 선정된 장소의 사진을 나열하는 표가 나오게 된다. 이를 통해 가고자 하는 장소들을 한 눈에 볼 수 있다. 그 뒤에 그래프를 보고자 한다면 0번을 눌러 그래프를 볼 수 있다. 먼저 장소들의 위/경도 그래프가 나오고, 탐방 순서대로 해당 위경도를 탐색하는 터틀 그래픽을 확인할 수 있다. 좌표 차이가 매우 작아 기존 단위는 움직이지 않아 쓸데없는 숫자를 뺀 다음 보정치(10000)을 곱해 움직임을 나타낼 수 있게 하였다.



3번은 2번과 유사하지만 처음 선택을 랜덤으로 정해준다. 이 또한 사전에 정한 조건을 따르기 때문에, 도저히 여행을 어디로 갈지 모를 땐 해당 기능을 이용해 여행지를 선정할 수 있다.



해당 기능 또한 2번과 똑같이 사진 나열 표, 위경도 그래프, 터틀 그래픽까지 정상적으로 출력되는 것을 확인할 수 있다.

7. 결론

해당 프로그램은 이 프로그램은 사용자가 여행지, 선호 활동 등을 입력하면 이를 바탕으로 관련 데이터를 수집하고, 추천 일정을 자동으로 생성하여 제공한다. 이를 통해 여행 계획에 소요되는 시간과 노력을 줄이고, 여행 일정을 효율적으로 작성할 수 있도록 도와줄 수 있다. 이 과정에서 웹 크롤링 기술을 활용하여 여행지와 관련된 정보를 수집하고, 이를 효율적으로 처리하여 사용자가 원하는 일정을 제공할 수 있었다.

개발한 시스템은 여행 계획을 세우는 데 필요한 시간을 크게 단축시킬 수 있으며, 여행지에 대한 정보가 시각적으로 제공되어, 사용자에게 더 직관적인 여행 계획을 제공할 수 있었다.

이 시스템은 특히 여행 준비에 어려움을 겪는 사람들에게 매우 유용할 것이다. 매번 새로운 여행을 떠날 때마다 계획을 세우는 번거로움이 줄어들며, 맞춤형 일정으로 여행을 즐길 수 있는 기회를 제공할 수 있다. 향후 추가적으로 더 다양한 여행지와 활동을 반영할 수 있도록 확장하거나, 사용자 맞춤형 기능을 더욱 강화하는 방향으로 발전시킬 수 있겠다.

8. 참고자료

<https://api.ncloud-docs.com/docs/ai-naver-mapsdirections-driving> (네이버 Directions 5 API Guide)

[https://developers.google.com/maps/documentation/javascript/geocoding?hl=ko&\_gl=1\*nbmcgl\*\_up\*MQ..\*\_ga\*OTU5NjQyODYuMTczNDI2OTk4Nw..\*\_ga\_NRWSTWS78N\*MTczNDI2OTk4Ny4xLjAuMTczNDI2OTk4Ny4wLjAuMA..](https://developers.google.com/maps/documentation/javascript/geocoding?hl=ko&_gl=1*nbmcgl*_up*MQ..*_ga*OTU5NjQyODYuMTczNDI2OTk4Nw..*_ga_NRWSTWS78N*MTczNDI2OTk4Ny4xLjAuMTczNDI2OTk4Ny4wLjAuMA..) (Google Geocoding Service Guide)

<https://sphinfo.com/blog/read/562> (구글 Routes API 가이드 블로그)

<https://korean.visitkorea.or.kr/main/main.do> (대한민국 구석구석)