Homework 2

SNU L0444.200, 2017 Fall

Due: 11/1 (Wed), 23:59

TA: Jinyung Kim, Prof: Kwangkeun Yi

이번 숙제에서는,

- 첫 번째 숙제에서 다룬, 위치 데이터와 문자열 데이터를 함께 다루는 프로그램을 만들어 본다. 필요한 경우 첫 번째 과제에서 작성한 함수를 조합하거나 수정해가며 좀 더 큰 프로그램을 만드는 경험을 할 수 있을 것이다.
- json과 파일 입출력을 통해 큰 규모의 데이터를 처리해보며, 조금 더 실제 세계에서 사용할 만한 프로그램에 가까워지는 맛을 본다.

Exercise 1 "이름과 키워드가 기록된 위치정보"

앞으로 다음과 같은 네 개의 키를 가지는 파이썬의 딕셔너리(Dictionary) 자료형을 이용하여, 이름과 키워드가 기록된 위치정보를 표현한다.

- `latitude`: 위도를 나타내는 실수값 (`float`)
- `longitude`: 경도를 나타내는 실수값 (`float`)
- `name`: 이 장소의 이름을 나타내는 문자열 (`string`)
- `keywords`: 이 장소를 표현하는 키워드들을 나타내는 문자열들의 리스트 (`list[string]`)

위도를 나타내는 실수, 경도를 나타내는 실수, 이름을 나타내는 문자열, 키워드들을 나타내는 문자열들의 리스트가 순서대로 인자로 주어지면, 위와 같은 딕셔너리를 돌려주는 함수 `make_named_loc()`를 작성해 본다. (`function: float, float, string, list[string] -> dict`)

예를 들어, `make_named_loc(90.0, 0.0, "북극", ["빙하", "북극곰"])`는 다음과 같은 딕셔너리를 만들어 돌려준다: {'latitude': 90.0, 'longitude': 0.0, 'name': "북극", 'keywords': ["빙하", "북극곰"]}

이 이후의 문제들에 대해서는, 여러분의 코드가 실행되는 위치에, 이름과 키워드가 기록된 위치정보(1번 문제에서 주어진 것과 같은 자료)들의 리스트가 들어있는 파일 `named_locs.json`이 함께 주어질 것이다. 이를 "위치정보들의 목록"으로 부르자. 해당 파일로부터 이름과 키워드가 기록된 위치정보들의 목록을 읽어들인 뒤, 이를 바탕으로 주어진 내용을 구현하면 된다.

Exercise 2 "빈번한 키워드"

위치정보들의 목록 전체를 아우르는 핵심 키워드들을 선정하고자 한다. 근거 중에 하나로 키워드의 빈도수를 생각해 볼 수 있다. 중요한 단어라면 여러 번 반복되었을 것이기 때문이다. 모든 키워드들의 리스트 전체에서 등장한 키워드들을 빈번한 순으로 정렬해 이를 돌려주는 함수 `frequent_keywords()`를 작성해 본다. 빈도수가 같다면 가나다순으로 나열되도록 한다. (`function: none -> list[string]`)

예를 들어, '피자'가 키워드에 포함된 곳이 세 곳, '파스타'가 두 곳, '와인'이 네 곳, '커피'가 두 곳이었다면, `['와인', '피자', '커피', '파스타']`를 돌려준다.

Exercise 3 "가까운 곳을 찾아서"

현재 위치를 나타내는 위도와 경도와, 키워드 하나가 순서대로 인자로 주어지면, 위치정보들의 목록에서 해당 키워드를 포함하는 위치 중 가장 가까운 곳의 이름을 돌려주는 함수 `nearest()`를 작성해 본다. (`function: float, float, string -> string`)

이 함수를 이용해 다음과 같은 것을 구할 수 있을 것이다: 지하철을 타야겠다면 현재 위치에서 가장 가까운 지하철역, 급히 요기를 해야 할 때 현재 위치에서 가장 가까운 분식집 등.

위도와 경도로 표현되는 위치에 대해, 두 지점 사이의 거리는 복잡한 계산을 필요로 한다. 이를 계산해주는 함수 `dist()`가 숙제1에서 제공된 바 있다. 같은 라이브러리를 사용하여 계산하면 된다.

Exercise 4 "적당히 가까운 곳을 들러서"

현재 위치를 나타내는 위도와 경도와, 목적지의 위치를 나타내는 위도와 경도와, 키워드들의 리스트와, 가능한 최대 거리가 순서대로 인자로 주어진다. 현재 위치에서 목적지로 이동하는 도중, 어떤 곳을 들러야 하는 상황을 상상하자. 예를 들어: 적당한 식당에서 밥을 먹고 다음 강의실로 간다거나, 친구집에 놀러가는데 편의점에 들러 맥주를 사 간다거나.

현재 위치로부터, 어떤 위치 하나를 들러, 목적지로 이동할 것이다. 그 위치는 아래의 조건들을 만족해야 한다.

- 위치정보들의 목록에 있고,
- 그 위치에 키워드는 주어진 키워드들을 모두 포함하고 있으며,
- 이 곳을 들러 목적지까지 이동할 때 총 거리가, 주어진 가능한 최대 거리보다 짧아야 한다.

위치정보들의 목록에서 이 조건을 만족하는 모든 곳의 이름들의 리스트를 돌려주는 함수 `stop_by()`를 작성해 본다. (`function: float, float, float, float, list[string], float -> list[string]`)

위도와 경도로 표현되는 위치에 대해, 두 지점 사이의 거리는 복잡한 계산을 필요로 한다. 이를 계산해주는 함수 `dist()`가 숙제1에서 제공된 바 있다. 같은 라이브러리를 사용하여 계산하면 된다.