Задания

Покажите карту Москвы, добавив на неё метки стадионов:

Спартак

Динамо

Лужники

Полезным будет использование словаря с координатами

stadiums\_location = {"Лужники": "37.554191,55.715551",

                     "Спартак": "37.440262,55.818015",

                     "Динамо": "37.559809,55.791540"

                     }

Код:

import folium

stadiums\_location = {

    "Лужники": [55.715551, 37.554191],

    "Спартак": [55.818015, 37.440262],

    "Динамо": [55.791540, 37.559809]

}

# Создаем базовую карту с центром в Москве

moscow\_map = folium.Map(location=[55.755825, 37.617298], zoom\_start=11)

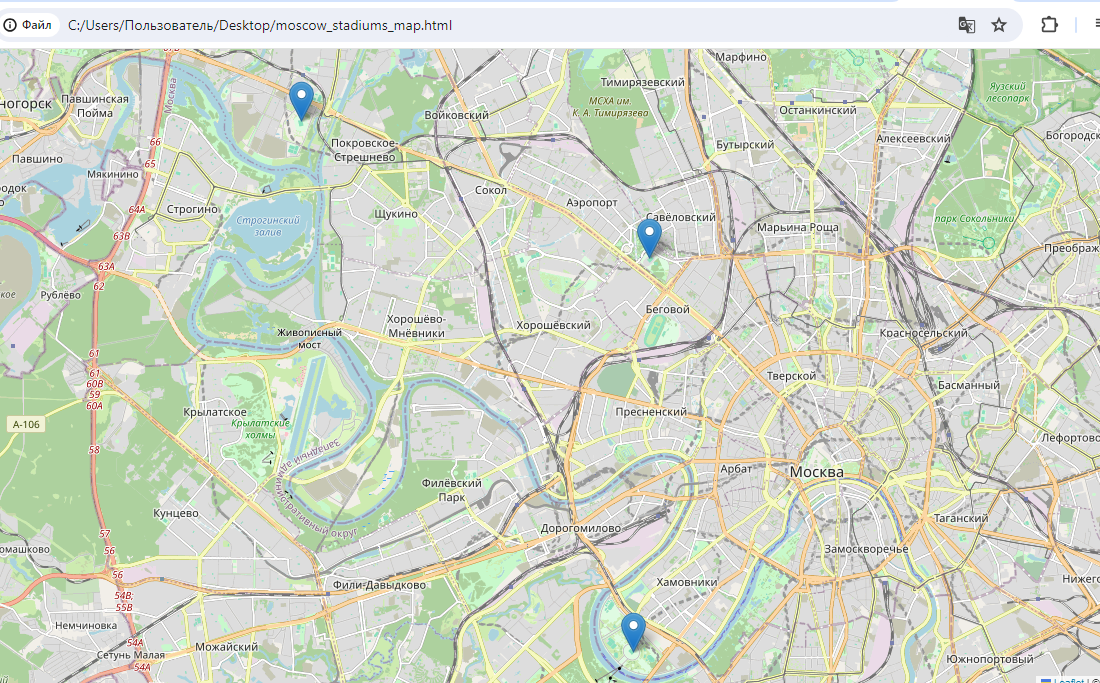
# Добавляем метки стадионов на карту

for stadium, location in stadiums\_location.items():

    folium.Marker(location=location, tooltip=stadium).add\_to(moscow\_map)

# Сохраняем карту в HTML файл

moscow\_map.save("moscow\_stadiums\_map.html")



Задания

2. Определите длину пути, заданного последовательностью точек.

Отобразите заданный путь на карте, а в средней его точке поставьте метку.

Последовательность точек задайте по своему усмотрению, например, списком координат.

Код:

import folium

from geopy.distance import geodesic

# Задаем последовательность точек в виде списка координат [широта, долгота]

path\_coordinates = [

    (55.7558, 37.6173), # Москва

    (59.9343, 30.3351), # Санкт-Петербург

    (56.8380, 60.5970), # Екатеринбург

    (45.0355, 38.9753)  # Краснодар

]

# Создаем базовую карту с центром в середине пути

average\_point = [sum(p[0] for p in path\_coordinates)/len(path\_coordinates),

                 sum(p[1] for p in path\_coordinates)/len(path\_coordinates)]

path\_map = folium.Map(location=average\_point, zoom\_start=4)

# Добавляем метку в среднюю точку пути

folium.Marker(location=average\_point, popup="Средняя точка пути",

              icon=folium.Icon(color='green')).add\_to(path\_map)

# Добавляем линию, представляющую путь, на карту

folium.PolyLine(path\_coordinates, color="blue", weight=2.5, opacity=1).add\_to(path\_map)

# Определяем длину пути

distance = 0

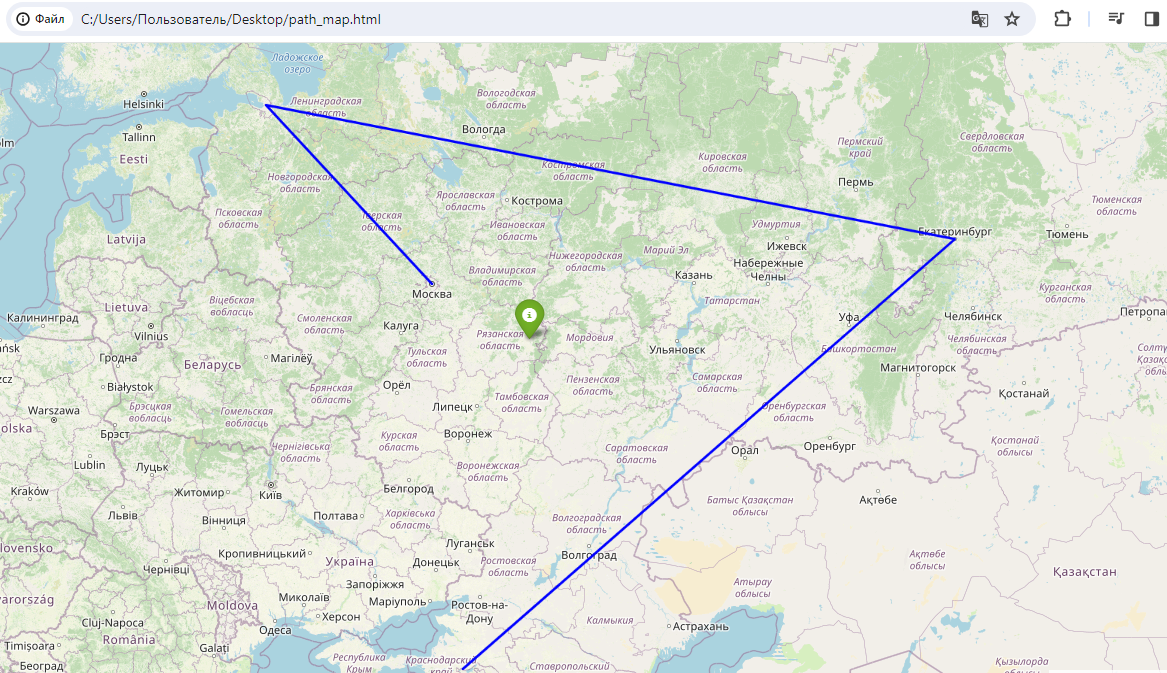
for i in range(len(path\_coordinates)-1):

    distance += geodesic(path\_coordinates[i], path\_coordinates[i+1]).kilometers

print("Длина пути:", distance, "километров")

# Сохраняем карту в HTML файл

path\_map.save("path\_map.html")



Задания

3.   Напишите программу, которая определяет, какой из списка городов расположен южнее всех остальных. Программа должна быть реализована как консольное приложение, список городов вводится с клавиатуры через запятую.

В результате своей работы программа должна напечатать название самого южного из введённых городов.

Код:

Эта программа определяет самый южный город из списка, введенного пользователем. Она использует словарь координат для определения широты каждого города и затем находит город с самой южной широтой.

def find\_southernmost\_city(cities):

    city\_coordinates = {

        "Москва":  (55.755825, 37.617298),

        "Санкт-Петербург": (59.9343, 30.3351),

        "Екатеринбург": (56.8380, 60.5970),

        "Краснодар": (45.0355, 38.9753)

        # Добавьте остальные города и их координаты

    }

    southernmost\_city = None

    southernmost\_latitude = 90  # Начальное значение для сравнения

    for city in cities:

        if city in city\_coordinates:

            latitude = city\_coordinates[city][0]

            if latitude < southernmost\_latitude:

                southernmost\_latitude = latitude

                southernmost\_city = city

    return southernmost\_city

# Получаем список городов от пользователя

cities = input("Введите названия городов через запятую: ").split(",")

# Находим самый южный город

southernmost = find\_southernmost\_city([city.strip() for city in cities])

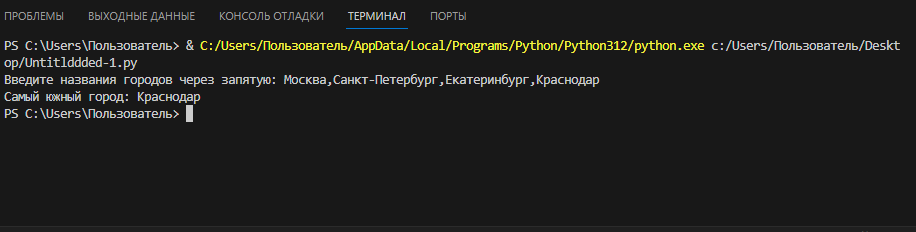
# Выводим результат

if southernmost:

    print(f"Самый южный город: {southernmost}")

else:

    print("Указаны недопустимые города")



Задания

Найдите ближайшую аптеку к точке, заданной адресом, который пользователь вводит с клавиатуры.

Код:

import requests

def find\_nearest\_pharmacy(address):

    # Получаем координаты заданного адреса с помощью геокодирования OpenStreetMap

    response = requests.get(f"https://nominatim.openstreetmap.org/search?format=json&q={address}")

    location = response.json()[0]  # Берем первый результат

    # Ищем ближайшие аптеки к полученным координатам

    pharmacy\_response = requests.get(f"https://nominatim.openstreetmap.org/search?format=json&amenity=pharmacy&lat={location['lat']}&lon={location['lon']}&limit=1")

    pharmacy = pharmacy\_response.json()[0]  # Берем ближайшую аптеку

    return pharmacy

# Получаем адрес от пользователя

user\_address = input("Введите адрес: ")

# Находим ближайшую аптеку

nearest\_pharmacy = find\_nearest\_pharmacy(user\_address)

# Выводим результат

print(f"Ближайшая аптека находится по адресу: {nearest\_pharmacy['display\_name']}")

программа отправляет запрос к сервисам геокодирования и поиска объектов OpenStreetMap для нахождения координат указанного адреса и ближайшей аптеки. Результатом будет вывод адреса ближайшей аптеки.

6.  Создайте заготовку к игре «угадай-ка город».

import folium

import random

import geopy

from geopy.geocoders import Nominatim

# Ваш API ключ

API\_KEY = "3d8d59aa-3220-4de8-8596-5a1dc8aa9be7"

# Список городов

cities = ["Москва", "Санкт-Петербург", "Новосибирск", "Екатеринбург", "Нижний Новгород", "Казань", "Челябинск", "Омск", "Самара", "Ростов-на-Дону"]

def get\_random\_city():

    return random.choice(cities)

def get\_random\_map\_part(city):

    geolocator = Nominatim(user\_agent="my\_app")

    location = geolocator.geocode(city)

    zoom = random.randint(12, 15)

    width = random.randint(200, 400)

    height = random.randint(200, 400)

    offset = random.randint(-width // 2, width // 2), random.randint(-height // 2, height // 2)

    map = folium.Map(location=[location.latitude, location.longitude], zoom\_start=zoom)

    folium.Rectangle(bounds=[[location.latitude + offset[0], location.longitude + offset[1]],

                            [location.latitude + offset[0] + width, location.longitude + offset[1] + height]], color="red").add\_to(map)

    if random.random() < 0.5:

        folium.TileLayer('cartodbpositron').add\_to(map)

    else:

        folium.TileLayer('satellite').add\_to(map)

    map.save(f"{city}.html")

city = get\_random\_city()

get\_random\_map\_part(city)

Задания

**8.** Напишите программу, считающую (приближённо) расстояние от вашего дома до университета

Адреса УУНиТ и дома вводятся пользователем в консоль после запуска программы, а в качестве метрики расстояния вам нужно использовать декартову метрику на градусной сетке, считая 1 градус по широте равным 111 километрам, а отношение градуса широты и градуса долготы, равным косинусу широты.

Пояснение: На экваторе длина градуса широты и долготы в метрах совпадают, и примерно равны 111 километрам (длина экватора примерно 40000 км, поделить на 360 градусов). Однако с увеличением широты появляется разница, так как длина окружности по параллели с ростом широты уменьшается. Представьте себе сечения земного шара плоскостями, параллельными экватору. На полюсах это сечение вырождается в точку касания. Поэтому там длина градуса долготы стремится к нулю. Несложно заметить, что эта зависимость длины градуса по долготе от широты точки выражается косинусом широты. Такой способ вычисления расстояний между точками, заданными широтой и долготой подходит, если точки не сильно удалены друг от друга. На больших расстояниях такой метод дает большую погрешность и потому неприменим.

Итого: 1 грд. широты = 111 км. 1 грд. долготы = 111 км \* cos(широты).

Код:

import math

# Вводим координаты дома

home\_lat = float(input("Введите широту вашего дома: "))

home\_long = float(input("Введите долготу вашего дома: "))

# Вводим координаты университета

uni\_lat = float(input("Введите широту университета: "))

uni\_long = float(input("Введите долготу университета: "))

# Переводим градусы в радианы

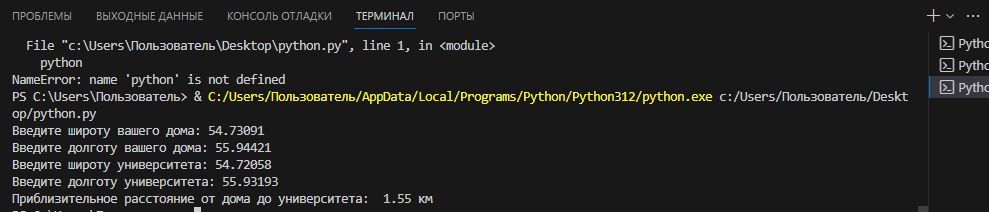
home\_lat\_rad = math.radians(home\_lat)

uni\_lat\_rad = math.radians(uni\_lat)

# Считаем декартову метрику

distance = 111 \* math.sqrt((home\_lat - uni\_lat)\*\*2 + (home\_long - uni\_long)\*\*2 \* math.cos((home\_lat\_rad + uni\_lat\_rad) / 2))

print("Приблизительное расстояние от дома до университета: ", round(distance, 2), "км")



7. Напишите программу, которая поможет вам определить, в каком районе находится заданный адрес. Адрес передаётся в командной строке.

Подсказка. По введённому адресу надо сперва найти координаты, а с полученными координатами надо «сходить» в геокодер, указав в параметре kind значение district (район).

import requests

def find\_district(address):

    api\_key = '3d8d59aa-3220-4de8-8596-5a1dc8aa9be7'

    try:

        geocode\_response = requests.get(f'https://geocode-maps.yandex.ru/1.x/?apikey={api\_key}&format=json&geocode={address}')

        geocode\_data = geocode\_response.json()

        coordinates = geocode\_data['response']['GeoObjectCollection']['featureMember'][0]['GeoObject']['Point']['pos']

        longitude, latitude = map(float, coordinates.split())

        district\_response = requests.get(f'https://geocode-maps.yandex.ru/1.x/?apikey={api\_key}&format=json&geocode={longitude},{latitude}&kind=district')

        district\_data = district\_response.json()

        district = district\_data['response']['GeoObjectCollection']['featureMember'][0]['GeoObject']['name']

        return district

    except Exception as e:

        return "Ошибка при определении района:", str(e)

if \_\_name\_\_ == "\_\_main\_\_":

    address = input("Введите адрес: ")

    result = find\_district(address)

    print("Район, в котором находится адрес:", result)