1. 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"

}

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.

3. Напишите программу, которая принимает на вход координаты объекта и записывает в png-файл его спутниковый снимок.

4. 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.

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

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

Суть игры в следующем: один игрок загадывает город и показывает другому игроку некоторую часть карты (или фотоснимка) этого города. Второй игрок должен город угадать. Затем они меняются инициативой. Конечно же, на показываемом участке не должно фигурировать имя города.

Пусть дан список городов, заданный константно в коде программы.

Пользуясь функцией подбора точки-размеров города, сформируйте запрос и получите спутниковый снимок или карту части города (так, чтобы название города не отображалось на этом масштабе).

Что показывать — спутник или карту — выбирайте случайным образом.

Далее в режиме слайд-шоу пролистываем рисунки. Порядок следования слайдов также случайный.

Примечание. Ключевым пунктом является работа с API, поэтому не важно, как именно будет рисоваться полученное изображение. В дальнейшем эту наработку можно использовать для проекта или его части: предлагать пользователю угадать и ввести название города, считать правильные и неправильные ответы и т.п.

Попробуйте подойти к задаче творчески.

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).

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).

Проверьте: расстояние между точками (долгота, широта): (37.6222, 55.7566) и (37.6589, 55.7583) составляет 2300 метров.

Полезной будет функция, вычисляющая расстояния по данному алгоритму

import math

# Определяем функцию, считающую расстояние между двумя точками, заданными координатами

def lonlat\_distance(a, b):

degree\_to\_meters\_factor = 111 \* 1000 # 111 километров в метрах

a\_lon, a\_lat = a

b\_lon, b\_lat = b

# Берем среднюю по широте точку и считаем коэффициент для нее.

radians\_lattitude = math.radians((a\_lat + b\_lat) / 2.)

lat\_lon\_factor = math.cos(radians\_lattitude)

# Вычисляем смещения в метрах по вертикали и горизонтали.

dx = abs(a\_lon - b\_lon) \* degree\_to\_meters\_factor \* lat\_lon\_factor

dy = abs(a\_lat - b\_lat) \* degree\_to\_meters\_factor

# Вычисляем расстояние между точками.

distance = math.sqrt(dx \* dx + dy \* dy)

return distance