empenoso
12 дек 2024 в 03:23

С бумаги на цифровую карту: генерация файла из таблицы для импорта на карту и геокодирование адресов с помощью Python

Простой 15 мин 2.5K

Python*, Open source*, Геоинформационные сервисы*, OpenStreetMap*

Кейс

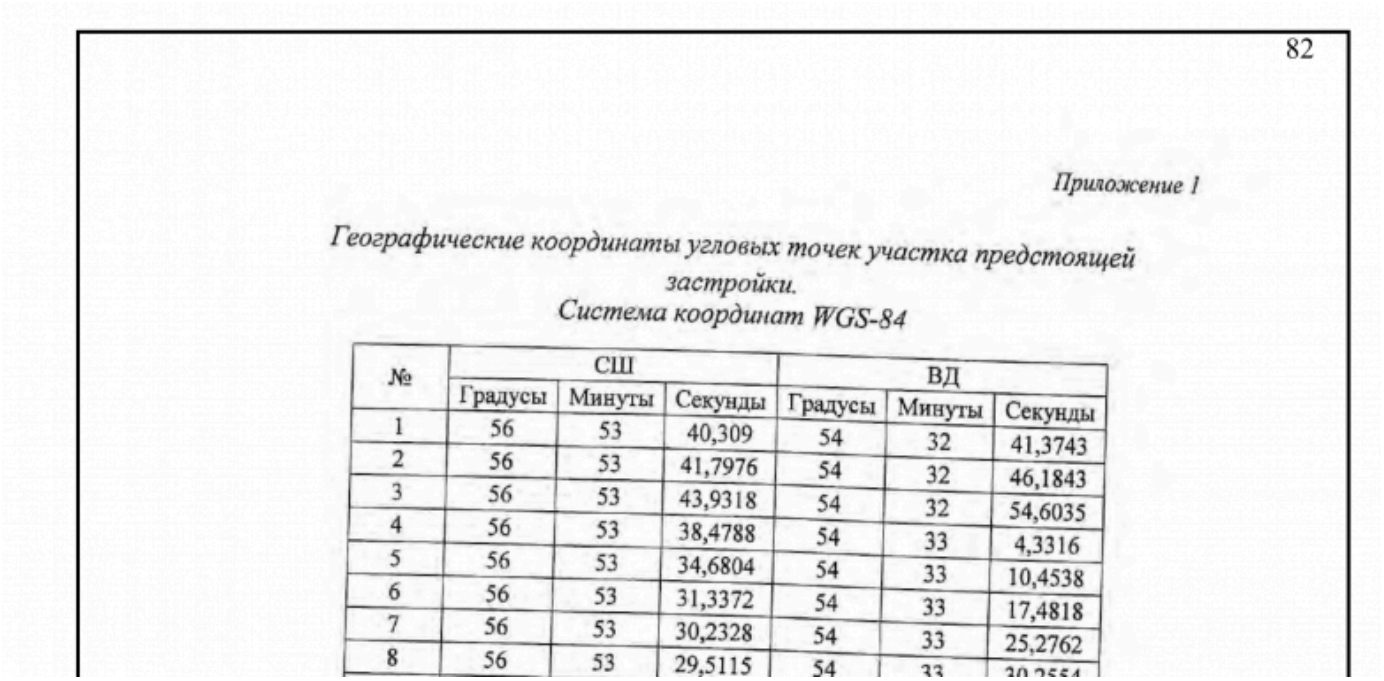
Сразу возникает вопрос - кому в 2024 году может понадобиться переносить данные с бумажного носителя на цифровой, ведь большинство данных уже в цифровом виде. Тем не менее есть реальная задача. В исходных данных - растровая картинка проекта в виде таблицы с географическими координатами, выраженными в градусах, минутах и секундах, а на выходе должно получиться текстовое описание маршрутов с длинами и карта с точками и сегментами.

Предстоящие действия включают следующие шаги: из бумажного проекта взять таблицу с географическими координатами предстоящей застройки, оцифровать эти данные, а затем с помощью Python скрипта создать GPX-файл с точками и отрезками для нанесения на карту.

Затем, создав другой Python-скрипт, провести геокодирование координат для получения текстовых описаний с адресами и автоматически рассчитать расстояния между точками и сегментами.

Все эти действия гораздо быстрее ручного нанесения точек на карту и ручного подсчёта расстояний.

Исходные данные



82

Приложение 1

Географические координаты угловых точек участка предстоящей застройки.
Система координат WGS-84

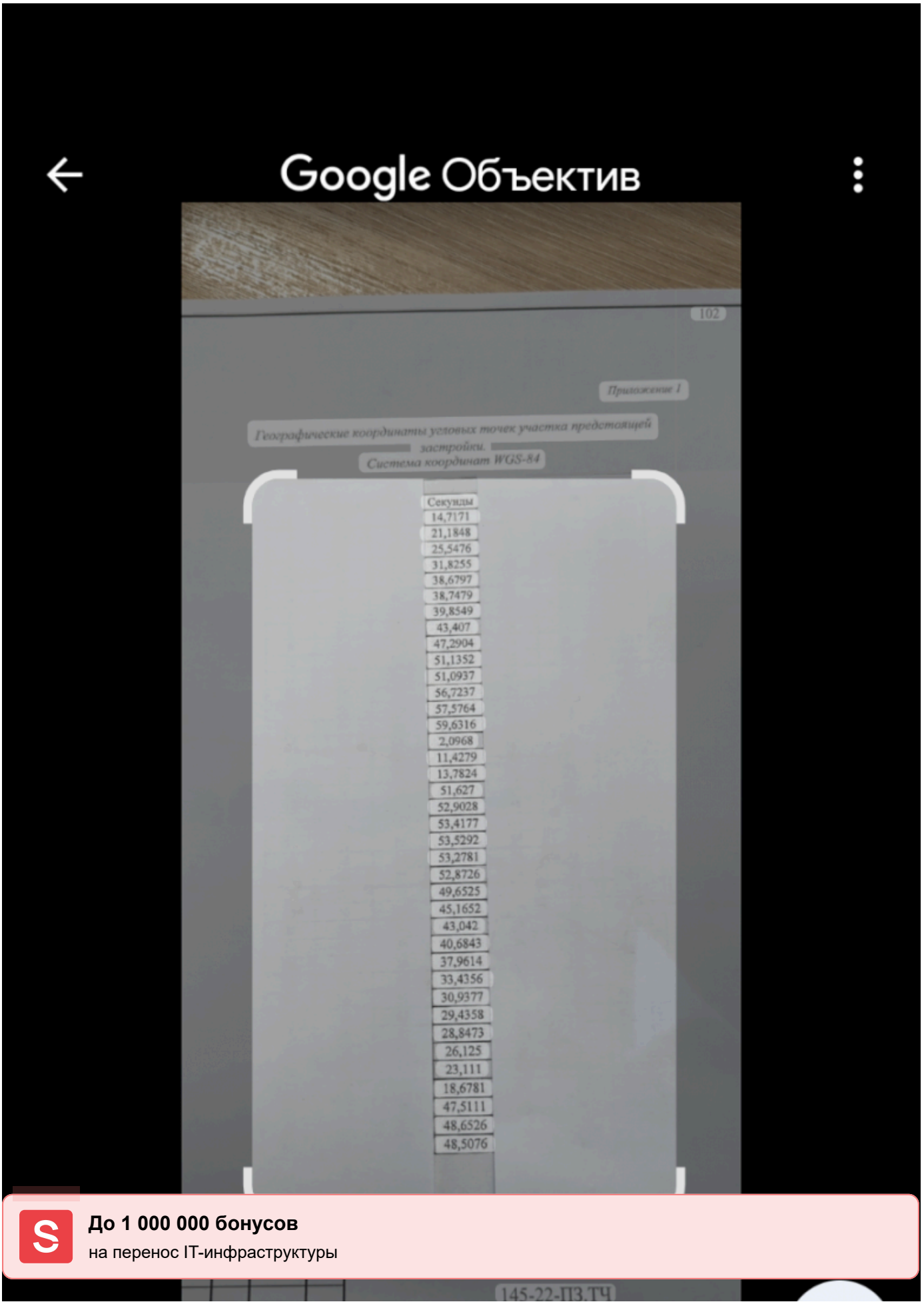
№	СШ			ВД		
	Градусы	Минуты	Секунды	Градусы	Минуты	Секунды
1	56	53	40,309	54	32	41,3743
2	56	53	41,7976	54	32	46,1843
3	56	53	43,9318	54	32	54,6035
4	56	53	38,4788	54	33	4,3316
5	56	53	34,6804	54	33	10,4538
6	56	53	31,3372	54	33	17,4818
7	56	53	30,2328	54	33	25,2762
8	56	53	29,5115	54	33	30,2654

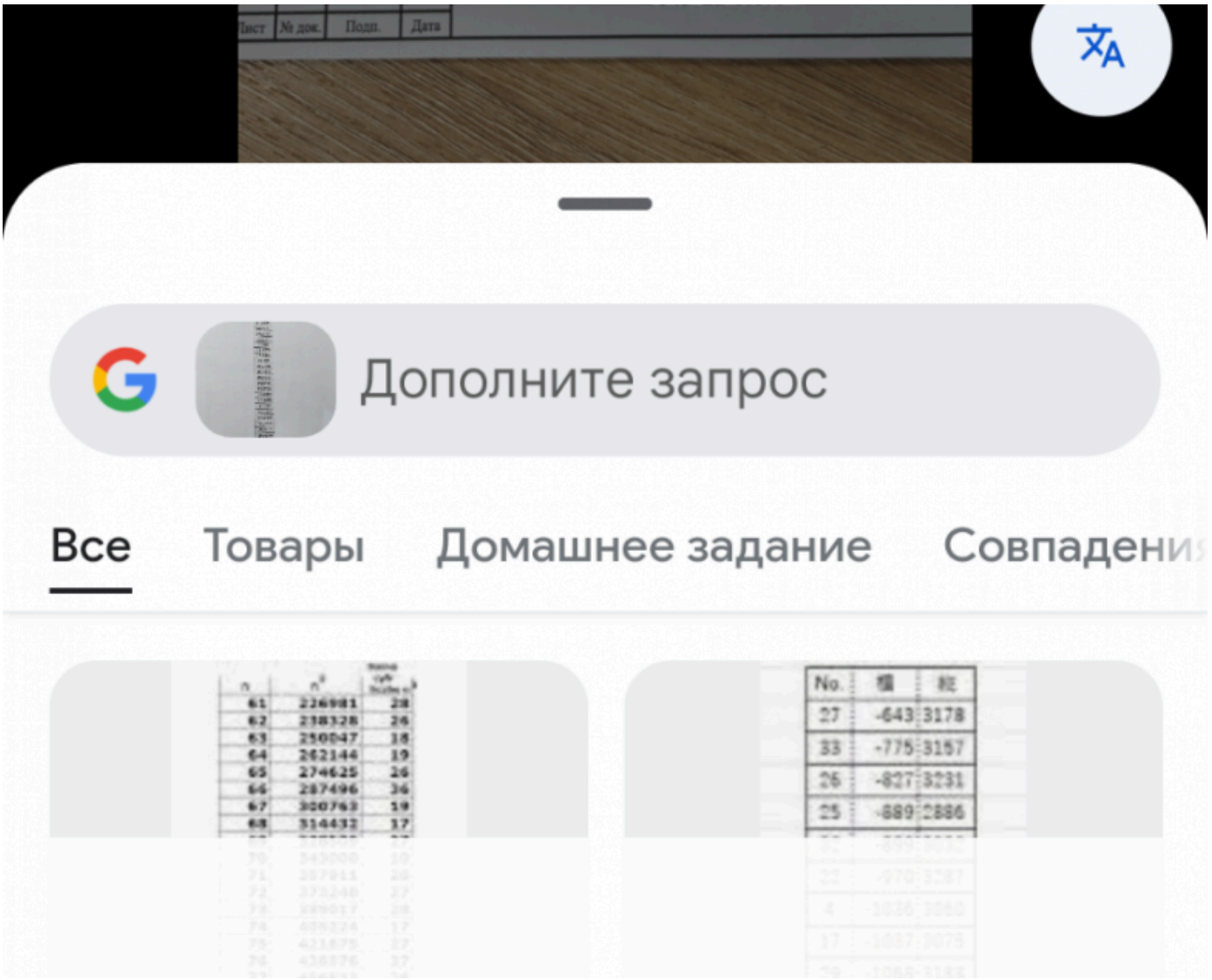
Лист проекта

В документе содержатся растровые изображения таблиц с географическими координатами планируемой застройки. Есть несколько проектов, причем все изображения вставлены одинаково небрежно – с наклоном только в разные стороны.

Поскольку страниц, содержащих точки не так много - всего по две страницы на проект, то выбрал использовать телефон с Google Lens (Гугл Объектив), вместо специализированной программы для оптического распознавания символов.

С помощью Google Объектив, закрывая двумя кусочками страницы соседние столбцы можно легко и корректно распознать полностью всю таблицу. Это быстро и является хорошим вариантом при отсутствии сканера.





Google Объектив для распознавания таблицы

Подготовка данных

Мне показалось правильным перевести градусы, минуты и секунды в десятичные градусы следующим образом:

> Десятичные градусы = градусы + (минуты / 60) + (секунды / 3600)

Провёл все вычисления в таблице:

Далее в Notepad++ при помощи макросов привёл данные к неизменяемому виду данных в Python, который используется для хранения упорядоченной последовательности элементов. Такая запись в Python называется кортежем (`tuple`). Кортеж представляет собой неизменяемый упорядоченный набор элементов, заключённых в круглые скобки. Каждый элемент кортежа отделяется запятой.

Генерация GPX файла

всего без разницы какой формат выбрать для этой промежуточной стадии.

GPX (GPS eXchange Format) - это формат хранения и обмена данными устройств позиционирования GPS. Был создан в 2002, файл может содержать различные элементы, такие как треки <rte> и путевые точки <trk>.

Visual Studio Code

Python код генерации. Скрипт начинается с настройки среды для обработки выходных данных в кодировке UTF-8 и импортирует необходимую библиотеку XML (`xml.etree.ElementTree`). Это гарантирует, что выходной файл и любые сообщения терминала будут правильно обрабатывать специальные символы. В самом начале идёт определение данных:

- **Координаты:** определяется список пар широты и долготы. Каждая координата соответствует определенной географической точке. Скрипт начинает нумерацию этих точек с `1` , хотя в начале добавляется неиспользуемая точка-заполнитель для целей индексации.
- **Сегменты:** Набор списков определяет «треки» или «маршруты», которые являются последовательностями точек, представленных их индексами в списке координат.

- **Путевые точки:** каждая координата добавляется как элемент (путевая точка). Вложенный элемент назначает метку, например «Точка 1», «Точка 2» и т. д.
- **Треки:** список segments используется для определения элементов (трек). Каждый трек имеет для идентификации (например, «Сегмент 1») и содержит последовательность элементов (точка трека), соответствующих индексам в сегменте. Они также включают элементы для маркировки.

Сконструированное дерево XML сохраняется в файле с именем `output.gpx` с кодировкой UTF-8 и декларацией XML. Подтверждающее сообщение выводится на консоль. Точки приведены просто как пример:

```
import sys
import time
sys.stdout.reconfigure(encoding='utf-8')

import xml.etree.ElementTree as ET

# Список координат
coordinates = [
        (56.89453028, 54.54482619), # точка 0, которой не существует и которая введена
        (56.89453028, 54.54482619), # точка 1 по документам
        (56.89494378, 54.54616231),
        (56.89553661, 54.54850097),
        (56.89402189, 54.55120322),
        (56.89296678, 54.55290383),
        (56.89203811, 54.55485606),
        (56.89173133, 54.55702117),
        (56.89153097, 54.55840428),
        (56.89142722, 54.56011819),
        (56.89126439, 54.56732872),
        (56.89131197, 54.57094603),
        (56.89005297, 54.57331617),
        (56.88962892, 54.57386478),
        (56.88968192, 54.57491517),
        (56.88949189, 54.57596556),
        (56.89364464, 54.55838172),
        (56.89369056, 54.55817894),
        (56.89001561, 54.56590686),
```

```
        (56.89127153, 54.56665189),
        (56.88974133, 54.57257369),
        (56.88887294, 54.56933550)
]

# Сегменты
segments = [
    [1, 2, 3, 4, 5, 6, 7, 8, 9, 22, 10, 11, 12, 13],  # Сегмент 1: Точки 1, 2, 3, ..
    [7, 16, 17],
    [18, 19, 20, 21, 22]
]

# Создаем корневой элемент GPX
gpx = ET.Element('gpx', version="1.1", xmlns="http://www.topografix.com/GPX/1/1")

# Добавляем точки в GPX как waypoints
for idx, (lat, lon) in enumerate(coordinates, start=1):
    wpt = ET.SubElement(gpx, 'wpt', lat=str(lat), lon=str(lon))
    name = ET.SubElement(wpt, 'name')
    name.text = f"Point {idx}"

# Добавляем сегменты как треки
for segment_idx, segment in enumerate(segments, start=1):
    trk = ET.SubElement(gpx, 'trk')
    trk_name = ET.SubElement(trk, 'name')
    trk_name.text = f"Segment {segment_idx}"

    trkseg = ET.SubElement(trk, 'trkseg')

    for point_idx in segment:
        lat, lon = coordinates[point_idx]
        trkpt = ET.SubElement(trkseg, 'trkpt', lat=str(lat), lon=str(lon))
        name = ET.SubElement(trkpt, 'name')
        name.text = f"Point {point_idx + 1}"

# Создаем и записываем GPX файл
tree = ET.ElementTree(gpx)
tree.write('output.gpx', xml_declaration=True, encoding='utf-8')

print("GPX файл успешно создан и сохранен как 'output.gpx'")
```


На выходе GPX файл (сделал красивым, иначе в одну строчку был).

```
<?xml version='1.0' encoding='utf-8'?>
<gpx version="1.1"
    xmlns="http://www.topografix.com/GPX/1/1">
    <wpt lat="56.89453028" lon="54.54482619">
        <name>Point 1</name>
    </wpt>
    <wpt lat="56.89453028" lon="54.54482619">
        <name>Point 2</name>
    </wpt>
    <wpt lat="56.89494378" lon="54.54616231">
        <name>Point 3</name>
    </wpt>
    <wpt lat="56.89553661" lon="54.54850097">
        <name>Point 4</name>
    </wpt>
    <wpt lat="56.89402189" lon="54.55120322">
        <name>Point 5</name>
    </wpt>
    <wpt lat="56.89296678" lon="54.55290383">
        <name>Point 6</name>
    </wpt>
    <wpt lat="56.89203811" lon="54.55485606">
        <name>Point 7</name>
    </wpt>
    <wpt lat="56.89173133" lon="54.55702117">
        <name>Point 8</name>
    </wpt>
    <wpt lat="56.89153097" lon="54.55840428">
        <name>Point 9</name>
    </wpt>
    <wpt lat="56.89142722" lon="54.56011819">
        <name>Point 10</name>
    </wpt>
    <wpt lat="56.89126439" lon="54.56732872">
        <name>Point 11</name>
    </wpt>
    <wpt lat="56.89131197" lon="54.57094603">
        <name>Point 12</name>
    </wpt>
```

```
    <wpt lat="56.88962892" lon="54.57386478">
        <name>Point 14</name>
    </wpt>
    <wpt lat="56.88968192" lon="54.57491517">
        <name>Point 15</name>
    </wpt>
    <wpt lat="56.88949189" lon="54.57596556">
        <name>Point 16</name>
    </wpt>
    <wpt lat="56.89364464" lon="54.55838172">
        <name>Point 17</name>
    </wpt>
    <wpt lat="56.89369056" lon="54.55817894">
        <name>Point 18</name>
    </wpt>
    <wpt lat="56.89001561" lon="54.56590686">
        <name>Point 19</name>
    </wpt>
    <wpt lat="56.89032456" lon="54.56633664">
        <name>Point 20</name>
    </wpt>
    <wpt lat="56.89042917" lon="54.56636036">
        <name>Point 21</name>
    </wpt>
    <wpt lat="56.89065008" lon="54.56661533">
        <name>Point 22</name>
    </wpt>
    <wpt lat="56.89127153" lon="54.56665189">
        <name>Point 23</name>
    </wpt>
    <wpt lat="56.88974133" lon="54.57257369">
        <name>Point 24</name>
    </wpt>
    <wpt lat="56.88887294" lon="54.5693355">
        <name>Point 25</name>
    </wpt>
    <trk>
        <name>Segment 1</name>
        <trkseg>
            <trkpt lat="56.89453028" lon="54.54482619">
```

```
            <name>Point 3</name>
        </trkpt>
        <trkpt lat="56.89553661" lon="54.54850097">
            <name>Point 4</name>
        </trkpt>
        <trkpt lat="56.89402189" lon="54.55120322">
            <name>Point 5</name>
        </trkpt>
        <trkpt lat="56.89296678" lon="54.55290383">
            <name>Point 6</name>
        </trkpt>
        <trkpt lat="56.89203811" lon="54.55485606">
            <name>Point 7</name>
        </trkpt>
        <trkpt lat="56.89173133" lon="54.55702117">
            <name>Point 8</name>
        </trkpt>
        <trkpt lat="56.89153097" lon="54.55840428">
            <name>Point 9</name>
        </trkpt>
        <trkpt lat="56.89142722" lon="54.56011819">
            <name>Point 10</name>
        </trkpt>
        <trkpt lat="56.89127153" lon="54.56665189">
            <name>Point 23</name>
        </trkpt>
        <trkpt lat="56.89126439" lon="54.56732872">
            <name>Point 11</name>
        </trkpt>
        <trkpt lat="56.89131197" lon="54.57094603">
            <name>Point 12</name>
        </trkpt>
        <trkpt lat="56.89005297" lon="54.57331617">
            <name>Point 13</name>
        </trkpt>
        <trkpt lat="56.88962892" lon="54.57386478">
            <name>Point 14</name>
        </trkpt>
    </trkseg>
</trk>
```

```
            <trkpt lat="56.89173133" lon="54.55702117">
                <name>Point 8</name>
            </trkpt>
            <trkpt lat="56.89364464" lon="54.55838172">
                <name>Point 17</name>
            </trkpt>
            <trkpt lat="56.89369056" lon="54.55817894">
                <name>Point 18</name>
            </trkpt>
        </trkseg>
    </trk>
    <trk>
        <name>Segment 3</name>
        <trkseg>
            <trkpt lat="56.89001561" lon="54.56590686">
                <name>Point 19</name>
            </trkpt>
            <trkpt lat="56.89032456" lon="54.56633664">
                <name>Point 20</name>
            </trkpt>
            <trkpt lat="56.89042917" lon="54.56636036">
                <name>Point 21</name>
            </trkpt>
            <trkpt lat="56.89065008" lon="54.56661533">
                <name>Point 22</name>
            </trkpt>
            <trkpt lat="56.89127153" lon="54.56665189">
                <name>Point 23</name>
            </trkpt>
        </trkseg>
    </trk>
</gpx>
```

Отображение GPX-файла с точками и отрезками на карте

GPX файл импортировал в SAS.Planet.Release.241111 для отображения на нужных слоях карты.

снимков и карт из различных онлайн-сервисов, таких как Google Maps, Яндекс.Карты и

другие. Она позволяет сохранять карты и снимки высокого разрешения на локальный компьютер для последующего использования без доступа к интернету.

`Яндекс Карта` и `Rosreestr.ru кадастровые границы`

На карте выбраны слои `Яндекс Карта` и `Rosreestr.ru кадастровые границы` - на них наложены точки и получившиеся сегменты пути.

Из SAS.Planet можно сохранить и распечатать слои с наложенными на них точками в любом формате включая A0 и A1.

Геокодирование координат для получения текстовых описаний с адресами и автоматический расчет расстояний между точками и внутри сегментов

Написал Python код, который производит геокодирование координат для получения текстовых описаний с адресами и делает автоматический расчет расстояний между точками и внутри сегментов. Использовал две библиотеки:

Shapely - это библиотека Python для создания, анализа и манипулирования

Geopy, с другой стороны, ориентирована на геокодирование и геопространственные вычисления. Она преобразует адреса в географические координаты и наоборот, а также может вычислять расстояния между местоположениями, используя различные геодезические методы.

Вместе эти библиотеки предоставляют мощный набор инструментов для обработки и анализа геопространственных данных.

Visual Studio Code

Код Python скрипта. В самом начале задаются:

- **Координаты:** список пар широты и долготы представляет различные географические точки. Первая запись — это заполнитель для выравнивания индексации с удобной для восприятия нумерацией.
- **Сегменты:** это группы точек, идентифицированных по их индексам, которые образуют непрерывные линии или пути.

Дальше библиотеки:

- `shapely.geometry`: используется для создания геометрических представлений, таких

- `geopy` : предоставляет инструменты для расчета расстояний и геокодирования (преобразования координат в адреса).
- `Nominatim` : геокодер из OpenStreetMap, используемый для обратного геокодирования координат в удобные для восприятия адреса.

Основные функции

- Обратное геокодирование: функция `reverse_geocode` преобразует широту и долготу в адреса. Она корректно обрабатывает ошибки, возвращая соответствующее сообщение, если адрес не может быть найден или если есть исключение.
- Расчет расстояния: функция `geodesic` из `geopy` вычисляет расстояние между последовательными точками в метрах.

Для каждого сегмента создаётся отчёт:

1. **Информация о пути:**

- Точки, образующие сегмент, соединяются в линию (`LineString`), а общая длина пути вычисляется путем суммирования расстояний между последовательными точками.
- Эта информация форматируется в виде описания.

2. **Сведения о точке:**

- Каждая точка в сегменте подвергается обратному геокодированию для предоставления адреса, удобного для восприятия человеком.
- Расстояние между каждой парой последовательных точек вычисляется и включается в отчет.

Точки приведены просто как пример:

```
import sys
import time
sys.stdout.reconfigure(encoding='utf-8')
```



```
from geopy.geocoders import Nominatim
```

```
# Определяем координаты точек
coordinates = [
        (56.89453028, 54.54482619), # точка 0, которой не существует и которая введена
        (56.89453028, 54.54482619), # точка 1 по документам
        (56.89494378, 54.54616231),
        (56.89553661, 54.54850097),
        (56.89402189, 54.55120322),
        (56.89296678, 54.55290383),
        (56.89203811, 54.55485606),
        (56.89173133, 54.55702117),
        (56.89153097, 54.55840428),
        (56.89142722, 54.56011819),
        (56.89126439, 54.56732872),
        (56.89131197, 54.57094603),
        (56.89005297, 54.57331617),
        (56.88962892, 54.57386478),
        (56.88968192, 54.57491517),
        (56.88949189, 54.57596556),
        (56.89364464, 54.55838172),
        (56.89369056, 54.55817894),
        (56.89001561, 54.56590686),
        (56.89032456, 54.56633664),
        (56.89042917, 54.56636036),
        (56.89065008, 54.56661533),
        (56.89127153, 54.56665189),
        (56.88974133, 54.57257369),
        (56.88887294, 54.56933550)
]

# Создаем объект геолокатора
geolocator = Nominatim(user_agent="geo_report")

# Функция для обратного геокодирования (получение адреса по координатам)
def reverse_geocode(lat, lon):
    try:
        location = geolocator.reverse((lat, lon), language="ru")
        return location.address if location else "Адрес не найден"
    except Exception as e:
        return f"Ошибка: {e}"
```

```
    [1, 2, 3, 4, 5, 6, 7, 8, 9, 22, 10, 11, 12, 13],  # Сегмент 1: Точки 1, 2, 3
    [7,16,17],
    [18, 19, 20, 21, 22]
]

# Генерация отчета
report = []
for idx, segment in enumerate(segments, start=1):
    points = [coordinates[i] for i in segment]
    line = LineString(points)  # Создаем линию из точек
    total_length = sum(geodesic(points[i], points[i + 1]).meters for i in range(len(poi
    report.append(f"Сегмент {idx}: точки {', '.join(str(i) for i in segment)}")
    report.append(f"Общая длина: {total_length:.2f} метров\n")

    # Описание каждой точки и расстояний между ними
    for i in range(len(points) - 1):
        lat1, lon1 = points[i]
        lat2, lon2 = points[i + 1]
        address1 = reverse_geocode(lat1, lon1)
        address2 = reverse_geocode(lat2, lon2)
        distance = geodesic((lat1, lon1), (lat2, lon2)).meters
        report.append(f"Точка {segment[i]}: {address1} [{lat1}, {lon1}]")
        report.append(f"Точка {segment[i + 1]}: {address2} [{lat2}, {lon2}]")
        report.append(f"Расстояние между точками {segment[i]} и {segment[i + 1]}: {dist

# Вывод отчета
print("\n".join(report))
```

Результат выполнения скрипта:

```
Сегмент 1: точки 1, 2, 3, 4, 5, 6, 7, 8, 9, 22, 10, 11, 12, 13
Общая длина: 2048.85 метров

Точка 1: 2, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволжс
Точка 2: 2, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволжс
Расстояние между точками 1 и 2: 93.55 метров
```

```
Точка 3: 1, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволжс
Точка 4: 18А, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Расстояние между точками 3 и 4: 235.74 метров

Точка 4: 18А, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Точка 5: 22, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволж
Расстояние между точками 4 и 5: 156.68 метров

Точка 5: 22, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволж
Точка 6: 24, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволж
Расстояние между точками 5 и 6: 157.64 метров

Точка 6: 24, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволж
Точка 7: 19, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволж
Расстояние между точками 6 и 7: 136.31 метров

Точка 7: 19, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволж
Точка 8: улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволжский
Расстояние между точками 7 и 8: 87.20 метров

Точка 8: улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволжский
Точка 9: 36, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволж
Расстояние между точками 8 и 9: 105.10 метров

Точка 9: 36, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволж
Точка 22: 56, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Расстояние между точками 9 и 22: 398.60 метров

Точка 22: 56, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Точка 10: 56, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Расстояние между точками 22 и 10: 41.26 метров

Точка 10: 56, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Точка 11: 70, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Расстояние между точками 10 и 11: 220.54 метров

Точка 11: 70, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Точка 12: 80, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Расстояние между точками 11 и 12: 201.31 метров
```

```
Расстояние между точками 12 и 13: 57.86 метров


Сегмент 2: точки 7, 16, 17
Общая длина: 242.00 метров

Точка 7: 19, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволж
Точка 16: улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволжски
Расстояние между точками 7 и 16: 228.63 метров

Точка 16: улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволжски
Точка 17: улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Приволжски
Расстояние между точками 16 и 17: 13.37 метров


Сегмент 3: точки 18, 19, 20, 21, 22
Общая длина: 153.32 метров

Точка 18: 41, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Точка 19: 41, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Расстояние между точками 18 и 19: 43.24 метров

Точка 19: 41, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Точка 20: 41, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Расстояние между точками 19 и 20: 11.74 метров

Точка 20: 41, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Точка 21: 41, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Расстояние между точками 20 и 21: 29.10 метров

Точка 21: 41, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Точка 22: 56, улица Чечкина, Вассята, Чайковский городской округ, Пермский край, Привол
Расстояние между точками 21 и 22: 69.24 метров


[Done] exited with code=0 in 22.818 seconds
```

Такой текстовый результат полностью устраивал.

Гораздо быстрее получилось создать автоматизацию для получения текстовое описания маршрутов с длинами и отображения карты с точками и сегментами, чем вручную наносить точки на карту и вручную делать расчёт расстояний.

Автор: Михаил Шардин
Моя онлайн-визитка
Telegram «Умный Дом Инвестора»

12 декабря 2024 г.

Теги: gpx, проектирование, SAS.Planet, Geopy, Shapely, геокодирование

Хабы: Python, Open source, Геоинформационные сервисы, OpenStreetMap

Комментарии 8

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

Exosphere
15 часов назад

Ещё 10 ошибок авторов Хабра

11 мин 3.2K

+93 35 65

vital_pavlenko
22 часа назад

Больше нет входа в IT. Только выход

2 мин 77K

duran-duran
21 час назад

Трамплин в интернет: как мы ускорили запуск Яндекс Браузера

6 мин 3.3K

+42 9 29

Nickmob
16 часов назад

Введение в Angie: краткая история и отличия от Nginx

Простой 7 мин 2.8K

Ретроспектива

+37 29 8

Myskat_90
21 час назад

Распределённый инференс и шардирование LLM. Часть 1: настройка GPU, проброс в Proxmox и настройка Kubernetes

Сложный 14 мин 1.8K

Туториал

+33 46 0

slava_rumin
15 часов назад

Мое производство электрощитов приносит 40 млн в год. Спасибо нейросетям и СССР за конструкторскую школу

Простой 14 мин 22K

Интервью

+28 48 53

Важное обновление BatteryTest 2

Простой 3 мин 2.5K

+27 37 9

ru_vds
16 часов назад

Как создавались вокальные эффекты Daft Punk

Средний 13 мин 840

Обзор Перевод

+23 5 1

GordienkoAnd
19 часов назад

Как настраивать сети: готовые решения Selectel для максимальной отказоустойчивости

Средний 20 мин 2.5K

Обзор

+21 28 0

alizar
20 часов назад

Почему из технологий делают культы

Простой 8 мин 1.3K

Обзор

+21 7 6

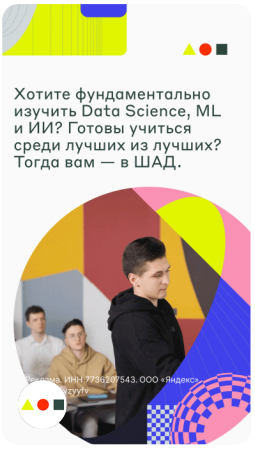
Превращаем бездушные тикеты в эпическое приключение

Промо

ИСТОРИИ

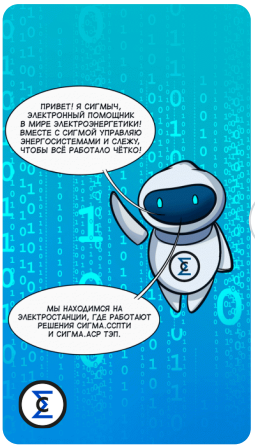
Старт гонки разработчиков

Торопись в сезон Open source



Открыт приём в Школу анализа данных

С Днём радио!



HELLO, WORLD — теперь на 110 киловольтах

Буду лучш

ВАКАНСИИ

Python-разработчик
от 2 000 до 3 500 $ · BCraft · Можно удаленно

Python разработчик
от 150 000 до 250 000 ₽ · Data Compass · Можно удаленно

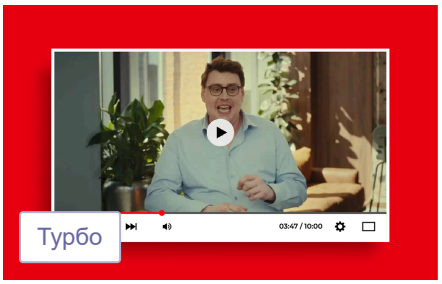
Python разработчик Senior
от 200 000 до 300 000 ₽ · Туроператор «Русь» · Москва

Python разработчик
от 1 500 до 3 000 $ · DevTeam.Space · Москва · Можно удаленно

FullStack QA (Python and NodeJS)
до 4 500 $ · Wanted. · Можно удаленно

Больше вакансий на Хабр Карьере

МИНУТОЧКУ ВНИМАНИЯ



Синтезируем речь без потери тембра спикера

Гонка разработчиков backend, frontend, mobile уже началась! Займи место на старте

Экономим деньги со скидками в Промокодусе

РАБОТА

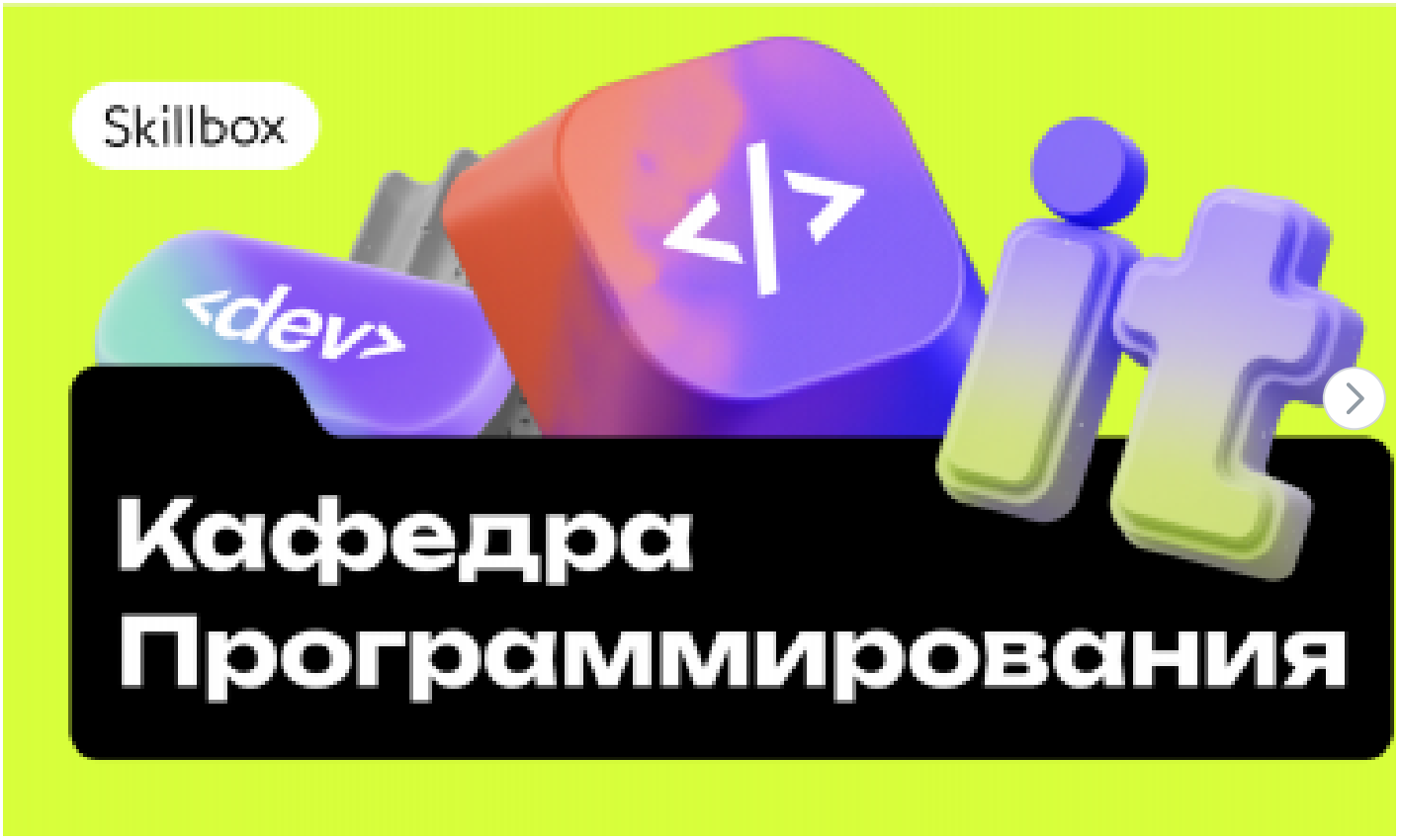
Django разработчик
17 вакансий

Python разработчик
62 вакансии

Data Scientist
43 вакансии

Все вакансии

БЛИЖАЙШИЕ СОБЫТИЯ



17 апреля – 29 мая

Серия бесплатных офлайн-конференций «Кафедра Программировани от Skillbox

Москва

Разработка

Больше событий в календаре

Хабр

Техническая поддержка