NMEA decoding

Questions and Answers

Decode the GPS positions from the two files given in the NMEA 0183 (Links to an external site.) ASCII format.

1. Look at output0.nmea and find the GPS coordinates in the file. Where are those points?

Answer:

Python Code

```python
import folium
import webbrowser
# Define the path of the uploaded file
file_path = '/content/output0.nmea'

# Function to extract GPS coordinates from GGA and RMC sentences
def extract_coordinates(file_path):
    coordinates = []
    with open(file_path, 'r') as file:
        for line in file:
            if "$GPGGA" in line or "$GPRMC" in line:
                parts = line.split(",")
                if len(parts) > 6:
                    # Latitude
                    raw_lat = parts[2]
                    lat_dir = parts[3]
                    # Longitude
                    raw_lon = parts[4]
                    lon_dir = parts[5]

                    # Ensure the fields are valid before processing
                    if raw_lat.replace('.', '', 1).isdigit() and raw_lon.replace(
    ↪ '.', '', 1).isdigit():
                        # Convert to decimal degrees
                        lat = float(raw_lat[:2]) + float(raw_lat[2:]) / 60.0
                        lon = float(raw_lon[:3]) + float(raw_lon[3:]) / 60.0
                        if lat_dir == "S":
                            lat = -lat
                        if lon_dir == "W":
                            lon = -lon
                        coordinates.append((lat, lon))
    return coordinates

# Extract coordinates from GGA and RMC sentences
coordinates = extract_coordinates(file_path)
print("GPS Coordinates", coordinates)
# Create and save the map for the extracted coordinates
if coordinates:
    gps_map = folium.Map(location=coordinates[0], zoom_start=15)
    for coord in coordinates:
```

```
        folium.Marker(location=coord, popup=f"Lat: {coord[0]}, Lon: {coord[1]}").
    ↪ add_to(gps_map)
    gps_map_path = 'gps_map.html'
    gps_map.save(gps_map_path)
    print(f"GPS Map saved at: {gps_map_path}")
else:
    print("No GPS coordinates found.")


# Generate Google Maps URL for the coordinates
def generate_google_maps_url(lat, lon):
    return f"https://www.google.com/maps?q={lat},{lon}"

# Coordinates to open in Google Maps
urls = [generate_google_maps_url(lat, lon) for lat, lon in coordinates]

# Open each location in the web browser
for url in urls:
    webbrowser.open(url)
print(urls)
```

0.0.1 Output

```
GPS Coordinates [(47.49836666666667, 19.040433333333333), (47.49956666666667, 19.04675)]
GPS Map saved at: gps_map.html
```

Figure 1: GPS Coordinates

The extracted latitude and longitude values are:
Latitude: 47.49837, Longitude: 19.04043
Location: closer to Clark Ádám tér, located on the Buda side near the Chain Bridge.(Figure 2 Latitude: 47.49957, Longitude: 19.04675
Location: Close to St. Stephen's Basilica, Szent István Square, Budapest (Figure 3)
The Google Map Locations are:
"https://www.google.com/maps?q=47.49836666666667,19.040433333333333"
"https://www.google.com/maps?q=47.49956666666667,19.04675"
These landmarks are prominent attractions in Budapest, situated near the Danube River.
2. Decode output1.nmea . You can use decoders. What is the path defined by the GPS coordinates? You can submit two map images also. **Answer:**

Python Code

```
import folium
from haversine import haversine, Unit
file_path = '/content/output1.nmea'
# Function to decode NMEA file and extract paths (using $GPGGA and $GPRMC
    ↪ sentences)
def decode_nmea_file(file_path):
```

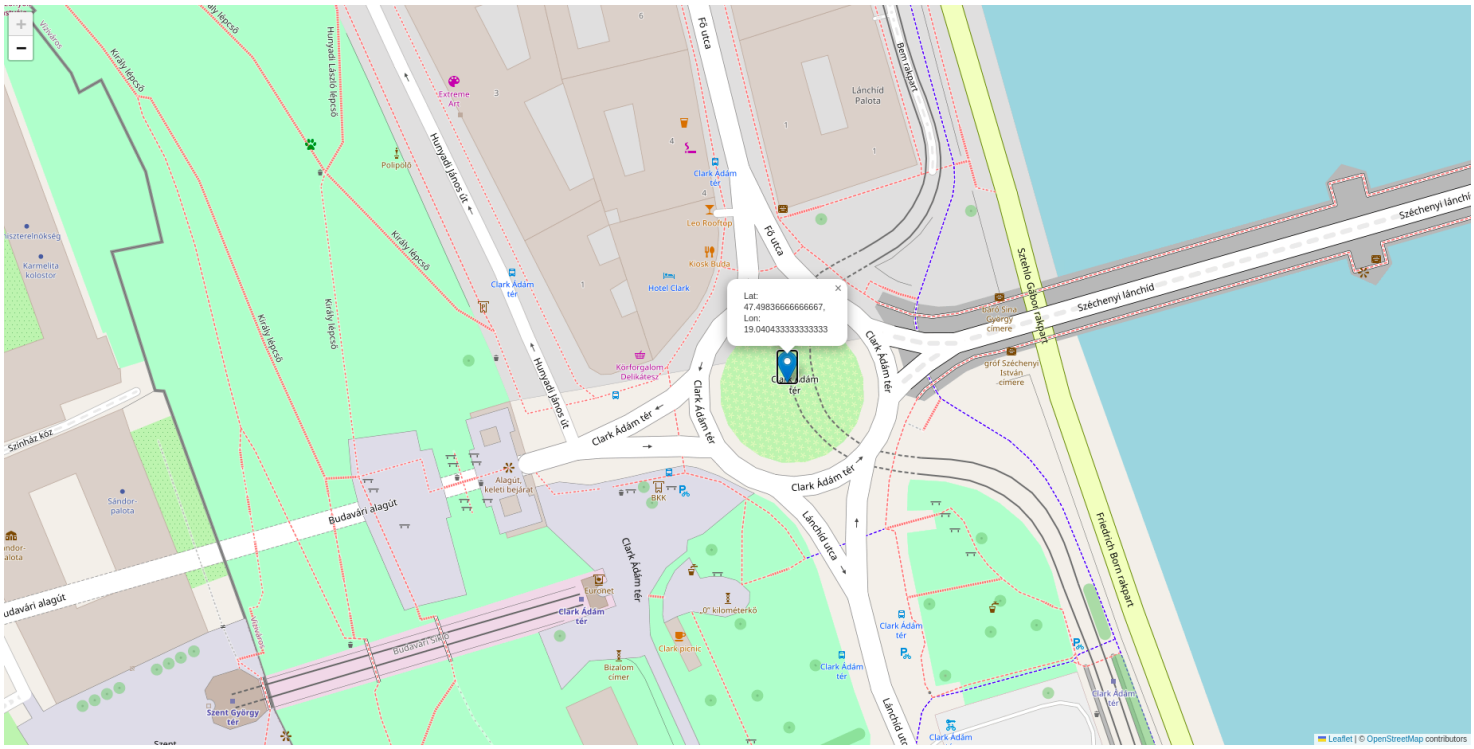

Figure 2: Latitude: 47.49837, Longitude: 19.04043

```
    coordinates = []
    with open(file_path, 'r') as file:
        for line in file:
            if "$GPGGA" in line or "$GPRMC" in line:
                parts = line.split(",")
                if len(parts) > 6:
                    # Latitude
                    raw_lat = parts[2]
                    lat_dir = parts[3]
                    # Longitude
                    raw_lon = parts[4]
                    lon_dir = parts[5]

                    # Ensure valid numeric fields
                    if raw_lat.replace('.', '', 1).isdigit() and raw_lon.replace(
    ↪ '.', '', 1).isdigit():
                        # Convert to decimal degrees
                        lat = float(raw_lat[:2]) + float(raw_lat[2:]) / 60.0
                        lon = float(raw_lon[:3]) + float(raw_lon[3:]) / 60.0
                        if lat_dir == "S":
                            lat = -lat
                        if lon_dir == "W":
                            lon = -lon
                        coordinates.append((lat, lon))
    return coordinates

```

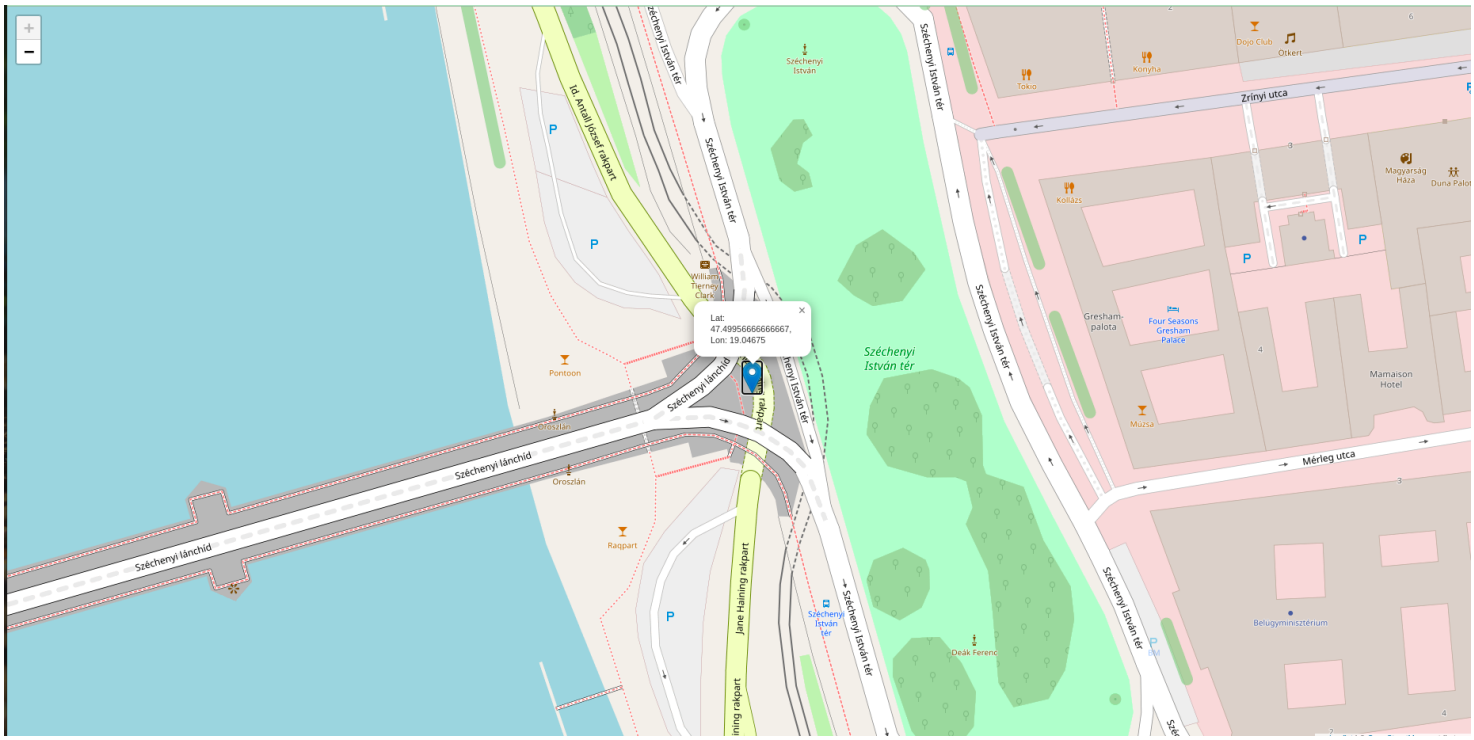

Figure 3: Latitude: 47.49957, Longitude: 19.04675

```
# Decode the new file and extract GPS path
path_coordinates = decode_nmea_file(file_path)

# Display the extracted path
path_coordinates[:10]

# Create a map centered around the first coordinate of the path
if path_coordinates:
    path_map = folium.Map(location=path_coordinates[0], zoom_start=17)

    # Add the coordinates to the map as a line
    folium.PolyLine(path_coordinates, color="blue", weight=2.5, opacity=1).add_to
    ↪ (path_map)

    # Add markers for the start and end points
    folium.Marker(location=path_coordinates[0], popup="Start Point", icon=folium.
    ↪ Icon(color="green")).add_to(path_map)
    folium.Marker(location=path_coordinates[-1], popup="End Point", icon=folium.
    ↪ Icon(color="red")).add_to(path_map)

    # Save and display the map
    path_map_path = 'gps_path_map.html'
    path_map.save(path_map_path)
    path_map_path
else:
    "No valid GPS path found."
```

```
54
55 # Function to calculate the total path length
56 def calculate_path_length(coordinates):
57     total_distance = 0
58     for i in range(len(coordinates) - 1):
59         point1 = coordinates[i]
60         point2 = coordinates[i + 1]
61         # Use haversine with tuples
62         total_distance += haversine(point1, point2, unit=Unit.KILOMETERS)
63     return total_distance
64
65 # Calculate total path length
66 total_path_length = calculate_path_length(path_coordinates)
67 total_path_length_km = total_path_length  # Already in kilometers
68 total_path_length_m = total_path_length * 1000  # Convert to meters
69 (total_path_length_km, total_path_length_m)
```

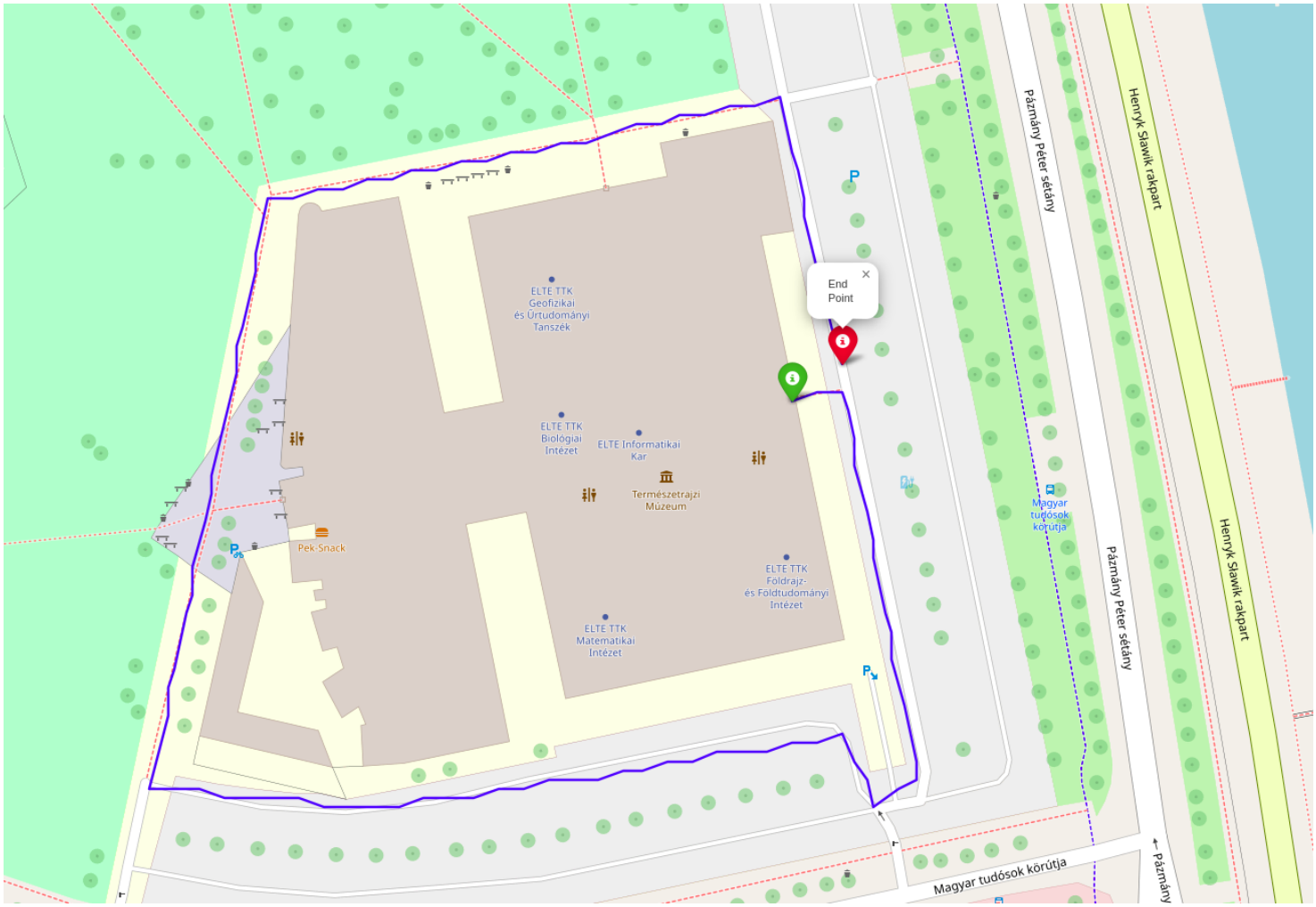

Figure 4: The path defined by the GPS coordinates

The green one is the Start point and Red one is the End point.
The path defined by the GPS coordinates represents a series of geographical points that likely trace a movement or

Figure 5: Total path length (kilometer and meter)

trajectory.
The path starts at **Latitude: 47.4724, Longitude: 19.0631.**
The path ends at **Latitude: 47.4721, Longitude: 19.0633.**
The path is short (Total Path Length is **542.75 meters** (approximately 0.54 kilometers from Figure 5)), it may represent a localized movement or trajectory, such as a pedestrian, vehicle, or robot path within a small area.