**README: Coordinate Correction and Terrain Classification Scripts**

This README provides an overview of the fix\_coordinates.py and terrain.py scripts, detailing the purpose, setup, and execution instructions for each.

**Overview**

1. **fix\_coordinates.py**: This script processes a set of GPS coordinates, detects out-of-sequence points, corrects the path using a moving average, stores data in an SQLite database, and visualizes the corrected path.
2. **terrain.py**: This script interacts with the SQLite database to filter specific terrain classifications, and retrieves relevant corrected GPS points.

**Requirements**

Both scripts require the following libraries:

* **Python Libraries**:
  + pandas
  + numpy
  + sqlite3
  + matplotlib
  + os
* **Data Files**:
  + latitude\_longitude\_details.csv: Contains GPS coordinates with columns for latitude and longitude.
  + terrain\_classification.csv: Contains terrain types and distances.
* **SQLite Database**: An SQLite database (geodata.db) will be created to store processed and filtered data.

**fix\_coordinates.py**

This script identifies out-of-sequence GPS coordinates, applies a moving average to correct them, saves corrected data to an SQLite database, and visualizes the results.

**1. Script Details**

* **Input Files**:
  + latitude\_longitude\_details.csv
  + terrain\_classification.csv
* **Database**:
  + geodata.db

**2. Script Functionality**

1. **Setup Directories**:
   * Creates a data directory for input files and an output directory for storing output files.
2. **Load Data**:
   * Loads latitude\_longitude\_details.csv and terrain\_classification.csv into pandas DataFrames.
3. **Identify Out-of-Sequence Points**:
   * Calculates the distance between consecutive points to detect outliers.
4. **Correct Path with Moving Average**:
   * Applies a moving average on latitude and longitude to correct the path, ignoring NaN values at the edges.
   * Saves corrected coordinates to latitude\_longitude\_corrected.csv.
5. **Store Data in SQLite**:
   * Creates three tables in geodata.db:
     + latitude\_longitude\_details: Stores original coordinates and distances.
     + terrain\_classification: Stores terrain data.
     + latitude\_longitude\_corrected: Stores corrected coordinates.
6. **Visualize Before and After**:
   * Plots the original and corrected paths for comparison.

**3. Execution**

Run the script with:

python scripts/fix\_coordinates.py

**4. Output**

* **CSV File**: latitude\_longitude\_corrected.csv containing corrected GPS coordinates.
* **Database**: geodata.db with three tables.
* **Visualization**: Shows plots comparing the original and corrected paths.

**terrain.py**

This script queries the database for specific terrain types and filtered GPS points based on the presence of "road" and exclusion of "civil station."

**1. Script Details**

* **Database File**:
  + geodata.db

**2. Script Functionality**

1. **Filter Terrain Types:**
   * **Queries the terrain\_classification table for rows containing "road" but excluding "civil station."**
2. **Retrieve Corrected GPS Points:**
   * **Fetches GPS points from the latitude\_longitude\_corrected table related to the filtered terrain types.**

**3. Execution**

**Run the script with:**

python scripts/terrain.py

**4. Output**

* **Console Output**:
  + Latitude and longitude points that meet the filtering criteria (coordinates of roads without civil station)

PROVIDING SCREENSHOTS OF THE OUTPUTS IN THE RESULTS FOLDER.