Project Overview

This project focuses on processing satellite data, specifically converting TLE data into satellite coordinates, filtering the data based on a user-defined polygon, and presenting the results to the user. The project uses a variety of libraries and modules to achieve its goals, including **sgp4** for satellite position calculations, **geopandas** for geometric operations, and **multiprocessing** for parallelization.

How the Project Works

1. **Coordinate Conversion**: The project starts by reading TLE (Two-Line Elements) data, which includes information about the orbits of multiple satellites. The sgp4 library is used to convert TLE data into satellite objects, representing the positions of these satellites at different times.
2. **Time Calculation**: The project calculates Julian Dates (JD) and Fractional Days (FR) for each minute of a specified day. These time values are crucial for extracting satellite positions accurately. They are calculated once and used with SatrecArray to make the code faster
3. **Position Extraction**: Using the sgp4 library, the project retrieves the positional data (ECEF coordinates) of the satellites at each minute throughout the day. These coordinates represent the location of each satellite in the Earth-Centered, Earth-Fixed (ECEF) reference frame. More specifically it uses SatrecArray operations which avoids multiple calls and gets all the data at once
4. **Geometric Operations**: The project involves geometric operations to determine whether these satellite positions fall within a user-defined polygon. It checks if each point (satellite position) is inside the polygon. The geopandas library efficiently handles this task, as it is built to be used alongside numpy vectorization.
5. **Efficient Filtering**: The project uses numpy vector operations to filter out data efficiently. It removes rows with NaN (Not-a-Number) values and performing geometric operations with geopandas. The use of numpy arrays and vectorized operations enhances performance by avoiding explicit loops.
6. **Output Presentation**: After processing and filtering the data, the project calculates and displays the time taken for the entire process. It also presents the filtered data to the user, showing the number of data points and their coordinates if any are found within the specified polygon.

Multiprocessing and Numpy Optimization

with 8 cpu-workers the average time it takes to process 30000 satellites TLE data is 30 Seconds, and the main reason for this is Multiprocessing and Numpy Vector Operations

- **Multiprocessing**: Multiprocessing is employed to parallelize the processing of satellite data. By dividing the data into smaller chunks and processing them simultaneously on multiple CPU cores, the project significantly reduces the time required for computation. This is especially beneficial when dealing with a large number of satellites or a substantial amount of time intervals.

- **Numpy Vector Operations**: Numpy, a powerful library for numerical operations in Python, is utilized for efficient data manipulation. Vectorized operations provided by numpy allow the code to process large arrays of data without explicit loops, resulting in improved performance. This is crucial when filtering out data and checking whether satellite positions are inside the polygon, as it minimizes computational overhead.

In summary, this project efficiently processes satellite data by leveraging multiprocessing for parallelization and numpy for optimized vector operations. These strategies enable the project to handle large datasets and complex geometric calculations while providing fast and accurate results to the user.