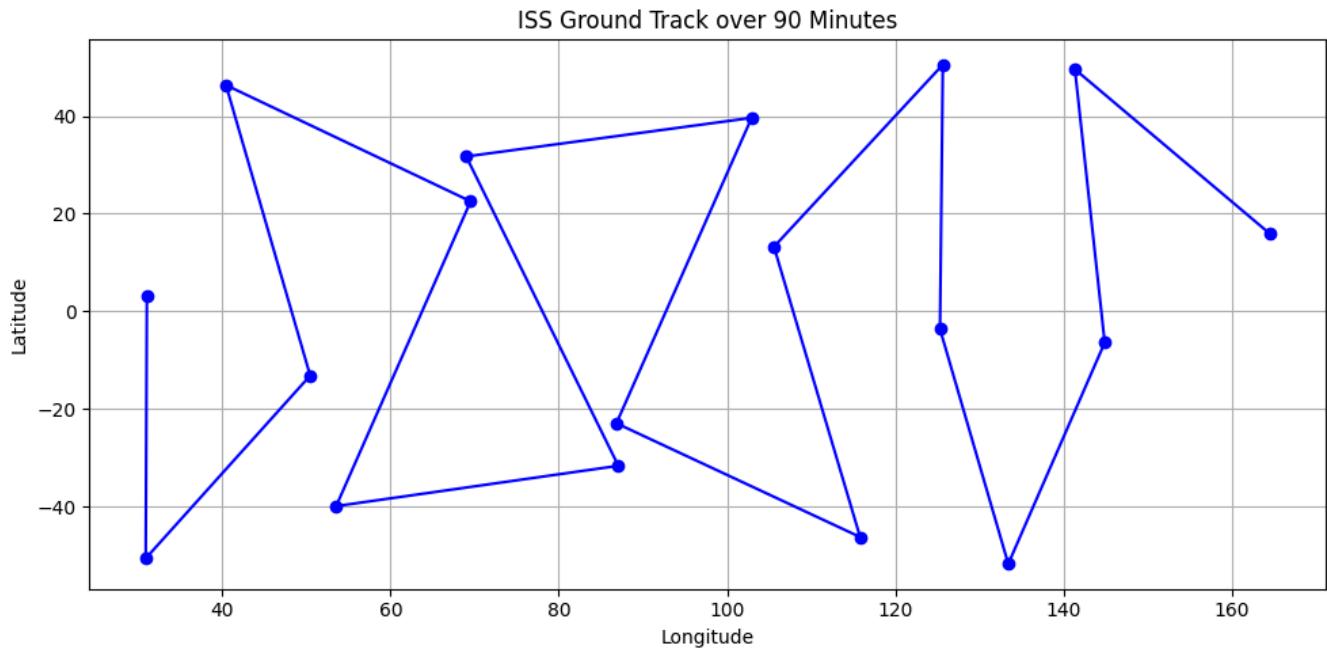


# SAT work

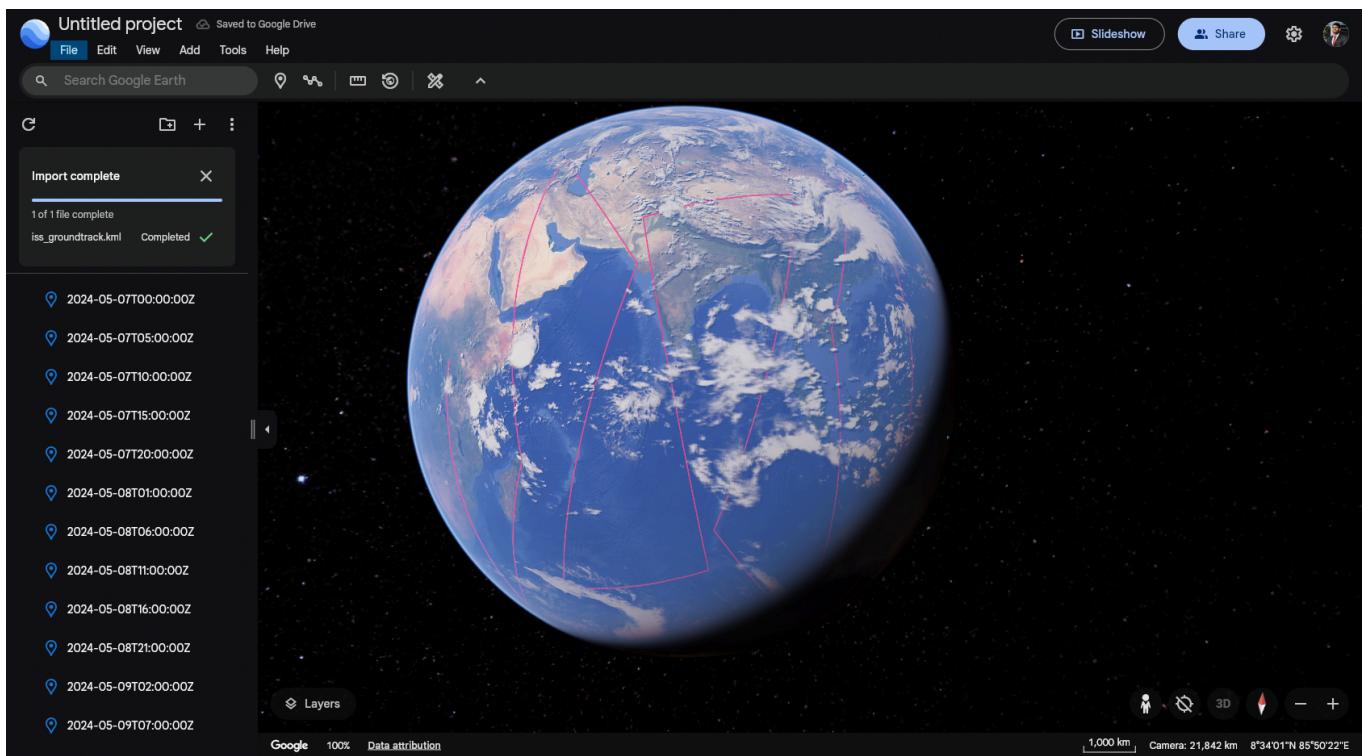
---



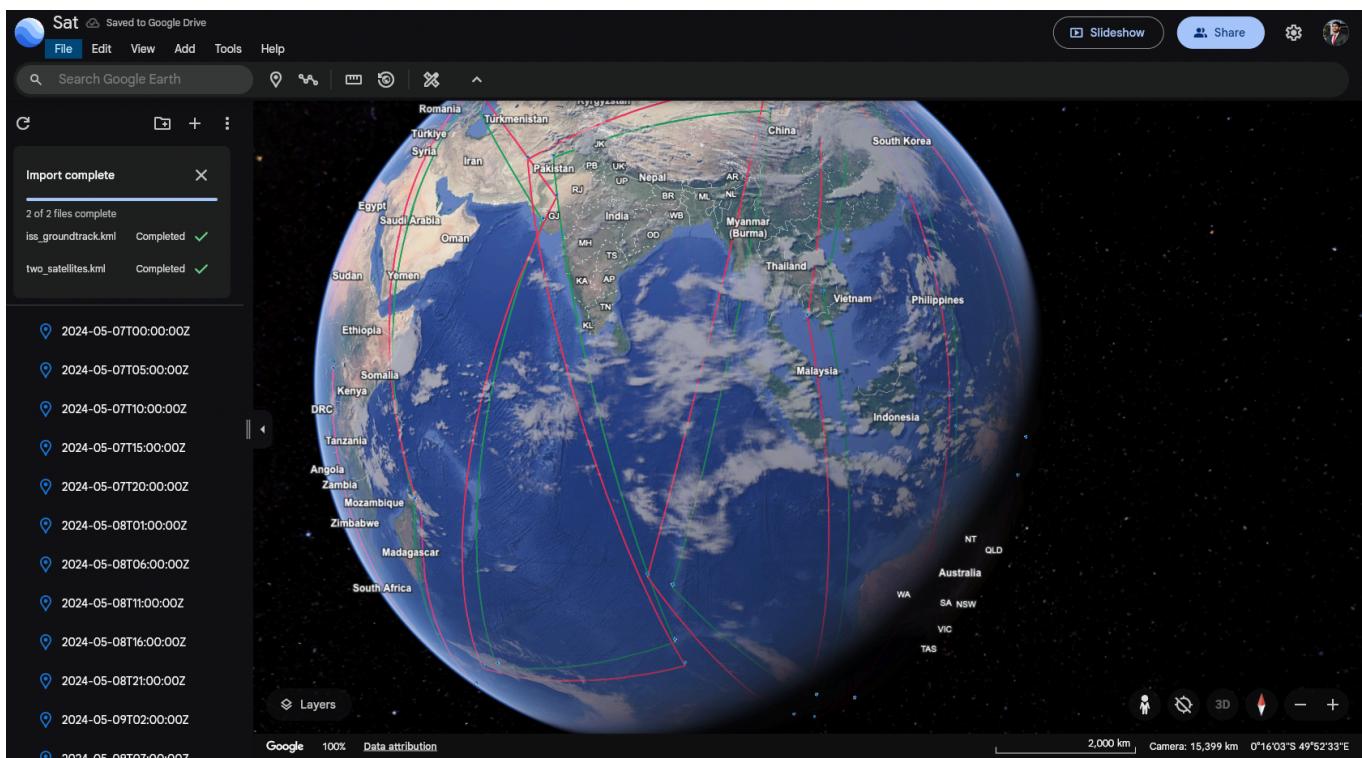
Satellite path in 2-D

**Exported it to KML -**

**Path of SAT 1**



## Path of both simulated SATs -



Adding a second satellite with a slightly offset orbit (e.g., 5° inclination difference) and plots both on the same map.

Also came up with further codes to calculate proximity (wasn't sure of the need) -

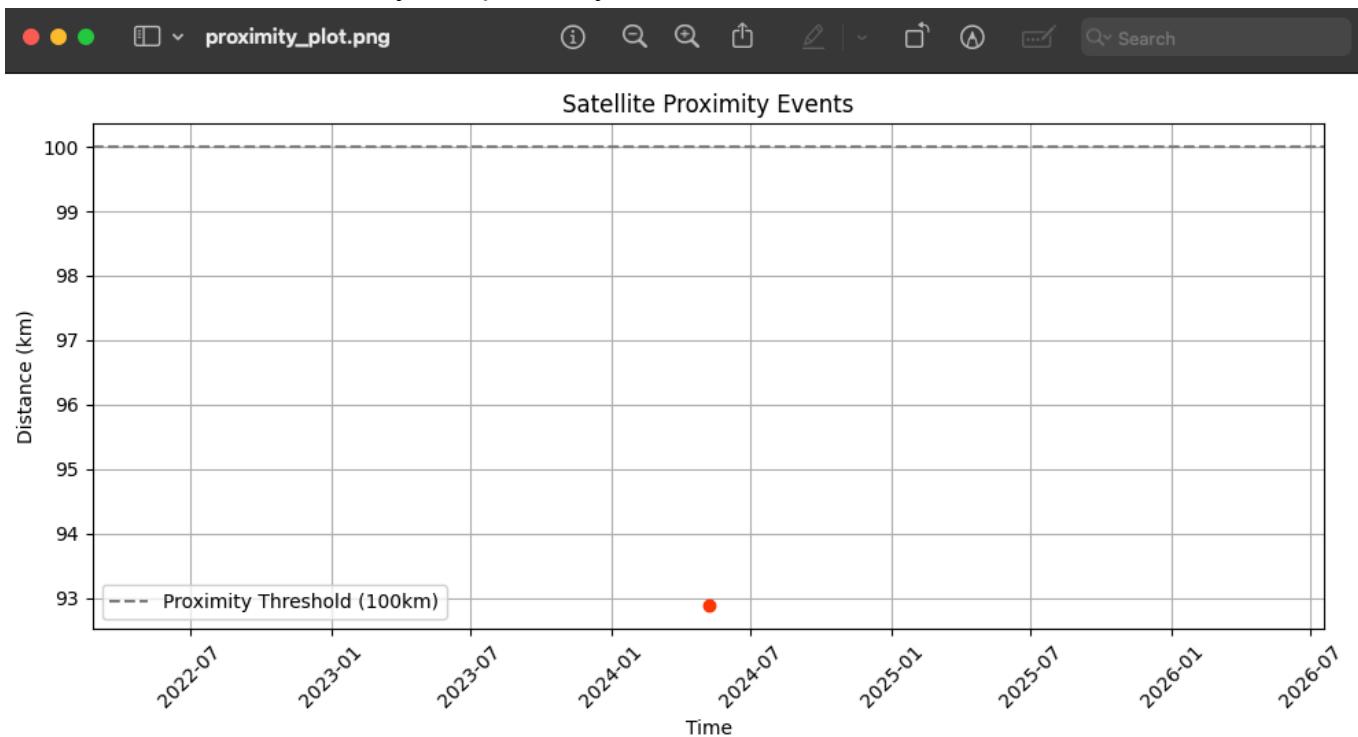
The screenshot shows a code editor interface with multiple tabs open. The active tab is 'proximity.py' (version 2). The code implements a proximity detection system between two EarthSatellite objects. It includes functions for loading satellites, calculating distances, and logging proximity events. The code editor also displays a terminal window showing log messages related to the execution of the script.

```
satellite_track.py 2 kml.py 2 attacker.py 2 proximity.py 2 x proximity_plot.py 1

# proximity.py ...
1 from skyfield.api import EarthSatellite, load
2 from geopy.distance import geodesic
3
4 # Load satellites and times
5 ts = load.timescale()
6
7 line1_v = "1 25544 98067A 24127.54791667 .00001764 00000+0 43262-4 0 9996"
8 line2_v = "2 25544 51.6448 81.1126 0004062 132.3403 38.2868 15.50620765393018"
9 victim = EarthSatellite(line1_v, line2_v, "Victim")
10
11 line1_a = "1 99999U 24081A 24127.54791667 .00001764 00000+0 50000-4 0 9990"
12 line2_a = "2 99999 56.6448 81.1126 0004062 132.3403 38.2868 15.50620765393018"
13 attacker = EarthSatellite(line1_a, line2_a, "Attacker")
14
15 times = ts.utc(2024, 5, 7, range(0, 96, 5))
16
17 log_file = open("proximity_log.txt", "w")
18 log_file.write("Time,Distance_km\n")
19
20 # Detect and log proximity
21 for t in times:
22     sp_v = victim.at(t).subpoint()
23     sp_a = attacker.at(t).subpoint()
24     coord_v = (sp_v.latitude.degrees, sp_v.longitude.degrees)
25     coord_a = (sp_a.latitude.degrees, sp_a.longitude.degrees)
26     distance_km = geodesic(coord_v, coord_a).km
27     if distance_km < 100:
28         log_file.write(f"{t}, {distance_km}\n")
29
30 log_file.close()

(satenv) kamaleh@Kamalehs-MacBook-Pro RA % python3 replay.py
Sent command: ['command': 'GET_POWER_LOW', 'timestamp': 1746734424, 'token': '990eb5ade0fbdcb6cf664135f7b79c5f17f922641b4572c47fd6bef46d81e596']
Reply validation result: True | Reason: Valid
(satenv) kamaleh@Kamalehs-MacBook-Pro RA % python3 proximity.py
[LOGGED] 2024-05-07T00:00:02Z,92.88
(satenv) kamaleh@Kamalehs-MacBook-Pro RA % python3 proximity.py
2025-05-08 13:03:54.375 Python[91538:23177391] +[IMKClient subclass] chose IMKClient_Modern
2025-05-08 13:03:54.375 Python[91538:23177391] +[IMKInputSession subclass] chose IMKInputSession_Modern
(satenv) kamaleh@Kamalehs-MacBook-Pro RA % python3 proximity_plot.py
2025-05-08 13:04:19.760 Python[91554:23178057] +[IMKClient subclass] chose IMKClient_Modern
2025-05-08 13:04:19.760 Python[91554:23178057] +[IMKInputSession subclass] chose IMKInputSession_Modern
(satenv) kamaleh@Kamalehs-MacBook-Pro RA % [
```

Since distance <100km.Only one proximity event had come.

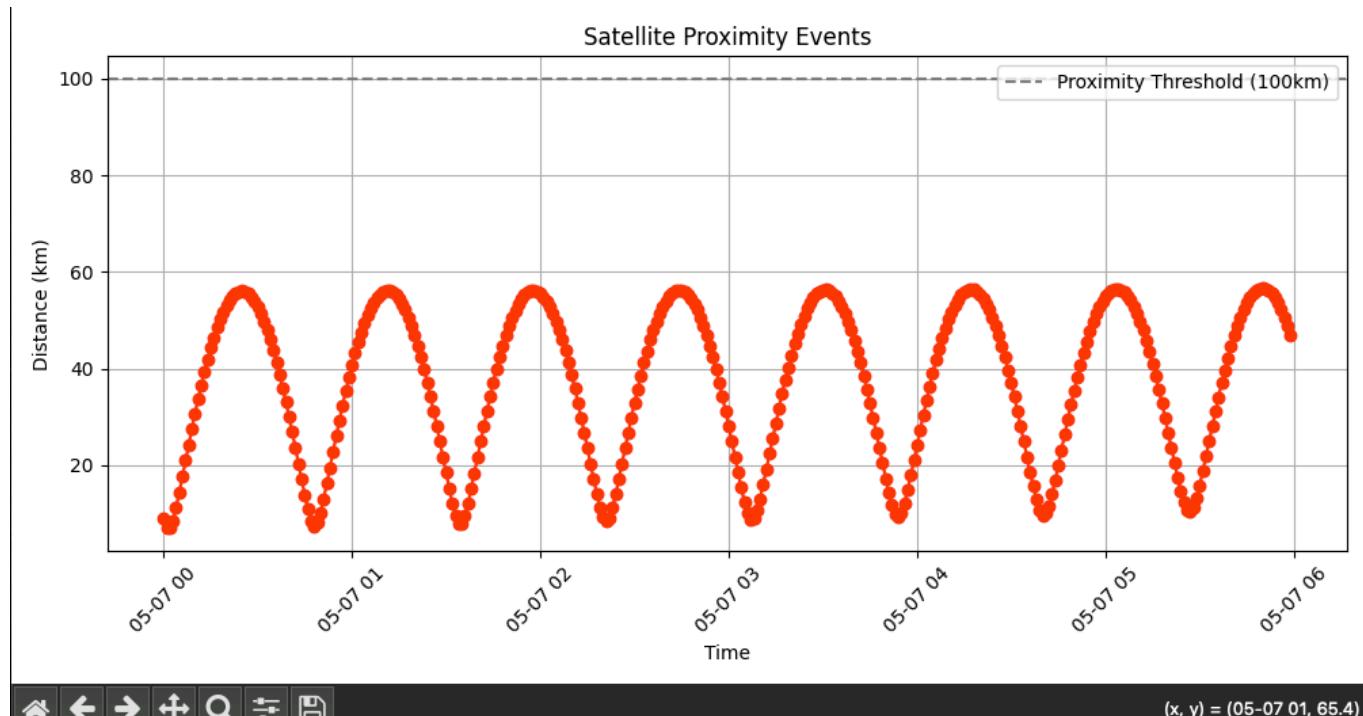


## Adjusted code with -

- **1-minute resolution** captures more moments of closeness
  - **6-hour window** increases chances of orbital crossing

- <300 km threshold captures more near misses that are still relevant in satellite comms range
- Slightly adjusted attacker orbit ensures it's not too far from the victim

and got this -



**PS : Not sure if this is concept proximity is necessary for us but just explored on it a little.**