Python in Astronomy: Tracking the International Space Station (ISS)

Presenter: Mr. Kritish Nagyal

Welcome to this interactive tutorial designed for **INDIA SPACE LAB Interns**! In this Jupyter Notebook, we'll learn how to use Python to track the International Space Station (ISS) from any city on Earth.

We'll explore key astronomy and programming concepts, including:

- **Geospatial data**: Converting city names to latitude and longitude.
- **Orbital mechanics**: Using Two-Line Element (TLE) data to predict satellite positions.
- **Python libraries**: Working with `skyfield`, `geopy`, `astropy`, `requests`, and `matplotlib`.
- **Visualization**: Plotting the ISS's path in the sky.

By the end, we'll have a working ISS tracker and understand how Python can be applied to real-world astronomy problems.

Prerequisites

Environment Setup

- Install Python 3.8 or above on your system.
- Ensure you have an IDE or code editor (e.g., VS Code, PyCharm).
- Confirm internet access for fetching live data.

Required Libraries

- `requests` (for HTTP requests)
- `skyfield` (for orbital calculations)
- `numpy` (for numerical operations)
- `matplotlib` (for plotting)
- `ipywidgets` (for interactive controls)

Task 1: Installing and Importing Libraries

Objective: Prepared a workspace by installing and importing all necessary packages.

In [4]:
```
!pip install requests skyfield numpy matplotlib ipywidgets pandas
```

```
Requirement already satisfied: requests in /opt/anaconda3/lib/python3.11/site-packages (2.31.0)
Requirement already satisfied: skyfield in /opt/anaconda3/lib/python3.11/site-packages (1.53)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.11/site-packages (1.26.4)
Requirement already satisfied: matplotlib in /opt/anaconda3/lib/python3.11/site-packages (3.8.0)
Requirement already satisfied: ipywidgets in /opt/anaconda3/lib/python3.11/site-packages (8.1.7)
Requirement already satisfied: pandas in /opt/anaconda3/lib/python3.11/site-packages (2.1.4)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/anaconda3/lib/python3.11/site-packages (from req
uests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/anaconda3/lib/python3.11/site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/anaconda3/lib/python3.11/site-packages (from requests)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /opt/anaconda3/lib/python3.11/site-packages (from requests)
(2025.7.14)
Requirement already satisfied: jplephem>=2.13 in /opt/anaconda3/lib/python3.11/site-packages (from skyfield) (2.
23)
Requirement already satisfied: sgp4>=2.13 in /opt/anaconda3/lib/python3.11/site-packages (from skyfield) (2.24)
Requirement already satisfied: contourpy>=1.0.1 in /opt/anaconda3/lib/python3.11/site-packages (from matplotlib)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in /opt/anaconda3/lib/python3.11/site-packages (from matplotlib) (0.
11.0)
Requirement already satisfied: fonttools>=4.22.0 in /opt/anaconda3/lib/python3.11/site-packages (from matplotlib
) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/anaconda3/lib/python3.11/site-packages (from matplotlib
) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /opt/anaconda3/lib/python3.11/site-packages (from matplotlib)
(23.1)
Requirement already satisfied: pillow>=6.2.0 in /opt/anaconda3/lib/python3.11/site-packages (from matplotlib) (1
0.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/anaconda3/lib/python3.11/site-packages (from matplotlib)
(3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in /opt/anaconda3/lib/python3.11/site-packages (from matplot
lib) (2.8.2)
Requirement already satisfied: comm>=0.1.3 in /opt/anaconda3/lib/python3.11/site-packages (from ipywidgets) (0.2
.2)
Requirement already satisfied: ipython>=6.1.0 in /opt/anaconda3/lib/python3.11/site-packages (from ipywidgets) (
8.20.0)
Requirement already satisfied: traitlets>=4.3.1 in /opt/anaconda3/lib/python3.11/site-packages (from ipywidgets)
(5.7.1)
Requirement already satisfied: widgetsnbextension~=4.0.14 in /opt/anaconda3/lib/python3.11/site-packages (from i
pywidgets) (4.0.14)
Requirement already satisfied: jupyterlab_widgets~=3.0.15 in /opt/anaconda3/lib/python3.11/site-packages (from i
pywidgets) (3.0.15)
Requirement already satisfied: pytz>=2020.1 in /opt/anaconda3/lib/python3.11/site-packages (from pandas) (2023.3
.post1)
Requirement already satisfied: tzdata>=2022.1 in /opt/anaconda3/lib/python3.11/site-packages (from pandas) (2023
.3)
Requirement already satisfied: decorator in /opt/anaconda3/lib/python3.11/site-packages (from ipython>=6.1.0->ip
ywidgets) (5.1.1)
Requirement already satisfied: jedi>=0.16 in /opt/anaconda3/lib/python3.11/site-packages (from ipython>=6.1.0->i
pywidgets) (0.18.1)
Requirement already satisfied: matplotlib-inline in /opt/anaconda3/lib/python3.11/site-packages (from ipython>=6
.1.0->ipywidgets) (0.1.6)
Requirement already satisfied: prompt-toolkit<3.1.0,>=3.0.41 in /opt/anaconda3/lib/python3.11/site-packages (fro
m ipython>=6.1.0->ipywidgets) (3.0.43)
Requirement already satisfied: pygments>=2.4.0 in /opt/anaconda3/lib/python3.11/site-packages (from ipython>=6.1
.0->ipywidgets) (2.15.1)
Requirement already satisfied: stack-data in /opt/anaconda3/lib/python3.11/site-packages (from ipython>=6.1.0->i
pywidgets) (0.2.0)
Requirement already satisfied: pexpect>4.3 in /opt/anaconda3/lib/python3.11/site-packages (from ipython>=6.1.0->
ipywidgets) (4.8.0)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.11/site-packages (from python-dateutil>=2.
7->matplotlib) (1.16.0)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /opt/anaconda3/lib/python3.11/site-packages (from jedi>=0.
16->ipython>=6.1.0->ipywidgets) (0.8.3)
Requirement already satisfied: ptyprocess>=0.5 in /opt/anaconda3/lib/python3.11/site-packages (from pexpect>4.3-
>ipython>=6.1.0->ipywidgets) (0.7.0)
Requirement already satisfied: wcwidth in /opt/anaconda3/lib/python3.11/site-packages (from prompt-toolkit<3.1.0
,>=3.0.41->ipython>=6.1.0->ipywidgets) (0.2.5)
Requirement already satisfied: executing in /opt/anaconda3/lib/python3.11/site-packages (from stack-data->ipytho
n>=6.1.0->ipywidgets) (0.8.3)
Requirement already satisfied: asttokens in /opt/anaconda3/lib/python3.11/site-packages (from stack-data->ipytho
n>=6.1.0->ipywidgets) (2.0.5)
Requirement already satisfied: pure-eval in /opt/anaconda3/lib/python3.11/site-packages (from stack-data->ipytho
n>=6.1.0->ipywidgets) (0.2.2)
```

In [3]:
```
conda install -c conda-forge cartopy
```

```
Channels:
 - conda-forge
 - defaults
Platform: osx-64
Collecting package metadata (repodata.json): done
Solving environment: done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
rm -rf /opt/anaconda3/pkgs/proj-9.3.1-h1972728_0
```

In [7]:
```
conda update -n base -c defaults conda
```

```
Channels:
 - defaults
 - conda-forge
Platform: osx-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda3

  added / updated specs:
    - conda


The following packages will be SUPERSEDED by a higher-priority channel:

  certifi            conda-forge/noarch::certifi-2025.7.14~ --> pkgs/main/osx-64::certifi-2025.7.14-py311hecd8cb
5_0



Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [9]:
```
conda install -c conda-forge cartopy --repodata-fn=repodata.json
```

```
Channels:
 - conda-forge
 - defaults
Platform: osx-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /opt/anaconda3

  added / updated specs:
    - cartopy


The following packages will be SUPERSEDED by a higher-priority channel:

  certifi            pkgs/main/osx-64::certifi-2025.7.14-p~ --> conda-forge/noarch::certifi-2025.7.14-pyhd8ed1ab
_0



Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done

Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
import cartopy.crs as ccrs
import cartopy.feature as cfeature
print("Cartopy imported successfully!")
```

```
Cartopy imported successfully!
```

In [7]:

```python
#required libraries
import requests
from skyfield.api import load, wgs84, EarthSatellite
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime, timedelta
import pandas as pd
import json, csv
from ipywidgets import Dropdown, FloatSlider, Button, VBox, HBox, Output
import cartopy.crs as ccrs
import cartopy.feature as cfeature
```

Task 2: Fetching ISS Two-Line Element (TLE) Data

Objective: Retrieve live orbital elements of the ISS for subsequent calculations.

In [9]:

```python
#collecting live TLE data from CelesTrak
TLE_URL = "https://celestrak.org/NORAD/elements/stations.txt"
def fetch_iss_tle():
    try:
        response = requests.get(TLE_URL)
        response.raise_for_status()
        lines = response.text.strip().split('\n')
        for i, line in enumerate(lines):
            if "ISS" in line:
                tle1 = lines[i+1].strip()
                tle2 = lines[i+2].strip()
                return tle1, tle2
    except Exception as e:
        print("Error fetching TLE:", e)
        return None, None
tle_line1, tle_line2 = fetch_iss_tle()
print("TLE Line1:", tle_line1)
print("TLE Line2:", tle_line2)
```

```
TLE Line1: 1 25544U 98067A   25204.25047604  .00008934  00000+0  16419-3 0  9998
TLE Line2: 2 25544  51.6351 128.3174 0002345 109.2368 347.8357 15.50024026520753
```

Task 3: Calculating ISS Pass Times for a Given Location

Objective: Determine when the ISS will be visible from a user-specified site.

Obtain Coordinates

- Prompt the user for latitude and longitude (decimal degrees).
- Validate inputs lie within –90° to +90° (latitude) and –180° to +180° (longitude).

In [11]:

```python
#for coordinates
lat = float(input("Enter latitude (-90 to 90): "))
lon = float(input("Enter longitude (-180 to 180): "))

if not (-90 <= lat <= 90 and -180 <= lon <= 180):
    raise ValueError("Invalid latitude or longitude.")

#load timescale
ts = load.timescale()
t0 = ts.now()

#load satellite
satellite = EarthSatellite(tle_line1, tle_line2, "ISS", ts)
location = wgs84.latlon(lat, lon)

#find events (next pass)
from skyfield.api import load, wgs84, EarthSatellite, utc
t1 = t0
t2 = ts.utc((datetime.utcnow().replace(tzinfo=utc)) + timedelta(days=1))
t, events = satellite.find_events(location, t0, t2, altitude_degrees=10.0)

event_names = ["Rise above 10°", "Highest point", "Set below 10°"]
print("\nNext visible passes from this location:")
for ti, event in zip(t, events):
    print(ti.utc_strftime('%Y-%m-%d %H:%M:%S UTC'), "->", event_names[event])
```

```
Next visible passes from this location:
2025-07-24 08:00:46 UTC -> Rise above 10°
2025-07-24 08:02:56 UTC -> Highest point
2025-07-24 08:05:06 UTC -> Set below 10°
2025-07-24 09:36:22 UTC -> Rise above 10°
2025-07-24 09:39:27 UTC -> Highest point
2025-07-24 09:42:32 UTC -> Set below 10°
2025-07-24 16:08:38 UTC -> Rise above 10°
2025-07-24 16:11:38 UTC -> Highest point
2025-07-24 16:14:38 UTC -> Set below 10°
2025-07-24 17:45:47 UTC -> Rise above 10°
2025-07-24 17:48:12 UTC -> Highest point
2025-07-24 17:50:36 UTC -> Set below 10°
```

Task 4: Visualizing the ISS Ground Track

Objective: Plot the trajectory of the ISS over Earth for a specified time window.

In [13]:

```python
from skyfield.api import utc
#define time range
minutes = 90
#get current UTC time
now = datetime.utcnow().replace(tzinfo=utc)
#build a list of datetime objects for each minute
time_list = [now + timedelta(minutes=m) for m in range(minutes)]
#convert to Skyfield Time object
times = ts.utc([dt.year for dt in time_list],
               [dt.month for dt in time_list],
               [dt.day for dt in time_list],
               [dt.hour for dt in time_list],
               [dt.minute for dt in time_list],
               [dt.second for dt in time_list])
#compute positions
latitudes, longitudes = [], []
for t in times:
    geocentric = satellite.at(t)
    subpoint = wgs84.subpoint(geocentric)
    latitudes.append(subpoint.latitude.degrees)
    longitudes.append(subpoint.longitude.degrees)
#plot with Cartopy
fig = plt.figure(figsize=(12,6))
ax = plt.axes(projection=ccrs.PlateCarree())
ax.add_feature(cfeature.LAND)
ax.add_feature(cfeature.OCEAN)
ax.add_feature(cfeature.COASTLINE)
ax.set_global()
ax.gridlines(draw_labels=True)
ax.plot(longitudes, latitudes, 'r-', transform=ccrs.Geodetic(), label='ISS path')
ax.scatter(longitudes[0], latitudes[0], color='blue', marker='o',
           transform=ccrs.PlateCarree(), label='Start')
ax.legend()
plt.title('ISS Ground Track (Next 90 minutes)')
plt.show()
```

Task 5: Predicting Passes Over Multiple Locations

Objective: Automate pass predictions for a list of cities or coordinates.

In [15]:
```
#creating csv
import csv
data = [
    ["City", "Latitude", "Longitude"],
    ["Jammu", 32.7185614, 74.8580917],
    ["London", 51.5074, -0.1278],
    ["New York", 40.7128, -74.0060],
    ["Tokyo", 35.6895, 139.6917],
    ["Sydney", -33.8688, 151.2093]
]
with open("locations.csv", "w", newline='') as f:
    writer = csv.writer(f)
    writer.writerows(data)
print("locations.csv created successfully!")
```

```
locations.csv created successfully!
```

In [17]:
```
import pandas as pd
from skyfield.api import wgs84
#loading our CSV
locations_df = pd.read_csv('locations.csv')
print("Columns in CSV:", locations_df.columns)
results = []
#loop over each row
for _, row in locations_df.iterrows():
    loc = wgs84.latlon(row['Latitude'], row['Longitude'])
    t, events = satellite.find_events(loc, t0, t2, altitude_degrees=10.0)
    for ti, event in zip(t, events):
        results.append([row['City'], ti.utc_datetime(), event_names[event]])

#create DataFrame and save
results_df = pd.DataFrame(results, columns=['Location', 'Time', 'Event'])
print(results_df.head())
results_df.to_csv('iss_passes.csv', index=False)
print("iss_passes.csv created successfully!")
```

```
Columns in CSV: Index(['City', 'Latitude', 'Longitude'], dtype='object')
  Location                             Time           Event
0    Jammu 2025-07-24 08:00:45.619374+00:00  Rise above 10°
1    Jammu 2025-07-24 08:02:55.844135+00:00   Highest point
2    Jammu 2025-07-24 08:05:06.499230+00:00    Set below 10°
3    Jammu 2025-07-24 09:36:21.664463+00:00  Rise above 10°
4    Jammu 2025-07-24 09:39:26.560503+00:00   Highest point
iss_passes.csv created successfully!
```


Task 6: Interactive Exploration

Objective: Enhance the notebook with interactive widgets for real-time control.

In [19]:
```
from ipywidgets import Dropdown, FloatSlider, Button, VBox, Output
from IPython.display import display
from datetime import datetime, timedelta
from skyfield.api import utc
out = Output()
def update_plot(change):
    with out:
        out.clear_output()
        print("Button clicked!")
        selected = location_dropdown.value
        lat, lon = locations_dict[selected]
        loc = wgs84.latlon(lat, lon)
        minutes_ahead = slider.value
        now = datetime.utcnow().replace(tzinfo=utc)
        time_list = [now + timedelta(minutes=m) for m in range(int(minutes_ahead))]
        t_range = ts.utc([dt.year for dt in time_list],
                         [dt.month for dt in time_list],
                         [dt.day for dt in time_list],
                         [dt.hour for dt in time_list],
                         [dt.minute for dt in time_list],
                         [dt.second for dt in time_list])
        lats, lons = [], []
        for t in t_range:
            sp = wgs84.subpoint(satellite.at(t))
            lats.append(sp.latitude.degrees)
            lons.append(sp.longitude.degrees)
        fig = plt.figure(figsize=(10,5))
        ax = plt.axes(projection=ccrs.PlateCarree())
```

```
        ax.add_feature(cfeature.LAND)
        ax.add_feature(cfeature.COASTLINE)
        ax.plot(lons, lats, 'r-')
        plt.show()
#prepare locations dictionary
locations_dict = {row['City']: (row['Latitude'], row['Longitude']) for _, row in locations_df.iterrows()}
location_dropdown = Dropdown(options=list(locations_dict.keys()), description='Location:')
slider = FloatSlider(value=60, min=10, max=180, step=10, description='Minutes:')
button = Button(description='Plot ISS Path')
button.on_click(update_plot)
display(VBox([location_dropdown, slider, button, out]))
```

```
VBox(children=(Dropdown(description='Location:', options=('Jammu', 'London', 'New York', 'Tokyo', 'Sydney'), v…
```

Task 7: Advanced Analysis – Orbital Drift over Time

Objective: Investigate long-term changes in ISS orbital elements.

Historical TLE Archive

- Download daily TLEs for one month into a local folder.
- Name files by date for easy ingestion.

In [21]:

```
import os
import requests
from datetime import datetime
#archive folder
os.makedirs('tle_archive', exist_ok=True)
#URL for ISS TLEs (Celestrak Stations file)
url = "https://celestrak.org/NORAD/elements/stations.txt"
#TLE data
response = requests.get(url)
if response.status_code == 200:
    tle_data = response.text.strip().splitlines()

    # Find ISS TLE (usually first 3 lines in stations.txt)
    # stations.txt often contains multiple satellites, but ISS (ZARYA) is usually first
    name = tle_data[0]
    line1 = tle_data[1]
    line2 = tle_data[2]

    # Create a dated filename
    today = datetime.utcnow().strftime("%Y%m%d")
    filename = f"tle_archive/{today}.txt"

    # Save to file
    with open(filename, "w") as f:
        f.write(f"{name}\n{line1}\n{line2}\n")

    print(f"Saved ISS TLE to {filename}")
else:
    print(f"Failed to fetch TLE data. Status code: {response.status_code}")
```

```
Saved ISS TLE to tle_archive/20250723.txt
```

In [23]:

```
import glob
files = sorted(glob.glob('tle_archive/*.txt'))
print("TLE files found:", files)
dates, inclinations, eccentricities, periods = [], [], [], []
for fpath in files:
    sat = parse_tle(fpath)
    epoch = sat.epoch.utc_datetime()
    dates.append(epoch)
    inclinations.append(sat.model.inclo * (180/np.pi))
    eccentricities.append(sat.model.ecco)
    periods.append(1440/(sat.model.no_kozai))
# Plot
plt.figure(figsize=(10,6))
plt.plot(dates, inclinations, marker='o', label='Inclination (deg)')
plt.plot(dates, eccentricities, marker='o', label='Eccentricity')
plt.plot(dates, periods, marker='o', label='Period (min)')
plt.legend()
plt.title('ISS Orbital Elements Over Time (Sample Data)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
TLE files found: ['tle_archive/20250720.txt', 'tle_archive/20250721.txt', 'tle_archive/20250723.txt']
```

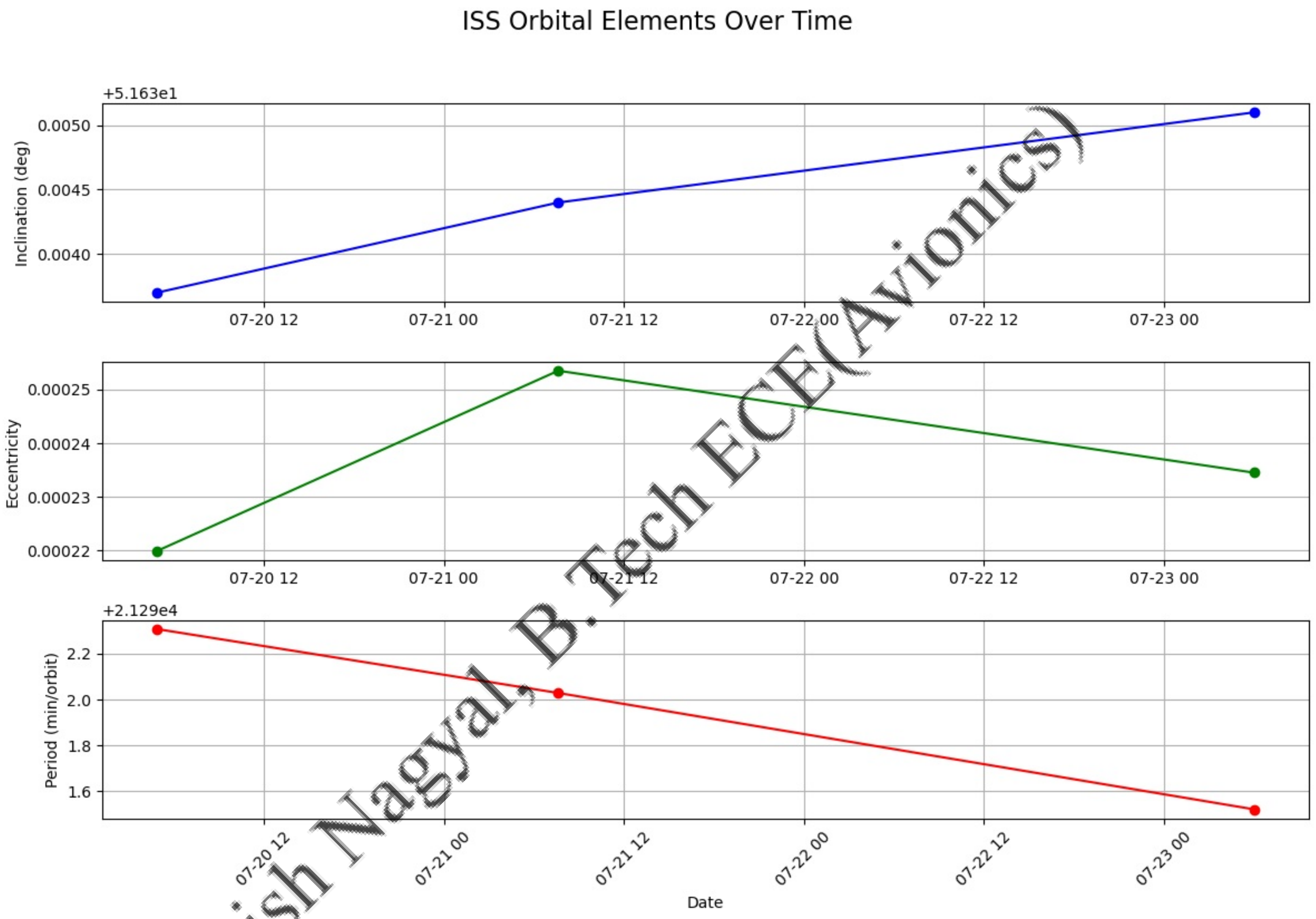
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[23], line 6
      4 dates, inclinations, eccentricities, periods = [], [], [], []
      5 for fpath in files:
----> 6     sat = parse_tle(fpath)
      7     epoch = sat.epoch.utc_datetime()
      8     dates.append(epoch)

NameError: name 'parse_tle' is not defined
```

In [25]:

```
import glob
import numpy as np
import matplotlib.pyplot as plt
from skyfield.api import load, EarthSatellite
#load timescale for Skyfield
ts = load.timescale()
#function to parse a TLE file and return an EarthSatellite
def parse_tle(file_path):
    with open(file_path) as f:
        lines = f.read().strip().split('\n')
        # Expect: 3 lines -> name, line1, line2
        if len(lines) < 3:
            raise ValueError(f"TLE file {file_path} does not have 3 lines")
        return EarthSatellite(lines[1], lines[2], lines[0], ts)
files = sorted(glob.glob('tle_archive/*.txt'))
print("TLE files found:", files)
dates, inclinations, eccentricities, periods = [], [], [], []
for fpath in files:
    sat = parse_tle(fpath)
    epoch = sat.epoch.utc_datetime()  # datetime object
    dates.append(epoch)
    inclinations.append(sat.model.inclo * (180/np.pi))   # radians → degrees
    eccentricities.append(sat.model.ecco)
    periods.append(1440.0 / sat.model.no_kozai)          # rev/day → minutes/orbit
#check we have data
if not dates:
    print("No TLE data found. Ensure tle_archive contains .txt files.")
else:
    print(f"Processed {len(dates)} TLE files.")
    plt.figure(figsize=(12, 8))
    # inclination
    plt.subplot(3, 1, 1)
    plt.plot(dates, inclinations, marker='o', color='b')
    plt.ylabel('Inclination (deg)')
    plt.grid(True)
    #eccentricity
    plt.subplot(3, 1, 2)
    plt.plot(dates, eccentricities, marker='o', color='g')
    plt.ylabel('Eccentricity')
    plt.grid(True)
    #period
    plt.subplot(3, 1, 3)
    plt.plot(dates, periods, marker='o', color='r')
    plt.ylabel('Period (min/orbit)')
    plt.xlabel('Date')
    plt.grid(True)
    plt.suptitle('ISS Orbital Elements Over Time', fontsize=16)
    plt.tight_layout(rect=[0,0,1,0.96])
    plt.xticks(rotation=45)
    plt.show()
```

```
TLE files found: ['tle_archive/20250720.txt', 'tle_archive/20250721.txt', 'tle_archive/20250723.txt']
Processed 3 TLE files.
```



Task 8: Building a Real-Time Dashboard Application

Objective: Package your functionality into a simple web app for live ISS tracking.

In [45]:
```
pip install streamlit requests skyfield folium streamlit-folium
```

```
Requirement already satisfied: streamlit in /opt/anaconda3/lib/python3.11/site-packages (1.47.0)
Requirement already satisfied: requests in /opt/anaconda3/lib/python3.11/site-packages (2.31.0)
Requirement already satisfied: skyfield in /opt/anaconda3/lib/python3.11/site-packages (1.53)
Requirement already satisfied: folium in /opt/anaconda3/lib/python3.11/site-packages (0.17.0)
Requirement already satisfied: streamlit-folium in /opt/anaconda3/lib/python3.11/site-packages (0.25.0)
Requirement already satisfied: altair<6,>=4.0 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit) (5
.0.1)
Requirement already satisfied: blinker<2,>=1.5.0 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit)
(1.6.2)
Requirement already satisfied: cachetools<7,>=4.0 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit
) (4.2.2)
Requirement already satisfied: click<9,>=7.0 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit) (8.
1.7)
Requirement already satisfied: numpy<3,>=1.23 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit) (1
.26.4)
Requirement already satisfied: packaging<26,>=20 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit)
(23.1)
Requirement already satisfied: pandas<3,>=1.4.0 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit)
(2.1.4)
Requirement already satisfied: pillow<12,>=7.1.0 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit)
(10.2.0)
Requirement already satisfied: protobuf<7,>=3.20 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit)
(3.20.3)
Requirement already satisfied: pyarrow>=7.0 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit) (14.
0.2)
Requirement already satisfied: tenacity<10,>=8.1.0 in /opt/anaconda3/lib/python3.11/site-packages (from streamli
t) (8.2.2)
Requirement already satisfied: toml<2,>=0.10.1 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit) (
0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /opt/anaconda3/lib/python3.11/site-packages (from
streamlit) (4.9.0)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /opt/anaconda3/lib/python3.11/site-packages (from
streamlit) (3.1.37)
Requirement already satisfied: pydeck<1,>=0.8.0b4 in /opt/anaconda3/lib/python3.11/site-packages (from streamlit
) (0.8.0)
Requirement already satisfied: tornado!=6.5.0,<7,>=6.0.3 in /opt/anaconda3/lib/python3.11/site-packages (from st
reamlit) (6.3.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/anaconda3/lib/python3.11/site-packages (from req
uests) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/anaconda3/lib/python3.11/site-packages (from requests) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/anaconda3/lib/python3.11/site-packages (from requests)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /opt/anaconda3/lib/python3.11/site-packages (from requests)
(2025.7.14)
Requirement already satisfied: jplephem>=2.13 in /opt/anaconda3/lib/python3.11/site-packages (from skyfield) (2.
23)
Requirement already satisfied: sgp4>=2.13 in /opt/anaconda3/lib/python3.11/site-packages (from skyfield) (2.24)
Requirement already satisfied: branca>=0.6.0 in /opt/anaconda3/lib/python3.11/site-packages (from folium) (0.7.2
)
Requirement already satisfied: jinja2>=2.9 in /opt/anaconda3/lib/python3.11/site-packages (from folium) (3.1.3)
Requirement already satisfied: xyzservices in /opt/anaconda3/lib/python3.11/site-packages (from folium) (2022.9.
0)
Requirement already satisfied: jsonschema>=3.0 in /opt/anaconda3/lib/python3.11/site-packages (from altair<6,>=4
.0->streamlit) (4.19.2)
Requirement already satisfied: toolz in /opt/anaconda3/lib/python3.11/site-packages (from altair<6,>=4.0->stream
lit) (0.12.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in /opt/anaconda3/lib/python3.11/site-packages (from gitpython!=3
.1.19,<4,>=3.0.7->streamlit) (4.0.7)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/anaconda3/lib/python3.11/site-packages (from jinja2>=2.9-
>folium) (2.1.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/anaconda3/lib/python3.11/site-packages (from panda
s<3,>=1.4.0->streamlit) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /opt/anaconda3/lib/python3.11/site-packages (from pandas<3,>=1.4.
0->streamlit) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in /opt/anaconda3/lib/python3.11/site-packages (from pandas<3,>=1.
4.0->streamlit) (2023.3)
Requirement already satisfied: smmap<5,>=3.0.1 in /opt/anaconda3/lib/python3.11/site-packages (from gitdb<5,>=4.
0.1->gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.0)
Requirement already satisfied: attrs>=22.2.0 in /opt/anaconda3/lib/python3.11/site-packages (from jsonschema>=3.
0->altair<6,>=4.0->streamlit) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /opt/anaconda3/lib/python3.11/site-packag
es (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in /opt/anaconda3/lib/python3.11/site-packages (from jsonsche
ma>=3.0->altair<6,>=4.0->streamlit) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in /opt/anaconda3/lib/python3.11/site-packages (from jsonschema>=3
.0->altair<6,>=4.0->streamlit) (0.10.6)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.11/site-packages (from python-dateutil>=2.
8.2->pandas<3,>=1.4.0->streamlit) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [27]:

```python
import streamlit as st
import requests
from datetime import datetime
```

```python
from skyfield.api import load, EarthSatellite, wgs84
import folium
from streamlit_folium import st_folium

# Load the Skyfield timescale
ts = load.timescale()

# Function to fetch current ISS TLE (no caching)
def fetch_iss_tle():
    url = "https://celestrak.org/NORAD/elements/stations.txt"
    response = requests.get(url)
    lines = response.text.strip().splitlines()
    # First 3 lines are usually ISS
    name = lines[0]
    line1 = lines[1]
    line2 = lines[2]
    sat = EarthSatellite(line1, line2, name, ts)
    return sat
# Function to get current position

def get_iss_position():
    satellite = fetch_iss_tle()
    t = ts.now()
    subpoint = wgs84.subpoint(satellite.at(t))
    lat = subpoint.latitude.degrees
    lon = subpoint.longitude.degrees
    alt = subpoint.elevation.km
    return lat, lon, alt

# Streamlit page config
st.set_page_config(page_title="ISS Live Tracker", layout="wide")
st.title("ISS Live Tracking Dashboard")
st.markdown("This dashboard shows the real-time position of the ISS on a world map.")

# Get position
lat, lon, alt = get_iss_position()

# Display current coordinates
st.metric(label="Latitude", value=f"{lat:.4f}°")
st.metric(label="Longitude", value=f"{lon:.4f}°")
st.metric(label="Altitude (km)", value=f"{alt:.2f} km")
# Create Folium map
m = folium.Map(location=[lat, lon], zoom_start=2, tiles="cartodbpositron")
folium.Marker(
    [lat, lon],
    popup="ISS Position",
    icon=folium.Icon(color="red", icon="info-sign")
).add_to(m)
# Render Folium map in Streamlit
st_folium(m, width=700, height=500)
# Refresh info
st.caption(f"Last updated: {datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S')} UTC")
st.info("Map auto-refreshes every time you refresh or rerun the app.")
```

```
2025-07-24 00:29:18.930 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:18.934 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:18.934 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:18.962
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run /opt/anaconda3/lib/python3.11/site-packages/ipykernel_launcher.py [ARGUMENTS]
2025-07-24 00:29:18.962 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:18.963 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:18.963 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:18.964 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:18.964 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.876 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.877 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.878 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.879 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.879 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.880 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.881 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.882 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.883 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.915 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.915 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.916 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.917 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.917 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.918 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.918 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.919 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.920 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.921 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
2025-07-24 00:29:20.921 Thread 'MainThread': missing ScriptRunContext! This warning can be ignored when running
in bare mode.
```

Out[27]: DeltaGenerator()

In [ ]:

```python
#saved this below code into a new file named in order to create the GUI Interface and track ISS
import streamlit as st
import requests
from datetime import datetime, timedelta
from skyfield.api import load, EarthSatellite, wgs84, utc
import folium
from streamlit_folium import st_folium
import numpy as np

ts = load.timescale()

def fetch_iss_tle():
    url = "https://celestrak.org/NORAD/elements/stations.txt"
    response = requests.get(url)
    lines = response.text.strip().splitlines()
    name = lines[0]
    line1 = lines[1]
    line2 = lines[2]
    sat = EarthSatellite(line1, line2, name, ts)
```

```python
    return sat

def get_iss_position():
    satellite = fetch_iss_tle()
    t_now = ts.now()
    pos1 = satellite.at(t_now).position.km

    t_future = ts.utc(datetime.utcnow().replace(tzinfo=utc) + timedelta(seconds=10))
    pos2 = satellite.at(t_future).position.km

    distance = np.linalg.norm(pos2 - pos1)
    speed_kms = distance / 10.0
    speed_kmh = speed_kms * 3600.0

    subpoint = wgs84.subpoint(satellite.at(t_now))
    lat = subpoint.latitude.degrees
    lon = subpoint.longitude.degrees
    alt = subpoint.elevation.km

    return lat, lon, alt, speed_kms, speed_kmh, satellite

def get_next_pass_over_jammu(satellite):
    jammu = wgs84.latlon(32.7186, 74.8581)  # Jammu coordinates
    t0 = ts.now()
    t1 = ts.utc(datetime.utcnow().replace(tzinfo=utc) + timedelta(days=1))
    t, events = satellite.find_events(jammu, t0, t1, altitude_degrees=10.0)
    event_names = ['Rise above 10°', 'Highest point', 'Set below 10°']
    return [(event_names[e], ti.utc_datetime()) for ti, e in zip(t, events)]

st.set_page_config(page_title="ISS Live Tracker", layout="wide")
st.title("  ISS Live Tracking Dashboard")
st.markdown("This dashboard shows the real-time position of the ISS on a world map.")

lat, lon, alt, speed_kms, speed_kmh, sat = get_iss_position()

st.metric(label="Latitude", value=f"{lat:.4f}°")
st.metric(label="Longitude", value=f"{lon:.4f}°")
st.metric(label="Altitude (km)", value=f"{alt:.2f} km")
st.metric(label="Speed", value=f"{speed_kms:.2f} km/s", delta=f"{speed_kmh:.0f} km/h")

m = folium.Map(location=[lat, lon], zoom_start=2, tiles="cartodbpositron")
folium.Marker([lat, lon], popup="ISS Position", icon=folium.Icon(color="red", icon="info-sign")).add_to(m)
st_folium(m, width=700, height=500)

#next pass over Jammu
st.subheader(" Next Pass Over Jammu (India)")
passes = get_next_pass_over_jammu(sat)
if passes:
    for name, time in passes:
        st.write(f"**{name}:** {time} UTC")
else:
    st.write("No visible pass in the next 24 hours.")

st.caption(f"Last updated: {datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S')} UTC")
st.info("Map updates when you refresh or rerun the app.")
```

Click on this link to live track the ISS, **http://192.168.29.68:8501**

Local URL: http://localhost:8501

Network URL: http://192.168.29.68:8501

In [1]:

```python
from IPython.display import Image, display
img_path = 'ISS Live Tracker.jpg'
display(Image(filename=img_path))
```

🛰️ ISS Live Tracking Dashboard

This dashboard shows the real-time position of the ISS on a world map.

Latitude

14.8469°

Longitude

1.4057°

Altitude (km)

416.55 km

Last updated: 2025-07-20 14:14:03 UTC

Click on this link to see the entire working dashboard **https://drive.google.com/file/d/1NAPZ-LqqdWY2u8uUjRrnX4n7ZabBf0Gq/view?usp=share_link**

For more details go to my GitHub **https://github.com/kritishnagyal**