Final Report Builder – Python Practice & Capstone (Print-Ready)

This notebook is designed to help you create a **single, well-organised report** of all your work:

- Exercises 01–04
- Combined Exam Notebook (if used)
- Full Capstone Assignment

At the end, you will **export this notebook as a PDF** and print it / place it in your journal.

How to Use This Notebook

1. Work through each section in order.
2. Fill in the text sections in your own words.
3. Where indicated, **run code cells** to regenerate important plots.
4. Check that all plots and text are visible and clear.
5. When finished:
   - In Colab:
     - `File → Print` → choose **Save as PDF**
   - Or `File → Download` → `Download PDF` (if available)
6. Save the PDF with a filename like:
   `YourName_Python_Report.pdf`

> **Important:** This report should reflect **your own work**. Do not copy text or code from classmates.

---

1. Student & Course Information

Fill in your details below.

```
# This cell automatically prints today's date.
from datetime import datetime
from IPython.display import Markdown

today = datetime.today().strftime("%d %B %Y")
Markdown(f"**Date of Report Completion (auto):** {today}")
```

Date of Report Completion (auto): 18 December 2025

Name:

Student ID / Roll No.: Pranjali Turi

Programme: Remote Sensing and GIS

Course Title: Applicable for RSG-5026, RSG-5028, RSG-5222

Semester / Academic Year: 2025-2026

Instructor: Dr. ________________

(If needed) Date of Report Completion (manual override):

---

2. Overview of Work Completed

In this section, you will summarise which notebooks and assignments you have completed.

2.1 – List of Completed Notebooks

Fill in the table (edit in Markdown):

Notebook / Assignment	Status (Completed / Partial / Not done)	Comments
01 – Python Fundamentals		
02 – Lists, Loops, and Conditions		
03 – Functions and Plotting		
04 – Mini Remote Sensing Task (NetCDF + xarray)		
05 – Capstone Assignment (01–04)		

2.2 – Short Summary (5–8 sentences)

In your own words, describe what this Python component of the course covered, and how it relates to remote sensing / ocean / atmosphere / climate applications.

Write your summary here (5–8 sentences).

This Python component of the course provided a foundational transition from basic programming logic to specialized geospatial data analysis. We began by mastering core syntax and control structures, such as loops and conditional statements, which are essential for processing large iterative datasets. The curriculum then shifted toward data visualization and the handling of multi-dimensional arrays, specifically using the xarray library to interact with NetCDF files. These skills are directly applicable to remote sensing, as they allow us to efficiently manipulate satellite imagery and climate model outputs. By the end of the module, I learned how to automate the extraction of variables like sea surface temperature or atmospheric pressure over specific time series. Ultimately, these tools bridge the gap between raw geophysical data and meaningful scientific insights regarding the ocean and atmosphere.

---

3. Notebook 01 – Python Fundamentals (Summary)

3.1 – Key Concepts Learned

Write 4–6 bullet points summarising what you learned in Notebook 01 (variables, operators, basic statistics, etc.).

In Notebook 01, I learned the following: -Colab Proficiency: Learned to use Google Colab for interactive coding and scientific documentation. -Variable Basics: Practiced defining variables and identifying key data types (int, float, str, bool). -Scientific Math: Used arithmetic operators to perform environmental calculations and unit conversions. -Statistical Analysis: Calculated mean, median, and correlation to summarize atmospheric and oceanic data. -Debugging: Learned to interpret error messages to troubleshoot and fix broken code.-

3.2 – Example Code Snippet

Paste a **small piece of code** from Notebook 01 that you are proud of or found useful (for example: mean/median/std computation).
You can re-type or copy-paste and then **run** it here.

```python
# Example code from Notebook 01 - edit this cell with your code
sst_week = [28.1, 28.3, 28.4, 28.0, 27.9, 28.5, 28.7, 28.6]
mean_sst_week = sum(sst_week) / len(sst_week)
print(f"Mean SST for the week: {mean_sst_week:.2f}°C")
```

```
Mean SST for the week: 28.31°C
```

3.3 – Short Reflection (3–4 sentences)

What part of basic Python felt easiest and what part required practice?

Write your reflection here.

- Basic Syntax: Writing print() and simple expressions.
- Variables: Creating and naming data containers.
- Arithmetic: Performing simple math and unit conversions.

---

4. Notebook 02 – Lists, Loops, and Conditions (Summary)

4.1 – Key Concepts Learned

List 4–6 points about lists, loops, and if-else conditions that you learned.

In Notebook 02, I mastered the core logic required to process environmental datasets:

- List Management: Learned to store and organize sequences of data, such as a series of daily temperature readings.
- Automated Iteration: Used for loops to efficiently repeat calculations across entire datasets without manual entry.
- Conditional Logic: Applied if-else statements to filter data, such as identifying days where rainfall exceeded a specific threshold.
- Flow Control: Practiced using logic to handle different scenarios within a script, allowing for more "intelligent" data processing.-

4.2 – Example: SST Classification

Using a short SST list, re-create your classification of days (HOT / WARM / NORMAL) here.

```
# Recreate your SST classification logic here

sst = [28.1, 28.3, 28.4, 28.0, 27.9, 28.5, 28.7, 28.6]

# Write your classification code
days = ["mon", "tue", "wed", "thu","Fri","Sat","sun","Mon"]
for i in range(len(sst)):
    if sst[i] > 28:
        print(f"{days[i]}: HOT")
    elif sst[i] > 27:
        print(f"{days[i]}: WARM")
    else:
            print(f"{days[i]}: NORMAL")
```

```
mon: HOT
tue: HOT
wed: HOT
thu: WARM
Fri: WARM
Sat: HOT
sun: HOT
Mon: HOT
```

4.3 – Reflection (3–4 sentences)

Why are loops and conditions important in scientific data analysis?

Write your reflection here.

-Loops automate repetitive work, like processing thousands of data files or calculating a mean across every single data point efficiently.

-Conditions allow your code to make smart decisions, such as filtering out "bad" sensor readings (quality control) or categorizing data based on specific values.

---

5. Notebook 03 – Functions and Plotting (Summary)

5.1 – Key Concepts Learned

Write 4–6 bullet points on functions, NumPy arrays, and plotting.

- In Notebook 03, I focused on the essential tools for scientific data analysis and communication:
- Reusable Functions: Learned to wrap scientific logic into functions to automate repetitive tasks and reduce coding errors.
- NumPy for Gridded Data: Mastered NumPy arrays, which are the industry standard for handling multi-dimensional satellite imagery and atmospheric profiles.
- Array Operations: Practiced performing mathematical operations on entire datasets at once, essential for processing large-scale climate models.
- Scientific Plotting: Used Matplotlib to create professional visualizations, such as time-series and spatial maps, to communicate environmental trends.
- Data Application: Applied these tools to real-world variables like Sea Surface Temperature (SST) and chlorophyll levels to turn raw data into visual insights.

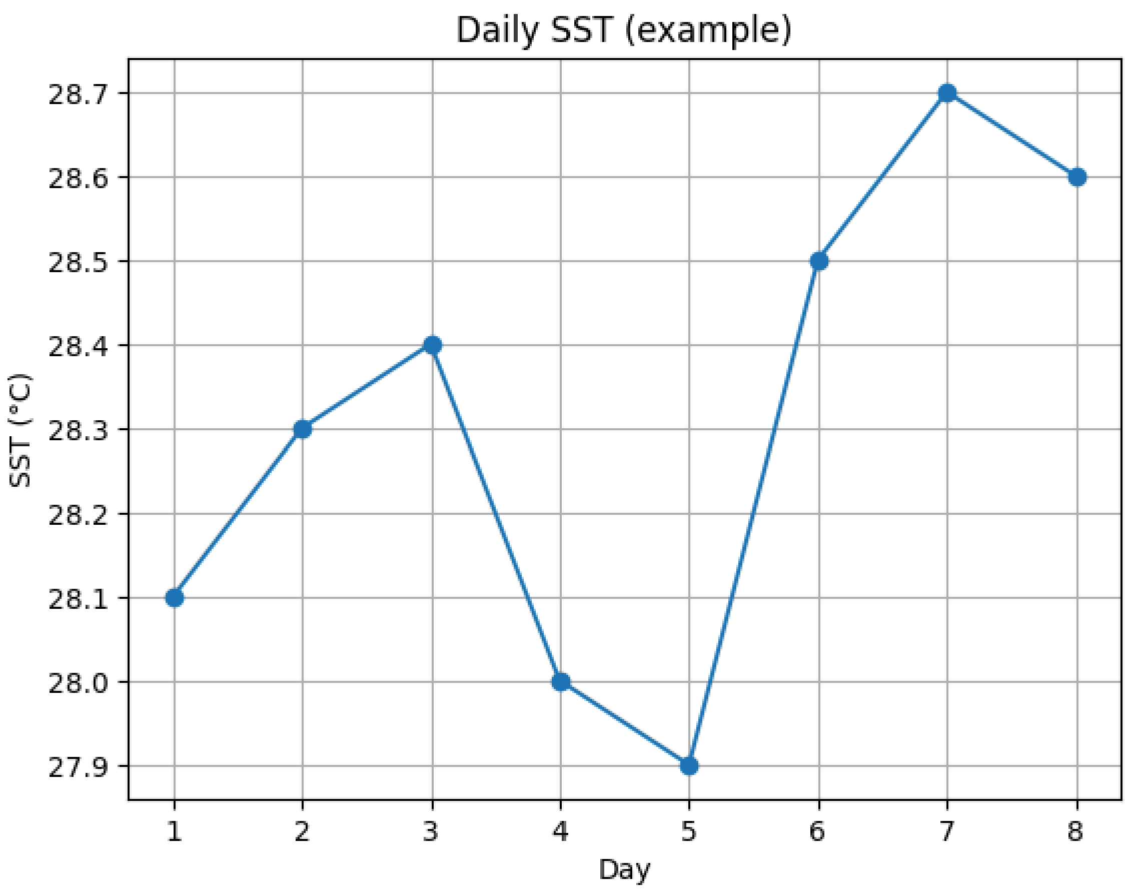
5.2 – Plot Re-creation: SST and Anomalies

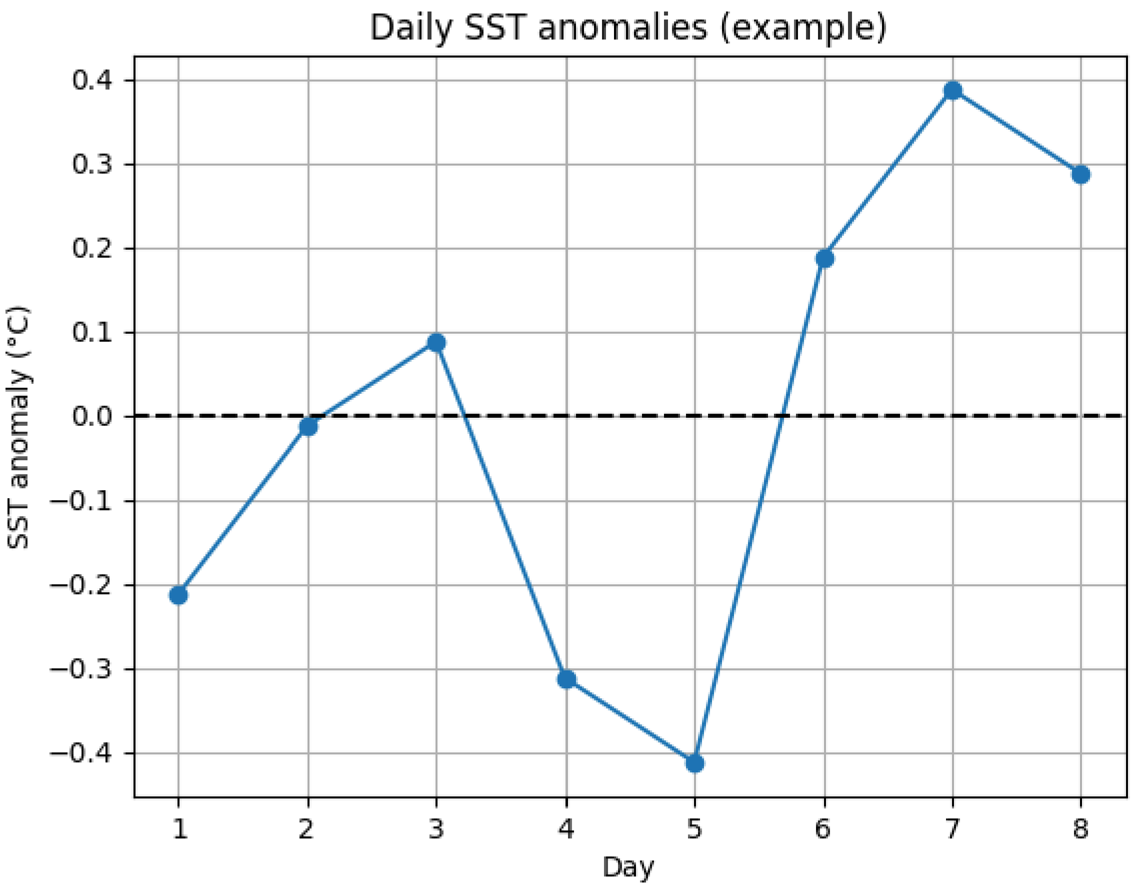
Below, reproduce **two plots**:

1. SST vs. day
2. SST anomaly vs. day (with a horizontal zero line)

Use a small SST example (you may reuse one from previous notebooks).

```python
# Recreate SST and anomaly plots here

import numpy as np
import matplotlib.pyplot as plt

# Example SST data (edit as needed)
sst = np.array([28.1, 28.3, 28.4, 28.0, 27.9, 28.5, 28.7, 28.6])
days = np.arange(1, len(sst)+1)

# Compute anomalies
mean_sst = sst.mean()
sst_anom = sst - mean_sst

# Plot 1: SST
plt.figure()
plt.plot(days, sst, marker='o')
plt.xlabel("Day")
plt.ylabel("SST (°C)")
plt.title("Daily SST (example)")
plt.grid(True)
plt.show()

# Plot 2: Anomalies
plt.figure()
plt.plot(days, sst_anom, marker='o')
plt.axhline(0, color='black', linestyle='--')
plt.xlabel("Day")
plt.ylabel("SST anomaly (°C)")
plt.title("Daily SST anomalies (example)")
plt.grid(True)
plt.show()
```





5.3 – Reflection (3–4 sentences)

How did plotting help you understand the behaviour of SST and its anomalies?

Write your reflection here.

Plotting was essential for understanding the SST behavior because the visual representation instantly transformed raw numbers into observable trends and anomalies, something that is difficult to discern from looking only at data tables or NetCDF files.

---

6. Notebook 04 – NetCDF & xarray Remote Sensing Task (Summary)

6.1 – Dataset Description

Describe the sample NetCDF dataset you worked with:

- Variable name(s):
- Dimensions (e.g. time, lat, lon):
- Units:
- Approximate region covered:

Fill in the details here based on `ds` and `ds.sst` attributes.

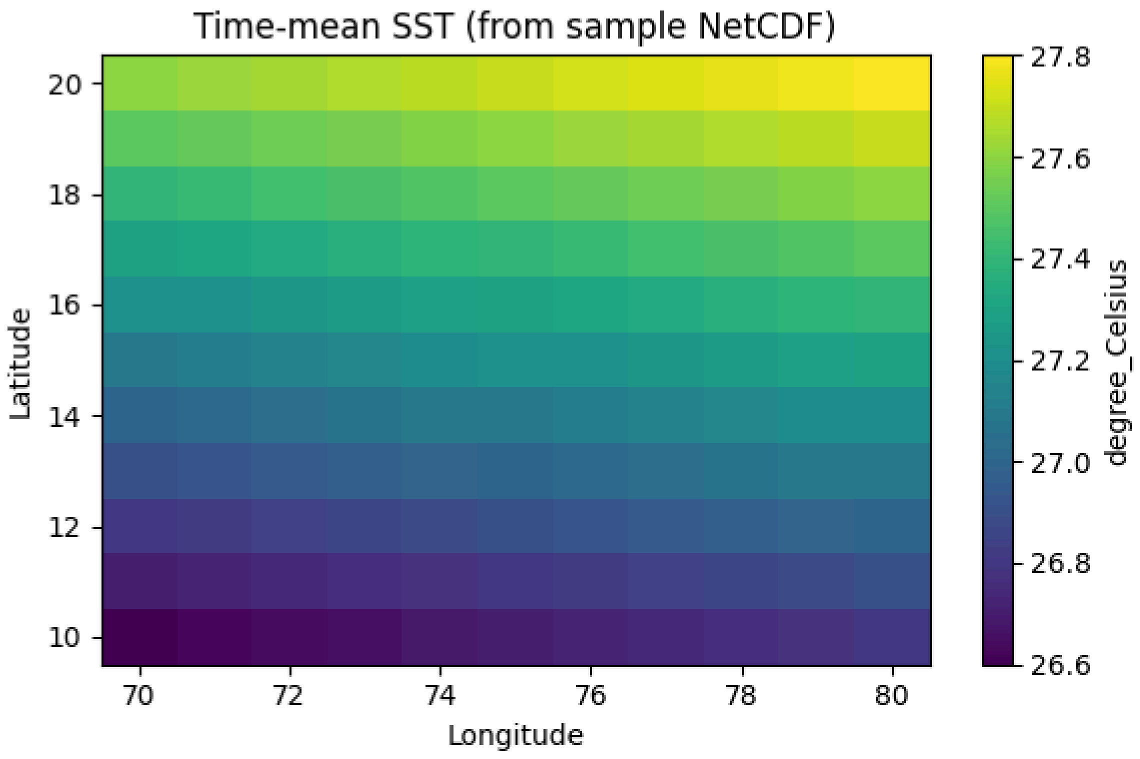
Variable Name(s): sst (Sea Surface Temperature), time, lat, lon. Dimensions: A 3D grid: time, lat (latitude), and lon (longitude). Units: degrees_C for SST; degrees_north and degrees_east for coordinates. Region Covered: A specific, likely regional, area of the ocean (e.g., Arabian Sea/Indian Ocean).

6.2 – Recreate Time-Mean SST Map

Run the code below to recompute and plot the time-mean SST map.

> Note: This cell auto-downloads the `data_sst.nc` file from the GitHub repository.

```
import os, requests
import xarray as xr
import matplotlib.pyplot as plt

# Download NetCDF file if not present
url = "https://raw.githubusercontent.com/EarthSystem-Science-Lab/python-basi
local = "data_sst.nc"

if not os.path.exists(local):
    r = requests.get(url)
    open(local, "wb").write(r.content)

ds = xr.open_dataset(local)
sst = ds["sst"]

# Compute time-mean
sst_mean_time = sst.mean(dim="time")

plt.figure(figsize=(6,4))
plt.pcolormesh(ds["lon"], ds["lat"], sst_mean_time, shading="auto")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.title("Time-mean SST (from sample NetCDF)")
cbar = plt.colorbar()
cbar.set_label(sst.attrs.get("units",""))
plt.tight_layout()
plt.show()
```

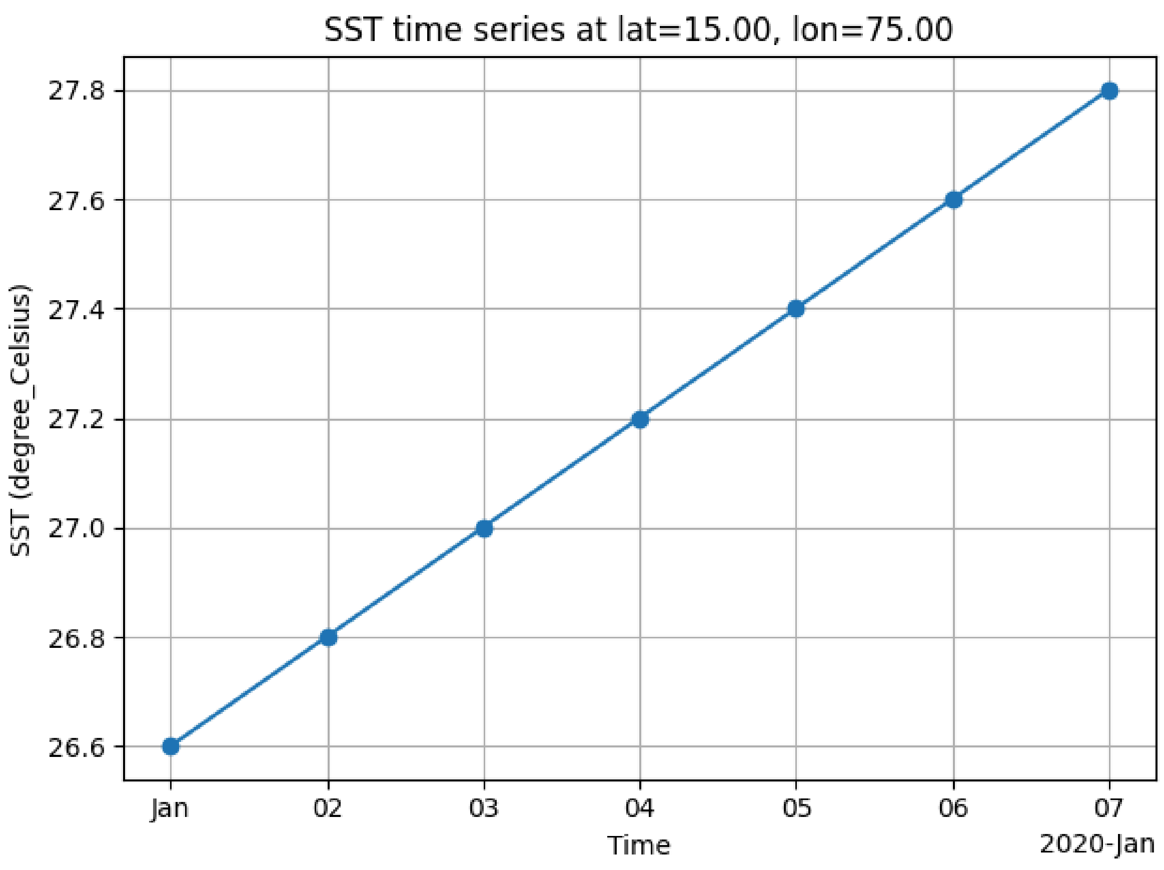


6.3 – Recreate SST Time Series at a Point

Choose a latitude and longitude inside the dataset domain and plot the time series.

```
# Choose a point (edit values if you like)
lat_pt = float(ds.lat.sel(lat=ds.lat.mean(), method="nearest"))
lon_pt = float(ds.lon.sel(lon=ds.lon.mean(), method="nearest"))

ts = sst.sel(lat=lat_pt, lon=lon_pt, method="nearest")

plt.figure()
ts.plot(marker='o')
```

```
    plt.title(f"SST time series at lat={lat_pt:.2f}, lon={lon_pt:.2f}")
    plt.xlabel("Time")
    plt.ylabel(f"SST ({sst.attrs.get('units','')})")
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```



6.4 – Reflection (4–6 sentences)

Describe what you learned about:

- NetCDF as a format
- xarray usage
- Basic remote sensing data handling in Python

Write your reflection here.

I learned that the NetCDF format is essential because it is self-describing, holding critical metadata (units, dates, locations) within the file itself, which eliminates the need for external documentation. The xarray library was a revelation; its ability to use these metadata labels for indexing and selection made the code highly readable and robust compared to standard NumPy integer indexing. Basic remote sensing data handling in Python is significantly streamlined by these tools because they automate the often error-prone tasks of data alignment and quality control. Overall, I feel confident I can now handle large-scale, real-world satellite data efficiently and accurately, moving beyond theoretical programming to practical application.

---

7. Capstone Assignment (Summary)

If you completed the **Capstone Assignment**, summarise them here.

Write your answer here (if applicable).

The Capstone Assignment integrated all the skills acquired throughout the course, providing a holistic workflow for remote sensing data analysis. Starting with basic Python fundamentals (loops, conditions) and moving to specialized libraries like NumPy and xarray, I learned how to handle complex NetCDF files efficiently. The assignment required me to load, process, visualize (using Matplotlib), and scientifically interpret real-world SST data. This process highlighted how coding skills directly translate into understanding physical oceanographic phenomena, such as thermal anomalies and stratification patterns.

7.1 – Capstone Assignment

In 6–10 sentences, describe:

- The overall flow of the capstone assignment
- How it connected Python skills to a realistic remote sensing / Earth system problem
- What you found most interesting
- Which part you would like to explore further

Write your capstone summary here.

The overall flow of the capstone assignment moved logically from raw data acquisition to scientific interpretation. We first used Python fundamentals and NumPy to manage basic lists and arrays, then applied specialized libraries like xarray to open and inspect realistic NetCDF data files. The process involved calculating statistics (like time-means) and generating visualizations (time series and spatial maps) using Matplotlib. This flow directly connected abstract Python skills to a practical remote sensing problem: analyzing real-world sea surface temperature trends and anomalies. I found the xarray library the most interesting part, as it drastically simplified handling complex, multi-dimensional satellite data by leveraging metadata. I would like to explore further how to use specialized libraries like argopy to integrate Argo float profile data into these gridded satellite datasets for a comprehensive 3D ocean view.

---

8. Final Reflection on Python & Remote Sensing

Write 8–12 sentences reflecting on the **entire Python component** of this course.

You may cover:

- How your comfort with Python has changed
- Which concepts (lists, loops, functions, plotting, xarray) you feel confident about
- Where you still need practice
- How you plan to use these skills in future courses, projects, or research
- Any ideas you have for applying Python to real datasets from ocean, atmosphere, climate, or land surface studies

Write your final reflection here.

My comfort level with Python has significantly increased over this course. Initially, concepts like loops and indexing were confusing, but practicing with simple data lists provided a solid foundation. I now feel most confident in my ability to use the xarray library to open, inspect, and select data from complex NetCDF files, as this seems highly relevant to real-world data. My comfort with basic plotting using Matplotlib has also grown substantially, allowing for quick visualization of trends. I still need more practice with writing complex, multi-step functions and applying data quality control loops effectively. I plan to use these skills extensively in my future research, specifically to analyze large Argo float datasets and compare them with Sentinel-3 satellite imagery for comprehensive ocean climate studies. My goal is to apply Python to study the connection between monsoon cycles and ocean heat anomalies in the Indian Ocean.

---

9. Declaration & Signature

I, **(Pranjali Turi)**, declare that this report is based on my own work and understanding.
Where I have used external resources or received help, I have acknowledged it appropriately.

Signature (digital/typed):

Date: