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): 10 December 2025

Name: Esther Smila Gnana

Student ID / Roll No.: 202205394

Programme: MSc Remote Sensing & GIS

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

Semester / Academic Year:2025-2026

Instructor: Dr. Arjun Adikari

(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).

In this python course we covered the basics and fundamentals of python. Learing how to define a variable (stores the values in it example: a = 12). How to display the data using print statment..(example : print("The SST of Day 1:,sst") and learned to use different data types like string,integer,float,boolen,etc. Worked to compute basic statistics and visualize data using plots. So all this relates to RS/ocean/atmosphere/climate applications by heling us analyse data of a particular study area for various parameters like SST,humidity,wind speed,etc. which helps in weather prediction,climate studies,oceanography,etc.And displays the data for better visualization using line graph,heat map and so on.

---

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.).

- Learned how to create and assign variables of different types (integers, floats, strings, booleans) and understood how Python stores and updates values.
- Practiced using basic statistical functions (mean, median, mode, min, max, range) to summarise datasets.
- Worked with type conversion (e.g., converting strings to numbers) to ensure variables are usable in calculations.
- Introduced to simple data structures like lists for storing collections of values and performing basic operations.
- Learned to use built-in functions and understood how functions help avoid repetition and improve clarity.

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.

```
# Example code from Notebook 01 – edit this cell with your code

import statistics as stats
```

```
sst = [28.1, 28.3, 28.3, 27.9, 28.5, 28.7, 28.3]

mean_sst = stats.mean(sst)
median_sst = stats.median(sst)
mode_sst = stats.mode(sst)

print("Mean SST:", mean_sst)
print("Median SST:", median_sst)
print("Mode SST:", mode_sst)
```

```
Mean SST: 28.3
Median SST: 28.3
Mode SST: 28.3
```

3.3 – Short Reflection (3–4 sentences)

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

Write your reflection here.

Python Fundamentals,Lists, Loops, Functions felt the easiest while Plotting, Conditions and getting the path in mini rs task was a bit difficult and require more practice.

---

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.

- Learned how to create, index, slice, and modify lists, and understood that lists can hold multiple data types.
- Practiced using loops (for and while) to repeat actions, iterate over lists, and automate repetitive tasks.
- Understood how if, elif, and else statements allow programs to make decisions based on conditions.
- Explored how to combine loops with conditions to filter data, count occurrences, or perform actions only when criteria are met.

- earned about common list operations (append, remove, sort, len) to manage and manipulate collections of data.

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

sst = [28.1, 28.3, 28.3, 27.9, 28.5, 28.7, 28.3]

hot = 0
warm = 0
normal = 0

for s in sst:
    if s > 29:
        hot += 1
    elif 28 < s <= 29:
        warm += 1
    else:
        normal += 1

print("Hot days:", hot)
print("Warm days:", warm)
print("Normal days:", normal)
```

```
Hot days: 0
Warm days: 6
Normal days: 1
```

4.3 – Reflection (3–4 sentences)

Why are loops and conditions important in scientific data analysis?

Write your reflection here.

Loops and conditions are crucial in scientific data analysis because they allow you to automate repetitive tasks and efficiently process large datasets. Loops let you perform the same calculation or transformation on many data points without manual work.

Conditions enable your code to make decisions, such as filtering data or identifying important patterns, which leads to more accurate and meaningful analyses.

---

5. Notebook 03 – Functions and Plotting (Summary)

5.1 – Key Concepts Learned

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

- Functions help organise analysis workflows by packaging repeated atmospheric calculations,such as aerosol settling velocity or humidity-dependent particle growth into reusable, clear code.
- NumPy arrays allow fast, vectorised operations on large atmospheric datasets, making it efficient to handle time series of concentration, size distributions, or vertical profiles.
- Plotting tools (e.g., Matplotlib) enable visualisation of aerosol properties, including size distributions, temporal concentration trends, and spatial variability.Visualisation helps identify key atmospheric processes, such as diurnal patterns, plume evolution, or boundary-layer mixing effects
- Combining functions, NumPy arrays, and plotting provides a powerful workflow for analysing, modelling, and interpreting aerosol and atmosphere interactions.

5.2 – Plot Re-creation: SST and Anomalies

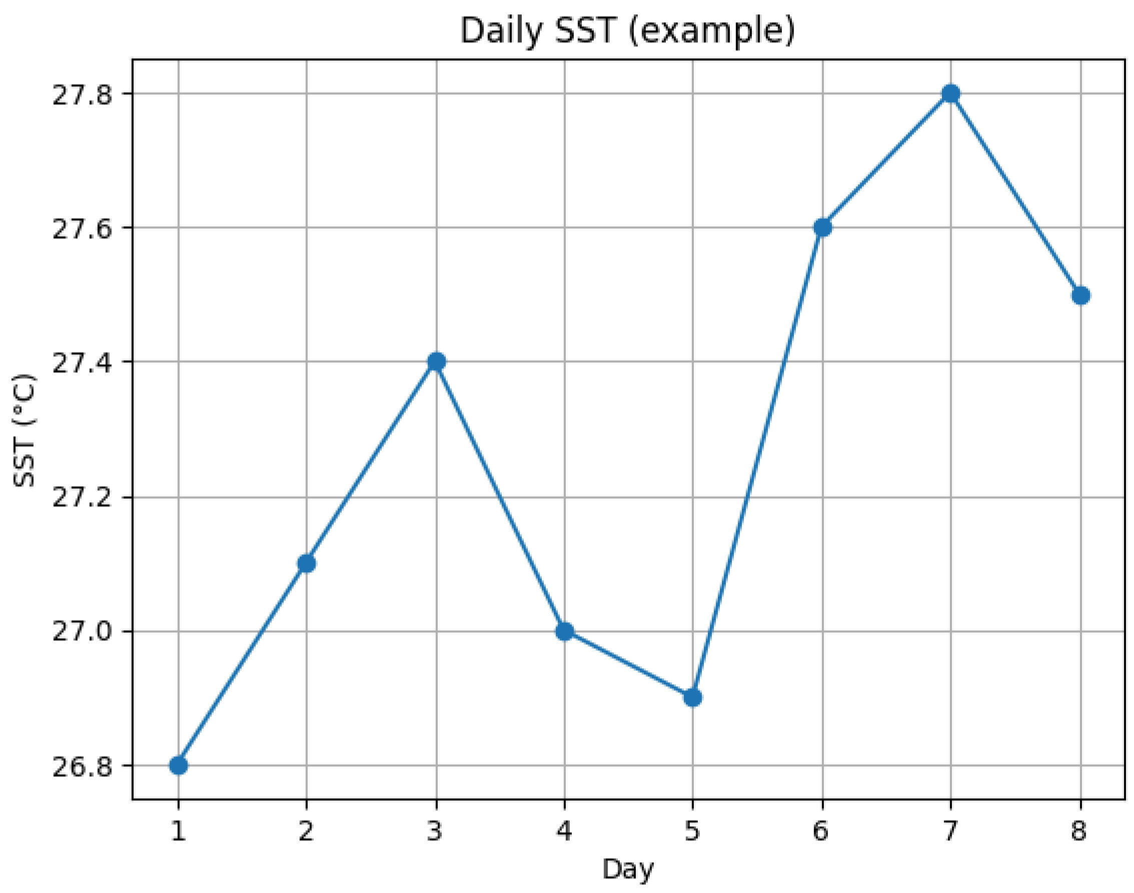
Below, reproduce **two plots**:

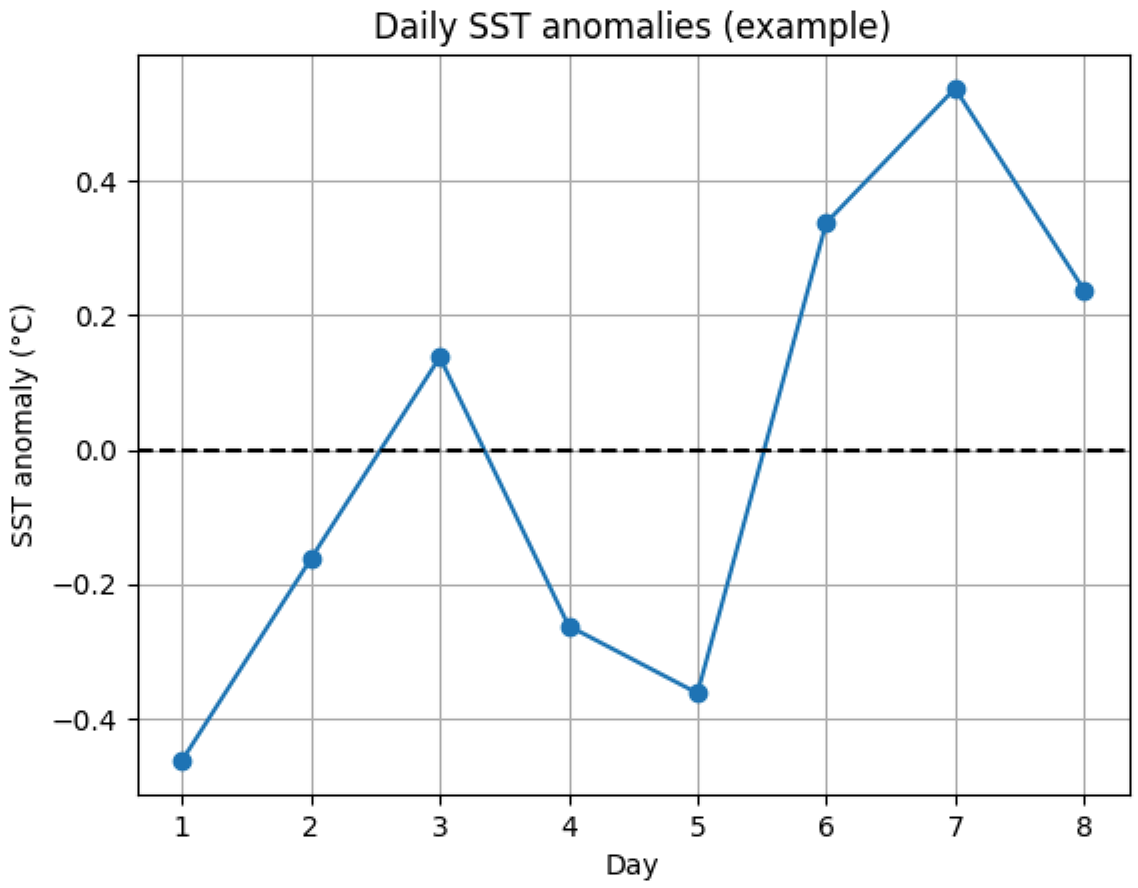
1. SST vs. day
2. SST anomaly vs. day (with a horizontal zero line)

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

```
# Recreate SST and anomaly plots here

import numpy as np
import matplotlib.pyplot as plt

# Example SST data (edit as needed)
sst = np.array([26.8, 27.1, 27.4, 27.0, 26.9, 27.6, 27.8, 27.5])
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()
```


Daily SST (example)
SST (°C)
27.8
27.6
27.4
27.2
27.0
26.8
1
2
3
4
5
6
7
8
Day


Daily SST anomalies (example)
SST anomaly (°C)
0.4
0.2
0.0
−0.2
−0.4
1
2
3
4
5
6
7
8
Day

5.3 – Reflection (3–4 sentences)

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

Write your reflection here.

Plotting made it easy to visualise how SST changes over time, revealing trends, peaks, and variability that aren't obvious from raw numbers. Comparing SST with its anomalies helped highlight deviations from the mean, making unusual warming or cooling events immediately visible.

---

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.

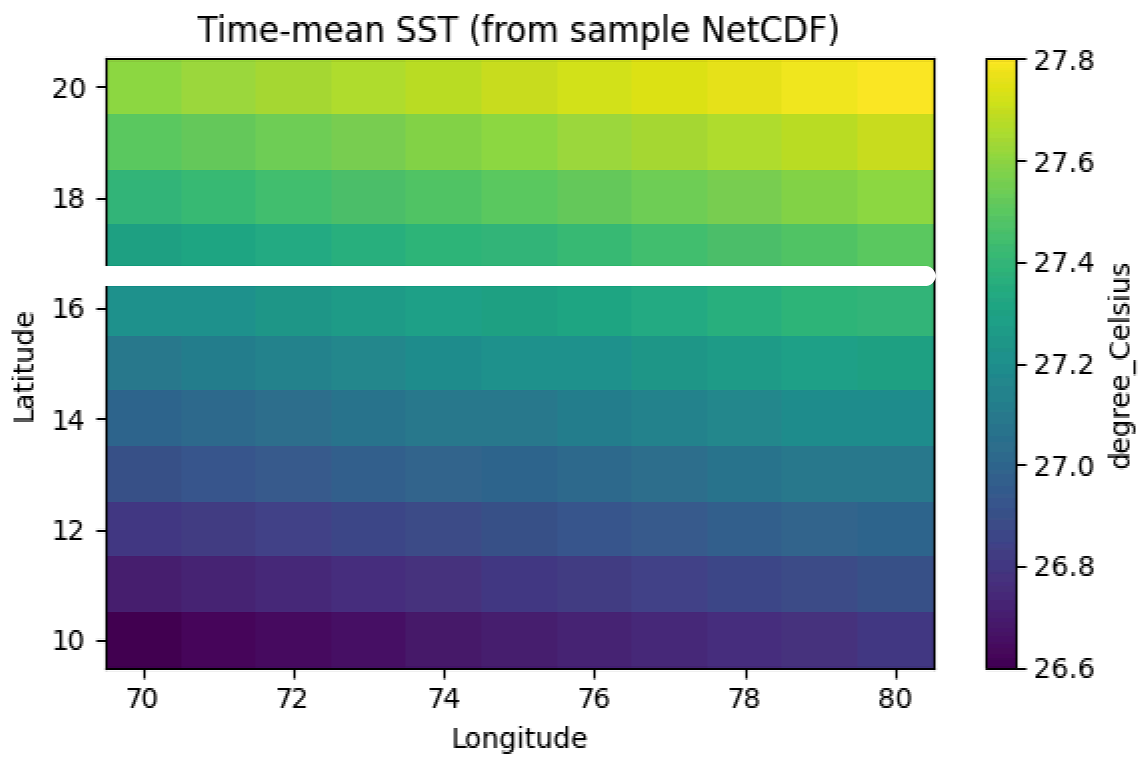
6.2 – Recreate Time-Mean SST Map

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-basics/
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()
```

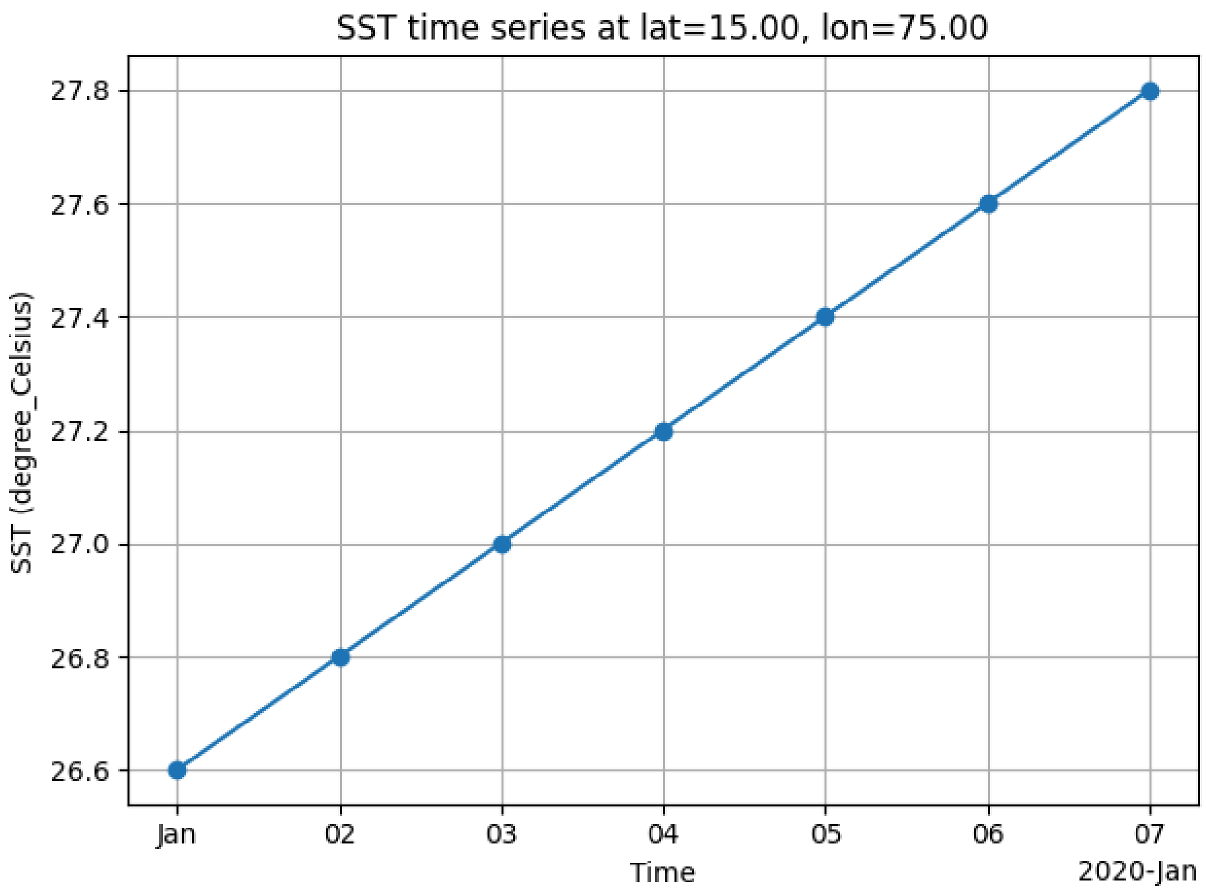


6.3 – Recreate SST Time Series at a Point

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 NetCDF is a self-describing, platform-independent format widely used in atmospheric and ocean sciences for storing multidimensional data such as time, latitude, longitude, and vertical levels. Using xarray, I learned how to open these datasets, explore their metadata, and work with labelled dimensions, which makes complex data easier to manipulate. I also practiced performing operations like selecting slices, computing statistics, and creating quick plots directly from xarray objects. Finally, I learned the basics of handling remote sensing data in Python.

---

7. Capstone Assignment (Summary)

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

Write your answer here (if applicable).

Capstone Assignment combines all major skills from Notebooks 01-04, including Python fundamentals, list operations, loops, conditions, functions, NumPy, xarray, NetCDF, plotting, and scientific interpretation. It begins with basic variable manipulation and statistical calculations, then progresses to working with lists and loops to classify and analyse SST data. You then apply functions and NumPy arrays to compute anomalies and produce visualisations that help interpret SST patterns. Finally, the assignment introduces handling real scientific data using NetCDF and xarray, including dataset exploration, time-mean mapping, time-series extraction, and connecting these results to physical oceanographic processes through a written scientific interpretation.

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 capstone assignment followed a progressive flow, starting with Python fundamentals such as variables, arithmetic, and basic statistics, then moving to lists, loops, and conditions to manipulate and classify SST data. It introduced functions, NumPy arrays, and plotting to compute anomalies and visualise patterns, bridging coding skills with data analysis. The final section focused on realistic remote sensing data, using NetCDF and xarray to explore, summarise, and plot sea surface temperature datasets. This structure showed clearly how foundational Python concepts can be applied to Earth system science, making abstract skills tangible by analysing temperature trends, gradients, and variability.

I found it most interesting to compute SST anomalies and visualise them, because this highlighted short-term variability that is not obvious from raw temperature values alone. Exploring the time-mean maps revealed the meridional gradient and allowed me to connect observed patterns to physical processes like mixing, stratification, and monsoon-driven currents. The ability to extract a time series at a specific location and interpret warming trends felt especially rewarding, as it mimicked real-world analysis of satellite or in-situ data. I would like to explore more advanced spatial analysis and animation of SST changes over time, which could reveal dynamic processes like upwelling or seasonal shifts in a more detailed way. Overall, the assignment effectively tied together Python programming, data manipulation, and scientific reasoning in a real-world Earth system context.

---

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.

Before this course, I had learned basic Python during my BSc in Computer Science, so I was familiar with syntax, variables, and simple programs. However, I had little experience applying Python to real scientific datasets or Earth system problems. This course helped me bridge that gap by showing how to use Python for data analysis, visualization, and working with multidimensional datasets like NetCDF. I now feel confident with lists, loops, functions, plotting, NumPy arrays, and xarray, and I understand how these tools can be applied to study phenomena like sea surface temperature patterns and anomalies.

I still need more practice with advanced data manipulation, handling large datasets, and automating analyses, but I feel much more prepared to explore real-world datasets. In the future, I plan to use these skills in projects or research related to oceanography, climate, atmospheric studies, or remote sensing. I am particularly excited to analyze satellite data, create time-series and spatial plots, and investigate variability and trends in environmental datasets. Overall, the course has strengthened both my Python skills and my ability to connect programming with scientific interpretation, making me more confident in tackling Earth system problems.
