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): 11 December 2025

Name: TRUPTI DNYANESHWAR SUTAR

Student ID / Roll No.: 202204978

Programme: MSC REMOTE SENSING AND GIS

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

Semester / Academic Year: 2025-26

Instructor: Dr.Arjun Adhikar

(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	Completed	
02 – Lists, Loops, and Conditions	completed	
03 – Functions and Plotting	completed	
04 – Mini Remote Sensing Task (NetCDF + xarray)	completed	
05 – Capstone Assignment (01–04)	completed	

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).

The Python part of the course taught us the basic skills we need to start working with scientific data. We learned how to use Google Colab, create variables, do simple maths, and calculate things like averages and correlations. it relates to remote sensing , ocean , atmosphere , climate applications such as sea surface temperature,

rainfall, or satellite images—and Python helps us organise, process, and understand that information quickly

---

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 variables and check their types int, float, string.

learned calculation like +, -, *, /, including decimal

Used print() and f-strings to show results

Learned to Calculated mean, median, and mode with Python lists and correlation between two datasets and interpret positive and negative relationships

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?

I found the basic calculations in Python the easiest because the syntax felt simple and required practice for computing correlation

---

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.

- Lists let us keep many pieces of information together, and we can choose any item by its position
- Loops help us do something again and again, like checking each item in a list.
- If-else statements help the program choose what to do, depending on whether something is true or not.
- A for loop repeats a set of steps for each item in a list.
- We can use indexes to access items in a list

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
hot_count = 0
warm_count = 0
normal_count = 0

for i, item in enumerate(sst):
  if item > 29:
    print(f"Day {i+1}: HOT")
    hot_count += 1
  elif 28 <= item <= 29:
    print(f"Day {i+1}: WARM")
    warm_count += 1
  else:
    print(f"Day {i+1}: NORMAL")
    normal_count += 1
```

```
print(f"\nCounts:")
print(f"HOT: {hot_count}")
print(f"WARM: {warm_count}")
print(f"NORMAL: {normal_count}")
```

```
Day 1: WARM
Day 2: WARM
Day 3: WARM
Day 4: WARM
Day 5: NORMAL
Day 6: WARM
Day 7: WARM
Day 8: WARM

Counts:
HOT: 0
WARM: 7
NORMAL: 1
```

4.3 – Reflection (3–4 sentences)

Why are loops and conditions important in scientific data analysis?

Write your reflection here.

Loops allow scientists to process many data points without writing repetitive code and Conditions help in making decisions based on data. Together, loops and conditions save time and reduce errors, letting researchers analyze data systematically and consistently.

---

5. Notebook 03 – Functions and Plotting (Summary)

5.1 – Key Concepts Learned

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

- Functions let you reuse code instead of repeating the same steps again and again
- NumPy arrays store data in a fast and efficient way, which is perfect for time-series and scientific datasets.by useing numpy can do math on whole arrays at once, which makes calculations quick and simple.
- Plots help you see patterns in your data, like trends or sudden changes. Anomaly plots show what is warmer,colder or higher,lower than normal.
- Graphs make patterns clear without lots of math. It's more visual, so it's quicker to understand

-

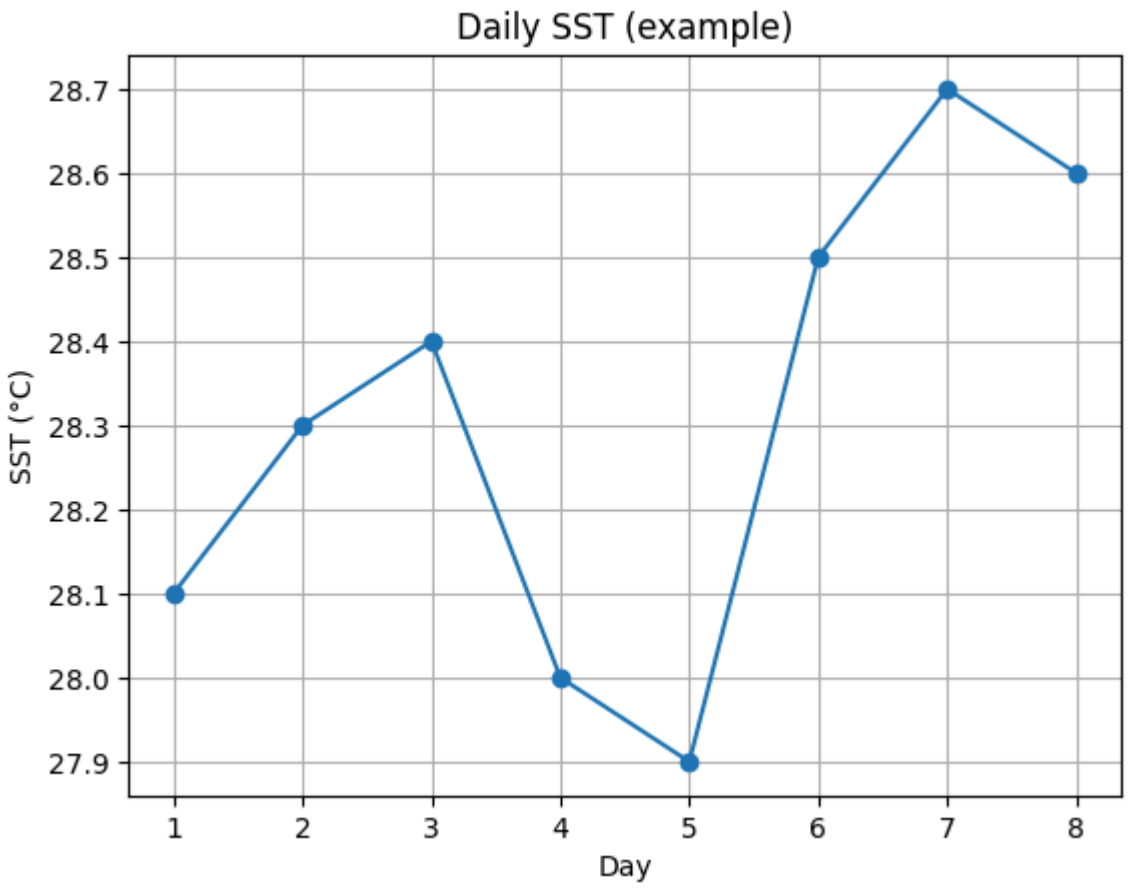
5.2 – Plot Re-creation: SST and Anomalies

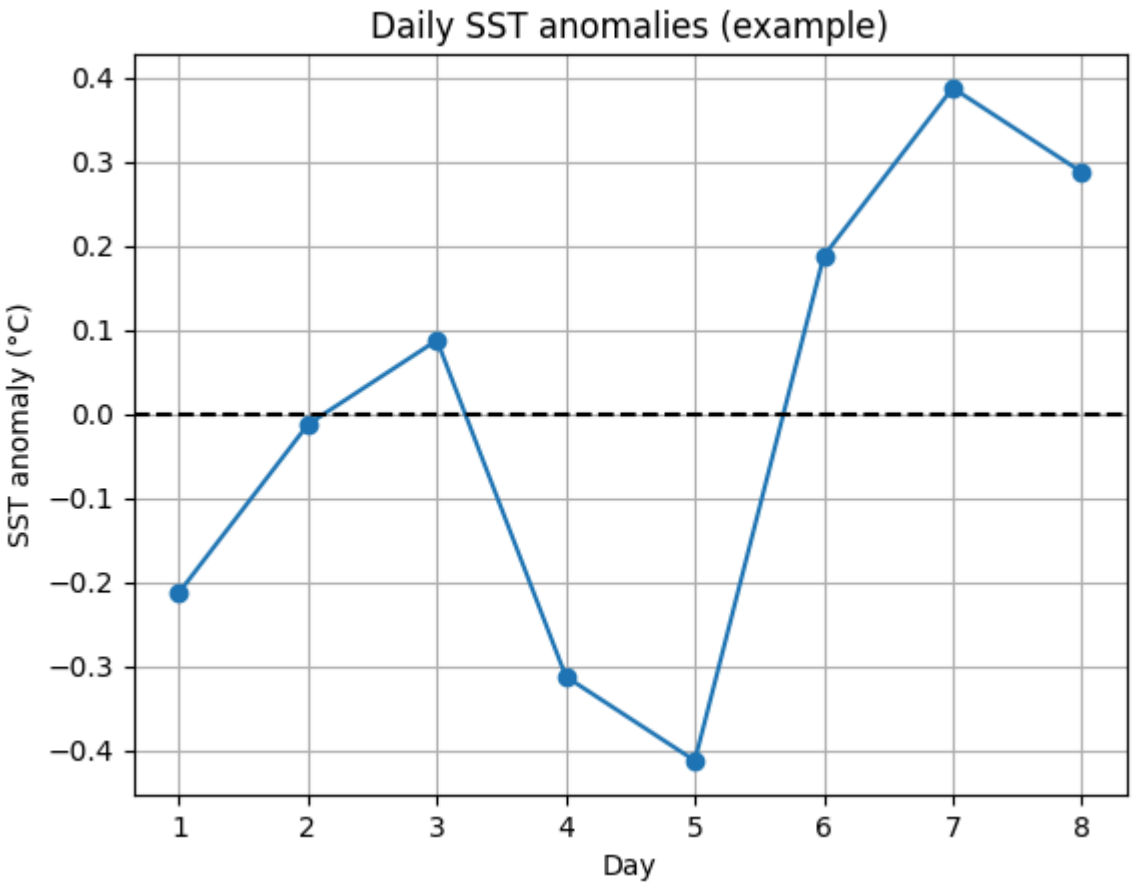
Below, reproduce **two plots**:

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([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 the SST helped me clearly see how the temperatures change over time, including the usual seasonal ups and downs. When I graphed the SST anomalies, it was easier to spot unusual warm or cold periods because the normal seasonal pattern was removed. This made events like sudden warming stand out more. Overall, the plots made the data much easier to understand.

---

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

6.1 – Dataset Description

Describe the sample NetCDF dataset you worked with:

- Variable name(s):sst (Sea Surface Temperature)
- Dimensions (e.g. time, lat, lon): time: 7 days (2020-01-01 to 2020-01-07)

lat: 11 values (10°N to 20°N)

lon: 11 values (70°E to 80°E)

- Units:degrees Celsius (degree_Celsius)
- Approximate region covered: A small grid in the tropical Indian Ocean, roughly between 10–20°N and 70–80°E.

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

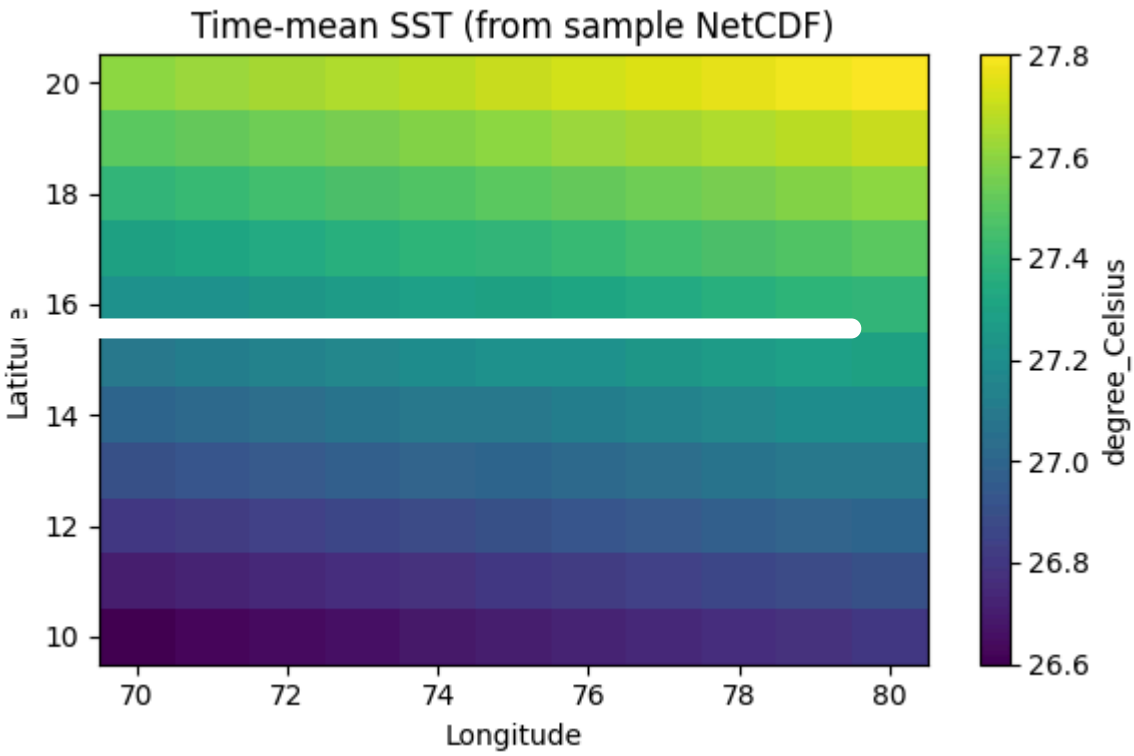
6.2 – Recreate Time-Mean SST Map

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-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()
```

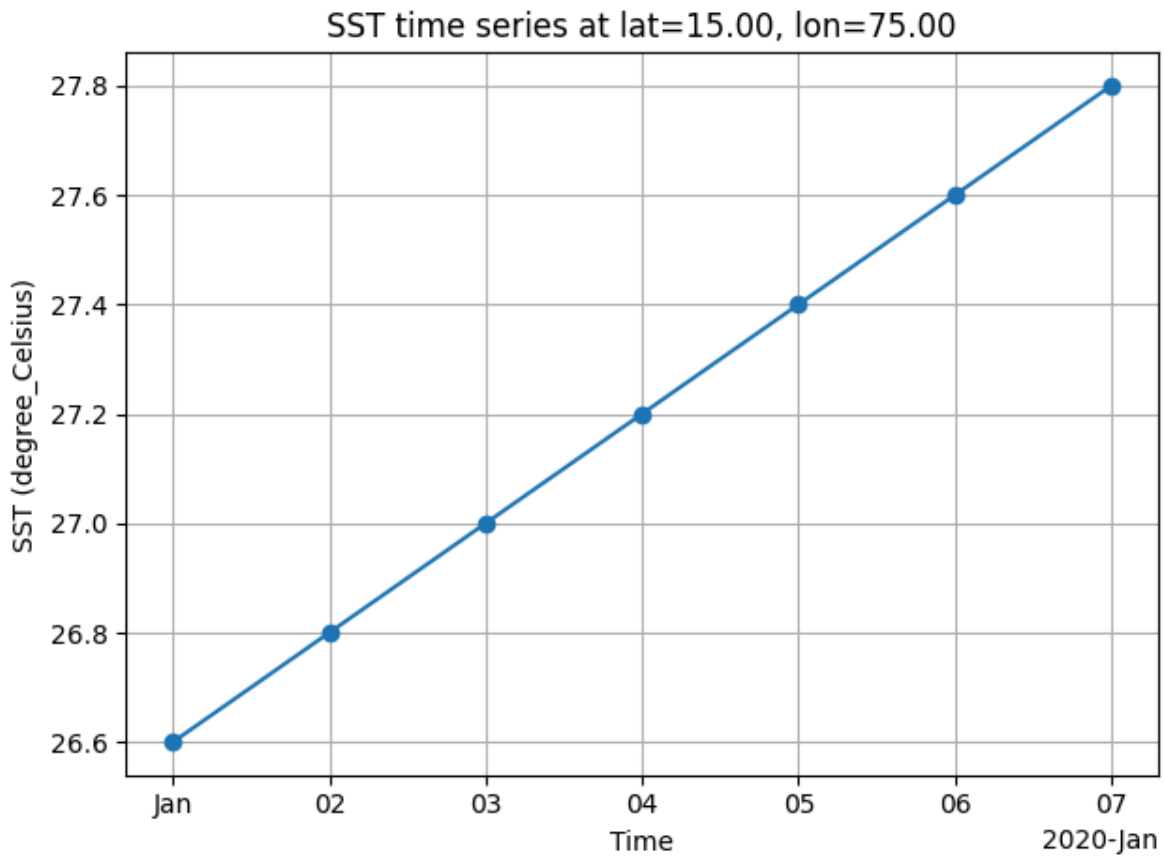


6.3 – Recreate SST Time Series at a Point

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 NetCDF is a format for storing scientific data with dimensions like time, latitude, and longitude, along with metadata. xarray makes it easy to open these files, select variables, and work with labeled arrays instead of just numbers. Using Python, I was able to calculate statistics and make plots like time-mean maps and time series. This helped me understand how to explore and visualize remote sensing data efficiently

7. Capstone Assignment (Summary)

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

Write your answer here (if applicable).

In this assignment, I practiced Python basics, including statistics, loops, and conditions, using SST data. I calculated daily SST statistics, classified days as HOT, WARM, or NORMAL, and computed a 3-day rolling mean. I wrote functions to convert Celsius to Kelvin and to compute anomalies, and then plotted SST and anomaly time series with proper labels and grids. Using xarray, I explored a NetCDF SST dataset, checking dimensions, variables, and attributes, and computed global statistics. I created a time-mean map and a time series at a chosen location to observe patterns and variability. This helped me understand how Python can be used to analyze remote sensing and oceanographic data, detect anomalies, and study physical processes in the ocean.

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 guided me through a full workflow of analyzing sea surface temperature (SST) data using Python. It started with basic Python skills like arithmetic, statistics, loops, and conditions, then moved to functions, arrays, and plotting. I applied these skills to compute rolling means, anomalies, and visualize time series. The assignment then introduced working with a real-world NetCDF dataset using xarray, where I explored dimensions, variables, global attributes, and created time-mean maps and point-based time series. This connected programming skills directly to a realistic Earth system problem, showing how to handle, analyze, and visualize remote sensing data. I found it most interesting to see spatial patterns and temporal variability in SST emerge clearly from the plots and anomaly calculations.

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.

Throughout this course, I have learned many Python skills, but I still feel that I need more practice to become fully confident. I now understand the basics of lists, loops, and conditions, and I can write simple functions to automate calculations like anomalies or temperature conversions. I have also learned to create plots with Matplotlib and explore xarray NetCDF datasets, which helps in visualizing and analyzing spatial and temporal data.

---

9. Declaration & Signature

I, **(TRUPTI DNYANESHWAR SUTAR)**, 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: 12-12-2025