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: Suroj Dhara

Student ID / Roll No.: 202204586

Programme: MSc Remote Sensing & GIS

Course Title: RSG-5015 Ocean Optical Modelling Practical

Semester / Academic Year: SEM 2 (2025-2026)

Instructor: Dr. Arjun Adhikari

(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.

The python component of the course introduced the core programming skills needed for scientific data analysis.

Starting from basic syntax and processing through loops, conditions & functions. Through plotting and data manipulation task, the course demonstrated how python can be used to explore spatial & temporal patterns in environmental data.

It emphasized working with real world dataset, Particularly NetCDF files,using libaries such as Numpy,Matplotlib And Xarray.

The mini remote sensing task showed how satellite derived variables can be read, processed & visualize efficiently.These are directly applicable to oceanic, Atmospheric & Climate studies.

---

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.).

- using Print() function to print
- understanding data types (int, float, str, bool)
- Converting datatypes(e.g., str to int)
- Mean, Median, Mode
- Basic maths using python.

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.

```
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?

Using Print function and basic maths felt easier Correlation betwwen to variables required parctice.

---

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.

- Creating weekly SST lists
- Loops
- Computing mean using loop
- Classify sst days(hot or normal days)
- Filtering SST values

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]
threshold = 28.5
for i in range(len(sst)):
    temp = sst[i]
    day = i + 1
    # TODO: write if condition here
    if temp > threshold:
      print("Day", day,":", "HOT")
    if temp < threshold:
      print("Day", day,":", "NORMAL")
```

```
Day 1 : NORMAL
Day 2 : NORMAL
Day 3 : NORMAL
Day 4 : NORMAL
Day 5 : NORMAL
Day 7 : HOT
Day 8 : HOT
```

4.3 – Reflection (3–4 sentences)

Why are loops and conditions important in scientific data analysis?

Loops and conditions are important in scientific data analysis because they allow automated, efficient, and flexible handling of large and complex datasets. Loops make it possible to repeat calculations over many observations, time steps, grid cells, or files without writing the same code multiple times. Conditions (if–else statements) help apply logical rules, such as filtering invalid data, handling missing values, or selecting data that meet specific criteria. Together, they enable scientists to process data systematically, reduce human error, and implement decision-based analyses. This is especially important in fields like remote sensing and climate science, where datasets are large and often require quality control and conditional processing.

---

5. Notebook 03 – Functions and Plotting (Summary)

5.1 – Key Concepts Learned

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

- Functions: reusable blocks of scientific logic
- NumPy arrays: the language of satellite grids, atmospheric profiles, and time-series
- Plots: the main way scientists understand and communicate results
- Compute SST anomalies & plotting
- Plotting SST time series

5.2 – Plot Re-creation: SST and Anomalies

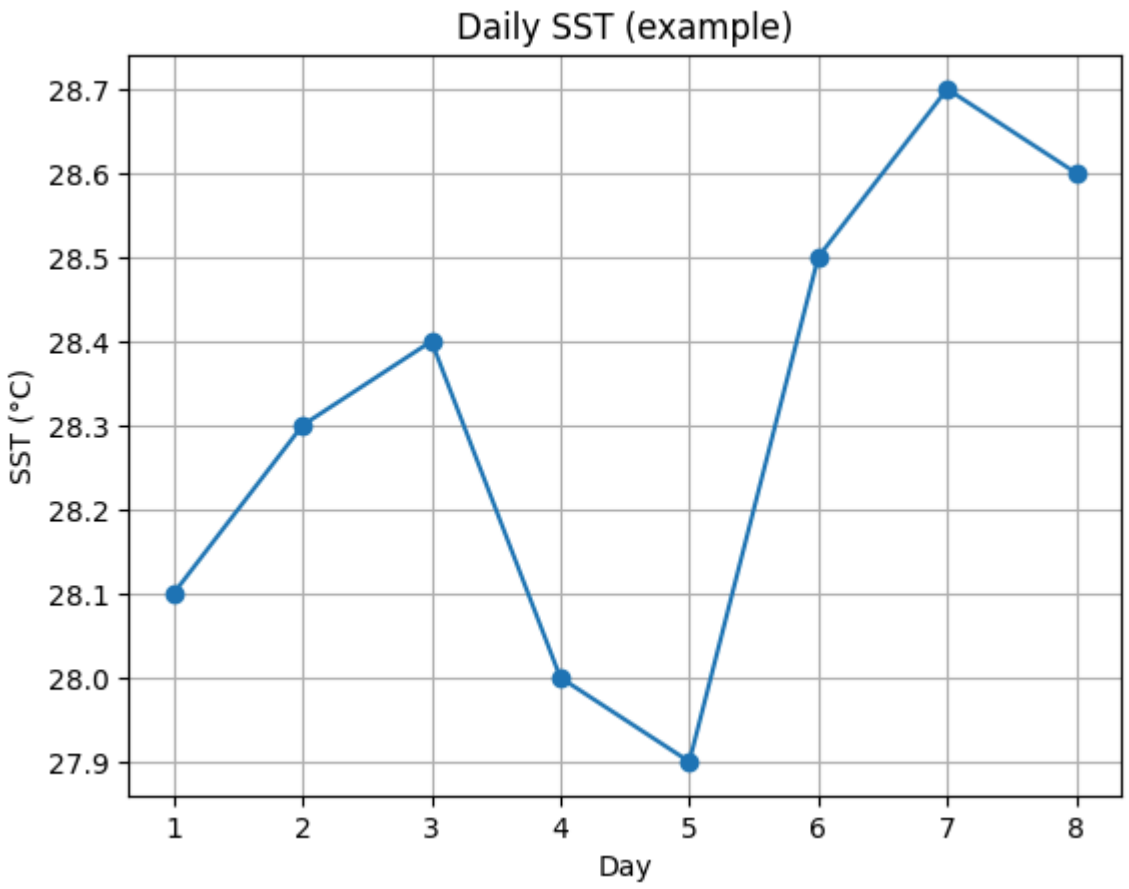
Below, reproduce **two plots**:

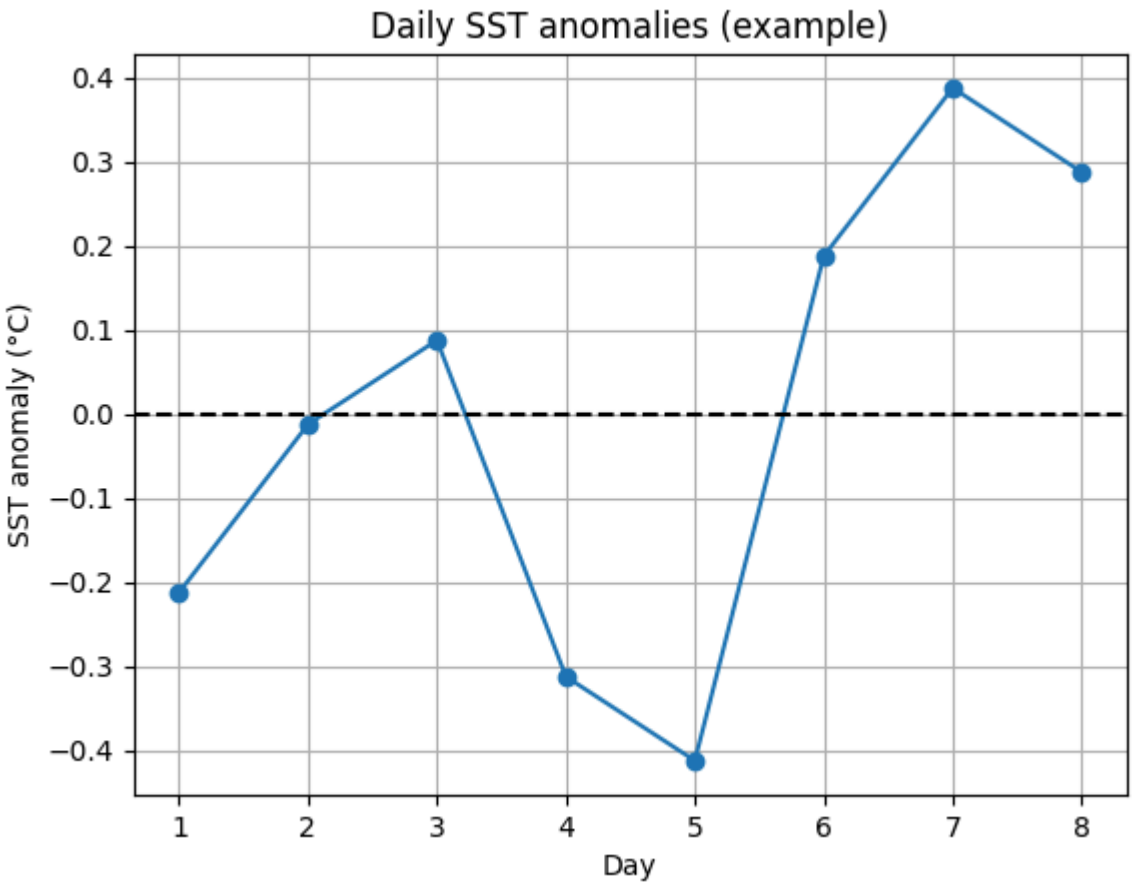
1. SST vs. day
2. SST anomaly vs. day (with a horizontal zero line)

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?

Plotting helped in clearly visualizing how sea surface temperature (SST) changes over time and space, which is difficult to understand from raw numerical data alone. Time-series plots made it easier to identify seasonal cycles, long-term trends, and sudden temperature variations. Spatial maps highlighted regions of warmer and cooler waters, revealing patterns such as coastal influences or large-scale ocean features. Anomaly plots, in particular, helped show deviations from the long-term average, making unusual warming or cooling events stand out clearly. Overall, plotting transformed complex SST data into intuitive visual patterns, improving interpretation of ocean variability and climate-related signals.

---

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.

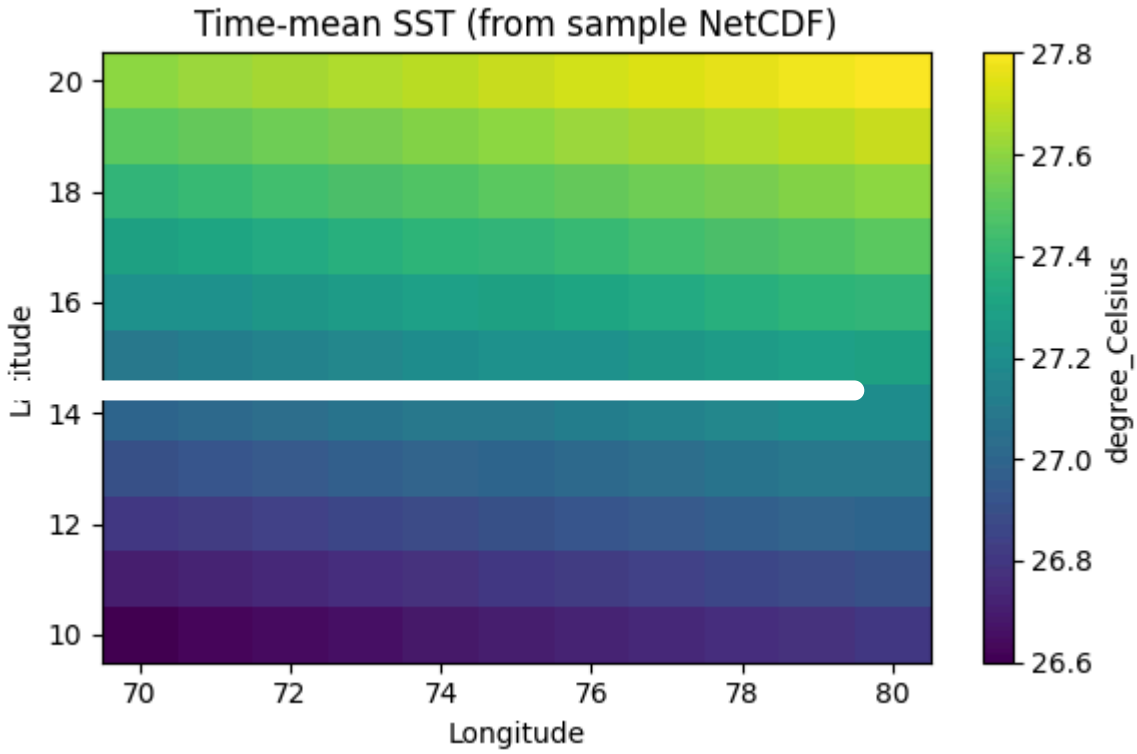
6.2 – Recreate Time-Mean SST Map

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-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()
```

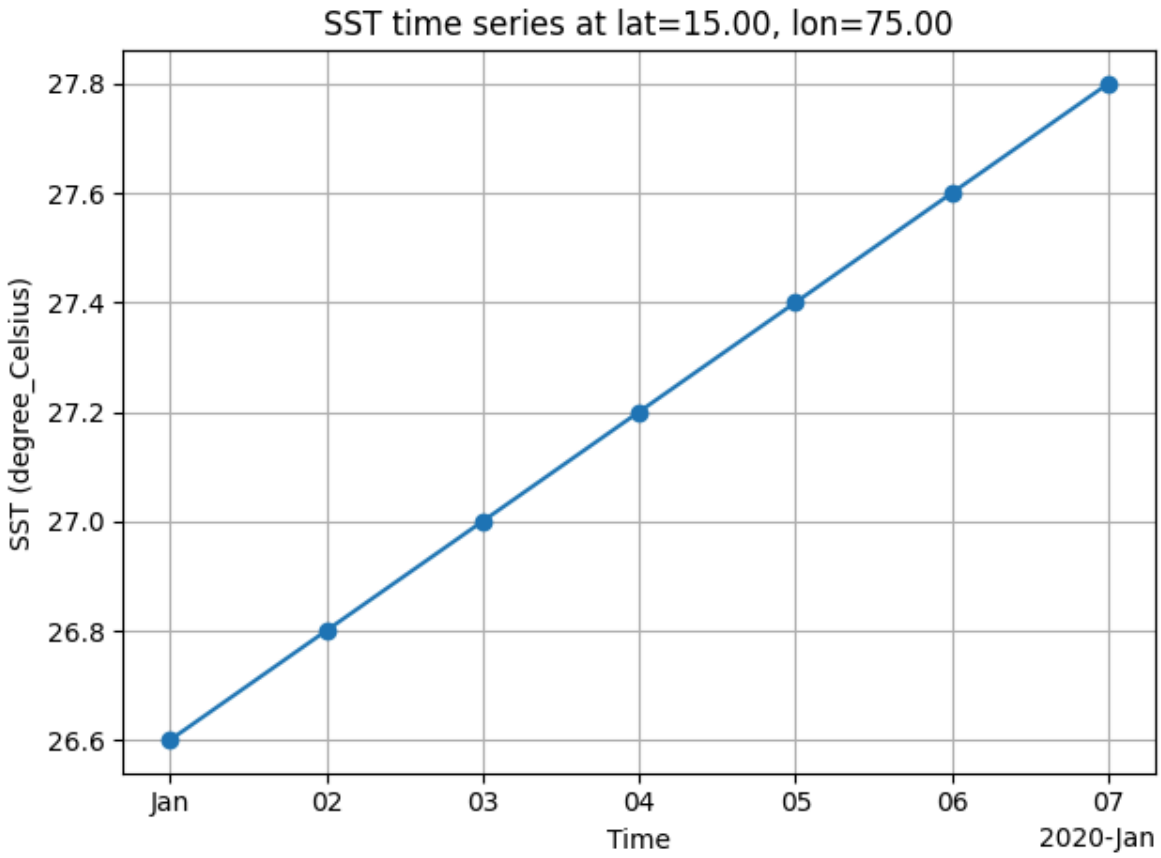


6.3 – Recreate SST Time Series at a Point

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

I learned that NetCDF is a widely used, self-describing data format designed for storing large, multidimensional scientific datasets such as climate, ocean, and atmospheric data. It efficiently organizes variables along dimensions like time, latitude, longitude, and depth, while also storing metadata that describes units, coordinates, and data sources.

I learned how xarray simplifies working with NetCDF data by representing datasets as labeled arrays instead of plain numerical matrices. Using xarray, it becomes easier to

select subsets of data, handle dimensions and coordinates, compute statistics, and perform time and spatial operations in a readable and efficient way.

I learned the basic workflow of remote sensing data analysis in Python, including reading satellite data files, inspecting metadata, cleaning or masking invalid values, and extracting variables of interest.

---

7. Capstone Assignment (Summary)

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

The capstone assignment integrates all majr skills from notebook 1-4, Python fundamentals,lists,loops,conditions,functions,plotting,Numpy,Xarray,NetCDF.

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

The overall flow of the capstone assignment is firstly,Setup that is required(like import numpy as np) next, Python fundamentals next, Basic Statistics Next, Lists,Loops & conditions Next, Functions, arrays, plotting Next, NetNDF + xarray: Remote Sensing dataset Next, Capstone scientific Interpretation.

It connected python skills to a realistic remote sensing / earth system problem with the NetCDF datasets, by extracting the datasets uning python code and by plotting we can understand the environmental process.

Most interesting i found covert to NumPy + plot section and dataset exploration

I would like to explore Map plotting using NumPy,Matplotlib,xarray.

---

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

My comfort with Python has increased significantly over the course, especially in using it as a tool for scientific data analysis rather than just basic programming.

I now feel confident working with lists, loops, and conditional statements to process data efficiently, as well as writing functions to organize and reuse code.

I still need more practice with advanced xarray operations, handling very large datasets efficiently, and performing more complex spatial and temporal analysis.

In the future, I plan to use these Python skills in coursework and research involving oceanic, atmospheric, and climate data analysis.

I am particularly interested in applying Python to real satellite and reanalysis datasets to study SST variability, aerosols, and climate trends.

---

9. Declaration & Signature

I, **(write your name here)**, declare that this report is based on my own work and understanding