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): 22 December 2025

Name: Ansif Mohammed K P

Student ID / Roll No.: 202501822

Programme: Msc Remote sensing and GIS

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

Semester / Academic Year: 2025

Instructor: Dr. ***Arjun Adhikari***________

(If needed) Date of Report Completion (manual override): 17/12/2025

---

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.

The Python component taught how to use Python and scientific libraries to read, analyze, and visualize environmental data, skills that are essential for working with satellite observations, ocean and atmospheric measurements, and climate model outputs.

---

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.).

- Types of data
- Different type of variables
- Basic Arithmetic in Python
- Mean, Median, and Mode
- Correlation

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

# TODO: create list and compute statistics

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 is basis and you feel comforable

The easiest part of basic Python was understanding simple syntax like variables, arithmetic, and basic plotting, while writing loops, functions, and debugging errors required more practice to feel comfortable.

---

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 store multiple values in a single variable and can hold numbers, strings, or mixed data types.

List elements are accessed using indexing, starting from zero.

Loops allow the same operation to be repeated over all elements in a list.

if–elif–else statements handle multiple conditions in a clear, structured way.

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
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, "WARM")
if temp > threshold:
      print("day", day, "NORMAL")
```

```
day 1 WARM
day 2 WARM
day 3 WARM
day 4 WARM
day 5 WARM
day 7 HOT
day 8 HOT
day 8 NORMAL
```

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 you to efficiently process large datasets, automate repetitive calculations, and apply different actions based on data values, which is essential when working with complex environmental, ocean, atmospheric, or climate data.

---

5. Notebook 03 – Functions and Plotting (Summary)

5.1 – Key Concepts Learned

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

Using functions helps reduce errors and improves readability when working with complex workflows.

NumPy arrays efficiently store and operate on large numerical datasets common in scientific analysis.

NumPy enables fast mathematical operations on entire arrays without explicit loops.

Plotting (e.g., with Matplotlib) is essential for visualizing patterns, trends, and relationships in data.

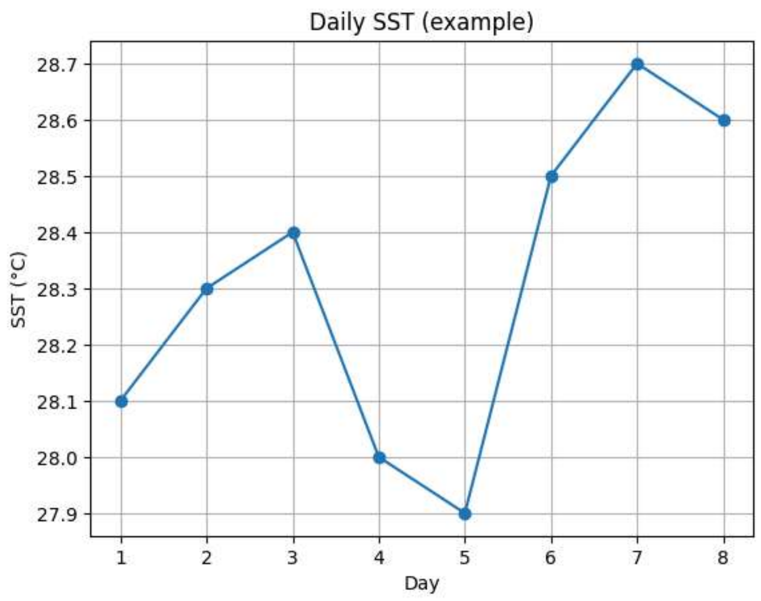
5.2 – Plot Re-creation: SST and Anomalies

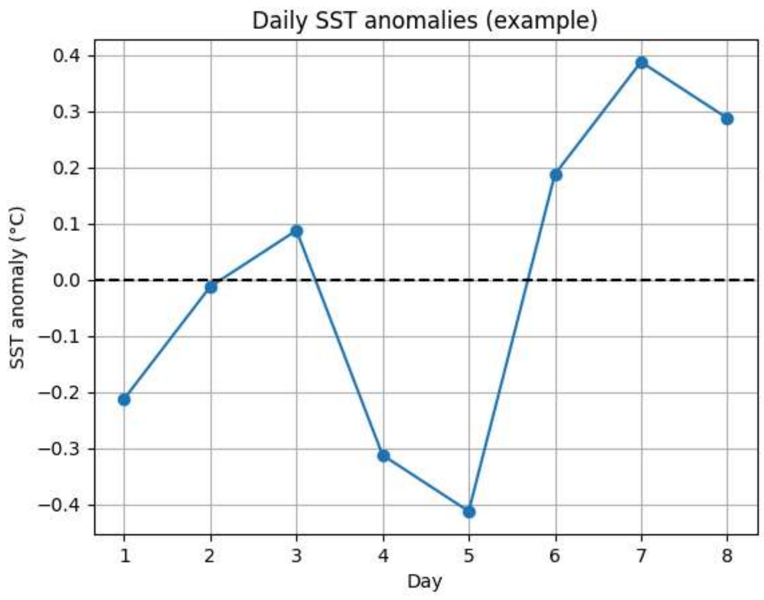
Below, reproduce **two plots**:

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 made it easy to see day-to-day SST changes and quickly identify positive and negative anomalies relative to the mean, which is harder to understand from numbers alone.

---

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:

Variable name(s): sst

Dimensions: time, lat, lon

Units: degree_Celsius (°C)

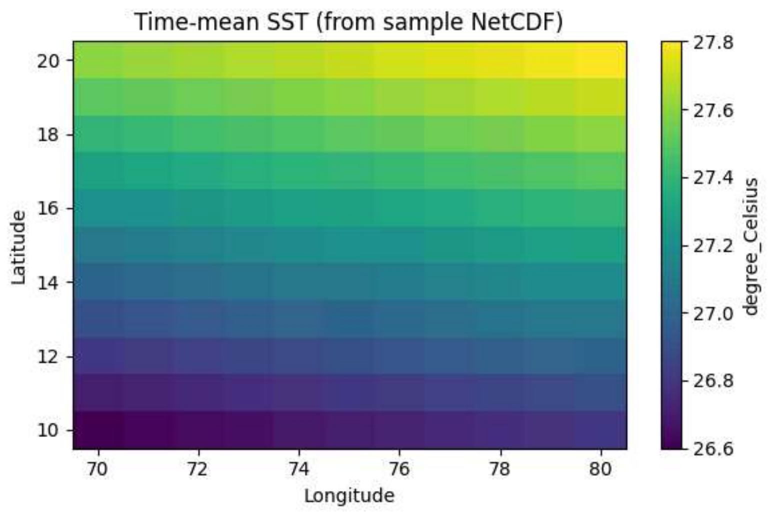
Approximate region covered: Latitude 10°N to 20°N and Longitude 70°E to 80°E (tropical Indian Ocean region)

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/main/data/data_sst.nc"
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()
```

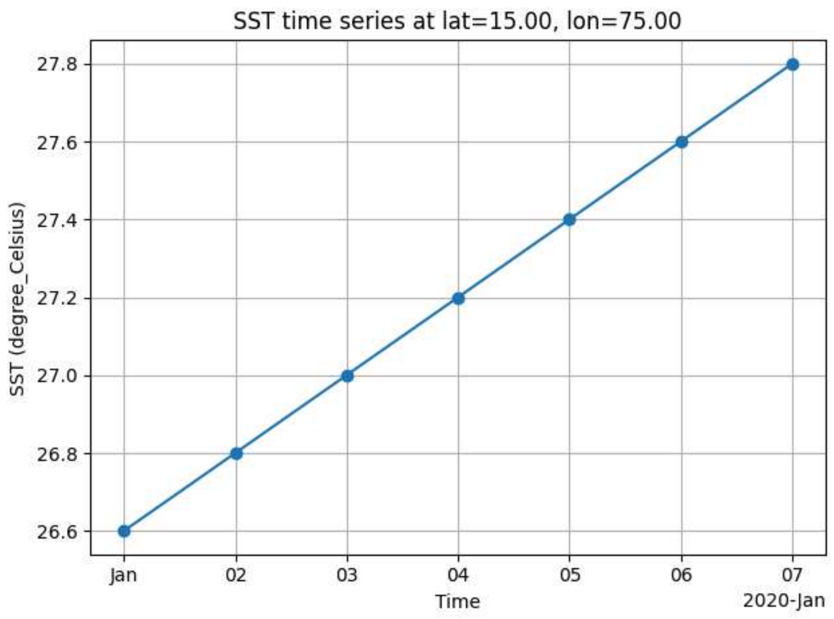


6.3 – Recreate SST Time Series at a Point

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.*

NetCDF as a format: NetCDF is a standard scientific data format designed to store large, multidimensional geophysical datasets along with detailed metadata, reanalysis, and model data.

xarray usage: xarray allows easy opening, exploring of NetCDF data using labeled dimensions (time, latitude, longitude), which simplifies analysis and reduces errors.

Basic remote sensing data handling in Python: I learned how to load gridded data, compute basic statistics, extract time series at specific locations, and visualize spatial maps and temporal variations using NumPy and Matplotlib.

---

7. Capstone Assignment (Summary)

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

I applied Python-based workflows to analyze geophysical data by integrating concepts from remote sensing, atmospheric science, and data analysis. The assignment helped me understand how satellite and model data are structured, how to extract meaningful information from NetCDF files, and how these methods can be extended to real-world datasets such as SST, aerosols, or reanalysis products.

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 followed a clear workflow from data loading and inspection to analysis and visualization. It connected Python skills with real remote sensing data by showing how NetCDF datasets are handled in Earth system studies. I found the visualization of spatial maps and time series most interesting. I would like to explore real satellite datasets and more advanced analysis in the future.

---

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.