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}")
```

Name: Jayant Dabhale

Student ID / Roll No.: 202200723

Programme: MSC remote sensing and GIS

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

Semester / Academic Year:

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		
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 course covered the basics of python including how to use 'print', basics expressions, how to create and manipulate variables. It explaines basic data types i.e. int,float,str,bool. It also includes meaning of errors shown in python and also helps how to fix those errors. It includes small earth,atmos,ocean related calculations which help us understand spatial patterns of the earth,atmos,ocean. It also covered basic math and simple statistics i.e. mean, median and mode which are used in plotting graphs and interpreting them.

Write your summary here (5–8 sentences).

---

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.).

- Create and manipulate variables
- Understand basic data types (int, float, str, bool)
- See and interpret a simple error
- Do small Earth/Atmos/Ocean-related calculations
- Practice basic math and simple statistics (mean, median, 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 part of basic Python felt easiest and what part required practice?

Easiest- 'Print' Basic math- addition, subtraction, division. Plotting Statistics and conversions of data types needs to be practiced.

---

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.

bold text-

- list
- loops
- conditions
-

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
import statistics as stats
```

```
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

# Write your classification code
    if temp > threshold:
        print("Day", day, ": HOT")
    elif temp <= threshold:
        print("Day", day, ": NORMAL")
```

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

4.3 – Reflection (3–4 sentences)

Why are loops and conditions important in scientific data analysis?

Loops allow you to apply the same calculation or operation to thousands of data points instantly. Conditions (If/Else statements) allow the code to "decide" how to treat specific data points based on set criteria.

---

5. Notebook 03 – Functions and Plotting (Summary)

5.1 – Key Concepts Learned

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

-
-
-
-
-

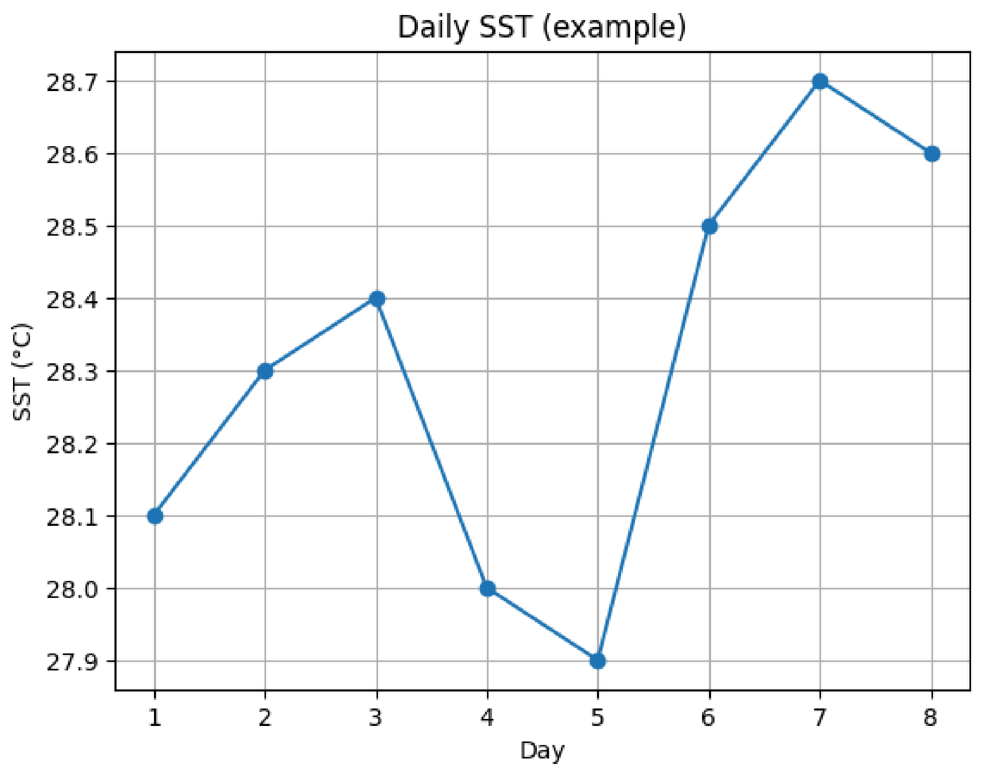
5.2 – Plot Re-creation: SST and Anomalies

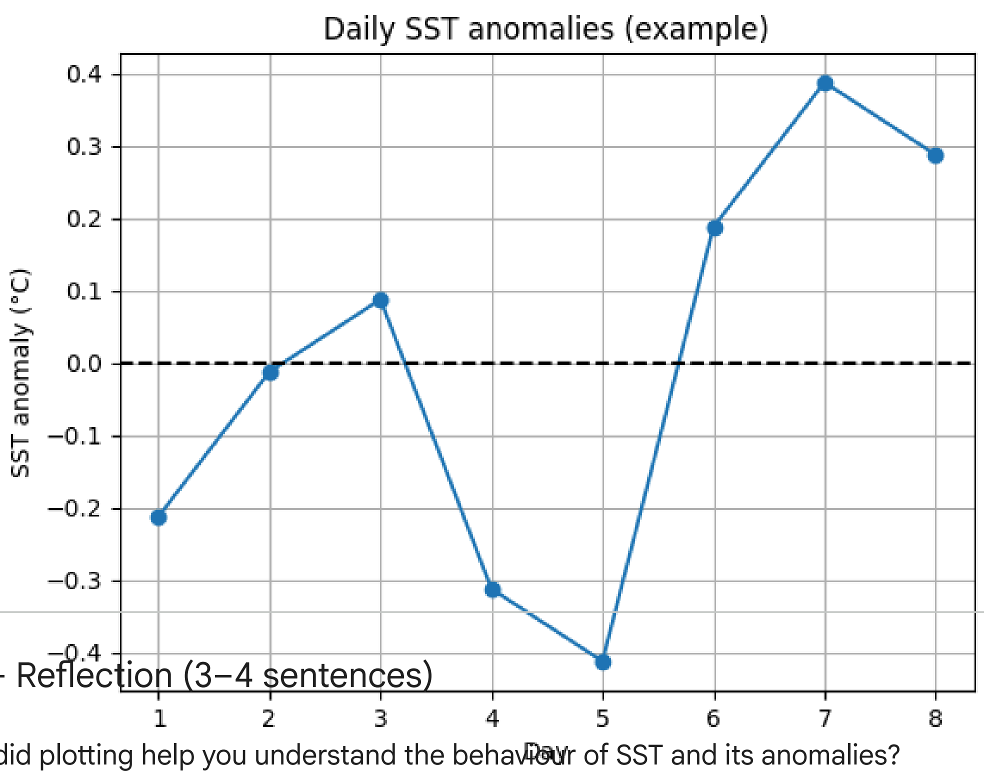
Below, reproduce **two plots**:

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 helps us understand long term warming or cooling trends, a line graph shows if the overall temperature is rising or falling.

---

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:

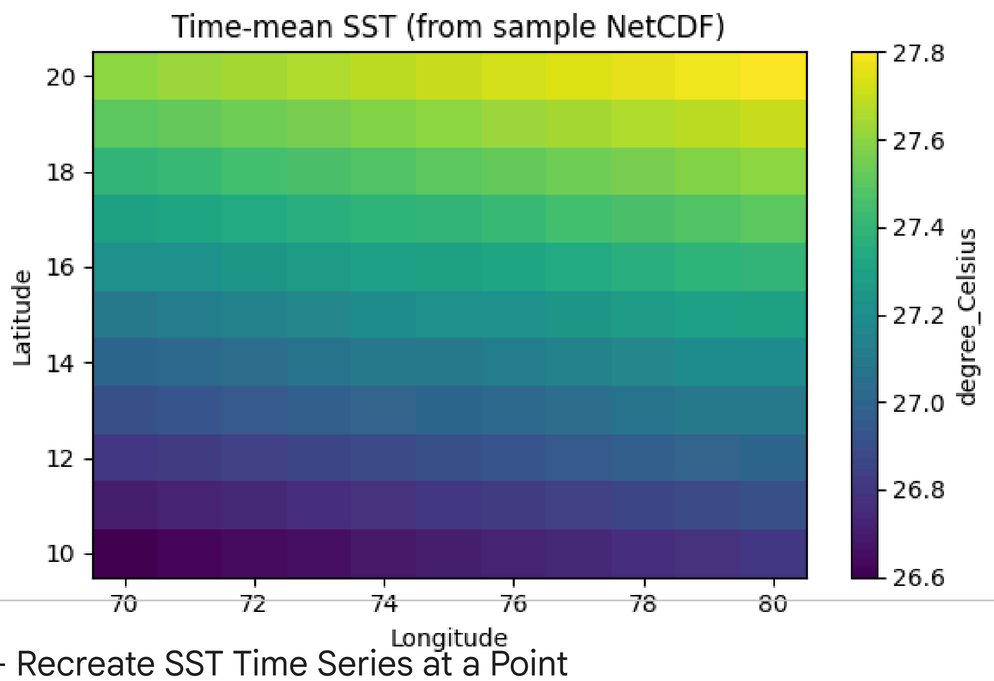
Fill in the details here based on `ds` and `ds.sst` attributes.

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.

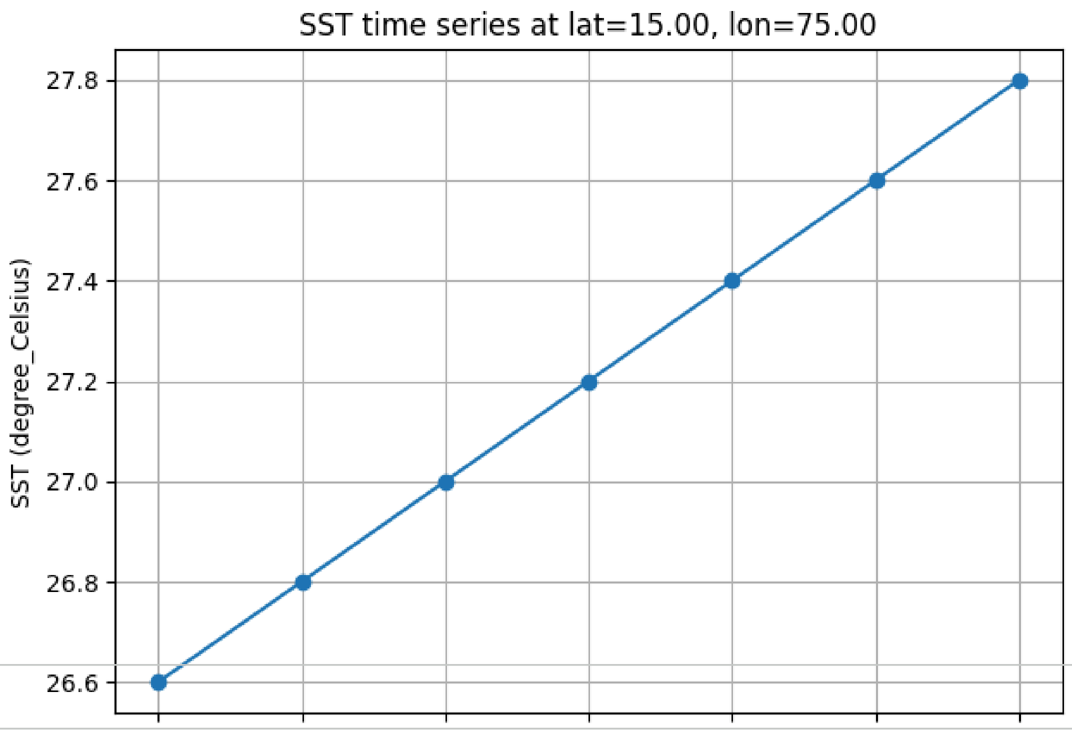
```
import os, requests
import xarray as xr
import matplotlib.pyplot as plt

# Download NetCDF file if not present
url = "https://raw.githubusercontent.com/EarthSystem-Science-Lab/python-basics/main/data/da
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()
```

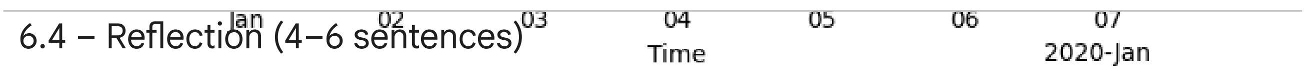


6.3 – Recreate SST Time Series at a Point

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

Netcdf file contains all the data including metadata (units, coordinates,etc.) and the dimensions. Xarray is the python library that brings the power of labels to raw arrays, it also aligns dates. Handling satellite data in Python focuses on converting raw pixels into physical values, it stores large data files without filling up the computer's storage.

---

7. Capstone Assignment (Summary)

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

The Capstone Assignment is kind of a test which includes exercises wherein we have to use all the basic functions of python that we learnt in the 1st-4th notebooks. It includes basic mathematics, statistcs, plotting, loops, arithmatic deviations, libraries of python etc.

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

Python skills are essential in monitoring overall earth's health.Using python libraries we can get environmental variables (sst, AOD,) which are very essential for montoring hazardous events. Plotting

Plotting

Python libraries

---

8. Final Reflection on Python & Remote Sensing

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