# **1 Month 1 Week:**

Python script and a summary of the concepts learned (on Google Classroom).

# **Python Script: Average Temperature Analyzer**

# Python Program to Analyze Temperatures

# Input: Collect temperatures for 7 days

temperatures = []

for i in range(1, 8):

temp = float(input(f"Enter temperature for day {i} in °C: "))

temperatures.append(temp)

# Process: Calculate total, average, min, max

total = sum(temperatures)

average = total / len(temperatures)

minimum = min(temperatures)

maximum = max(temperatures)

# Output the results

print("\nTemperature Report")

print("Temperatures Recorded:", temperatures)

print("Total Temperature:", total)

print("Average Temperature:", round(average, 2), "°C")

print("Lowest Temperature:", minimum, "°C")

print("Highest Temperature:", maximum, "°C")

# **Summary of Concepts Learned**

**Python Concepts Covered**

| **Concept** | **What Was Used** |
| --- | --- |
| Variables | temperatures, total, average, minimum, maximum |
| Data Types | float, list |
| Input/Output | input(), print() |
| Loops | for loop to gather daily temperatures |
| Operators | Arithmetic (+, /), Comparison (min, max) |
| Built-in Functions | sum(), len(), min(), max(), round() |

**What I Learned**

* How to take numeric input from users with input() and convert it using float().
* How to store multiple values in a list.
* How to repeat data collection using a for loop.
* How to process data using built-in functions like sum(), min(), max().
* How to calculate averages and summarize results.
* How to display clean output using print() and round().

# **2 Week:**

## **Summary of Concepts**

* **Lists** → Store multiple values (stock returns, blood pressure readings).
* **Dictionaries** → Store key-value pairs (student names & marks).
* **Functions** → Perform transformations (sum of squares, filtering, adding grace marks).
* **Data Transformation** → Cleaning, filtering, aggregating, or restructuring data to get useful insights.

## **Explanation Step by Step**

1. **Raw Data Issues**
   * Duplicate customer (David with ID 104 appears twice).
   * Missing Age for Charlie.
   * Negative Age for Eve.
   * Unrealistic purchase amount (₹10,00,000).
2. **Remove Duplicates**
   * Used drop\_duplicates() to keep only one entry for each customer.
3. **Handle Missing Values**
   * Replaced missing age with **mean of other ages** using .fillna().
   * Alternatives: median, mode, or dropping missing rows.
4. **Filter Invalid Data**
   * Removed rows with negative ages using a condition filter.
5. **Handle Outliers**
   * Removed purchases greater than ₹1,00,000 as they were unrealistic for this dataset.

## **Real World Applications of Data Cleaning**

* **Healthcare** → Removing duplicate patient records, fixing missing lab results.
* **Banking/Finance** → Removing invalid transactions, correcting missing account info.
* **E-commerce** → Handling duplicate product listings, filtering unrealistic prices.
* **Survey Data** → Cleaning incomplete or wrongly entered responses.

## **Summary (Google Classroom Note Style)**

**Python Concepts Covered**

* DataFrames with Pandas
* drop\_duplicates() for duplicates
* fillna() for missing values
* Filtering with conditions (df[df['Age'] > 0])
* Outlier removal

**What I Learned**

* Real-world datasets are **messy** and must be cleaned before analysis.
* Python’s **Pandas library** makes data cleaning simple and powerful.
* Each step (duplicates → missing values → invalid data → outliers) ensures better **data quality**.

# **Python Script: Data Cleaning (Client Project)**

import pandas as pd

# -------------------------------

# Step 1: Create sample dataset

# -------------------------------

data = {

'CustomerID': [101, 102, 103, 104, 104, 105],

'Name': ['Alice', 'Bob', 'Charlie', 'David', 'David', 'Eve'],

'Age': [25, 30, None, 45, 45, -5],

'PurchaseAmount': [2500, 3000, 4000, 5000, 5000, 1000000]

}

df = pd.DataFrame(data)

print("Raw Data:\n", df)

# -------------------------------

# Step 2: Remove duplicates

# -------------------------------

df = df.drop\_duplicates()

print("\nAfter Removing Duplicates:\n", df)

# -------------------------------

# Step 3: Handle missing values

# -------------------------------

df['Age'] = df['Age'].fillna(df['Age'].mean()) # Replace missing with mean

print("\nAfter Handling Missing Values:\n", df)

# -------------------------------

# Step 4: Remove invalid data

# -------------------------------

df = df[df['Age'] > 0] # Remove negative ages

print("\nAfter Removing Invalid Ages:\n", df)

# -------------------------------

# Step 5: Handle outliers

# -------------------------------

df = df[df['PurchaseAmount'] < 100000] # Remove unrealistic purchase

print("\nAfter Removing Outliers:\n", df)

# **Summary of Concepts Learned**

**Python Concepts Covered**

| **Concept** | **What Was Used** |
| --- | --- |
| Data Structures | Pandas DataFrame to store tabular data |
| Remove Duplicates | drop\_duplicates() |
| Handle Missing Data | fillna() with mean |
| Filtering Data | Conditions like df[df['Age'] > 0] |
| Outlier Removal | Filtering purchase amount < 100000 |

**What I Learned**

* How to clean **real-world messy data** step by step.
* How to remove **duplicates** to avoid double-counting records.
* How to handle **missing values** using averages.
* How to filter out **invalid data** (e.g., negative ages).
* How to detect and remove **outliers** to keep data realistic.
* Data cleaning is an **essential step** before any analytics or machine learning.

# 

# **3 Week:**

**Perform operations with NumPy and manipulate datasets with Pandas.**

| **Library** | **Key Functions** | **Used For** |
| --- | --- | --- |
| **NumPy** | mean(), array, arithmetic ops | Fast numeric operations on arrays |
| **Pandas** | DataFrame, groupby(), filter(), cut() | Data manipulation & analysis |
| **Both Together** | sum(), round(), normalize, merge | Real-world data processing pipelines |

**Clean and aggregate a dataset (e.g., remove missing values, calculate averages).**

**Python Concepts Used**

| **Concept** | **What Was Used** |
| --- | --- |
| Pandas DataFrame | pd.DataFrame() |
| Data Cleaning | dropna(), drop\_duplicates(), filtering |
| Aggregation | groupby() + mean() |
| Output Formatting | reset\_index(), print() |

**What I Learned**

* Real-world data is often **incomplete or messy**, requiring cleaning before analysis.
* Using **Pandas**, we can easily clean data by removing duplicates and null values.
* Aggregation helps us summarize data — e.g., calculating **average price, total revenue**, etc.
* Clean, structured data improves the quality of business insights.

# **4 Week:**

**Create visualizations for dataset analysis.**

## Summary: When to Use What

| Visualization | Library | Use Case |
| --- | --- | --- |
| Line Chart | Matplotlib | Time trends (sales, traffic) |
| Bar Plot | Seaborn | Compare categories (city-wise, product-wise) |

**Create a dashboard for visualizing relationships between features in a dataset (e.g., scatter plots, histograms).**

## Output Visuals:

1. Scatter Plot → Reveals how income affects spending, separated by gender
2. Histogram → Shows age distribution, useful for segmenting customers
3. Heatmap → Highlights which features are strongly correlated (e.g., income & spending)

## **Summary of Concepts Learned**

## Python Visualization Tools Used

| **Tool** | **Purpose** |
| --- | --- |
| Seaborn | Scatter plots, histograms, heatmaps |
| Matplotlib | Plot display customization |
| Pandas | Data wrangling and filtering |

🧠 **What I Learned**

* How to use **scatter plots** to analyze relationships between two features
* How to visualize **distribution of values** using histograms
* How to detect **correlations** between multiple variables using heatmaps
* Building dashboards gives **quick insights for business decisions**