## **🗓️ Week 1: Python Core + NumPy**

Goal: Get comfortable with Python syntax, data types, control flow, and NumPy for efficient numerical computation.

### **📅 Day 1: Python Basics**

* Variables, Data Types (int, float, string, bool)
* Lists, Tuples, Dictionaries, Sets
* String manipulations
* 🧠 *Mini Project*: Parse and clean a CSV-like string input

🔹 **Exercise 1**:  
 Write a Python script to:

* Parse a string like "name:John, age:30, role:engineer" into a dictionary.
* Convert list of such strings into a list of dictionaries.

🔹 **Exercise 2**:  
 Create a function to extract domains from a list of emails.

### **📅 Day 2: Control Flow + Functions**

* if-else, for, while
* Writing functions, lambda functions
* List comprehensions
* 🧠 *Practice*: Write a function to clean and transform tabular data

🔹 **Exercise 3**:  
 Build a function clean\_data(data\_list) that:

* Removes empty strings.
* Trims whitespace.
* Converts numeric strings to integers.

🔹 **Exercise 4**:  
 Write a custom implementation of map() to transform a list of temperatures from Celsius to Fahrenheit.

### **📅 Day 3: File Handling + Error Handling**

* Reading/writing .txt, .csv, .json
* Exception handling with try/except
* Pathlib and OS module basics
* 🧠 *Exercise*: Process a log file and summarize entries

🔹 **Exercise 5**:

* Read a log file (text).
* Count total errors ("ERROR") and warnings ("WARNING").
* Save the summary in JSON format.

🔹 **Exercise 6**:  
 Try reading a non-existent file and catch the error gracefully.

### **📅 Day 4: NumPy Essentials**

* Creating arrays, array shapes, slicing
* Element-wise operations
* Indexing, filtering, masking
* Broadcasting
* 🧠 *Practice*: Simulate weather data and do basic computations

🔹 **Exercise 7**:  
 Generate a 2D NumPy array with random integers (10x5) representing temperature readings.

🔹 **Exercise 8**:

* Replace all values below 0 with 0.
* Count how many entries are above 30.

### **📅 Day 5: NumPy for Aggregation and Transformation**

* np.mean, np.sum, np.std
* Aggregating over axes
* Boolean indexing, conditional replacement
* 🧠 *Mini Project*: Build a metrics dashboard using simulated data

🔹 **Exercise 9**:  
 Given a 2D NumPy array of daily sensor data (rows: days, cols: sensors):

* Compute the average per sensor.
* Find the day with maximum total reading.

### **📅 Day 6: Practical Use of NumPy in Data Engineering**

* Converting CSV to NumPy
* Memory-efficient transformations
* Simulating time series data
* 🧠 *Practice*: Create a data generator that outputs daily sensor data

🔹 **Exercise 10**:  
 Simulate one week of hourly power consumption data:

* 7x24 matrix.
* Calculate total, average, and variance per day.

### **📅 Day 7: Weekly Review + Mini Project**

* Revise NumPy and core Python
* 🧠 *Project*: Simulate a basic ETL pipeline for IoT data using NumPy

🔹 **Mini Project**:

Simulate and process IoT sensor data (temperature, humidity, motion):

* Generate 1000 entries with timestamps using NumPy.
* Filter entries where temperature > 35 and motion detected.
* Save to CSV.

## **🗓️ Week 2: Pandas + Data Engineering Concepts**

Goal: Master data wrangling and transformation with Pandas, and relate it to real data engineering tasks.

### **📅 Day 8: Pandas Fundamentals**

* Series and DataFrame creation
* Reading CSV, JSON, Excel
* Inspecting data (head(), info(), describe())
* 🧠 *Exercise*: Load a sample dataset and explore its structure

🔹 **Exercise 11**:

* Load a sample CSV (e.g., sales.csv).
* Print column types, shape, top 5 rows.
* Rename one column.

### **📅 Day 9: Data Cleaning and Filtering**

* Handling missing values (dropna, fillna)
* Filtering rows, conditional operations
* Data type conversion
* 🧠 *Practice*: Clean and prepare a dataset for analysis

🔹 **Exercise 12**:  
 Given a dataset:

* Replace "N/A" and "?" with np.nan.
* Drop columns with >50% nulls.
* Fill remaining nulls with mean (for numeric) or mode (for categorical).

### **📅 Day 10: Data Transformation**

* apply, map, replace
* Creating new columns
* Renaming, reindexing
* 🧠 *Exercise*: Normalize numeric columns and create derived metrics
* 🔹 **Exercise 13**:  
   Add a revenue column = price \* quantity.  
   Map product categories to short codes ("Beverages" → "BEV").
* 🔹 **Exercise 14**:  
   Write a function using apply() to classify transactions as "high", "medium", "low" based on revenue thresholds.

### **📅 Day 11: GroupBy and Aggregations**

* groupby(), agg(), pivot tables
* Multi-level indexes
* Rolling statistics
* 🧠 *Mini Project*: Generate daily, weekly, monthly sales summaries

🔹 **Exercise 15**:

* Group data by region and category.
* Calculate total sales and average order value.
* Sort regions by total revenue.

🔹 **Exercise 16**:  
 Use rolling averages to smooth out daily sales data over 7-day windows.

### **📅 Day 12: Joins and Merging**

* merge(), join(), concat()
* Handling mismatched data
* Combining multiple sources
* 🧠 *Exercise*: Merge weather and sales data to prepare for analysis

🔹 **Exercise 17**:  
 Merge two DataFrames:

* sales.csv: Contains store\_id, date, sales
* weather.csv: Contains store\_id, date, temperature

Join on store\_id and date. Then analyze how temperature affects sales.

### **📅 Day 13: Time Series and Exporting Data**

* datetime handling, resampling
* Sorting, slicing by date
* Exporting to CSV, Parquet
* 🧠 *Practice*: Analyze and export monthly trends

🔹 **Exercise 18**:

* Parse date column as datetime.
* Resample data to weekly aggregates.
* Calculate % change week over week.

### **📅 Day 14: Final Capstone Project**

* 🧠 *Capstone*: Build a mini ETL pipeline that:  
  + Reads raw CSV files (e.g., sales + weather)
  + Cleans and transforms data using Pandas
  + Merges and aggregates data
  + Exports cleaned results to multiple formats (CSV, JSON, Parquet)
  + (Bonus) Use NumPy to simulate additional data input

🔹 **Capstone**:

Create a full data pipeline using Pandas:

1. Read multiple CSVs (e.g., stores, sales, weather).
2. Clean and join them.
3. Perform groupby aggregations and rolling statistics.
4. Export:  
   * Clean dataset to CSV
   * Summary statistics to JSON
   * Optimized dataset to Parquet

## **🛠️ Tools & Resources**

* **Jupyter Notebook / Google Colab**
* **Datasets**: Use open datasets like:  
  + [Kaggle](https://www.kaggle.com/)
  + [UCI ML Repo](https://archive.ics.uci.edu/ml/index.php)
  + data.gov
* **Practice Sites**:  
  + LeetCode - Python
  + W3Schools Python
  + [RealPython](https://realpython.com/)

# **🧠 7-Day OOP Mastery Plan in Python**

## **🔧 Day 1: OOP Fundamentals**

### **📘 Topics:**

* What is OOP? Why use it?
* Class and Object
* Attributes and Methods
* The \_\_init\_\_() method
* The self keyword

### **🧪 Exercises:**

* Define a Truck class with attributes: location, load\_capacity, driver.
* Create multiple instances and print their properties.
* Add a method is\_overloaded(current\_weight).

## **🔐 Day 2: Encapsulation & Getters/Setters**

### **📘 Topics:**

* Public vs. Private Attributes (\_ and \_\_)
* Getter and Setter methods
* @property decorator

### **🧪 Exercises:**

* Add \_fuel\_level as a private attribute to Truck.
* Create getter/setter using @property.
* Prevent setting fuel above 100.

## **🧱 Day 3: Class Methods, Static Methods, Dunder Methods**

### **📘 Topics:**

* @classmethod vs @staticmethod
* Common dunder methods: \_\_str\_\_, \_\_repr\_\_, \_\_eq\_\_, \_\_len\_\_

### **🧪 Exercises:**

* Add a class variable to count total trucks.
* Add @classmethod to get the count.
* Implement \_\_str\_\_() for pretty print.
* Create TruckFleet class, and define \_\_len\_\_ to return number of trucks.

## **🔁 Day 4: Inheritance and Method Overriding**

### **📘 Topics:**

* Inheritance syntax
* Method Overriding
* Using super()

### **🧪 Exercises:**

* Create a base class Vehicle, with common attributes like plate\_no, fuel\_type.
* Inherit Truck and Van from Vehicle.
* Override the is\_overloaded() method for each type.

## **🔄 Day 5: Polymorphism and Abstraction**

### **📘 Topics:**

* Polymorphism through method overriding
* Abstract base classes (abc module)
* Interfaces in Python

### **🧪 Exercises:**

* Create abstract class DataProcessor with method process().
* Inherit it in CSVProcessor, JSONProcessor with custom process() logic.
* Use a loop to call process() on different processor objects.

## **🔗 Day 6: Composition, Aggregation, and Real-World Modeling**

### **📘 Topics:**

* "Has-A" relationship (Composition)
* Modeling real systems using objects
* Managing object interactions

### **🧪 Exercises:**

* Create a class Driver with name and license attributes.
* Use composition: A Truck has a Driver.
* Create a FleetManager that aggregates multiple Truck objects and prints a dispatch list.

## **🏁 Day 7: Capstone Project – OOP for a Mini Data Pipeline**

### **🎯 Project Overview:**

Build a modular pipeline using OOP for processing data files and logging results.

### **🔧 Classes to Create:**

1. DataLoader – load CSV/JSON files.
2. DataCleaner – drop nulls, fix types.
3. DataTransformer – add calculated fields.
4. Logger – logs events with timestamps.
5. PipelineManager – orchestrates the above classes.

### **💡 Bonus:**

* Add unit test support (unittest or pytest)
* Implement logging using built-in logging module
* Design it for extension: make it work for both sales and weather datasets

# **🧰 Tools & Resources**

| **Tool** | **Purpose** |
| --- | --- |
| abc | Abstract base classes |
| logging | Structured logging |
| dataclasses | Cleaner syntax for data containers |
| pydantic (optional) | Data validation |
| unittest / pytest | Test your classes and behavior |