**🧱 OVERVIEW**

**Goal:**  
Master Python for Data Engineering — file handling, APIs, data processing, automation, and integration with SQL, AWS, and Airflow.

**Outcome:**  
You’ll be able to write **data ingestion, transformation, and automation scripts** used in real-world ETL systems.

**🗓️ WEEK 1 — Practical Data Handling with Python**

**Goal:** Learn to handle files, directories, and data formats like CSV, JSON, and Parquet — core skills for any data wrangler.

**Topics**

* Working with file paths using os, pathlib
* Reading/writing large CSVs with pandas
* Handling JSON, XML, and YAML
* Basic exception handling in data I/O
* Logging progress and errors

**Hands-on Tasks**

1. Load a CSV of 1M+ rows and measure load time.
2. Convert CSV → JSON → Parquet.
3. Write a script that scans a folder and lists files by size/date.
4. Implement a logger that records all file conversions.

**Mini Project**

**“File Format Converter”**  
A CLI tool that takes an input folder of files (CSV/JSON) and outputs them all as cleaned Parquet files with logging.

**Outcome:**  
You can handle and automate real-world data file ingestion.

**🗓️ WEEK 2 — Data Cleaning & Transformation**

**Goal:** Learn to clean, reshape, and transform messy datasets with pandas and Python tools.

**Topics**

* Handling missing values, duplicates, outliers
* Data type conversions and parsing dates
* Vectorized operations in pandas
* Joining/merging datasets
* Aggregations, groupby, pivot tables
* Writing reusable functions

**Hands-on Tasks**

1. Load a “dirty” dataset (missing columns, weird dates).
2. Clean and standardize it with pandas.
3. Build a small cleaning function and apply it to multiple files.
4. Profile datasets (rows, cols, nulls, data types).

**Mini Project**

**“Data Cleaning Pipeline”**  
Automate cleaning for multiple CSVs — output cleaned and summarized data in a “processed” folder.

**Outcome:**  
You can write a Python ETL step to transform raw data into structured tables.

**🗓️ WEEK 3 — APIs, JSON & Web Data Extraction**

**Goal:** Learn to fetch and process data from REST APIs (a must for every modern data engineer).

**Topics**

* Using requests library (GET, POST, headers, params)
* Handling authentication (API keys, tokens)
* Parsing complex JSON responses
* Pagination and rate-limiting
* Saving API responses as JSON or CSV
* Error handling and retries

**Hands-on Tasks**

1. Connect to the **Open Exchange Rates API** or **GitHub API**.
2. Fetch data in intervals and store JSON responses.
3. Normalize nested JSON into pandas DataFrames.
4. Build retry logic with exponential backoff.

**Mini Project**

**“API Data Collector”**  
Fetch currency or weather data every hour, save as timestamped JSON files, and merge them into one CSV.

**Outcome:**  
You can build ingestion scripts that pull live data from any REST API.

**🗓️ WEEK 4 — Working with Databases (SQL + Python)**

**Goal:** Learn to connect Python with databases, query data, and load results efficiently.

**Topics**

* Connecting to SQL databases using:
  + sqlite3
  + sqlalchemy
  + psycopg2 or snowflake.connector
* Reading/writing pandas DataFrames to SQL tables
* Parameterized queries and transactions
* Bulk inserts and batch processing
* Using .env files and secrets for credentials

**Hands-on Tasks**

1. Create a SQLite DB and load cleaned CSVs into it.
2. Query and fetch results into pandas.
3. Transform and reload data back into another table.
4. Write a small wrapper for running SQL scripts.

**Mini Project**

**“Python + SQL Loader”**  
Automate: read processed CSVs → load to SQL DB → log inserted rows.

**Outcome:**  
You can connect Python to real databases and handle data I/O like a pro.

**🗓️ WEEK 5 — Automation, Scheduling, and Parallelism**

**Goal:** Turn your scripts into efficient, automated, and concurrent data pipelines.

**Topics**

* Scheduling scripts (cron / Airflow / schedule lib)
* File watchers with watchdog
* Using multiprocessing & concurrent.futures
* Caching results (joblib, functools.lru\_cache)
* Sending notifications (email, Slack webhook)

**Hands-on Tasks**

1. Write a script that monitors a folder and processes any new CSV file.
2. Parallelize data cleaning for 3–4 files simultaneously.
3. Automate daily execution using schedule.every().day.at("09:00").

**Mini Project**

**“Automated Data Processing Bot”**  
Script that watches a folder for new files → processes → saves cleaned data → sends a Slack/email update.

**Outcome:**  
You can automate and parallelize tasks just like Airflow does — but in pure Python.

**🗓️ WEEK 6 — Cloud, AWS & Integration**

**Goal:** Learn to integrate Python with AWS — where your data pipelines will actually run.

**Topics**

* Working with AWS SDK (boto3)
  + Upload/download S3 files
  + Manage S3 buckets
* Using AWS Secrets Manager for credentials
* Triggering Lambda functions or SNS alerts
* Integrating Python scripts with Airflow or EC2
* Packaging Python projects (virtualenv, requirements.txt)

**Hands-on Tasks**

1. Upload transformed data to S3.
2. Fetch data from S3 and load into Snowflake (or mock DB).
3. Store credentials securely using boto3 and Secrets Manager.
4. Run your script from EC2 (via cron or Airflow).

**Mini Project**

**“Mini Cloud ETL”**  
API → Python → Cleaned CSV → Load to S3 → Log to CloudWatch or local log.

**Outcome:**  
You can integrate Python scripts into a cloud-based data pipeline — production-ready and AWS-native.

**🔧 Optional Bonus After Week 6**

If you want to keep going:

* Week 7–8: Learn **PySpark** for large-scale transformations.
* Week 9: Integrate **dbt + Python** workflows.
* Week 10: Build a complete **ETL pipeline orchestrated by Airflow** (Python operators only).

**📘 Summary Table**

| **Week** | **Focus** | **Project** |
| --- | --- | --- |
| 1 | File & format handling | File Format Converter |
| 2 | Data cleaning & transformation | Cleaning Pipeline |
| 3 | APIs & web data | API Data Collector |
| 4 | Databases & SQL | SQL Loader |
| 5 | Automation & concurrency | Data Processing Bot |
| 6 | Cloud & AWS integration | Mini Cloud ETL |