API & AUTOMATION WITH PYTHON – 2025 EDITION

" Automate the boring, connect the world."

Market Demand Note

APIs power modern applications, and automation helps businesses save time, reduce errors, and scale operations. Python's simplicity makes it the go-to language for both API integrations and process automation. This program teaches how to connect systems, extract data, and automate repetitive tasks.

Duration: 6 weeks | **Mode**: Online/Offline

📘 APIs & Automation with Python — 2025 Edition

Table of Contents

Detailed Content

Week 1: Introduction to APIs & Python Basics

- **Understand APIs: What are they? Why are they important in software and business?**
- **REST vs. SOAP: Compare architectures, understand when to use each.**
- **Python refresher: Variables, loops, conditionals, functions, basic I/O.**
- **Environment setup: Python 3.x, Jupyter Notebook, Postman (API testing tool).**
- **Hands-on:**
    - **Write simple Python scripts.**
    - **Use Postman to test public APIs (e.g., https://jsonplaceholder.typicode.com/).**
    - **Discuss real-world API use cases (e.g., weather, payments, social media).**

Week 2: Consuming APIs

- **HTTP methods: GET, POST, PUT, DELETE.**
- **Python's requests library: Making API calls, handling responses.**
- **Parsing JSON: Extract and manipulate data from API responses.**
- **Error handling: Status codes, exceptions, retries.**
- **Hands-on:**
    - **Fetch data from a public API (e.g., OpenWeatherMap, GitHub).**
    - **Parse and display JSON data in Python.**
    - **Save API data to CSV or JSON files.**

Week 3: API Authentication & Real-World Integrations

- **Authentication types: API keys, OAuth, tokens.**
- **Securing API calls: Headers, environment variables.**
- **Integrate with real APIs:**
    - **Weather API (get current weather).**
    - **Payment Gateway API (simulate a payment flow).**
    - **Social Media API (post/retrieve tweets or statuses).**
- **Hands-on:**
    - **Get an API key, make authenticated requests.**
    - **Build a Python script that interacts with 2–3 different APIs.**
    - **Handle rate limits and authentication errors.**

Week 4: Building APIs with Python

- **Introduction to web frameworks: Flask, FastAPI.**
- **API endpoints: Define routes, handle GET/POST requests.**
- **Return JSON: Structure responses for clients.**
- **Testing APIs: Use Postman to test your own API.**
- **Hands-on:**
    - **Build a simple REST API with Flask or FastAPI.**
    - **Create endpoints for data CRUD (Create, Read, Update, Delete).**
    - **Test your API with Postman and curl.**

Week 5: Automation with Python

- **File handling: Read, write, move, delete files.**
- **Data cleaning: Filter, transform, aggregate data.**
- **Email automation: Send emails with attachments (smtplib, yagmail).**
- **Scheduling: Set up Cron Jobs (Linux/macOS) or Task Scheduler (Windows).**
- **Hands-on:**
    - **Automate a daily data download, cleaning, and email report.**
    - **Schedule the script to run automatically.**
    - **Log script activity and errors.**

Week 6: Final Project – Automated Data Fetch & Report Generator

- **Project scope: Fetch data daily from an API, process it, generate a report, and email it.**
- **Steps:**
    - **Call the API (e.g., stock prices, weather, news headlines).**
    - **Process and analyze the data (e.g., compute averages, totals, trends).**
    - **Format a report (PDF, HTML, or plain text).**
    - **Email the report to a list of recipients.**
- **Deliverables:**
    - **Fully automated, production-ready Python script.**
    - **Documentation (README, comments).**
    - **Presentation of your workflow and results.**
- **Assessment: Functionality, automation, error handling, code quality, documentation.**

Tools Covered

- **Python (core programming)**
- **requests (API calls)**
- **Flask / FastAPI (API development)**
- **JSON (data interchange)**
- **Postman (API testing)**
- **Cron Jobs / Task Scheduler (automation)**
- **Email libraries (smtplib, yagmail)**

Final Project Example

Title: Automated Data Fetch & Report Generator
Goal: Automate the daily collection, processing, and reporting of data from a public API.
Tasks:

- **Fetch data from API.**
- **Clean and analyze data.**
- **Format report.**
- **Email report.**
- **Schedule the script.**
  Output: Production-ready, documented Python automation.