Advanced Python Programming — Certified Developer Track (48 Hours)

Course description

This 48-hour, project-driven course builds on Python Basics and moves learners into professional development practices: object-oriented design, common patterns, GUI development, data persistence, REST APIs, testing, and packaging. Learners progressively assemble a full end-to-end application (GUI/API + database + analytics + tests), finishing with a capstone demo and certification-style review.

Target audience

- Graduates of the Python Basics course (or equivalent experience)
- Python developers pursuing advanced backend/full-stack roles
- QA automation engineers and API developers
- Learners exploring data/analytics or AI/ML-adjacent Python work

Prerequisites

- Comfortable with Python syntax, functions, and core collections (list/tuple/set/dict)
- Familiarity with basic OOP concepts (classes, objects, methods)
- Completion of the Python Basics course or equivalent knowledge

Duration and schedule

- **Total instructional time:** 48 hours
- **Schedule: 12 class meetings × 4 hours each**
- **Delivery format:** Instructor-led, hands-on labs, project milestones, and demos
- **Recommended session rhythm:** recap → concept → demo → guided lab → debrief (with a midpoint break)

Learning outcomes

By the end of the course, learners will be able to: - Design object-oriented Python applications using best practices and patterns - Implement common design patterns (e.g., Factory, Strategy) in Pythonic ways - Build a desktop GUI using Tkinter/ttk (or PyQt where applicable) - Create data-driven apps using SQLite and optionally an ORM - Build and secure RESTful services using Flask (or Django) - Clean, visualize, and analyze data with Pandas/Matplotlib; demonstrate basic regression - Write unit tests with pytest and improve reliability with coverage/edge-case testing - Package and deliver a runnable project (requirements, virtual environment, optional executable) - Use Git/GitHub workflows for version control and project submission

---

Course modules

1. **Designing Object-Oriented Applications** (UML thinking, encapsulation, inheritance)
2. **Creating Object-Oriented Applications** (patterns, built-ins, Pythonic class ergonomics)
3. **GUI Development with Python** (Tkinter or PyQt5 fundamentals and CRUD workflows)
4. **Data-Driven Applications** (SQLite, schema thinking, parameterized queries, optional ORM)
5. **Web Applications and RESTful APIs** (Flask/Django, HTTP methods, security basics)
6. **Python for Data Science** (NumPy/Pandas/Matplotlib, practical regression demo)
7. **Exception Handling and Unit Testing** (pytest, coverage, reliability practices)
8. **Packaging and Deployment** (venv, requirements, packaging, optional PyInstaller)
9. **Capstone** — **Full-stack Python project** integrating GUI/API + DB + analytics + tests

Modules → Checkpoints: CP1: Modules 1–2; CP2: Modules 1–2 (persistence-ready core + JSON contracts); CP3: Module 3; CP4: Module 4; CP5: Module 5; Final demo: Modules 6–8 + Capstone

---

12-day syllabus schedule (12 sessions × 4 hours)

Day	4-Hour Session Focus	Key deliverables
1	Advanced kickoff + Git workflow; OOP design from requirements; invariants and custom exceptions; composition vs inheritance	Project repo initialized; baseline OOP design
2	Patterns in practice (Factory/Strategy); Pythonic class ergonomics (repr/str/equality); milestone review	**Checkpoint 1** — domain + service layer
3	Project structure (packages/imports/config); logging; context managers; decorators (practical)	Hardened project skeleton
4	HTTP client work (requests) + JSON contracts; security basics; capstone planning workshop	**Checkpoint 2** — persistence-ready core + JSON save/load
5	GUI fundamentals (Tkinter/ttk or PyQt); layout/usability; forms + validation; list UI	GUI shell with validated inputs
6	GUI CRUD wiring; GUI architecture patterns; sprint + polish	**Checkpoint 3** — GUI milestone demo
7	From objects to tables: schema thinking; sqlite3 CRUD; mapping rows to objects; transactions	Working DB layer (SQLite)
8	Optional ORM slice or deeper SQL; integrate DB into the app; search/filter + pagination patterns	**Checkpoint 4** — data-driven app milestone
9	REST fundamentals + Flask setup; CRUD endpoints; serialization + validation; app structure options	API v1 running locally

Day	4-Hour Session Focus	Key deliverables
10	API security basics (appropriate scope); consuming your own API (Python client); GUI↔API integration	**Checkpoint 5** — API milestone demo
11	Pandas essentials (load/clean/summarize); Matplotlib charts; practical regression demo; integrate analytics report feature	Analytics/report feature
12	Testing with pytest; coverage + edge cases; packaging/delivery (requirements, runnable app, optional executable); final demo + certification-style review	Final capstone demo + review

Assessments and completion criteria

Checkpoints

Five project milestones validate mastery across the build: - **CP1:** Domain model + service layer - **CP2:** Persistence-ready core + JSON contracts/save-load - **CP3:** GUI milestone demo - **CP4:** Data-driven milestone (DB integrated) - **CP5:** API milestone demo

Capstone

A full end-to-end project integrating: - A maintainable **domain + service layer** - **GUI** (or API client) workflows - **Database persistence** (SQLite; optional ORM) - **REST API** endpoints and a client integration path - **Analytics/reporting** feature using Pandas/Matplotlib - **Tests** (pytest) and basic coverage/edge-case handling

Suggested capstone submission artifacts: - Git repo with clean structure - `README.md` (run instructions + feature overview) - `requirements.txt` (or equivalent) - Sample data / seed script - Demonstration script or recorded demo outline

Final assessment

- Capstone demonstration
- Certification-style review/quiz

Certification alignment

Primary alignment: Python Institute PCAP™

This course reinforces PCAP objectives through deeper applied design, project structure, and reliability practices.

Progression alignment: Path toward PCPP1™

The course includes topic coverage that supports advancement toward PCPP1 domains, including: - Advanced OOP and best practices - GUI programming fundamentals - Network programming concepts via HTTP/REST client/server work - File processing/environment interactions via persistence and packaging

> Exam versions evolve over time; recommended exam targets and practice strategies should be confirmed at course delivery.

Tools, platforms, and environment

- **OS:** Windows 10/11 (64-bit) or macOS/Linux
- **Core stack:** Python 3.x, pip, virtual environments
- **Data/analytics:** Jupyter (optional), Pandas, Matplotlib, scikit-learn (for regression demo)
- **Web:** Flask or Django (course delivery chooses one primary path)
- **GUI:** Tkinter/ttk (default) or PyQt5 (optional)
- **Database:** SQLite (default); optional MySQL/PostgreSQL discussion if environment supports
- **VM/Lab:** VirtualBox + preconfigured Lubuntu VM *or* a managed remote lab environment (if provided)
- **IDE:** VS Code or PyCharm Community
- **Version control:** Git + GitHub

Scope boundaries

To keep this course focused and finishable within 48 hours, the following are intentionally limited or out of scope: - Deep ML pipelines, model tuning, or production MLOps - Production-grade deployment (cloud infra, containers/Kubernetes) - Advanced security beyond entry-level API hardening patterns - Enterprise database administration (beyond app-level CRUD and schema fundamentals)

Suggested pacing notes (instructor)

- Push learners to keep **UI separate from core logic** early.
- Treat checkpoints as “merge gates” (feature must run and pass quick checks before proceeding).
- Encourage incremental commits and small pull requests to reduce integration pain.
- Preserve optionality: ORM and PyQt can be offered as extensions without derailing the core path.