

Python & FastAPI

Complete Detailed eBook (2026 Edition)

Includes Python 3.12/3.13, FastAPI, Pydantic v2, Starlette, Uvicorn, SQLAlchemy

Author: M365 Copilot | Date: Feb 9, 2026

Table of Contents

1. Executive Summary
2. Python Platform: 3.12 & 3.13 Highlights
3. Typing & Patterns Essentials
4. FastAPI Overview & Core Concepts
5. Pydantic v2: Validation & Serialization
6. Starlette & Uvicorn: ASGI Foundation
7. Data Layer with SQLAlchemy
8. Building Production APIs (Auth, Docs, Testing)
9. Performance & Observability
10. Deployment Recipes (Containers, Gunicorn/Uvicorn)
11. Migration Playbook (Flask/Django → FastAPI)
12. Security Hardening Checklist
13. References

1. Executive Summary

Python 3.12 and 3.13 deliver substantial improvements to typing, performance and tooling (e.g., PEP 695 type parameters, f-string grammar, per-interpreter GIL; plus 3.13's experimental free-threaded build and basic JIT). FastAPI builds on Starlette (ASGI) and Pydantic to offer a high-performance, standards-based API framework with automatic OpenAPI docs and strong type-driven validation. Uvicorn serves FastAPI with HTTP/1.1 & WebSockets support; SQLAlchemy unifies SQLAlchemy ORM and Pydantic models for simple, type-safe persistence. [Sources: docs.python.org 3.12/3.13; FastAPI docs]

2. Python Platform: 3.12 & 3.13 Highlights

Python 3.12 (Oct 2, 2023): PEP 695 type parameter syntax & type statement; PEP 701 f-strings formalized; PEP 684 per-interpreter GIL; PEP 669 low-impact monitoring; numerous stdlib and performance upgrades. [Source: What's New in Python 3.12]

Python 3.13 (Oct 7, 2024): new interactive interpreter with colors; experimental free-threaded CPython (PEP 703) and basic JIT (PEP 744); defined semantics for locals(); typing defaults and other typing PEPs; removals from stdlib per PEP 594. [Source: What's New in Python 3.13]

3. Typing & Patterns Essentials

PEP 695 introduces concise generic syntax (type parameters and type aliases).

3.13 adds defaults for type parameters and other typing features;

`typing_extensions` backports new features for older runtimes. Structural pattern matching (3.10+) remains powerful for declarative control flow.

[Sources: PEP 695; Python 3.13 What's New; `typing_extensions`; PEP 636]

4. FastAPI Overview & Core Concepts

FastAPI is a modern, high-performance web framework for building APIs with Python type hints. It automatically generates OpenAPI/Swagger docs, leverages Pydantic for data validation, and runs atop Starlette for async I/O. Core ideas: path operations, dependency injection, data models, automatic docs (Swagger UI/Redoc). [Sources: fastapi.tiangolo.com; GitHub README]

Best practices: prefer `async def` for I/O; validate with Pydantic models; use `response_model` for schema; configure lifespan events; version your APIs; use middleware for CORS, gzip; document errors; add health endpoints.

5. Pydantic v2: Validation & Serialization

Pydantic v2 is a ground-up rewrite with a Rust core (pydantic-core) delivering major performance gains and new features (strict mode, improved JSON Schema, flexible serialization). It remains the de facto validation layer in FastAPI.

[Sources: Pydantic docs; PyPI project; Cisco v2 notes]

6. Starlette & Uvicorn: ASGI Foundation

Starlette is a lightweight ASGI toolkit providing routing, WebSockets, background tasks, middleware, CORS, GZip, and great performance—forming FastAPI's web layer. [Sources: starlette.dev; PyPI]

Uvicorn is a high■performance ASGI server supporting HTTP/1.1 and WebSockets, with optional uvloop/httpptools for speed; includes updated WebSocket protocols (SansIO, 2025). [Sources: uvicorn.dev; PyPI]

7. Data Layer with SQLAlchemy

SQLModel unifies SQLAlchemy ORM with Pydantic models so one class can serve as both DB table and request/response schema—reducing duplication and integrating naturally with FastAPI. [Sources: sqlmodel.tiangolo.com; PyPI]

8. Building Production APIs (Auth, Docs, Testing)

Auth: OAuth2 with JWT bearer, API keys, or session cookies; prefer OAuth2PasswordBearer for token flows. Docs: customize OpenAPI metadata and tags. Testing: use TestClient/httpx, dependency overrides for isolation, and pytest with factories; add contract tests against generated OpenAPI. [Sources: fastapi docs; starlette docs]

9. Performance & Observability

Use Uvicorn with uvloop/httptools; enable keep-alive tuning; leverage async DB drivers; profile with asyncio tools. Observability: structured logs, metrics/traces via OpenTelemetry; track validation errors and latency percentiles. [Sources: uvicorn PyPI; starlette docs]

10. Deployment Recipes (Containers, Gunicorn/Uvicorn)

Containerize with a slim base; use Gunicorn with Uvicorn workers for multi-process serving; choose workers by CPU and workload; disable access logs in high-throughput paths; use health checks and readiness probes; prefer environment-driven configuration. [Sources: starlette docs performance notes; uvicorn docs]

11. Migration Playbook (Flask/Django → FastAPI)

Incremental approach: retain WSGI for legacy routes; build new ASGI services in FastAPI; share domain models via Pydantic; introduce SQLAlchemy to consolidate schemas; front with a gateway; deprecate legacy endpoints iteratively.

12. Security Hardening Checklist

Validate all inputs with Pydantic; set CORS correctly; enable HTTPS/HSTS; limit request sizes; sanitize headers; rotate secrets; pin versions; run with non-root user; use dependency overrides in tests to avoid real secrets.

13. References

- [Python 3.12 What's New — docs.python.org](#) (turn11search141)
- [Python 3.12 docs index — DevDocs](#) (turn11search146)
- [Python 3.13 What's New — docs.python.org](#) (turn11search147)
- [Python versions status — devguide.python.org](#) (turn11search150)
- [PEP 636 Structural Pattern Matching tutorial](#) (turn11search153)
- [typing_extensions PyPI](#) (turn11search155)
- [FastAPI docs homepage](#) (turn11search159)
- [FastAPI GitHub README](#) (turn11search160)
- [Starlette docs](#) (turn11search123) & [PyPI](#) (turn11search125)
- [Uvicorn docs WebSockets](#) (turn11search135) & [PyPI](#) (turn11search136)
- [Pydantic docs latest](#) (turn11search129) & [PyPI](#) (turn11search130)
- [SQLModel docs](#) (turn11search117) & [PyPI](#) (turn11search118)