

Python Full-Stack Development

Build, test and deploy web applications with Python, Django REST Framework, PostgreSQL and React

Full-Stack Development · intermediate · Live online, instructor-led

This syllabus is a draft awaiting instructor review and may change.

A 108-hour live course: advanced Python, PostgreSQL and SQL, Django and Django REST Framework APIs, authentication and permissions, a React front end, pytest, Celery and Redis, Docker, deployment and a reviewed capstone.

Length and schedule

108 hours of live teaching, on either schedule:

- Weekday Intensive (Mon–Fri): about 22 weeks before holidays
- Weekend Evening (Sat & Sun evenings): about 27 weeks before holidays

Fees per student, whole course

- Group: Rs. 19,999
- Duo: Rs. 32,999
- One-to-One: Rs. 59,999

Launch fee — GST treatment to be confirmed.

What you will be able to do

- Write idiomatic, typed Python using comprehensions, generators, decorators and context managers
- Design a normalised PostgreSQL schema and write the SQL that Django generates and that it does not
- Model data with the Django ORM and build versioned REST APIs with Django REST Framework
- Implement authentication, object-level permissions and protection against common OWASP risks
- Build a React and TypeScript front end that consumes your own API
- Test the back end with pytest and the front end with component and end-to-end tests
- Run background work with Celery and Redis
- Containerise, deploy and monitor the application, and integrate an AI API responsibly

Before you start

- Basic programming in any language: variables, conditions, loops and functions
- HTML and CSS basics: you can build and style a simple static page
- Comfortable using a terminal to run commands and navigate folders
- A laptop with at least 8 GB RAM (16 GB recommended for Docker) running Windows, macOS or Linux

Tools

Python 3.13 or later, Django 5.2 LTS, Django REST Framework, PostgreSQL 17 or 18, psql and pgAdmin, pytest and pytest-django, Celery, Redis, Node.js 24 LTS, React 19, TypeScript, Vite, TanStack Query, Playwright, Docker and Docker Compose, GitHub Actions, Gunicorn, Git and GitHub, Visual Studio Code or PyCharm

Syllabus

Module 1: Python essentials for programmers

A fast, thorough grounding in Python for people who already program, with a professional project setup from day one.

1. Python setup, virtual environments and core types (2 h)
2. Collections and control flow (2 h)
3. Functions and modules (2 h)
4. Classes, exceptions and files (2 h)
5. Python essentials review and doubt-clearing (2 h)

Module 2: Advanced Python

The language features professional Python code relies on, and how to package and check code properly.

6. Type hints and static checking (2 h)
7. Comprehensions, iterators and generators (2 h)
8. Decorators and context managers (2 h)
9. Packaging, dependencies and code quality (2 h)
10. Advanced Python review and doubt-clearing (2 h)

Module 3: SQL and PostgreSQL

Relational modelling and the SQL every Django developer needs to understand what the ORM is doing.

11. Relational modelling and PostgreSQL setup (2 h)
12. Querying with joins and aggregation (2 h)
13. Subqueries, CTEs and window functions (2 h)
14. Indexes and transactions (2 h)
15. SQL review and doubt-clearing (2 h)

Module 4: Django fundamentals and the ORM

Build a Django project, model data with the ORM and migrations, and query it efficiently.

16. Django project structure and request cycle (2 h)
17. Models, fields and relationships (2 h)
18. Migrations and database constraints (2 h)
19. Querying with the ORM (2 h)
20. Transactions, managers and the Django admin (2 h)
21. Django review and doubt-clearing (2 h)

Module 5: REST APIs with Django REST Framework

Design and build a clean, versioned REST API with DRF, and understand how FastAPI differs.

22. REST API design (2 h)
23. Serializers and validation (2 h)
24. Views, viewsets and routers (2 h)
25. Filtering, pagination and error handling (2 h)
26. API documentation and a FastAPI comparison (2 h)
27. API review and doubt-clearing (2 h)

Module 6: Authentication, permissions and security

Sign users in safely, enforce who may do what on the server, and defend against common web vulnerabilities.

- 28. Authentication options for an API (2 h)
- 29. Permissions and object-level access (2 h)
- 30. OWASP basics and Django security (2 h)
- 31. Password reset, email and personal data (2 h)
- 32. Security review and doubt-clearing (2 h)

Module 7: React front end

Build a typed React client for your own API, from JavaScript essentials to routing, server state and forms.

- 33. JavaScript and TypeScript for Python developers (2 h)
- 34. React components, props and state (2 h)
- 35. Effects, custom hooks and project structure (2 h)
- 36. Routing and server state (2 h)
- 37. Forms, authentication and API integration (2 h)
- 38. Front-end review and doubt-clearing (2 h)

Module 8: Automated testing

Write the right tests at the right level so the application can change with confidence.

- 39. pytest fundamentals (2 h)
- 40. Testing Django and DRF (2 h)
- 41. Front-end and end-to-end testing with doubt-clearing (2 h)

Module 9: Background processing with Celery and Redis

Move slow and scheduled work out of the request cycle, and understand where async Python fits.

- 42. Background work and Celery basics (2 h)
- 43. Scheduling, retries and idempotency (2 h)
- 44. Async Python, caching and doubt-clearing (2 h)

Module 10: Docker, deployment, monitoring and AI integration

Package and deploy the full application, keep it observable in production, and integrate an AI API responsibly as an advanced project.

- 45. Docker and Docker Compose (2 h)
- 46. CI/CD and production deployment (2 h)
- 47. Logging, monitoring and production debugging (2 h)
- 48. Advanced project: integrating an AI API (2 h)

Module 11: Capstone project and final assessment

Build, harden and present the capstone application, with structured reviews and the final assessment.

- 49. Capstone kickoff and planning (2 h)
- 50. Capstone build lab: models, API and permissions (2 h)
- 51. Capstone build lab: front end, tasks and deployment (2 h)
- 52. Capstone mid-point review (2 h)
- 53. Final assessment (2 h)
- 54. Capstone presentations and final review (2 h)

Final project

Build and deploy a community food-bank distribution system. A city food bank coordinates donations from shops and households and distributes them through several partner centres. It needs a web application

where donors log donations with quantities and expiry dates; centre staff record stock and schedule distribution slots; registered households book a collection slot; and coordinators see stock levels, items nearing expiry and distribution per centre. You receive a written brief with user stories and acceptance criteria, and build it individually or in a pair.

Deliverables- A PostgreSQL schema designed through Django models, with constraints, migrations and documented indexes for the main queries.- A Django REST Framework API with versioned endpoints, serializer validation, pagination, filtering and a consistent error format.- Authentication with secure sessions or short-lived tokens, and object-level permissions proving that staff at one centre cannot change another centre's stock and that households see only their own bookings.- Safe concurrent booking: transactions and row locking so a collection slot cannot be over-booked.- A Celery task, backed by Redis, that sends daily near-expiry alerts and one scheduled report.- A React and TypeScript front end with routing, validated forms and loading, empty and error states on every data view.- Automated tests: pytest unit and API tests including permission denials, and one Playwright journey for booking a collection slot.- A Dockerfile and Docker Compose setup, a GitHub Actions workflow running linting, type checks and tests, and a live deployment with no secrets in the repository, plus health checks and structured logging.- A README with setup steps, an architecture diagram and known limitations.

Optional extension: an AI-assisted feature, such as suggesting product categories from free-text donation descriptions, that is opt-in, explains what is sent to the AI provider and lets staff correct every suggestion. The project is presented and reviewed live in the final weeks, with feedback on code quality, security and design decisions.

Certificate

- At least 80% attendance
- All assignments submitted
- Projects passed
- Final assessment passed (60%)

A ZORK Academy course completion certificate; paying for the course does not by itself earn it.

Next step

Book a free consultation: <https://zorkacademy.online/consultation?course=python-full-stack-development>

We help you prepare for work; we do not guarantee jobs, placements or salaries.