Python Course 2026

A comprehensive journey through modern Python development, from foundational IT concepts to advanced frameworks. This course equips you with essential skills in artificial intelligence, data analysis, and full-stack web development using industry-leading tools and frameworks.

Fundamentals of IT & AI

Build your foundation with application lifecycle management, Agile methodologies, and AI concepts

Python for Data

Master Python syntax, data structures, and advanced programming techniques

SQL Databases

Learn PostgreSQL for robust data management and complex queries

Data Libraries

Explore NumPy, Pandas, and visualization tools for data analysis

Django Framework

Build full-stack web applications with Django's MVT architecture

FastAPI Framework

Create modern, high-performance APIs with async capabilities

Digital Edify

India's First AI-Native Training Institute

Learn AI. Build Agents. Lead Future.

hello@digitaledify.ai

+91 6304982304

About Digital Edify

India's #1 Training Institute for the AI Era

Established: 2016 **Headquarters:** Hyderabad, Telangana **Reach:** Global (Online + Offline)

The Transformation Narrative

Digital Edify has evolved from a premium training institute in the Automation Era to an AI-first organisation leading the Agentic AI revolution. Since 2016, we've transformed over 100,000 professionals and built partnerships with more than 1,000 industry leaders. Our journey reflects the technological evolution of our time—from traditional job placement to career transformation, and now to building AI-native professionals who will shape the future of work.

1

Automation Era (2016-2023)

Premium Training Institute focused on job placement with 100K+ students trained

2

AI Revolution (2024-2025)

AI-Powered Training with industry-AI integration and career transformation focus

3

Agentic AI Leadership (2026+)

AI First Institute building AI-Native Professionals with 1 Million AI-Native Vision

> "We started in the Automation Era. We evolved through the AI Revolution. Now, we're leading the Agentic AI Future—with 100,000+ professionals already transformed and 1,000+ industry partners trusting our graduates."

Vision & Mission

Vision

"To Create 1 Million AI-Native Professionals Who Will Build the Agentic Future of Work"

Mission

"We transform learners into AI-native professionals through industry-aligned programmes that integrate Agentic AI into every discipline—from development to data science to enterprise platforms."

Course Highlights

Section 1: Fundamentals of IT & AI

Learn how modern applications are built, managed, and powered by agile practices, cloud computing, and AI technologies.

Section 2: Python for AI & Data

Develop strong Python programming foundations covering data structures, OOP, file handling, and advanced concepts for AI and data work.

Section 3: SQL for AI & Data

Design, query, optimize, and automate relational databases using advanced PostgreSQL and SQL techniques.

Section 4: Python Libraries for AI & Data

Analyze, manipulate, and visualize data using NumPy, Pandas, and powerful Python visualization libraries.

Section 5: FullStack Python Framework Django

Build full-stack web applications using Django with ORM, authentication, templates, and REST APIs.

Section 6: Modern Python Framework FastAPI

Create high-performance, secure, and scalable APIs using FastAPI, async programming, and modern authentication.

Section 1: Fundamentals of IT & AI

Module 1: Application Life Cycle Management

Application Fundamentals

Understanding what applications are, their various types, and the core principles of web application architecture.

Web Technologies

Frontend technologies including HTML, CSS, JavaScript, and React. Backend with Python, Java, and Node.js.

Database Systems

SQL databases (MySQL, PostgreSQL) and NoSQL solutions (MongoDB) for data management.

The Software Development Life Cycle (SDLC) encompasses seven critical phases: Planning, Analysis, Design, Implementation, Testing, Deployment, and Maintenance. Each phase builds upon the previous, ensuring systematic development of robust applications. Understanding this lifecycle is fundamental to managing complex software projects and delivering quality solutions that meet user requirements whilst maintaining scalability and performance standards.

Module 2: Agile & Scrum Framework

Methodology Evolution

Traditional Waterfall approaches follow sequential phases, whilst Agile embraces iterative development with continuous feedback. The Agile mindset prioritises flexibility, collaboration, and rapid response to change.

Popular frameworks include Scrum, Kanban, and Extreme Programming (XP), each offering unique approaches to project management.

01

Scrum Roles

Product Owner, Scrum Master, and Development Team work collaboratively.

02

Scrum Events

Sprint, Planning, Daily Scrum, Review, and Retrospective ceremonies.

03

Scrum Artifacts

Product Backlog, Sprint Backlog, and Increment deliverables.

04

User Stories

Writing effective stories with Epics, Themes, and Acceptance Criteria.

Estimating user stories and managing backlogs effectively requires tools like Google Sheets and Azure Boards. These platforms facilitate transparent communication, progress tracking, and collaborative planning across distributed teams.

Module 3: Computing & Data

CPU Technology

The Central Processing Unit serves as the brain of computing systems, executing instructions and managing data flow. Modern CPUs feature multiple cores for parallel processing, enabling efficient multitasking and complex computational tasks.

GPU Technology

Graphics Processing Units excel at parallel computations, making them essential for AI training, data processing, and scientific simulations. Their architecture handles thousands of simultaneous operations efficiently.

IaaS

Infrastructure as a Service provides virtualised computing resources over the internet, offering scalable servers, storage, and networking.

PaaS

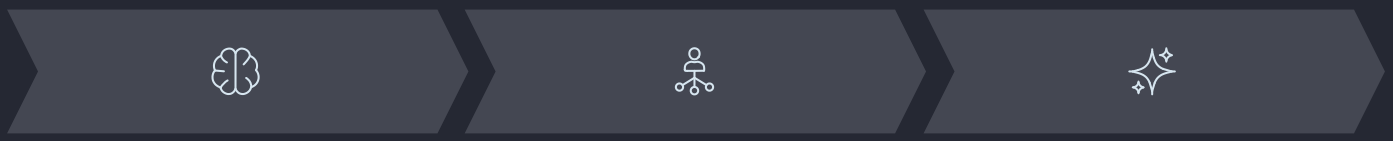
Platform as a Service delivers development and deployment environments, enabling developers to build applications without infrastructure management.

SaaS

Software as a Service provides ready-to-use applications accessible via web browsers, eliminating installation and maintenance requirements.

Module 4: Introduction to AI, Generative AI & Agentic AI

Artificial Intelligence represents the simulation of human intelligence in machines programmed to think and learn. AI systems process vast amounts of data, identify patterns, and make decisions with minimal human intervention.



Machine Learning

Algorithms that improve through experience, learning from data patterns without explicit programming.

Deep Learning

Neural networks with multiple layers that process complex patterns in large datasets.

Generative AI

Systems that create new content, from text to images, based on learned patterns.

Large Language Models

LLMs process and generate human-like text, understanding context and nuance. These models power chatbots, content creation, and language translation services.

Image Generation

AI models create original images from text descriptions, revolutionising creative industries and visual content production workflows.

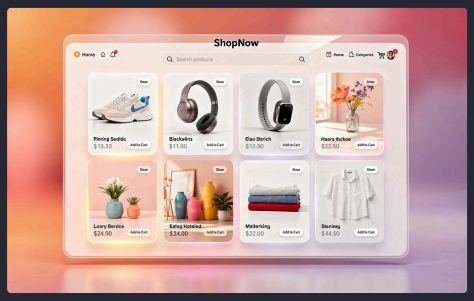
Module 5: Real-World Applications

Customer Relationship Management

CRM systems centralise customer interactions, sales pipelines, and marketing campaigns. These platforms enhance customer satisfaction through personalised engagement and data-driven insights.

Human Resource Management Systems

HRMS solutions streamline recruitment, payroll, performance management, and employee records. Automated workflows reduce administrative burden whilst improving compliance and employee experience.



Retail & E-Commerce

Digital commerce platforms integrate inventory management, payment processing, and customer analytics. These systems enable seamless shopping experiences across multiple channels.

Healthcare Applications

Medical software manages patient records, appointment scheduling, and diagnostic support. These systems improve care coordination whilst ensuring data security and regulatory compliance.

SECTION 2

Python for AI & Data

Module 1: Python Fundamentals

Python's elegant syntax and powerful capabilities make it the preferred language for AI and data science. This module establishes foundational programming concepts essential for advanced applications.

Environment Setup

Installing Python interpreter for Windows and Mac, configuring Visual Studio Code IDE for optimal development experience.

Core Syntax

Python's 35 keywords, identifiers, naming conventions, variables, and memory management fundamentals.

Data Types

Simple and complex data types, type conversion, type casting, and operators for arithmetic and logical operations.

01

Conditional Statements

if, elif, else, and match-case structures for decision-making logic.

02

Loops

while and for loops with range() function for iteration control.

03

Control Flow

break, continue, and pass statements for advanced flow management.

Module 2: String Manipulation

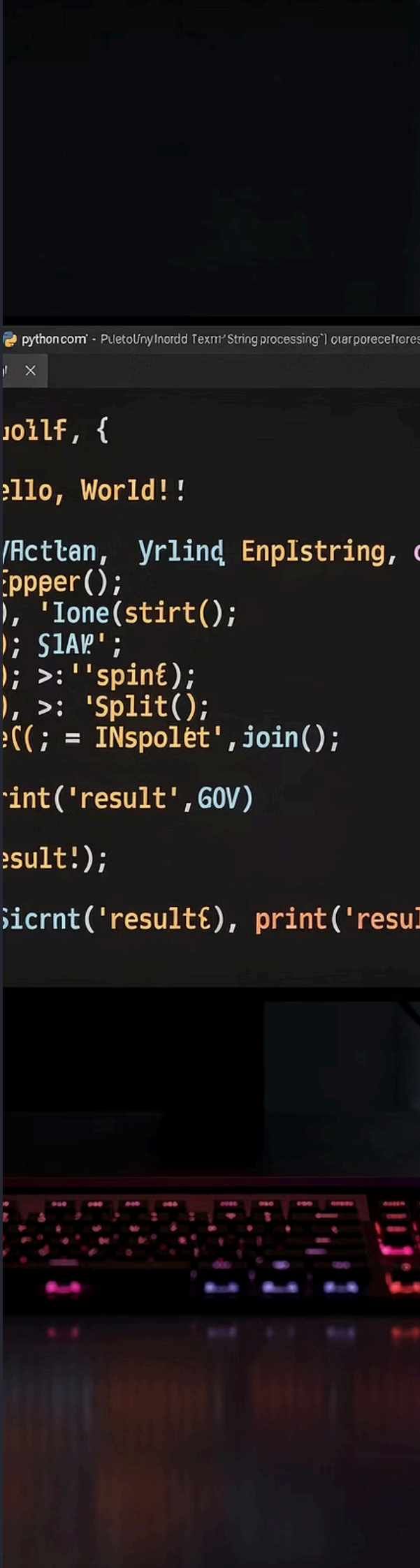
Strings form the foundation of text processing in Python. Understanding string operations enables efficient data cleaning, text analysis, and content manipulation across diverse applications.

String Fundamentals

- String definition and syntax rules
- Positive and negative indexing
- Slicing with start:end:step notation
- Concatenation and repetition operations
- String formatting with f-strings and format()
- Immutability concept and implications

String Methods

- Case conversion: upper(), lower(), title()
- Search methods: find(), index(), count()
- Checking methods: isalpha(), isdigit(), isspace()
- Trimming: strip(), lstrip(), rstrip()
- Replacement and split/join operations
- Alignment methods for formatting



Module 3: Data Structures - Lists & Tuples

Lists

Mutable sequences supporting dynamic data storage. Lists enable creation, indexing, slicing, and comprehensive operations for data manipulation.

- Adding: append(), insert(), extend()
- Removing: remove(), pop(), clear()
- Searching: index(), count()
- Sorting: sort(), reverse()

Tuples

Immutable sequences offering data integrity and performance benefits. Tuples support packing, unpacking, and efficient memory usage.

- Creation and basic operations
- Immutability advantages
- Packing and unpacking techniques
- Performance comparisons with lists

List comprehensions provide elegant syntax for creating lists based on existing sequences. This Pythonic approach combines loops and conditional logic into concise, readable expressions that enhance code clarity whilst maintaining performance.

Module 4: Data Structures - Dictionaries & Sets

Dictionaries

Key-value pairs enabling efficient data lookup and storage. Dictionaries support dynamic operations including keys(), values(), and items() methods.

Dictionary comprehensions create dictionaries concisely. Nested dictionaries handle complex hierarchical data structures effectively.

Union

Combines all unique elements from multiple sets.

Intersection

Returns common elements shared between sets.

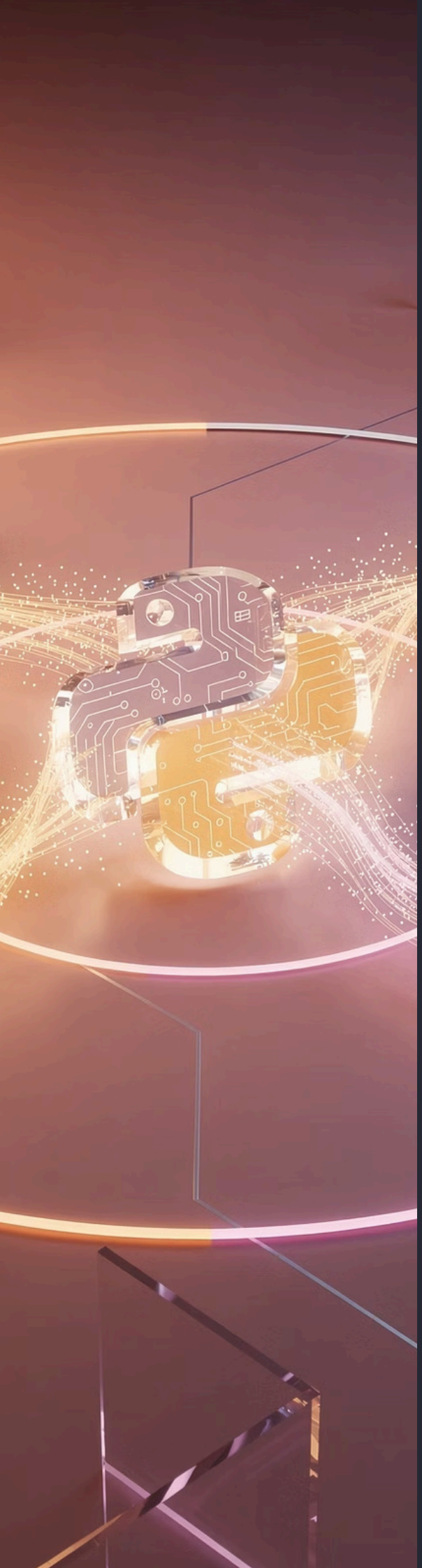
Subset & Superset

Checks hierarchical relationships between sets.

Difference

Elements present in one set but not another.

Sets maintain unique, unordered collections with mathematical operations. Frozen sets provide immutable alternatives for scenarios requiring hashable collections. These structures excel in membership testing and eliminating duplicates efficiently.



Module 5: Advanced Collections & Iterators

Collections Module

Specialised container datatypes: namedtuple for readable tuples, Counter for counting hashable objects, defaultdict for default values, and deque for efficient queue operations.

Iterators & Generators

Iteration protocol enables custom iterators. Generators using yield statements provide memory-efficient data streaming through lazy evaluation.

Functional Programming

Lambda functions create anonymous functions. Higher-order functions like map(), filter(), and reduce() enable functional programming paradigms.

Generator expressions offer concise syntax for creating generators, similar to list comprehensions but with lazy evaluation. Generator pipelines chain multiple operations efficiently, processing data streams without loading entire datasets into memory. This approach proves essential for handling large-scale data processing tasks.

hello@digitaledify.ai

Module 6: Functions & Scope

Functions encapsulate reusable code blocks, promoting modularity and maintainability. Understanding function mechanics and scope rules enables sophisticated program architecture.

Function Basics

Definition, calling, parameters, and arguments form the foundation of function usage.

Return Values

Single and multiple return values enable functions to communicate results effectively.

Argument Types

Positional, keyword, default, *args, and **kwargs provide flexible parameter handling.

Scope Management

Local and global scope with global keyword control variable accessibility.

Built-in Functions

Python provides extensive built-in functions for common operations, reducing development time.

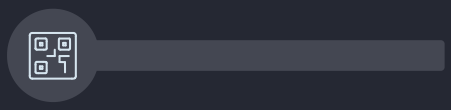
Lambda Functions

Anonymous functions enable concise inline operations and IIFE patterns.

Recursion

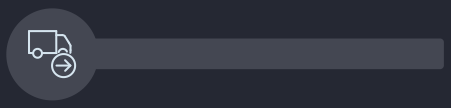
Functions calling themselves solve problems through elegant divide-and-conquer approaches.

Module 7: Modules & Packages



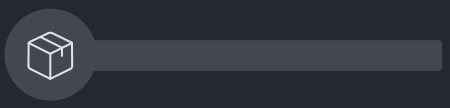
Module Types

Built-in modules ship with Python, user-defined modules organise custom code, and external packages extend functionality.



Importing Techniques

Various import methods enable selective or complete module access, supporting namespace management.



Package Structure

Packages organise related modules hierarchically. __init__.py files define package behaviour and nested structures.

Common Built-in Modules

math for mathematical operations, random for randomisation, datetime for temporal data, os for operating system interaction, and sys for system-specific parameters.

Package Management

pip installs external packages from PyPI. requirements.txt files manage project dependencies, ensuring reproducible environments across development teams.

Popular Packages

requests simplifies HTTP operations, pandas enables data analysis, and numpy provides numerical computing capabilities essential for scientific applications.

Module 8: Working with Data Formats

File operations and data format handling enable persistent storage and data exchange. Mastering these techniques facilitates integration with external systems and data pipelines.

File Operations

CRUD operations with open() function, supporting various file modes for reading, writing, and appending data.

Directory Management

os and shutil modules provide comprehensive file path operations and directory manipulation capabilities.

CSV Files

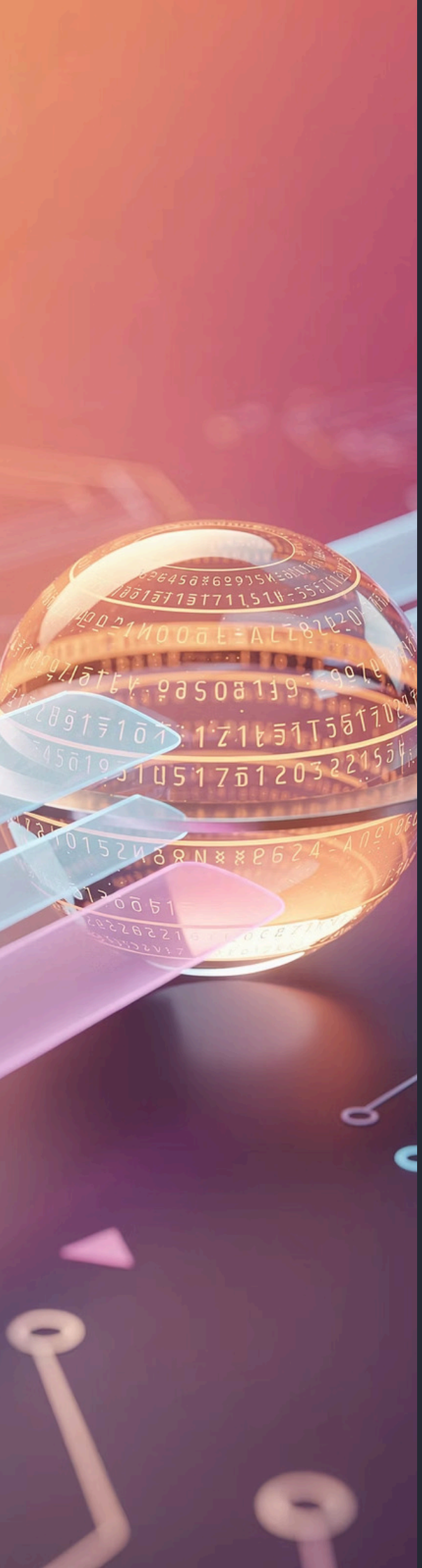
Comma-separated values represent tabular data efficiently. Python's csv module provides reader, writer, DictReader, and DictWriter classes for structured data handling.

CSV files integrate seamlessly with data analysis workflows, supporting import and export operations for spreadsheet applications.

JSON Files

JavaScript Object Notation offers human-readable data interchange format. JSON operations include dump(), dumps(), load(), and loads() for serialisation and deserialisation.

JSON's flexibility makes it ideal for configuration files, API responses, and structured data storage.



Module 9: Advanced Python Concepts

1

Exception Handling

try-except-else-finally blocks manage errors gracefully, preventing program crashes.

2

Custom Exceptions

Creating exception classes enables domain-specific error handling strategies.

3

Decorators

Function and class decorators modify behaviour without altering source code.

4

Context Managers

with statements ensure proper resource management and cleanup operations.

Exception handling encompasses catching specific exceptions, raising custom exceptions, and re-raising for upstream handling. Built-in exception types cover common error scenarios. Decorators accept arguments and stack multiple modifications. Practical applications include logging, timing, authentication, and caching. Generators enable infinite sequences and memory-efficient data processing. Custom context managers implement __enter__ and __exit__ methods for resource management protocols.

Module 10: Object-Oriented Programming

Object-Oriented Programming (OOP) models real-world entities through classes and objects. This paradigm promotes code reusability, maintainability, and scalability through encapsulation, inheritance, abstraction, and polymorphism.

Encapsulation

Access modifiers (public, protected, private) control data visibility and protect internal state.

Polymorphism

Method overriding and duck typing enable flexible object interactions.

Inheritance

Single, multi-level, and multiple inheritance enable code reuse and hierarchical relationships.

Abstraction

Abstract classes and methods define interfaces without implementation details.

- ### Classes & Objects

  Classes define blueprints; objects represent instances with attributes and methods.

- ### Methods

  Instance methods, class methods (@classmethod), and static methods (@staticmethod) serve different purposes.

- ### Special Methods

  Magic methods like __str__, __repr__, and __len__ customise object behaviour.

SECTION 3

SQL for AI & Data

Module 1: Foundations of Databases & PostgreSQL

Relational Database Management Systems (RDBMS) organise data into structured tables with defined relationships. PostgreSQL, an advanced open-source RDBMS, provides robust features for data integrity, complex queries, and scalability.

1

ACID Properties

Atomicity ensures complete transactions, Consistency maintains data validity, Isolation prevents interference, and Durability guarantees persistence.

2

Installation & Setup

PostgreSQL installation across Windows, Mac, and Linux platforms. Configuration of psql command-line tool and pgAdmin 4 graphical interface.

3

Database Objects

Databases contain schemas, which organise tables. Tables store structured data with defined columns and data types.

Data Types

- Numeric: INTEGER, DECIMAL, FLOAT
- Character: VARCHAR, TEXT, CHAR
- Date/Time: DATE, TIME, TIMESTAMP
- Boolean: TRUE/FALSE values
- Special: JSON, ARRAY, UUID

Constraints

- PRIMARY KEY: Unique identifier
- FOREIGN KEY: Referential integrity
- UNIQUE: No duplicate values
- NOT NULL: Required values
- CHECK: Custom validation rules
- DEFAULT: Automatic values

Module 2: Querying and Analysing Data

SQL queries extract, filter, and analyse data from databases. Mastering query techniques enables sophisticated data exploration and insight generation.

Basic Queries

SELECT statements retrieve data with column aliases and expressions. WHERE clauses filter results using comparison and logical operators.

Advanced Filtering

BETWEEN, IN, and LIKE operators enable range, list, and pattern matching. NULL handling with IS NULL and IS NOT NULL ensures accurate results.

Result Control

ORDER BY sorts data, DISTINCT removes duplicates, and LIMIT with OFFSET implements pagination for large result sets.

1 Functions

String functions (UPPER, LOWER, CONCAT, SUBSTRING), numeric functions (ROUND, CEIL, FLOOR, ABS), and date/time functions (CURRENT_DATE, EXTRACT, DATE_TRUNC) transform data.

2 Aggregations

Aggregate functions (COUNT, SUM, AVG, MIN, MAX) summarise data. GROUP BY groups rows, whilst HAVING filters grouped results.

3 Window Functions

ROW_NUMBER, RANK, LAG, and LEAD perform calculations across row sets without grouping, enabling sophisticated analytical queries.

Module 2: Querying - JOIN Operations

INNER JOIN

Returns matching rows from both tables based on join condition.

SELF JOIN

Joins table to itself for hierarchical relationships.

CROSS JOIN

Returns Cartesian product of both tables.

LEFT JOIN

Returns all rows from left table plus matching rows from right table.

RIGHT JOIN

Returns all rows from right table plus matching rows from left table.

FULL OUTER JOIN

Returns all rows from both tables, matching where possible.

Multi-table joins combine data from three or more tables, enabling complex queries across related entities. Join optimisation through proper indexing and query structure ensures efficient execution, particularly important for large datasets and production environments.

Module 3: Advanced Queries & Data Manipulation

Subqueries

Subqueries in WHERE, SELECT, and FROM clauses enable nested logic. Correlated subqueries reference outer query values. EXISTS and NOT EXISTS test for row existence efficiently.

Common Table Expressions

CTEs create temporary named result sets, improving query readability. Recursive CTEs handle hierarchical data like organisational structures. Multiple CTEs chain complex operations.

UNION

Combines results from multiple queries, removing duplicates. UNION ALL includes duplicates.

INTERSECT

Returns common rows appearing in both query results.

EXCEPT

Returns rows from first query not present in second query.

UPDATE statements modify existing records with expressions and JOIN operations. DELETE removes records based on conditions or subqueries. TRUNCATE quickly removes all rows whilst preserving table structure. Transaction management with BEGIN, COMMIT, and ROLLBACK ensures data consistency. Savepoints enable partial rollbacks. Transaction isolation levels control concurrent access, whilst concurrency control prevents conflicts in multi-user environments.

Module 4: Database Programming & Automation

ALTER TABLE

Modify table structure: add, modify, or drop columns and constraints dynamically.

Indexes

B-tree, Hash, GIN, and GiST indexes accelerate queries. Strategic indexing balances performance and storage.

Views

Virtual tables simplify complex queries. Materialised views cache results for performance.

Functions

PL/pgSQL stored functions encapsulate logic with parameters, return types, and control structures.

Procedures

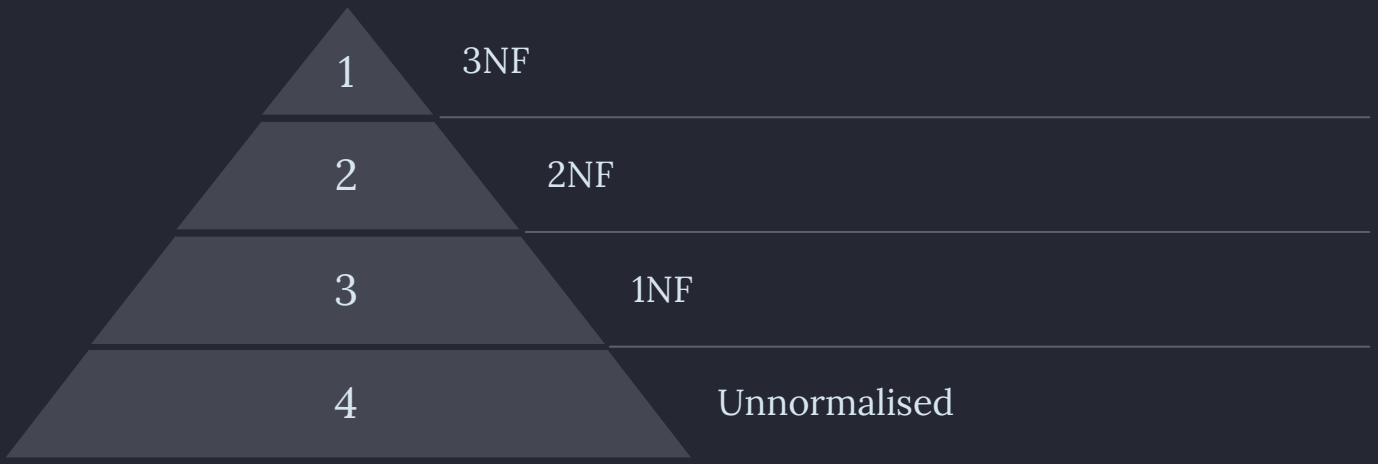
Stored procedures execute complex operations with transaction control and exception handling.

Triggers

BEFORE, AFTER, and INSTEAD OF triggers automate actions on data changes for audit logging and validation.

Module 5: Database Design & Optimisation

Effective database design ensures data integrity, query performance, and scalability. Entity-Relationship modelling visualises data structures before implementation.



Normalisation eliminates redundancy through progressive normal forms. First Normal Form (1NF) ensures atomic values. Second Normal Form (2NF) removes partial dependencies. Third Normal Form (3NF) eliminates transitive dependencies. Strategic denormalisation trades redundancy for query performance in specific scenarios.

01

Design Best Practices

Consistent naming conventions, appropriate data type selection, and strategic primary key design.

02

Query Optimisation

EXPLAIN ANALYZE reveals execution plans. Index strategies and query rewriting improve performance.

03

Performance Tuning

Database statistics with ANALYZE, VACUUM maintenance, connection pooling, and table partitioning enhance scalability.

SECTION 4

Python Libraries for AI & Data

Module 1: NumPy - Numerical Computing

NumPy provides the foundation for numerical computing in Python. Its efficient array operations and mathematical functions enable high-performance scientific computing essential for AI and data science applications.

Array Creation

Multiple methods create arrays: zeros(), ones(), arange(), linspace(), and random() functions generate arrays with specific patterns or values.

Array Operations

Indexing, slicing, fancy indexing, reshaping, and transposing manipulate array structures efficiently.

Broadcasting

Broadcasting rules enable operations on arrays of different shapes, eliminating explicit loops for vectorised computations.

Mathematical Operations

Universal functions (ufuncs) perform element-wise operations efficiently. Linear algebra operations include dot products and matrix multiplication.

Statistical Analysis

Statistical functions calculate mean, median, standard deviation, variance, and percentiles for data analysis.

Module 2: Pandas - Data Manipulation & Analysis

Pandas revolutionises data manipulation in Python through intuitive data structures and comprehensive functionality. Series and DataFrame objects handle one-dimensional and two-dimensional data respectively.

1

Data Creation

Creating DataFrames from various sources: CSV, Excel, JSON files, and Python data structures.

2

Data Exploration

head(), tail(), info(), and describe() methods provide quick data insights and statistical summaries.

3

Data Selection

loc, iloc, at, and iat enable precise data access through labels and integer positions.

4

Data Cleaning

Handling missing data with isna(), dropna(), and fillna(). Removing duplicates ensures data quality.

5

Data Transformation

Apply, map, and lambda functions transform data. GroupBy operations and aggregations summarise information.

6

Data Combination

Merging, joining, and concatenating DataFrames. Reshaping with stack, unstack, melt, and pivot operations.

Module 3: Matplotlib - Static Data Visualisation

Line & Scatter Plots

Line plots show trends over time. Scatter plots reveal relationships between variables with customisable markers and colours.

Bar & Histogram

Bar charts compare categories. Histograms display data distributions across bins, revealing frequency patterns.

Pie Charts

Pie charts illustrate proportional relationships, showing how parts contribute to the whole with percentage labels.

Matplotlib's architecture separates Figure and Axes objects, enabling sophisticated plot composition. The pyplot interface provides MATLAB-like functionality, whilst the object-oriented approach offers greater control. Customisation options include colours, markers, line styles, labels, titles, legends, ticks, grid lines, and axis formatting. Annotations and text elements enhance communication. Multiple subplots with grid layouts enable comprehensive dashboards. Box plots and violin plots visualise statistical distributions effectively.

Module 4: Seaborn - Statistical Data Visualisation

Seaborn builds upon Matplotlib, providing high-level interfaces for statistical graphics. Its integration with Pandas DataFrames streamlines visualisation workflows whilst offering sophisticated default aesthetics.

Distributions

histplot, kdeplot, boxplot, and violinplot reveal data distributions and identify outliers effectively.

Categorical

barplot, countplot, and pointplot compare categories and show relationships between categorical variables.

Relational

scatterplot, lineplot, and relplot explore relationships between continuous variables across dimensions.

Matrix Visualisations

Heatmaps display correlation matrices and data tables with colour encoding. Clustermaps add hierarchical clustering for pattern discovery.

Advanced Plots

Regression plots (regplot, lmplot, residplot) show linear relationships. Pair plots and joint plots explore multivariate relationships comprehensively.

Facet grids enable multi-dimensional data exploration by creating subplot matrices. Seaborn's colour palettes and themes ensure professional, publication-ready visualisations with minimal customisation effort.

Module 5: Plotly - Interactive Data Visualisation

Plotly transforms static visualisations into interactive experiences. Users can hover for details, zoom into regions, pan across data, and select subsets dynamically, enhancing data exploration and presentation.

Basic Charts

Plotly Express provides concise syntax for line charts, scatter plots, and bar charts with automatic interactivity and responsive design.

Statistical Plots

Box plots, violin plots, and histograms reveal distributions. Interactive features enable detailed statistical exploration.

3D Visualisations

Three-dimensional scatter plots and surfaces visualise complex relationships across multiple dimensions with rotation capabilities.

Geographic Maps

Choropleth maps and geographic visualisations display spatial data patterns across regions and countries.

Animations

Animated visualisations show data evolution over time, revealing trends and patterns through dynamic transitions.

Financial Charts

Candlestick charts display financial data with open, high, low, and close values for market analysis.

SECTION 5

FullStack Python Framework Django

Module 1: Django Fundamentals & Project Setup

Django, a high-level Python web framework, enables rapid development of secure, maintainable websites. Its "batteries-included" philosophy provides comprehensive tools for common web development tasks.

MVT Architecture

Model-View-Template pattern separates data, logic, and presentation layers for clean code organisation.

Project Setup

django-admin startproject creates project structure. python manage.py startapp generates application modules.

URL Routing

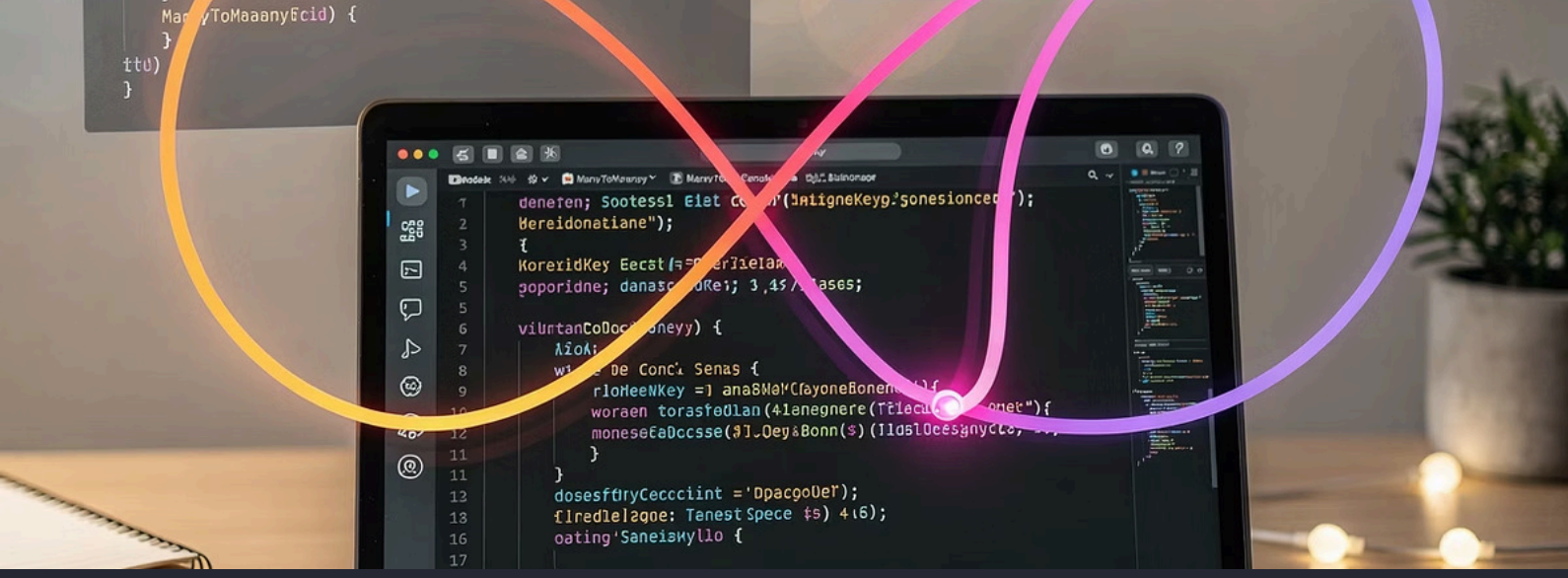
URL patterns map requests to views. Django's routing system enables clean, maintainable URL structures.

Development Server

Django's built-in development server enables rapid testing and iteration. The admin panel provides immediate database management capabilities.

Project Structure

Organised directory layout separates settings, URLs, and applications. Configuration management supports multiple environments.



Module 2: Models, ORM & Database Operations

Model Definition

CharField, TextField, IntegerField, DateField, and other field types define data structures with options like max_length, null, blank, and default.

Migrations

makemigrations creates migration files. migrate applies changes to database schema, enabling version-controlled database evolution.

CRUD Operations

Model.objects.create() adds records. all(), filter(), and get() retrieve data. update() and delete() modify and remove records.

ForeignKey

One-to-Many relationships link related records across tables.

ManyToManyField

Many-to-Many relationships enable complex associations between models.

OneToOneField

One-to-One relationships extend models with additional information.

Module 3: Views, Templates & Forms

Function-Based Views

FBVs handle GET and POST requests, render templates with context data, and process user input. They provide straightforward request-response handling for simple scenarios.

Template Engine

Template inheritance with {% extends %} and {% block %} enables reusable layouts. Template tags and filters transform data for display.

Static Files

Configuration for CSS, JavaScript, and images ensures proper asset serving in development and production environments.

Django Forms

forms.Form and ModelForm classes handle validation, rendering, and data processing. Crispy Forms enhances form presentation.

Security

CSRF protection prevents cross-site request forgery attacks. File upload handling with ImageField supports media content.

Module 4: Class-Based Views & Authentication

Class-Based Views (CBVs) provide reusable, object-oriented approaches to common patterns. Generic views reduce boilerplate code for standard operations.

CreateView

Handles object creation with automatic form generation and validation.

ListView

Displays multiple objects with pagination and filtering capabilities.

DetailView

Shows individual object details with customisable templates.

UpdateView

Modifies existing objects with form handling and validation.

DeleteView

Removes objects with confirmation and success redirection.

Django's built-in authentication system manages users, permissions, and sessions. User registration views create accounts. Login and logout views handle authentication. The @login_required decorator protects views, whilst LoginRequiredMixin secures CBVs. Session management maintains user state. Password reset flows enable account recovery through email verification.

Module 5: Django REST Framework

Django REST Framework (DRF) transforms Django applications into powerful APIs. It provides serialisation, authentication, permissions, and browsable API interfaces for modern web services.

Serializers

Convert complex data types to JSON. ModelSerializer automatically generates fields from models. Nested serializers handle relationships.

API Views

@api_view decorator creates function-based API endpoints. APIView class provides object-oriented approach with method handlers.

Generic Views

ListCreateAPIView and RetrieveUpdateDestroyAPIView reduce boilerplate for common patterns.

ViewSets & Routers

ViewSets combine related views. Routers automatically generate URL patterns for standard operations.

Authentication

Token authentication, session authentication, and JWT (JSON Web Tokens) secure API endpoints. Each method suits different use cases and security requirements.

Advanced Features

Permissions control access. Pagination manages large datasets. Filtering and searching enable data discovery. Swagger documentation provides interactive API exploration.

SECTION 6

Modern Python Framework FastAPI

Module 1: FastAPI Fundamentals & Setup

FastAPI represents the next generation of Python web frameworks, combining speed, modern Python features, and automatic documentation. Built on Starlette and Pydantic, it delivers exceptional performance through asynchronous capabilities.

3x	100%	Auto
Performance	Type Safety	Documentation
FastAPI matches Node.js and Go performance, significantly faster than traditional Python frameworks.	Complete type hints enable automatic validation, serialisation, and IDE support.	Swagger UI and ReDoc generated automatically from code annotations.

01

Environment Setup

Create virtual environment and install FastAPI with Uvicorn ASGI server.

02

First Endpoint

Define path operations using decorators: @app.get, @app.post, @app.put, @app.delete.

03

Run Server

Launch development server with automatic reload for rapid iteration.

Module 2-5: Advanced FastAPI Concepts

Parameters & Validation

Path parameters with type hints, query parameters with defaults, and Pydantic models for request bodies ensure automatic validation.

Database Integration

SQLAlchemy ORM with dependency injection manages database sessions. CRUD operations with Alembic migrations enable schema evolution.

Project Structure

APIRouter organises code into modular components. Repository pattern separates database logic. Environment variables configure deployments.

Password Security

Bcrypt hashing protects user passwords with industry-standard encryption.

OAuth2 Flow

OAuth2PasswordBearer dependency verifies tokens and extracts current user information.

1

2

3

4

JWT Tokens

JSON Web Tokens with expiration enable stateless authentication across requests.

Protected Routes

Authentication dependencies secure endpoints. Role-based access control manages permissions.

FastAPI's modern approach combines Python 3.6+ type hints with automatic validation, serialisation, and documentation generation. This comprehensive course equips you with skills spanning IT fundamentals, Python programming, SQL databases, data science libraries, and full-stack web development frameworks, preparing you for diverse roles in modern software development and AI applications.