Python Full Stack Development with AI Integration Syllabus

Build modern web applications with Python, Django, React and practical AI features. Industry-Oriented, Project-Based, Beginner Friendly.

Course Overview

Python Full Stack Development covers the skills required to build complete web applications, from responsive frontend interfaces to Python-based backend services and database integration. The programme follows the source syllabus structure: frontend development, Python programming, SQL, Django, CRUD, authentication, APIs and deployment fundamentals. The AI section adds practical ways to connect AI services to these applications without requiring advanced machine learning theory.

Technologies covered: Python, HTML / CSS, JavaScript, React.js, Django, Oracle SQL, REST APIs, GitHub, OpenAI API, Gemini API

Why learn Python Full Stack?

Learn frontend development with HTML5, CSS3, Bootstrap, JavaScript and React. Build backend applications using Python and Django. Work with SQL databases, ORM, CRUD operations and REST APIs. Understand version control and deployment workflow.

Why add AI to Python applications?

Add useful chatbot, summarisation, document and image-analysis features to web applications. Connect Python services to OpenAI and Gemini APIs. Integrate AI features into Flask, Django or FastAPI-based workflows. Learn practical API security, data handling and error management.

Course Roadmap

Stage 1 - Programming Fundamentals: Python syntax, data types, control flow, functions and OOP

Stage 2 - Front-End Development: HTML, CSS, Bootstrap and responsive interfaces

Stage 3 - Advanced JavaScript and React: modern JavaScript, React components, hooks and API calls

Stage 4 - SQL and Database Essentials: Oracle SQL, queries, joins, functions and views

Stage 5 - Web Development Basics: Django setup, apps, views, templates and forms

Stage 6 - Backend Development with Django: models, ORM, forms and application structure

Stage 7 - Database Integration: models, migrations, queries and persistent data

Stage 8 - Full CRUD and Authentication: create, read, update, delete, login and roles

Stage 9 - Building and Consuming APIs: Django REST Framework and API authentication

Stage 10 - Deployment and DevOps Basics: Git, EC2 deployment and production settings

AI - AI Integration: OpenAI / Gemini, chatbots, document AI and mini project

Course 01 - Front-End Development (6 Modules)

Course objective

Design responsive interfaces and interactive web experiences. Tools: HTML5, CSS3, Bootstrap, JavaScript and React, Visual Studio Code, Notepad++ and Google Chrome DevTools, GitHub for source-code version control.

1. HTML5

Basic HTML structure and semantic tags, Forms, tables and lists, Linking and embedding content, Multimedia tags: images, audio and video

Greens Technologies

2. CSS3

Selectors: element, class and ID, Box model: padding, margin and border, Layout techniques using Flexbox, Positioning: absolute, relative and fixed, Responsive design using media queries, Animations: transform, translate and keyframes

LAB: Create a responsive multi-section webpage using semantic HTML and CSS

3. Bootstrap

Selectors and components, Bootstrap grid system and layout techniques, Flexbox utilities and grid-based layouts, Responsive design: breakpoints and media queries

4. JavaScript

Variables and data types: numbers, strings, booleans, arrays and objects, Operators: arithmetic, logical and comparison, Control flow: if / else, switch and loops, Functions, parameter passing and return values, Arrays and array methods, object manipulation, DOM manipulation and event handling: click, hover and change

LAB: Build interactive forms and update webpage elements using JavaScript

5. Advanced JavaScript

Asynchronous programming: callbacks, promises and async/await, JavaScript modules: CommonJS and ES modules, Event handling, forms and routing, Higher-order functions

6. React.js

JSX syntax and functional / class-based components, Lists and keys, lifecycle methods and props, Conditional rendering and event handling in React, React Hooks: useState, useEffect, useRef, useContext and useReducer, Routing using React Router, API integration and data fetching using Axios

LAB: Build reusable React components and connect them to a sample REST API

Front-end projects

BMI Calculator based on user input, To-Do App to create, update and track daily tasks, Registration form validation, E-commerce platform with product listings and shopping carts, Weather dashboard using APIs to display weather updates.

Job placement training

Resume preparation and building a Naukri profile, Development placement training, Interview preparation: mock interviews, MCQs and group discussion.

Outcome: students can create responsive frontends and prepare to demonstrate their work.

Course 02 - Python Programming (13 Modules)

Course objective

Build Python programming skills from basics to advanced: functions, files, exception handling and object-oriented concepts.

1. Introduction to Python

What is Python? Features of Python, Installing Python and IDEs (e.g. PyCharm), Writing and running your first Python script, Python syntax and indentation

2. Basics of Python

Variables and data types, Input and output functions, Basic operators: arithmetic, comparison, logical and assignment, Type conversion and casting

Greens Technologies

3. Control Structures

Conditional statements: if, else and elif, Loops: for loop and while loop, Loop control statements: break, continue and exit

4. Data Types

Lists: creating and accessing elements, methods and slicing, Tuples: properties and uses, Sets: properties and operations, Dictionaries: key-value pairs, methods and operations

LAB: Solve programming exercises using conditions, loops and Python collections

5. Functions

Defining and calling functions, Function arguments: default, keyword and arbitrary, Return statement and recursion, Anonymous functions (lambda), Higher-order functions: map, filter and reduce

6. Modules and Packages

Importing modules, Creating and using your own modules

7. File Handling

Reading and writing files using open() and with, File modes: r, w, a, r+ and related modes, Working with Notepad, image, audio and video files, Excel files, ZipFile and folders, Handling file exceptions

8. Error and Exception Handling

Common exceptions and user-defined exceptions in Python, try, except, else and finally blocks

9. Object-Oriented Programming (OOPS)

Classes and objects, attributes and methods, Inheritance and constructors, Polymorphism and abstraction, Encapsulation

10. Types of Variables

Overview of variable types in Python, Local, instance and static variables

11. Types of Methods

Understanding methods in Python, Global, class and static methods

12. Date and Time

Working with date and time in Python

13. Project

Hands-on assignments and real-world applications

LAB: Complete trainer-guided Python exercises and a small practical application

Course 03 - SQL Oracle: Database Essentials (15 Modules)

Course objective

Learn relational database concepts and SQL query development: filtering, aggregation, functions, joins and reusable query structures.

1. Introduction to SQL

What is SQL? History and evolution of SQL, Importance of SQL in databases, SQL syntax and structure

Greens Technologies

2. Database, DBMS and RDBMS

What is a database? Database vs file system, Types of databases: relational, NoSQL and others, DBMS features and components, RDBMS and differences from DBMS

3. SQL Commands

DDL: CREATE, ALTER, DROP and TRUNCATE, DQL: SELECT, DML: INSERT, UPDATE and DELETE, DCL: GRANT and REVOKE, TCL: COMMIT, ROLLBACK and SAVEPOINT

4. Data Types

Numeric, string, date/time, binary and other data types

5. Constraints

Constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY, CHECK and DEFAULT, Indexes

6. WHERE Clause

Filtering data using WHERE, Operators: =, <>, >, <, BETWEEN, IN and LIKE, Logical operators: AND, OR and NOT, Using WHERE with multiple conditions

7. Aggregate Functions

COUNT(), SUM(), AVG(), MIN() and MAX()

8. GROUP BY Function

Purpose of GROUP BY and use with aggregate functions, HAVING clause vs WHERE clause, GROUP BY with multiple columns

9. Window Functions

ROW_NUMBER(), RANK() and DENSE_RANK(), LEAD() and LAG(), PARTITION BY vs GROUP BY

10. Single Row Functions

String: UPPER, LOWER, LENGTH, SUBSTRING and REPLACE, Numeric: ABS, ROUND, CEIL, FLOOR and POWER, Date: NOW, CURDATE, DATEADD, DATEDIFF and EXTRACT, Conversion: CAST and CONVERT

11. Joins

INNER JOIN, LEFT JOIN (LEFT OUTER JOIN), RIGHT JOIN (RIGHT OUTER JOIN), FULL JOIN (FULL OUTER JOIN), SELF JOIN and CROSS JOIN

12. Subqueries

Single-row, multi-row and correlated subqueries, Subqueries in SELECT, FROM and WHERE, Subqueries vs joins

13. Set Operators

UNION and UNION ALL, INTERSECT and MINUS (EXCEPT in some databases), Differences between joins and set operators

14. ORDER BY Clause

Sorting in ascending and descending order, ORDER BY with multiple columns and expressions, Using ORDER BY with LIMIT

Greens Technologies

15. Views

Creating views using CREATE VIEW, Updating and dropping views, Types of views, advantages and use cases

LAB: Write SQL queries to filter, join, aggregate and retrieve application data

Course 04 - Backend Development with Django (12 Modules)

Course objective

Build Python web applications using Django, connect application logic to a database, and prepare applications for real-world use.

1. Introduction to Django

What is Django? Overview and key features, Installing Django and setting up the environment, Creating the first Django project, Running the development server and understanding project structure

2. Creating an App and Views

Creating the first Django app and understanding the role of apps, Writing the first view and connecting it to a URL, Rendering views using HttpResponse

3. Templates and Static Files

Rendering templates with the render() function, Passing data to templates using context, Adding and managing CSS, images and JavaScript static files, Creating and styling HTML tables using CSS

4. Handling Forms and Basic GET Requests

Creating an HTML form for GET requests, Handling form submissions in views, Displaying form data in templates

5. Models and the Database

Creating models and fields, Applying migrations and database setup, ORM basics: querying data from the database, Creating views to fetch model data and display it in templates

6. Forms for Data Management

Using Django forms to add data to the database, Creating and using Django Form and ModelForm, Validating form data and displaying errors

7. Advanced Template Features

Template inheritance using block content, Displaying messages with Django's messaging framework

8. CRUD Operations

Creating, reading, updating and deleting data using Django views, Working with Django ORM and QuerySets for CRUD, Using the Faker library to seed the database with test data

LAB: Build a database-backed application with forms and complete CRUD workflows

9. Authentication System

Django's built-in authentication framework, User registration, login and logout, Implementing role-based access control

10. Pagination and Search

Adding pagination to display large datasets, Implementing keyword search to filter data

Greens Technologies

11. REST APIs

Introduction to Django REST Framework (DRF), Creating CRUD APIs using DRF, Implementing basic authentication in APIs

12. Version Control and Deployment

Introduction to Git and version control, Setting up and deploying a Django app on an EC2 instance, Configuring production-ready settings

LAB: Expose CRUD data through a REST API and review the deployment workflow

Course 05 - AI Integration (4 Modules)

Course objective

Understand the core concepts behind modern generative AI tools, connect Python applications to AI providers safely, and build tool-using agents and MCP servers. Learning path: Generative AI fundamentals, Python integration with AI, AI agents and tool calling, Model Context Protocol (MCP).

1. Generative AI Fundamentals

What is Generative AI?, Large Language Models (LLMs) and how they work at a practical level, Overview and practical comparison of ChatGPT, Gemini and Claude, Tokens, context windows and temperature, System prompts, user prompts and prompt engineering, Structured output using JSON, Function calling and tool calling, Understanding hallucinations and AI limitations

LAB: Explore the same task with different AI assistants and compare their responses

Outcome: students understand the basic concepts, terminology and limitations of generative AI, and can write clear prompts to obtain structured responses.

2. Python Integration with AI

OpenAI API integration using Python, Gemini API integration, Calling AI models using SDKs and HTTP requests, Sending prompts and receiving responses, Parsing and working with JSON responses, Streaming responses, Error handling, token usage and API costs, Environment variables and API security

LAB: Build an AI-powered text generator and summarizer using Python

Basic integration flow: Python application (prepare prompt and input) -> AI API request (send request securely) -> Model response (parse and display output)

Practical build - AI Text Generator and Summarizer: Create a Python application that accepts user text, sends a prompt to OpenAI or Gemini, receives the response, parses the output and displays generated or summarized content. Keep API credentials in environment variables

3. AI Agents and Tool Calling

What is an AI agent?, Difference between a chatbot and an AI agent, Agent workflow and decision-making, Function calling, Connecting Python functions to AI, Giving AI access to external tools, Multiple tool execution, Human approval before important actions, Building a simple AI agent

LAB: Create a simple Python agent that selects and calls an appropriate function based on a user's request

Agent workflow: User request (task or question) -> AI agent (chooses a tool) -> Python tool (runs approved function) -> Result (returns response)

Important practice: Students learn to keep tool permissions limited and require human approval before actions that could have important consequences

Greens Technologies

4. Model Context Protocol (MCP)

What is MCP?, Why MCP was introduced, MCP architecture, MCP Host, Client and Server, MCP tools, resources and prompts, How an AI client communicates with an MCP server, Creating an MCP server using Python, Connecting MCP servers to an AI client

LAB: Create a basic MCP server in Python, expose a simple tool and connect it to a compatible AI client

MCP architecture at a glance: MCP Host (AI application), MCP Client (communicates using MCP), MCP Server (Python tools, resources and prompts)

Key components: Host: the AI application. Client: manages the connection. Server: exposes tools, resources and prompts that the client can access through MCP

Small Project: Python AI Task Assistant

Apply the four AI modules in one compact Python project. Build a small Python application where a user enters a task or question. The application sends the request to an AI model, and the AI agent can select from a limited set of Python tools. Expose one or more of those tools through a basic MCP server and connect it to a compatible AI client.

Project build steps

Create a Python application and configure the AI provider credentials securely, Send a user prompt to the selected AI model and display its response, Define Python functions that perform simple, safe tasks, Connect function calling so the AI agent can select an appropriate tool, Create a basic MCP server and expose a selected Python tool, Connect the MCP server to a compatible AI client and test the workflow, Demonstrate the application and explain its request, tool-call and response flow.

Student deliverables

Working small Python AI project, Python source code and a short project walkthrough, Demonstration of AI response, tool calling and MCP connection.

Learning outcomes

By the end of this section, students can explain generative AI fundamentals, integrate OpenAI or Gemini with Python, build a simple tool-using AI agent, and create and connect a basic Python MCP server.

Greens Technologies