

appin tech
COIMBAT



FULL STACK PYTHON COURSE

SYLLABUS & FEES



Why Choose **Appin Technology** for Learning **Python**?

Python continues to be one of the most powerful and in-demand programming languages across industries. At Appin Technology, Coimbatore, we go beyond teaching syntax we equip you with full-stack development skills that make you job ready. Our career-focused training covers everything from Core Python to advanced modules like React, DevOps, Testing, Deployment, and AI integration, ensuring you're ready to build real-world applications and succeed in interviews.

Learn from the No 1 Python Training Institute in Coimbatore with 100% Placement Support.

Course **Highlights**

100% Practical Training

Industry-Trained Mentors

Beginner to Advanced Level

Real-Time Projects

Course Completion, Project Experience and Internship Certificate

Internship & Placement Support

Classroom + Practice Lab Environment

10000+ Students Trained Successfully

50+ Active Hiring Partners

Who Can Join?

- ▶ College Students
- ▶ Job Seekers
- ▶ Non-IT to IT Career Switchers

Full Stack Python Course **Syllabus** 8hrs

Core Python

Data Types, Functions, Loops, Conditionals

Introduction to fundamental programming constructs.

Subtopics:

- Built-in data types: int, float, string, list, tuple, dict, set
- Type conversion and casting
- Defining and calling functions
- Conditional statements: if, else, elif
- Looping constructs: for, while, and control flow (break, continue, pass)

OOP (Classes, Objects, Inheritance)

Understanding Object-Oriented Programming in Python.

Subtopics:

- Defining classes and creating objects
- `__init__()` constructor and instance variables
- Encapsulation, Abstraction
- Inheritance and Polymorphism
- Method Overriding, `super()`
- Special methods like `__str__`, `__repr__`, etc.

File Handling

Reading from and writing to files.

Subtopics:

- Opening and closing files (`open()`, `with`)
- Modes: r, w, a, r+
- Reading methods: `read()`, `readline()`, `readlines()`
- Writing and appending data
- Working with CSV/JSON files

Exception Handling

Gracefully managing runtime errors.

Subtopics:

- try, except, else, finally blocks
- Handling specific exceptions
- Raising custom exceptions using `raise`
- Creating user-defined exceptions

Modules and Packages

Code organization and reuse.

Subtopics:

- Importing standard modules (math, random, os, etc.)
- Creating and using custom modules
- `__init__.py` and Python packages
- pip for installing external packages

Web Framework – Django

Django Project and App Structure

Understanding how Django is structured.

- Managing projects with django-admin
- Creating apps using manage.py
- Folder layout and configurations

ORM (Object Relational Mapping)

Interacting with databases using Python objects.

- CRUD operations using models
- QuerySets and filtering
- Aggregations and annotations

Admin Panel & User Panel

Managing application from built-in admin.

- Customizing Django admin interface
- Registering models
- Creating superuser and managing users

REST APIs with Django REST Framework (DRF)

Building backend APIs.

- Serializers and ViewSets
- Routers and URL patterns
- Class-based vs function-based views

Models, Views, Templates (MVT Pattern)

Core architecture of Django.

- Defining models and database schema
- Writing views (function-based or class-based)
- Connecting HTML templates with views
- Using template tags and filters

Forms and Validation

Creating and processing web forms.

- Django Forms vs ModelForms
- Field types and input validation
- Custom error messages

Middlewares

Request and response processing.

- What are middlewares
- Creating custom middleware
- Using built-in ones (SecurityMiddleware, etc.)

Authentication & Authorization (Login/Logout, JWT)

User management in web apps.

- Session-based authentication
- Token-based authentication (JWT with DRF)
- Permissions and role-based access

Database

SQL (PostgreSQL / MySQL)

CRUD Operations

- SELECT, INSERT, UPDATE, DELETE

Joins, Subqueries

- INNER JOIN, LEFT JOIN, RIGHT JOIN
- Nested queries and aliasing

DDL, DML

- DDL: CREATE, ALTER, DROP
- DML: INSERT, UPDATE, DELETE

NoSQL (MongoDB)

MongoDB Basics (CRUD, Collections, Documents)

- NoSQL structure and differences from RDBMS
- Creating collections, inserting and querying documents
- MongoDB with Python using PyMongo or MongoEngine

DevOps, Deployment & Tools

Version Control

Git & GitHub

- Initializing repo, commits, branches
- Pull/push/clone
- GitHub collaboration (pull requests, issues)

ORM in Django

QuerySet APIs

- .filter(), .exclude(), .get()
- .all(), .order_by(), .values()

Model Relationships (OneToOne, ForeignKey, ManyToMany)

- Setting up relations using fields
- Reverse lookups and related_name

Migrations

- Creating migrations
- Applying them (makemigrations, migrate)
- Managing schema changes

Deployment

Deploying Django App

- Deploying application on anyone platform
(Heroku, Render, or AWS Ec2)
- Running migrations in production

Environment Variables

- Using .env files
- Keeping secrets secure (API keys, DB creds)

CI/CD Basics (Optional Advanced)

Docker

- Creating Dockerfiles
- Building and running containers
- Docker Compose for multi-container setups

GitHub Actions

- Creating workflows for testing and deployment
- CI pipelines for automated builds

Cloud (AWS)

- **AWS EC2 – Hosting Django apps on virtual servers**
- **AWS S3 – Storing static and media files**
- **AWS RDS – Managed relational databases like PostgreSQL or MySQL**

Other Must-Know Concepts

REST API Concepts

- HTTP methods (GET, POST, PUT, DELETE)
- Status codes and RESTful conventions

JSON Handling

- Serializing/deserializing with json module
- Working with JSON in APIs

AJAX

- Sending asynchronous requests from frontend
- Updating UI without page reload

Postman for API Testing

- Testing REST APIs with headers, params, and auth
- Automation with collections

CORS

- Cross-Origin Resource Sharing issues and solutions
- Using Django CORS headers

Security (CSRF, XSS basics)

- Enabling CSRF protection in forms
- Understanding XSS vulnerabilities

Unit Testing (PyTest)

- Writing and running test cases
- Using Django's test framework or PyTest

JWT or OAuth for Auth

- Secure token generation and verification
- Role-based access control

Data Structures and algorithm

Time and Space Complexity

- Big O Notation ($O(n)$, $O(1)$, $O(\log n)$, etc.)

Arrays and Strings

- Basics of arrays (1D, 2D)
- Searching and sorting arrays
- Sliding Window Technique
- Prefix Sum, Difference Arrays

Hashing (HashMap / HashSet)

- Frequency Counting
- HashMap vs TreeMap

Recursion and Backtracking

- Factorial, Fibonacci
- Subsets, Permutations

Sorting Algorithms

- Bubble, Selection, Insertion (basic)
- Merge Sort, Quick Sort (optional)
- Sorting before two-pointer
- Sorting before binary search

Searching Algorithms

- Linear Search, Binary Search
- Binary Search Variants:
- First/Last Occurrence
- Search in Rotated Array
- Median of Two Arrays

Two Pointers and Sliding Window

- Two Sum (Sorted)
- Container With Most Water
- Longest Substring Without Repeating Characters

Stacks and Queues

- Stack using Array
- Next Greater Element, Balanced Parentheses
- Queue, Circular Queue
- Implement Stack using Queue and vice versa

Linked Lists

- Singly, Doubly Linked Lists

Debugging and optimization

Introduction to Debugging

- What is Debugging?
- Common Python Errors:
- SyntaxError, NameError, TypeError, IndexError, KeyError
- Find bugs in short code snippets and fix them

FRONTEND

HTML (HyperText Markup Language)

HTML is the backbone of web pages used to structure content.

HTML Tags and Attributes

- Basic tags: div, span, h1 to h6, p, a, img, ul, ol, li, table, etc.
- Global attributes: id, class, style, title, alt
- Structural tags: header, nav, main, section, footer
- Inline vs block-level elements

Forms and Validation

- Creating input forms using form, input, textarea, select, button
- Input types: text, email, password, date, number
- Built-in validations: required, minlength, pattern, etc.
- Form submission using action and method

Semantic HTML

- Using meaningful tags for better accessibility and SEO
- Tags: article, aside, section, figure, figcaption, time, mark, etc.

Code Optimization

Loop Optimization:

- Avoid unnecessary loops
- Use built-in functions (e.g., sum(), min()) instead of custom loops
- Using List Comprehensions
- Efficient String Operations (avoid + in loops, use .join)

CSS (Cascading Style Sheets)

CSS styles the HTML structure for appearance, layout, and responsiveness.

Selectors, Box Model, Positioning

- Element, class, ID, attribute selectors
- Box model: margin, border, padding, content
- Positioning: static, relative, absolute, fixed, sticky

Flexbox, Grid

- Flexbox: one-dimensional layout control
Properties: display: flex, justify-content, align-items, flex-direction, etc.
- CSS Grid: two-dimensional layout system
Properties: grid-template-columns, grid-gap, grid-area

Media Queries (Responsive Design)

- Creating responsive designs for mobile, tablet, and desktop
- Mobile-first and desktop-first approaches

CSS Frameworks: Bootstrap / Tailwind

- Bootstrap: utility and component classes
- Grids, cards, navbars, buttons
- Tailwind CSS: utility-first framework
- Custom styling using utility classes (flex, text-lg, bg-blue-500, etc.)

JavaScript (JS)

JS is the programming language of the web, enabling interactivity and logic.

Variables, Data Types, Operators

- Declaring variables using var, let, and const
- Primitive types: string, number, boolean, null, undefined
- Objects and arrays
- Arithmetic, logical, and comparison operators

DOM Manipulation

- Accessing elements using getElementById, querySelector, etc.
- Changing content: innerText, innerHTML
- Modifying styles and attributes
- Creating and removing elements dynamically

Functions, Events

- Function declarations, expressions, arrow functions
- Event handling: onclick, addEventListener
- Event types: click, change, input, mouseover, keypress

ES6 Features

- let and const
- Arrow functions (=>)
- Template literals (Hello \${name})
- Destructuring, spread/rest operators
- Promises and async/await
- Modules: import/export

Fetch API / Axios

- Making HTTP requests to external APIs
- Handling responses with Promises
- GET, POST, PUT, DELETE requests
- Error handling and loading states

Form Validation

- Client-side form validation using JS
- Custom validation rules
- Real-time feedback with events like on input or on blur

Single Page Applications (SPA) Basics

- Concept of SPAs: dynamic page updates without full reload
- History API and URL handling
- Benefits of SPA over traditional MPA

Frontend Framework – React

React is a popular JS library for building component-based Uis.

Components, Props, State

- Function and Class components
- Props: passing data between components
- State: managing dynamic data inside components
- Component re-rendering and lifecycle

Hooks (useState, useEffect)

- useState: managing component-level state
- useEffect: side effects (e.g., fetching data, timers)
- Cleanup and dependency array
- Other hooks (intro): useRef, useContext, useMemo, etc.

Routing (React Router)

- React-router-dom for navigation
- BrowserRouter, Route, Link, useNavigate
- Dynamic routes and URL parameters

Form Handling

- Controlled components (value and onChange)
- Form submission and validation in React
- Using libraries like Formik or React Hook Form (optional)

Conditional Rendering

- Using if statements, ternary operators
- && logical short-circuiting
- Conditional UI elements

API Integration

- Calling REST APIs from React using fetch or axios
- Updating UI based on API responses
- Error and loading state management

Redux (Advanced)

- Centralized state management
- Actions, Reducers, Store
- Middleware like redux-thunk for async logic
- Integration with React using react-redux

AI Integration in Fullstack Applications

24/7 Support Assistance

AI technologies enhance this support by automating responses, resolving common queries, and escalating complex issues.

Key Features:

- **AI-Driven Chatbots:** Deploy intelligent conversational agents that can understand, interpret, and respond to user queries in real-time.
- **Real-Time Query Handling:** Offer solutions instantly without human intervention, reducing response time and operational costs.

Chat Bot and Query Resolver

Developing a smart assistant to handle common queries, guide users, and support issue resolution

Key Features:

- Use REST APIs or WebSocket to connect chatbot with backend.
- Store conversation logs and user interactions in database.

Code and Text Generator

Use Cases:

- Auto-generate frontend/backend code based on user input.
- Draft emails, summaries, or blog content.
- Assist with debugging or code explanation.

Tools:

- OpenAI GPT APIs for text/code generation.
- Frontend interfaces integrated with Backend to input prompts and display results.

Generative AI

- Introduction to LLMs (e.g., OpenAI GPT, Hugging Face)
- Use cases in full-stack apps (summarization, translation, etc.)

Testimonials



Magudapathi

Appin Technology's Python Development Course is a comprehensive training program that offers a solid foundation in Python programming. The instructors are knowledgeable and provide hands-on experience, making it perfect for beginners. However, some advanced topics could be covered in more depth.



Varun Karthick

I recently completed the Python course at Appin Technology, and it was an incredible experience! The curriculum was comprehensive and well-structured, covering everything from basic syntax to advanced concepts. This course has given me a strong foundation in Python programming.



Kiruthika

Appin Technology's Python course exceeded my expectations in every way. The hands-on projects and practical exercises were especially helpful in reinforcing my learning. The faculty members were patient and skilled at explaining complex topics in a way that was easy to understand. Thank you



Santhosh

The projects and assignments were challenging but rewarding, helping me to gain a solid understanding of Python programming. I now feel prepared to apply these skills in my career and continue learning. Thank you, Appin Technology!



Sujitha

The Python course at Appin Technology was an incredible experience. The curriculum was comprehensive, and the instructors were knowledgeable and approachable. The hands-on projects and real-world applications helped me grasp the concepts quickly.



Muthu Selvam

Completing the Python course at Appin Technology was a turning point in my career. The hands-on approach and the focus on practical applications made learning enjoyable and effective. The projects and assignments helped me build a strong portfolio. I

About **Appin Technology**


Appin Technology Coimbatore is a premier Python training institute that offers comprehensive courses designed to equip students with the essential skills required for mastering Python programming. Located in Coimbatore, this institute is renowned for its practical approach to learning, industry-relevant curriculum, and experienced trainers.

The Python training at Appin Technology covers a wide range of topics, from basic syntax and data structures to advanced concepts such as web development, data analysis, and machine learning. Each module is meticulously designed to ensure students gain hands-on experience through real-world projects and assignments.

In addition to Python, Appin Technology also provides training in various other technologies, making it a hub for aspiring tech professionals. The institute's commitment to quality education, coupled with its state-of-the-art facilities and supportive learning environment, makes it an ideal choice for anyone looking to advance their career in the tech industry.

Contact **Info**

 144, Floor, Sengupta Street, Ram Nagar, Coimbatore – 641009.

 +91 77080 40308 | +91 90256 20238

 support@appincoimbatore.com

 www.appincoimbatore.com