InLustro
Your Learning Comrade

ADVANCED PYTHON VALUE ADDED PROGRAM FOR STUDENTS OF ViMEET - Phase 1 2026

PROGRAM DURATION:

- 6 days training with 6 hours per day
- followed by online project supervision throughout the semester
- 2 offline external project reviews

MODE OF DELIVERY: Hands-on and in-person

TENTATIVE TRAINING START DATE:

- 27th January 2026

CURRICULUM & PROJECTS:

Target Audience: Third-Year Engineering Students
Program Duration: 6 Days (6 Hours/Day) = **36 Hours**

Day 1 – Deep Dive into Python Internals & Best Practices

Session 1: Python Advanced Concepts

- Memory management & Garbage Collection
- Mutability vs Immutability
- Deep Copy vs Shallow Copy
- `*args`, `**kwargs` usage
- Decorators and Closures (Real-world examples)

Session 2: Object-Oriented Programming (OOP) – Pro Level

- Advanced OOP Principles
- Dunder/Magic Methods (`__str__`, `__repr__`, `__eq__`, etc.)
- Composition vs Inheritance
- Abstract Classes, Interfaces (`abc` module)

Hands-On Practice:

- Build a custom data container class using OOP.
- Create a mini project to simulate a banking transaction system using OOP.

Day 2 – File Handling, Exception Handling, and Best Practices

Session 1: Advanced File Operations

- Reading and writing JSON, CSV, XML files
- Working with ZIP, tar files using `zipfile`, `tarfile`
- File locking, error handling during file operations

Session 2: Robust Exception Handling

- Custom Exception classes
- Logging vs Raising Errors
- `try-except-else-finally` deep dive
- Logging Framework (`logging` module)

Hands-On Practice:

- Build a File-based CRUD app (Student Records Manager).
- Exception Handling Simulator — Handling multiple failure points gracefully.

Day 3 – Working with APIs, Data Parsing, and Web Scraping

Session 1: API Consumption

- Consuming Public APIs (GET, POST)
- Authentication handling (Bearer Token, OAuth2 intro)
- Introduction to Python `requests`, `http.client`

Session 2: Web Scraping Essentials

- Parsing HTML with `BeautifulSoup`
- Automating browsing with `Selenium`
- Introduction to Async Scraping (`aiohttp`)

Hands-On Practice:

- Build a Weather App by integrating OpenWeatherMap API.
- Scrape Job Postings (e.g., from Indeed) into a structured CSV.

Day 4 – Data Handling, Analysis, and Visualization with Python

Session 1: Python for Data Analysis

- Introduction to `Pandas` and `NumPy`
- Data wrangling techniques
- Handling missing data, merging, reshaping datasets

Session 2: Basic Visualization

- Quick plots using `Matplotlib` and `Seaborn`
- Creating basic dashboards

Hands-On Practice:

- Analyze and visualize a COVID-19 Dataset or E-commerce Sales dataset.
- Create a dashboard showing real-time COVID statistics.

Day 5 – Introduction to Async Programming and Design Patterns

Session 1: Asynchronous Programming

- Need for Async: CPU-bound vs I/O-bound
- `asyncio`, `await`, `async` basics
- Writing concurrent code in Python

Session 2: Essential Design Patterns in Python

- Singleton, Factory, Observer Patterns
- Best practices with examples

Hands-On Practice:

- Build an Async Web Scraper.
- Implement Factory Design Pattern to simulate a Restaurant Ordering System.

Day 6 – Advanced Python Concepts for Industry Applications

Session 1 (Morning Slot – 3 Hours): Advanced Libraries and Tools

Working with Advanced Python Libraries

- Introduction to `SQLAlchemy` for Database ORM
- Introduction to `Pydantic` for Data Validation
- Introduction to `FastAPI` (lightweight)

InLustro
Your Learning Comrade

  - How APIs are built faster compared to Flask
  - Building your first RESTful API with FastAPI

Code Quality, Testing & Packaging

- Writing Unit Tests with `unittest` and `pytest`
- Setting up a Basic Test Case
- Code Linting & Formatting (using `flake8`, `black`)
- Introduction to creating and packaging a simple Python module

Hands-On Practice:

- Build a basic User Registration API using FastAPI
- Write unit tests for API routes

Session 2 (Afternoon Slot – 3 Hours): Performance Tuning and Security

Performance Optimization Techniques

- Time & Space Complexity Review (Python specific)
- Profiling Code using `cProfile` and `timeit`
- Memory Optimization using Generators & Iterators

Security Best Practices in Python Applications

- Input validation and data sanitization
- Avoiding common security flaws (SQL Injection, XSS basics)
- Secure API building tips
- Managing Secrets using Environment Variables (`dotenv`)

Hands-On Practice:

- Refactor an inefficient Python script to optimize runtime and memory.
- Harden an API endpoint for secure user data handling.

---

15 Industry-Aligned Advanced Python Projects

---

1. Microservice API Rate Limiter

- Build a distributed rate-limiting service (like what APIs like Stripe, Twilio use).

- Use Redis or Memcached for token-bucket algorithms.
- Stack: Python (FastAPI), Redis, Docker, JWT Auth.

---

2. Real-Time Log Monitoring & Alert System

- Build a system to monitor server logs, detect anomalies, and send real-time alerts (email/SMS).
- Bonus: Integrate basic ML for anomaly detection.
- Stack: Python, Kafka (for log streaming), ElasticSearch.

---

3. Distributed Task Queue Framework

- Build a custom lightweight version of Celery — manage background tasks across multiple worker nodes.
- Stack: Python (asyncio), RabbitMQ/Redis.

---

4. Personalized AI News Aggregator

- Build an RSS feed reader that **classifies** news articles and **personalizes** them based on user behavior.
- Stack: Python, scikit-learn/NLP, Streamlit, MongoDB.

---

5. Multi-Client Chat Application with End-to-End Encryption

- Develop a secure real-time chat app using socket programming + RSA/AES encryption.
- Stack: Python (socket, cryptography), WebSocket (optional).

---

6. Smart Document Search Engine

- Build a mini-Google for PDFs or text files using embeddings (e.g., Sentence Transformers).
- Semantic search across thousands of documents.
- Stack: Python, FAISS (Facebook AI Similarity Search).

---

7. AI-Powered Auto-Resume Screening API

- Build a REST API that screens resumes against job descriptions and ranks them — completely automated!
- Add Explainability (show why candidate scored high/low).
- Stack: Python, Transformers (BERT), Flask/FastAPI.

---

8. IoT Device Simulator + Data Ingestion System

- Simulate 1000+ IoT devices sending dummy sensor data.
- Store, process and visualize real-time IoT data streams.
- Stack: Python (asyncio), MQTT, InfluxDB, Grafana.

---

9. Self-Healing Kubernetes Monitoring Bot

- Build a bot that watches Kubernetes pods and auto-heals common crashes (restart, notify, recreate).
- Stack: Python (kubernetes-client SDK), Prometheus.

---

10. Personal AI Research Assistant

- Scrape new arXiv papers, summarize them, and email a digest to the user every day.
- Bonus: Build a Slack or Telegram bot for it.
- Stack: Python, HuggingFace Transformers, Flask, SMTP API.

---

11. Advanced Fraud Detection Simulator

- Simulate transactions and detect fraud patterns using Machine Learning (isolation forests, autoencoders).
- Stack: Python, scikit-learn, PyCaret.

---

12. Auto Infrastructure as Code Generator

- Build a system that reads a cloud architecture diagram and outputs Terraform scripts!
- Bonus: Add LLMs for architecture interpretation.
- Stack: Python, Terraform, OpenAI API.

---

13. Healthcare Predictive Analytics Engine

- Predict patient readmission chances from historical data (real Kaggle datasets exist).
- Stack: Python, TensorFlow/PyTorch, XGBoost.

---

14. Voice-to-SQL Query Engine

- Speak into the mic, and system converts it to SQL queries that hit a real database.
- Stack: Python (SpeechRecognition), NLP Parsing, MySQL.

---

15. Secure API Gateway with Threat Detection

- Build an API proxy that filters, logs, and blocks malicious traffic (SQLi, XSS attacks detection).
- Stack: Python (FastAPI, regex), OWASP standards.

Please note: *The proposed list and curriculum is only indicative, and will be modified to fit the requirements of the students at the institution. Estimated program delivery durations may vary based on client approvals and the actual length of the course.*