

## 2-Month Python Training Program, Feb, 25

### **Training Schedule**

#### **Month 1: Git, Python, SQL, Multi-Threaded Media Review System**

<b>Topics</b>	<b>Sub-Topics</b>	<b>Days Required</b>	<b>Date</b>	<b>Session Recordings/Resources</b>
Git Basics	Cloning, Branching , Merging, Rebase	1 Day	17 Feb 20 25	#00 GIT Concepts.mp4
Python Fundamentals	Syntax, Data Types, Control Flow, Functions , Modules	3 Days	20 21 22 23 24 25	#01 Python_Introduction(DataType.mp4), #02 Python DataType_Session.mp4, #04 Python_Functions.mp4, #05 Python_Iterator_Generator_Container.mp4, #06 Python_Decorators.mp4, #09 Python_Exceptions_ContextManagers.mp4
Python OOP	Classes, Inheritance, Polymorphism, Encapsulation	2 Days	24 25	#07 Python_OOPS_Session1.mp4 #08 Python_OOPS_Session2.mp4
Debugging & Logging	Debugging Techniques, Logging Frameworks	1 Day	25 Feb 20 25	<ol style="list-style-type: none"><li>1. <a href="https://sematext.com/blog/python-logging/">https://sematext.com/blog/python-logging/</a></li><li>2. <a href="https://realpython.com/python-logging/">https://realpython.com/python-logging/</a></li></ol>

Multithreading & Multiprocessing	Threading Module, Multiprocessing, AsyncIO	2 Days	27 Feb 2025	#10 MultiThreading_MultiProsessing_Session1.mp4, #11 MultiThreading_MultiProcessing_Session2.mp4
Unit Testing	Pytest, Unittest, Mocking	1 Day	28 Feb 2025	#12 Pytest_Session1.mp4, #13 Pytest_Session2.mp4
Packaging & Code Quality	Virtual Environments, PEP8, Linting	1 Day	03 Mar 2025	<ul style="list-style-type: none"> <li>1. <a href="https://realpython.com/python-code-quality/">https://realpython.com/python-code-quality/</a></li> <li>2. <a href="https://www.digitalocean.com/blog/top-python-best-practices-for-better-code-quality/">https://www.digitalocean.com/blog/top-python-best-practices-for-better-code-quality/</a></li> <li>3. <a href="https://packaging.python.org/en/latest/tutorials/packaging-projects/">https://packaging.python.org/en/latest/tutorials/packaging-projects/</a></li> <li>4. <a href="https://www.pyopenSci.org/python-package-guide/tutorials/intro.html">https://www.pyopenSci.org/python-package-guide/tutorials/intro.html</a></li> </ul> #03 Python_VirtualEnv.mp4
SQL	CRUD, Joins, Aggregations, Indexing, Transactions, Views, Stored Procedures	4 Days	07 Mar 2025	#14 SQL.mp4
Mini Project: Multi-	SQLite, Redis Caching,	5 Days	14 Mar	See The details below

Threaded Media Review System	Multithreading, Design Patterns	20	25	
------------------------------	---------------------------------	----	----	--

## Month 1 Mini Project: Multi-Threaded Media Review System

### Objective

Develop a **CLI-based Media Review System** that allows users to **review movies, web shows, and songs**, store ratings, and retrieve recommendations. This project will test **Git, Python, SQL, Caching, Multithreading, and Design Patterns**.

### Technologies Used

- **Python** (Core development)
- **SQLite** (Relational Database for data storage)
- **Redis** (Caching frequently accessed reviews)
- **Multithreading** (Handling multiple review submissions concurrently)
- **Factory Pattern** (Managing different media types)
- **Observer Pattern** (Real-time updates on new reviews)
- **Git** (Version control and collaboration)

### Features & Functionalities

- **User Management:** Add and track users reviewing media.
- **Media Storage & Reviews:** Users can review and rate Movies, Web Shows, and Songs.
- **Multithreading for Concurrent Submissions:** Ensures multiple users can submit reviews simultaneously.
- **Redis Caching for Fast Data Retrieval:** Frequently accessed reviews are stored in memory.
- **Factory Pattern for Media Types:** A structured approach to handling different media categories.
- **Observer Pattern for Notifications:** Users get alerts when a new review is added for their favorite media.
- **Git Version Control:** Maintain structured workflow using Git best practices.

### CLI-Based Commands

<b>Operation</b>	<b>Command</b>
View All Media	<code>python media_review.py --list</code>
Add a Review	<code>python media_review.py --review &lt;media_id&gt; &lt;rating&gt; &lt;comment&gt;</code>
Search by Title	<code>python media_review.py --search &lt;title&gt;</code>
Get Top-Rated	<code>python media_review.py --top-rated</code>
Get Recommendations	<code>python media_review.py --recommend &lt;user_id&gt;</code>

**Month 2: Docker, DSA, Software Engineering, Web Frameworks, Pub/Sub, Advanced Real-Time Discussion Forum**

<b>Topics</b>	<b>Days Required</b>	<b>Date</b>	<b>Session Recordings/Resources</b>	<b>Status</b>
Docker	2 days	18 Mar 2025	#17 Docker_Session1.mp4 #18 Docker_Session2.mp4	
Data Structures & Algorithms	5 Days	25 Mar 2025	Refer learn portal	
Software Engineering Best Practices	4 Days	31 Mar 2025	<a href="https://testdriven.io/blog/clean-code-python/">https://testdriven.io/blog/clean-code-python/</a> , <a href="https://github.com/heykarimoff/solid.python">https://github.com/heykarimoff/solid.python</a> , <a href="https://github.com/faif/python-patterns">https://github.com/faif/python-patterns</a>	

Python Web Frameworks	5 Days	07 Apr 2025	#19 Flask_Session1.mp4 #20 Flask_Session2.mp4 #FastAPI_Session.mp4	
Mini Project: <b>Book Rental Microservice</b>	4 Days	11 Apr 2025	See Details Below	

### Project Details :

#### Project Title: **Book Rental Microservice**

#### Description:

Build a microservice for managing a simple book rental system. The system should include two main services:

1. **Book Service** - Handles book information (CRUD operations).
2. **User Service** - Manages user information (CRUD and rental history).

Both services should communicate through **REST APIs**.

#### Features:

##### **Book Service:**

- Add, update, delete, and list books.
- Each book should have fields like:
  - id (UUID)
  - title
  - author
  - genre
  - available\_copies

##### **User Service:**

- Add, update, delete, and list users.
- Rent a book:

- Update the available\_copies in the Book Service.
- Save the rental record to the database.
- Fields for users:
  - id (UUID)
  - name
  - email
  - rented\_books (store book IDs or details).

## Tech Stack:

- **Framework:** FastAPI
- **Database:** SQLite or PostgreSQL (depending on comfort level).
- **Communication:** REST APIs.
- **ORM:** SQLAlchemy or Tortoise ORM.

## Requirements:

### 1. Database Design:

- Create a separate database for each service (simulating microservices).
- Define tables for books, users, and rentals.

### 2. Endpoints:

- Implement endpoints for all CRUD operations.
- Add a rent\_book endpoint in the **User Service** that calls the **Book Service** to decrement available\_copies.

### 3. Validation:

- Validate input data using Pydantic models.
- Ensure users cannot rent a book if available\_copies is 0.

### 4. Asynchronous Programming:

- Use async def for all database and API calls.

### 5. Optional Features (if time permits):

- Add JWT-based authentication for sensitive operations (like renting books).
- Include Docker support for both services.

4 MONTHS TRAINING PROGRAMMING + 2 MONTHS

**Month 3: AWS, Relational & NoSQL Databases, CI/CD, AI-Powered Automated Resume Screener**

Topics	Days Required	Date	Session Recordings/Resources	Status
AWS	4 Days	03 Apr 2025	<a href="https://learn.epam.com/detailsPage?id=63188ae7-095a-45a7-9e7f-21bcba61d4">https://learn.epam.com/detailsPage?id=63188ae7-095a-45a7-9e7f-21bcba61d4</a>	
Relational Databases	3 Days	08 Apr 2025	<p>Database design : <a href="https://support.microsoft.com/en-us/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5">https://support.microsoft.com/en-us/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5</a></p> <p>SQL Databases Comparison : <a href="https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems">https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems</a></p> <p>ACID Properties : <a href="https://www.geeksforgeeks.org/acid-properties-in-dbms/">https://www.geeksforgeeks.org/acid-properties-in-dbms/</a></p> <p>Postgres Isolation Level : <a href="https://www.postgresql.org/docs/current/transaction-iso.html">https://www.postgresql.org/docs/current/transaction-iso.html</a></p> <p>Indexing : <a href="https://use-the-index-luke.com/">https://use-the-index-luke.com/</a></p> <p>Query Optimizations : <a href="https://www.geeksforgeeks.org/best-practices-for-sql-query-optimizations/">https://www.geeksforgeeks.org/best-practices-for-sql-query-optimizations/</a></p> <p>Using psycopg2 to Connect to PostgreSQL : <a href="https://www.datacamp.com/tutorial/tutorial-psycopg2-python">https://www.datacamp.com/tutorial/tutorial-psycopg2-python</a></p>	

			<p><b>SQLAlchemy</b>  : <a href="https://auth0.com/blog/sqlalchemy-orm-tutorial-for-python-developers/">https://auth0.com/blog/sqlalchemy-orm-tutorial-for-python-developers/</a></p>	
Non-Relational Databases	2 Days	10 Apr 2025	<p>Intro to NoSQL databases  - <a href="https://www.altexsoft.com/blog/nosql-databases/">https://www.altexsoft.com/blog/nosql-databases/</a></p> <p>MongoDB Basics  - <a href="https://www.sitepoint.com/an-introduction-to-mongodb/">https://www.sitepoint.com/an-introduction-to-mongodb/</a></p> <p>PyMongo  - <a href="https://www.mongodb.com/resources/languages/pymongo-tutorial">https://www.mongodb.com/resources/languages/pymongo-tutorial</a></p> <p>Redis - <a href="https://blog.logrocket.com/guide-to-fully-understanding-redis/">https://blog.logrocket.com/guide-to-fully-understanding-redis/</a></p> <p>DynamoDB  - <a href="https://medium.com/@_amanarora/dynamodb-2efdbcaebabc">https://medium.com/@_amanarora/dynamodb-2efdbcaebabc</a></p> <p>Cassandra  - <a href="https://www.datastax.com/blog/introduction-to-apache-cassandra-the-lamborghini-of-the-nosql-world">https://www.datastax.com/blog/introduction-to-apache-cassandra-the-lamborghini-of-the-nosql-world</a></p>	
CI/CD	3 Days	15 Apr 2025	<p>CI/CD Fundamentals  : <a href="https://about.gitlab.com/blog/2025/01/06/ultimate-guide-to-ci-cd-fundamentals-to-advanced-implementation/">https://about.gitlab.com/blog/2025/01/06/ultimate-guide-to-ci-cd-fundamentals-to-advanced-implementation/</a></p> <p>GitHub Actions for CI/CD  : <a href="https://realpython.com/github-actions-python/">https://realpython.com/github-actions-python/</a></p> <p>CI/CD Deployment Strategies  : <a href="https://blog.mergify.com/real-world-ci-cd-pipeline-examples-devops-success/">https://blog.mergify.com/real-world-ci-cd-pipeline-examples-devops-success/</a></p>	

			<p>Security in CI/CD Pipelines : <a href="https://cycode.com/blog/ci-cd-pipeline-security-best-practices/">https://cycode.com/blog/ci-cd-pipeline-security-best-practices/</a></p> <p>Infrastructure as Code (IaC) with Terraform : <a href="https://developer.hashicorp.com/terraform/tutorials/aws-get-started/infrastructure-as-code">https://developer.hashicorp.com/terraform/tutorials/aws-get-started/infrastructure-as-code</a></p> <p>CI/CD Monitoring &amp; Observability : <a href="https://www.infoq.com/articles/ci-cd-observability/">https://www.infoq.com/articles/ci-cd-observability/</a></p>	
Mini Project: AI-Powered Automated Resume Screener	8 Days	25 Apr 2025	Details are added below	

## AI-Powered Automated Resume Screener

### Objective

Build a system that allows users to **upload resumes**, automatically **extract key details**, and **rank candidates** based on **job relevance** using AI and search technologies.

### Technologies Used

- **FastAPI/Flask** – REST API for resume submission and retrieval
- **PostgreSQL** – Structured storage for jobs and candidate metadata
- **MongoDB** – NoSQL storage for raw resume content
- **OpenSearch** – Full-text search and ranking of resumes
- **Redis** – Caching frequent searches
- **GitHub Actions + Docker** – CI/CD pipeline for deployment
- **Python Libraries** – For parsing, scoring (e.g., spaCy, TF-IDF)

### Features

- Resume upload and parsing (PDF/DOCX)
- AI-powered keyword extraction and matching
- Search resumes by skills, experience, and location
- Rank candidates by relevance to a job description
- Cache frequently queried results with Redis

### **API Operations**

- POST /upload\_resume – Upload and parse a resume
- GET /candidates?skills=Python – Filter candidates by skill
- GET /rank\_candidates?job\_id=123 – Get ranked candidates
- GET /resume/{id} – View parsed resume details

### **Month 4: Generative AI, Machine Learning, Generative AI-Powered Content Generator**

Topics	Days Required	Date	Session Recordings/Resources	Status
Machine Learning	5 Days	16 May 2025	<a href="https://epam-my.sharepoint.com/:w/p/sumit_kumar3/EaKKS9yV_FDuAmDrF2pmLABnLPZKljW2QBfLuF8di0Fcw?e=4TlxpO">https://epam-my.sharepoint.com/:w/p/sumit_kumar3/EaKKS9yV_FDuAmDrF2pmLABnLPZKljW2QBfLuF8di0Fcw?e=4TlxpO</a>	
Generative AI	7 Days	30 May 2025	<a href="https://videoportal.epam.com/playlist/oYVzXAY0">https://videoportal.epam.com/playlist/oYVzXAY0</a>	

Mini Project	8 Days	12 Jun 2025	Details are added below	
--------------	--------	-------------	-------------------------	--

## Hate Speech Detection Assistant

### Overview

This project is designed to help interns build a hate speech detection tool using GenAI techniques. The application will classify user input, retrieve related policies, explain the decision, and suggest a moderation action — all within a simple web interface.

### Objective

Develop a web-based assistant that:

- Accepts user-submitted text
- Classifies it as Hate, Toxic, Offensive, Neutral, or Ambiguous
- Retrieves relevant policy/guideline documents using Hybrid RAG
- Explains the classification using LLMs
- Recommends an action (e.g., remove, warn, flag)

### Core Components

#### 1. **HateSpeechDetectionAgent**

- Uses OpenAI to classify the input
- Returns a label and a short explanation

#### 1. **HybridRetrieverAgent**

- Retrieves the top relevant .txt policy files using sentence-transformer + FAISS

#### 1. **PolicyReasoningAgent**

- Combines classification and retrieved docs to justify the decision via OpenAI

#### 1. **ActionRecommenderAgent**

- Maps the classification to an action (e.g., ban, warn, flag, allow)

#### 1. **ErrorHandlerAgent**

- Handles errors gracefully and informs the user

## **UI Requirements**

- Built in Streamlit
- Input: textarea for content
- Output:
  - Classification label
  - Reason
  - Retrieved policy snippets
  - Recommended action