

# ZERO TO HERO IN RETRIEVAL AUGMENTED GENERATION

Yogesh Haribhau Kulkarni



# Program at a Glance

## Zero to Hero in RAG - 15 Day Training



# From Zero to Hero in RAG

Then from RAG to Riches

**Learning Path:** Python → DSA → System Design → NLP → RAG → Production

## Program Overview

- ▶ Duration: 15 working days (3 weeks)
- ▶ Daily: 8 hours (4 learning + 4 coding)
- ▶ Target: Software engineers with basic programming knowledge

## What You'll Build

- ▶ 15 progressive projects
- ▶ 1 production capstone
- ▶ Complete end-to-end RAG

## Portfolio Highlights:

- ▶ Week 1: 5 Python projects + System design
- ▶ Week 2: NLP pipeline + Embeddings + Vector stores
- ▶ Week 3: Production RAG system with UI

## Weekly Structure:

- ▶ Saturday (Pre-week): Online overview + QnA (2 hours)
- ▶ Monday-Friday: Daily learning + coding (8 hours)
- ▶ Final Saturday: Capstone review + QnA (4 hours)



# Program Architecture

## Phase 1: Foundations (Days 1-5)

- ▶ Python Fundamentals
- ▶ File Handling
- ▶ Async Programming
- ▶ DSA Essentials
- ▶ System Design Basics

## Phase 2: NLP & ML (Days 6-10)

- ▶ Text Processing
- ▶ NLP with spaCy
- ▶ Vectorization
- ▶ Sentiment Analysis
- ▶ Word Embeddings

## Phase 3: RAG (Days 11-15)

- ▶ RAG from Scratch
- ▶ Docstring Parsing
- ▶ LangChain RAG
- ▶ Streamlit UI
- ▶ Capstone Project

### Daily Structure:

- ▶ 09:00-13:00: Learning (tutorials + documentation)
- ▶ 14:00-18:00: Coding (2 project options - pick 1)

### Weekly Milestones:

- ▶ Pre-Week Saturday: Online overview session + QnA (2 hrs)
- ▶ End-Week Assessment: Project review + readiness check
- ▶ Final Saturday (Week 3): All capstone reviews + QnA (4 hrs)



# Day 1: Python Fundamentals & OOP

## Morning Session (4 hrs)

- ▶ Variables & data types
- ▶ Control flow (if, loops)
- ▶ Functions & scope
- ▶ Object-oriented programming
- ▶ Classes & objects
- ▶ Inheritance & polymorphism

## Afternoon: Pick 1 Project

- ▶ **Library Management:** Build a system with books, members, lending with OOP classes and inheritance
- ▶ **Shopping Cart System:** Create product catalog, cart operations, checkout with discount calculations using OOP

## Learning Resources:

- ▶ Python Basics (<https://www.freecodecamp.org/news/learn-python-basics/>)
- ▶ Python OOP (<https://www.freecodecamp.org/news/object-oriented-programming-python/>)
- ▶ Documentation: Python Official Docs (<https://docs.python.org/3/tutorial>)

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working code + README

# Day 2: Python File Handling & Decorators

## Morning Session (4 hrs)

- ▶ File I/O operations
- ▶ Reading/writing text files
- ▶ JSON handling
- ▶ CSV processing
- ▶ Exception handling
- ▶ Decorators basics

## Afternoon: Pick 1 Project

- ▶ **CSV Data Processor:** Read CSV files, filter/transform data, export results with error handling and logging decorator
- ▶ **Config File Manager:** Build JSON/YAML config reader/writer with validation, backup system using context managers

## Learning Resources:

- ▶ File Handling (<https://www.freecodecamp.org/news/file-handling-in-python/>)
- ▶ Decorators (<https://www.freecodecamp.org/news/python-decorators-explained-with-examples/>)

**Difficulty:** Medium |**Time:** 4 hours |**Deliverable:** Working code + README



# Day 3: Python Advanced Deep Dive

## Morning Session (4 hrs)

- ▶ List/dict/set comprehensions
- ▶ Generator expressions
- ▶ Async/await basics
- ▶ Asyncio fundamentals
- ▶ Concurrent operations
- ▶ Async file operations

## Afternoon: Pick 1 Project

- ▶ **Async Web Scraper:** Fetch multiple URLs concurrently, parse HTML, save results using asyncio and aiohttp
- ▶ **Concurrent File Processor:** Process multiple large files in parallel with async I/O and progress tracking

## Learning Resources:

- ▶ Concurrency vs. Parallelism  
(<https://www.freecodecamp.org/news/concurrency-vs-parallelism-whats-the-difference-and-why-should-you-care/>)
- ▶ Getting Started With Async Features in Python  
(<https://realpython.com/python-async-features/>)

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working code + README



# Day 4: Data Structures & Algorithms

## Morning Session (4 hrs)

- ▶ Arrays & lists
- ▶ Hash tables/dictionaries
- ▶ Stacks & queues
- ▶ Binary search
- ▶ Sorting algorithms
- ▶ Big O complexity

## Afternoon: Pick 1 Project

- ▶ **LeetCode Problem Set:** Solve 5 medium problems on arrays, hashmaps, and sorting with optimized solutions
- ▶ **Custom Data Structures:** Implement stack, queue, and hashmap from scratch with test cases and complexity analysis

## Learning Resources:

- ▶ Learn Algorithms and Data Structures in Python (<https://www.freecodecamp.org/news/algorithms-and-data-structures-free-treehouse-course/>)
- ▶ Practice: LeetCode (<https://leetcode.com/explore/learn/>)

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working code + README



# Day 5: System Design Fundamentals

## Morning Session (4 hrs)

- ▶ Requirements analysis
- ▶ Scalability patterns
- ▶ Caching strategies
- ▶ Load balancing
- ▶ Database design
- ▶ API design

## Afternoon: Pick 1 Project

- ▶ **URL Shortener:** Design and implement TinyURL-like service with hashing, database, and caching layer
- ▶ **API Rate Limiter:** Build token bucket/sliding window rate limiter with Redis-like cache and API endpoints

## Learning Resources:

- ▶ System Design (<https://www.freecodecamp.org/news/software-system-design-for-beginners/>)
- ▶ System Design Interview Question  
<https://www.freecodecamp.org/news/systems-design-for-interviews/>
- ▶ Gaurav Sen Playlist  
(<https://www.youtube.com/playlist?list=PLMCXHnjXnTnv06alSjVkgxV-VH6EPyvoX>)

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Design doc + Working code

# Week 1 Milestone Assessment

## Learning Outcomes

- ▶ Python fundamentals mastery
- ▶ File & async operations
- ▶ DSA problem-solving
- ▶ System design thinking

## Success Criteria

- ▶ 5 projects completed
- ▶ Code quality ≥ 80%
- ▶ Core concepts understood
- ▶ Ready for NLP phase

## Assessment Activities:

- ▶ Code review of all 5 projects
- ▶ Technical Q&A on concepts
- ▶ System design explanation
- ▶ Readiness verification for Week 2

## Assessment Rubric (100 pts):

- ▶ Functionality (50 pts) - All features working correctly
- ▶ Code Quality (20 pts) - Clean, documented, readable
- ▶ Design Principles (20 pts) - Architecture and patterns
- ▶ Documentation (10 pts) - README and comments

# Day 6: Text Processing Fundamentals

## Morning Session (4 hrs)

- ▶ Text preprocessing
- ▶ Tokenization
- ▶ Lowercasing & cleaning
- ▶ Stop word removal
- ▶ Stemming & lemmatization
- ▶ Regular expressions

## Afternoon: Pick 1 Project

- ▶ **Text Preprocessing Pipeline:** Build modular pipeline for cleaning, tokenizing, and normalizing text data with configurable steps
- ▶ **Document Cleaner:** Create tool to extract clean text from messy documents with noise removal and format standardization

## Learning Resources:

- ▶ NLP Tutorial (<https://www.freecodecamp.org/news/natural-language-processing-techniques-for-beginners/>)
- ▶ NLP using spaCy (<https://www.freecodecamp.org/news/getting-started-with-nlp-using-spacy/>)
- ▶ Documentation: Spacy Docs (<https://spacy.io/usage/spacy-101>)

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working code + README

# Day 7: NLP with spaCy

## Morning Session (4 hrs)

- ▶ spaCy pipeline basics
- ▶ Part-of-speech tagging
- ▶ Named Entity Recognition
- ▶ Dependency parsing
- ▶ Text similarity
- ▶ Pipeline customization

## Afternoon: Pick 1 Project

- ▶ **Named Entity Extractor:** Build NER system to extract people, organizations, locations from documents with visualization
- ▶ **Text Similarity Engine:** Create document comparison tool using spaCy embeddings with ranking and scoring

## Learning Resources:

- ▶ Natural Language Processing with spaCy  
(<https://www.freecodecamp.org/news/natural-language-processing-with-spacy-python-full-course/>)
- ▶ How to Extract Insights from Text  
(<https://www.freecodecamp.org/news/extract-insights-from-text-using-named-entity-recognition/>)

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working code + README

# Day 8: Text Vectorization

## Morning Session (4 hrs)

- ▶ Bag of Words
- ▶ TF-IDF vectorization
- ▶ Feature extraction
- ▶ Cosine similarity
- ▶ Vector operations
- ▶ Scikit-learn vectorizers

## Afternoon: Pick 1 Project

- ▶ **TF-IDF Search Engine:** Build keyword-based search using TF-IDF with document ranking and relevance scoring
- ▶ **Document Clusterer:** Create text clustering system using vectorization with visualization of document groups

## Learning Resources:

- ▶ Understanding Word Embeddings  
(<https://www.freecodecamp.org/news/understanding-word-embeddings-the-building-blocks-of-nlp-and-gpts/>)
- ▶ How to process textual data using TF-IDF in Python  
(<https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/>)

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working code + README

# Day 9: Sentiment Analysis

## Morning Session (4 hrs)

- ▶ Sentiment analysis basics
- ▶ Feature engineering
- ▶ ML classifiers (Naive Bayes, SVM)
- ▶ Model training & evaluation
- ▶ Cross-validation
- ▶ Model metrics

## Afternoon: Pick 1 Project

- ▶ **Review Sentiment Analyzer:**  
Build classifier for product reviews with training pipeline and accuracy metrics
- ▶ **Tweet Sentiment Dashboard:**  
Create real-time sentiment analysis tool with classification and trend visualization

## Learning Resources:

- ▶ What is Sentiment Analysis? (<https://www.freecodecamp.org/news/what-is-sentiment-analysis-a-complete-guide-to-for-beginners/>)
- ▶ Sentiment Analysis (<https://www.freecodecamp.org/news/sentiment-analysis-with-text-mining/>)

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working code + README



# Day 10: Word Embeddings

## Morning Session (4 hrs)

- ▶ Word embeddings concepts
- ▶ Word2Vec (CBOW, Skip-Gram)
- ▶ Pre-trained embeddings
- ▶ Embedding similarity
- ▶ Sentence embeddings
- ▶ Vector databases basics

## Afternoon: Pick 1 Project

- ▶ **Semantic Search Engine:** Build document search using embeddings with similarity ranking and query expansion
- ▶ **Word Analogy Solver:** Create tool for word analogies and relationships using pre-trained Word2Vec models

## Learning Resources:

- ▶ Tutorial: freeCodeCamp Word Embeddings (<https://www.freecodecamp.org/news/how-to-get-started-with-word2vec-and-then-how-to-make-it-work-d0a2fca9dad3/>)
- ▶ Documentation: Gensim Word2Vec (<https://radimrehurek.com/gensim/models/word2vec.html>)

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working code + README



## Week 2 Milestone Assessment

### Learning Outcomes

- ▶ Text processing mastery
- ▶ NLP pipeline creation
- ▶ ML classification skills
- ▶ Embeddings & similarity

### Success Criteria

- ▶ 5 NLP projects completed
- ▶ ML model  $\geq 75\%$  accuracy
- ▶ Vector operations working
- ▶ Ready for RAG phase

### Assessment Activities:

- ▶ NLP pipeline demonstration
- ▶ Sentiment model evaluation
- ▶ Embedding system walkthrough
- ▶ Component integration check

### Assessment Rubric (100 pts):

- ▶ Functionality (50 pts) - All NLP components working
- ▶ Model Quality (20 pts) - Accuracy and performance
- ▶ Code Quality (20 pts) - Clean and documented
- ▶ Integration (10 pts) - Components work together

# Day 11: RAG from Scratch

## Morning Session (4 hrs)

- ▶ RAG architecture overview
- ▶ Document chunking strategies
- ▶ Vector stores (FAISS/ChromaDB)
- ▶ Retrieval mechanisms
- ▶ Re-ranking techniques
- ▶ Basic RAG pipeline

## Afternoon: Pick 1 Project

- ▶ **Basic RAG System:** Build end-to-end RAG with document loading, chunking, embedding, retrieval, and generation
- ▶ **Multi-Document RAG:** Create RAG system handling multiple documents with metadata filtering and source attribution

## Learning Resources:

- ▶ RAG Handbook (<https://www.freecodecamp.org/news/retrieval-augmented-generation-rag-handbook/>)
- ▶ LangChain RAG  
(<https://www.freecodecamp.org/news/mastering-rag-from-scratch/>)

**Key Pipeline:** Load → Chunk → Embed → Store → Retrieve → Generate  
**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working RAG + README



# Day 12: Document Parsing with Docling

## Morning Session (4 hrs)

- ▶ Docling architecture
- ▶ PDF/DOCX/PPTX parsing
- ▶ Layout analysis
- ▶ Table extraction
- ▶ Image extraction
- ▶ Metadata handling

## Afternoon: Pick 1 Project

- ▶ **Multi-Format Parser:** Build document parser supporting PDF/DOCX/PPTX with text, tables, and image extraction
- ▶ **Structured Data Extractor:** Create tool to extract hierarchical content with headings, sections, and metadata preservation

## Learning Resources:

- ▶ Docling: A Step-by-Step Guide  
(<https://www.datacamp.com/tutorial/docling>)
- ▶ Documentation: Docling Official Docs  
([https://docling-project.github.io/docling/getting\\_started/](https://docling-project.github.io/docling/getting_started/))

**Difficulty:** Medium | **Time:** 4 hours | **Deliverable:** Working parser + README



# Day 13: LangChain RAG Implementation

## Morning Session (4 hrs)

- ▶ LangChain Essentials course
- ▶ Document loaders
- ▶ Text splitters
- ▶ Embeddings integration
- ▶ Vector stores
- ▶ Retrieval chains
- ▶ LCEL expressions

## Afternoon: Pick 1 Project

- ▶ **PDF Q&A System:** Build LangChain-based RAG for PDF documents with chat history and source citations
- ▶ **Conversational RAG:** Create multi-turn RAG chatbot with memory, context awareness, and follow-up questions

## Learning Resources:

- ▶ Course: LangChain Academy Essentials (<https://academy.langchain.com/courses/langchain-essentials-python>)
- ▶ What is Retrieval Augmented Generation (RAG)? (<https://www.datacamp.com/blog/what-is-retrieval-augmented-generation-rag>)

**Difficulty:** Medium |**Time:** 4 hours |**Deliverable:** Working RAG + README



# Day 14: Streamlit UI Development

## Morning Session (4 hrs)

- ▶ Streamlit basics
- ▶ Layout components
- ▶ Interactive widgets
- ▶ File uploads
- ▶ Session state management
- ▶ Chat interfaces

## Afternoon: Pick 1 Project

- ▶ **RAG Chatbot Interface:** Create interactive chat UI with file upload, message history, and streaming responses
- ▶ **Document Q&A Dashboard:** Build multi-page app with document upload, processing status, and query interface

## Learning Resources:

- ▶ Streamlit (Build 12 Data Science Apps with Python and Streamlit)
- ▶ Documentation: Streamlit Official Docs (<https://docs.streamlit.io/>)

**Difficulty:** Medium |**Time:** 4 hours |**Deliverable:** Working UI + README

# Day 15: End-to-End Capstone Project

## Morning Session (4 hrs)

- ▶ Architecture planning
- ▶ Component integration
- ▶ Testing strategies
- ▶ Performance optimization
- ▶ Error handling
- ▶ Deployment preparation

## Pick 1 Full-Stack Project

- ▶ **Document Intelligence System:**  
Multi-format parser + RAG + chat UI with source attribution and metadata
- ▶ **Enterprise Knowledge Base:**  
Multi-document RAG with advanced search, filters, and analytics dashboard

## Integration Requirements:

- ▶ Docling: Multi-format document parsing (PDF/DOCX/PPTX)
- ▶ LangChain: Complete RAG orchestration with chains
- ▶ Streamlit: Professional interactive web interface
- ▶ All components seamlessly integrated

**Capstone Evaluation (100 pts):** Functionality (40) | Code Quality (30)  
| Architecture (20) | Documentation (10)  
**Minimum Passing Score:** 70/100



## Week 3 Milestone & Final Assessment

### Learning Outcomes

- ▶ Complete RAG systems
- ▶ Document processing mastery
- ▶ LLM orchestration
- ▶ Production-ready apps

### Success Criteria

- ▶ Capstone fully functional
- ▶ All components integrated
- ▶ Code well-documented
- ▶ Production-ready quality

### Production Readiness Checklist:

- ▶ Error handling & validation implemented
- ▶ Performance optimized (chunking, caching)
- ▶ Comprehensive documentation
- ▶ User-friendly interface
- ▶ Code quality >80%

### Final Assessment Activities:

- ▶ Individual capstone project demonstrations (20 mins each)
- ▶ Architecture & design explanation
- ▶ Live code walkthrough
- ▶ Q&A on technical decisions
- ▶ Future enhancements discussion

# Final Saturday: Capstone Reviews & QnA

## Session Structure (4 hours):

- ▶ 10:00-12:30: Individual capstone presentations (20 mins each)
- ▶ 12:30-13:00: Break
- ▶ 13:00-14:00: Open QnA & technical discussions

## Presentation Format (20 mins):

- ▶ Project demo (8 mins) - Live demonstration
- ▶ Architecture overview (5 mins) - System design & components
- ▶ Code walkthrough (5 mins) - Key implementation details
- ▶ Q&A (2 mins) - Questions from instructors

## Evaluation Criteria:

- ▶ **Functionality (40 pts):** All features working, error-free
- ▶ **Code Quality (30 pts):** Clean, modular, documented
- ▶ **Architecture (20 pts):** Sound design, scalable structure
- ▶ **Documentation (10 pts):** Clear README, setup instructions

## QnA Topics:

- ▶ Technical challenges faced & solutions
- ▶ RAG optimization techniques

# Congratulations! From RAG to Riches

## Technical Skills Acquired

- ▶ Advanced Python programming
- ▶ NLP & text processing
- ▶ RAG architecture & implementation
- ▶ Full-stack AI development
- ▶ Production deployment

## Career Pathways

- ▶ AI/ML Engineer
- ▶ RAG Solutions Architect
- ▶ LLM Application Developer
- ▶ AI Product Engineer
- ▶ NLP Engineer

## Recommended Certifications:

- ▶ LangChain Academy Certification ([academy.langchain.com](https://academy.langchain.com))
- ▶ Azure AI Fundamentals (AI-900)
- ▶ AWS Machine Learning Specialty

## Continuous Learning Resources:

- ▶ LangChain Academy ([academy.langchain.com](https://academy.langchain.com))
- ▶ DeepLearning.AI Short Courses ([deeplearning.ai](https://deeplearning.ai))
- ▶ Hugging Face NLP Course ([huggingface.co/learn/nlp-course](https://huggingface.co/learn/nlp-course))
- ▶ LangChain Documentation ([python.langchain.com](https://python.langchain.com))
- ▶ Explore open-source RAG projects on GitHub



# Weekly Saturday Sessions Schedule

## Pre-Week Saturday Sessions (2 hours each):

### Saturday Week 0 (Before Day 1):

- ▶ Program overview & expectations
- ▶ Week 1 preview: Python fundamentals to system design
- ▶ Development environment setup
- ▶ QnA session

### Saturday Week 1 (Before Day 6):

- ▶ Week 1 recap & learnings
- ▶ Week 2 preview: NLP & ML foundations
- ▶ Introduction to text processing and embeddings
- ▶ QnA session

### Saturday Week 2 (Before Day 11):

- ▶ Week 2 recap & portfolio review
- ▶ Week 3 preview: RAG systems & production
- ▶ RAG architecture deep dive
- ▶ QnA session

### Final Saturday (After Day 15):