# Python Code Assistant - RAG + LangGraph System

## Introduction

This document outlines the architecture and functionality of the Python Code Assistant application, which combines Retrieval-Augmented Generation (RAG) and LangGraph for intelligent code generation, debugging, explanation, and optimization of Python code.

## Part 1: Smarter Intent Routing with LangGraph

The assistant uses a LangGraph-powered state machine to route user prompts based on intent. Instead of using keywords, the system uses an LLM-based intent classifier to categorize user prompts into tasks such as generate, explain, debug, optimize, or exit.

Key Components:

* • LLMIntentClassifier: Classifies the intent using a custom prompt and LLM response.
* • LangGraphCodeAssistant: A LangGraph state machine that conditionally routes the task.

## Part 2: Gradio-Based Chat Interface

The assistant is accessible via a Gradio interface with a chatbot UI. The assistant class processes the input, while LangGraph manages routing and generation. Debug and evaluation tools are included to assess RAG performance.

Chatbot Functionality:

* • Send and receive Python-related queries in natural language.
* • Evaluate MBPP tasks and debug examples with a button click.

## Part 3: Retrieval-Augmented Generation (RAG) on MBPP + HumanEval

The assistant uses LangChain's vector store and retriever to enhance prompt context using examples from HumanEval and MBPP datasets. FAISS is used for efficient document embedding and retrieval.

Key Features:

* • Uses `HuggingFaceEmbeddings` with MiniLM model.
* • Supports MMR-based retrieval and document splitting.
* • Retrieval quality metrics and diversity tracking are included.

## Evaluation

An evaluation pipeline tests the assistant on MBPP test samples. It generates code, runs provided tests, and reports pass/fail results and retrieval diversity.

Metrics Tracked:

* • Pass rate on test cases
* • Average retrieved examples
* • Source diversity and context length

## Conclusion

This assistant integrates cutting-edge techniques in retrieval, generation, and structured workflow management to provide an intelligent Python coding tool. The modular design ensures future extensibility for datasets, tools, and interfaces.