COSC2669 – WIL Project: Milestone 1
Project: Guide to Python Documentation (RAG)

Group Short Report (≤ 2 pages, excluding references & appendices)

Date: 21 Aug 2025 | Course: COSC2669 Case Studies in Data Science | Milestone: 1

1. Group ID & Project Aim Statement

Group ID (Canvas): WIL Project 74

Group/Project Name: PyDocs Guide (RAG)

Aim: As a Python learner/practitioner, I want to ask questions in natural language and receive simple, accurate explanations grounded in the official Python documentation, with runnable code examples and links back to the relevant docs.

Primary User Story: "As a Python user, I want to query the docs in natural language so that I can quickly understand the right concept and see a minimal working example."

2. Group Members & Roles

Student ID	Full Name	Skills/Strengths
4124226	Harithaa Kannan	I have intermediate-level Python skills with experience in developing end-to-end machine learning pipelines. My strengths are in preparing and analyzing data, applying retrieval techniques for RAG, and contributing effectively in team-based projects.
4057208	Harshini Chandra Bose	Skilled in Python, data analysis, and machine learning with knowledge of SQL and visualization. Experienced in frontend design, UX, project management, and collaborative documentation.
4159932	Devanshu Nalavade	I am at intermediate level in python and have prior experience developing data pipelines, and various machine learning models.
4115427	Rithika Janardhan Kothur	I am at beginner level in Python and R programming, data analysis, and dataset cleaning, with knowledge of SQL and basic visualization; beginner with Git/GitHub.
4075975	Sachin SureshKumar	I am at an intermediate level in Python with experience building data pipelines and machine learning models. I have worked on data preprocessing, NLP tasks, and integrating retrieval techniques for RAG systems. I am also skilled in evaluation, documentation, and effective teamwork.

3. Progress So Far

- Finalised project topic: a RAG-based assistant that answers Python questions using the official documentation as the knowledge base.
- Reviewed RAG concepts and baseline via Walert (paper, website, and GitHub) and noted evaluation ideas (e.g., % unanswered, groundedness).
- Set up shared repo/board (e.g., GitHub + Kanban) and aligned roles and responsibilities for the team.
- Drafted knowledge base plan: start with a small subset of Python docs (tutorial + library reference pages) to keep scope manageable.
- Experimented locally with retrieval pipeline (embeddings + chunking strategy, e.g., by section/heading) and verified basic retrieval quality.
- Outlined evaluation framework covering effectiveness, faithfulness to sources, and latency; prepared a small gold set of Q&A pairs for testing.

4. Plan for the Next Three Weeks

- Week 1: Curate ~20–40 key Python doc pages; implement ingestion, chunking-by-heading, and metadata (URL anchors) for precise citations.
- Week 1: Reproduce the Walert evaluation metric for % unanswered and define acceptance thresholds for our domain.
- Week 2: Build the RAG chain (retriever + reader) with source attribution; integrate a simple web UI (single-page prototype).
- Week 2: Add guardrails for hallucinations (e.g., answer only from retrieved spans; include 'I don't know' fallback).
- Week 3: Run the evaluation suite (effectiveness, faithfulness, latency); iterate on chunk size, re-ranking, and prompt tuning.
- Week 3: Prepare demo, documentation, and slides; compile sprint evidence (git commits, board screenshots) and finalize submission.

5. Evaluation Framework

We will report:

- Effectiveness: % unanswered; exact-match/semantic similarity of answers to ground-truth; top-k retrieval precision@k.
- Faithfulness: percentage of answers fully supported by cited spans; manual audit of quoted citations; no unsupported claims.
- Usefulness: short user study (team-internal) rating clarity and code-example quality on a 1–5 Likert scale.
- Efficiency: median latency (p95) for end-to-end answers on the gold set.

6. Declaration of Generative AI Use

This report and planning text were prepared with assistance from an AI writing assistant for wording and formatting. All project design choices, evaluation plans, and technical content reflect our team's decisions.

7. References

- Walert: 'Your FAQ Open Day Buddy' (paper, project, GitHub).
- LangChain RAG Tutorial.
- Official Python Documentation (https://docs.python.org/).

8. Appendix: Retrieval Evaluation (NDCG)

Setup: Python 3.8.20 • nmslib 2.1.1 • faiss-cpu 1.7.4

Metric: NDCG at cutoffs k = 1, 3, 5 (Järvelin & Kekäläinen formulation).

	Model	NDCG@1	NDCG@3 / NDCG@5
a	walert_intent	0.0833	0.0391 / 0.0391
b	walert.rag.bm25	0.1667	0.2566 / 0.3291
c	walert.rag.dense.faiss	0.2500	0.2380 / 0.2380

- Dense FAISS has the best top-1 precision (NDCG@1 = 0.25).
- BM25 provides stronger broader-ranking quality (best @3/@5).
- Implication: a hybrid (BM25 + dense) or re-ranking step is likely to improve both @1 and @5.