🧠 Agentic Python Blog Writer Assignment

For: Python Developer Role (Smart, Optimized, API-Oriented)
Objective: Build an autonomous content generation agent that mimics the role of a junior blog writer + SEO optimizer.

🎯 Overview

You are tasked with building a **Python-based AI Blog Writing Agent**. This autonomous system should take a **topic as input**, conduct **contextual research**, generate **long-form SEO blogs**, enrich them with relevant content, and finally export structured blog files and metadata.

The agent should behave like a **junior content creator** capable of planning, writing, editing, and optimizing — all autonomously and smartly.

🧩 Core Agent Responsibilities

1. Understand the Topic

- Input example: `"How Python is used in AI"`
- Your agent should:
  - Break the topic into subtopics or themes
  - Decide on tone and depth of content (e.g., beginner-friendly, technical)
  - Optionally allow user to specify tone (educational, formal, creative, etc.)

---

2. Conduct Research (Context Agent)

- Use public APIs to gather background info, news references, and SEO context.
- Recommended:
    - **NewsData.io** to fetch trending news or recent updates related to the topic
    - **Datamuse API** to find semantic keyword variations for SEO
    - **Quotable.io** to fetch relevant quotes to include in the article

---

3. Generate Content (Writing Agent)

- Use **Google Gemini API (Pro)** to:
    - Draft an outline (H2-level headings)
    - Write an engaging introduction (~100–150 words)
    - Fill in each section with ~200–300 words per subheading
    - Write a strong conclusion with a call-to-action
- Follow a clear structure with Markdown formatting (Headings, Bullet Points, etc.)

---

4. SEO Optimization (SEO Agent)

Generate and export the following:

- Title
- Meta-description (max 160 characters)
- Tags/keywords
- Estimated reading time
- Suggested URL slug

Use both Gemini and Datamuse to help identify high-impact keywords.

---

5. Export and Summarize (Execution Agent)

Your agent must:

- Export the final blog in `.md` (Markdown) format
- Export structured metadata (title, description, keywords, slug, etc.) as `.json`
- Display a CLI summary once the process is completed

🧪 APIs You Can Use (Free Tiers)

Purpose	Tool	Free Access
Content Generation	Google Gemini API	Free via GCP
Research/Context	NewsData.io	200 requests/day
SEO Keywords	Datamuse API	Unlimited
Quotes	Quotable API	Unlimited

⚙️ Smart Work Expectations

Your implementation will be evaluated not only on the output but also on **efficiency, intelligence, and smart engineering**.

Please demonstrate:

- Modular design (separate agents for writing, SEO, export, etc.)
- Retry logic for API failure or rate limits

- Use of `asyncio` for concurrent API requests
- Use of caching/memoization (e.g., `functools.lru_cache`)
- Clean CLI interface with input arguments

✅ Evaluation Criteria

Criterion	Description
🔍 Content Quality	Structure, clarity, SEO alignment, formatting
🧠 API Integration	Intelligent, purposeful use of APIs
💡 Smart Engineering	Async, caching, clean logic
⚙️ Reusability	Modularity, maintainability, and documentation
📤 Output & UX	Quality of markdown + metadata export, CLI logs

---

🔧 Optional Bonus

These are not mandatory but will earn bonus points:

- Add a `--tone` flag to change writing style
- Show blog readability score (e.g., Flesch-Kincaid)
- Add support for multiple topics in batch mode
- Create a simple web interface with Streamlit

📌 Submission Instructions

- Upload a GitHub repo or zipped folder of your project
- Include a `README.md` with:
  - How to run the script
  - Setup instructions (e.g., API keys in `.env`)
  - Example blog outputs
- (Optional) Share a Loom/YouTube video demo explaining your approach (2 mins max)

💬 Final Note

We’re **not looking for brute force** — we’re looking for **developers who build smart, agent-like systems**. Prioritize **autonomy, modularity**, and **optimization**.

Good luck, we’re excited to see what your blog agent can write!