Lecture 19: Real-World Use Case – Autonomous Writing Agent

🎯 Learning Objectives

By the end of this lecture, you should be able to:

- Design an autonomous agent that generates structured written content.
- Combine planning, research, writing, and revision in a single loop.
- Integrate tools for fact-checking, formatting, and outlining.
- Build a pipeline that outputs a publish-ready blog post or report.

🧩 Key Concepts

Autonomous Writing Workflow

1. **Goal Decomposition**: Break the writing task into outline or section-based goals.
2. **Research Phase**: Use search tools or document memory to gather facts.
3. **Drafting Phase**: Generate section drafts based on the outline and research.
4. **Review Phase**: Revise or fact-check using secondary agents or tools.
5. **Final Assembly**: Concatenate content and apply formatting.

Key Features

- Dynamic prompting and context building
- Long-term memory for topic tracking
- Modular, role-based agent loop (Planner → Researcher → Writer → Editor)

🛠️ Required Tools/Libraries

- Python
- OpenAI API (or equivalent LLM)
- LangChain
- FAISS or ChromaDB for retrieval
- Markdown/HTML formatter (optional)

🔬 Hands-on Exercise: Build a Multi-Phase Writing Agent

Goal: Create an agent that writes a blog post titled "*The Future of Agentic AI*".

Step 1: Define task and outline

```
task = "Write a 1000-word blog post on 'The Future of Agentic AI'"
outline = [
    "Introduction",
    "What is Agentic AI?",
    "Recent Advances",
    "Use Cases and Applications",
    "Ethical and Technical Challenges",
    "Conclusion"
]
```

Step 2: Create section-wise prompts

```
for section in outline:
    prompt = f"Write the '{section}' section of the blog post: {task}"
    response = llm.generate(prompt)
    save_to_document(section, response)
```

Step 3: Add review logic

```
def review_section(text):
    prompt = f"Review the following for clarity, accuracy, and tone:\n\n{text}"
    return llm.generate(prompt)
```

Step 4: Compile final post

```
blog_post = assemble_sections_from_file()
format_as_markdown(blog_post)
```

Bonus:

- Add citations using a search tool.
- Use CrewAI to split Planner, Researcher, Writer, and Editor into separate agents.
- Deploy as a Streamlit or Flask app for interactive blog generation.
